INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAMPINA GRANDE
CURSO DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

QUESTBOOK: APLICAÇÃO DE PROCESSAMENTO DE LINGUAGEM NATURAL PARA RECUPERAÇÃO CONTEXTUAL DE QUESTÕES

ÁLISSON BRENER DA SILVA
CAIO LÍVIO LEITE MUNIZ DANTAS
VINÍCIUS GONZAGA CAVALCANTE RODRIGUES

ORIENTADOR: PROF. DR. ELMANO RAMALHO CAVALCANTI

CAMPINA GRANDE – PB
MAIO DE 2026

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

CAMPUS CAMPINA GRANDE

CURSO DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

QuestBook: Aplicação de Processamento de Linguagem Natural para Recuperação Contextual de Questões

Álisson Brener da Silva
Caio Lívio Leite Muniz Dantas
Vinícius Gonzaga Cavalcante Rodrigues

Orientador: Prof. Dr. Elmano Ramalho Cavalcanti

Artigo apresentado à Coordenação do Curso de Bacharelado em Engenharia de Computação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Campina Grande – PB
Maio de 2026

IFPB - Dados Internacionais de Catalogação na Publicação (CIP) – Thiago Cabral CRB15 - 628

S586q

SILVA, Álisson Brener da

Questbook: aplicação de processamento de linguagem natural para recuperação contextual de questões - campina grande. Alisson Brener da Silva; Caio lívio Leite Muniz Dantas; Vinícius Gonzaga Cavalcante Rodrigues . 2026. 45f..

TCC (PDF)

Orientador: Elmano Ramalho Cavalcanti

.

1. Representação Numérica 2. Recuperação . 3. NLP 4. Embeddings . I. Titulo

CDU: 004.7

Ficha catalográfica elaborada pelo Departamento de Bibliotecas DBIBLIO/IFPB/Reitoria

Agradecimentos

Primeiramente, agradecemos a Deus por nos conceder saúde, força e determinação ao longo de toda essa jornada, permitindo que superássemos os desafios e chegássemos à conclusão deste curso.

Aos nossos familiares, que sempre estiveram ao nosso lado. O apoio, o incentivo e a compreensão de vocês foram o alicerce para que pudéssemos focar em nossa trajetória acadêmica e alcançar este objetivo.

Ao nosso orientador, Prof. Dr. Elmano Ramalho Cavalcanti, por guiar nossa equipe com dedicação e paciência. Suas valiosas contribuições e seu apoio constante foram indispensáveis para o sucesso deste trabalho.

Aos professores do curso de Bacharelado em Engenharia de Computação e aos colaboradores do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), pelo papel essencial em nossa formação profissional.

Agradecemos também uns aos outros. A conclusão deste projeto é fruto de muito companheirismo, respeito mútuo, divisão de responsabilidades e perseverança como equipe.

Aos amigos e colegas de turma, pelos momentos compartilhados, e a todos que, direta ou indiretamente, fizeram parte desta conquista.

QuestBook: Aplicação de Processamento de Linguagem Natural para Recuperação Contextual de Questões

Álisson B. da Silva[1], Caio Lívio L. M. Dantas[2], Vinícius Gonzaga C. Rodrigues[3], Elmano R. Cavalcanti[1]

[1]Instituto Federal da Paraíba (IFPB) – Campina Grande, PB – Brazil

{alisson.brener,dantas.caio,vinicius.gonzaga}@academico.ifpb.edu.br, elmano.cavalcanti@ifpb.edu.br

Abstract. *Preparing for competitive exams requires both theoretical study and practice through solving questions. However, students often struggle to find questions related to the content studied in books or PDF materials. Most existing platforms rely on keyword searches or metadata filters, which limits the retrieval of conceptually related questions. This work proposes QuestBook, a system that performs semantic retrieval of questions based on chapters extracted from PDF study materials. The approach uses Natural Language Processing techniques and embeddings to identify relationships between textual content and a structured question database.*

Keywords: *Natural Language Processing, Semantic Retrieval, Embeddings.*

Resumo. *A preparação para concursos e exames acadêmicos exige estudo teórico aliado à prática com resolução de questões. Entretanto, estudantes frequentemente encontram dificuldades para localizar exercícios relacionados ao conteúdo estudado em livros ou materiais em PDF. As plataformas atuais utilizam principalmente buscas por palavras-chave, o que limita a identificação de questões conceitualmente semelhantes. Este trabalho propõe o QuestBook, um sistema capaz de realizar a recuperação semântica de questões a partir de capítulos de materiais didáticos em PDF utilizando técnicas de Processamento de Linguagem Natural e embeddings.*

Palavras-chave: *Processamento de Linguagem Natural, Recuperação Semântica, Embeddings.*

1. Introdução

A preparação para concursos públicos, certificações profissionais e avaliações acadêmicas envolve o estudo teórico aliado à resolução de exercícios. A prática com questões contribui para a consolidação do conhecimento e auxilia na compreensão da forma como os conteúdos são cobrados em avaliações. Entretanto, apesar da ampla disponibilidade de materiais didáticos digitais, ainda existem dificuldades em relacionar diretamente o conteúdo estudado com questões relevantes para prática.

As plataformas tradicionais de bancos de questões utilizam predominantemente

mecanismos de busca baseados em palavras-chave e filtros estruturados, como disciplina, banca organizadora e ano de aplicação. Embora essas abordagens permitam recuperar questões de forma rápida, elas apresentam limitações na identificação de conteúdos semanticamente relacionados. Em muitos casos, questões relevantes deixam de ser recuperadas devido ao uso de terminologias diferentes para representar um mesmo conceito.

Os avanços em Processamento de Linguagem Natural (PLN) ampliaram a capacidade de sistemas computacionais realizarem tarefas de recuperação de informação e análise semântica [Jurafsky e Martin 2023]. Nesse contexto, modelos de representação vetorial conhecidos como *embeddings* passaram a ser amplamente utilizados para representar relações semânticas entre palavras, sentenças e documentos. Esses modelos permitem identificar similaridade contextual mesmo na ausência de correspondência lexical direta.

A evolução dessas técnicas foi impulsionada pela arquitetura Transformer, proposta por [Ashish Vaswani et al. 2017], que introduziu mecanismos de atenção capazes de capturar dependências contextuais de forma mais eficiente que abordagens anteriores. A partir dessa arquitetura surgiram modelos como o BERT (*Bidirectional Encoder Representations from Transformers*), proposto por [Jacob Devlin et al. 2019], amplamente utilizado em tarefas de recuperação de informação, classificação textual e similaridade semântica.

Diante desse cenário, este trabalho propõe o desenvolvimento do QuestBook, um sistema para recuperação semântica de questões a partir de materiais didáticos em formato PDF. A proposta utiliza *embeddings* textuais e busca vetorial para automatizar a associação entre conteúdos estudados e questões semanticamente relacionadas, reduzindo o esforço manual necessário para localizar exercícios relevantes.

O sistema foi desenvolvido como uma Prova de Conceito (*Proof of Concept* — PoC) composta por um pipeline de processamento textual, geração de representações vetoriais e recuperação semântica de questões. A solução permite o envio de documentos PDF, a extração de conteúdo textual e a recuperação contextual de questões em uma base estruturada. Adicionalmente, o sistema disponibiliza uma interface web para interação do usuário e um módulo de curadoria para validação das questões recuperadas.

2. Fundamentação Teórica

O Processamento de Linguagem Natural (PLN) e a Recuperação de Informação constituem áreas centrais para o desenvolvimento de sistemas capazes de interpretar, processar e recuperar conteúdo textual. A evolução recente dessas áreas foi impulsionada por técnicas baseadas em aprendizado profundo, que ampliaram a capacidade de representação semântica e recuperação contextual de informações [Jurafsky e Martin 2020; Goldberg 2017].

Nesse contexto, modelos de representação vetorial passaram a desempenhar papel fundamental na transformação de textos em estruturas numéricas capazes de

representar relações semânticas entre documentos. Essas abordagens são amplamente utilizadas em sistemas modernos de busca e recomendação, reduzindo limitações de métodos baseados exclusivamente em correspondência lexical [Manning et al. 2008; Reimers e Gurevych 2019].

A aplicação dessas técnicas possibilita o desenvolvimento de sistemas capazes de recuperar conteúdos semanticamente relacionados em grandes volumes de dados textuais. As próximas seções apresentam os principais conceitos relacionados à representação textual, modelos contextuais, recuperação semântica e arquitetura de sistemas inteligentes.

2.1 Representação de Texto Tradicional

As primeiras técnicas de representação textual, como *Bag-of-Words* e TF-IDF, baseavam-se na frequência de ocorrência dos termos em documentos, sem considerar contexto semântico ou ordem das palavras [Manning et al. 2008]. Embora amplamente utilizadas em sistemas clássicos de recuperação de informação, essas abordagens apresentam limitações na interpretação de relações semânticas, sinonímia e polissemia.

Como consequência, documentos semanticamente semelhantes podem não ser relacionados quando utilizam vocabulários distintos. Para reduzir esse problema, surgiram modelos distribuídos de representação textual, como Word2Vec e GloVe [Mikolov et al. 2013; Pennington et al. 2014], que representam palavras por meio de vetores densos capazes de capturar relações semânticas.

Entretanto, esses modelos ainda possuem limitações relevantes, pois cada palavra recebe uma representação vetorial fixa independentemente do contexto em que aparece. Essa característica dificulta a interpretação adequada de ambiguidades linguísticas presentes em diferentes sentenças.

2.2 Modelos Contextuais: Transformers, BERT e Sentence-BERT

Um avanço significativo em PLN ocorreu com a introdução da arquitetura Transformer por [Ashish Vaswani et al. 2017]. O modelo utiliza mecanismos de atenção (*attention mechanisms*) capazes de representar dependências contextuais entre palavras ao longo de uma sequência textual.

A arquitetura Transformer possibilitou ganhos expressivos em desempenho e paralelização computacional quando comparada a arquiteturas recorrentes tradicionais. A partir dessa arquitetura surgiu o BERT (*Bidirectional Encoder Representations from Transformers*), proposto por [Jacob Devlin et al. 2019], baseado em representações bidirecionais contextualizadas.

No BERT, o significado de cada palavra é calculado considerando simultaneamente os termos anteriores e posteriores da sequência textual. Essa abordagem proporcionou avanços relevantes em tarefas como recuperação de informação, classificação textual e análise de similaridade semântica.

Posteriormente, modelos voltados à representação vetorial de sentenças

completas passaram a ser utilizados em aplicações de busca semântica. Entre eles, destaca-se o Sentence-BERT [Reimers e Gurevych 2019], capaz de gerar *embeddings* semânticos de documentos e sentenças inteiras, permitindo comparações textuais por meio de métricas vetoriais, como similaridade cosseno.

2.3 Modelos de Grande Escala e Engenharia de Prompt

Os Modelos de Linguagem de Grande Escala (*Large Language Models* — LLMs) ampliaram significativamente as aplicações de PLN. Esses modelos são treinados sobre grandes volumes de dados textuais e apresentam elevada capacidade de generalização para diferentes tarefas linguísticas [Brown et al. 2020; Bommasani et al. 2021].

Entre os avanços proporcionados pelos LLMs destaca-se o aprendizado *few-shot*, no qual o modelo executa tarefas a partir de poucos exemplos fornecidos no *prompt*. Entretanto, esses modelos também apresentam desafios relacionados à confiabilidade das respostas, principalmente devido ao fenômeno de alucinação, caracterizado pela geração de informações semanticamente plausíveis, porém incorretas.

Com o objetivo de reduzir esse problema, diferentes técnicas de *prompt engineering* vêm sendo propostas. Entre elas destaca-se o método *Chain-of-Thought*, apresentado por [Wei et al. 2022], que incentiva o modelo a produzir etapas intermediárias de raciocínio antes da resposta final.

No contexto deste trabalho, essas abordagens são utilizadas para estruturar a interpretação das consultas dos usuários e auxiliar a associação entre conteúdos extraídos de documentos PDF e questões recuperadas semanticamente.

2.4 Recuperação de Informação e Busca Semântica

Sistemas tradicionais de Recuperação de Informação utilizam predominantemente correspondência lexical entre os termos da consulta e os documentos armazenados. Em contrapartida, abordagens modernas exploram recuperação semântica baseada em representações vetoriais de alta dimensionalidade geradas por *embeddings* [Manning et al. 2008; Reimers e Gurevych 2019].

Nesse modelo, a similaridade entre documentos pode ser calculada por métricas matemáticas, como similaridade cosseno, permitindo identificar conteúdos semanticamente relacionados mesmo sem correspondência lexical direta.

A similaridade cosseno é definida por:

$$\text{similaridade}(A, B) = \frac{A \cdot B}{|A||B|}$$

Em que:

- (A) representa o vetor correspondente ao primeiro documento;
- (B) representa o vetor correspondente ao segundo documento;

- (A \cdot B) corresponde ao produto escalar entre os vetores;
- (|A|) e (|B|) representam as magnitudes dos vetores.

O processo de recuperação semântica inicia-se pela extração textual do documento PDF, seguida pela geração de *embeddings* e armazenamento em um banco vetorial. Posteriormente, os vetores são comparados utilizando métricas de similaridade, permitindo recuperar questões semanticamente relacionadas ao conteúdo estudado [Reimers e Gurevych 2019; Zhang et al. 2023].

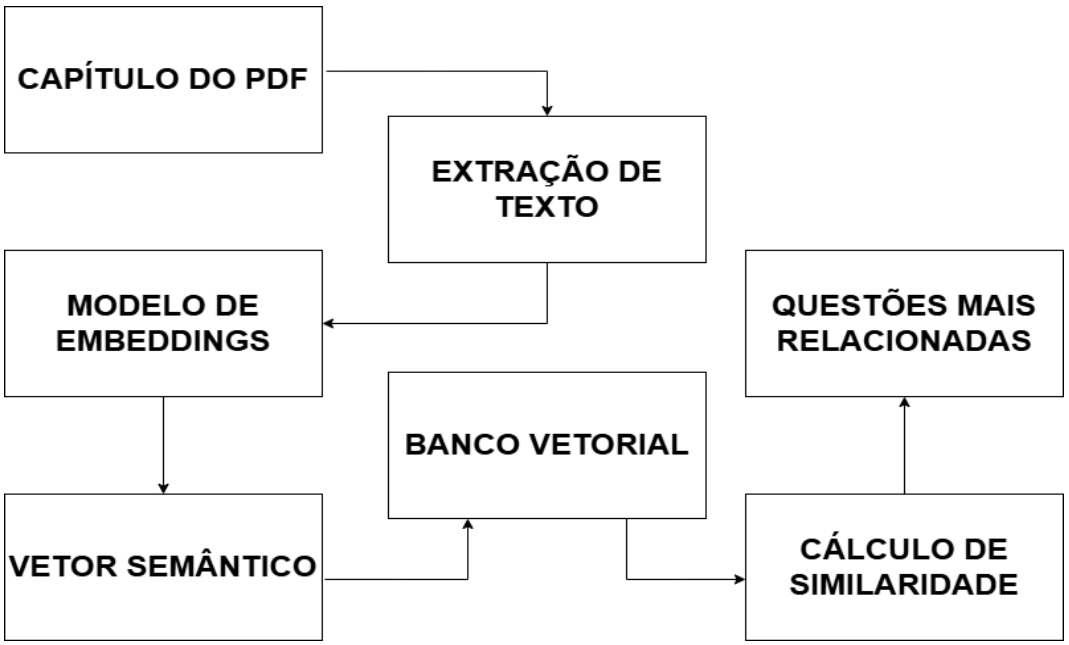

Figura 1 – Fluxo simplificado de recuperação semântica utilizando *embeddings*.

A Tabela 1 apresenta uma comparação entre abordagens tradicionais de recuperação textual e métodos de busca semântica.

Tabela 1 – Comparação entre métodos tradicionais de busca e recuperação semântica.

Método	Característica principal	Limitação
Busca por palavras-chave	Correspondência direta entre termos da consulta e do	Não considera contexto semântico

	documento	
TF-IDF	Representação baseada na frequência de termos	Sensível a variações de vocabulário
Embeddings semânticos	Representação vetorial contextual do texto	Maior custo computacional
Recuperação semântica	Busca baseada em similaridade entre vetores	Requer modelos treinados e banco vetorial

2.5 Bancos Vetoriais

A recuperação semântica depende do uso de bancos vetoriais responsáveis pelo armazenamento de *embeddings* e execução eficiente de consultas por similaridade. Esses sistemas utilizam algoritmos de *Approximate Nearest Neighbors* (ANN), projetados para reduzir o custo computacional da busca em espaços vetoriais de alta dimensionalidade [Zhang et al. 2023].

Bancos vetoriais permitem localizar rapidamente vetores semanticamente próximos mesmo em grandes volumes de dados. Para isso, empregam técnicas específicas de indexação e otimização voltadas à busca aproximada.

Dessa forma, bancos vetoriais tornaram-se componentes centrais em aplicações modernas de recuperação semântica, recomendação de conteúdo e sistemas inteligentes baseados em linguagem natural [Johnson et al. 2019].

2.6 Arquitetura de Software para Sistemas Inteligentes

Sob a perspectiva da Engenharia de Software, sistemas inteligentes demandam arquiteturas capazes de integrar componentes heterogêneos de forma modular e escalável. Arquiteturas modernas geralmente adotam separação entre frontend, backend e camada de persistência, favorecendo manutenção, extensibilidade e desacoplamento [Bass et al. 2013].

Nesse modelo, o backend é responsável pela lógica de negócio, processamento semântico e integração com serviços externos, enquanto o frontend concentra os mecanismos de interação com o usuário.

Além disso, aplicações modernas frequentemente utilizam persistência híbrida, combinando bancos relacionais para dados estruturados e bancos vetoriais para armazenamento de *embeddings*. Essa abordagem permite tratar eficientemente diferentes tipos de dados dentro da mesma arquitetura.

3. Modelagem do sistema

O QuestBook foi projetado utilizando uma arquitetura modular em camadas, visando separação de responsabilidades, manutenibilidade e escalabilidade. A arquitetura é composta por três camadas principais: frontend, backend e persistência.

O frontend é responsável pela interação com os usuários, incluindo envio de documentos, visualização de questões recuperadas e validação das sugestões. A interface foi desenvolvida utilizando tecnologias web modernas voltadas à construção de aplicações responsivas.

O backend concentra a lógica de negócio, o processamento semântico e a integração com os modelos de PLN. A implementação foi realizada em Python devido à ampla adoção da linguagem em aplicações de Inteligência Artificial e à disponibilidade de bibliotecas especializadas para processamento textual e recuperação semântica.

A camada de persistência utiliza abordagem híbrida, combinando banco relacional para armazenamento de dados estruturados e banco vetorial para armazenamento de *embeddings*. Essa arquitetura possibilita consultas por similaridade semântica com baixo custo computacional.

3.1. Diagrama de caso de uso

A Figura 2 apresenta o diagrama de casos de uso do sistema QuestBook. O modelo identifica dois perfis principais de usuário: estudante e curador.

O estudante interage com funcionalidades relacionadas ao envio de documentos e recuperação de questões semanticamente relacionadas ao conteúdo estudado. O curador é responsável pela validação das questões recuperadas, atuando como mecanismo adicional de controle de qualidade.

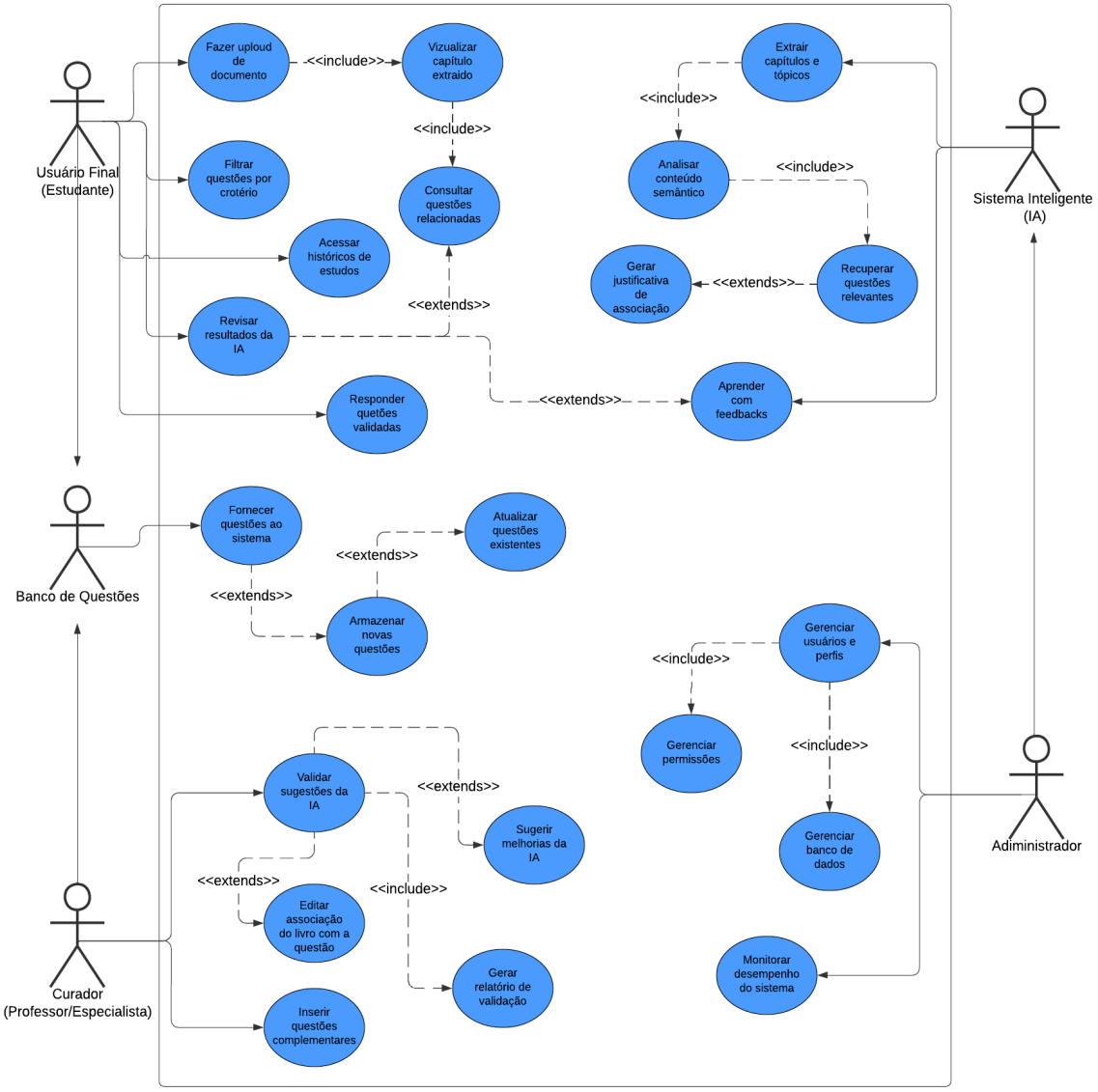

Figura 2 – Diagrama de casos de uso do sistema QuestBook.

3.2. Diagrama de blocos

A Figura 3 apresenta a organização dos principais componentes do sistema e o fluxo de processamento adotado pela aplicação.

O fluxo inicia-se pelo envio de um documento PDF pelo usuário. Em seguida, o conteúdo textual é extraído e segmentado em unidades menores, como capítulos ou seções. Cada segmento é convertido em um *embedding* vetorial, permitindo sua representação em espaço semântico.

Os vetores gerados são armazenados em banco vetorial juntamente com os *embeddings* das questões previamente cadastradas. Durante a consulta, o sistema calcula a similaridade entre vetores para recuperar questões semanticamente relacionadas ao conteúdo estudado.

As questões recuperadas passam por uma etapa de curadoria humana antes de serem disponibilizadas ao usuário final. Esse mecanismo reduz inconsistências provenientes do modelo semântico e aumenta a confiabilidade dos resultados apresentados.

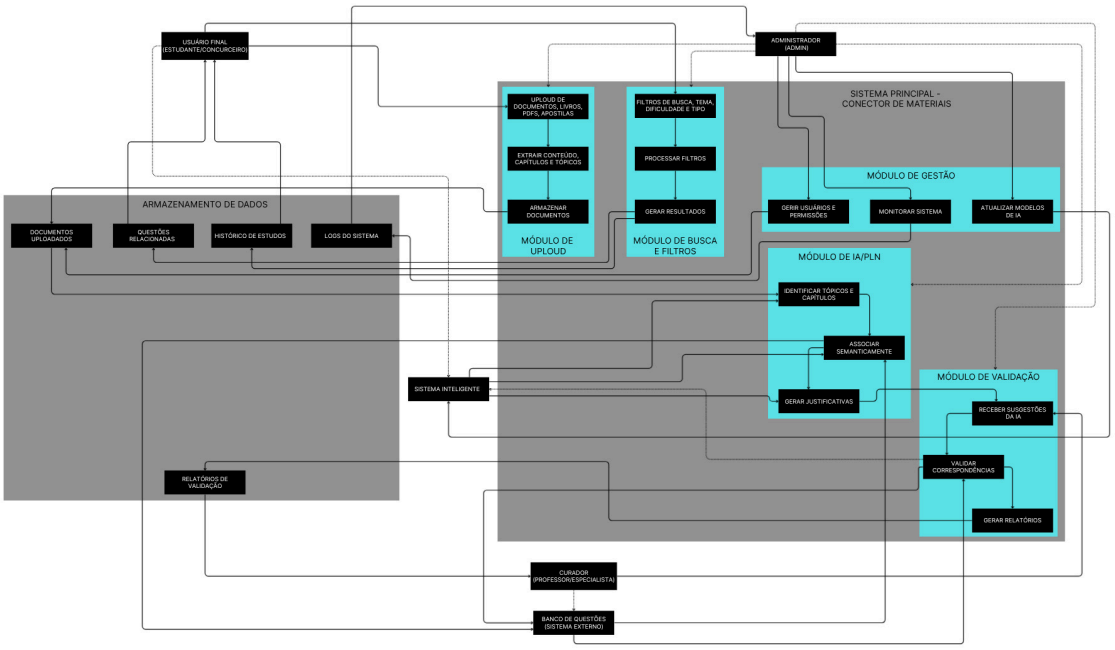

Figura 3 – Diagrama de blocos do sistema QuestBook.

3.3. Restrições

As restrições representam limitações não negociáveis impostas pelo escopo do projeto, recursos disponíveis ou fatores tecnológicos que impactam diretamente a arquitetura e o planejamento do desenvolvimento do sistema.

- **R-01: Restrição de Escopo (TCC):** O sistema foi desenvolvido como uma Prova de Conceito (*Proof of Concept* — PoC) no contexto de um Trabalho de Conclusão de Curso. Essa restrição limita o tempo disponível para desenvolvimento e prioriza funcionalidades essenciais em detrimento de recursos avançados.
- **R-02: Restrição de Recursos Humanos e Financeiros:** O projeto será desenvolvido exclusivamente pelos recursos humanos definidos no escopo do TCC, sem orçamento dedicado para aquisição de bases de dados comerciais, interfaces de programação de aplicações (*Application Programming Interface* — API) pagas ou infraestrutura computacional de alto desempenho em nuvem.

- **R-03: Restrição Tecnológica (Dependência de PLN):** A solução depende diretamente da utilização de modelos de Processamento de Linguagem Natural para análise semântica. A adoção de arquiteturas baseadas em modelos Transformer, como BERT, impõe restrições relacionadas à capacidade computacional necessária para inferência e processamento textual, além de limitar a aplicação a idiomas que possuam modelos pré-treinados adequados, especialmente a Língua Portuguesa.
- **R-04: Restrição de Fonte de Dados (Input):** O sistema está restrito às seguintes fontes de dados:
    1. **Documentos de estudo:** o sistema processa apenas documentos PDF textuais gerados digitalmente, não contemplando documentos digitalizados por imagem, cuja interpretação exigiria mecanismos de Reconhecimento Óptico de Caracteres (*Optical Character Recognition* — OCR), atualmente fora do escopo do projeto.
    2. **Base de questões:** o sistema depende de uma base de questões previamente estruturada, contendo enunciados, alternativas, gabaritos e metadados associados.

3.4. Suposições

As suposições representam hipóteses relacionadas ao ambiente operacional e aos dados utilizados pelo sistema. Caso tais hipóteses não sejam atendidas, poderão surgir impactos relevantes no funcionamento da solução proposta.

- **S-01: Suposição de Qualidade e Estrutura dos Dados (Input):** Assume-se que os documentos PDF de entrada possuam estrutura lógica mínima, contendo títulos, capítulos e seções passíveis de identificação algorítmica. Documentos sem qualquer organização estrutural podem comprometer a qualidade da extração textual e da segmentação semântica.

  Além disso, assume-se que a base de questões utilizada seja íntegra, contendo gabaritos corretos e metadados confiáveis. O sistema opera sob o princípio conhecido como *Garbage In, Garbage Out* (GIGO), segundo o qual entradas inconsistentes tendem a produzir resultados inadequados.

- **S-02: Suposição de Perfil do Curador:** Assume-se a existência de um usuário com perfil de curador, responsável pela validação das questões recuperadas pelo sistema. A qualidade final das sugestões apresentadas ao estudante depende diretamente da eficácia desse processo de curadoria humana.
- **S-03: Suposição de Relevância Semântica:** Assume-se que o relacionamento semântico entre capítulos e questões seja suficientemente representável em espaço vetorial, permitindo que modelos de PLN

capturem proximidade conceitual entre conteúdos textuais semanticamente relacionados.

3.5. Lista de Requisitos

A definição dos requisitos do sistema constitui etapa fundamental para o desenvolvimento do QuestBook, pois estabelece as funcionalidades necessárias para a recuperação semântica de questões a partir de materiais didáticos em formato PDF, bem como restrições relacionadas ao processamento textual e à interação com os usuários.

Considerando a natureza da aplicação, que envolve técnicas de Processamento de Linguagem Natural e mecanismos de busca semântica, torna-se necessário definir não apenas funcionalidades operacionais, mas também critérios relacionados ao desempenho, escalabilidade e qualidade das respostas geradas.

Os requisitos foram organizados em três categorias:

- requisitos funcionais;
- requisitos não funcionais;
- requisitos de domínio.

Essa classificação possibilita uma visão estruturada das diferentes características do sistema.

3.5.1 Requisitos Funcionais

Os requisitos funcionais descrevem as operações executadas pelo sistema QuestBook, incluindo funcionalidades relacionadas ao envio de documentos, extração textual, geração de *embeddings*, recuperação semântica e validação das questões sugeridas.

Tabela 2 – Requisitos funcionais do sistema.

ID	Descrição	Fonte (Rastreio)	Prioridade
RF-01	Upload e processamento de PDF	Escopo Inicial	Obrigatório
RF-02	Extração de contexto semântico por capítulo (heurística)	RB-001	Obrigatório

ID	Descrição	Fonte (Rastreio)	Prioridade
RF-03	Recuperação de questões por similaridade semântica	RB-001, RB-004	Obrigatório
RF-04	Interface de curadoria com trilha de auditoria	RB-006	Obrigatório
RF-05	Exclusão de questões sinalizadas (is_flawed) do resultado do aluno	RB-006	Obrigatório
RF-06	Rastreabilidade via SearchEvaluation e *score* de confiança	RB-005	Desejável
RF-07	Filtragem por banca e anti-repetição histórica	RB-002	Desejável

3.5.2 Requisitos Não Funcionais (RNF)

Os requisitos não funcionais estabelecem os critérios de qualidade e as restrições associadas ao sistema QuestBook, especialmente no que se refere ao desempenho no processamento de textos e na recuperação de informações, considerando operações como geração de embeddings e cálculo de similaridade; além disso, aspectos relacionados à usabilidade, responsividade da interface e escalabilidade do sistema devem ser considerados, de modo a garantir uma experiência adequada ao usuário e eficiência na execução das funcionalidades, conforme descrito na tabela 3.

Tabela 3 – Requisitos não funcionais do sistema.

ID	Descrição	Fonte (Rastreio)	Prioridade
RNF-01	O tempo de resposta para a geração da	RB-007	Obrigatório

	lista de questões sugeridas (RF-03) não deve exceder 10 segundos.		
RNF-02	A interface do sistema deve ser responsiva, adaptando-se para uso em desktops (navegador) e dispositivos móveis (celular).	RB-008	Desejável
RNF-03	O mecanismo de *matching* semântico deve utilizar modelos de *embeddings* (ex: BERT) para garantir alta precisão, em detrimento de métodos de palavra-chave (TF-IDF).	RB-004	Obrigatório
RNF-04	O sistema deve ser desenvolvido utilizando as linguagens Python para o *backend* e React para o *frontend*.	Decisão de Projeto	Obrigatório

3.5.3 Requisitos de Domínio (RD)

Os requisitos de domínio referem-se às características específicas do contexto educacional no qual o sistema QuestBook está inserido, considerando aspectos como a estruturação de materiais didáticos em formato PDF, organizados em capítulos ou seções, e a necessidade de associar esses conteúdos a questões avaliativas de forma contextualizada; além disso, inclui-se a validação das questões recuperadas por meio de curadoria humana, garantindo maior confiabilidade dos resultados apresentados, conforme apresentado na tabela 4.

Tabela 4 – Requisitos de domínio do sistema.

ID	Descrição	Fonte (Rastreio)	Prioridade
RD-01	O sistema deve ser capaz de processar questões de múltipla escolha com 4 ou 5 alternativas, nas quais todas possuem gabarito e metadados (ex.: ano, banca, etc).	Escopo Inicial	Obrigatório
RD-02	O sistema deve automatizar o processo de "ponte" entre o material de estudo e a plataforma de questões, que atualmente é predominantemente manual.	RB-003	Obrigatório

4. Detalhamento da Implementação e Arquitetura

Com o objetivo de validar os requisitos funcionais e não funcionais definidos anteriormente, foi desenvolvida uma Prova de Conceito (*Proof of Concept* — PoC) baseada em uma arquitetura modular inspirada no modelo de microsserviços. A

implementação priorizou a separação de responsabilidades entre os módulos de processamento semântico, persistência de dados e interface de usuário, organizando o sistema em três camadas lógicas principais: camada de serviços de Inteligência Artificial, camada de persistência poliglota e camada de exposição via Interface de Programação de Aplicações (*Application Programming Interface* — API) RESTful.

4.1 Stack Tecnológica e Backend

O núcleo do sistema foi implementado em Python 3.10 devido à ampla adoção da linguagem em aplicações de Inteligência Artificial e à disponibilidade de bibliotecas especializadas em Processamento de Linguagem Natural (PLN). Como framework web, utilizou-se o FastAPI [Ramírez 2018], escolhido pelo suporte nativo a operações assíncronas, tipagem estática e geração automática de documentação interativa para APIs REST.

A utilização do FastAPI contribuiu para simplificar o desenvolvimento dos endpoints e facilitar os testes de integração durante a implementação do sistema.

O servidor de aplicação utilizado foi o Uvicorn, compatível com o padrão *Asynchronous Server Gateway Interface* (ASGI), permitindo o processamento eficiente de operações assíncronas.

Conforme ilustrado na Figura 4, a arquitetura do sistema QuestBook organiza o fluxo de dados entre a interface do usuário, os serviços de Inteligência Artificial e a camada de persistência híbrida.

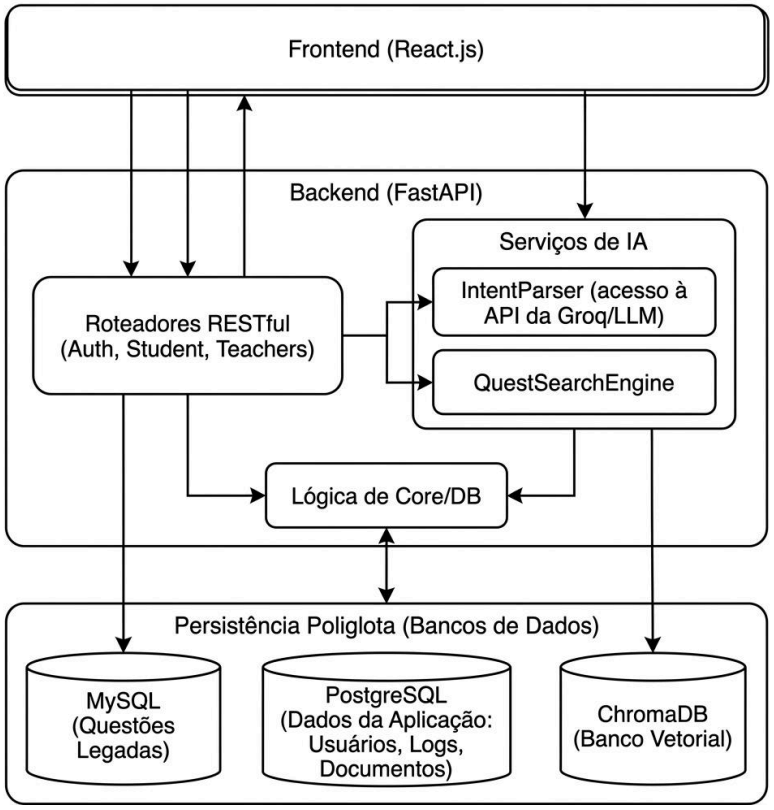

Figura 4 – Arquitetura geral do sistema QuestBook.

A aplicação é inicializada em um módulo central responsável por instanciar o servidor FastAPI, registrar os módulos de roteamento e configurar mecanismos de compartilhamento de recursos entre origens distintas (*Cross-Origin Resource Sharing* — CORS), permitindo a comunicação desacoplada entre o *frontend* React e o *backend* Python.

As tabelas do banco de dados relacional são criadas automaticamente durante a inicialização da aplicação por meio do uso de Mapeamento Objeto-Relacional (*Object-Relational Mapping* — ORM), reduzindo a necessidade de operações manuais durante o desenvolvimento da PoC.

4.2 Estrategia de Banco de Dados Híbrido

Uma das principais decisões arquiteturais do projeto diz respeito à estratégia de persistência. Optou-se por uma abordagem poliglota, na qual três tecnologias de banco de dados distintas coexistem, cada uma atendendo a uma necessidade específica do sistema. Esta decisão foi motivada pela necessidade de simular um cenário realista de integração com sistemas legados — comum em instituições de ensino — ao mesmo tempo em que se incorporavam as tecnologias mais adequadas para os novos requisitos do sistema.

A caracterização da base de questões utilizada na validação do sistema compreende um corpus histórico de concursos públicos na área de Tecnologia da Informação. Ao todo, a base relacional MySQL armazena 15.427 questões estruturadas, divididas em duas grandes disciplinas centrais do domínio de computação: Engenharia de Software (12.456 questões) e Arquitetura de Software (2.971 questões). O corpus é composto por questões de diversas bancas organizadoras de relevância nacional, com destaque quantitativo para o CESPE/CEBRASPE (4.401 questões), FCC (2.386 questões), CESGRANRIO (913 questões), FGV (786 questões) e Quadrix (442 questões).

Cada registro na tabela relacional questoes_engenharia_software possui os seguintes campos estruturados: identificador único (questao_id), disciplina, assunto, banca organizadora, ano de aplicação, enunciado textual, cinco alternativas de múltipla escolha (de 'A' a 'E') e o gabarito oficial correspondente. Durante a etapa de ingestão de dados para o banco vetorial, o pipeline de preparação realiza um tratamento de dados que inclui a normalização léxica dos metadados de banca (conversão para letras maiúsculas e remoção de espaços em branco adicionais), o preenchimento de campos nulos com valores padrão (e.g., 'DESCONHECIDA' ou '0' para anos ausentes), e a eliminação de registros cuja descrição textual do enunciado seja inferior a 60 caracteres, mitigando o ruído no banco de dados.

- **MySQL (Banco Legado de Questões):**
    - O banco relacional é responsável pelo armazenamento da base estruturada de questões, incluindo enunciados, alternativas, gabaritos e metadados, como banca organizadora e ano de aplicação.

    - Utilizou-se o MySQL devido à ampla adoção em sistemas legados de instituições educacionais e à compatibilidade com aplicações corporativas tradicionais. Essa decisão busca aproximar o ambiente experimental do contexto operacional encontrado em plataformas acadêmicas reais, nas quais bases históricas frequentemente permanecem em bancos relacionais já consolidados.

    - O acesso ao banco é realizado por meio de uma *session factory* dedicada (*get_questions_db*), separada da conexão principal da aplicação, conforme implementado no módulo backend/core/database.py. Essa abordagem favorece isolamento de responsabilidades e maior flexibilidade na integração entre múltiplas fontes de persistência.

- **PostgreSQL (Banco da Aplicação):**

- O banco relacional da aplicação é responsável pelo gerenciamento dos metadados dos documentos submetidos, dos capítulos extraídos e dos registros de curadoria e avaliação das buscas.

- A modelagem relacional contempla entidades como Document, Chapter e SearchEvaluation, permitindo rastreabilidade das operações realizadas pelo sistema e organização estruturada dos dados processados.

- O esquema relacional estabelece relações de integridade referencial entre as entidades, incluindo políticas de exclusão em cascata entre documentos e capítulos associados. Essa abordagem contribui para manter consistência dos dados e simplificar operações de manutenção na camada de persistência.

- **ChromaDB (Banco Vetorial):**
  - O ChromaDB [Chroma 2023] foi utilizado como banco vetorial responsável pelo armazenamento dos *embeddings* semânticos das questões e consultas processadas pelo sistema.

  - Esse componente desempenha papel central no mecanismo de recuperação semântica, permitindo consultas por similaridade vetorial em espaços de alta dimensionalidade. A utilização do banco vetorial possibilita identificar conteúdos semanticamente relacionados mesmo na ausência de correspondência lexical direta.

  - A camada de persistência foi estruturada de forma modular, separando o gerenciamento das conexões relacionais e vetoriais da aplicação. Essa abordagem favorece desacoplamento arquitetural, manutenção da aplicação e flexibilidade na integração entre diferentes mecanismos de armazenamento.

4.3 Pipeline de Ingestão e Processamento de Documentos

O fluxo de ingestão de documentos é executado de forma assíncrona e compreende diferentes etapas de validação, extração textual e persistência.

Inicialmente, o sistema verifica a integridade e o formato do arquivo PDF submetido pelo usuário. Em seguida, o conteúdo textual é extraído e consolidado em uma estrutura única para posterior processamento semântico.

Posteriormente, os metadados do documento e o conteúdo textual extraído são persistidos no banco relacional, permitindo rastreabilidade e reutilização futura das informações processadas.

Por decisão arquitetural, o processo de geração automática de sugestões de questões foi desacoplado da etapa de envio do documento. Essa estratégia reduz a latência percebida pelo usuário durante o *upload* do arquivo e transfere a recuperação semântica para o fluxo interativo de consulta via *chat*.

Um dos principais desafios técnicos identificados nessa etapa relaciona-se ao processamento de documentos PDF heterogêneos, contendo tabelas, imagens e estruturas complexas, que podem introduzir ruído semântico durante a extração textual. No contexto desta PoC, optou-se por tratar os documentos predominantemente como blocos textuais contínuos.

4.4 Módulo de Inteligência Artificial e Integração com LLM

O módulo de Inteligência Artificial foi estruturado em duas camadas principais: interpretação de intenção e recuperação semântica.

A camada de interpretação de intenção utiliza um Modelo de Linguagem de Grande Escala (*Large Language Model* — LLM) para converter consultas em linguagem natural em estruturas semânticas organizadas contendo tema da busca, quantidade de questões e critérios adicionais de filtragem.

A implementação utilizou a plataforma Groq e o modelo Llama 3.3 70B devido à baixa latência de inferência observada durante os testes experimentais, favorecendo maior fluidez na interação via chat.

Um dos principais desafios dessa etapa consistiu em garantir consistência estrutural nas respostas geradas pelo modelo. Para reduzir ambiguidades e minimizar respostas inconsistentes, adotou-se uma estratégia de *prompt engineering* baseada em:

- utilização de *few-shot prompting*;
- definição explícita de regras positivas e negativas;
- configuração determinística da geração textual;
- utilização de respostas estruturadas em formato JSON.

Adicionalmente, foi implementado um mecanismo de memória conversacional volátil, permitindo considerar interações anteriores durante novas consultas realizadas pelo usuário.

Outro componente relevante consiste na injeção contextual de trechos extraídos dos documentos PDF submetidos ao sistema. Quando capítulos específicos são mencionados durante a interação, o sistema identifica automaticamente os trechos

correspondentes e os utiliza como contexto adicional para geração da consulta semântica.

4.5 Motor de Busca Vetorial: Fundamentos Matemáticos e Implementação

A camada de recuperação semântica foi implementada utilizando modelos de representação vetorial capazes de converter textos em *embeddings* semânticos.

4.5.1 *Representação Vetorial de Texto (Embeddings)*

Um modelo de *embedding* textual é uma função $\varphi: T \rightarrow \mathbb{R}^n$ que mapeia um texto $t \in T$ para um vetor de números reais em um espaço de dimensão n. No presente sistema, o modelo paraphrase-multilingual-MiniLM-L12-v2 da biblioteca sentence-transformers (REIMERS; GUREVYCH, 2019) produz vetores em $\mathbb{R}^{384}$, ou seja, cada texto — seja o enunciado de uma questão ou a consulta do usuário — é representado por um vetor de 384 dimensões. A propriedade central deste espaço é que textos semanticamente próximos produzem vetores geometricamente próximos, independentemente da sobreposição lexical entre eles. Esta propriedade é herdada do pré-treinamento do modelo sobre grandes corpora multilíngues, que induz uma geometria semântica ao espaço vetorial.

A seleção do modelo paraphrase-multilingual-MiniLM-L12-v2 da biblioteca sentence-transformers baseou-se em critérios de eficiência computacional e adequação ao idioma português. O modelo adota uma arquitetura de destilação de conhecimento baseada em redes Transformers (MiniLM), contendo 12 camadas e gerando representações densas em um espaço de dimensão reduzida (384 dimensões em R^384). Essa característica resulta em uma economia significativa de memória e redução drástica no tempo de latência durante as consultas vetoriais no ChromaDB, quando comparado a modelos maiores de 768 dimensões (como o BERTimbau nativo para português) ou APIs comerciais proprietárias.

Ademais, embora modelos baseados unicamente em inglês ou representações brutas de codificadores como o BERT tradicional capturem relações sintáticas, eles carecem de calibração para similaridade de sentenças inteiras em português. O modelo selecionado foi pré-treinado especificamente em tarefas de identificação de paráfrases e tradução alinhada para mais de 50 idiomas, garantindo que sentenças semanticamente equivalentes em português sejam projetadas em coordenadas geometricamente próximas na base vetorial, viabilizando a execução eficiente do sistema em ambientes locais de baixo custo (sem necessidade obrigatória de placas gráficas aceleradoras GPU).

A escolha do modelo multilíngue justifica-se pela natureza do corpus de questões em português. Modelos treinados exclusivamente em inglês produziriam representações inadequadas para o domínio, dado que as relações semânticas capturadas dependem diretamente da distribuição estatística da língua observada durante o treinamento.

4.5.2 Métricas de Similaridade: Distância Cosseno e Distância Euclidiana

Dadas duas representações vetoriais u, v $\in \mathbb{R}^n$, duas métricas são amplamente empregadas para quantificar a similaridade semântica entre os textos correspondentes:

Similaridade Cosseno: Mede o cosseno do ângulo θ formado entre os dois vetores no espaço de alta dimensão:

$$cos(\theta) = \frac{u \cdot v}{\|u\| \cdot \|v\|} \quad (1)$$

- (u) representa o vetor correspondente ao primeiro texto;
- (v) representa o vetor correspondente ao segundo texto;
- (u \cdot v) corresponde ao produto interno entre os vetores;
- (|u|) e (|v|) representam as magnitudes vetoriais.

Onde $u \cdot v = \Sigma_i u_i v_i$ é o produto interno e $\|u\| = \sqrt{(\Sigma_i u_i^2)}$ é a norma L2 do vetor. O resultado varia no intervalo [-1, 1], onde 1 indica vetores paralelos (máxima similaridade), 0 indica vetores ortogonais (sem relação semântica) e -1 indica vetores antiparalelos (sentidos opostos).

Distância Euclidiana (L2): Mede a distância geométrica entre os dois pontos no espaço $\mathbb{R}^n$:

$$d\_L2(u, v) = \|u - v\| = \sqrt{(\Sigma_i (u_i - v_i)^2)} \quad (2)$$

- (d_{L2}(u,v)) representa a distância euclidiana entre os vetores;
- (u_i) e (v_i) representam os componentes vetoriais em cada dimensão;
- (n) representa a dimensionalidade do espaço vetorial.

O valor varia no intervalo [0, +∞), onde 0 indica vetores idênticos.

Equivalência com Vetores Normalizados: Uma propriedade de fundamental importância para a implementação do QuestBook é a equivalência entre as duas métricas quando os vetores são normalizados (i.e., quando $\|u\| = \|v\| = 1$). Neste caso, expandindo o quadrado da norma da diferença:

$$\|u - v\|^2 = \|u\|^2 - 2(u \cdot v) + \|v\|^2 = 1 - 2cos(\theta) + 1 = 2(1 - cos(\theta)) \quad (3)$$

Portanto, $d\_L2(u, v)^2 = 2(1 - cos(\theta))$, o que demonstra que, com vetores normalizados, minimizar a distância L2 é matematicamente equivalente a maximizar a similaridade cosseno. Esta equivalência é explorada explicitamente na implementação: o parâmetro normalize_embeddings=True é passado ao método model.encode(), garantindo que todos os vetores gerados possuam norma unitária ($\|u\| = 1$), o que habilita o ChromaDB a utilizar a distância L2 como substituto direto e computacionalmente eficiente para a similaridade cosseno.

4.5.3 Implementação do Pipeline de Recuperação

O método central search_relevant_questions executa o pipeline de recuperação em quatro etapas:

1. **Geração do Vetor de Consulta:** O texto da search_query, produzido pelo IntentParser, é codificado com normalização L2, produzindo o vetor unitário $q \in \mathbb{R}^{384}$ que representa semanticamente a intenção do usuário.
2. **Estratégia de Sobra (Over-fetching):** O sistema solicita ao ChromaDB um número de candidatos equivalente a três vezes a quantidade final desejada (candidates_limit = limit × 3). Esta margem é necessária porque os filtros de pós-processamento subsequentes podem descartar candidatos, e uma seleção prematuramente enxuta comprometeria a cobertura do resultado final.
3. **Filtragem Estrutural:** A consulta ao ChromaDB pode incluir um filtro de metadados (where clause) composto. Filtros obrigatórios — como o tamanho mínimo do enunciado (tamanho ≥ 60 caracteres) — são sempre aplicados para eliminar questões espúrias com conteúdo insuficiente. Filtros opcionais, como a banca organizadora, são compostos via operador lógico $and quando detectados pelo IntentParser.
4. **Filtros de Pós-Processamento Anti-Duplicação:** Sobre o conjunto de candidatos retornado pelo ChromaDB, dois filtros adicionais são aplicados em cascata, operando diretamente sobre os embeddings recuperados:

    a. **Anti-Duplicação Léxica:** O texto de cada questão é normalizado (remoção de caracteres não-alfanuméricos via regex e conversão para minúsculas) e comparado a um conjunto seen_texts de textos já aceitos. Este filtro tem complexidade O(1) por lookup e elimina cópias textuais exatas.

    b. **Anti-Duplicação Semântica:** Para cada candidato com vetor v_i, calcula-se a distância L2 ao quadrado em relação ao vetor v_j de cada questão já presente no conjunto aceito:

$$dist\_sq(v_i, v_j) = \Sigma_k (v_{ik} - v_{jk})^2 \quad (4)$$

Candidatos com dist_sq < 0,15 são descartados como semanticamente redundantes. Dado que os vetores são normalizados, e utilizando a equivalência demonstrada na Seção 4.5.2, este limiar corresponde a:

$$dist\_sq = 2(1 - cos(\theta)) < 0,15 \Rightarrow cos(\theta) > 0,925 \quad (5)$$

Ou seja, o filtro rejeita questões com similaridade cosseno superior a 92,5%, tratando-as como semanticamente equivalentes e, portanto, redundantes para o propósito pedagógico do sistema.

4.5.4 Limitação de Entrada e Truncamento de Sentenças

O modelo de representação vetorial adotado possui uma restrição inerente quanto ao tamanho máximo de sequência de entrada (context window), limitada a 128 tokens (aproximadamente 90 a 110 palavras na língua portuguesa). No contexto de um banco de questões de concursos públicos, onde enunciados frequentemente contêm textos de apoio extensos, trechos de código de programação, tabelas estruturadas ou fórmulas matemáticas, essa limitação exige uma análise de impacto.

Caso o texto de uma questão exceda o limite de 128 tokens, o mecanismo de geração de embeddings realiza o truncamento automático do conteúdo excedente, processando apenas o fragmento inicial para compor a coordenada vetorial da questão. O impacto direto deste truncamento é a perda de termos técnicos ou enunciados de comandos localizados no terço final de questões longas, reduzindo ligeiramente a precisão semântica de busca para esses casos específicos. Para mitigar esse problema no QuestBook, o sistema adota duas abordagens complementares: (i) a indexação mantém o texto completo no banco relacional MySQL para que, após a busca por similaridade semântica (baseada no início do enunciado), a exibição e a interação do estudante ocorram com a totalidade do texto original (etapa de hidratação); e (ii) o processamento de materiais didáticos em PDF realiza a segmentação textual (chunking) baseada na estrutura lógica de parágrafos e subtópicos, garantindo que o contexto injetado no LLM permaneça abaixo do limite crítico e evite a perda de trechos conceituais relevantes.

4.5.5 Cálculo do Score de Confiança

O ChromaDB retorna, para cada resultado, a distância L2 d entre o vetor de consulta q e o vetor do documento recuperado v_i. Com vetores normalizados, esta distância varia no intervalo [0, 2], onde 0 representa similaridade perfeita (cos(θ) = 1) e 2 representa dissimilaridade máxima (cos(θ) = -1).

O score de confiança c é calculado pela seguinte transformação linear:

$$c = max(0,0;\ 1,0 - \frac{d}{2}) \quad (6)$$

Esta transformação mapeia bijetivamente o intervalo de distâncias [0, 2] para o intervalo de confiança [0, 1], onde c = 1,0 indica similaridade perfeita e c = 0,0 indica dissimilaridade máxima. O operador max(0,0; ·) garante que distâncias superiores a 2 (que podem ocorrer em casos numéricos extremos) não produzam scores negativos. Esta normalização é equivalente a expressar a similaridade cosseno em uma escala de zero a um:

$$c = 1,0 - \frac{d}{2} = 1,0 - \frac{\sqrt{(2(1 - cos(\theta)))}}{2} \approx \frac{(1 + cos(\theta))}{2} \quad (7)$$

[para vetores normalizados]

O valor resultante é arredondado a quatro casas decimais e exposto na resposta da API como o campo confidence de cada questão retornada.

4.6 Pipeline de Recuperação e Hidratação de Dados (Rota /chat_questions)

O endpoint POST /student/chat_questions orquestra o fluxo completo de recuperação de questões em oito etapas sequenciais:

1. **Extração de Contexto Documental** (heurística de capítulo, quando aplicável).
2. **Parsing de Intenção** via IntentParser → produz topic, limit, banca e search_query.
3. **Validação de Intenção:** Requisições com topic == "INVALIDO" são interceptadas e retornam uma mensagem de sistema sem consultar os bancos de dados.

4. **Busca Vetorial** via QuestSearchEngine → retorna lista de candidatos ordenados por score de confiança decrescente (conforme fórmula descrita na Seção 4.5.4).
5. **Trava de Confiança (*Confidence Gate*):** Candidatos com score $c < 0{,}65$ são descartados antes da etapa de hidratação. Traduzindo para a métrica de similaridade cosseno (via a equivalência $c \approx (1 + \cos(\theta))/2$), o limiar $c = 0{,}65$ corresponde a $\cos(\theta) \approx 0{,}30$, o que significa que apenas questões com uma correlação angular moderada a alta em relação à consulta do usuário são consideradas. Este limiar foi calibrado empiricamente para o domínio de concursos públicos em português, balanceando abrangência e precisão dos resultados.
6. **Anti-Repetição Histórica:** Para usuários autenticados, os IDs de questões já respondidas (tabela UserAnswer) são recuperados do PostgreSQL e removidos da lista de candidatos, garantindo a diversidade do conteúdo apresentado em sessões recorrentes.
7. **Hard Ban de Curadoria:** Os IDs restantes são cruzados com a tabela SearchEvaluation. Questões marcadas com is_flawed = 1 por curadores humanos são permanentemente excluídas da resposta, implementando o mecanismo de curadoria ativa descrito em RF-04.
8. **Hidratação via ORM:** Os IDs finais aprovados são utilizados para recuperar os objetos QuestaoLegada completos (enunciado, alternativas e gabarito) do MySQL, por meio de uma consulta IN otimizada. A resposta é montada respeitando a ordem de relevância original retornada pelo banco vetorial, de modo que a questão mais similar à consulta do usuário seja sempre apresentada em primeiro lugar.

4.7 Painel de Curadoria e Audit Trail

Para atender ao requisito RF-04, foi implementado um módulo de curadoria acessível exclusivamente a usuários com perfil de curador.

O módulo disponibiliza o endpoint POST /teachers/audit_evaluate, responsável pelo recebimento de avaliações em lote (*batch evaluation*). Cada avaliação contém a consulta executada pelo usuário, o identificador da questão avaliada, a pontuação de relevância atribuída pelo curador e o sinalizador booleano is_flawed, utilizado para indicar inconsistências ou inadequações na questão.

As avaliações são persistidas na entidade SearchEvaluation do banco PostgreSQL, associando cada registro ao identificador do curador autenticado (teacher_id). Essa abordagem implementa uma trilha de auditoria (*audit trail*) capaz de rastrear retrospectivamente quais questões foram avaliadas, quais foram sinalizadas como inadequadas e em resposta a quais consultas foram recuperadas.

Além de apoiar o controle de qualidade das recomendações, esse mecanismo contribui para rastreabilidade operacional e futuras análises de desempenho do mecanismo de recuperação semântica.

4.8 Interface de Usuário (Frontend)

A camada de apresentação do QuestBook foi desenvolvida utilizando React na versão 18. Como ferramenta de *build*, utilizou-se o Vite, escolhido pela baixa latência no processo de compilação e pelo suporte nativo a módulos ES (*ECMAScript Modules*).

A comunicação entre frontend e backend ocorre exclusivamente por meio de requisições HTTP assíncronas gerenciadas pela biblioteca Axios. Essa abordagem promove desacoplamento entre a camada de apresentação e a lógica de negócio da aplicação, seguindo os princípios arquiteturais de APIs REST.

A separação entre frontend e backend também favorece modularidade, manutenção da aplicação e futura escalabilidade da arquitetura.

4.8.1 Roteamento e Controle de Acesso

O sistema de navegação foi implementado utilizando a biblioteca react-router-dom (v6), responsável pelo gerenciamento declarativo das rotas da aplicação.

As rotas definidas incluem funcionalidades de autenticação, submissão de documentos e acesso aos painéis específicos de cada perfil de usuário, incluindo:

- /login;
- /register;
- /upload;
- /student-dashboard;
- /dashboard;
- /forgot-password.

O controle de acesso é aplicado diretamente na camada de roteamento por meio do componente Navigate. O mecanismo considera dois critérios principais:

- presença de token JWT válido armazenado localmente;
- papel do usuário autenticado (userRole).

Com base nessas informações, o sistema realiza redirecionamento automático para as interfaces compatíveis com o perfil autenticado. Usuários com papel de curador são direcionados ao painel de auditoria, enquanto usuários do perfil estudante acessam exclusivamente as funcionalidades acadêmicas correspondentes.

Essa estratégia centraliza as regras de autorização na camada de navegação, reduzindo inconsistências de acesso entre diferentes interfaces da aplicação.

4.8.2 Gerenciamento de Estado e Autenticação

O gerenciamento de estado da aplicação foi implementado utilizando os hooks nativos do React, principalmente useState e useEffect. A decisão de não utilizar bibliotecas externas de gerenciamento de estado, como Redux, buscou reduzir a complexidade arquitetural da Prova de Conceito (PoC).

Entre os principais estados mantidos pela aplicação destacam-se:

- estado de autenticação do usuário;
- histórico de sessões de chat;
- sessão de chat ativa;
- conteúdo acumulado das interações.

O histórico de conversas é persistido localmente de forma associada ao identificador do usuário autenticado, permitindo isolamento entre contas distintas e preservação das sessões após recarregamentos da interface.

O mecanismo de autenticação foi implementado utilizando JWT com suporte a *refresh token*. Para automatizar o gerenciamento das credenciais, foram configurados interceptores globais na camada de comunicação HTTP da aplicação.

O primeiro interceptor adiciona automaticamente o *access token* ao cabeçalho Authorization das requisições enviadas ao backend. O segundo interceptor monitora respostas HTTP com código 401 (*Unauthorized*) e realiza automaticamente o processo de renovação do token de acesso por meio do endpoint de atualização de autenticação.

Caso a renovação falhe, o sistema invalida a sessão do usuário e redireciona automaticamente a interface para o fluxo de autenticação. Essa estratégia reduz interrupções durante a navegação e melhora a experiência de uso da aplicação.

4.8.3 Componentes Principais

A interface é organizada em componentes React reutilizáveis, seguindo o princípio de responsabilidade única:

Sidebar.jsx: Componente de navegação lateral que exibe o histórico de sessões de chat do usuário, com suporte a busca textual por título, fixação (pin), renomeação inline e exclusão de sessões. O histórico é ordenado de modo que sessões fixadas apareçam sempre no topo. O componente recebe todas as ações de manipulação do histórico via props (padrão callback lifting), mantendo o estado autoritativo no componente pai App.jsx.

ChatQuestions.jsx: Componente central da interface do aluno, responsável pela entrada de texto e pelo disparo das requisições ao backend. Implementa um campo de texto com redimensionamento automático (auto-resize) limitado a 200px de altura e suporte ao atalho Ctrl+Enter / Enter para envio. O fluxo de envio é bifurcado: se um arquivo PDF foi selecionado, a requisição POST /student/upload_document é executada primeiro com o payload em multipart/form-data, e o document_id retornado é então acoplado à requisição subsequente POST /student/chat_questions. Durante o

processamento, o campo é desabilitado e um ícone de carregamento animado substitui o botão de envio, fornecendo feedback visual ao usuário (RNF-05).

QuestionList.jsx: Componente responsável pela renderização das questões retornadas pelo backend. Suporta o modelo de histórico acumulado, renderizando cada interação de chat como um bloco distinto dentro de uma mesma sessão. Para cada questão exibida, o componente gerencia um estado local de resposta do usuário (selectedAnswers), fornece feedback visual imediato com código de cores (verde para correta, vermelho para incorreta) após a submissão, e registra a resposta assincronamente no backend via POST /student/answer para atualização do histórico de desempenho do aluno (RF-05).

SearchAudit.jsx: Componente exclusivo da interface do curador, integrado ao TeacherDashboard.jsx. Permite ao professor simular buscas como um aluno faria, visualizando as questões retornadas pela IA com seus respectivos scores de confiança (exibidos como percentual) e as alternativas com o gabarito destacado. Para cada questão, o curador pode atribuir uma nota de relevância de 1 a 5 e ativar o flag is_flawed para inativar permanentemente a questão. As avaliações são acumuladas em estado local e submetidas em lote ao endpoint POST /teachers/audit_evaluate via requisição autenticada com JWT, implementando o ciclo completo de curadoria descrito nos requisitos RF-04, RF-05 e RF-06.

.

4.8.4 Utilitário de Exportação PDF

Para suportar estudo offline e portabilidade do conteúdo recuperado, foi implementado um módulo de exportação de questões em formato PDF.

O módulo utiliza a biblioteca jsPDF para gerar documentos contendo as questões recuperadas durante a sessão de interação do usuário. O documento exportado inclui enunciados, alternativas e uma seção final de gabarito, organizada com paginação automática.

Durante o processo de geração do arquivo, conteúdos textuais com marcação HTML são normalizados antes da renderização no documento final, garantindo consistência visual e compatibilidade de formatação.

A funcionalidade de exportação é disponibilizada diretamente na interface da aplicação sempre que existe uma sessão de questões ativa, permitindo ao usuário armazenar localmente o material recuperado pelo sistema.

4.8.5 Decisões de Projeto da Interface

Uma decisão arquitetural relevante consistiu na persistência local do histórico de conversas no lado do cliente, sem armazenamento dedicado no backend. Essa abordagem eliminou a necessidade de endpoints específicos para recuperação de sessões históricas, reduzindo a complexidade da infraestrutura durante a fase de Prova de Conceito (PoC).

Como contrapartida, o histórico permanece vinculado ao navegador utilizado pelo usuário, sem mecanismos de sincronização entre dispositivos. Essa limitação foi considerada aceitável dentro do escopo experimental do projeto e registrada como possibilidade de evolução futura.

Outro aspecto relacionado à experiência de uso envolve a geração automática dos títulos das sessões de chat. O sistema utiliza preferencialmente o atributo topic, retornado pelo módulo de interpretação de intenção, para gerar títulos semanticamente representativos das consultas realizadas.

Quando o título semântico não apresenta conteúdo válido ou suficientemente descritivo, o sistema utiliza estratégias de fallback baseadas no nome do documento submetido ou em trechos da mensagem original do usuário. Essa abordagem melhora a organização das sessões e facilita a navegação no histórico de interações.

5. Resultados

Ao término da presente etapa de desenvolvimento da Prova de Conceito, foram implementadas e validadas as funcionalidades centrais do sistema QuestBook. Esta seção apresenta os resultados obtidos, organizados em torno das funcionalidades desenvolvidas, da estratégia de validação empregada e das observações qualitativas relacionadas ao desempenho da recuperação semântica.

5.1 Funcionalidades Implementadas

As funcionalidades previstas para a presente etapa do projeto foram concluídas e integradas ao sistema. A Tabela 8 apresenta os requisitos funcionais implementados e seus respectivos estados de conclusão.

Tabela 8 — Funcionalidades implementadas no sistema QuestBook.

ID	Descrição	Status
RF-01	Upload e processamento de PDF	Implementado
RF-02	Extração de contexto semântico por capítulo (heurística)	Implementado
RF-03	Recuperação de questões por similaridade semântica	Implementado
RF-04	Interface de curadoria com trilha de auditoria	Implementado

RF-05	Exclusão de questões sinalizadas (is_flawed) do resultado do aluno	Implementado
RF-06	Rastreabilidade via SearchEvaluation e *score* de confiança	Implementado
RF-07	Filtragem por banca e anti-repetição histórica	Implementado

A validação metodológica do QuestBook foi estruturada a partir de um delineamento experimental que integrou dados reais de concursos públicos com materiais didáticos de referência acadêmica. O corpus de questões utilizado como base de dados foi originado de exames históricos reais, totalizando 15.427 questões estruturadas e divididas nas disciplinas de Engenharia de Software (12.456 questões) e Arquitetura de Software (2.971 questões). O processo de preparação desses dados para a indexação vetorial envolveu etapas sistemáticas de limpeza e transformação por meio do script de indexação (indexer.py). Cada registro passou por normalização lexical dos metadados de banca organizadora e ano de aplicação, remoção de enunciados inconsistentes ou excessivamente curtos (com limite mínimo de 60 caracteres) e higienização de tags HTML residuais. Posteriormente, as questões tratadas foram convertidas em vetores densos de 384 dimensões utilizando o modelo paraphrase-multilingual-MiniLM-L12-v2 e persistidas no banco vetorial ChromaDB.

Para a simulação de uso e indução de intenção de busca, foram testados materiais didáticos digitais no formato PDF de nível universitário, especificamente capítulos de livros de referência de Engenharia de Software (abordando tópicos de Engenharia de Requisitos, Modelagem UML, Ciclos de Vida e Testes de Software) e Sistemas de Banco de Dados. O pipeline do sistema realizou a extração assíncrona do conteúdo textual utilizando a biblioteca pypdf, seguida de uma indexação relacional na tabela Chapter do PostgreSQL. A correspondência contextual e a resolução de referências no chat foram testadas por meio da inserção de parâmetros de busca contextualizada, onde o sistema extrai fragmentos dinâmicos de até 20.000 caracteres ao redor dos termos identificados pela heurística de localização de capítulos (e.g., 'Capítulo 2' ou 'Seção 3'), garantindo que o contexto alimentado ao LLM contivesse a informação conceitual exata da seção que estava sendo lida pelo aluno.

Por fim, a avaliação da qualidade da recuperação semântica foi executada por meio de um protocolo de auditoria simulado. O experimento contou com a participação de dois professores especialistas da área de desenvolvimento de software, que atuaram como curadores. Eles simularam interações típicas de estudantes na interface do curador (SearchAudit.jsx), avaliando individualmente a relevância das primeiras 5 questões recomendadas pela IA para cada um dos tópicos de teste em uma escala ordinal de 1 (totalmente irrelevante) a 5 (altamente relevante). Os resultados dessas avaliações e as sinalizações de erros estruturais (gabaritos inconsistentes ou formatações inadequadas) foram registrados de forma rastreável na tabela SearchEvaluation do PostgreSQL. Esse

desenho experimental permitiu não apenas auditar a adequação pedagógica das questões recomendadas, mas também validar a eficácia da trava de confiança de 0,65 e calcular a taxa de precisão geral do sistema de busca semântica.

5.2 Estratégia de Validação e Testes Automatizados

A validação do sistema foi conduzida por meio de uma suíte de testes automatizados desenvolvida com o framework Pytest, organizada em duas camadas distintas segundo os princípios da pirâmide de testes.

Testes Unitários (backend/tests/unit/test_llm_agent.py): Os testes unitários focam no isolamento do comportamento da classe IntentParser, verificando se a lógica de interpretação de intenção produz os resultados esperados sem realizar chamadas reais à API da Groq. O isolamento é obtido por meio do mock do cliente Groq via pytest-mock, substituindo as chamadas de rede por respostas pré-definidas (stubs). Os cenários validados incluem:

- **Prompt básico:** Verifica se uma solicitação genérica ("Me dê 5 questões de engenharia de software") produz um JSON com os campos topic, limit e search_query corretamente preenchidos.
- **Detecção de intenção inválida:** Verifica se solicitações fora do domínio de estudo (e.g., "receita de bolo") resultam no valor "INVALIDO" no campo topic.
- **Injeção de contexto documental:** Verifica se, quando um trecho de documento é fornecido (e.g., "O Capítulo 2 descreve os Processos Ágeis"), o modelo produz uma search_query contendo os assuntos reais do capítulo, e não o termo literal "capítulo 2".
- **Qualidade da *search_query*:** Verifica se a search_query gerada não contém termos genéricos como "questões sobre", assegurando que apenas conceitos técnicos sejam transmitidos ao banco vetorial.

Testes de Integração (backend/tests/integration/test_student_routes.py): Os testes de integração validam o comportamento do endpoint POST /student/chat_questions em nível de rota HTTP, utilizando o TestClient do FastAPI para simular requisições completas. Ambos os serviços de IA (IntentParser e QuestSearchEngine) são substituídos por mocks via unittest.mock.patch, enquanto as interações com o banco de dados ocorrem em um banco SQLite em memória, provisionado pela fixture conftest.py. Os cenários de integração validados incluem:

- **Fluxo nominal sem documento:** Valida que uma requisição de chat resulta em uma resposta HTTP 200 com a estrutura correta (questions, topic).
- **Heurística de capítulo:** Valida que, quando o usuário menciona "capítulo 2" e o documento contém esse termo, a janela de 20.000 caracteres capturada inclui o conteúdo semântico correto da seção referenciada.
- **Rejeição por baixa confiança:** Valida que questões com score de confiança inferior a 0,65 não são incluídas na resposta final, retornando em seu lugar uma mensagem de sistema informativa.
- **Anti-repetição histórica:** Valida que questões previamente respondidas pelo usuário (presentes na tabela UserAnswer) são excluídas do conjunto de resultados, mesmo que o banco vetorial as retorne como candidatas.

5.3 Observações Qualitativas sobre o Desempenho da Busca Semântica

Em simulações realizadas com a base de questões disponível, o sistema demonstrou capacidade de recuperar questões semanticamente relacionadas ao conteúdo consultado mesmo na ausência de correspondência lexical direta. Por exemplo, a query "gerenciamento de memória, alocação dinâmica, ponteiros" foi capaz de recuperar questões sobre "estruturas de dados" e "linguagem C" sem que esses termos estivessem presentes na consulta, evidenciando o benefício da abordagem vetorial sobre mecanismos de busca baseados em palavras-chave.

A trava de confiança em 0,65 mostrou-se eficaz em eliminar sugestões de baixa relevância semântica, reduzindo o ruído nos resultados apresentados ao usuário. O limiar foi calibrado empiricamente com base na análise da distribuição dos scores retornados pelo ChromaDB para consultas representativas do domínio de concursos públicos.

O principal gargalo de desempenho identificado durante os testes foi o tempo de carregamento inicial do modelo de embedding (SentenceTransformer) na inicialização do servidor, que pode demandar alguns segundos. Este comportamento é esperado e aceitável no contexto de uma PoC, dado que o carregamento ocorre uma única vez e o modelo permanece em memória para todas as requisições subsequentes, em conformidade com o requisito não-funcional RNF-02.

5.4 Demonstração Visual do Protótipo

A validação de um sistema interativo baseado em inteligência artificial não se esgota nos testes automatizados. A experiência de uso — a forma como a interface comunica os resultados ao usuário e fornece feedback adequado — constitui uma dimensão de qualidade igualmente relevante para a avaliação do cumprimento dos requisitos funcionais e não-funcionais. Neste sentido, esta seção apresenta capturas de tela representativas do sistema QuestBook em operação, evidenciando, de forma tangível, os fluxos de interação implementados nas camadas de frontend e de backend descritas nas seções anteriores. As figuras foram obtidas a partir da execução local da PoC em ambiente de desenvolvimento.

A Figura 5 apresenta a interface do módulo de recuperação de questões, utilizada pelos estudantes durante as interações com o sistema.

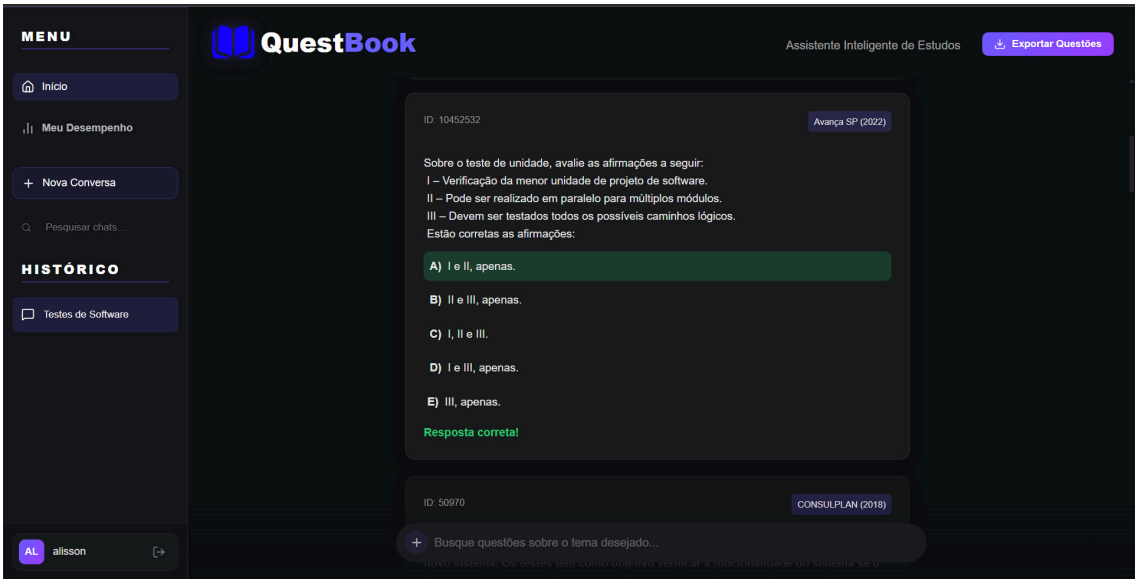

Figura 5 — Interface do Aluno: Recuperação de Questões e Feedback de Resposta

A interface exibida na Figura 5 evidencia o funcionamento do fluxo completo de recuperação semântica implementado no sistema. O usuário realiza uma consulta em linguagem natural e recebe questões relacionadas semanticamente ao conteúdo solicitado. Observa-se também a presença de mecanismos de feedback visual imediato após a submissão da resposta, indicando acertos e erros por meio de diferenciação visual. Esse comportamento atende aos requisitos relacionados à usabilidade e à interação em tempo real previstos para a aplicação.

A Figura 6 apresenta o painel de auditoria destinado aos usuários com papel de curador.

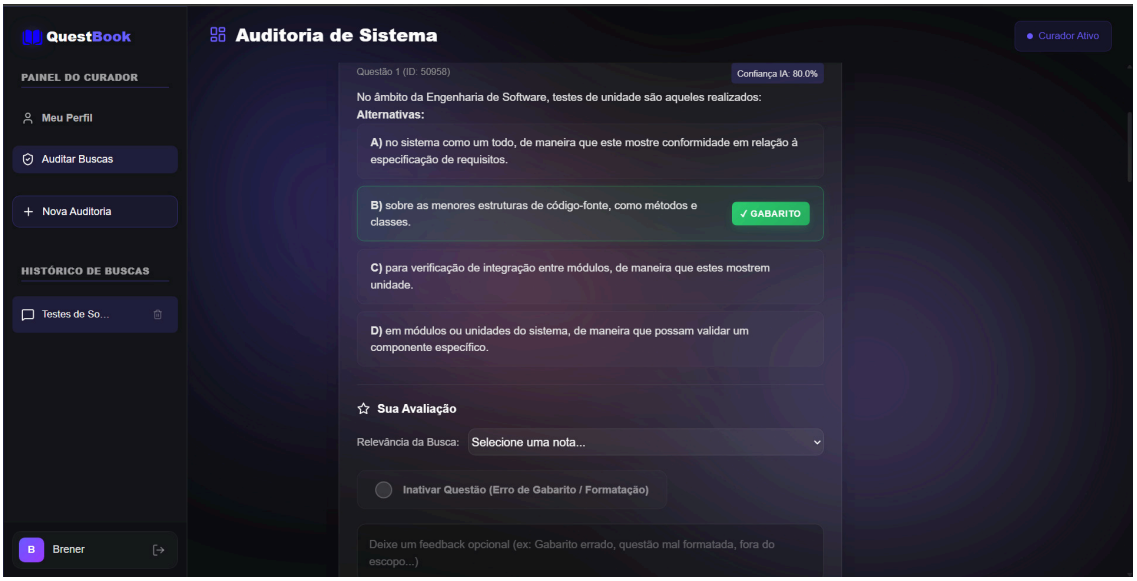

Figura 6 — Painel do Curador para Auditoria de Questões

O painel de curadoria apresentado na Figura 6 demonstra a implementação dos mecanismos de auditoria e validação humana previstos nos requisitos funcionais do sistema. Por meio dessa interface, o curador pode avaliar a relevância das questões recuperadas, visualizar os respectivos níveis de confiança calculados pela camada semântica e sinalizar conteúdos inadequados ou inconsistentes. Essa etapa de validação contribui diretamente para o aumento da confiabilidade das sugestões disponibilizadas aos estudantes.

A Figura 7 apresenta a documentação interativa da API gerada automaticamente pelo FastAPI.

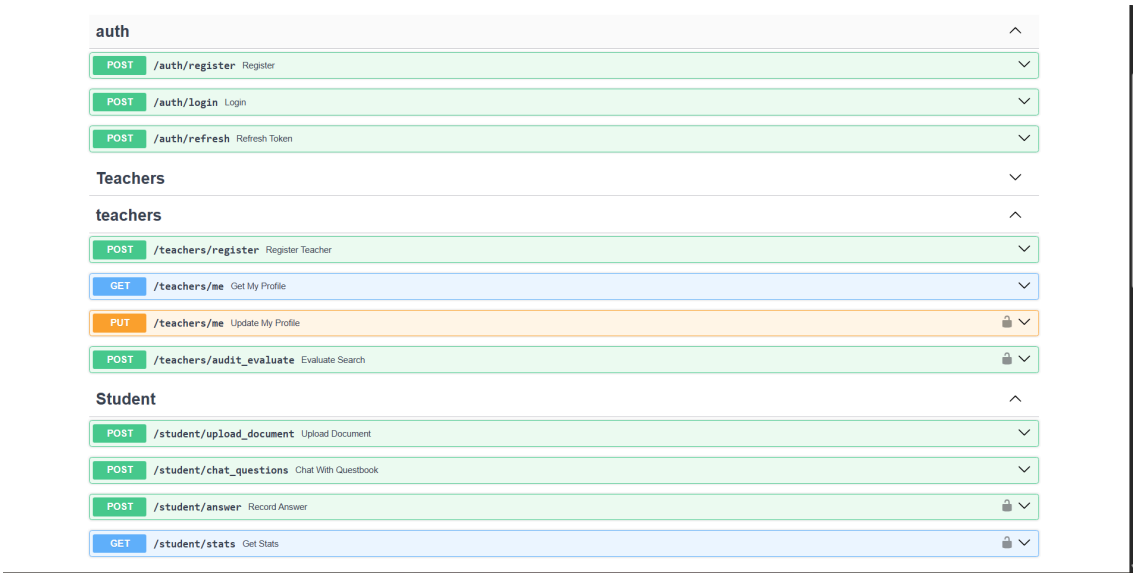

Figura 7 — Documentação Interativa da API (Swagger UI): Arquitetura RESTfuls

A documentação ilustrada na Figura 7 evidencia a organização dos endpoints da aplicação segundo os princípios arquiteturais REST. A geração automática da documentação por meio do padrão OpenAPI contribuiu significativamente para os testes manuais realizados durante o desenvolvimento da PoC, além de facilitar a integração entre os módulos do sistema. Esse recurso também reduz inconsistências entre implementação e documentação técnica, problema recorrente em aplicações distribuídas.

6. Conclusão

O presente trabalho apresentou o desenvolvimento do QuestBook, um sistema de recuperação semântica de questões a partir de materiais didáticos em formato PDF, fundamentado em técnicas de Processamento de Linguagem Natural e representações vetoriais. O sistema foi concebido como uma Prova de Conceito com o objetivo de demonstrar a viabilidade técnica da associação automática entre conteúdos estudados e

questões semanticamente relacionadas, reduzindo o esforço manual tradicionalmente exigido nesse processo.

A arquitetura proposta integrou componentes de extração textual, geração de *embeddings* semânticos, busca vetorial por similaridade e interpretação de intenção via Modelos de Linguagem de Grande Escala, organizados em uma estrutura modular de três camadas. A estratégia de persistência híbrida — combinando bancos relacionais (MySQL e PostgreSQL) com banco vetorial (ChromaDB) — mostrou-se adequada para atender simultaneamente aos requisitos de armazenamento estruturado e de recuperação semântica eficiente.

Os resultados obtidos indicam que o sistema foi capaz de recuperar questões contextualmente relevantes mesmo na ausência de correspondência lexical direta entre a consulta do usuário e os enunciados das questões, validando a hipótese central do trabalho quanto à eficácia da abordagem vetorial frente a métodos tradicionais baseados em palavras-chave. Todos os requisitos funcionais definidos na fase de modelagem foram implementados e validados, incluindo o mecanismo de curadoria humana com trilha de auditoria, a filtragem por banca organizadora e o sistema de anti-repetição histórica.

A estratégia de validação, conduzida com a participação de professores especialistas atuando como curadores, permitiu avaliar qualitativamente a relevância das questões recuperadas e verificar a eficácia da trava de confiança adotada. Os testes automatizados, organizados em camadas unitária e de integração, contribuíram para assegurar a robustez dos componentes de interpretação de intenção e do pipeline de recuperação.

Entre as principais limitações identificadas, destaca-se a restrição do modelo de **embeddings** ao processamento de sequências de até 128 tokens, o que pode comprometer a representação semântica de enunciados mais extensos. Além disso, o sistema atualmente processa apenas documentos PDF textuais gerados digitalmente, não contemplando documentos digitalizados por imagem. A persistência do histórico de conversas exclusivamente no navegador do usuário também representa uma limitação em termos de portabilidade entre dispositivos.

Como trabalhos futuros, propõe-se a adoção de modelos de **embeddings** com maior capacidade de contexto, a implementação de mecanismos de Reconhecimento Óptico de Caracteres (OCR) para ampliação dos tipos de documentos suportados, a sincronização do histórico de sessões entre dispositivos por meio de persistência no **backend**, e a realização de avaliações quantitativas com métricas formais de recuperação de informação, como **precision@k** e **recall@k**, aplicadas a um conjunto de dados anotado. Adicionalmente, a expansão da base de questões para outros domínios do conhecimento

e a incorporação de questões discursivas constituem direções relevantes para a evolução do sistema.

Referências

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. 3. ed. Upper Saddle River: Addison-Wesley, 2013.

BOMMASANI, R. et al. **On the Opportunities and Risks of Foundation Models**. arXiv, 2021. Disponível em: https://arxiv.org/abs/2108.07258. Acesso em: 16 dez. 2025.

BROWN, T. B. et al. Language Models are Few-Shot Learners. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2020. **Proceedings** [...]. [S.l.: s.n.], 2020.

CHROMA. **Chroma**: the AI-native open-source embedding database. 2023. Disponível em: https://docs.trychroma.com/. Acesso em: 16 dez. 2025.

DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *In*: PROCEEDINGS OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES (NAACL-HLT), 2019. **Proceedings** [...]. [S.l.: s.n.], 2019.

GOLDBERG, Y. **Neural Network Methods for Natural Language Processing**. [S.l.]: Morgan & Claypool Publishers, 2017.

GROQ INC. **Groq API Documentation: Pricing**. 2026. Disponível em: https://console.groq.com/docs/pricing. Acesso em: 20 maio 2026.

GROQ INC. **Groq Console: Rate Limits**. 2026. Disponível em: https://console.groq.com/settings/limits. Acesso em: 20 maio 2026.

GROQ INC. **Groq: Fast AI Inference**. 2024. Disponível em: https://groq.com/. Acesso em: 16 dez. 2025.

JOHNSON, J.; DOUZE, M.; JÉGOU, H. Billion-scale similarity search with GPUs. **IEEE Transactions on Big Data**, v. 7, n. 3, p. 535–547, 2019.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. 3. ed. Stanford: Stanford University, 2020.

KOJIMA, T. et al. **Large Language Models are Zero-Shot Reasoners**. arXiv, 2022.

Disponível em: https://arxiv.org/abs/2205.11916. Acesso em: 16 dez. 2025.

MANNING, C.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to Information Retrieval**. Cambridge: Cambridge University Press, 2008.

META AI. **Llama 3.3**: Open Foundation Large Language Models. 2024. Disponível em: https://ai.meta.com/blog/llama-3-3/. Acesso em: 20 maio 2026.

META PLATFORMS. **React**: the library for web and native user interfaces. 2013. Disponível em: https://react.dev/. Acesso em: 16 dez. 2025.

RAMÍREZ, S. **FastAPI**: web framework for building APIs with Python. 2018. Disponível em: https://fastapi.tiangolo.com/. Acesso em: 16 dez. 2025.

REIMERS, N.; GUREVYCH, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *In*: CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), 2019, Hong Kong. **Proceedings** [...]. Hong Kong: Association for Computational Linguistics, 2019. p. 3982–3992.

VASWANI, A. et al. Attention Is All You Need. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS), 2017. **Proceedings** [...]. [S.l.: s.n.], 2017.

WEI, J. et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2022. **Proceedings** [...]. [S.l.: s.n.], 2022. Disponível em: https://arxiv.org/abs/2201.11903. Acesso em: 16 dez. 2025.

WU, Y. et al. **Google's Neural Machine Translation System**: Bridging the Gap between Human and Machine Translation. arXiv preprint arXiv:1609.08144, 2016.

ZHANG, J. et al. **Vector Database Management Systems**: Concepts and Applications. IEEE, 2023.

APÊNDICE A - Análise de Custos Operacionais dos Componentes de Inteligência Artificial

Uma dimensão frequentemente negligenciada no desenvolvimento de sistemas baseados em Inteligência Artificial diz respeito aos custos operacionais associados à inferência dos modelos. No contexto do QuestBook, esta análise é particularmente relevante porque o sistema integra dois componentes de IA com perfis de custo fundamentalmente distintos: um modelo de embeddings executado localmente e um Large Language Model (LLM) consumido via API remota. A presente seção detalha o

consumo de recursos computacionais e financeiros de cada componente, apresentando estimativas para cenários de uso representativos.

A.1 Modelo de Embeddings: Execução Local e Custo Zero

O modelo paraphrase-multilingual-MiniLM-L12-v2, utilizado tanto no pipeline de indexação quanto no mecanismo de recuperação semântica em tempo real, opera integralmente em infraestrutura local, sem dependência de APIs externas.

Trata-se de um modelo *open source* distribuído sob licença Apache 2.0 [Reimers e Gurevych 2019], contendo aproximadamente 118 milhões de parâmetros e capacidade de execução eficiente em CPUs convencionais, dispensando o uso de hardware especializado, como GPUs. Essa característica reduz custos operacionais e simplifica a implantação da solução em ambientes computacionais de menor capacidade.

O modelo utiliza tokenização baseada em WordPiece [Wu et al. 2016], derivada da arquitetura BERT, segmentando o texto em subpalavras para representação semântica contextual.

Nos testes realizados com textos em Língua Portuguesa, observou-se taxa média aproximada de 1,3 tokens por palavra, resultado favorecido pela presença de morfemas frequentes do português no vocabulário multilíngue do modelo. Essa característica reduz fragmentação excessiva durante o processo de tokenização.

O limite máximo de entrada suportado pelo modelo corresponde a 128 tokens por sequência textual. Textos que excedem esse limite são truncados automaticamente durante o processo de tokenização.

Para o contexto de questões de concursos públicos analisado neste trabalho, esse limite mostrou-se adequado, uma vez que o conteúdo semanticamente mais relevante tende a concentrar-se nas sentenças iniciais dos enunciados avaliados.

O custo computacional deste componente divide-se em duas operações distintas:

- Indexação (offline): A geração de embeddings para a base completa de aproximadamente 15.000 questões constitui uma operação executada uma única vez. Em um processador de uso geral (CPU x86-64, 2 vCPUs compartilhados), o tempo de processamento observado situou-se entre 3 e 8 minutos, com o modelo processando lotes de 1.000 questões sequencialmente. O resultado é persistido no ChromaDB em modo PersistentClient, sobrevivendo a reinicializações do servidor sem necessidade de reprocessamento.
- Busca em tempo real (online): Cada consulta do usuário requer a geração de um único vetor de 384 dimensões, operação que demanda menos de 50 milissegundos em CPU. O modelo permanece carregado em memória (aproximadamente 450 MB de RAM) durante todo o ciclo de vida do servidor, eliminando a latência de carregamento a cada requisição —

estratégia implementada por meio da instanciação singleton da classe QuestSearchEngine no módulo de rotas.

O Quadro 1 sintetiza o perfil de recursos do modelo de embeddings.

Quadro 1 — Perfil de recursos do modelo paraphrase-multilingual-MiniLM-L12-v2.

Característica	Valor
Parâmetros	~118 milhões
Dimensão do embedding	384
Limite de entrada	128 tokens (~100 palavras)
Ocupação em memória RAM	~450 MB
Latência por consulta (CPU)	< 50 ms
Licença	Apache 2.0 (uso livre)
Custo operacional	R$ 0,00

A.2 IntentParser e API da Groq: Consumo de Tokens e Precificação

O segundo componente de Inteligência Artificial do sistema, denominado IntentParser, é responsável pela interpretação semântica das consultas realizadas pelos usuários.

Diferentemente do mecanismo de embeddings executado localmente, esse componente depende da plataforma Groq para execução do modelo Llama 3.3 70B Versatile [Meta AI 2024], processado em hardware especializado baseado em LPUs (*Language Processing Units*).

A utilização de inferência externa foi motivada pela necessidade de reduzir latência e ampliar a capacidade de interpretação contextual das consultas em linguagem natural. Entretanto, essa abordagem introduz dependência de infraestrutura terceirizada e custos operacionais proporcionais ao volume de tokens processados.

Essa separação arquitetural permitiu manter localmente os componentes de recuperação semântica de maior volume computacional, enquanto o processamento de intenção foi delegado ao modelo de linguagem de grande escala hospedado externamente.

A precificação da API da Groq, vigente em maio de 2026, segue o modelo de cobrança por milhão de tokens, conforme o Quadro 2.

Quadro 2 — Precificação da API Groq para o modelo Llama 3.3-70b-versatile.

Tipo de token	Custo por 1 milhão de tokens (USD)

Tokens de entrada (input)	US$ 0,59
Tokens de saída (output)	US$ 0,79

Fonte: Groq Inc. (2026). Disponível em: https://console.groq.com/docs/pricing.

Para estimar o consumo por requisição, faz-se necessário decompor a estrutura do prompt enviado à API. O IntentParser constrói uma mensagem composta por quatro componentes de tamanho variável, conforme detalhado na tabela 5.

Tabela 5 — Decomposição do consumo de tokens por requisição ao IntentParser.

Componente do prompt	Tokens estimados	Descrição
Prompt do sistema (regras + few-shot)	500–600	Instruções fixas, regras de segurança e exemplos de entrada/saída
Histórico conversacional	50–150	Últimas 3 mensagens da sessão (memória volátil)
Contexto documental (quando presente)	3.000–5.000	Trecho de ~20.000 caracteres do PDF do usuário
Mensagem do usuário	15–40	Texto livre digitado pelo aluno
Total de entrada (sem documento)	~600–800	Cenário mais frequente
Total de entrada (com documento)	~3.600–5.800	Quando o aluno referencia um capítulo do PDF
Total de saída (JSON gerado)	~40–80	Objeto JSON com topic, limit, search_query, banca

A saída do IntentParser é deliberadamente compacta — um objeto JSON com no máximo quatro campos — o que mantém o consumo de tokens de saída consistentemente abaixo de 80 tokens por requisição, independentemente da complexidade da entrada.

A.3 Projeção de Custos para Cenários de Uso

Com base nos dados de consumo apresentados, foram calculadas projeções de custo para três cenários representativos de uso do sistema, variando o número de usuários simultâneos e a frequência de interações. Os cálculos consideram a fórmula:

$C = (T\_input \times P\_input + T\_output \times P\_output) \times N$, onde C é o custo total, T_input e T_output são os tokens médios por requisição (entrada e saída, respectivamente), P_input e P_output são os preços por token, e N é o número total de requisições no período.

Tabela 6 — Projeção de custos mensais da API Groq para cenários de uso representativos.

Cenário	Usuários	Consultas/mês	Tokens entrada/req.	Custo mensal (USD)	Custo mensal (BRL)[1]
A: Validação (PoC)	30	300	~700 (sem PDF)	US$ 0,14	~R$ 0,77
B: Pesquisa de campo	100	1.000	~700 (sem PDF)	US$ 0,46	~R$ 2,53
C: Uso misto com PDFs	100	1.000	~2.500 (misto)	US$ 1,52	~R$ 8,36
D: Escala moderada	500	10.000	~2.500 (misto)	US$ 15,22	~R$ 83,71

[1] Câmbio de referência: 1 USD = 5,50 BRL (maio/2026).

Os resultados evidenciam que, mesmo no cenário D, que extrapola significativamente o escopo de uma PoC acadêmica, o custo mensal da API de LLM permanece na ordem de dezenas de reais, valor inferior ao de serviços de infraestrutura básica como domínio web ou hospedagem

A.4 Plano Gratuito da Groq e Viabilidade para Validação Acadêmica

Um fator adicional de viabilidade econômica reside na disponibilidade de um plano gratuito oferecido pela plataforma Groq, que dispensa a vinculação de cartão de crédito e impõe limites de uso por organização, conforme sintetizado no Quadro 3.

Quadro 3 — Limites do plano gratuito da API Groq (maio/2026).

Limite	Valor
Requisições por minuto (RPM)	~30
Requisições por dia (RPD)	~1.000 (modelo Llama 3.3-70b)
Tokens por minuto (TPM)	~6.000–30.000 (variável por modelo)

Fonte: Groq Inc. (2026). Disponível em: https://console.groq.com/settings/limits.

O limite de 1.000 requisições por dia comporta com folga os cenários A e B descritos no Quadro 4, que representam os volumes de uso esperados durante a fase de

validação do presente trabalho. Desta forma, o custo efetivo com a API de LLM durante o período de pesquisa é nulo.

A.5 Síntese Comparativa e Discussão

Tabela 7 — Mapa de custos operacionais dos componentes de IA do QuestBook.

Componente	Tecnologia	Execução	Custo operacional
Geração de embeddings (indexação)	SentenceTransformer (MiniLM)	Local (CPU)	R$ 0,00
Geração de embeddings (busca)	SentenceTransformer (MiniLM)	Local (CPU)	R$ 0,00
Armazenamento vetorial	ChromaDB (PersistentClient)	Local (disco)	R$ 0,00
Interpretação de intenção	Llama 3.3-70b via Groq API	Remoto (API)	R$ 0,00 (free tier)

A análise demonstra que a arquitetura adotada no QuestBook viabiliza a operação de um sistema de busca semântica baseado em IA com custo operacional efetivamente nulo para o contexto de validação acadêmica. Este resultado decorre de duas decisões arquiteturais deliberadas: (i) a adoção de um modelo de embeddings leve e open source, executado localmente sem dependência de APIs pagas; e (ii) a utilização da API da Groq exclusivamente para a tarefa de parsing de intenção — operação que gera um volume reduzido de tokens de saída (~60 tokens por requisição), mantendo o consumo dentro dos limites do plano gratuito.

Em um cenário hipotético de evolução para produção com escala moderada (500 usuários, 10.000 consultas mensais), o custo projetado com IA seria de aproximadamente R$ 84/mês — valor que permanece marginal quando comparado aos custos de infraestrutura de hospedagem. A principal decisão de escalabilidade futura envolveria avaliar a substituição do modelo de embeddings local por uma API de embeddings gerenciada (e.g., OpenAI text-embedding-3-small, ao custo de US$ 0,02 por milhão de tokens), o que eliminaria a necessidade de manter o modelo em memória no servidor, ao custo de uma dependência externa adicional e latência de rede na operação de busca.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Restrito

Versão Final do TCC - Álisson Brener da Silva

Assunto:	Versão Final do TCC - Álisson Brener da Silva
Assinado por:	Álisson Silva
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Restrito
Hipótese Legal:	Direito Autoral (Art. 24, III, da Lei no 9.610/1998)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Álisson Brener da Silva, DISCENTE (202121250009) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 27/07/2026 10:45:04.

Este documento foi armazenado no SUAP em 27/07/2026. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1925968
Código de Autenticação: b707cafe5b