

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

EDUARDO CHAGAS PEREIRA
LUCAS MENDUIÑA RAMOS EVANGELISTA

**Prometheus: Uma Plataforma de
Ensino Supervisionada por Inteligência
Artificial**

Prof. Filipe Braidão do Carmo, D.Sc.
Orientador

Nova Iguaçu, Julho de 2026

Prometheus: Uma Plataforma de Ensino Supervisionada por Inteligência Artificial

Eduardo Chagas Pereira

Lucas Menduiña Ramos Evangelista

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

Eduardo Chagas Pereira

Lucas Menduiña Ramos Evangelista

Aprovado por:

Prof. Filipe Braida do Carmo, D.Sc.

Prof. Natália Chaves Lessa, D.Sc.

Paulo Roberto Xavier Júnior, M.Sc

NOVA IGUAÇU, RJ - BRASIL

Julho de 2026

Agradecimentos

Eduardo Chagas Pereira

Gostaria de agradecer, primeiramente, à minha família, em especial à minha mãe, Andrea Chagas, que sempre acreditou em mim e enxergou meu potencial, mesmo quando eu próprio duvidei. Sinto que concluir esta etapa é uma forma de provar, depois de todo esse tempo, que ela sempre esteve certa. Eu te amo até o infinito, e um pouco mais.

Agradeço aos meus amigos, tanto os que caminham comigo desde antes da graduação quanto aqueles que a universidade colocou em meu caminho. Aos amigos da faculdade: Augusto Belarmino, Erick Azevedo, Pedro Calbo, Paulo Henrique e Lucas Wong, deixo um agradecimento especial pelos inúmeros momentos de estudo, pelas conversas, pelas risadas e pelo companheirismo ao longo desses anos. Aos amigos de fora da universidade, agradeço por permanecerem presentes, oferecendo apoio, incentivo e me lembrando da importância de manter o equilíbrio durante essa jornada.

Agradeço, de forma profunda e especial, ao Lucas Menduiña e ao Victor Miguez, dois grandes presentes que a graduação me deu e que levarei para a vida. Ao Lucas, meu primeiro amigo da faculdade e dupla neste trabalho: nossa parceria acadêmica rapidamente se transformou em irmandade. Compartilhamos incertezas, desafios e sonhos, e não haveria forma mais correta de encerrar este ciclo do que realizar juntos este último projeto.

Ao Victor, obrigado por me mostrar um lado mais humano da vida e lembrar que a graduação não se resume apenas a cobranças, prazos e resultados. Obrigado por

sempre querer me ver no meu melhor, por estender a mão quando eu mais precisei, por ter disposição para ajudar mesmo nos momentos difíceis e por me oferecer novas perspectivas. Sua presença me ensinou a viver esta etapa de uma forma mais especial, e sou profundamente grato por poder compartilhar esta conquista com você.

Agradeço ao meu orientador, Filipe Braidá, pela orientação durante este trabalho e pelos ensinamentos compartilhados ao longo da graduação. Além de tudo, um excelente professor, que sempre esteve disposto a oferecer conselhos e incentivar meu crescimento acadêmico e profissional.

Por fim, agradeço a mim mesmo por não ter desistido diante das dificuldades, por ter sustentado as madrugadas à base de uma quantidade industrial de café e por acreditar no meu potencial mesmo nos momentos de maior incerteza. Seguir em frente até a conclusão desta etapa foi a minha maior prova de resistência, e tenho muito orgulho de ter chegado até aqui.

"I don't know where I'm going from here, but I promise it won't be boring."

— David Bowie

Lucas Menduiña Ramos Evangelista

Primeiramente, gostaria de agradecer a Deus, pois, sem Ele, me faltariam resiliência e coragem para seguir com a mente firme, confiante de que dias melhores chegariam. Com Ele, fui capaz de abrir meus olhos para enxergar aqueles que torcem por mim e confiam em mim. Também fui capaz de dar orgulho àqueles que me amam e de corresponder à fé que, durante anos, depositaram em meu futuro.

Agradeço aos meus pais, meus maiores fãs, que sempre me deram a oportunidade e as condições necessárias para estudar e buscar meus maiores sonhos. Devo um agradecimento especial à minha mãe, Valéria Menduiña, que me apoiou nos momentos mais difíceis, me consolou nos momentos mais tristes e comemorou minhas conquistas como se fossem suas. Não posso deixar de mencionar minha avó, Amélia Menduiña, que, embora tenha partido há alguns anos, continua presente em minha memória. Sua história de luta, amor e dedicação me motiva a cada dia. Tenho muito orgulho do nosso sobrenome e de tudo aquilo que ele representa: a história, a força e o amor de cada pessoa que o carregou e o carrega.

À minha namorada, Juliana Cordeiro, que, nos momentos de aflição, angústia e ansiedade, me acolheu e se tornou minha segunda casa. Com suas palavras de amor, carinho e respeito, sempre buscou incentivar meu crescimento e me tornar mais forte. Nos momentos difíceis, às vezes prevaleceu a razão; em outros, a emoção. Mas, no fim, reconheço que foi Deus quem guiou nossas escolhas até aqui.

Agradeço também aos meus amigos, que sempre transformaram o ambiente exaustivo e rotineiro de estudo em um espaço mais leve e descontraído: Augusto Belarmino, Erick Azevedo, Eduardo Chagas, Lucas Wong, Pedro Calbo, Paulo Henrique e Victor Miguez. Um agradecimento especial ao Eduardo Chagas, que, durante esta caminhada, tornou-se mais que um amigo: um irmão. Dividimos angústias, problemas e sonhos, e ele se tornou um parceiro não apenas dentro da faculdade, mas também na vida. Eduardo foi meu primeiro amigo na faculdade; por isso, nada mais justo do que realizar com ele nosso último projeto acadêmico.

Meus sinceros agradecimentos ao professor Filipe Braidá, pela orientação, pelos

ensinamentos e pela preocupação demonstrada durante o curso. Suas correções e conversas foram muito importantes para o meu crescimento profissional, fazendo dele um exemplo para mim.

Por fim, agradeço a mim mesmo pelas noites mal dormidas, pelas madrugadas de estudo, pelas horas dedicadas aos livros e também pelo tempo passado dentro do transporte público, sempre em busca de tornar esse sonho realidade.

“I’m still standing.”

— Elton John

*“Prometeu, ladrão da luz, doador da luz, preso pelos deuses
devia ter sido um **livro**.”*

— Mark Z. Danielewski, *Casa de Folhas*

RESUMO

Prometheus: Uma Plataforma de Ensino Supervisionada por Inteligência Artificial

Eduardo Chagas Pereira e Lucas Menduiña Ramos Evangelista

Julho/2026

Orientador: Filipe Braidão do Carmo, D.Sc.

Estudar materiais extensos e tecnicamente densos exige mais do que leitura linear: o estudante precisa organizar o conteúdo, praticar a recuperação ativa e reconhecer quais tópicos ainda demandam revisão. No entanto, esse acompanhamento costuma depender de autoavaliações subjetivas, especialmente quando o material está distribuído em documentos longos, como arquivos em *Portable Document Format* (PDF). Diante desse cenário, este trabalho apresenta o Prometheus, uma plataforma web voltada à avaliação inteligente de conhecimento e ao apoio ao estudo supervisionado. A solução permite que o usuário crie bases de conhecimento a partir de documentos, extraia e organize seus conteúdos, gere resumos, tópicos, simulados personalizados e *flashcards*, além de acompanhar indicadores de desempenho. Do ponto de vista técnico, o sistema combina processamento assíncrono, persistência relacional, indexação vetorial, busca semântica e técnicas de *Retrieval-Augmented Generation* (RAG) para produzir materiais de estudo contextualizados a partir do conteúdo enviado. Com isso, o Prometheus busca transformar o estudo passivo em um ciclo contínuo de síntese, prática, avaliação e *feedback*, oferecendo ao estudante recursos mais objetivos para identificar lacunas de conhecimento e orientar sua revisão.

Palavras-chave: Autoavaliação. Busca semântica. Simulados personalizados. Flashcards. Recuperação aumentada por geração.

ABSTRACT

Prometheus: Uma Plataforma de Ensino Supervisionada por Inteligência Artificial

Eduardo Chagas Pereira and Lucas Menduiña Ramos Evangelista

Julho/2026

Advisor: Filipe Braidão do Carmo, D.Sc.

Studying extensive and technically dense materials requires more than linear reading: students need to organize content, practice active recall, and recognize which topics still require review. However, this monitoring process often depends on subjective self-assessment, especially when the study material is distributed across long documents, such as PDF files. In this context, this work presents Prometheus, a web platform focused on intelligent knowledge assessment and supervised study support. The solution enables users to create knowledge bases from documents, extract and organize their content, generate summaries, topics, personalized mock exams, and flashcards, as well as monitor performance indicators. From a technical perspective, the system combines asynchronous processing, relational persistence, vector indexing, semantic search, and RAG techniques to produce contextualized study materials based on the submitted content. Thus, Prometheus aims to transform passive study into a continuous cycle of synthesis, practice, assessment, and feedback, providing students with more objective resources to identify knowledge gaps and guide their review.

Keywords: *Self-assessment. Semantic search. Personalized mock exams. Flashcards. Retrieval-augmented generation.*

Lista de Figuras

3.1	Fluxo macro do processo de estudo implementado pelo Prometheus.	13
3.2	Diagrama de Casos de Uso do sistema Prometheus	20
3.3	Fluxo de processamento paralelo e consolidação do resumo.	23
3.4	Diagrama Entidade-Relacionamento do sistema Prometheus.	27
3.5	Página de visão geral da plataforma Examica, com destaque para recursos de estudo, acompanhamento de progresso e ações adicionais.	33
3.6	Interface do NotebookLM demonstrando a centralização de múltiplas fontes de estudo e a geração automatizada de respostas baseada no contexto enviado.	34
3.7	Exemplo de cartão de resposta no Anki, em que o termo em inglês aparece na parte superior e sua tradução em alemão logo abaixo. A interface apresenta ainda opções de avaliação da dificuldade da lembrança ('Again', 'Good' e 'Easy'), permitindo ao usuário controlar o intervalo de repetição do conteúdo de acordo com seu desempenho.	35
4.1	Estrutura de diretórios da aplicação web, organizada em módulos. A disposição evidencia a separação de responsabilidades e a modularidade do sistema.	43
4.2	Tela inicial do Prometheus com visão geral das bases, simulados e indicadores de acompanhamento.	45

4.3	Listagem das bases de conhecimento cadastradas e seus respectivos recursos associados.	46
4.4	Tela de detalhes de uma base de conhecimento com resumo, indicadores e tópicos principais.	47
4.5	Aba de baralhos de uma base de conhecimento com opção de criação automática ou manual de <i>deck</i>	48
4.6	Estados da interface de <i>flashcards</i> : pergunta, resposta revelada, classificação por dificuldade e criação manual de cartão.	49
4.7	Aba de simulados com criação de nova avaliação, estado de conclusão, progresso e resultado.	50
4.8	Tela de correção de simulado com indicação da alternativa incorreta, resposta correta e explicação do gabarito.	51
B.1	Diagrama de sequência referente à criação das <i>knowledges</i>	81
B.2	Diagrama de sequência referente à criação dos <i>topics</i>	82
B.3	Diagrama de sequência referente à criação das <i>questions</i> e <i>answers</i>	82

Lista de Tabelas

3.1	Estrutura da Tabela <code>users</code>	28
3.2	Tabela <code>knowledges</code>	28
3.3	Tabela <code>pages</code>	29
3.4	Tabela <code>knowledge_chunks</code>	29
3.5	Tabela <code>topics</code>	30
3.6	Tabela <code>user_favorite_knowledges</code>	30
3.7	Tabela <code>exams</code>	31
3.8	Tabela <code>questions</code>	31
3.9	Tabela <code>answers</code>	31
3.10	Tabela <code>decks</code>	32
3.11	Tabela <code>flashcards</code>	32

Lista de Abreviaturas e Siglas

ACID *Atomicity, Consistency, Isolation, Durability*

API *Application Programming Interface*

DER Diagrama Entidade-Relacionamento

FSRS *Free Spaced Repetition Scheduler*

HTTP *Hypertext Transfer Protocol*

IA Inteligência Artificial

LLM *Large Language Model*

MVC *Model-View-Controller*

PDF *Portable Document Format*

RAG *Retrieval-Augmented Generation*

RF Requisitos Funcionais

RN Regras de Negócio

SPA *Single Page Application*

SRS Sistema de Repetição Espaçada

Sumário

Agradecimentos	i
Resumo	vi
Abstract	vii
Lista de Figuras	viii
Lista de Tabelas	x
Lista de Abreviaturas e Siglas	xi
1 Introdução	1
1.1 Objetivo	2
1.2 Uso de Inteligência Artificial	3
1.3 Organização do Trabalho	4
2 <i>Retrieval-Augmented Generation</i>	5
2.1 Modelos de Linguagem de Grande Escala	5
2.2 Limitações dos Modelos de Linguagem	6
2.3 <i>Retrieval-Augmented Generation</i>	8

2.4	Qualidade da Recuperação em Sistemas RAG	9
3	Prometheus	11
3.1	Mini-mundo e Visão Geral do Sistema	12
3.2	Requisitos do Sistema	14
3.2.1	Requisitos funcionais	14
3.2.2	Regras de Negócio	18
3.2.3	Casos de Uso	19
3.3	<i>Workflow</i> de Criação da Base de Conhecimento	20
3.4	Processamento do Documento e Geração do Resumo	22
3.5	Tópicos Centrais e RAG Orientado por Tópicos	23
3.6	Arquitetura Operacional e Limitações	25
3.7	Modelagem de Dados	26
3.7.1	Entidades	27
3.8	Trabalhos relacionados	32
4	Implementação e Resultado do Sistema	36
4.1	Da Proposta à Implementação	37
4.2	Base Técnica e Organização da Aplicação	37
4.2.1	Fundação do Projeto e Produtividade Inicial	38
4.2.2	Servidor, Regras de Negócio e Persistência Operacional	38
4.2.3	Integração entre Servidor e Interface	39
4.3	Processamento, Dados e Geração de Conteúdo	40
4.3.1	Persistência Relacional e Busca Semântica	41

4.3.2	Orquestração do <i>Pipeline</i> de Inteligência Artificial	41
4.3.3	Modularização por Funcionalidade	42
4.4	Interface e Resultado do Sistema	44
4.4.1	Construção Visual e Componentização	44
4.4.2	Fluxo de Uso Implementado	44
4.4.3	Resultado Obtido	51
5	Conclusão	53
5.1	Considerações finais	53
5.2	Limitações e trabalhos futuros	55
	Referências	57
A	Especificação de Casos de Uso	58
B	Diagramas de Sequência	81

Capítulo 1

Introdução

O avanço das tecnologias digitais tem ampliado significativamente o acesso à informação e aos materiais de estudo, especialmente em ambientes acadêmicos. Livros técnicos, documentações, artigos científicos e materiais didáticos digitais podem ser consultados com facilidade cada vez maior. No entanto, a simples disponibilidade desses conteúdos não garante, por si só, um processo de aprendizagem efetivo. Em áreas científicas e tecnológicas, nas quais o estudante precisa lidar com conceitos abstratos, linguagem especializada e materiais extensos, torna-se frequente a dificuldade de transformar o acesso à informação em estudo orientado, mensurável e progressivo.

Nesse contexto, o processo de aprendizagem autônoma tende a tornar-se subjetivo, pois depende fortemente da percepção individual do estudante sobre seu desempenho. O estudante pode concluir a leitura de um material sem saber, com clareza, quais tópicos foram realmente compreendidos, quais conteúdos ainda apresentam lacunas e que tipo de revisão deve ser priorizado. Estudos na área de educação e aprendizagem autorregulada apontam que a ausência de mecanismos de avaliação contínua e *feedback* estruturado dificulta o desenvolvimento de estratégias eficazes de estudo e autoaperfeiçoamento (CRUZ; FERENC, 2025).

Esse cenário evidencia uma lacuna prática: ainda que existam diversas ferramentas para armazenar, consultar ou resumir materiais, nem sempre elas oferecem um

acompanhamento supervisionado do aprendizado, capaz de relacionar o conteúdo estudado ao desempenho do usuário. Para enfrentar esse problema, torna-se relevante investigar soluções que combinem bases de conhecimento fornecidas pelo próprio estudante com mecanismos de avaliação, geração de questões, análise de respostas, indicação de desempenho e apoio à revisão. A ideia central é permitir que o usuário saiba o que está aprendendo, o que ainda precisa estudar e como pode organizar melhor seu processo de aprendizagem.

Com o crescimento do uso de tecnologias baseadas em Inteligência Artificial (IA), surgem novas possibilidades para apoiar esse tipo de acompanhamento educacional. Modelos de linguagem podem auxiliar na interpretação de materiais textuais, na geração de perguntas contextualizadas, na produção de explicações e na elaboração de recursos de revisão, desde que utilizados de forma controlada e alinhada ao conteúdo fornecido. Lan e Zhou (2025) já mostraram que a aplicação equilibrada de IA no contexto educacional pode funcionar como um importante auxílio no estudo sem comprometer a autoeficácia dos estudantes.

1.1 Objetivo

Este trabalho propõe o desenvolvimento de um sistema *web* de ensino supervisionado, apoiado por IA, voltado para o acompanhamento do aprendizado de um usuário a partir de uma ou mais bases de conhecimento fornecidas. A plataforma deverá permitir que o estudante importe materiais de estudo, organize esses conteúdos em bases de conhecimento e utilize o sistema para gerar avaliações alinhadas ao material selecionado.

Como parte do processo avaliativo, o sistema deverá criar questões sobre todo o conteúdo ou sobre seções específicas, de acordo com a necessidade do usuário. Após a submissão das respostas, a plataforma deverá analisar o desempenho obtido, apresentar explicações sobre as alternativas corretas e fornecer *feedback* que auxilie o estudante a refletir sobre seu progresso, reconhecer pontos de domínio e identificar lacunas de conhecimento. Dessa forma, busca-se reduzir a dependência exclusiva

da percepção subjetiva do usuário e oferecer indicadores mais concretos sobre sua aprendizagem.

Além do processo avaliativo, o sistema também deverá oferecer suporte ao uso de *flashcards* como estratégia ativa de estudo, possibilitando tanto a criação automática de cartões a partir do conteúdo estudado quanto a elaboração de *flashcards* personalizados pelo próprio estudante. Com isso, o trabalho pretende integrar avaliação, *feedback* e revisão em um mesmo ambiente, favorecendo um ciclo contínuo de estudo supervisionado.

1.2 Uso de Inteligência Artificial

É importante mencionar que durante o desenvolvimento deste trabalho, ferramentas baseadas em IA foram utilizadas como apoio ao processo de pesquisa, planejamento e escrita acadêmica. Esse uso ocorreu principalmente na organização de ideias, no auxílio à busca e seleção inicial da literatura relacionada, na comparação de alternativas para o projeto e no refinamento de decisões de concepção e *design* do sistema. Assim, a utilização dessas ferramentas esteve associada ao processo de elaboração do trabalho e de tomada de decisão, não à substituição da análise crítica necessária à pesquisa.

Também foram empregadas ferramentas de IA como apoio ao processo de redação, especialmente para revisar clareza, coesão, estrutura textual e adequação da escrita acadêmica. No entanto, todo conteúdo sugerido por essas ferramentas foi analisado, revisado e adaptado pelos autores, de modo a garantir coerência com os objetivos do trabalho, precisão conceitual e alinhamento com as fontes utilizadas. Dessa forma, a IA atuou como instrumento auxiliar, enquanto as decisões finais sobre conteúdo, interpretação, projeto, desenvolvimento técnico e escrita permaneceram sob responsabilidade dos autores.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

- **Capítulo 2:** Estabelece a fundamentação teórica ao detalhar conceitos relacionados a *Large Language Models* (LLMs) e RAG.
- **Capítulo 3:** Detalha a concepção do sistema proposto, abordando a modelagem dos dados, a arquitetura, as funcionalidades previstas e a comparação com trabalhos relacionados.
- **Capítulo 4:** Descreve o desenvolvimento do sistema, detalhando as tecnologias utilizadas, a implementação das funcionalidades, a interface e os principais aspectos da solução construída.
- **Capítulo 5:** Apresentam-se as considerações finais do trabalho, suas limitações e a indicação de possíveis direções para pesquisas e trabalhos futuros.

Capítulo 2

Retrieval-Augmented Generation

Este capítulo apresenta os conceitos necessários para compreender a proposta deste trabalho. A exposição parte de uma explicação introdutória sobre LLMs, enfatizando o que esses modelos fazem, como recebem instruções e por que se tornaram relevantes em aplicações de linguagem natural. Em seguida, são discutidas suas limitações práticas, especialmente alucinação, dificuldade de atribuição de fontes e desatualização do conhecimento. Por fim, apresenta-se o RAG como uma estratégia para conectar modelos generativos a uma memória externa recuperável, aproximando a geração textual dos documentos que fundamentam a resposta.

2.1 Modelos de Linguagem de Grande Escala

Um modelo de linguagem é um sistema treinado para estimar padrões em sequências de texto. Em termos práticos, ele recebe uma entrada textual e produz uma continuação provável para essa entrada. Quando esse tipo de modelo é treinado com grandes volumes de dados e possui grande quantidade de parâmetros, passa a ser chamado de *Large Language Model* (LLM). Assim, um LLM não deve ser entendido como uma base de dados ou como um mecanismo de consulta em tempo real, mas como uma representação estatística de padrões linguísticos aprendidos durante o treinamento.

O avanço recente dos LLMs está fortemente associado à arquitetura Transformer. Diferentemente de arquiteturas recorrentes, o Transformer utiliza mecanismos de atenção para relacionar diferentes posições de uma sequência, permitindo que o modelo observe dependências entre palavras distantes e processe os elementos de forma mais paralelizável. Essa arquitetura foi proposta por Vaswani et al. (2017) e se tornou a base de diversos modelos modernos de linguagem.

Para interagir com um LLM, o usuário envia um *prompt*. O *prompt* é a instrução, pergunta ou tarefa fornecida ao modelo, podendo incluir também exemplos, restrições de formato e informações de contexto. Antes de ser processado, esse texto é dividido em unidades menores chamadas *tokens*, que podem corresponder a palavras, partes de palavras ou caracteres. Esses *tokens* são convertidos em representações numéricas, chamadas *embeddings*, que permitem ao modelo operar matematicamente sobre o texto.

A resposta gerada por um LLM é produzida de forma sequencial. A cada passo, o modelo estima qual próximo *token* é mais provável considerando os *tokens* já observados no *prompt* e na própria resposta em construção. Esse comportamento explica por que esses modelos conseguem produzir textos coerentes, resumir conteúdos, responder perguntas e seguir instruções em linguagem natural. Ao mesmo tempo, também evidencia uma limitação central: a geração é baseada em probabilidade e em padrões aprendidos, não em verificação automática da verdade dos fatos.

2.2 Limitações dos Modelos de Linguagem

Embora os LLMs apresentem bom desempenho em tarefas textuais, sua utilização em sistemas que exigem confiabilidade demanda cuidado. A primeira limitação relevante é a alucinação, isto é, a produção de afirmações incorretas, imprecisas ou não sustentadas pelo contexto disponível. Esse problema ocorre porque o modelo pode gerar uma continuação linguisticamente plausível mesmo quando não possui evidência suficiente para responder corretamente (HUANG et al., 2025).

Outra limitação é a dificuldade de atribuição. Como parte do conhecimento

aprendido por um LLM fica distribuída nos pesos da rede neural, nem sempre é possível identificar de qual documento, trecho ou fonte uma afirmação foi derivada. Essa característica é problemática em aplicações acadêmicas e educacionais, nas quais a resposta deve estar associada ao material de estudo utilizado e precisa ser auditável pelo usuário. Esse ponto é destacado por Lewis et al. (2020) ao apontarem que modelos paramétricos não fornecem, de forma direta, a proveniência de suas predições.

Também há o problema da desatualização. O conhecimento interno de um LLM reflete os dados disponíveis durante seu treinamento e não é automaticamente atualizado quando novas informações surgem. Atualizar esse conhecimento por meio de novo treinamento ou *fine-tuning* pode ser custoso, demorado e insuficiente quando o sistema precisa consultar coleções específicas de documentos, como apostilas, livros técnicos ou arquivos em PDF fornecidos pelo usuário (LEWIS et al., 2020).

Além disso, os modelos possuem uma janela de contexto limitada, isto é, uma quantidade máxima de *tokens* que conseguem considerar simultaneamente. Mesmo quando essa janela é ampliada, inserir documentos extensos diretamente no *prompt* pode reduzir a qualidade da recuperação de informações, pois modelos podem ter dificuldade para utilizar trechos localizados no meio de contextos longos, fenômeno conhecido como *Lost in the Middle* (LIU et al., 2024). Portanto, em vez de simplesmente enviar todo o material ao modelo, torna-se necessário selecionar previamente os trechos mais relevantes para cada pergunta.

Essas limitações motivam o uso de uma arquitetura em que o modelo não dependa apenas do conhecimento paramétrico aprendido no treinamento. Para aplicações como a proposta neste trabalho, o objetivo não é apenas produzir uma resposta bem escrita, mas produzir uma resposta ancorada nos documentos de estudo do usuário. Nesse cenário, o desafio passa de simplesmente gerar texto para recuperar evidências adequadas e usá-las como base da geração.

2.3 *Retrieval-Augmented Generation*

O *Retrieval-Augmented Generation* (RAG) é uma abordagem que combina geração textual com recuperação de informação externa. A ideia central é acoplar ao LLM uma memória não paramétrica, composta por documentos consultáveis, de modo que o modelo possa receber trechos relevantes no momento da inferência. Na formulação de Lewis et al. (2020), o RAG combina uma memória paramétrica, representada pelo modelo gerador, com uma memória não paramétrica, representada por um índice de documentos recuperados por similaridade.

Essa combinação altera a natureza da tarefa. Sem RAG, o modelo tende a responder principalmente com base nos padrões e conhecimentos armazenados em seus parâmetros. Com RAG, a resposta passa a ser condicionada por trechos recuperados de uma coleção externa. Dessa forma, o sistema aumenta a chance de gerar respostas mais específicas, factuais e verificáveis, pois o conteúdo usado na geração pode ser revisado, substituído e inspecionado (LEWIS et al., 2020).

O funcionamento básico de um sistema RAG pode ser entendido em três etapas. A primeira é a indexação, realizada antes da consulta do usuário. Nessa fase, documentos extensos são extraídos, divididos em partes menores e convertidos em vetores por um modelo de *embedding*. A divisão em partes, frequentemente chamadas de *chunks*, é necessária porque tanto os modelos de *embedding* quanto os LLMs possuem limites de contexto. Essa etapa deve respeitar, sempre que possível, fronteiras reconhecíveis por humanos, como capítulos, seções, tópicos ou parágrafos, para evitar que informações dependentes fiquem separadas de forma inadequada.

A segunda etapa é a recuperação. Quando o usuário envia uma pergunta, essa consulta também é convertida em vetor e comparada aos vetores dos trechos indexados. O sistema então seleciona os trechos mais próximos segundo alguma medida de similaridade, como similaridade de cosseno, distância euclidiana ou produto interno. A recuperação pode ser lexical, quando busca palavras-chave; vetorial, quando busca proximidade semântica; híbrida, quando combina as duas estratégias; ou ainda baseada em relações explícitas entre documentos, como em grafos de conhecimento.

A terceira etapa é a geração. Os trechos recuperados são inseridos no contexto do *prompt* juntamente com a pergunta do usuário, e o LLM produz uma resposta considerando essas evidências. Nesse ponto, o modelo deixa de atuar apenas como fonte implícita de conhecimento e passa a funcionar como um sintetizador do material recuperado. Para o presente trabalho, essa característica é essencial, pois permite gerar resumos, questões e *flashcards* associados ao conteúdo de uma base de conhecimento construída a partir dos materiais de estudo.

2.4 Qualidade da Recuperação em Sistemas RAG

O desempenho de um sistema RAG depende diretamente da qualidade da recuperação. Se os trechos retornados forem incorretos, incompletos ou irrelevantes, o LLM receberá uma base inadequada para responder e a chance de erro aumenta. Portanto, o RAG não elimina automaticamente as alucinações; ele reduz sua probabilidade quando a memória externa é bem construída, quando a consulta é bem formulada e quando os trechos recuperados realmente contêm as evidências necessárias.

Na literatura, a evolução dos sistemas RAG é organizada em três paradigmas principais: *Naive RAG*, *Advanced RAG* e *Modular RAG* (GAO et al., 2024). O *Naive RAG* corresponde ao fluxo essencial descrito anteriormente: primeiro os documentos são indexados, depois uma consulta recupera os trechos semanticamente mais próximos e, por fim, esses trechos são inseridos no *prompt* para orientar a geração. Essa estrutura é simples, de baixo acoplamento e adequada quando o objetivo central é conectar uma coleção documental específica ao modelo gerador sem modificar o próprio LLM.

O *Advanced RAG* surge como uma extensão desse fluxo básico para melhorar a qualidade da recuperação. É nesse paradigma que se situam as estratégias de *pre-retrieval* e *post-retrieval* mencionadas neste capítulo (GAO et al., 2024). As estratégias de *pre-retrieval* atuam antes da busca e procuram melhorar aquilo que será pesquisado ou a forma como a consulta será representada. Elas podem incluir limpeza dos documentos, enriquecimento com metadados, escolha cuidadosa do

tamanho dos *chunks*, otimização da estrutura de índice, expansão ou reescrita da consulta. Em sistemas educacionais, essa etapa é importante porque estudantes podem formular perguntas incompletas, vagas ou com vocabulário diferente daquele presente no material didático.

As estratégias de *post-retrieval*, também associadas ao *Advanced RAG*, atuam depois da recuperação inicial e antes da geração. Nessa fase, os trechos retornados podem ser reordenados, filtrados ou comprimidos antes de serem enviados ao LLM. O *re-ranking*, por exemplo, reavalia os candidatos recuperados para priorizar os trechos mais úteis à consulta; já a compressão de contexto busca remover conteúdo redundante ou pouco relevante, preservando as evidências necessárias dentro da janela de contexto do modelo (GAO et al., 2024).

O *Modular RAG* representa um nível mais flexível de organização. Em vez de manter um *pipeline* fixo, o sistema passa a ser entendido como um conjunto de módulos que podem ser substituídos, combinados ou acionados conforme a tarefa. Nesse paradigma, a aplicação pode decidir quando recuperar informações, qual fonte consultar, quantos trechos utilizar, se deve reescrever a consulta ou se uma nova busca é necessária (GAO et al., 2024). Técnicas como *Self-RAG* exemplificam essa direção ao incorporar mecanismos de reflexão sobre recuperar, gerar e criticar a própria resposta (ASAI et al., 2023).

No escopo deste trabalho, esses paradigmas são utilizados para posicionar tecnicamente a solução proposta. O Prometheus implementa uma arquitetura de *Naive RAG*: o documento do estudante é extraído e indexado, os tópicos centrais funcionam como consultas para recuperar páginas semanticamente próximas e os trechos recuperados são enviados ao modelo para gerar resumos, simulados e *flashcards*. As estratégias de *pre-retrieval*, *post-retrieval* e a orquestração modular são apresentadas como referências conceituais que explicam possíveis evoluções do sistema, mas não constituem o núcleo implementado nesta versão.

Capítulo 3

Prometheus

As discussões apresentadas no Capítulo 1 evidenciam falhas nos métodos tradicionais de estudo, especialmente quando aplicados para ambientes digitais e ao aprendizado autônomo. Apesar da ampla disponibilidade de materiais educacionais, muitos estudantes não dispõem de ferramentas que permitam acompanhar, de forma sistemática, sua evolução ou compreender com clareza quais conteúdos foram efetivamente assimilados.

Nesse cenário, o estudo tende a ocorrer de maneira passiva, sem mecanismos consistentes de *feedback* ou identificação de lacunas de conhecimento, que podem comprometer a consolidação do aprendizado ao longo do tempo. Torna-se, portanto, necessário adotar uma solução capaz de centralizar conteúdos, avaliações e estratégias de reforço do aprendizado em um único ambiente, promovendo maior autonomia e consciência no processo de estudo.

Diante desse contexto, é proposto o *Prometheus*, uma plataforma digital de estudo supervisionado que transforma documentos enviados pelo estudante em bases de conhecimento, resumos, *decks* de *flashcards*, simulados personalizados e indicadores de desempenho. A proposta parte da ideia de que estudar não deve significar apenas armazenar arquivos ou ler textos de forma linear, mas organizar materiais em um ciclo contínuo de síntese, recuperação ativa, avaliação e revisão orientada por evidências.

3.1 Mini-mundo e Visão Geral do Sistema

No contexto de modelagem de sistemas, o mini-mundo corresponde ao recorte da realidade que será representado computacionalmente. Esse recorte delimita quais atores, objetos, eventos e relações fazem parte do domínio analisado, permitindo compreender quais situações o sistema deve registrar, processar e apresentar ao usuário. Assim, em vez de descrever todo o processo educacional de forma ampla, o mini-mundo concentra-se apenas nas interações relevantes para os objetivos da plataforma.

O mini-mundo do Prometheus é composto por um ator principal, o Estudante, e por um conjunto de objetos que representam o percurso de estudo dentro da plataforma. O Estudante é responsável por criar uma base de conhecimento, enviar um material de estudo em formato PDF, consultar os conteúdos processados, realizar simulados, revisar *flashcards* e acompanhar seu desempenho. A base de conhecimento é o conceito central do sistema, pois agrupa o documento original, o texto extraído, as páginas indexadas, o resumo final, os tópicos centrais e os artefatos educacionais gerados a partir desse material.

O sistema considera como entrada principal um documento textual submetido pelo usuário. Esse documento é tratado em diferentes níveis de granularidade: primeiro como arquivo original, depois como páginas extraídas, em seguida como fragmentos menores de texto, denominados *chunks*, e finalmente como representações semânticas vetoriais. A partir dessa estrutura, o Prometheus produz saídas educacionais conectadas entre si: um resumo geral do material, uma lista controlada de tópicos centrais, questões de múltipla escolha para simulados, *flashcards* para revisão e estatísticas de desempenho associadas às respostas do estudante.

Essa organização evita que as funcionalidades sejam compreendidas como recursos independentes. O resumo não é apenas uma síntese isolada do documento; ele também serve como etapa intermediária para identificar os tópicos mais importantes. Os tópicos, por sua vez, orientam a recuperação de trechos relevantes do documento e são reaproveitados na geração de simulados e *flashcards*. Os simulados registram respostas

e produzem indicadores de desempenho, enquanto os *flashcards* oferecem uma forma complementar de revisão. Assim, o grande problema tratado pelo Prometheus pode ser dividido em três componentes interdependentes: transformar documentos extensos em conhecimento estruturado, gerar materiais de estudo fundamentados nesse conhecimento e fornecer retorno ao estudante sobre sua evolução.

Como apresentado na Figura 3.1, o funcionamento do Prometheus segue uma lógica *top-down*. Em nível mais amplo, o estudante deseja compreender um material e avaliar seu domínio sobre ele. Em nível intermediário, o sistema transforma esse material em uma base de conhecimento estruturada. Em nível operacional, a plataforma executa etapas de extração, indexação, sumarização, recuperação semântica, geração e armazenamento dos resultados. Essa decomposição orienta tanto os requisitos do sistema quanto a modelagem de dados apresentada nas seções seguintes.

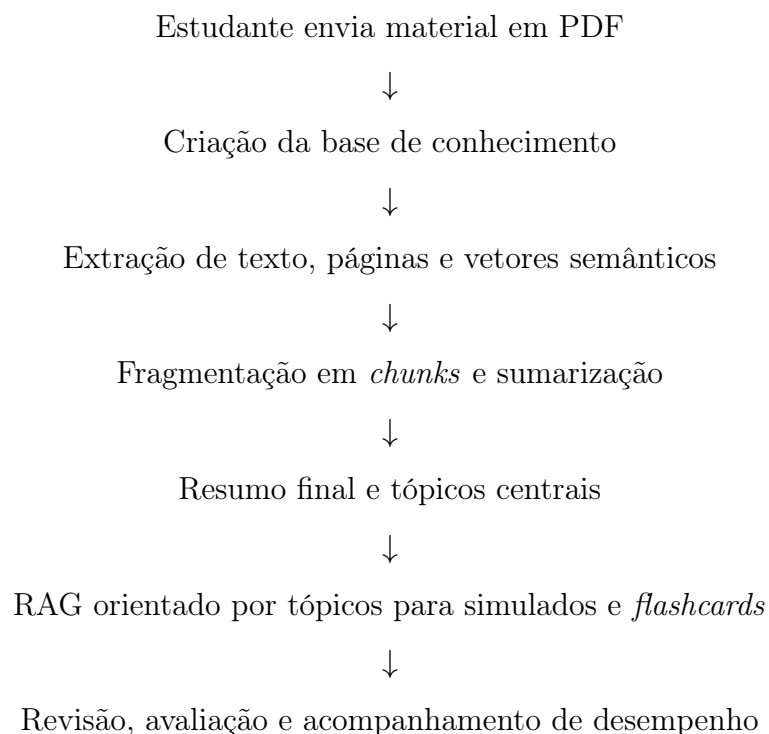


Figura 3.1: Fluxo macro do processo de estudo implementado pelo Prometheus.

3.2 Requisitos do Sistema

Nesta seção são apresentados os Requisitos Funcionais (RF) que orientam o desenvolvimento do sistema proposto, definindo de forma clara e estruturada o que o sistema deve fazer, bem como as condições e restrições sob as quais deverá operar. Segundo Sommerville (2019), os requisitos representam uma especificação compreensível e formal das funcionalidades e propriedades que um sistema deve possuir para atender às necessidades dos seus usuários e demais partes interessadas, servindo como base de comunicação entre *stakeholders*, desenvolvedores e demais participantes do projeto.

3.2.1 Requisitos funcionais

- **RF-01: Cadastrar Usuário**

O sistema deve permitir que um novo usuário realize seu cadastro informando nome completo, e-mail e senha.

- **RF-02: Gerenciar de Perfil**

O sistema deve permitir que o usuário visualize e atualize suas informações cadastrais, incluindo a imagem de avatar.

- **RF-03: Criar Base de Conhecimento**

O sistema deve permitir que o usuário crie uma nova base de conhecimento informando nome e objetivo de estudo.

- **RF-04: *Upload* de Documento**

O sistema deve permitir o envio de documentos (ex.: PDF) para compor o conteúdo de uma base de conhecimento.

- **RF-05: Listar Base de Conhecimento**

O sistema deve listar todas as bases de conhecimento criadas pelo usuário.

- **RF-06: Buscar Base de Conhecimento**

O sistema deve permitir que o usuário busque bases de conhecimento por termos relacionados na tela principal.

- **RF-07: Favoritar Base de Conhecimento**

O sistema deve permitir que o usuário marque bases de conhecimento como favoritas.

- **RF-08: Listar Bases de Conhecimento favoritas**

O sistema deve listar exclusivamente as bases de conhecimento marcadas como favoritas pelo usuário.

- **RF-09: Editar Bases de Conhecimento**

O sistema deve permitir que o usuário edite as informações de suas bases de conhecimento.

- **RF-10: Excluir Bases de Conhecimento**

O sistema deve permitir que o usuário exclua suas bases de conhecimento.

- **RF-11: Armazenamento Estruturado do Documento**

O sistema deve armazenar o documento original enviado e seus metadados associados.

- **RF-12: Gerar Resumo**

O sistema deve gerar e exibir automaticamente um resumo textual sobre o conteúdo da base de conhecimento processada.

- **RF-13: Criar Simulados**

O sistema deve permitir que o usuário crie simulados associados a uma base de conhecimento.

- **RF-14: Listar Simulados**

O sistema deve listar todos os simulados criados pelo usuário dentro de uma base de conhecimento.

- **RF-15: Excluir Simulados**

O sistema deve permitir ao usuário excluir simulados gerados.

- **RF-16: Acessar Simulados**

O sistema deve permitir que o usuário acesse um simulado e visualize suas respectivas questões.

- **RF-17: Gerar Questões automaticamente**

O sistema deve gerar automaticamente as questões do simulado com base no conteúdo da base de conhecimento associada.

- **RF-18: Listar Questões**

O sistema deve listar as questões e suas alternativas dentro do simulado correspondente.

- **RF-19: Registro de Respostas do Usuário**

O sistema deve permitir que o usuário responda às questões apresentadas e submeta suas respostas.

- **RF-20: Explicação de Gabarito**

O sistema deve exibir a explicação da alternativa correta imediatamente após a submissão das respostas.

- **RF-21: Exibir estatísticas de usuário**

O sistema deve exibir, no painel do usuário, a quantidade de bases de conhecimento criadas, o número de simulados não finalizados, o percentual médio de conclusão dos simulados e um gráfico de desempenho por base de conhecimento.

- **RF-22: Extrair Texto do Documento**

O sistema deve extrair o conteúdo textual do documento importado para torná-lo disponível para geração de materiais de estudo.

- **RF-23: Indexação de Base de Conhecimento:**

O sistema deve indexar o conteúdo extraído do documento para permitir a recuperação de contexto pelo modelo de IA durante a geração de materiais.

- **RF-24: Criar *Decks***

O sistema deve permitir que o usuário crie *decks* de *flashcards* vinculados a uma base de conhecimento.

- **RF-25: Listar *decks***

O sistema deve listar todos os *decks* criados pelo usuário dentro de uma base de conhecimento.

- **RF-26: Acessar *decks***

O sistema deve permitir que o usuário acesse um *deck* e visualize seus *flashcards* gerados.

- **RF-27: Excluir *decks***

O sistema deve permitir que o usuário exclua um *deck* e todos os seus cartões associados.

- **RF-28: Gerar *Flashcards* automaticamente**

O sistema deve gerar automaticamente *flashcards* com frente e verso com base no conteúdo da base de conhecimento, inserindo-os no *deck* correspondente.

- **RF-29: Criar Manual de *Flashcards***

O sistema deve permitir que o usuário crie manualmente novos *flashcards* em um *deck*, informando o conteúdo da frente e do verso do cartão.

- **RF-30: Visualizar *Flashcards***

O sistema deve exibir a frente do *flashcard* e permitir que o usuário revele o verso mediante interação.

- **RF-31: Excluir *Flashcards***

O sistema deve permitir que o usuário exclua *flashcards* de um *deck*.

- **RF-32: Classificar Dificuldade dos *Flashcards***

O sistema deve permitir que o usuário classifique a dificuldade percebida de cada *flashcard* nos níveis fácil, médio e difícil.

- **RF-33: Inserir Tópico Personalizado**

O sistema deve permitir que o usuário insira um tópico personalizado para orientar a geração de materiais de estudo.

- **RF-34: Selecionar Tópicos Sugeridos**

O sistema deve permitir que o usuário selecione um dos 10 tópicos retornados automaticamente pelo sistema.

3.2.2 Regras de Negócio

As Regras de Negócio (RN) representam as restrições, condições e diretrizes que orientam o funcionamento do sistema dentro do domínio da aplicação. Diferentemente dos requisitos funcionais, que descrevem as funcionalidades disponibilizadas ao usuário, as regras de negócio definem as políticas e limites que devem ser respeitados durante a execução dessas funcionalidades. No contexto do sistema proposto, tais regras estabelecem condições para a criação, organização e utilização das bases de conhecimento, simulados e materiais de estudo, garantindo consistência dos dados, coerência das operações realizadas e alinhamento com os objetivos educacionais da aplicação.

- **RN-01: Limite de *Upload* de PDF:**

O arquivo enviado não pode ultrapassar 100MB ou 500 páginas.

- **RN-02: Validação de Extração Mínima:**

O sistema deve abortar a criação da base e alertar o usuário se o PDF processado retornar menos de 100 caracteres legíveis.

- **RN-03: Registro de Desempenho:**

O sistema deve armazenar o desempenho do usuário em cada avaliação realizada.

- **RN-04: Volume de Questões:**

A geração automática de um simulado deve sempre produzir exatamente 10 questões de múltipla escolha.

- **RN-05: Limite de *Flashcards* por *Deck*:**

O motor de IA deve extrair e gerar um máximo de 20 *flashcards* por requisição de processamento.

- **RN-06: Tamanho do Resumo:**

O resumo textual automático gerado pela IA deve ser parametrizado para conter entre 300 e 400 palavras.

- **RN-07: Integridade de Exclusão:**

Ao excluir uma Base de Conhecimento, todos os simulados, *decks* e históricos de desempenho atrelados a ela devem ser excluídos.

- **RN-08: Imutabilidade do Simulado:**

Uma vez que as respostas de um simulado são submetidas, ele é bloqueado para novas tentativas.

- **RN-09: Complexidade de Senha:**

A senha cadastrada deve ter no mínimo 8 caracteres, contendo ao menos uma letra e um número.

3.2.3 Casos de Uso

A modelagem por Casos de Uso tem como objetivo representar, de forma estruturada, as funcionalidades oferecidas pelo sistema a partir das interações realizadas pelos seus usuários. Por meio dessa técnica, é possível identificar quais serviços a aplicação disponibiliza e como eles são acionados no contexto de uso real. Cada caso de uso expressa uma finalidade clara do ator ao utilizar o sistema, contribuindo para a delimitação do escopo funcional e para a compreensão das operações que o software deve suportar (SOMMERVILLE, 2019).

O Diagrama de Casos de Uso do sistema (Figura 3.2) ilustra as interações funcionais entre o ator envolvido e a plataforma. Diferentemente de sistemas que apresentam múltiplos perfis de acesso, o sistema proposto possui apenas um ator externo, denominado Estudante, responsável por todas as interações com a aplicação.

Esse ator atua no núcleo operacional da plataforma, sendo responsável por gerenciar suas bases de conhecimento ao poder enviar documentos que as alimente, realizar simulados e acompanhar sua evolução por meio de painéis de estatísticas. O detalhamento completo dos fluxos de interação encontra-se na Especificação de Casos de Uso, disponível no Apêndice deste trabalho.



Figura 3.2: Diagrama de Casos de Uso do sistema Prometheus

3.3 Workflow de Criação da Base de Conhecimento

O fluxo tem início quando o estudante preenche um formulário na interface, informa o nome ou objetivo de estudo da base e anexa um arquivo PDF. A submissão é realizada por uma requisição *Hypertext Transfer Protocol* (HTTP) do tipo *POST*, enviada em formato *multipart/form-data*, contendo os metadados da base e

o documento selecionado. Antes de iniciar o processamento, o servidor valida se o arquivo foi enviado corretamente e se atende às restrições estabelecidas pelas regras de negócio, como tamanho máximo, quantidade de páginas e presença mínima de texto extraível.

Após a validação, o Prometheus cria imediatamente os registros iniciais no banco de dados. A base de conhecimento é cadastrada com estado pendente, o arquivo original é armazenado no servidor e seus metadados são associados à entidade *knowledges*. Em paralelo, o sistema cria estruturas iniciais para o *deck*, que servirá para agrupar os *flashcards* e para o simulado, ambos em estado de processamento. Essa decisão permite que a interface apresente ao estudante que a solicitação foi aceita e que os materiais estão sendo preparados, mesmo antes da conclusão das etapas de geração do material.

O processamento do documento não ocorre integralmente durante a requisição inicial, pois operações como leitura de PDF, extração de texto, geração de *embeddings* e chamadas a modelos de linguagem podem demandar tempo superior ao aceitável em uma interação síncrona. Por isso, a aplicação enfileira uma tarefa de segundo plano responsável por carregar o documento, extrair o conteúdo por página e gerar uma representação vetorial para cada página. Cada registro de página passa a conter, de forma associada à base de conhecimento, o número da página, o texto extraído e o vetor semântico correspondente.

Essa primeira etapa define a base material do sistema. O arquivo original permanece preservado como referência, enquanto o texto extraído e os vetores armazenados tornam possível recuperar trechos relevantes posteriormente. Dessa forma, o conteúdo não é tratado apenas como um arquivo estático, mas como uma coleção consultável de unidades textuais que podem ser utilizadas em diferentes momentos do processo de estudo.

3.4 Processamento do Documento e Geração do Resumo

Depois que o texto do documento é extraído, o Prometheus inicia a etapa de sumarização. O objetivo dessa etapa é produzir uma síntese compreensível do material e, ao mesmo tempo, identificar os tópicos centrais que servirão como guia para os demais artefatos. Para isso, o texto completo é fragmentado em *chunks*, isto é, partes menores que podem ser processadas individualmente por uma LLM.

A fragmentação é necessária porque modelos de linguagem possuem limite de janela de contexto, ou seja, não conseguem analisar uma quantidade ilimitada de texto em uma única chamada. Mesmo quando um documento cabe tecnicamente nesse limite, enviar todo o conteúdo de uma vez pode aumentar custos, prejudicar a recuperação de informações localizadas no meio do texto e reduzir o controle sobre a qualidade da síntese (LIU et al., 2024).

No Prometheus, o texto é dividido em *chunks* de 4000 caracteres, com uma sobreposição de 256 caracteres entre fragmentos adjacentes. O tamanho do *chunk* permite processar trechos com contexto suficiente, enquanto a sobreposição reduz a perda de continuidade entre uma parte e outra do documento. Essa estratégia é importante porque um corte arbitrário poderia separar conceitos relacionados e gerar resumos parciais incoerentes. O resultado é uma lista de *chunks* que representa o documento em unidades menores, adequadas ao processamento paralelo e à síntese incremental.

Cada *chunk* é enviado a um primeiro *prompt* de sumarização. Esse *prompt* define o papel do modelo como um editor especialista e extrator de ideias, solicita a identificação da ideia central e dos ensinamentos fundamentais do trecho, exige uma resposta no mesmo idioma do material original e orienta a escrita em narrativa contínua, sem listas ou metalinguagem. O resultado esperado é um resumo parcial, suficientemente informativo para preservar a ideia central daquele fragmento.

Em seguida, os resumos parciais são combinados por meio de uma etapa de consolidação. O segundo *prompt* recebe a coleção de resumos e solicita uma síntese abrangente, estruturada e envolvente do documento. Além do texto final, esse *prompt*

exige uma saída estruturada contendo exatamente dez tópicos centrais. A resposta é validada por um esquema de dados, garantindo que o resumo e a lista de tópicos sejam retornados em formato previsível para armazenamento e uso nas etapas seguintes.

A Figura 3.3 sintetiza a lógica de processamento adotada. Primeiro, cada parte do texto é resumida de forma independente, o que permite executar diferentes análises ao mesmo tempo quando a infraestrutura utilizada suporta esse comportamento. Depois, os resumos parciais são reunidos e submetidos a uma nova etapa de síntese, responsável por produzir o resumo final e a lista de tópicos. Essa estratégia evita tentar resumir o documento inteiro em uma única chamada e torna o processo mais controlável, escalável e menos sensível às limitações de contexto da LLM.

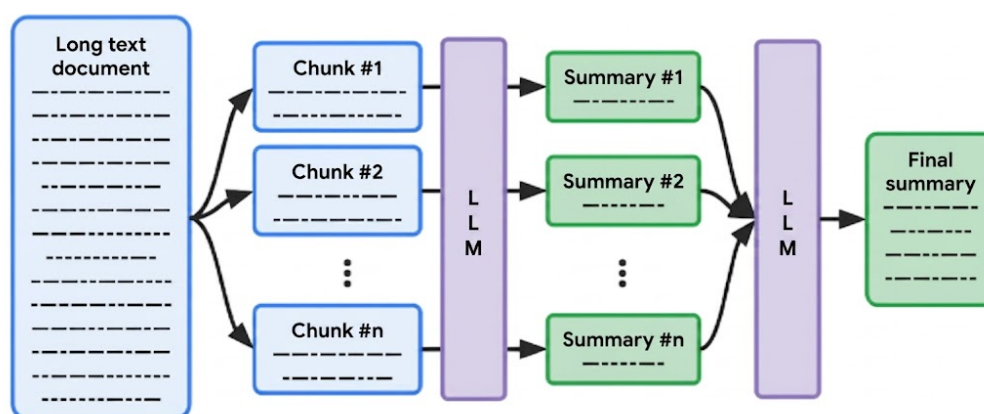


Figura 3.3: Fluxo de processamento paralelo e consolidação do resumo.

3.5 Tópicos Centrais e RAG Orientado por Tópicos

Os tópicos centrais funcionam como uma ponte entre a etapa de sumarização e a geração de materiais de estudo. Eles não são cadastrados manualmente pelo usuário, mas extraídos automaticamente no momento em que o resumo final é produzido. A escolha de limitar a lista a dez tópicos é uma decisão de projeto: uma quantidade pequena favorece uma visão acionável do documento, evita listas extensas pouco úteis e obriga o modelo a selecionar conceitos de maior impacto.

Para reduzir redundâncias, o *prompt* de combinação orienta o modelo a gerar tópicos distintos, amplos e não repetitivos. Além disso, o formato estruturado da

resposta impõe um tamanho fixo para a lista. Caso a saída não respeite o esquema esperado, o resultado é rejeitado pelo processo de validação. Nas etapas consumidoras, como geração de simulados e *flashcards*, o sistema também utiliza apenas o conjunto controlado de tópicos; se ocorrer uma falha excepcional e nenhum tópico estiver disponível, o conteúdo geral da base é utilizado como alternativa para que a geração não dependa exclusivamente dessa lista.

A geração de simulados e *flashcards* utiliza uma implementação de RAG orientada por esses tópicos. Durante a indexação, realizada após o envio do PDF, cada página do documento tem seu texto salvo no banco e recebe um *embedding* com 1536 dimensões. Esses vetores são armazenados em uma estrutura capaz de permitir comparações por similaridade semântica, seguindo a ideia de memória não paramétrica recuperável associada ao RAG (LEWIS et al., 2020; GAO et al., 2024). Assim, a base de conhecimento passa a funcionar como uma memória consultável do material enviado pelo estudante.

Na etapa de recuperação, cada tópico central é convertido em um vetor por meio do mesmo modelo de *embedding*. Esse vetor é comparado com os vetores das páginas pertencentes à base de conhecimento, utilizando distância de cosseno. O sistema recupera as três páginas mais próximas semanticamente de cada tópico, equilibrando quantidade de contexto e custo de processamento. Em seguida, o conteúdo dessas páginas é concatenado e inserido no *prompt* de geração, de modo que a LLM produza questões ou cartões com base em trechos efetivamente extraídos do documento.

No caso dos simulados, o modelo gera questões de múltipla escolha, cada uma contendo enunciado, alternativas, indicação da resposta correta e explicação pedagógica do gabarito. No caso dos *flashcards*, o modelo gera cartões com frente e verso, nos quais a frente apresenta uma pergunta, conceito ou estímulo de recuperação, e o verso apresenta a resposta ou explicação. Embora sejam artefatos diferentes, ambos derivam da mesma sequência lógica: tópicos centrais orientam a busca semântica, a busca recupera contexto do documento, e o contexto fundamenta a geração pela LLM.

Essa escolha reduz a dependência do conhecimento paramétrico do modelo e

diminui a chance de geração desconectada do material de estudo. O objetivo não é apenas usar IA para criar conteúdo, mas controlar de onde vem o contexto utilizado na geração. Por isso, a arquitetura de RAG é essencial para manter a rastreabilidade entre documento original, tópicos extraídos, páginas recuperadas e artefatos produzidos (LEWIS et al., 2020; GAO et al., 2024).

3.6 Arquitetura Operacional e Limitações

A arquitetura operacional do Prometheus utiliza tarefas assíncronas para evitar que o usuário permaneça aguardando operações demoradas durante uma requisição HTTP. Um *job*, ou tarefa de segundo plano, encapsula uma etapa autônoma do processamento, como extração de páginas, sumarização, geração de simulados ou geração de *flashcards*. Assim que a base é criada, o usuário é redirecionado para sua página, enquanto os *jobs* continuam executando no servidor.

A adoção de uma arquitetura baseada em *jobs* justifica-se pela inadequação do processamento síncrono em sistemas *web* que executam operações de alta latência. Etapas como comunicação com modelos de linguagem, geração de *embeddings* e extração de conteúdo de documentos podem exceder o tempo tolerado por uma requisição HTTP convencional, causando interrupções prematuras, falhas de conexão ou bloqueio da interface enquanto o usuário aguarda a resposta. Sem essa estrutura, a experiência de uso seria degradada, pois a aplicação permaneceria dependente da conclusão integral do processamento antes de liberar a navegação ou apresentar qualquer retorno ao estudante.

Para informar o andamento do processamento, a interface utiliza *polling*. Nesse modelo, o *front-end* consulta periodicamente o servidor, em intervalos de 2 segundos, para verificar se os estados da base, do *deck* e do simulado foram atualizados. Essa estratégia evita manter uma conexão síncrona aberta durante todo o processamento e permite que a interface seja atualizada quando os resultados estiverem disponíveis. Com isso, o estudante percebe o sistema como responsivo, mesmo quando há tarefas de alta latência em execução.

Apesar dessas decisões, a proposta possui limitações importantes associadas à extração de PDF. Esse formato nem sempre armazena texto como uma sequência linear; em muitos casos, o conteúdo é composto por objetos posicionados visualmente na página. Por isso, documentos com múltiplas colunas podem ter trechos extraídos fora de ordem, tabelas podem perder sua estrutura relacional e imagens, gráficos ou diagramas podem ser ignorados quando não há texto associado. Essas limitações afetam diretamente a qualidade dos *chunks*, do resumo e dos materiais gerados.

Também há casos em que o PDF contém texto escaneado, fontes vetoriais, caracteres corrompidos ou restrições de acesso. Nessas situações, a extração pode retornar conteúdo vazio ou incoerente, exigindo validações mínimas antes de prosseguir com a criação da base. O Prometheus considera essas restrições por meio de regras de validação e estados de processamento, mas não resolve completamente problemas que dependeriam de técnicas adicionais, como reconhecimento óptico de caracteres ou processamento multimodal de imagens e tabelas. Portanto, a qualidade dos artefatos gerados está diretamente ligada à qualidade do texto extraído do documento original.

3.7 Modelagem de Dados

A definição da estrutura de dados do sistema é um elemento central para garantir o funcionamento adequado das funcionalidades propostas, especialmente considerando o tratamento de documentos, a geração de conteúdos educacionais e o acompanhamento do desempenho do usuário. Para representar essa estrutura de forma clara e sistemática, foi elaborado um Diagrama Entidade-Relacionamento (DER), permitindo visualizar como as tabelas do banco de dados se conectam e sustentam as operações do sistema.

No contexto da aplicação proposta, a modelagem de dados assume um papel ainda mais relevante, uma vez que o sistema depende da correta associação entre conteúdos extraídos de documentos e resultados gerados por modelos de linguagem. Dessa forma, o modelo definido não apenas viabiliza o armazenamento das informações, mas também assegura sua coerência, rastreabilidade e reutilização ao longo das diferentes

funcionalidades da plataforma, caso o estudante queira gerar outro simulado a partir da mesma base de conhecimento, por exemplo. A partir dessa modelagem, foi estruturado o banco de dados responsável por dar suporte às operações do sistema, contribuindo para sua consistência, organização e evolução, presumindo uma base possivelmente escalável.

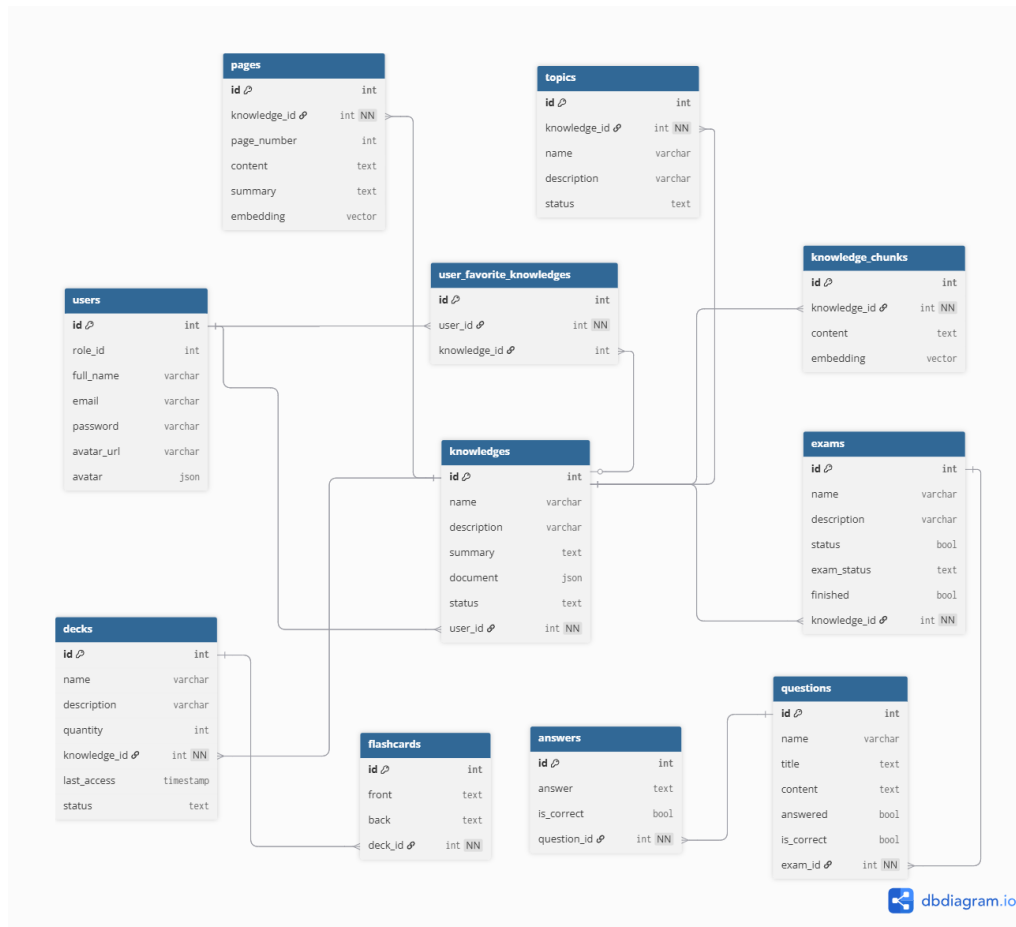


Figura 3.4: Diagrama Entidade-Relacionamento do sistema Prometheus.

3.7.1 Entidades

A entidade *users* é responsável por armazenar os dados cadastrais dos usuários do sistema, atendendo diretamente aos requisitos de cadastro e gerenciamento de perfil (RF-01 e RF-02). Nela são mantidas informações como nome completo, e-mail, senha e avatar. Além disso, a estrutura dessa entidade deve respeitar regras de validação, conforme especificado na RN-09, garantindo maior segurança no acesso. Essa entidade também atua como ponto central de relacionamento com as demais,

uma vez que todas as operações do sistema são realizadas no contexto de um usuário autenticado.

Tabela 3.1: Estrutura da Tabela *users*.

Coluna	Descrição
id	Identificador único do usuário.
role_id	Chave que dita qual cargo do usuário.
full_name	Nome do usuário.
email	Endereço de email do usuário.
password	Senha do usuário.
avatar_url	URL para o avatar do usuário.
avatar	JSON para armazenar metadados do avatar.

A entidade *knowledges* representa as bases de conhecimento criadas pelos usuários, sendo o elemento central do sistema. Ela atende aos requisitos de criação, busca, edição, exclusão, listagem de bases, conforme consta nos RF-03 a RF-06 e RF-09 a RF-10. Além de armazenar informações como nome, descrição, resumo e documento associado. Essa estrutura também suporta a ponte para o armazenamento do conteúdo processado, de acordo com RF-11 e RF-12, incluindo o resumo gerado automaticamente, respeitando a limitação de tamanho definida no RN-06. A relação com o usuário garante a individualização das bases, enquanto sua exclusão deve obedecer à integridade referencial definida no RN-07, removendo todos os dados associados. Além, é ditado pela RN-01 que o documento não deve exceder um tamanho específico de dados.

Tabela 3.2: Tabela *knowledges*.

Coluna	Descrição
id	Identificador único da base de conhecimento.
user_id	Referência para o usuário dono da base.
name	Nome da Base de Conhecimento.
description	descrição fornecida pelo usuário sobre a base.
summary	Resumo gerado automaticamente para a base.
document	documento fornecido pelo usuário.
status	URL para o avatar do usuário.

A entidade *pages* é responsável por armazenar o conteúdo segmentado dos documentos enviados pelo usuário, permitindo uma organização mais granular das

informações extraídas, como consta o RF-22. Cada registro representa uma página do documento original, contendo seu conteúdo textual, resumo parcial e representação vetorial, utilizada para operações de recuperação de contexto. Essa estrutura viabiliza o processamento eficiente dos dados e contribui para a geração de materiais de estudo. Por fim, auxiliando na verificação do cumprimento da RN-02 do tamanho mínimo que deve ser enviado.

Tabela 3.3: Tabela *pages*.

Coluna	Descrição
id	Identificador único da página.
knowledge_id	Referência para a base de conhecimento específica.
page_number	Número da página do documento.
content	Conteúdo textual completo da página.
summary	Resumo gerado automaticamente para a página.
embedding	representação vetorial do conteúdo da página.

A entidade *knowledge_chunks* armazena fragmentos menores do conteúdo textual das bases de conhecimento, sendo utilizada principalmente para indexação e recuperação semântica das informações. Cada *chunk* possui seu conteúdo e respectivo vetor de representação, permitindo que o sistema realize buscas contextuais e forneça suporte ao modelo de geração automática de questões e *flashcards*. Essa estrutura é essencial para o funcionamento eficiente das funcionalidades baseadas em IA.

Tabela 3.4: Tabela *knowledge_chunks*.

Coluna	Descrição
id	Identificador único do chunk.
knowledge_id	Referência para a base de conhecimento específica.
content	Conteúdo original do fragmento.
embedding	representação vetorial do trecho para buscar semânticas.

A entidade *topics* permite a organização temática das bases de conhecimento, associando conteúdos a tópicos específicos. Esta tabela contribui para a estruturação lógica dos dados e pode ser utilizada para categorizar conteúdos, facilitando buscas, organização e segmentação por assuntos.

A entidade *user_favorite_knowledges* foi modelada para representar a relação entre usuários e bases de conhecimento marcadas como favoritas, atendendo aos

Tabela 3.5: Tabela `topics`.

Coluna	Descrição
<code>id</code>	Identificador único do tópico.
<code>knowledge_id</code>	Referência para a base de conhecimento específica.
<code>name</code>	Nome do tópico identificado do conteúdo.
<code>description</code>	Descrição ou detalhamento do tópico.
<code>status</code>	Status do tópico dentro do processamento.

requisitos de favoritar e listar favoritos, descritos em RF-07 e RF-08. Trata-se de uma tabela associativa que permite que um usuário mantenha um conjunto personalizado de bases de interesse, sem alterar a estrutura original das bases. Essa separação garante flexibilidade e escalabilidade na gestão das preferências individuais.

Tabela 3.6: Tabela `user_favorite_knowledges`.

Coluna	Descrição
<code>id</code>	Identificador único da tabela.
<code>user_id</code>	Referência para o usuário específico.
<code>knowledge_id</code>	Referência para a base de conhecimento específica.

A entidade *exams* representa os simulados criados a partir das bases de conhecimento, atendendo aos requisitos de criação, listagem, acesso e exclusão de simulados, conforme RF-13 a RF-16. Cada simulado está vinculado a uma base de conhecimento e possui atributos que indicam seu estado, como finalização e *status*. Essa entidade também está relacionada às regras de negócio que definem o comportamento dos simulados, como a geração fixa de questões (RN-04) e a impossibilidade de nova tentativa após submissão (RN-08).

A entidade *questions* armazena as questões geradas automaticamente para cada simulado, atendendo aos requisitos de geração e listagem de questões descritas em RF-17 e RF-18. Cada questão contém seu enunciado, conteúdo e informações sobre resposta e correção, permitindo o controle do desempenho do usuário durante a realização do simulado. Seu relacionamento com a tabela *exams* garante que cada simulado possua um conjunto estruturado de questões.

A entidade *answers* armazena as alternativas associadas às questões dos simulados, incluindo a indicação da resposta correta. Ela dá suporte direto ao registro de

Tabela 3.7: Tabela *exams*.

Coluna	Descrição
id	Identificador único do simulado.
knowledge_id	Referência para a base de conhecimento específica.
name	Nome atribuído ao simulado.
description	Descrição breve sobre o conteúdo abordado.
status	Indicação se o simulado está ativo ou disponível.
exam_status	Estado do progresso do simulado (e.g.: iniciado, finalizado).
finished	Indicação de valor verdade se o simulado está concluído ou não.

Tabela 3.8: Tabela *questions*.

Coluna	Descrição
id	Identificador único da questão.
exam_id	Referência para o simulado específico.
name	Nome ou identificador interno da questão.
title	Título resumido da questão.
content	Enunciado completo da questão.
answered	Indicação de valor verdade se a questão foi respondida ou não.
is_correct	Indicação de valor verdade se a questão está correta ou não.

respostas do usuário e à apresentação do gabarito com explicações concordando com RF-19 e RF-20.

Tabela 3.9: Tabela *answers*.

Coluna	Descrição
id	Identificador único das respostas.
question_id	Referência para a determinada questão.
is_correct	Indicação valor verdade se a opção é correta.

De maneira análoga ao que ocorre com os simulados, nos quais há um relacionamento de um-para-muitos entre *exams* e *questions*, e de um-para-muitos entre *questions* e *answers*, a entidade *decks* também estabelece a mesma relação com *flashcards*, permitindo que um único *deck* agrupe múltiplos cartões de estudo.

Tabela 3.10: Tabela *decks*.

Coluna	Descrição
id	Identificador único do <i>deck</i> .
knowledge_id	Referência à base de conhecimento associada.
name	Nome atribuído ao <i>deck</i> .
description	Descrição breve sobre o conteúdo do <i>deck</i> .
quantity	Quantidade de <i>flashcards</i> presentes no <i>deck</i> .
last_access	Data e hora do último acesso ao <i>deck</i> .
status	Estado atual do <i>deck</i> .

Tabela 3.11: Tabela *flashcards*.

Coluna	Descrição
id	Identificador único do <i>flashcard</i> .
deck_id	Referência para o <i>deck</i> específico.
front	Conteúdo exibido na frente do <i>flashcard</i> .
back	Conteúdo exibido no verso do <i>flashcard</i> .
difficulty	Dificuldade do <i>flashcard</i> .

3.8 Trabalhos relacionados

A análise de soluções consolidadas permite posicionar o *Prometheus* em relação a ferramentas já utilizadas no apoio ao estudo. Neste trabalho, são considerados o *Examica*, o *NotebookLM* e o *Anki*, por representarem, respectivamente, a geração automatizada de materiais, a exploração contextual de documentos e a memorização por repetição espaçada. A comparação evidencia que, embora essas soluções ofereçam recursos relevantes, ainda há oportunidades de integração entre rastreabilidade do conteúdo, automação do fluxo de estudo e acompanhamento do desempenho.

A plataforma Examica¹ utiliza modelos de linguagem para gerar materiais de

¹<https://www.examica.io/>

estudo a partir de conteúdos fornecidos pelo usuário. Suas principais vantagens são a interface intuitiva, apresentada na Figura B.3, a criação rápida de conjuntos de estudo e os elementos de gamificação, que incentivam a recorrência. Como ponto fraco, a plataforma pode apresentar menor controle sobre os trechos usados como base pelas respostas em documentos extensos, além de depender de processamentos que podem comprometer a fluidez da experiência. O *Prometheus* aborda esse contexto com RAG, condicionando as gerações a trechos recuperados semanticamente do documento, e com *jobs* assíncronos, que mantêm a aplicação responsiva durante etapas demoradas.

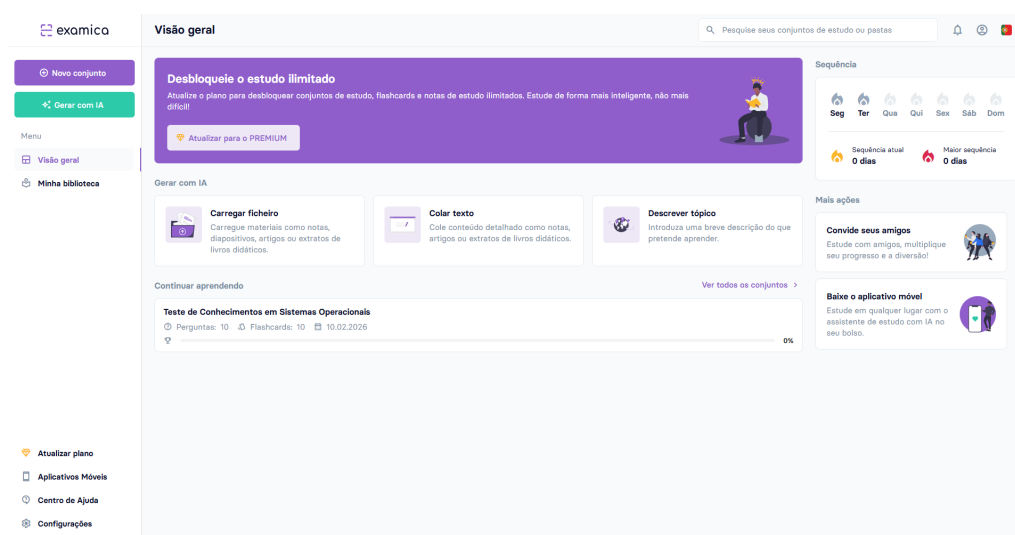


Figura 3.5: Página de visão geral da plataforma Examica, com destaque para recursos de estudo, acompanhamento de progresso e ações adicionais.

O NotebookLM² é uma plataforma voltada à organização e exploração de documentos por meio de modelos de linguagem. Sua principal vantagem é a centralização de múltiplas fontes em um ambiente interativo, como ilustrado na Figura 3.6, permitindo resumos, respostas contextualizadas e perguntas sugeridas. Como limitação, a ferramenta está mais orientada à consulta assistida do que à construção de um ciclo completo de estudo, não priorizando simulados, *flashcards* e métricas de desempenho integradas. O *Prometheus* aborda essa lacuna ao transformar o documento enviado em uma base de conhecimento para estudo supervisionado, integrando compreensão, prática, revisão e monitoramento do progresso.

²<https://notebooklm.google>

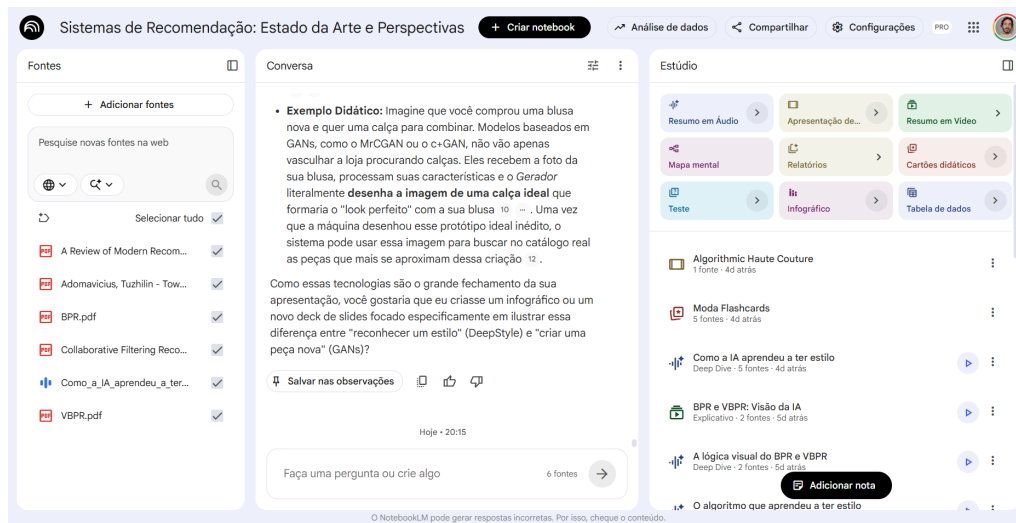


Figura 3.6: Interface do NotebookLM demonstrando a centralização de múltiplas fontes de estudo e a geração automatizada de respostas baseada no contexto enviado.

O Anki³ é uma das ferramentas mais difundidas para memorização de longo prazo, fundamentando-se no Sistema de Repetição Espaçada (SRS). Sua principal vantagem é a recordação ativa por meio de *flashcards* revisados em intervalos ajustados ao desempenho do usuário, conforme exemplificado na Figura 3.7. Entretanto, a criação manual dos cartões pode ser trabalhosa, especialmente quando o conteúdo está em livros, apostilas ou artigos em PDF, e a automação costuma depender de extensões externas. O *Prometheus* preserva o benefício pedagógico dos cartões, mas automatiza sua geração a partir do documento enviado, utilizando extração de tópicos e recuperação semântica para produzir *flashcards* integrados a simulados e métricas de desempenho.

³<https://apps.ankiweb.net/>

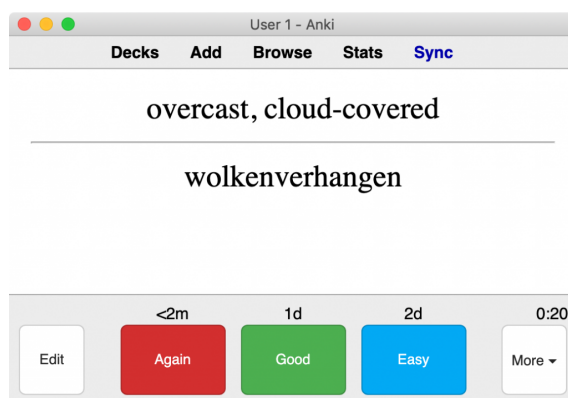


Figura 3.7: Exemplo de cartão de resposta no Anki, em que o termo em inglês aparece na parte superior e sua tradução em alemão logo abaixo. A interface apresenta ainda opções de avaliação da dificuldade da lembrança ('Again', 'Good' e 'Easy'), permitindo ao usuário controlar o intervalo de repetição do conteúdo de acordo com seu desempenho.

Capítulo 4

Implementação e Resultado do Sistema

Este capítulo apresenta como a proposta do Prometheus foi materializada em uma aplicação funcional. A narrativa adotada parte do problema de implementação enfrentado pelo sistema: transformar documentos enviados pelo estudante em uma base de conhecimento estruturada, processar esse conteúdo com apoio de IA, gerar materiais de estudo e disponibilizar os resultados em uma interface simples de acompanhar. Dessa forma, as decisões técnicas não são apresentadas como uma lista isolada de ferramentas, mas como escolhas relacionadas às responsabilidades que cada parte da aplicação precisou cumprir.

O capítulo está organizado em três momentos. Primeiro, são descritas as decisões que sustentam a base técnica da aplicação, incluindo a organização do projeto, a separação de responsabilidades e a integração entre servidor e interface. Em seguida, são discutidos os componentes responsáveis pelo processamento dos documentos, pela persistência dos dados e pela geração dos artefatos educacionais. Por fim, são apresentadas as interfaces implementadas e o resultado obtido do ponto de vista do usuário, evidenciando como a aplicação conduz o estudante por um ciclo de organização, síntese, prática e avaliação.

4.1 Da Proposta à Implementação

A implementação do Prometheus foi guiada pelo fluxo de estudo definido no capítulo anterior. O sistema não deveria apenas permitir o envio de arquivos, mas transformar o documento em uma estrutura consultável e reaproveitável ao longo de diferentes funcionalidades. Assim, cada decisão de implementação foi orientada por uma sequência de objetivos: receber o material do estudante, processar o conteúdo textual, persistir informações relacionais e vetoriais, gerar resumos, simulados e *flashcards*, e apresentar esses resultados de forma progressiva na interface.

Esse fluxo impôs dois desafios principais. O primeiro foi técnico: integrar operações tradicionais de uma aplicação *web*, como autenticação, rotas, validação e persistência, com tarefas de maior latência, como extração de texto, geração de *embeddings*, busca semântica e chamadas a modelos de linguagem. O segundo foi de experiência de uso: permitir que o estudante interagisse com o sistema sem precisar compreender a complexidade interna do processamento. A interface, portanto, deveria comunicar estados, organizar os resultados gerados e manter a navegação coerente com o percurso pedagógico proposto.

Por esse motivo, a arquitetura adotada combina uma aplicação *web* estruturada, uma base de dados capaz de armazenar tanto informações relacionais quanto representações vetoriais e uma camada de orquestração para os fluxos baseados em LLMs. Em vez de separar essas escolhas por nome de tecnologia, as próximas seções explicam como cada uma contribui para implementar uma parte específica da solução.

4.2 Base Técnica e Organização da Aplicação

A primeira etapa da implementação consistiu em estabelecer uma fundação capaz de sustentar o desenvolvimento das funcionalidades principais do Prometheus. Como o objetivo do projeto estava concentrado no processamento de documentos e na geração de materiais de estudo, era necessário reduzir o esforço com configurações iniciais repetitivas, mantendo ao mesmo tempo uma estrutura suficientemente organizada

para evoluir ao longo do desenvolvimento.

4.2.1 Fundação do Projeto e Produtividade Inicial

Para acelerar o início da implementação, foi utilizado um *starter kit* baseado em *AdonisJS*¹, *React*² e *InertiaJS*³. Essa escolha não teve como finalidade apenas adotar tecnologias modernas, mas iniciar o projeto a partir de uma base funcional que já contemplava elementos comuns a aplicações *web*, como autenticação, cadastro, login, recuperação de senha, estrutura de rotas e organização inicial das camadas do sistema.

Essa base reduziu o retrabalho associado à configuração do ambiente e permitiu concentrar o esforço nas funcionalidades específicas do Prometheus, como criação de bases de conhecimento, processamento de documentos, geração de resumos, simulados, *decks* e *flashcards*. Assim, o *starter kit* atuou como uma fundação técnica para a aplicação, favorecendo padronização, consistência e velocidade de desenvolvimento sem deslocar o foco do objetivo principal do sistema.

O projeto também foi organizado em formato de *monorepo*, mantendo em um único repositório os recursos relacionados ao *back-end*, ao *front-end* e aos componentes compartilhados. Essa organização foi adequada ao Prometheus porque as funcionalidades dependem de forte integração entre as regras executadas no servidor e as telas utilizadas pelo estudante. Com as camadas mantidas em uma mesma base de código, tornou-se mais simples compartilhar tipos, componentes, configurações e utilitários, além de acompanhar em conjunto alterações que afetam simultaneamente a lógica de negócio e a interface.

4.2.2 Servidor, Regras de Negócio e Persistência Operacional

O *back-end* da aplicação foi desenvolvido com *AdonisJS*, um *framework web* para *Node.js*. Sua adoção está relacionada à necessidade de estruturar uma aplicação

¹<<https://adonisjs.com/>>

²<<https://react.dev/>>

³<<https://inertiajs.com/>>

com autenticação, validação de dados, controle de rotas, manipulação de banco de dados, migrações e execução de tarefas em segundo plano. Essas características são importantes para o Prometheus porque o sistema precisa lidar tanto com operações imediatas, como cadastro e autenticação do usuário, quanto com processos mais longos, como extração de texto, indexação e geração de materiais.

O modelo assíncrono e orientado a eventos do *Node.js* favorece aplicações que processam múltiplas requisições e operações de entrada e saída sem bloqueio. No contexto do Prometheus, essa característica contribui para a execução de fluxos nos quais o usuário cria uma base de conhecimento e segue navegando enquanto etapas de processamento continuam sendo executadas no servidor. O suporte ao *TypeScript* também foi relevante, pois melhora a segurança de tipos durante o desenvolvimento e reduz a ocorrência de erros em tempo de execução em uma aplicação composta por diferentes entidades e fluxos.

Outro aspecto importante do *AdonisJS* é sua aderência ao padrão arquitetural *Model-View-Controller* (MVC). Esse padrão organiza a aplicação em camadas com responsabilidades bem definidas, favorecendo a manutenção e a evolução do sistema (KRASNER; POPE, 1988). No Prometheus, a camada *Model* representa entidades como usuários, bases de conhecimento, simulados, questões, respostas, *decks* e *flashcards*; a camada *Controller* coordena requisições HTTP e aciona as regras necessárias para cada operação; e a camada *View*, implementada com *React* e *InertiaJS*, apresenta os dados ao usuário por meio de interfaces interativas.

O *Object-Relational Mapping* (ORM) do *AdonisJS* simplificou a manipulação das entidades do domínio, permitindo representar os principais objetos do sistema por meio de classes e relações da própria aplicação. Além disso, o uso de *migrations* possibilitou versionar alterações no esquema do banco de dados, garantindo que a evolução da estrutura de persistência ocorresse de forma controlada e rastreável.

4.2.3 Integração entre Servidor e Interface

A integração entre o servidor e a interface foi realizada com *InertiaJS*. Sua função na arquitetura do Prometheus foi permitir uma experiência próxima à de uma *Single*

Page Application (SPA), sem exigir a criação de uma *Application Programming Interface* (API) REST separada para os fluxos principais da aplicação. Com isso, o servidor continua responsável por rotas, autenticação, autorização e preparação dos dados, enquanto a interface recebe essas informações como propriedades consumidas pelos componentes *React*.

Essa decisão contribuiu para manter a aplicação mais coesa. Como os dados de páginas, estados de processamento, bases de conhecimento, simulados e *decks* são preparados no servidor e entregues diretamente à interface, evita-se a duplicação de regras de roteamento e autenticação no lado do cliente. Para o Prometheus, essa característica é especialmente útil porque os fluxos de estudo dependem de dados associados ao usuário autenticado e de estados gerados no servidor durante o processamento documental.

O *React*, por sua vez, foi adotado para estruturar a interface em componentes reutilizáveis e capazes de reagir a alterações de estado. Essa característica é importante em uma plataforma que precisa exibir carregamentos, resultados parciais, listas de bases, simulados, questões, respostas e cartões de estudo. A possibilidade de atualizar partes da tela sem recarregar toda a página contribui para uma navegação mais fluida e para uma experiência mais adequada aos fluxos de avaliação e revisão.

4.3 Processamento, Dados e Geração de Conteúdo

Após a definição da base técnica da aplicação, a implementação concentrou-se no núcleo funcional do Prometheus: transformar documentos em conhecimento estruturado e reaproveitável. Essa etapa envolve armazenar dados de forma consistente, consultar trechos relevantes do material enviado e coordenar chamadas a modelos de linguagem para gerar artefatos educacionais.

4.3.1 Persistência Relacional e Busca Semântica

O PostgreSQL⁴ foi utilizado como sistema de gerenciamento de banco de dados por oferecer confiabilidade, consistência transacional e capacidade de expansão. Sua conformidade com os princípios *Atomicity, Consistency, Isolation, Durability* (ACID) é relevante para preservar a integridade de informações como usuários, documentos processados, bases de conhecimento, simulados, questões, respostas e resultados de avaliação.

No Prometheus, o banco de dados também precisou armazenar representações vetoriais dos conteúdos extraídos dos documentos. Para isso, foi empregada a extensão *pgvector*, que permite persistir e consultar *embeddings* diretamente no PostgreSQL. Essa escolha aproximou os dados relacionais e semânticos: uma página ou fragmento textual permanece associado à base de conhecimento correspondente, enquanto sua representação vetorial pode ser utilizada em consultas por similaridade.

Essa integração é central para o fluxo de RAG. Quando o sistema precisa gerar um simulado ou um conjunto de *flashcards*, os tópicos centrais ou consultas derivadas do material são comparados com os vetores armazenados. A partir dessa comparação, o sistema recupera trechos semanticamente próximos e os utiliza como contexto para a geração. Dessa forma, os artefatos produzidos permanecem vinculados ao documento original, reduzindo a dependência exclusiva do conhecimento paramétrico do modelo de linguagem (LEWIS et al., 2020; GAO et al., 2024).

4.3.2 Orquestração do *Pipeline* de Inteligência Artificial

O *LangChain*⁵ foi empregado para organizar o *pipeline* baseado em LLMs. Sua função no Prometheus não se limita a chamar uma API de inteligência artificial; ele atua como camada de orquestração das etapas de carregamento do documento, fragmentação do texto, geração de *embeddings*, recuperação semântica e consolidação de respostas.

⁴<<https://www.postgresql.org/about/>>

⁵<<https://www.langchain.com/>>

Na etapa de preparação da base de conhecimento, o documento é tratado em partes menores para preservar contexto e permitir consultas posteriores. Cada fragmento ou página relevante recebe uma representação vetorial, que é armazenada no banco de dados. Na etapa de geração, o sistema utiliza a recuperação semântica para selecionar trechos relacionados aos tópicos centrais e compõe o contexto enviado à LLM. Essa sequência permite que resumos, questões e *flashcards* sejam produzidos com base em conteúdo efetivamente extraído do material do estudante.

O *LangChain* também apoia o tratamento de textos extensos por meio de cadeias de processamento, como a estratégia de *Map-Reduce*. Na fase de *Map*, partes relevantes do documento podem ser processadas separadamente, permitindo resumir ou analisar trechos que ultrapassariam a janela de contexto do modelo. Na fase de *Reduce*, os resultados parciais são consolidados em uma resposta final mais coesa. Essa organização simplifica a implementação do fluxo de geração e contribui para manter a rastreabilidade entre entrada, trechos recuperados e saída produzida.

4.3.3 Modularização por Funcionalidade

A organização interna do Prometheus segue uma arquitetura modular por funcionalidade, conforme exemplificado na Figura 4.1. Essa escolha foi necessária porque o sistema reúne domínios diferentes, como autenticação, gestão de usuários, bases de conhecimento, simulados, questões, respostas, *decks* e *flashcards*. Em vez de concentrar regras, rotas e serviços em uma estrutura genérica, cada módulo agrupa os elementos diretamente associados ao seu domínio, favorecendo separação de responsabilidades e menor acoplamento entre partes do sistema.

Os módulos *auth* e *users* sustentam a camada de acesso da aplicação. O primeiro concentra autenticação e controle de sessão, garantindo que apenas usuários identificados acessem recursos protegidos. O segundo organiza dados do estudante e sua associação com as operações realizadas no sistema. Em conjunto, esses módulos asseguram que bases de conhecimento, simulados e materiais de revisão estejam vinculados ao usuário correto.

O módulo *knowledges* representa o núcleo funcional da aplicação, pois organiza

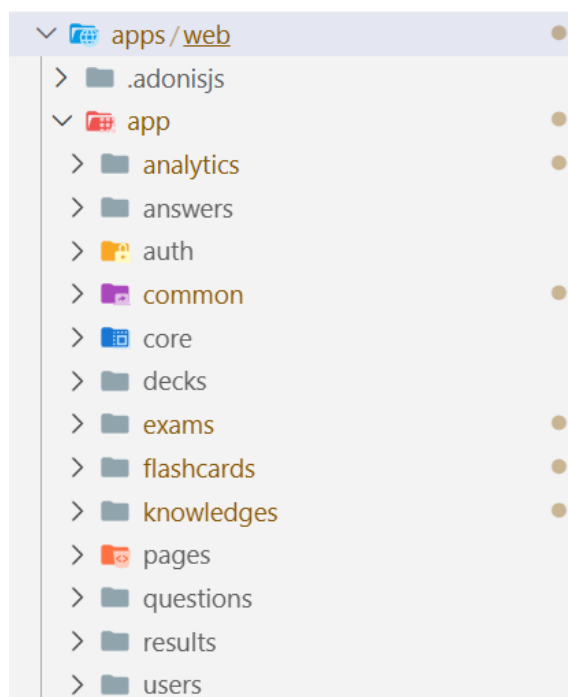


Figura 4.1: Estrutura de diretórios da aplicação web, organizada em módulos. A disposição evidencia a separação de responsabilidades e a modularidade do sistema.

as bases de conhecimento criadas a partir dos documentos enviados. Ele coordena operações de criação, listagem, edição, exclusão e acesso aos conteúdos processados, além de servir como ponto de ligação para os artefatos gerados automaticamente. A partir dele, são acionados fluxos relacionados a simulados e *decks*, mantendo a rastreabilidade entre documento original, tópicos extraídos e materiais produzidos.

Os módulos *exams*, *questions* e *answers* estruturam o ciclo avaliativo. O módulo *exams* gerencia simulados vinculados a uma base de conhecimento, enquanto *questions* organiza as questões geradas e *answers* mantém alternativas, indicação da resposta correta e informações necessárias para correção comentada. De maneira complementar, os módulos *decks* e *flashcards* sustentam o estudo por recuperação ativa, armazenando conjuntos de cartões e seus respectivos conteúdos de frente e verso.

Essa modularização facilita a manutenção e a evolução do Prometheus. Como cada funcionalidade possui escopo definido, alterações em um fluxo tendem a gerar menor impacto nos demais. Além disso, a divisão favorece testes e validações mais específicas, pois cada domínio pode ser analisado de forma isolada antes da integração

com o restante da aplicação.

4.4 Interface e Resultado do Sistema

A etapa final da implementação consistiu em transformar os fluxos internos do Prometheus em uma experiência compreensível para o estudante. Como o sistema executa operações potencialmente complexas, a interface foi planejada para reduzir a carga cognitiva do usuário e apresentar apenas as ações necessárias em cada momento. O objetivo foi fazer com que o estudante percebesse o sistema como um ambiente de estudo guiado, e não como um conjunto de funcionalidades técnicas desconectadas.

4.4.1 Construção Visual e Componentização

Para padronizar a interface e acelerar a construção das telas, foi utilizado o *shadcn/ui*⁶, um conjunto de componentes reutilizáveis que são incorporados diretamente ao código do projeto. Essa característica foi importante porque permitiu adaptar componentes às necessidades do Prometheus sem depender de uma biblioteca fechada ou de personalizações superficiais.

A adoção do *shadcn/ui*, em conjunto com o *React*, contribuiu para manter consistência visual entre páginas, formulários, cartões, botões, listas e elementos de navegação. Como a proposta do Prometheus envolve fluxos recorrentes de criação, consulta, avaliação e revisão, a reutilização de componentes favoreceu uma experiência mais coesa e reduziu o tempo necessário para implementar novas telas. Além disso, a padronização visual contribuiu para que o usuário reconhecesse ações semelhantes em diferentes partes do sistema.

4.4.2 Fluxo de Uso Implementado

A interface implementada organiza a experiência do usuário de forma hierárquica, partindo de uma visão geral da plataforma até chegar às atividades específicas de estudo e avaliação. Essa organização *top-down* evita que o estudante precise navegar

⁶<<https://ui.shadcn.com/>>

diretamente por funcionalidades isoladas, pois cada tela apresenta um nível de detalhamento maior sobre o mesmo ciclo de aprendizagem: primeiro o desempenho geral, depois as bases de conhecimento, em seguida os materiais derivados de uma base e, por fim, as práticas com *flashcards* e simulados.

O primeiro nível desse fluxo é o *Dashboard*, apresentado na Figura 4.2. Essa tela funciona como ponto de entrada do sistema e reúne uma saudação ao usuário, campo de busca, atalhos para criação ou importação de conteúdos e indicadores gerais da plataforma. Os cartões informam a quantidade de bases de conhecimento, simulados em progresso e taxa de conclusão, enquanto a seção de estudos recentes lista bases já cadastradas e oferece ações rápidas. Dessa forma, o *Dashboard* cumpre a função de síntese operacional: ele mostra ao estudante onde existem materiais disponíveis, quais atividades estão em andamento e quais caminhos podem ser acessados em seguida.

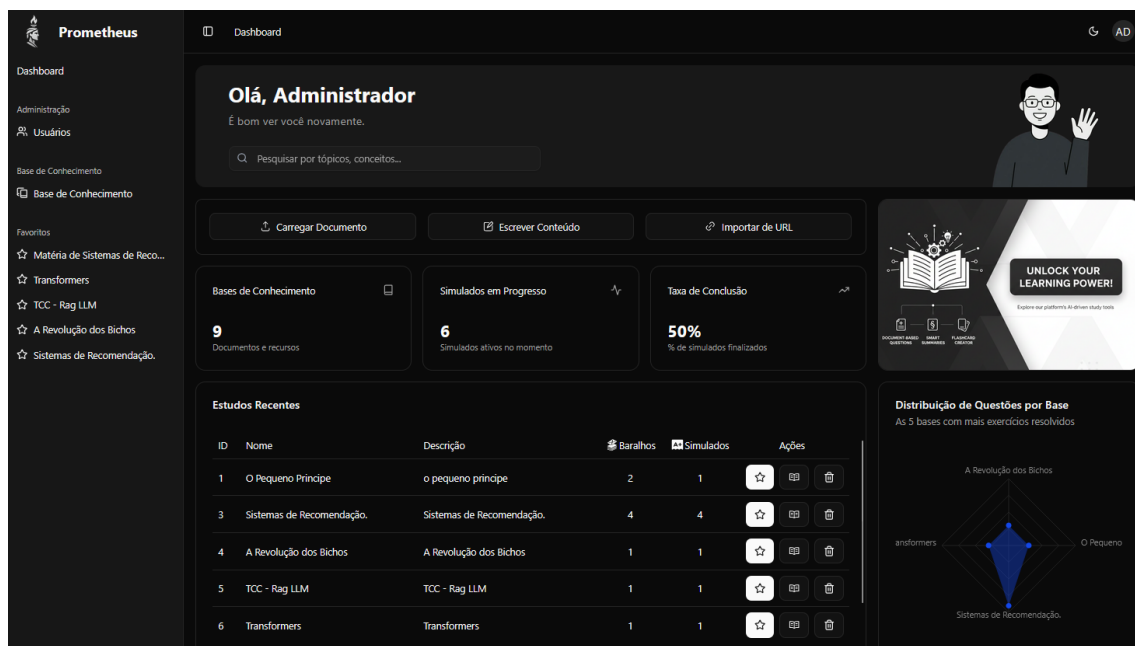
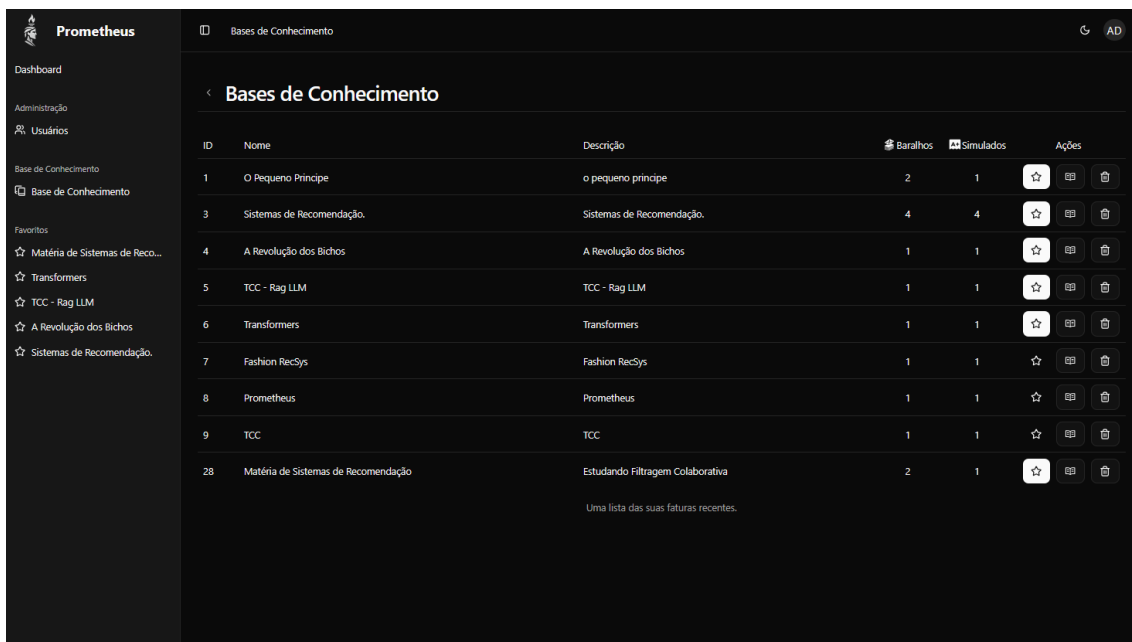


Figura 4.2: Tela inicial do Prometheus com visão geral das bases, simulados e indicadores de acompanhamento.

A partir dessa visão geral, o usuário pode acessar a listagem de bases de conhecimento, ilustrada na Figura 4.3. Essa tela representa o segundo nível do fluxo, pois transforma o conjunto de estudos recentes em uma visão tabular mais completa. Cada linha apresenta o identificador da base, nome, descrição, quantidade de *decks*,

quantidade de simulados e ações disponíveis. A presença desses dados permite comparar rapidamente diferentes materiais de estudo e identificar quais bases já possuem recursos derivados. Além disso, as ações de favoritar, acessar e excluir reforçam o papel da listagem como espaço de organização e manutenção dos documentos cadastrados.



ID	Nome	Descrição	Baralhos	Simulados	Ações
1	O Pequeno Príncipe	o pequeno principe	2	1	[Favoritar] [Editar] [Excluir]
3	Sistemas de Recomendação	Sistemas de Recomendação	4	4	[Favoritar] [Editar] [Excluir]
4	A Revolução dos Bichos	A Revolução dos Bichos	1	1	[Favoritar] [Editar] [Excluir]
5	TCC - Rag LLM	TCC - Rag LLM	1	1	[Favoritar] [Editar] [Excluir]
6	Transformers	Transformers	1	1	[Favoritar] [Editar] [Excluir]
7	Fashion RecSys	Fashion RecSys	1	1	[Favoritar] [Editar] [Excluir]
8	Prometheus	Prometheus	1	1	[Favoritar] [Editar] [Excluir]
9	TCC	TCC	1	1	[Favoritar] [Editar] [Excluir]
28	Matéria de Sistemas de Recomendação	Estudando Filtragem Colaborativa	2	1	[Favoritar] [Editar] [Excluir]

Uma lista das suas faturas recentes.

Figura 4.3: Listagem das bases de conhecimento cadastradas e seus respectivos recursos associados.

Ao selecionar uma base específica, o sistema conduz o estudante para a tela de detalhes da base de conhecimento, apresentada na Figura 4.4. Nesse terceiro nível, a interface deixa de mostrar apenas a coleção de documentos e passa a concentrar as informações de um material específico. A página exibe indicadores próprios da base, como total de *flashcards*, total de simulados e percentual de questões corretas, além de organizar o conteúdo em abas de resumo, *decks* e simulados. Na aba de resumo, o estudante visualiza a síntese gerada a partir do documento e uma lista de tópicos principais. Essa estrutura conecta a etapa de processamento textual ao uso pedagógico da plataforma, pois mostra que o documento enviado foi transformado em conteúdo resumido, temas centrais e pontos de partida para novas atividades.

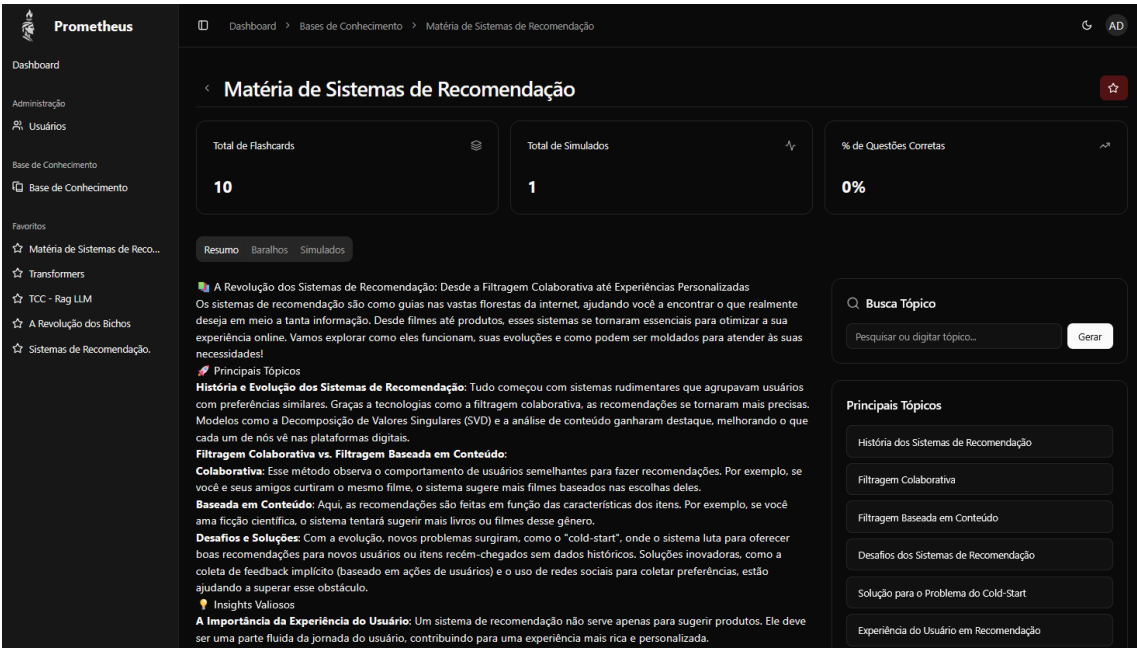


Figura 4.4: Tela de detalhes de uma base de conhecimento com resumo, indicadores e tópicos principais.

O acesso à aba de *decks* leva à organização dos *decks*, conforme mostrado na Figura 4.5. Nessa etapa, o sistema apresenta os conjuntos de cartões associados à base de conhecimento e oferece a criação de um novo *deck*. O painel lateral de criação permite informar nome, descrição e selecionar uma opção de geração, como criar automaticamente um baralho com dez *flashcards* ou criar um baralho vazio. Essa tela evidencia a transição entre o conteúdo processado e a prática de revisão: a base deixa de ser apenas um documento resumido e passa a originar materiais reutilizáveis para recuperação ativa.

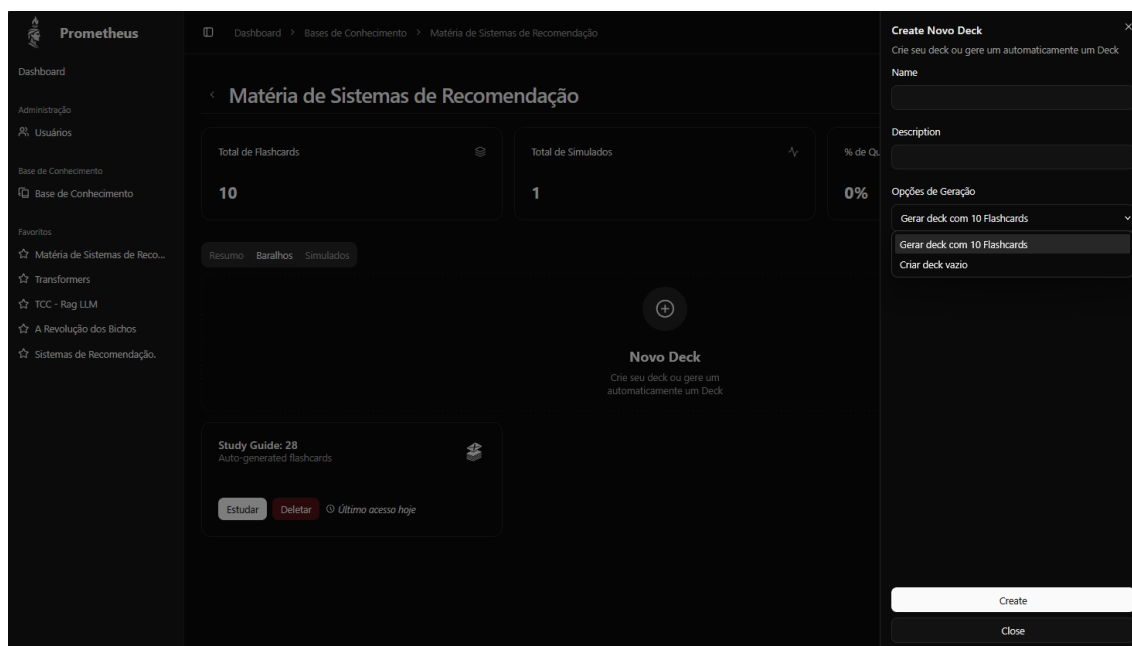


Figura 4.5: Aba de baralhos de uma base de conhecimento com opção de criação automática ou manual de *deck*.

Dentro de um *deck*, o estudante acessa a tela de *flashcards*, representada na Figura 4.6. Os cartões são apresentados em formato de grade, permitindo visualizar perguntas individualmente, revelar respostas e registrar a dificuldade percebida por meio das opções “Difícil”, “Médio” e “Fácil”. Os diferentes estados visuais indicam cartões ainda não avaliados e cartões já classificados, usando cores para facilitar o reconhecimento do progresso. A interface também permite criar novos cartões manualmente, por meio de um painel lateral com campos de frente e verso. Com isso, o sistema combina geração automática e intervenção do estudante, mantendo a revisão ativa vinculada à base de conhecimento de origem.

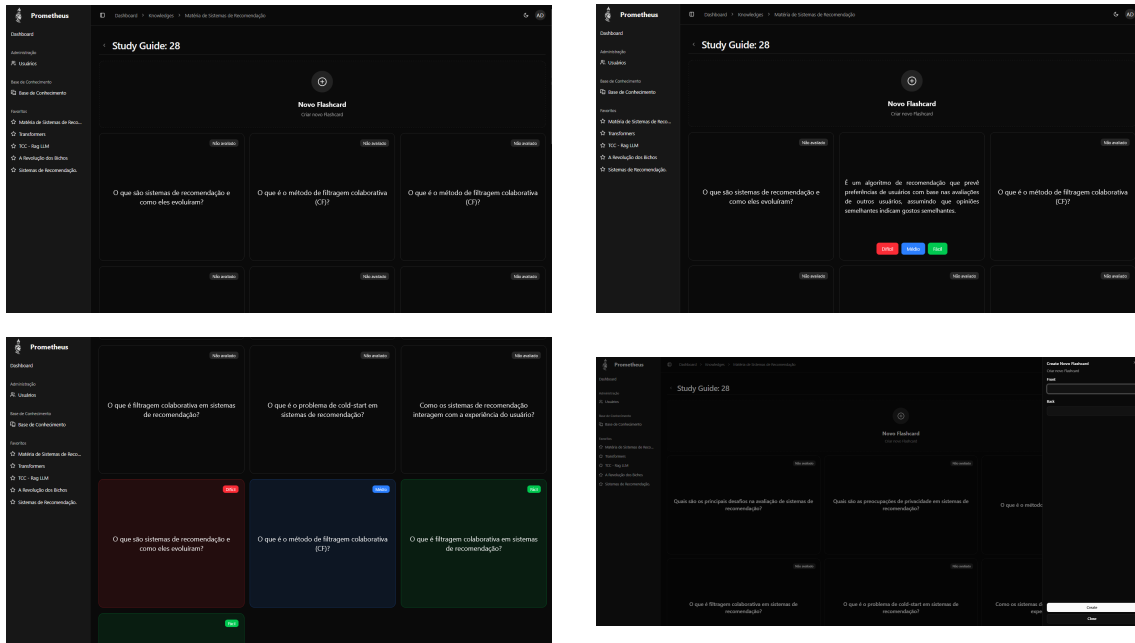


Figura 4.6: Estados da interface de *flashcards*: pergunta, resposta revelada, classificação por dificuldade e criação manual de cartão.

De forma complementar aos *flashcards*, a aba de simulados organiza as avaliações associadas à base de conhecimento. Como apresentado na Figura 4.7, essa tela exibe a criação de um novo simulado, a listagem dos simulados existentes, o estado de conclusão, o progresso e o resultado obtido. Essa organização permite que o estudante acompanhe não apenas a existência de uma avaliação, mas também seu desempenho nela, reforçando a proposta de acompanhamento supervisionado do aprendizado.

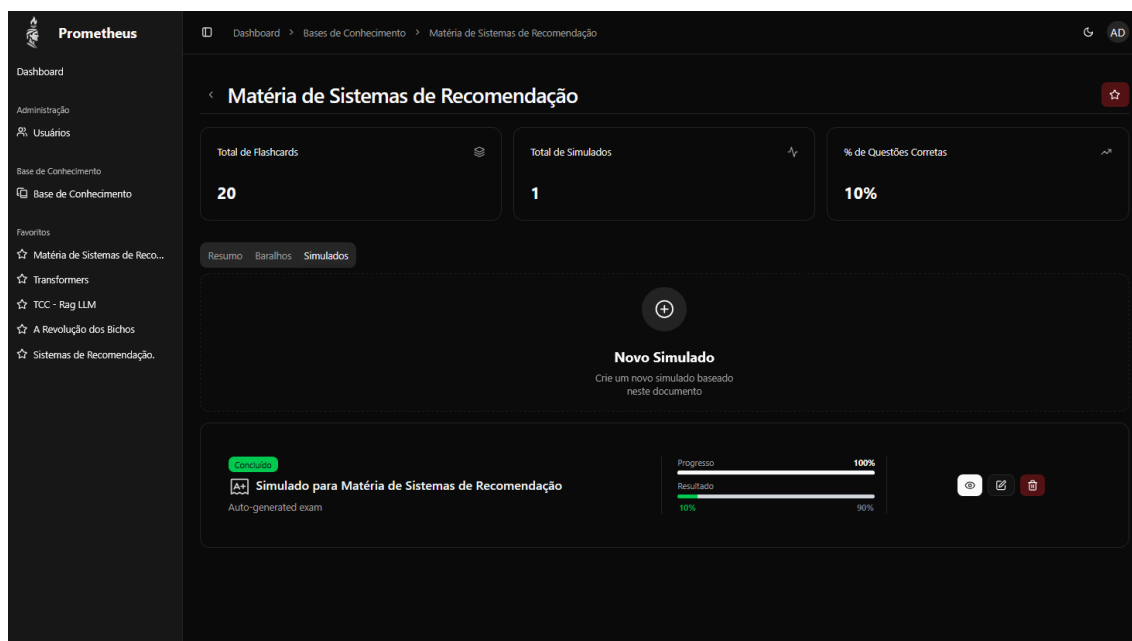


Figura 4.7: Aba de simulados com criação de nova avaliação, estado de conclusão, progresso e resultado.

Ao abrir um simulado, o sistema apresenta as questões geradas a partir do conteúdo recuperado da base. A Figura 4.8 ilustra a visualização de uma questão corrigida, na qual a alternativa selecionada incorretamente é destacada em vermelho, a resposta correta é destacada em verde e uma explicação pedagógica é exibida abaixo da alternativa correta. Essa tela fecha o fluxo *top-down* da interface: o estudante parte do painel geral, escolhe uma base, consulta seus materiais derivados, pratica com cartões e realiza avaliações que retornam evidências sobre seu desempenho. Assim, o simulado não funciona apenas como mecanismo de verificação, mas como recurso de aprendizagem, pois transforma o erro em informação útil para revisão.

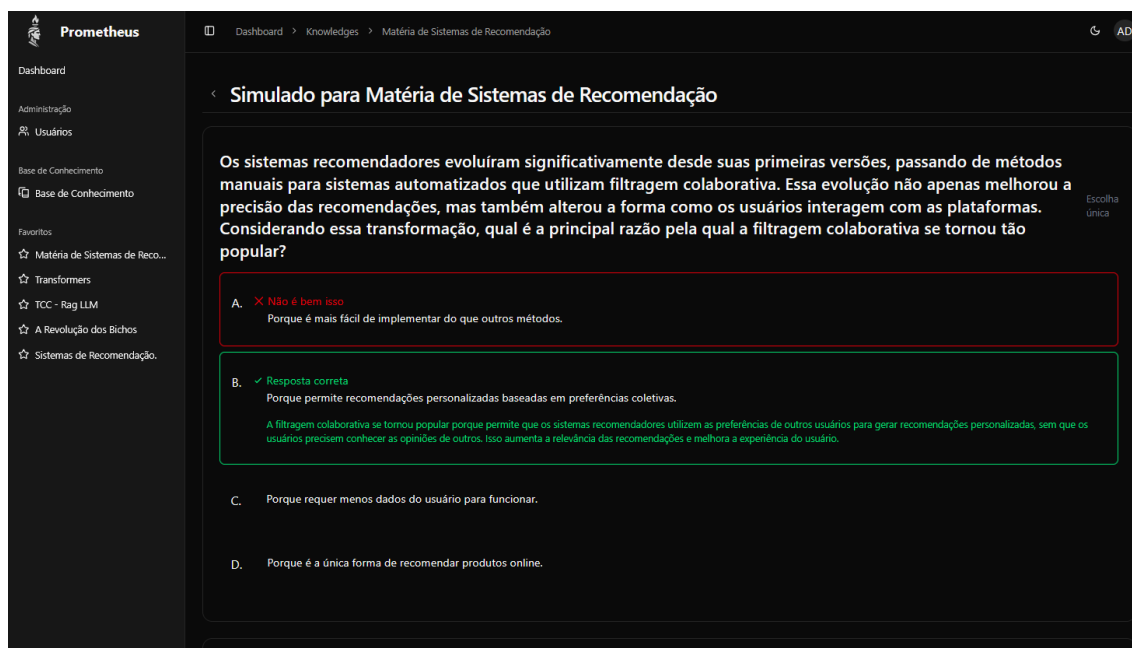


Figura 4.8: Tela de correção de simulado com indicação da alternativa incorreta, resposta correta e explicação do gabarito.

De modo geral, o fluxo implementado evidencia uma progressão coerente entre os níveis da aplicação. O *Dashboard* resume a situação global do estudante; a listagem de bases organiza os materiais disponíveis; a tela de uma base detalha resumo, tópicos e indicadores; os *decks* e *flashcards* apoiam a recuperação ativa; e os simulados oferecem avaliação e *feedback*. Essa sequência reforça o objetivo central do Prometheus, pois transforma documentos extensos em atividades de estudo menores, navegáveis e acompanhadas por indicadores de desempenho.

4.4.3 Resultado Obtido

Como resultado da implementação, o Prometheus consolida em uma única aplicação o ciclo de estudo proposto: organização do material, extração de conteúdo, síntese, geração de artefatos de prática e acompanhamento de desempenho. Do ponto de vista do estudante, o sistema transforma um documento extenso em recursos menores e acionáveis, como resumo, tópicos, simulados e *flashcards*. Do ponto de vista técnico, a aplicação integra uma arquitetura *web* modular, persistência relacional e vetorial, busca semântica e orquestração de LLMs.

Assim, o capítulo demonstra que as tecnologias adotadas não aparecem como elementos independentes, mas como partes de uma solução orientada ao objetivo central do Prometheus. O *starter kit* e o *monorepo* oferecem a fundação de desenvolvimento; o *AdonisJS* organiza regras, rotas e persistência operacional; o *InertiaJS* conecta servidor e interface; o *React* e o *shadcn/ui* sustentam a experiência visual; o PostgreSQL com *pgvector* viabiliza dados relacionais e busca semântica; e o *LangChain* coordena o fluxo de geração baseado em RAG. Em conjunto, essas escolhas permitem que a proposta conceitual apresentada anteriormente seja entregue como um sistema funcional, navegável e alinhado ao processo de aprendizagem pretendido.

Capítulo 5

Conclusão

Este capítulo apresenta as considerações finais deste trabalho, retomando a motivação inicial, os objetivos propostos e as principais contribuições obtidas a partir da concepção e do desenvolvimento do sistema Prometheus. Também são discutidas as limitações identificadas ao longo do projeto e possíveis direções para trabalhos futuros, considerando tanto a evolução técnica da plataforma quanto sua ampliação como ferramenta de apoio ao estudo supervisionado.

5.1 Considerações finais

Este trabalho partiu do estudo de que o aumento no acesso a materiais digitais de estudo não garante, por si só, um processo de aprendizagem efetivo. Embora livros técnicos, documentações, artigos científicos e materiais didáticos possam ser consultados com facilidade cada vez maior, estudantes de áreas científicas e tecnológicas ainda enfrentam dificuldades para transformar esse acesso em um estudo orientado, mensurável e progressivo. Em especial, a aprendizagem autônoma tende a depender fortemente da percepção subjetiva do próprio estudante, que nem sempre consegue identificar com clareza quais tópicos foram compreendidos, quais conteúdos ainda apresentam lacunas e quais revisões devem ser priorizadas.

Diante desse problema, o objetivo deste trabalho foi desenvolver um sistema

web de ensino supervisionado, apoiado por IA, voltado para o acompanhamento do aprendizado a partir de uma ou mais bases de conhecimento fornecidas pelo usuário. A proposta buscou integrar, em um mesmo ambiente, a importação e organização de materiais de estudo, a geração de avaliações alinhadas ao conteúdo selecionado, a análise do desempenho obtido, a apresentação de explicações e *feedback* e o apoio à revisão por meio de *flashcards*. Dessa forma, o Prometheus foi concebido para auxiliar o estudante a compreender não apenas o conteúdo estudado, mas também seu próprio progresso ao longo do processo de aprendizagem.

A plataforma desenvolvida contribui ao propor um fluxo de estudo que relaciona diretamente o material fornecido pelo usuário às atividades avaliativas e aos recursos de revisão gerados pelo sistema. Esse fluxo permite que o estudante importe uma base de conhecimento, utilize o conteúdo como referência para simulados e *flashcards*, receba retornos sobre suas respostas e identifique pontos que exigem maior atenção. Assim, o sistema busca reduzir a dependência exclusiva da autoavaliação subjetiva, oferecendo indicadores e devolutivas mais concretas sobre o desempenho do usuário.

Outro aspecto relevante do trabalho foi a aplicação de técnicas associadas a LLMs, *embeddings* e RAG para apoiar a geração de materiais contextualizados. A utilização de recuperação semântica permite que questões, explicações e cartões de estudo sejam produzidos a partir de trechos relacionados ao conteúdo enviado, favorecendo maior alinhamento entre os materiais gerados e a base de conhecimento original. Esse uso controlado de IA reforça a proposta de empregar modelos de linguagem como instrumentos de apoio ao estudo, sem dissociar as respostas geradas do material selecionado pelo usuário.

Além das contribuições funcionais, o desenvolvimento do Prometheus permitiu consolidar aprendizados relacionados à arquitetura de aplicações web que integram recursos de IA. A adoção de uma organização modular, associada ao uso de processamento assíncrono, banco de dados relacional, busca semântica e integração com modelos de linguagem, evidenciou a importância de separar responsabilidades entre os componentes do sistema. Essa estrutura favorece a manutenção da plataforma, reduz impactos entre módulos e oferece uma base mais adequada para futuras expansões

do sistema.

Assim, considera-se que o trabalho atingiu seu objetivo principal ao propor e estruturar uma plataforma capaz de integrar avaliação, *feedback* e revisão em um ciclo contínuo de estudo supervisionado. Ainda que o sistema possua limitações e demande aprimoramentos futuros, sua concepção demonstra o potencial de soluções apoiadas por IA para auxiliar estudantes na organização do estudo, na identificação de lacunas de conhecimento e no acompanhamento mais objetivo de sua aprendizagem.

5.2 Limitações e trabalhos futuros

Apesar dos resultados alcançados com a concepção e o desenvolvimento do sistema, algumas limitações devem ser consideradas. A primeira delas está relacionada ao escopo dos materiais suportados pela plataforma, que se concentra em documentos textuais. Embora esse recorte seja coerente com o objetivo inicial do trabalho, outros formatos relevantes para o estudo, como vídeos, apresentações, imagens, áudios e conteúdos multimodais, não foram contemplados nesta versão.

Outra limitação diz respeito à dependência de modelos de linguagem e de técnicas de recuperação semântica. Embora o uso de RAG contribua para aproximar as saídas geradas do conteúdo original, ainda podem ocorrer inconsistências, imprecisões ou variações na qualidade das questões, explicações e *flashcards* produzidos. Por esse motivo, permanece importante considerar mecanismos de revisão, validação e controle de qualidade dos materiais gerados, especialmente em contextos educacionais nos quais a precisão conceitual é essencial.

Outra possibilidade de evolução está relacionada ao aprimoramento do módulo de *flashcards* por meio de um algoritmo de repetição espaçada, como o *Free Spaced Repetition Scheduler* (FSRS). Atualmente, os cartões apoiam a revisão ativa do conteúdo, mas a priorização das revisões poderia ser tornada mais inteligente a partir do histórico de desempenho do estudante. Com a adoção de um algoritmo desse tipo, o sistema poderia estimar o melhor momento para reapresentar cada cartão, considerando fatores como acertos, erros, dificuldade percebida e tempo

decorrido desde a última revisão. Dessa forma, os *flashcards* deixariam de atuar apenas como um repositório de perguntas e respostas e passariam a compor uma estratégia adaptativa de estudo, favorecendo a retenção de longo prazo e o uso mais eficiente do tempo de revisão.

Por fim, se destaca a limitação relacionada à validação empírica e pedagógica da plataforma. Nesta etapa do trabalho, o foco esteve na concepção, modelagem e desenvolvimento do sistema, sem abranger uma validação ampla com usuários finais em um ambiente real de estudo. Assim, aspectos como usabilidade, percepção de utilidade, impacto no desempenho acadêmico, engajamento com os recursos de revisão e contribuição para a aprendizagem autorregulada ainda precisam ser investigados de forma mais aprofundada. Portanto, como trabalhos futuros, propõe-se a realização de estudos com estudantes, a fim de avaliar a experiência de uso da plataforma e verificar em que medida os recursos oferecidos contribuem para o acompanhamento do aprendizado. Esses estudos podem incluir questionários, entrevistas, análise de desempenho em simulados, observação do uso dos *flashcards* e comparação entre diferentes estratégias de estudo. Com isso, seria possível compreender melhor os impactos do Prometheus no processo de aprendizagem e identificar ajustes necessários a partir da experiência dos usuários.

Referências

ASAI, A. et al. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023. Disponível em: <<https://arxiv.org/abs/2310.11511>>.

CRUZ, B. M. d.; FERENC, A. V. F. Os processos avaliativos em educação: autonomia e autorregulação das aprendizagens. *Cordis: Revista Eletrônica de História Social da Cidade*, n. 35, p. 72775, 2025. Disponível em: <<https://revistas.pucsp.br/index.php/cordis/article/view/72775>>.

GAO, Y. et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2024. Disponível em: <<https://arxiv.org/abs/2312.10997>>.

HUANG, L. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, v. 43, n. 1, p. 1–38, 2025. Disponível em: <<https://arxiv.org/abs/2311.05232>>.

KRASNER, G. E.; POPE, S. T. *A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System*. Mountain View, 1988.

LAN, M.; ZHOU, X. A qualitative systematic review on AI empowered self-regulated learning in higher education. *npj Science of Learning*, v. 10, n. 21, 2025. Disponível em: <<https://www.nature.com/articles/s41539-025-00319-0>>.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: *Advances in Neural Information Processing Systems*. [s.n.], 2020. v. 33, p. 9459–9474. Disponível em: <<https://arxiv.org/abs/2005.11401>>.

LIU, N. F. et al. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, v. 12, p. 157–173, 2024. Disponível em: <https://doi.org/10.1162/tacl_a_00638>.

SOMMERVILLE, I. *Engenharia de Software*. 10. ed. São Paulo: Pearson Education do Brasil, 2019.

VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [s.n.], 2017. v. 30. Disponível em: <<https://arxiv.org/abs/1706.03762>>.

Apêndice A

Especificação de Casos de Uso

Este apêndice apresenta a especificação dos casos de uso representados no Diagrama de Casos de Uso do Prometheus. Considera-se caso de uso uma representação abstrata da interação entre um sistema e seus usuários, descrevendo como o sistema responde às ações executadas pelo ator. Essa especificação apoia o detalhamento dos requisitos ao identificar as funcionalidades, o ator envolvido, os cenários de interação, as pré-condições e as pós-condições de cada funcionalidade.

As descrições a seguir consideram o Estudante como ator principal e foram elaboradas de acordo com as interações previstas para o Prometheus: cadastro e gerenciamento de usuário, criação e consulta de bases de conhecimento, envio de documentos, uso de tópicos, geração de simulados, revisão por *decks* e interação com *flashcards*.

- **UC-01: Cadastrar Usuário**

Ator: Estudante.

Descrição: O Estudante pode criar uma conta no Prometheus informando seus dados básicos de identificação e suas credenciais de acesso.

Pré-condição: O Estudante não deve estar autenticado no sistema.

Fluxo principal:

1. O Estudante solicita cadastro na plataforma.

2. O sistema solicita os dados para a criação de conta.
3. O Estudante informa nome completo, e-mail, senha e confirmação de senha.
4. O sistema valida os dados obrigatórios, o formato do e-mail e a complexidade da senha.
5. O sistema verifica se o e-mail informado ainda não está associado a outro usuário.
6. O sistema cadastra o usuário e confirma a operação.
7. O sistema direciona o Estudante para o acesso inicial à plataforma.
8. Fim do caso de uso.

Fluxos de Exceção:

4.a: Campos obrigatórios não preenchidos ou inválidos.

1. O sistema impede o cadastro e exibe uma mensagem indicando os campos que precisam ser corrigidos.
2. Retorna ao passo 3 do Fluxo Principal.

4.b: Senha fora do padrão mínimo de segurança.

1. O sistema informa que a senha deve possuir no mínimo 8 caracteres, contendo ao menos uma letra e um número.
2. Retorna ao passo 3 do Fluxo Principal.

5.a: E-mail já cadastrado.

1. O sistema rejeita a operação e informa que o e-mail já está em uso.
2. Retorna ao passo 3 do Fluxo Principal.

Pós-condição: A conta do Estudante é criada e fica disponível para autenticação no Prometheus.

● UC-02: Gerenciar Usuário

Ator: Estudante.

Descrição: O Estudante pode visualizar e atualizar suas informações de perfil, incluindo dados cadastrais, senha e imagem de avatar.

Pré-condição: O Estudante deve estar autenticado no sistema.

Fluxo principal:

1. O Estudante acessa a área de gerenciamento de perfil.
2. O sistema exibe os dados atuais do usuário.
3. O Estudante solicita a edição das informações desejadas.
4. O sistema exibe os campos editáveis do perfil.
5. O Estudante altera nome, e-mail, senha ou avatar e confirma a operação.
6. O sistema valida os dados informados.
7. O sistema atualiza o perfil e confirma a alteração.
8. Fim do caso de uso.

Fluxos de Exceção:

6.a: Dados inválidos ou obrigatórios não preenchidos.

1. O sistema impede a atualização e informa os campos que precisam ser corrigidos.
2. Retorna ao passo 4 do Fluxo Principal.

6.b: E-mail informado pertence a outro usuário.

1. O sistema rejeita a atualização e informa a duplicidade do e-mail.
2. Retorna ao passo 4 do Fluxo Principal.

6.c: Nova senha não atende à regra de complexidade.

1. O sistema informa o padrão mínimo exigido para a senha.
2. Retorna ao passo 4 do Fluxo Principal.

Pós-condição: As informações do Estudante são atualizadas e passam a ser utilizadas nas próximas interações com o sistema.

- **UC-03: Criar Base de Conhecimento**

Ator: Estudante.

Descrição: O Estudante pode criar uma base de conhecimento informando nome, objetivo de estudo e documento em PDF. A base reúne o material enviado, o resumo, os tópicos sugeridos e os materiais de estudo derivados.

Pré-condição: O Estudante deve estar autenticado no sistema.

Fluxo principal:

1. O Estudante acessa a tela inicial.
2. O Estudante solicita a criação de uma nova base de conhecimento ao clicar no botão “Carregar Documento”.
3. O sistema exibe o formulário de criação.
4. O Estudante informa nome e objetivo de estudo da base.
5. O Estudante executa o caso de uso **UC-04: Fazer Upload de Documento**.
6. O sistema confere se as informações da base e o documento atendem às regras da plataforma.
7. O sistema cria a base e informa que o material está sendo preparado.
8. O sistema analisa o documento enviado para produzir o resumo e os tópicos sugeridos.
9. O sistema redireciona o Estudante para a tela da base criada, exibindo a situação atual da preparação.
10. Ao concluir a preparação, o sistema disponibiliza resumo, tópicos sugeridos e recursos para criação de simulados e *decks*.
11. Opcionalmente, após a base estar disponível, o Estudante pode executar o caso de uso **UC-05: Gerar Resumo por Tópico Personalizado**.
12. Fim do caso de uso.

Fluxos de Exceção:

6.a: Nome ou objetivo de estudo não preenchido.

1. O sistema impede a criação da base e informa os campos obrigatórios pendentes.
2. Retorna ao passo 4 do Fluxo Principal.

6.b: Documento ausente ou inválido.

1. O sistema informa que é necessário enviar um documento válido em PDF.
2. Retorna ao passo 5 do Fluxo Principal.

6.c: Documento excede os limites permitidos.

1. O sistema rejeita o envio quando o arquivo ultrapassa 100 MB ou 500 páginas.
2. Retorna ao passo 5 do Fluxo Principal.

8.a: Documento não pode ser analisado pelo sistema.

1. O sistema informa que o documento não possui texto legível suficiente ou não pôde ser analisado adequadamente.
2. O Estudante pode excluir a base ou criar uma nova base com outro documento.

Pós-condição: A base de conhecimento é cadastrada e fica disponível para consulta, simulados e *flashcards* quando a preparação do material é concluída com sucesso.

● **UC-04: Fazer Upload de Documento**

Ator: Estudante.

Descrição: O Estudante pode anexar o documento que alimentará uma base de conhecimento. Esse caso de uso é incluído obrigatoriamente no caso de uso

UC-03: Criar Base de Conhecimento.

Pré-condição: O Estudante deve estar autenticado e estar no fluxo de criação de uma base de conhecimento.

Fluxo principal:

1. O sistema exibe o campo para seleção de arquivo.
2. O Estudante seleciona um documento em PDF em seu dispositivo.

3. O sistema associa o arquivo selecionado ao formulário de criação da base.
4. O Estudante confirma o envio juntamente com os demais dados da base.
5. O sistema confirma o recebimento do arquivo selecionado.
6. O sistema valida formato, tamanho e quantidade de páginas do documento.
7. O sistema vincula o documento à base de conhecimento em criação.
8. Fim do caso de uso.

Fluxos de Exceção:**6.a: Arquivo não é um PDF válido.**

1. O sistema rejeita o arquivo e solicita a seleção de um documento compatível.
2. Retorna ao passo 1 do Fluxo Principal.

6.b: Arquivo excede os limites de tamanho ou páginas.

1. O sistema informa os limites permitidos para upload.
2. Retorna ao passo 1 do Fluxo Principal.

Pós-condição: O documento fica associado à base de conhecimento em criação e pronto para ser analisado pelo sistema.

● UC-05: Gerar Resumo por Tópico Personalizado

Ator: Estudante.

Descrição: Após acessar uma base de conhecimento já criada, o Estudante pode digitar um tópico personalizado no campo de busca de tópico para solicitar um resumo focado nesse assunto. Esse tópico é informado livremente pelo Estudante e não precisa fazer parte dos tópicos sugeridos automaticamente pelo sistema.

Pré-condição: O Estudante deve estar autenticado, a base de conhecimento deve existir e o resumo geral da base deve estar disponível.

Fluxo principal:

1. O Estudante acessa uma base de conhecimento já criada.

2. O sistema exibe a aba de resumo da base, os principais tópicos sugeridos e o campo de busca de tópico personalizado.
3. O Estudante digita no campo de texto o tópico personalizado sobre o qual deseja estudar.
4. O Estudante solicita a geração ao clicar no botão “Gerar”.
5. O sistema verifica se o tópico informado é suficiente para orientar a geração.
6. O sistema gera um resumo focado no tópico personalizado, utilizando o conteúdo da base de conhecimento.
7. O sistema exibe o resumo gerado na tela da base de conhecimento.
8. Fim do caso de uso.

Fluxos de Exceção:

5.a: Tópico vazio ou pouco descritivo.

1. O sistema informa que é necessário digitar um tópico compreensível para gerar o resumo.
2. Retorna ao passo 3 do Fluxo Principal.

6.a: Resumo não pode ser gerado.

1. O sistema informa que não foi possível gerar o resumo para o tópico informado naquele momento.
2. O Estudante pode tentar novamente ou digitar outro tópico personalizado.

Pós-condição: Um resumo focado no tópico personalizado é exibido para o Estudante na base de conhecimento.

● UC-06: Gerenciar Base de Conhecimento

Ator: Estudante.

Descrição: O Estudante pode consultar a listagem de suas bases, editar informações, excluir bases e acionar operações complementares de busca e favoritos.

Pré-condição: O Estudante deve estar autenticado no sistema.

Fluxo principal:

1. O Estudante acessa a área de bases de conhecimento.
2. O sistema lista as bases vinculadas ao Estudante, indicando nome, objetivo, situação da base e marcação de favorito quando existente.
3. O Estudante seleciona uma operação de gerenciamento sobre uma base.
4. O sistema permite editar informações, excluir a base, executar o caso de uso **UC-07: Buscar Base de Conhecimento** ou executar o caso de uso **UC-08: Favoritar Base de Conhecimento**.
5. O Estudante confirma a operação escolhida.
6. O sistema executa a operação e atualiza a listagem.
7. Fim do caso de uso.

Fluxos de Exceção:

2.a: Nenhuma base cadastrada.

1. O sistema informa que ainda não existem bases de conhecimento cadastradas.
2. O sistema oferece a opção de criar uma nova base.

5.a: Exclusão de base não confirmada.

1. O sistema cancela a exclusão e mantém a base inalterada.
2. Retorna ao passo 2 do Fluxo Principal.

6.a: Falha ao editar ou excluir a base.

1. O sistema informa que a operação não pôde ser concluída.
2. Retorna ao passo 2 do Fluxo Principal.

Pós-condição: A listagem de bases permanece atualizada conforme a operação realizada. Em caso de exclusão, os simulados, *decks*, *flashcards* e históricos associados à base também são removidos.

• UC-07: Buscar Base de Conhecimento

Ator: Estudante.

Descrição: O Estudante pode localizar bases de conhecimento por termos relacionados ao nome, objetivo ou conteúdo apresentado na listagem. Esse caso de uso estende o caso de uso **UC-06: Gerenciar Base de Conhecimento**.

Pré-condição: O Estudante deve estar autenticado e possuir acesso à listagem de bases de conhecimento.

Fluxo principal:

1. O Estudante acessa o campo de busca na tela inicial.
2. O Estudante informa o termo de busca.
3. O sistema consulta as bases associadas ao Estudante que correspondem ao termo informado.
4. O sistema exibe a listagem filtrada.
5. O Estudante pode limpar a busca, refinar o termo ou selecionar uma base retornada.
6. Fim do caso de uso.

Fluxos de Exceção:

3.a: Termo de busca vazio.

1. O sistema mantém ou restaura a listagem completa de bases.
2. Retorna ao passo 1 do Fluxo Principal.

4.a: Nenhuma base encontrada.

1. O sistema informa que não há resultados para o termo pesquisado.
2. O Estudante pode alterar o termo de busca.

Pós-condição: A tela apresenta apenas as bases correspondentes ao filtro informado ou retorna à listagem completa quando a busca é limpa.

● **UC-08: Favoritar Base de Conhecimento**

Ator: Estudante.

Descrição: O Estudante pode marcar ou desmarcar uma base como favorita para facilitar o acesso aos materiais mais relevantes. Esse caso de uso estende o caso de uso **UC-06: Gerenciar Base de Conhecimento**.

Pré-condição: O Estudante deve estar autenticado e a base de conhecimento deve pertencer ao seu conjunto de bases acessíveis.

Fluxo principal:

1. O Estudante visualiza uma base de conhecimento na listagem.
2. O Estudante aciona a opção de favoritar ou remover dos favoritos.
3. O sistema verifica se a base já está marcada como favorita.
4. O sistema cria ou remove a associação de favorito entre o Estudante e a base.
5. O sistema atualiza a interface, destacando a nova marcação da base.
6. Fim do caso de uso.

Fluxos de Exceção:

3.a: Base não encontrada ou indisponível.

1. O sistema informa que a base não pode ser favoritada no momento.
2. Retorna à listagem de bases.

4.a: Falha ao atualizar favorito.

1. O sistema mantém a marcação anterior da base e exibe mensagem de erro.
2. Retorna ao passo 1 do Fluxo Principal.

Pós-condição: A marcação de favorito da base é atualizada e refletida nas listagens do Estudante.

● **UC-09: Acessar Base de Conhecimento**

Ator: Estudante.

Descrição: O Estudante pode abrir uma base de conhecimento para consultar seu resumo, tópicos, situação atual, materiais derivados, simulados e *decks*.

Pré-condição: O Estudante deve estar autenticado e a base de conhecimento deve existir.

Fluxo principal:

1. O Estudante seleciona uma base na listagem de bases de conhecimento.
2. O sistema verifica se a base está disponível para o Estudante.
3. O sistema exibe a tela da base selecionada.
4. O sistema apresenta a situação atual da base, o resumo gerado, os tópicos sugeridos e os indicadores disponíveis.
5. O sistema disponibiliza as abas ou áreas de acesso a simulados, *decks* e *flashcards* vinculados à base.
6. Opcionalmente, o Estudante executa o caso de uso **UC-10: Selecionar Tópico Sugerido**.
7. Fim do caso de uso.

Fluxos de Exceção:

2.a: Base inexistente ou sem permissão de acesso.

1. O sistema impede o acesso e informa que a base não foi encontrada.
2. Retorna à listagem de bases.

4.b: Preparação da base falhou.

1. O sistema informa que não foi possível preparar a base e não libera os materiais que dependem dessa preparação.
2. O Estudante pode excluir a base ou criar outra base com documento diferente.

Pós-condição: A base fica aberta para consulta e para execução de operações relacionadas a tópicos, simulados, *decks* e *flashcards*.

● **UC-10: Selecionar Tópico Sugerido**

Ator: Estudante.

Descrição: O Estudante pode selecionar um dos tópicos sugeridos automaticamente pelo Prometheus para orientar sua navegação ou a criação de materiais de estudo. Esse caso de uso estende o caso de uso **UC-09: Acessar Base de Conhecimento**.

Pré-condição: O Estudante deve estar autenticado, a base deve estar disponível e os tópicos sugeridos devem estar visíveis.

Fluxo principal:

1. O sistema apresenta a lista de tópicos sugeridos para a base de conhecimento.
2. O Estudante seleciona um tópico sugerido.
3. O sistema destaca o tópico selecionado na interface.
4. O sistema apresenta as próximas ações de estudo usando o tópico como referência, quando aplicável.
5. Fim do caso de uso.

Fluxos de Exceção:

1.a: Tópicos ainda não disponíveis.

1. O sistema informa que a base ainda está em preparação ou que não foi possível gerar tópicos.
2. Retorna à tela da base de conhecimento.

2.a: Tópico selecionado não pertence à base.

1. O sistema ignora a seleção e recarrega os tópicos válidos da base.
2. Retorna ao passo 1 do Fluxo Principal.

Pós-condição: O tópico selecionado passa a orientar as interações subsequentes relacionadas à base.

• **UC-11: Criar Simulado**

Ator: Estudante.

Descrição: O Estudante pode criar um simulado associado a uma base de conhecimento. O sistema gera automaticamente questões de múltipla escolha com alternativas, resposta correta e explicação pedagógica.

Pré-condição: O Estudante deve estar autenticado, a base de conhecimento deve existir e o material da base deve estar disponível para uso.

Fluxo principal:

1. O Estudante acessa a aba de simulados de uma base de conhecimento.
2. O sistema lista os simulados existentes e exibe a opção de criar novo simulado.
3. O Estudante solicita a criação de um simulado.
4. O sistema solicita nome, descrição ou foco do simulado quando aplicável.
5. O Estudante confirma a criação.
6. O sistema cria o simulado e informa que as questões estão sendo geradas.
7. O sistema utiliza o conteúdo da base de conhecimento para gerar as questões.
8. O sistema gera exatamente 10 questões de múltipla escolha, com alternativas, gabarito e explicação.
9. O sistema disponibiliza o simulado com as questões geradas.
10. O sistema atualiza a listagem de simulados da base.
11. Fim do caso de uso.

Fluxos de Exceção:

5.a: Dados obrigatórios do simulado não preenchidos.

1. O sistema informa os campos que precisam ser preenchidos.
2. Retorna ao passo 4 do Fluxo Principal.

7.a: Base sem conteúdo suficiente para geração.

1. O sistema cancela a geração e informa que a base não possui conteúdo suficiente para criar o simulado.
2. Retorna à aba de simulados.

8.a: Falha na geração automática.

1. O sistema informa que não foi possível gerar o simulado naquele momento.
2. O Estudante pode solicitar uma nova criação posteriormente.

Pós-condição: Um simulado com 10 questões é disponibilizado para acesso dentro da base de conhecimento.

• UC-12: Excluir Simulado

Ator: Estudante.

Descrição: O Estudante pode remover um simulado associado a uma base de conhecimento, incluindo suas questões e alternativas.

Pré-condição: O Estudante deve estar autenticado e o simulado deve existir em uma base acessível.

Fluxo principal:

1. O Estudante acessa a aba de simulados de uma base de conhecimento.
2. O sistema lista os simulados existentes.
3. O Estudante solicita a exclusão de um simulado.
4. O sistema solicita confirmação da exclusão.
5. O Estudante confirma a operação.
6. O sistema remove o simulado, suas questões, alternativas e registros associados.
7. O sistema atualiza a listagem de simulados.
8. Fim do caso de uso.

Fluxos de Exceção:**5.a: Exclusão não confirmada.**

1. O sistema cancela a operação e mantém o simulado na base.
2. Retorna ao passo 2 do Fluxo Principal.

6.a: Simulado não encontrado.

1. O sistema informa que o simulado não está mais disponível.
2. O sistema atualiza a listagem de simulados.

Pós-condição: O simulado deixa de aparecer na base de conhecimento e seus dados associados são removidos.

- **UC-13: Acessar Simulado**

Ator: Estudante.

Descrição: O Estudante pode abrir um simulado para visualizar suas questões, acompanhar seu progresso ou consultar o resultado quando já finalizado.

Pré-condição: O Estudante deve estar autenticado e o simulado deve estar vinculado a uma base acessível.

Fluxo principal:

1. O Estudante acessa a aba de simulados de uma base.
2. O sistema apresenta a lista de simulados, suas situações de conclusão, progresso e resultado quando disponível.
3. O Estudante seleciona um simulado.
4. O sistema carrega as questões e alternativas do simulado.
5. O sistema exibe as questões para resposta ou, se o simulado estiver finalizado, exibe a correção e as explicações do gabarito.
6. Fim do caso de uso.

Fluxos de Exceção:

3.a: Simulado ainda em preparação.

1. O sistema informa que as questões ainda estão sendo geradas.
2. O Estudante pode aguardar a atualização na tela da base de conhecimento.

Pós-condição: O simulado fica aberto para resposta ou consulta do resultado.

- **UC-14: Responder Simulado**

Ator: Estudante.

Descrição: O Estudante pode responder às questões de um simulado e receber correção com explicação da alternativa correta.

Pré-condição: O Estudante deve estar autenticado, o simulado deve estar disponível e ainda não deve ter sido finalizado.

Fluxo principal:

1. O Estudante acessa um simulado disponível.
2. O sistema exibe as questões e alternativas de múltipla escolha.
3. O Estudante seleciona uma alternativa para cada questão.
4. O Estudante submete suas respostas.
5. O sistema valida se as questões foram respondidas.
6. O sistema registra as respostas do Estudante.
7. O sistema calcula o resultado, identifica acertos e erros e registra o desempenho.
8. O sistema bloqueia o simulado para novas tentativas.
9. O sistema exibe a correção, destacando a alternativa escolhida, a resposta correta e a explicação pedagógica do gabarito.
10. Fim do caso de uso.

Fluxos de Exceção:

6.a: Falha ao registrar respostas.

1. O sistema informa que não foi possível salvar as respostas no momento.
2. O Estudante pode tentar submeter novamente antes de sair da tela.

Pós-condição: As respostas e o desempenho do Estudante são registrados, e o simulado passa a ficar bloqueado para novas tentativas.

● **UC-15: Criar Deck**

Ator: Estudante.

Descrição: O Estudante pode criar um *deck* de *flashcards* associado a uma base de conhecimento, de forma automática ou com inclusão manual de cartões.

Pré-condição: O Estudante deve estar autenticado, a base de conhecimento deve existir e o material da base deve estar disponível quando houver geração automática.

Fluxo principal:

1. O Estudante acessa a aba de *decks* de uma base de conhecimento.
2. O sistema lista os *decks* existentes e exibe a opção de criação.
3. O Estudante solicita a criação de um novo *deck*.
4. O sistema solicita nome, descrição e modo de criação.
5. O Estudante escolhe criar um *deck* automático ou vazio para preenchimento manual.
6. O sistema cria o *deck* associado à base.
7. Se a criação for automática, o sistema utiliza o conteúdo da base para gerar até 10 *flashcards* com frente e verso.
8. Se a criação for manual, o Estudante pode executar o caso de uso **UC-16: Criar Flashcard**.
9. O sistema atualiza a listagem de *decks*.
10. Fim do caso de uso.

Fluxos de Exceção:

4.a: Dados obrigatórios do *deck* não preenchidos.

1. O sistema informa os campos pendentes.
2. Retorna ao passo 4 do Fluxo Principal.

7.a: Base sem conteúdo suficiente para geração automática.

1. O sistema informa que não foi possível gerar *flashcards* automaticamente.
2. O Estudante pode manter o *deck* vazio e criar cartões manualmente.

7.b: Falha na geração automática.

1. O sistema informa que não foi possível gerar os cartões automaticamente e mantém o *deck* disponível para criação manual, quando possível.
2. Retorna à aba de *decks*.

Pós-condição: Um *deck* é criado na base de conhecimento e passa a reunir *flashcards* gerados automaticamente ou criados manualmente.

- **UC-16: Criar Flashcard**

Ator: Estudante.

Descrição: O Estudante pode criar manualmente um *flashcard*, informando o conteúdo da frente e do verso do cartão. Esse caso de uso estende o caso de uso **UC-15: Criar Deck**.

Pré-condição: O Estudante deve estar autenticado e deve existir um *deck* acessível para receber o cartão.

Fluxo principal:

1. O Estudante acessa a criação manual de *flashcard* durante a criação do *deck* ou dentro de um *deck* existente.
2. O sistema exibe os campos de frente e verso do cartão.
3. O Estudante informa a pergunta, conceito ou estímulo na frente do cartão.
4. O Estudante informa a resposta ou explicação no verso do cartão.
5. O sistema valida os campos preenchidos.
6. O sistema salva o *flashcard* no *deck* correspondente.
7. O sistema atualiza a quantidade de cartões do *deck*.
8. Fim do caso de uso.

Fluxos de Exceção:

5.a: Frente ou verso do cartão não preenchido.

1. O sistema impede a criação e informa que ambos os campos são obrigatórios.
2. Retorna ao passo 2 do Fluxo Principal.

6.a: Deck não encontrado.

1. O sistema informa que o cartão não pôde ser associado a um *deck* válido.
2. Retorna à aba de *decks*.

Pós-condição: O *flashcard* é adicionado ao *deck* e fica disponível para revisão.

• UC-17: Acessar Deck

Ator: Estudante.

Descrição: O Estudante pode abrir um *deck* para visualizar seus *flashcards*, acompanhar a situação dos cartões e iniciar a revisão.

Pré-condição: O Estudante deve estar autenticado e o *deck* deve existir em uma base acessível.

Fluxo principal:

1. O Estudante acessa a aba de *decks* de uma base de conhecimento.
2. O sistema lista os *decks* vinculados à base.
3. O Estudante seleciona um *deck*.
4. O sistema exibe os *flashcards* do *deck*.
5. O sistema apresenta os cartões em grade, indicando perguntas e classificações de dificuldade quando existentes.
6. O Estudante pode executar o caso de uso **UC-18: Excluir Flashcard** ou iniciar o caso de uso **UC-20: Interagir com Flashcard**.
7. Fim do caso de uso.

Fluxos de Exceção:**3.a: Deck ainda em preparação.**

1. O sistema informa que os cartões ainda estão sendo preparados.
2. O Estudante pode aguardar a atualização ou retornar à aba de *decks*.

4.a: Deck sem flashcards.

1. O sistema informa que ainda não há cartões cadastrados.
2. O sistema oferece a opção de criar *flashcards* manualmente.

Pós-condição: O *deck* fica aberto para revisão, criação manual ou exclusão de *flashcards*.

● **UC-18: Excluir Flashcard**

Ator: Estudante.

Descrição: O Estudante pode excluir um *flashcard* específico de um *deck*. Esse caso de uso estende o caso de uso **UC-17: Acessar Deck**.

Pré-condição: O Estudante deve estar autenticado, o *deck* deve estar aberto e o *flashcard* deve existir.

Fluxo principal:

1. O Estudante visualiza os *flashcards* de um *deck*.
2. O Estudante seleciona a opção de excluir um cartão.
3. O sistema solicita confirmação da exclusão.
4. O Estudante confirma a operação.
5. O sistema remove o *flashcard* do *deck*.
6. O sistema atualiza a quantidade de cartões e a visualização do *deck*.
7. Fim do caso de uso.

Fluxos de Exceção:

4.a: Exclusão não confirmada.

1. O sistema cancela a operação e mantém o *flashcard* no *deck*.
2. Retorna ao passo 1 do Fluxo Principal.

5.a: *Flashcard* não encontrado.

1. O sistema informa que o cartão não está mais disponível.
2. O sistema atualiza a lista de cartões do *deck*.

Pós-condição: O *flashcard* é removido do *deck* e deixa de aparecer na revisão.

● **UC-19: Excluir Deck**

Ator: Estudante.

Descrição: O Estudante pode excluir um *deck* vinculado a uma base de conhecimento, removendo também todos os *flashcards* associados.

Pré-condição: O Estudante deve estar autenticado e o *deck* deve existir em uma base acessível.

Fluxo principal:

1. O Estudante acessa a aba de *decks* de uma base de conhecimento.
2. O sistema lista os *decks* existentes.
3. O Estudante solicita a exclusão de um *deck*.
4. O sistema solicita confirmação da exclusão.
5. O Estudante confirma a operação.
6. O sistema remove o *deck* e todos os seus *flashcards* associados.
7. O sistema atualiza a listagem de *decks*.
8. Fim do caso de uso.

Fluxos de Exceção:

5.a: Exclusão não confirmada.

1. O sistema cancela a operação e mantém o *deck* inalterado.
2. Retorna ao passo 2 do Fluxo Principal.

6.a: Deck não encontrado.

1. O sistema informa que o *deck* não está mais disponível.
2. O sistema atualiza a listagem de *decks*.

Pós-condição: O *deck* e seus *flashcards* são removidos da base de conhecimento.

- **UC-20: Interagir com Flashcard**

Ator: Estudante.

Descrição: O Estudante pode revisar um *flashcard*, visualizar sua frente, revelar o verso e comparar sua resposta mental com a explicação apresentada.

Pré-condição: O Estudante deve estar autenticado, deve ter acessado um *deck* e deve existir ao menos um *flashcard* para revisão.

Fluxo principal:

1. O sistema exibe a frente do *flashcard* ao Estudante.
2. O Estudante lê a pergunta, conceito ou estímulo apresentado no cartão.
3. O Estudante solicita a revelação do verso do cartão.
4. O sistema exibe a resposta ou explicação no verso.
5. O Estudante compara sua lembrança com o conteúdo exibido.
6. Opcionalmente, o Estudante executa o caso de uso **UC-21: Classificar Dificuldade do Flashcard**.
7. O sistema mantém o cartão na lista de revisão do *deck*.
8. Fim do caso de uso.

Fluxos de Exceção:

1.a: Nenhum *flashcard* disponível.

1. O sistema informa que o *deck* não possui cartões para revisão.
2. O Estudante pode criar novos *flashcards* ou retornar à lista de *decks*.

4.a: Cartão não pode ser exibido.

1. O sistema informa que o conteúdo do cartão não pôde ser exibido.
2. Retorna à visualização do *deck*.

Pós-condição: O Estudante revisa o conteúdo do cartão e pode registrar sua percepção de dificuldade.

- **UC-21: Classificar Dificuldade do Flashcard**

Ator: Estudante.

Descrição: O Estudante pode classificar a dificuldade percebida de um *flashcard* nos níveis “Difícil”, “Médio” e “Fácil”. Esse caso de uso estende o caso de uso

UC-20: Interagir com Flashcard.

Pré-condição: O Estudante deve estar autenticado, o *flashcard* deve estar aberto e o verso do cartão deve ter sido revelado.

Fluxo principal:

1. O sistema exibe as opções de dificuldade do *flashcard*.
2. O Estudante seleciona uma das opções: “Difícil”, “Médio” ou “Fácil”.
3. O sistema registra a dificuldade escolhida no cartão.
4. O sistema atualiza a indicação de dificuldade do *flashcard* na interface.
5. O sistema exibe a classificação como parte do acompanhamento de revisão do Estudante.
6. Fim do caso de uso.

Fluxos de Exceção:

2.a: Nenhuma dificuldade selecionada.

1. O sistema mantém o cartão sem classificação de dificuldade.
2. O Estudante pode selecionar uma dificuldade posteriormente.

3.a: Classificação não registrada.

1. O sistema informa que a classificação não pôde ser registrada.
2. O Estudante pode tentar selecionar a dificuldade novamente.

Pós-condição: A dificuldade percebida do *flashcard* é registrada e exibida na interface do *deck*.

Apêndice B

Diagramas de Sequência

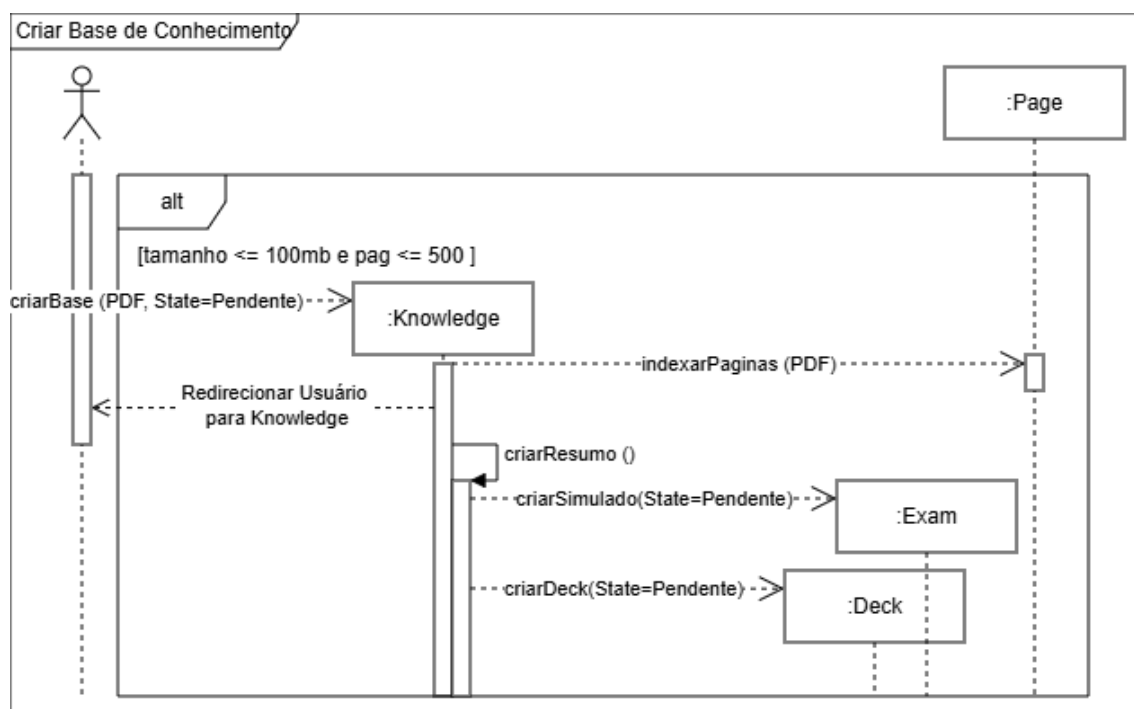


Figura B.1: Diagrama de sequência referente à criação das *knowledges*.

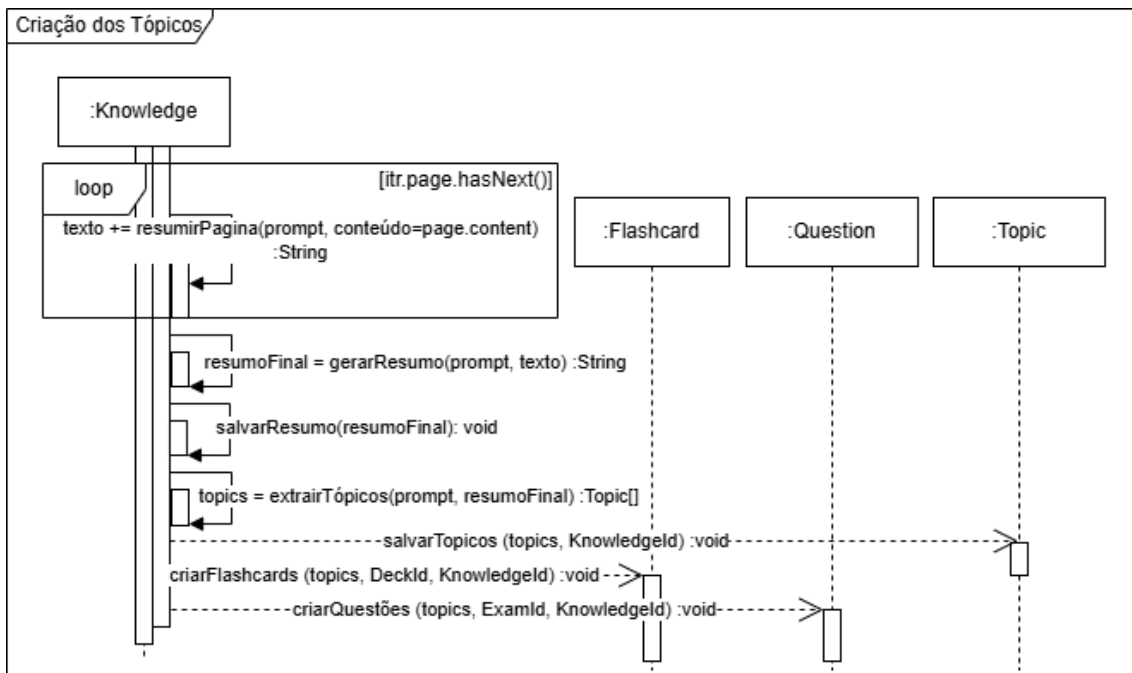


Figura B.2: Diagrama de sequência referente à criação dos *topics*.

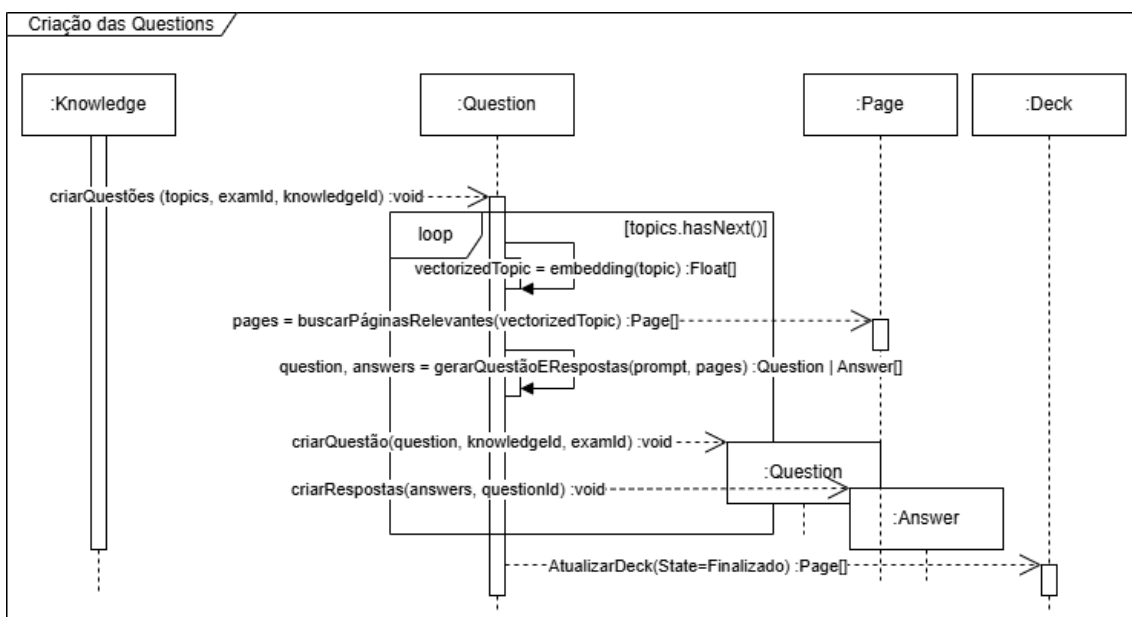


Figura B.3: Diagrama de sequência referente à criação das *questions* e *answers*.