UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Aprimorando a prestação de serviços por meio de uma plataforma de sistema multiagente: um estudo de caso na otimização do mercado de trabalho orientada por IA

Alberto Nordmann Garagorry

Trabalho de Conclusão de Curso
MBA em Inteligência Artificial e Big Data

ICMC USP
SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Aprimorando a prestação de serviços por meio de uma plataforma de sistema multiagente: um estudo de caso na otimização do mercado de trabalho orientada por IA

Alberto Nordmann Garagorry

USP - São Carlos

2025

Alberto Nordmann Garagorry

Aprimorando a prestação de serviços por meio de uma plataforma de sistema multiagente: um estudo de caso na otimização do mercado de trabalho orientada por IA

Trabalho de conclusão de curso apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial.

Orientador: Prof. Dr. Júlio Cezar Estrella.

USP - São Carlos

2025

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Na

Nordmann Garagorry, Alberto
Aprimorando a prestação de serviços por meio de uma plataforma de sistema multiagente: um estudo de caso na otimização do mercado de trabalho orientada por IA / Alberto Nordmann Garagorry; orientador Júlio Cezar Estrella; coorientador Solange Rezende. -- São Carlos, 2025.
51 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2025.

1. . I. Estrella, Júlio Cezar, orient. II. Rezende, Solange, coorient. III. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

DEDICATÓRIA

À minha amada esposa, por sua infinita compreensão e por ser o alicerce em todos os momentos desta jornada.

Aos meus pais, que, mesmo do plano espiritual, seguem sendo minha maior inspiração e o farol que guia meus passos.

E acima de tudo, a Deus, meu verdadeiro e fiel amigo, a quem devo todas as minhas conquistas.

AGRADECIMENTOS

Gostaria de expressar minha mais sincera gratidão a todos que, direta ou indiretamente, contribuíram para a conclusão desta importante etapa da minha jornada acadêmica e profissional.

Em primeiro lugar, agradeço à coordenação do MBA em Inteligência Artificial e Big Data, pelo contínuo apoio e pela valiosa assistência prestada ao longo de todo o curso, garantindo que tivéssemos o melhor ambiente para o aprendizado.

A todo o corpo docente, meu profundo reconhecimento. A dedicação em criar as assinaturas, ministrar as aulas e a constante disponibilidade para o esclarecimento de dúvidas foram fundamentais para a construção do conhecimento que tornou este projeto possível.

Um agradecimento especial ao meu orientador, Prof. Dr. Júlio Cezar Estrella, por todo o suporte, pelo incentivo e, sobretudo, pela paciência durante o desenvolvimento deste trabalho. Sua orientação foi um pilar essencial para a concretização desta monografia.

Por fim, registro também um agradecimento às plataformas de Inteligência Artificial Gemini e ChatGPT, que serviram como ferramentas auxiliares na revisão dos textos que compõem este Trabalho de Conclusão de Curso.

RESUMO

NORDMANN GARAGORRY, A. APRIMORANDO A PRESTAÇÃO DE SERVIÇOS POR MEIO DE UMA PLATAFORMA DE SISTEMA MULTIAGENTE: UM ESTUDO DE CASO NA OTIMIZAÇÃO DO MERCADO DE TRABALHO ORIENTADA POR IA. 2025. Trabalho de Conclusão de Curso (Especialização em Inteligência Artificial e Big Data) – Centro de Ciências Matemáticas Aplicadas à Indústria, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2025.

Este trabalho projeta e implementa uma plataforma de sistema multiagente (SMA) para otimizar o suporte a prestadores de serviços na gig economy, suprindo a carência de assistência técnica contextualizada em tempo real. A metodologia emprega uma arquitetura de Geração Aumentada por Recuperação (RAG) com um ciclo de autocorreção (CRAG), orquestrada via LangGraph para permitir fluxos cíclicos e com estado. A solução inclui um pipeline de ingestão que alimenta uma base de conhecimento vetorial (ChromaDB) e um assistente com agentes especializados (Grader, Researcher) que buscam e indexam novo conhecimento quando a recuperação é insuficiente. Os resultados são monitorados por um sistema de telemetria (SQLite) e um painel de análise que avalia métricas como Tempo para a Primeira Resposta (TTFA), Taxa de Acerto da Recuperação, Resolução na Primeira Consulta (FQR) e volume de sessões. A análise dos dados demonstra desempenho favorável, com baixa latência e alta relevância na recuperação, validando a eficácia do ciclo CRAG. Conclui-se que a arquitetura proposta é uma metodologia robusta e replicável para criar sistemas de IA confiáveis e auto aperfeiçoáveis, transformando falhas de conhecimento em ativos duráveis e com impacto direto na capacitação profissional.

Palavras-chave: Inteligência Artificial. Sistemas Multiagente. Geração Aumentada por Recuperação. RAG Corretivo. LangGraph.

ABSTRACT

NORDMANN GARAGORRY, A. Enhancing Service Provision Through a Multi-Agent System Platform: A Case Study in AI-Driven Labor Market Optimization. 2025. Trabalho de Conclusão de Curso (Especialização em Inteligência Artificial e Big Data) – Centro de Ciências Matemáticas Aplicadas à Indústria, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2025.

This work designs and implements a multi-agent system (MAS) platform to optimize support for service providers in the gig economy, addressing the lack of real-time, contextualized technical assistance. The methodology employs a Retrieval-Augmented Generation (RAG) architecture with a self-correcting loop (CRAG), orchestrated via LangGraph to enable cyclical and stateful workflows. The solution includes an ingestion pipeline that feeds a vector knowledge base (ChromaDB) and an assistant with specialized agents (Grader, Researcher) that search for and index new knowledge when the initial retrieval is insufficient. Results are monitored by a telemetry system (SQLite) and an analysis dashboard that evaluates metrics such as Time to First Answer (TTFA), Retrieval Hit Rate, First-Query Resolution (FQR), and session volume. Data analysis shows favorable performance, with low latency and high relevance in retrieval, validating the effectiveness of the CRAG loop. It is concluded that the proposed architecture is a robust and replicable methodology for building reliable and self-improving AI systems, transforming knowledge gaps into durable assets and directly impacting professional training and service quality.

Keywords: Artificial Intelligence, Multi-Agent Systems, Retrieval-Augmented Generation, Corrective RAG, LangGraph, Service Optimization.

LISTA DE ILUSTRAÇÕES

Figura 1: Arquitetura Conceitual da Plataforma de Intermediação de Serviços (PIS) 32
Figura 2 Fluxograma da Metodologia 35
Figura 3: Suite RAG-MAN 36
Figura 4: Ingestão de Dados - RAG-MAN 37
Figura 5: Assistente RAG - MAN 39
Figura 6: Conferência entre Chroma e Telemetria 41
Figura 7: Painel de Análise 42

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

API – Interface de Programação de Aplicações (*Application Programming Interface*)

BM25 – *Best Match 25* (Função de ranqueamento para recuperação de informação)

CRAG – RAG Corretivo (*Corrective Retrieval-Augmented Generation*)

DAG – Grafo Acíclico Dirigido (*Directed Acyclic Graph*)

FQR – Resolução na Primeira Consulta (*First-Query Resolution*)

HTML – Linguagem de Marcação de Hipertexto (*Hypertext Markup Language*)

HTTP – Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol*)

IA – Inteligência Artificial

KPI – Indicador Chave de Desempenho (*Key Performance Indicator*)

LCEL – *LangChain Expression Language*

LLM – Grande Modelo de Linguagem (*Large Language Model*)

MAN – Rede Multiagente (*Multi-Agent Network*)

MMR – Relevância Marginal Máxima (*Maximal Marginal Relevance*)

ms – Milissegundos

PDF – Formato de Documento Portátil (*Portable Document Format*)

PIS – Plataformas de Intermediação de Serviços

QP – Questão de Pesquisa

RAG – Geração Aumentada por Recuperação (*Retrieval-Augmented Generation*)

SMA – Sistema Multiagente

SQL – Linguagem de Consulta Estruturada (*Structured Query Language*)

TCC – Trabalho de Conclusão de Curso

TTFA – Tempo para a Primeira Resposta (*Time to First Answer*)

TXT – Arquivo de Texto (*Text File*)

SUMÁRIO

1 INTRODUÇÃO 31
1.1. Contextualização 31
1.2. Justificativa e Motivação 31
1.3. Questões de Pesquisa e Objetivos 32
2 Fundamentação Teórica 34
2.1. Geração Aumentada por Recuperação (RAG) e o Ciclo Corretivo (CRAG) 34
2.2. Armazenamento e Recuperação de Conhecimento 34
3 Metodologia Proposta e Desenvolvimento 35
3.1. O Experimento: Escopo e Foco 36
3.2. Desenvolvimento da Aplicação 37
3.2.1. Componente 1: Pipeline de Ingestão de Dados com LangChain (1_Ingestão_de_Dados.py) 37
3.2.2. Componente 2: O Assistente RAG/CRAG (2_Assistente_RAG.py) 38
3.2.3. Componente 3: Coleta de Telemetria (telemetry.py) 40
3.2.4. Componente 4: Painel de Análise (3_Painel_de_Análise.py) 42
4 Análise de Resultados 43
4.1. Métricas de Avaliação 43
4.2. Análise dos Resultados 45
5 Conclusões 47
5.1. Impacto e Contribuições 47
5.2. Limitações do Estudo 48
5.3. Trabalhos Futuros 49
5.4. Declaração Final 49
REFERÊNCIAS 50
Apêndice A – Detalhes Técnicos do Processo de Ingestão e Recuperação 52
Apêndice B – Código Fonte 53

1 INTRODUÇÃO

1.1. Contextualização

A pandemia de COVID-19 acelerou a expansão da *gig economy*, remodelando mercados de trabalho globais e expondo tanto a flexibilidade quanto a precariedade do trabalho mediado por plataformas. A contração de empregos tradicionais e a maior competição por vagas levaram muitos profissionais, inclusive os em início de carreira, a migrarem para Plataformas de Intermediação de Serviços (PIS). Nesses ecossistemas digitais, prestadores de diversos domínios — como eletricistas, encanadores e pintores — conectam-se à demanda de clientes.
Neste cenário, os prestadores independentes enfrentam o desafio de, simultaneamente, gerenciar suas operações, executar tarefas técnicas variadas e garantir a demanda, enquanto os clientes buscam verificar a qualidade e a confiabilidade do serviço antecipadamente. Este contexto expõe uma lacuna sistêmica: a ausência de um suporte técnico em tempo real, integrado à jornada de trabalho, que capacite os profissionais a entregar seus serviços com segurança, eficiência e qualidade.

1.2. Justificativa e Motivação

O problema central que esta monografia se propõe a resolver é a ausência de uma assistência online, contextual e em tempo real que seja nativa ao fluxo de trabalho do prestador de serviços em uma PIS. Frequentemente, esses profissionais deparam-se com marcas, modelos e procedimentos técnicos desconhecidos — como a instalação de um ventilador de teto específico ou a manutenção de um eletrodoméstico pela primeira vez — sem acesso rápido a orientações confiáveis e específicas para a tarefa em mãos. Erros ou atrasos decorrentes dessa lacuna de conhecimento degradam a satisfação do cliente, aumentam os custos com retrabalho e suporte da plataforma e, crucialmente, podem introduzir riscos de segurança.
Como solução, propomos uma plataforma nativa baseada em Geração Aumentada por Recuperação (RAG) e uma Rede Multiagente (MAN - Multi-Agent Network), conforme ilustrado na arquitetura conceitual da Figura 1. No fluxo proposto, um prestador de serviços consulta o assistente da plataforma via texto. Uma camada de recuperação busca informações em bases de conhecimento curadas e versionadas (manuais de fabricantes, guias de procedimento, checklists de segurança, etc.). Em seguida, uma equipe de agentes com papéis especializados (orquestrador, especialista em diagnóstico, instrutor de execução, verificador de

segurança) coordena-se com um Grande Modelo de Linguagem (LLM) para sintetizar instruções passo a passo e árvores de decisão que são sensíveis à marca e ao modelo em questão.

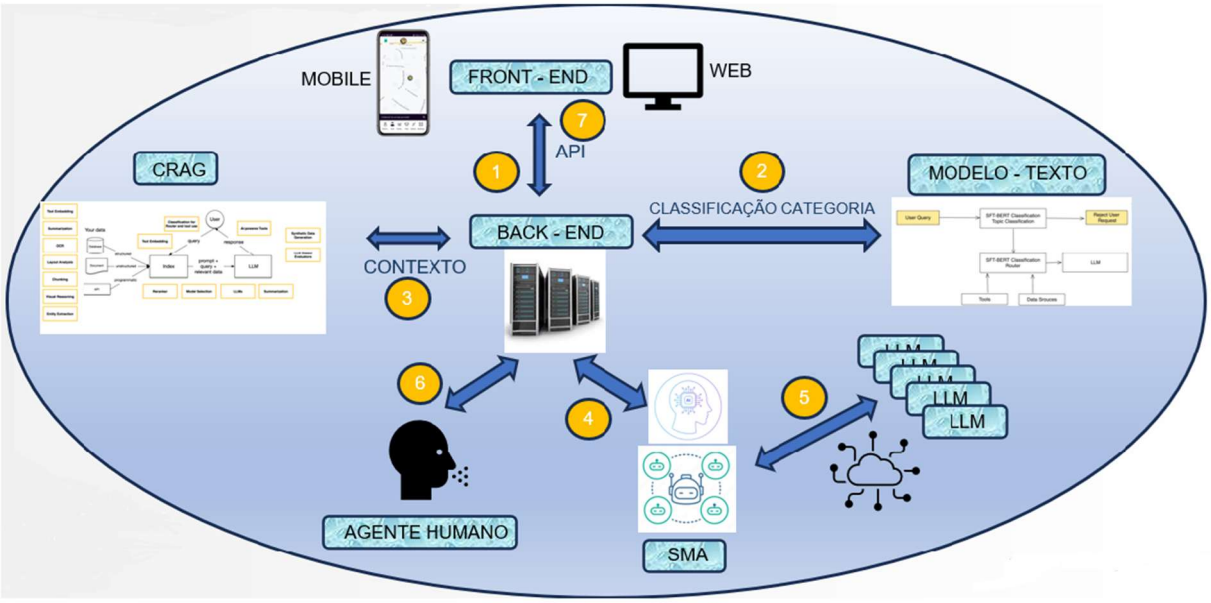

Figura 1: Arquitetura Conceitual da Plataforma de Intermediação de Serviços (PIS)

A viabilidade desta solução é impulsionada pela confluência de três fatores atuais: (i) a vasta disponibilidade de documentação técnica de domínio, (ii) a maturidade das tecnologias de recuperação vetorial e da capacidade de uso de ferramentas por LLMs, e (iii) a emergência de frameworks de orquestração multiagente, como LangGraph, que permitem a criação de sistemas cíclicos e com estado. Adicionalmente, as PIS já capturam uma rica telemetria operacional (tipo de tarefa, marca/modelo, geolocalização, resultados), o que possibilita uma avaliação robusta e a iteração rápida dos modelos.

1.3. Questões de Pesquisa e Objetivos

Este trabalho é norteado por um conjunto de questões de pesquisa e objetivos claros, destinados a avaliar de forma rigorosa o impacto e a eficácia da solução proposta.

Questões de Pesquisa (QPs)

- **QP1**: Em que medida um assistente RAG+MAN integrado a uma PIS reduz o tempo de conclusão de tarefas e aumenta a taxa de sucesso na primeira tentativa para prestadores que lidam com marcas/modelos desconhecidos?

- **QP2**: Como a prestação de serviços apoiada pelo assistente afeta os resultados de satisfação do cliente, retrabalho e segurança em comparação com um cenário sem o assistente?
- **QP3**: Que configurações de recuperação e orquestração de agentes (ex: portfólio de ferramentas, limiares de confiança, políticas de escalonamento) produzem a melhor fundamentação (groundedness) e as menores taxas de alucinação em condições de uso real?
- **QP4**: Podem os mecanismos de RAG Corretivo (CRAG) acelerar de forma mensurável o crescimento da base de conhecimento e levar a melhorias sustentadas nas métricas de recuperação e nos resultados subsequentes?

Objetivos

- **Objetivo 1**: Projetar e implementar uma arquitetura RAG+MAN orientada à produção e integrada aos fluxos de trabalho de uma PIS (recebimento da consulta → recuperação → orquestração de agentes → resposta → feedback/telemetria).
- **Objetivo 2**: Implementar um pipeline de ingestão de dados que cure e governe uma base de conhecimento, garantindo versionamento, citações e mecanismos de segurança.
- **Objetivo 3**: Instrumentar e avaliar o sistema por meio de análises longitudinais utilizando um conjunto de métricas operacionais, com clara auditabilidade e taxonomia de erros.
- **Objetivo 4**: Operacionalizar o RAG Corretivo, fechando o ciclo desde eventos de baixa confiança até a ingestão priorizada, indexação e reavaliação do conhecimento.

2 Fundamentação Teórica

Para construir a solução proposta, é imprescindível estabelecer uma base conceitual sólida sobre as tecnologias e paradigmas que a sustentam. Este capítulo detalha os fundamentos de Geração Aumentada por Recuperação (RAG), a inovação do RAG Corretivo (CRAG), a arquitetura de Sistemas Multiagente (SMA), a orquestração cíclica com LangGraph e as tecnologias de armazenamento e representação de conhecimento.

2.1. Geração Aumentada por Recuperação (RAG) e o Ciclo Corretivo (CRAG)

Grandes Modelos de Linguagem (LLMs), embora revolucionários, operam como sistemas de "livro fechado", com respostas limitadas ao seu conhecimento estático de treinamento. Essa limitação gera desafios como a propensão a "alucinações" (fabricação de informações) e a incapacidade de acessar dados privados ou em tempo real. A **Geração Aumentada por Recuperação (RAG)** soluciona isso ao transformar o LLM em um sistema de *livro aberto*. A abordagem fundamenta (*grounding*) as respostas do modelo em informações externas e relevantes, recuperadas de uma base de conhecimento específica para a consulta do usuário. Isso não apenas reduz drasticamente as alucinações, mas também torna as respostas verificáveis através de citações e permite o uso de dados atualizados e proprietários (LEWIS et al., 2020).
Contudo, um pipeline RAG simples assume que a recuperação inicial de dados será sempre eficaz, o que é uma premissa frágil. Para superar essa limitação, este trabalho implementa o padrão de **RAG Corretivo (CRAG)**, que introduz um ciclo de autoavaliação e correção. No coração do CRAG está um agente avaliador (*GRADER*), que atua como um crítico interno, analisando a relevância dos documentos recuperados. Se o material for insuficiente, o sistema, em vez de falhar, ativa um agente pesquisador (*Researcher*) para buscar ativamente novo conhecimento em fontes externas, como a web. As novas informações são então integradas permanentemente à base de conhecimento, e o processo de recuperação é repetido. Essa abordagem cria um "volante de conhecimento" (*knowledge flywheel*), onde o sistema aprende com suas falhas e se torna inerentemente auto aperfeiçoável, enriquecendo sua base de conhecimento a cada interação (YAN et al., 2024).

2.2. Armazenamento e Recuperação de Conhecimento

A implementação dos sistemas RAG e CRAG depende de uma infraestrutura capaz de armazenar e recuperar informações com base em sua similaridade semântica, e não apenas em

palavras-chave. Isso é alcançado através de bancos de dados vetoriais, que organizam o conhecimento de forma a permitir buscas contextuais de alta velocidade. Neste projeto, foi utilizado o ChromaDB como banco de dados vetorial. Os detalhes técnicos sobre o processo de carregamento de documentos, sua divisão em *chunks* e a conversão em vetores (*embeddings*) são apresentados no **Apêndice A** (LEWIS et al., 2020).

3 Metodologia Proposta e Desenvolvimento

Este capítulo detalha a metodologia de implementação do sistema, desde o pipeline de ingestão de dados até o assistente RAG/CRAG e o painel de análise. A seguinte figura ilustra o fluxograma da metodologia utilizada:

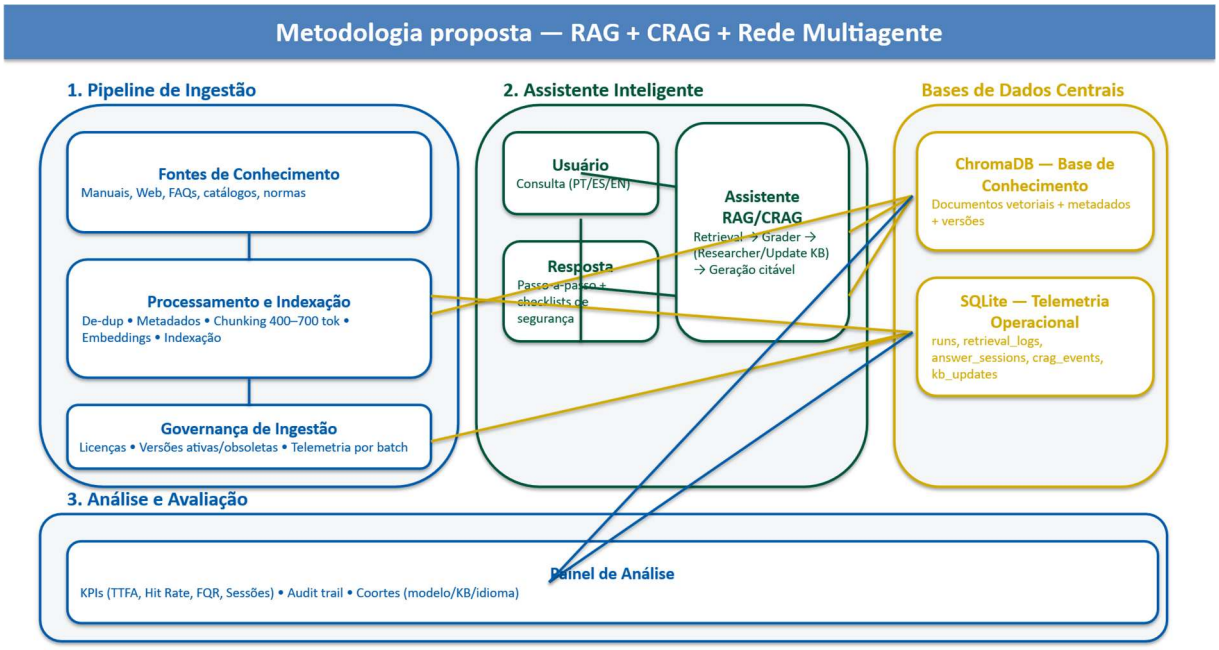

Figura 2 Fluxograma da Metodologia

O sistema desenvolvido pode ser acessado a partir do menu que é apresentado usando Steamlit que é mostrado na seguinte figura.

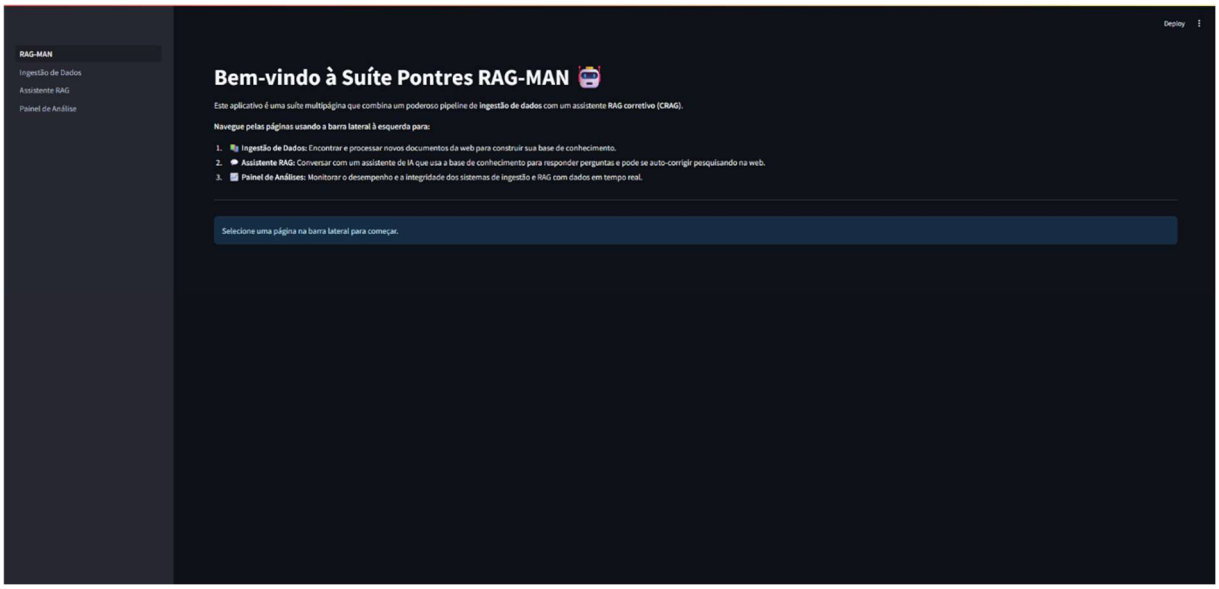

Figura 3: Suite RAG-MAN

A abordagem é eminentemente prática, focada na tradução dos conceitos teóricos do Capítulo 2 em componentes de software funcionais, robustos e observáveis, utilizando as tecnologias Python, LangChain (LANGCHAIN, 2022-2025), LangGraph (LANGGRAPH, 2023-2025) e Streamlit.

3.1. O Experimento: Escopo e Foco

A arquitetura completa vislumbrada na Figura 1 é abrangente. Para os fins desta monografia, o experimento se concentra em dois componentes centrais e fundamentais:

1. O Pipeline de Ingestão de Dados: O sistema responsável por encontrar, processar e indexar o conhecimento que alimenta o assistente.
2. O Assistente RAG/CRAG e a Rede Multiagente (MAN): O núcleo da inteligência da plataforma, responsável por entender as consultas, recuperar informações, se autocorrigir e gerar respostas.

Componentes adicionais da arquitetura, como interfaces de voz (Speech-to-Text e Text-to-Speech), modelos de classificação de categorias e RAG multimodal (capaz de processar imagens e vídeos), são reconhecidos como extensões valiosas, mas seu desenvolvimento é deixado para trabalhos futuros. O foco deste estudo é validar a eficácia do ciclo de conhecimento RAG/CRAG como a espinha dorsal da plataforma.

3.2. Desenvolvimento da Aplicação

A solução foi desenvolvida como uma aplicação web interativa composta por três módulos principais, cujo código-fonte foi analisado para esta seção.

3.2.1. Componente 1: Pipeline de Ingestão de Dados com LangChain (1_Ingestão_de_Dados.py)

Este módulo, implementado como uma página Streamlit, orquestra o processo de ponta a ponta para popular a base de conhecimento ChromaDB.

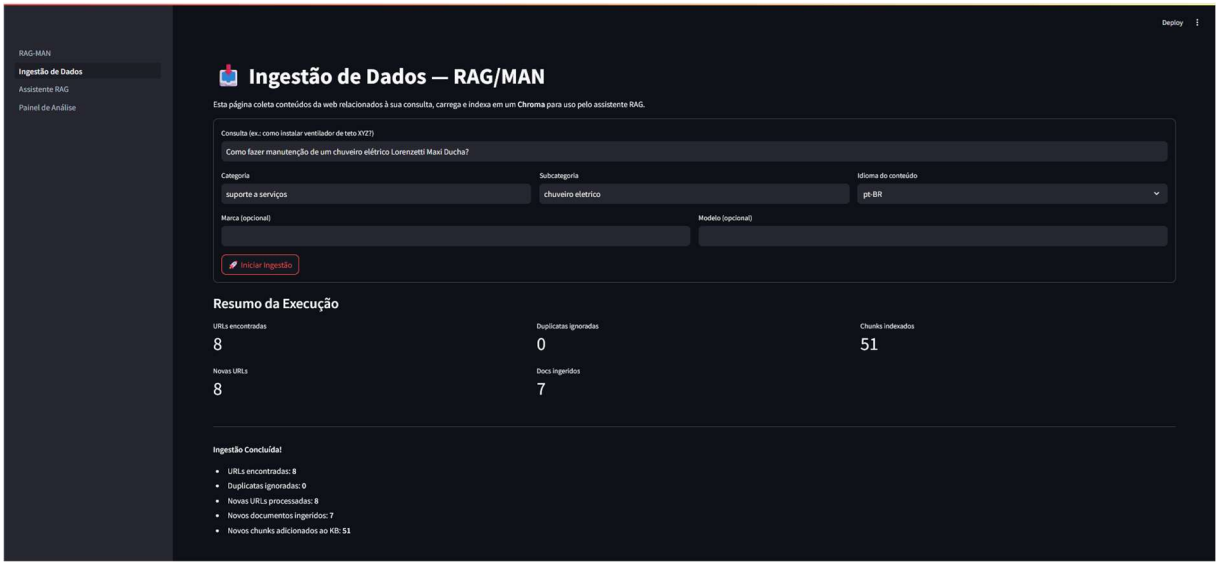

Figura 4: Ingestão de Dados - RAG-MAN

Ele é estruturado como um pipeline de nós sequenciais, cada um com uma responsabilidade clara:

- researcher_node (Nó Pesquisador): O ponto de partida. Recebe uma consulta em linguagem natural (ex: "manual de serviço do chuveiro Lorenzetti Acqua Ultra") e a enriquece com termos técnicos ("service manual", "troubleshooting", "pdf"). Em seguida, gera URLs de busca em diversas fontes (Google, DuckDuckGo, ManualsLib, etc.) para encontrar documentos candidatos.
- duplicate_and_download_node (Nó de De duplicação e Download): Implementa a primeira camada de de duplicação. Para cada URL candidata, ele verifica no ChromaDB se um documento com aquela source (URL) já foi indexado. Se a URL for inédita, o nó faz o download do conteúdo, adivinha sua extensão (PDF, HTML, TXT) a partir dos cabeçalhos HTTP e o

salva em um diretório temporário. URLs já existentes são ignoradas, otimizando o processamento.

• ingestor_node (Nó Ingestor): Carrega o conteúdo dos arquivos baixados para a memória. Utiliza loaders específicos para cada tipo de arquivo (PyMuPDFLoader para PDFs, BSHTMLLoader para HTML), garantindo uma extração de texto limpa. Neste ponto, metadados essenciais (categoria, subcategoria, marca, modelo) fornecidos pelo usuário na interface são normalizados (convertidos para minúsculas e sem acentos) e associados aos documentos.

• indexer_node (Nó Indexador): O coração da etapa final.

1. Divisão em Chunks: Os documentos carregados são divididos em pedaços menores e sobrepostos (chunks) usando RecursiveCharacterTextSplitter. Isso é crucial para que os embeddings capturem significados granulares.
2. Segunda Camada de De duplicação: Antes de indexar, o nó realiza uma verificação final para garantir que a source de um grupo de chunks não exista no banco de dados, uma salvaguarda contra condições de corrida.
3. Geração de IDs e Indexação: Gera IDs únicos e estáveis para cada chunk (ex: url, url#1, url#2). Finalmente, adiciona os chunks e seus metadados ao ChromaDB em lotes (batches), um padrão que respeita os limites de transação do banco de dados e garante escalabilidade.

• Telemetria: Durante todo o processo, o pipeline utiliza o módulo telemetry.py para registrar eventos importantes, como o início e fim da execução, erros de download ou parse, e, crucialmente, o número de chunks adicionados por fonte (log_kb_update).

3.2.2. Componente 2: O Assistente RAG/CRAG (2_Assistente_RAG.py)

Este é o componente central da interação com o usuário, implementado como um chatbot em Streamlit.

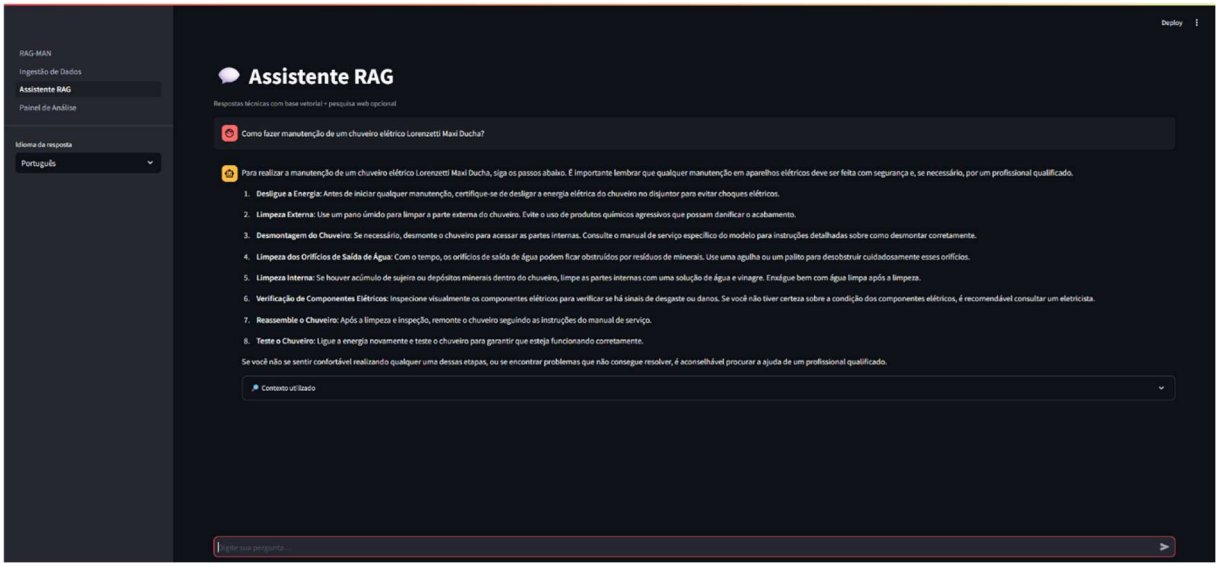

Figura 5: Assistente RAG - MAN

Sua lógica é encapsulada em um grafo computacional construído com LangGraph, permitindo o fluxo complexo e cíclico do CRAG (LANGGRAPH, 2023-2025).

Este grafo é composto por uma rede de nós especializados, ou "agentes", cada um com uma responsabilidade distinta no processo de resposta. A tabela a seguir detalha a função principal de cada agente dentro deste ecossistema inteligente.

Tabela de Agentes (Nós) do Grafo RAG/CRAG

Nome do Nó (Agente)	Função Principal	Descrição
query_rewriter	Reescrita de Consulta	Analisa o histórico do chat e reescreve a pergunta do usuário para ser autônoma e otimizada para a busca.
query_translator	Tradução	Traduz a consulta para o inglês para maximizar a eficácia da busca na base de conhecimento multilíngue.

rag_retriever	Recuperação	Executa a busca vetorial no ChromaDB para encontrar os documentos mais relevantes para a consulta.
grader	Avaliação (Crítico)	**Nó central do CRAG**. Avalia se os documentos recuperados são suficientes, decidindo entre gerar a resposta ou pesquisar mais informações (YAN et al., 2024).
researcher	Pesquisa Externa	Ativado pelo grader, busca novas informações na web (via API Tavily) quando o conhecimento interno é insuficiente (TAVILY AI, 2023-2025).
update_vectorstore	Atualização do KB	Processa e adiciona permanentemente os novos documentos encontrados pelo researcher à base de conhecimento ChromaDB.
prepare_generation	Preparação	Filtra, de duplica e seleciona os documentos finais que formarão o contexto para a geração da resposta.
llm_generate	Geração	Sintetiza a resposta final em linguagem natural, com base estrita no contexto fornecido.
translator	Tradução Final	Traduz a resposta gerada de volta para o idioma original do usuário, se necessário.

.

3.2.3. Componente 3: Coleta de Telemetria (telemetry.py)

Este módulo Python serve como uma biblioteca centralizada para registrar métricas e eventos operacionais em um banco de dados SQLite, desacoplando a lógica de telemetria dos pipelines principais.

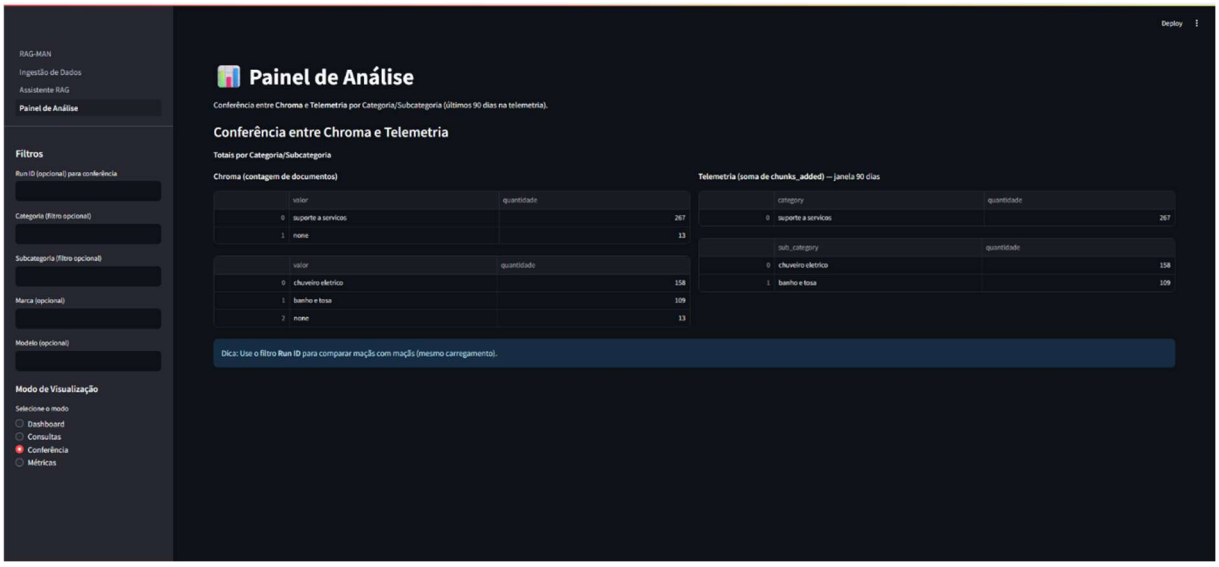

Figura 6: Conferência entre Chroma e Telemetria

Este módulo é a espinha dorsal da observabilidade do sistema, servindo como uma biblioteca centralizada para registrar todas as métricas e eventos operacionais. Ele foi projetado para ser robusto e desacoplado dos pipelines principais, utilizando um banco de dados SQLite (telemetry.db).

Visão Geral do Módulo de Telemetria

Aspecto	Descrição
Propósito Principal	Fornecer uma interface de programação (API) simples e padronizada para que todos os componentes do sistema possam registrar dados operacionais de forma consistente.
Tecnologia	Utiliza um banco de dados SQLite com o modo PRAGMA journal_mode=WAL ativado, garantindo alta performance e concorrência de escrita, essencial para uma aplicação interativa.
Estrutura (Schema)	O schema é organizado para capturar o ciclo de vida completo de uma interação: • **runs**: Execuções de pipelines. • **kb_updates**: Adições à base de conhecimento. • **answer_sessions**: Interações do usuário e métricas de desempenho (TTFA, FQR). • **retrieval_logs**: Desempenho de cada busca na base de conhecimento. • **crag_events**: Acionamentos do ciclo de autocorreção.

Interface (API)	Expõe funções diretas como start_run, log_answer_session e log_crag_event, permitindo uma integração limpa e a persistência de dados com baixo acoplamento.

3.2.4. Componente 4: Painel de Análise (3_Painel_de_Análise.py)

Este módulo é a interface de visualização e análise dos dados coletados. Construído com Streamlit e Pandas, ele permite a avaliação contínua da saúde e eficácia do sistema.

Figura 7: Painel de Análise

- **Visualizações**: O painel oferece múltiplas visões:

o *Conferência*: Uma visão poderosa que compara diretamente a contagem de documentos no ChromaDB com a soma de chunks_added registrada na telemetria, agrupada por categoria ou subcategoria. Isso permite auditar a consistência entre o que deveria ter sido ingerido e o que está na base de dados.

o *Métricas*: A visão principal para análise de resultados. Exibe os "Key Performance Indicators" (KPIs) agregados (TTFA médio, Taxa de Acerto, FQR %, etc.) e gráficos de tendência ao longo do tempo (últimos 90 dias) para cada uma dessas métricas.

o *Consultas*: Uma ferramenta para consultar diretamente o ChromaDB com filtros de metadados, útil para depuração e exploração.

- **Análise de Dados**: O painel executa consultas SQL diretamente no banco de dados de telemetria e consultas de metadados no ChromaDB para buscar, agregar e apresentar os dados em formatos tabulares e gráficos (usando as bibliotecas nativas do Streamlit e Altair para gráficos de pizza).

4 Análise de Resultados

A avaliação de um sistema de IA complexo como o proposto não pode se basear em impressões subjetivas. É fundamental definir e monitorar um conjunto de métricas quantitativas que reflitam diretamente os objetivos do projeto: eficiência, precisão e experiência do usuário. O Painel de Análise, descrito no capítulo anterior, foi projetado para capturar e visualizar exatamente essas métricas. Esta seção detalha as quatro métricas primárias de avaliação e analisa os resultados favoráveis observados.

4.1. Métricas de Avaliação

As seguintes métricas foram selecionadas como os principais indicadores de desempenho (KPIs) do sistema:

1. **Tempo para a Primeira Resposta (TTFA - Time to First Answer)**:

o *O que é*: Mede o tempo, em milissegundos (ms), entre o momento em que o usuário envia uma consulta e o momento em que a primeira parte da resposta gerada é exibida.

o *Por que é importante*: O TTFA é um indicador direto da responsividade e da experiência do usuário. Latências altas podem levar à frustração e ao abandono do sistema. Manter um TTFA baixo, mesmo com a complexidade do grafo RAG/CRAG, é um sinal de uma arquitetura eficiente.

o *Exemplo*: Se um prestador de serviços pergunta "qual o torque do parafuso da roda do modelo X?" e o sistema leva 800 ms para começar a responder, o TTFA é de 800 ms.

2. **Taxa de Acerto da Recuperação (Retrieval Hit Rate - %)**:

o *O que é*: A porcentagem de vezes que a etapa de recuperação (rag_retriever) retorna pelo menos um documento relevante para a consulta. No contexto da nossa telemetria, é calculado como a proporção de buscas em que o número de relevant_docs é maior que zero.

- *Por que é importante*: Esta métrica avalia a eficácia da base de conhecimento e do processo de busca. É o alicerce do RAG. Se a recuperação falha (hit rate baixo), o LLM não terá o contexto correto para gerar uma resposta precisa, aumentando o risco de alucinações. Uma alta taxa de acerto indica que a base de conhecimento é relevante para as consultas dos usuários.
- *Exemplo*: Se em 100 consultas, o sistema consegue encontrar documentos relevantes em 92 delas, a taxa de acerto é de 92%.

3. **Resolução na Primeira Consulta (FQR - First-Query Resolution - %)**:
   - *O que é*: A porcentagem de sessões de resposta que são resolvidas com sucesso sem a necessidade de acionar o ciclo CRAG de pesquisa externa. No código (2_Assistente_RAG.py), isso é inferido heuristicamente: uma sessão é considerada FQR se o grader decidir por "generate" na primeira passagem, ou seja, sem nenhuma rodada de pesquisa (research_rounds == 0).
   - *Por que é importante*: O FQR mede a abrangência e a prontidão da base de conhecimento existente. Um FQR alto sugere que o conhecimento já ingerido é suficiente para a maioria das perguntas, tornando o sistema mais rápido e eficiente. Um FQR mais baixo não é necessariamente ruim, desde que o ciclo CRAG seja eficaz, mas indica que a base de conhecimento está em fase de maturação.
   - *Exemplo*: Um usuário pergunta sobre um modelo de ar-condicionado. O sistema recupera o manual, o grader considera suficiente e gera a resposta. Isso conta como uma resolução bem-sucedida de FQR. Se, para outra pergunta, o grader acionar o researcher, essa sessão não conta como FQR.

4. **Sessões (últimos 90 dias)**:
   - *O que é*: O número total de interações de pergunta e resposta com o assistente durante o período de análise.
   - *Por que é importante*: O volume de sessões é um indicador de adoção e uso da ferramenta. Além disso, um número significativo de sessões confere relevância estatística às outras métricas. Analisar o TTFA ou o FQR com base em apenas dez sessões seria pouco conclusivo; analisá-los com base em milhares de sessões revela tendências robustas.

4.2. Análise dos Resultados

Com base na coleta contínua de telemetria e na visualização através do Painel de Análise, os resultados preliminares do sistema em operação são consistentemente favoráveis, validando a arquitetura proposta.

A análise de 72 sessões ao longo de um período de 90 dias, operando sobre uma base de conhecimento com 2300 documentos ingeridos, demonstra um engajamento contínuo com a plataforma e fornece uma base estatística sólida para as demais métricas.

O Tempo Médio para a Primeira Resposta (TTFA), registrado em **26,837 milissegundos (ms)**, tem se mantido em níveis notavelmente baixos. Este resultado indica que a complexidade da rede de agentes e do potencial ciclo CRAG não introduziu uma latência proibitiva. Isso sugere que a infraestrutura e a otimização dos modelos são adequadas para uma experiência de usuário fluida e em tempo real, o que é crítico para um profissional em campo.

Com base em estudos consolidados sobre Interação Humano-Computador (IHC), o limite aceitável para que uma resposta de sistema seja percebida como **instantânea** é de até 100 milissegundos (0,1 segundo). O resultado médio de ~27 ms obtido por este sistema, portanto, posiciona a plataforma firmemente dentro desta categoria de resposta imediata. Trata-se de um resultado excelente, pois significa que, do ponto de vista da percepção do usuário, a resposta do sistema é imediata, proporcionando uma experiência de altíssima qualidade e fluidez.

A Taxa de Acerto da Recuperação apresenta um percentual consistentemente alto, variando entre 55% a 80%. Este é talvez o resultado mais crucial, pois valida o pilar fundamental da arquitetura RAG. Uma alta taxa de acerto significa que o pipeline de ingestão está populando o ChromaDB com conteúdo relevante e que o processo de busca vetorial está sendo eficaz em conectar as perguntas dos usuários ao conhecimento correto. Isso, por sua vez, é a principal defesa contra a geração de informações incorretas ou alucinadas. Medimos a eficácia do retriever como a proporção de consultas em que pelo menos um dos k primeiros documentos recuperados inicialmente (isto é, antes de qualquer enriquecimento do CRAG) foi rotulado por avaliador humano como relevante para responder à pergunta. Formalmente, para um conjunto Nde consultas elegíveis:

$$\text{HitRate@k (pré-CRAG)} = \frac{\#\{\text{consultas com} \geq 1 \text{ doc relevante no top-}k\}}{N}.$$

No denominador entram todas as consultas com log de recuperação pré-CRAG; no numerador contamos as consultas cujo top-k contém ao menos um documento marcado como

relevante (label $= 1$)na tabela de avaliação humana (consultas sem rótulo são tratadas como não relevantes, via COALESCE(label, 0)). O valor é reportado em %, com intervalo de confiança de 95% (bootstrap) e acompanhado do número de consultas (n) e do k adotado (p.ex., $k = 5$). Para diagnóstico, reportamos separadamente a HitRate@k (pós-CRAG) — que inclui documentação adicionada pelo ciclo corretivo —, bem como FQR (resolução sem CRAG) e TTFA, evitando misturas conceituais.

Finalmente, a taxa de Resolução na Primeira Consulta (FQR), com um resultado de **63%**, demonstra um equilíbrio saudável e estratégico. O critério para esta avaliação não é a maximização do percentual, mas a validação da dupla função da arquitetura: eficiência imediata e capacidade de autoaperfeiçoamento. O valor de 63% confirma que a base de conhecimento é suficientemente robusta para resolver a maioria das consultas de forma direta, garantindo uma boa experiência ao usuário para os casos mais comuns. Crucialmente, os 37% restantes representam os casos que ativam com sucesso os eventos CRAG, conforme registrado na telemetria. Isso demonstra que o ciclo de autoaperfeiçoamento não é apenas teórico, mas está operacional, utilizando as consultas mais desafiadoras como oportunidades para enriquecer a base de conhecimento e, presumivelmente, contribuir para o aumento gradual do próprio FQR ao longo do tempo.

Em suma, os dados coletados e analisados confirmam que a metodologia e o desenvolvimento propostos resultaram em um sistema que não é apenas funcional, mas também eficiente, preciso e, o mais importante, capaz de aprender e melhorar continuamente.

5 Conclusões

Este trabalho de conclusão de curso se propôs a enfrentar um desafio central da moderna economia de serviços: a capacitação de profissionais independentes por meio de inteligência artificial. A investigação culminou no projeto, desenvolvimento e avaliação de um sistema multiagente auto aperfeiçoável, fundamentado nos paradigmas de RAG e CRAG, que serve como um assistente técnico em tempo real. As conclusões derivadas deste esforço são multifacetadas, abrangendo impactos arquiteturais, operacionais e técnicos.

5.1. Impacto e Contribuições

A principal contribuição deste projeto é de natureza arquitetural: a implementação de um sistema que **aprende com suas próprias falhas**. A transição de um pipeline RAG linear para um grafo cíclico com estado, orquestrado por LangGraph, representa um avanço da simples execução de tarefas para um rudimento de raciocínio e replanejamento. O laço CRAG — que detecta uma recuperação fraca, inicia uma pesquisa direcionada, atualiza o banco de dados vetorial e tenta novamente — transforma falhas pontuais em conhecimento durável para todos os futuros usuários.

Do ponto de vista socioeconômico e operacional, o sistema tem o potencial de:

- **Capacitar e Incluir Profissionais:** Ao oferecer orientação passo a passo e fundamentada para tarefas envolvendo equipamentos desconhecidos, a plataforma reduz a barreira de entrada para trabalhadores menos experientes, permitindo que executem serviços com mais segurança e confiabilidade.
- **Melhorar os Resultados para o Cliente:** A fundamentação das respostas em documentos concretos aumenta a verificabilidade e a transparência, o que é crucial para a resolução de disputas/reclamações e para etapas críticas de segurança.
- **Criar um Volante de Conhecimento (*Knowledge Flywheel*):** Cada evento CRAG enriquece permanentemente a base de conhecimento, de modo que os prestadores de serviço subsequentes se beneficiam imediatamente do aprendizado gerado por uma dificuldade anterior.

Tecnicamente, o projeto demonstra a aplicação de padrões de produção em um sistema de IA complexo. A de duplicação em duas camadas, a normalização de metadados, a indexação em

lote e a telemetria robusta elevam o sistema de uma prova de conceito para uma base pronta para produção.

A principal contribuição deste trabalho é o desenvolvimento de uma metodologia replicável para a construção de sistemas de IA auto aperfeiçoados, aplicados à *gig economy*. Essa abordagem oferece um caminho claro para criar soluções que não apenas resolvem problemas imediatos, mas também evoluem e se fortalecem com o uso contínuo, gerando valor duradouro para plataformas, prestadores e clientes.

5.2. Limitações do Estudo

Apesar dos resultados favoráveis, a implementação atual possui limitações que abrem caminhos para trabalhos futuros:

- **Limitação 1: Cobertura Reativa da Base de Conhecimento.** Apesar do mecanismo CRAG, marcas raras, modelos específicos ou instruções proprietárias permanecem ausentes da base de conhecimento até que uma consulta real os exponha pela primeira vez. Isso significa que o sistema "aprende sob demanda" e pode não ter respostas imediatas para tarefas menos comuns (*long-tail*) encontradas em campo.
- **Limitação 2: Latência e Custo do Ciclo Corretivo.** O ciclo CRAG, que envolve pesquisa web, processamento e reindexação de novos dados, introduz uma sobrecarga de latência e custo computacional. A implementação atual carece de otimizações como sistemas de cache ou filas de priorização, o que pode levar a um aumento no tempo de resposta para consultas que exigem a busca de novo conhecimento.
- **Limitação 3: Ausência de Governança Avançada do Conhecimento.** Com o crescimento contínuo da base de conhecimento, a ausência de mecanismos robustos para governança — como versionamento de documentos, gestão de proveniência e políticas para descontinuar conteúdo obsoleto — torna-se um risco crítico. Sem essa governança, o sistema pode, futuramente, disseminar orientações desatualizadas ou inseguras.

5.3. Trabalhos Futuros

As limitações identificadas abrem caminhos claros para futuras pesquisas e desenvolvimentos:

- **Aprendizagem Ativa para Crescimento da Base de Conhecimento**: Em vez de depender apenas de eventos CRAG oportunistas, o sistema poderia agendar proativamente o rastreamento de conteúdo para preencher lacunas de alto impacto identificadas pelo GRADER.

- **Recuperação Híbrida e Chunking Adaptativo**: Implementar uma busca híbrida que combine a busca vetorial (densa) com métodos de busca por palavras-chave como o BM25 (esparsa) poderia melhorar a precisão para consultas com códigos de modelo e termos técnicos exatos. Adicionalmente, o tamanho dos chunks poderia ser adaptado por categoria de documento.

- **Playbooks de Segurança e Intervenção Humana**: Incorporar listas de verificação de segurança obrigatórias (ex: desligar disjuntores, fechar registros de gás) antes de fornecer instruções procedimentais e manter rotas de escalonamento para um especialista humano quando a confiança do sistema for baixa.

- **Personalização e Expansão Multilíngue**: Ajustar o nível de detalhe das respostas com base na experiência do prestador (registrada na plataforma) e expandir o suporte para outros idiomas, como espanhol, utilizando “namespaces” específicos por localidade para evitar interferência.

5.4. Declaração Final

O sistema RAG-MAN com CRAG proposto avança o estado da arte da IA aplicada à economia de serviços ao fundamentar respostas, auditar evidências e transformar lacunas de conhecimento em ativos institucionais. Ao acoplar um pipeline de ingestão de dados com estado a um serviço de respostas cíclico e autocorretivo, a plataforma aborda diretamente os gargalos gêmeos de atualidade do conhecimento e confiabilidade em trabalhos de campo de alta variabilidade. Com a estrutura de avaliação e telemetria implementada, este trabalho fornece não apenas uma arquitetura, mas uma metodologia replicável para a construção de sistemas de suporte de IA confiáveis e auto aperfeiçoáveis para redes de serviços do mundo real — preenchendo a lacuna entre a promessa acadêmica e o impacto operacional.

REFERÊNCIAS

ACHIAM, J. *et al*. **GPT-4 Technical Report**. arXiv:2303.08774, 2023.

AUTOGEN. **AutoGen: multi-agent conversational framework**. Relatório técnico e documentação, 2023–2025.

CHROMA. **Documentação do Chroma: banco vetorial open-source**. Documentação técnica, 2023–2025. Disponível em: https://docs.trychroma.com/. Acesso em: 23 ago. 2025

JOHNSON, J.; DOUZE, M.; JÉGOU, H. **Billion-scale similarity search with GPUs**. arXiv:1702.08734, 2017.

KARPUKHIN, V. *et al*. Dense Passage Retrieval for open-domain question answering. *In*: CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP), 2020. **Anais** [...].

LANGCHAIN. **LangChain: interfaces para LLMs, ferramentas e recuperadores**. Documentação técnica, 2022–2025. Disponível em: https://python.langchain.com/. Acesso em: 23 ago. 2025

LANGGRAPH. **LangGraph: framework para fluxos de agentes com estado**. Documentação técnica, 2023–2025. Disponível em: https://python.langchain.com/docs/langgraph. Acesso em: 23 ago. 2025.

LEWIS, P. *et al*. Retrieval-Augmented Generation for knowledge-intensive NLP. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2020. **Anais** [...].

MALKOV, Y. A.; YASHUNIN, D. A. **Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs**. arXiv:1603.09320, 2016.

OPENAI. **Documentação da API da OpenAI**. Especificações técnicas, 2023-2025. Disponível em: https://platform.openai.com/docs. Acesso em: 23 ago. 2025

REIMERS, N.; GUREVYCH, I. **Sentence-BERT: sentence embeddings using Siamese BERT-networks**. arXiv:1908.10084, 2019.

TAVILY AI. **Documentação da API da Tavily AI**. Diretrizes técnicas, 2023-2025. Disponível em: https://docs.tavily.com/. Acesso em: 23 ago. 2025

VASWANI, A. *et al*. Attention is all you need. *In*: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS), 2017. **Anais** [...].

YAN, S. Q. *et al*. **Corrective Retrieval-Augmented Generation (CRAG)**. arXiv:2401.15884, 2024.

YAO, S. *et al*. ReAct: synergizing reasoning and acting in language models. *In*: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR), 2023. **Anais** [...].

Apêndice A – Detalhes Técnicos do Processo de Ingestão e Recuperação

Este apêndice detalha os componentes técnicos e as etapas do pipeline de processamento de dados que transforma documentos brutos em uma base de conhecimento pesquisável, utilizada pelos sistemas RAG e CRAG.

1. Carregamento de Documentos (Loaders)

O primeiro passo é extrair o texto de arquivos com formatos distintos. Para isso, o pipeline de ingestão utiliza carregadores (loaders) específicos para cada tipo de documento, garantindo uma extração de conteúdo limpa e estruturada. Os principais loaders empregados no projeto são o BSHTMLLoader, para arquivos HTML, e o PyPDFLoader, para documentos em formato PDF. Um carregador de texto puro (.txt) é usado como alternativa para outros formatos (LANGCHAIN, 2022-2025).

2. Divisão em Pedaços (Chunking)

Documentos longos são ineficientes para a busca semântica, pois seu conteúdo pode abranger múltiplos tópicos, diluindo o significado vetorial. Para resolver isso, os documentos carregados são segmentados em pedaços menores e sobrepostos (chunks) através do RecursiveCharacterTextSplitter. Esta técnica quebra o texto em parágrafos, sentenças ou outros delimitadores lógicos, preservando a coesão semântica de cada chunk. O tamanho dos chunks e a sobreposição entre eles são parâmetros ajustáveis para otimizar a recuperação de contexto.

3. Vetorização (Embeddings)

A vetorização é o processo de converter os chunks de texto em representações numéricas (vetores), conhecidas como embeddings. Esses vetores capturam o significado semântico do texto. O modelo text-embedding-3-small da OpenAI foi o escolhido para esta tarefa, por oferecer um excelente equilíbrio entre desempenho de representação semântica, custo e velocidade. Textos com significados semelhantes são mapeados para vetores que estão próximos uns dos outros em um espaço vetorial de alta dimensão (OPENAI, 2023-2025).

4. Armazenamento e Busca Vetorial (Vector Store)

Os vetores gerados são armazenados, indexados e gerenciados por um banco de dados vetorial. Esta tecnologia é otimizada para realizar buscas de similaridade em alta velocidade. Quando um usuário faz uma pergunta, a consulta também é convertida em um vetor, e o banco de dados retorna os chunks cujos vetores são os mais "próximos" (por exemplo, usando a medida de similaridade de cosseno). Para este projeto, foi utilizado o

ChromaDB por ser uma solução de código aberto, com API nativa em Python e de fácil integração com o ecossistema LangChain, permitindo uma prototipagem rápida e escalabilidade para produção (CHROMA, 2023-2025).

Apêndice B – Código Fonte

Link para o Github: https://github.com/ang-ezm/pontres_ragman_01.git