INSTITUTO FEDERAL
SANTA CATARINA

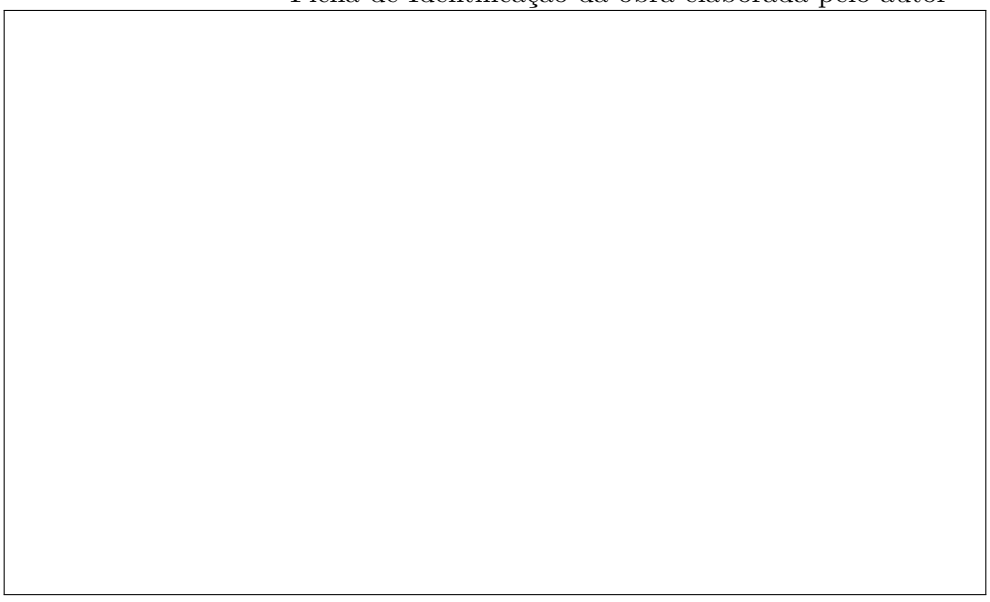
CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE GESTÃO DO CONHECIMENTO E TECNOLOGIAS COMPUTACIONAIS
CURSO SUPERIOR DE TECNOLOGIA EM GESTÃO DA TECNOLOGIA EM INFORMAÇÃO

MARCEL DA SILVA ALMEIDA

Proposta de uma Arquitetura de Software Multiagente para a Democratização da Informação Jurídica

Florianópolis–SC
2025

Ficha de Identificação da obra elaborada pelo autor



INSTITUTO FEDERAL DE SANTA CATARINA
DEPARTAMENTO ACADÊMICO DE GESTÃO DO CONHECIMENTO E TECNOLOGIAS COMPUTACIONAIS
CURSO SUPERIOR DE GESTÃO DA TECNOLOGIA EM INFORMAÇÃO

Marcel da Silva Almeida

Proposta de uma Arquitetura de Software Multiagente para a Democratização da Informação Jurídica

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Tecnólogo em Gestão da Tecnologia da Informação.

Prof. Dr. Egon Sewald Junior
(Orientador)

Florianópolis–SC
2025

PROPOSTA DE UMA ARQUITETURA DE SOFTWARE MULTIAGENTE PARA A DEMOCRATIZAÇÃO DA INFORMAÇÃO JURÍDICA

MARCEL DA SILVA ALMEIDA

Este trabalho foi julgado adequado para obtenção do Título de Tecnólogo em Gestão da Tecnologia da Informação e aprovado na sua forma final pela banca examinadora do Curso Superior de Tecnologia em Gestão da Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis-SC, ___ de _____ de 2025.

Prof. Dr. Egon Sewald Junior
Coordenador do CST em Gestão da Tecnologia da Informação
Instituto Federal de Santa Catarina

Banca Examinadora

Prof. Dr. Egon Sewald Junior
Orientador
Instituto Federal de Santa Catarina

Nome do Membro da Banca 1, Titulação
Nome de Membro da banca, Titulação
Instituto Federal de Santa Catarina

Nome do Membro da Banca 2, Titulação
Nome de Membro da banca, Titulação
Instituto Federal de Santa Catarina

A todos que fizeram parte da minha caminhada e, com seu apoio, tornaram este momento possível.

AGRADECIMENTOS

Ao meu orientador, pela paciência e por acreditar neste projeto mesmo quando os prazos pareciam distantes. A todos que fizeram parte desta caminhada, meu mais sincero agradecimento. Os últimos tempos foram particularmente complicados, e conciliar as demandas pessoais e profissionais com este trabalho foi um desafio imenso. A jornada foi marcada por altos e baixos, por períodos onde tudo parecia impossível. Neste contexto, a paciência e a sensibilidade que encontrei no corpo docente foram um pilar. Agradeço do fundo do coração por terem compreendido meus tempos, por me ajudarem a navegar por esses obstáculos e por todo o apoio incondicional. Sem a compreensão de vocês, eu não teria chegado até aqui. Este trabalho, portanto, representa não apenas uma conquista acadêmica, mas um profundo crescimento pessoal.

"Nós moldamos nossas ferramentas e, depois, nossas ferramentas nos moldam."
– *Marshall McLuhan*

RESUMO

O acesso à informação jurídica no Brasil constitui um desafio social complexo, marcado pela opacidade da linguagem legal e pela dificuldade de navegação nos sistemas formais de justiça. Este trabalho aborda este problema através da concepção, desenvolvimento e documentação de uma arquitetura de software inovadora. A solução proposta utiliza um sistema multiagente, orquestrado pelo framework CrewAI, para criar uma pipeline de processamento cognitivo em três estágios: Roteamento, Análise Técnica e Comunicação. A confiabilidade das informações é assegurada pela técnica de Geração Aumentada por Recuperação (RAG), que ancora as respostas dos agentes em uma base de conhecimento vetorial de documentos legais. O protótipo funcional, focado no Direito do Consumidor, valida a viabilidade da arquitetura. Este documento detalha a fundamentação teórica do projeto, abrangendo desde os modelos de agentes inteligentes (BDI) e a arquitetura Transformer até os princípios de Legal Design e IA Explicável (XAI). A metodologia de pesquisa e de desenvolvimento do software é explicitada, e um framework de avaliação, incluindo métricas de qualidade de IA e Indicadores Chave de Desempenho (KPIs) de TI, é proposto. Conclui-se que a arquitetura de especialização de agentes é uma abordagem robusta e escalável para a democratização do acesso à justiça, estabelecendo uma base sólida para futuros trabalhos.

Palavras-chave: Inteligência Artificial, Sistemas Multiagente, Geração Aumentada por Recuperação (RAG), Direito e Tecnologia, Arquitetura de Software, Legal Design.

ABSTRACT

Access to legal information in Brazil constitutes a complex social challenge, marked by the opacity of legal language and the difficulty of navigating formal justice systems. This work addresses this problem through the design, development, and documentation of an innovative software architecture. The proposed solution employs a multi-agent system, orchestrated by the CrewAI framework, to create a three-stage cognitive processing pipeline: Routing, Technical Analysis, and Communication. The reliability of the information is ensured by the Retrieval-Augmented Generation (RAG) technique, which grounds the agents responses in a vectorial knowledge base of legal documents. The functional prototype, focused on Consumer Law, validates the architecture's feasibility. This document details the projects theoretical foundation, covering concepts from intelligent agent models (BDI) and the Transformer architecture to the principles of Legal Design and Explainable AI (XAI). The research and software development methodologies are explained, and an evaluation framework, including AI quality metrics and IT Key Performance Indicators (KPIs), is proposed. It is concluded that the agent specialization architecture is a robust and scalable approach for democratizing access to justice, establishing a solid foundation for future work.

Keywords: Artificial Intelligence, Multi-Agent Systems, Retrieval-Augmented Generation (RAG), Law and Technology, Software Architecture, Legal Design.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BDI	Belief-Desire-Intention
CDC	Código de Defesa do Consumidor
CERT.br	Computer Emergency Response Team Brazil
CLT	Consolidação das Leis do Trabalho
CPI	Custo por Interação
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DNS	Domain Name Service
FAISS	Facebook AI Similarity Search
GPUs	Graphics Processing Units
GTI	Gestão de Tecnologia da Informação
HTTP	Hypertext Transfer Protocol
IA	Inteligência Artificial
IFSC	Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina
IETF	Internet Engineering Task Force
KPI	Key Performance Indicator
LLM	Large Language Model
MVP	Mínimo Produto Viável
NIC.br	Núcleo de Informação e Coordenação do Ponto BR
ODS	Objetivos de Desenvolvimento Sustentável
PoC	Prova de Conceito
RAG	Retrieval-Augmented Generation
RESTful	Representational State Transfer

RNNs	Redes Neurais Recorrentes
SGSI	Sistema de Gestão de Segurança da Informação
SMA	Sistema Multiagente
SUS	System Usability Scale
TI	Tecnologia da Informação
UX Writing	User Experience Writing
XAI	Inteligência Artificial Explicável

SUMÁRIO

1 INTRODUÇÃO

Este trabalho detalha a concepção e o desenvolvimento de uma arquitetura de software baseada em Inteligência Artificial, projetada para enfrentar um desafio social profundamente enraizado no Brasil: a dificuldade de acesso à informação jurídica pelo cidadão comum. Este capítulo inicial contextualiza o problema da assimetria informacional no campo do Direito, estabelece a justificativa para a criação de uma solução tecnológica, define o problema de pesquisa, os objetivos que nortearam o desenvolvimento e a estrutura geral deste documento.

1.1 JUSTIFICATIVA

O sistema jurídico brasileiro, embora robusto, é caracterizado por uma linguagem e por procedimentos que são, em grande parte, inacessíveis à população leiga. Termos técnicos, processos burocráticos e a própria dispersão das normas criam uma barreira significativa que afasta o cidadão de seus direitos mais básicos. Este fenômeno, conhecido como "assimetria de informação", tem consequências diretas e mensuráveis. Dados da plataforma nacional Consumidor.gov.br, por exemplo, que finalizou mais de 1,3 milhão de reclamações em 2023 (MINISTÉRIO DA JUSTIÇA E SEGURANÇA PÚBLICA, 2024), revelam apenas a ponta de um iceberg. Para cada reclamação formalizada, inúmeras outras são abandonadas devido à frustração, ao desconhecimento dos procedimentos ou a um sentimento generalizado de desamparo.

Essa lacuna cria uma vasta "demanda latente" por orientação jurídica que, quando não atendida, resulta na perda de direitos e na sobrecarga dos canais formais de justiça, como os Procons e os Juizados Especiais, com questões que poderiam ser resolvidas de forma mais simples e direta.

Do ponto de vista da Gestão de Tecnologia da Informação (GTI), este cenário representa um problema clássico de gestão da informação e otimização de serviços (FERNANDES; ABREU, 2014). A "demanda latente" por orientação jurídica é um gargalo operacional que sobrecarrega canais de atendimento e representa uma oportunidade de mercado não atendida. A decisão de um gestor de investir em uma solução tecnológica aqui não é apenas para ganho de eficiência, mas para transformar essa demanda reprimida em um serviço de valor agregado. A tecnologia, em particular a Inteligência Artificial (IA), tem o potencial de atuar como uma ponte, traduzindo a complexidade jurídica em orientação clara e acionável. Este trabalho se justifica, portanto, pela oportunidade de aplicar conceitos modernos de arquitetura de software e IA para desenvolver uma solução que não apenas fornece informação, mas que a torna genuinamente útil, gerando impacto social e servindo

como um estudo de caso sobre a tomada de decisão gerencial na aplicação de tecnologia em domínios especializados.

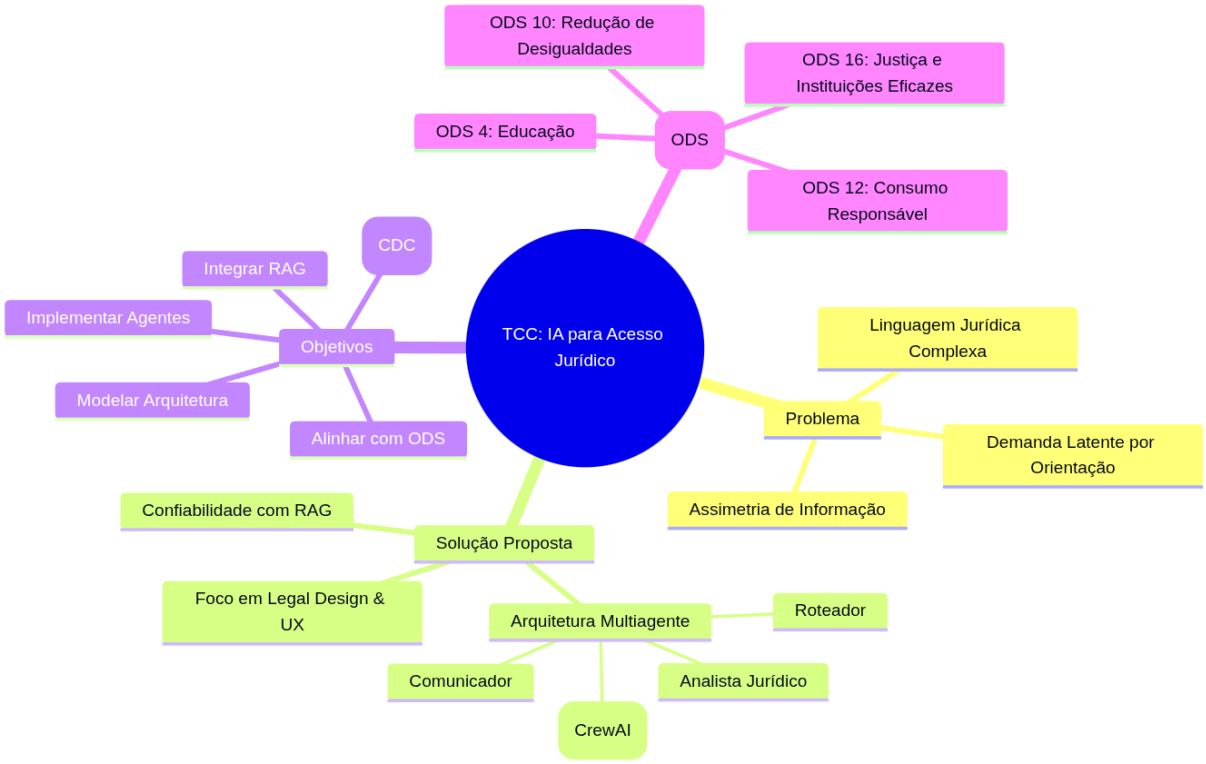
Para uma empresa, a implementação de uma solução como a proposta representa uma decisão de gestão com múltiplos retornos. Primeiramente, atua na redução de custos operacionais, automatizando o primeiro nível de atendimento a dúvidas de consumidores e liberando equipes humanas para casos de maior complexidade. Em segundo lugar, funciona como uma ferramenta de mitigação de riscos jurídicos, pois ao fornecer informação padronizada e baseada em fontes legais, a empresa pode reduzir a incidência de conflitos que escalam para o litígio. Mais importante, o sistema transforma um centro de custo (atendimento ao cliente) em um centro de inteligência de negócio. As dúvidas e problemas reportados pelos usuários tornam-se uma fonte primária de dados para a tomada de decisão, permitindo que gestores identifiquem falhas em produtos, melhorem a comunicação e ajustem estratégias de mercado de forma proativa.

Além do valor gerencial e de negócio, o projeto possui uma forte justificativa social, alinhando-se estrategicamente com os Objetivos de Desenvolvimento Sustentável (ODS), uma coleção de 17 metas globais estabelecidas pela Assembleia Geral das Nações Unidas em 2015 (NAÇÕES UNIDAS, 2015). A solução proposta contribui diretamente para:

- ODS 4 (Educação de Qualidade): Ao promover a educação para a cidadania e o consumo consciente, traduzindo conceitos jurídicos complexos em linguagem simples.
- ODS 10 (Redução das Desigualdades): Ao democratizar o acesso à orientação jurídica qualificada para todos, independentemente da sua condição socioeconômica.
- ODS 12 (Consumo e Produção Responsáveis): Ao capacitar consumidores com conhecimento, incentivando práticas comerciais mais justas.
- ODS 16 (Paz, Justiça e Instituições Eficazes): Ao fortalecer o acesso à justiça e aliviar potencialmente a carga sobre as instituições formais de proteção ao consumidor.

Para consolidar a visão geral do projeto apresentada neste capítulo, o mapa mental a seguir interliga o problema central, a solução proposta, os seus objetivos e o impacto social esperado.

Figure 1 – Mapa Mental: Visão geral do projeto, conectando o problema, a solução, os objetivos e o alinhamento com os ODS.



Fonte: Elaborado pelo autor (2025).

1.2 DEFINIÇÃO DO PROBLEMA

Com base na justificativa apresentada, o problema de pesquisa que este trabalho busca resolver é:

Como arquitetar e desenvolver um sistema de software que utilize Inteligência Artificial para analisar consultas jurídicas de usuários leigos, buscar informações em fontes confiáveis e entregar uma resposta que seja, ao mesmo tempo, precisa em seu conteúdo e simples em sua forma?

Este problema se desdobra em desafios técnicos e de gestão, tais como:

- Confiabilidade da IA: Grandes Modelos de Linguagem (LLMs) são conhecidos por ocasionalmente "alucinar". Como garantir que as respostas do sistema sejam baseadas em leis reais?

- Complexidade da Tarefa: Responder a uma consulta jurídica envolve várias etapas (compreensão, pesquisa, análise, comunicação). Uma única IA monolítica pode não ser eficiente para realizar todas essas tarefas bem.
- Manutenibilidade e Escalabilidade: Como projetar o sistema de forma que ele seja fácil de manter e que, no futuro, possa ser expandido para cobrir outras áreas do Direito sem a necessidade de reconstruir tudo do zero?

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Projetar, desenvolver e documentar uma arquitetura de software funcional que utiliza um sistema multiagente e técnicas de IA para fornecer orientação jurídica preliminar de forma automatizada, confiável e acessível.

1.3.2 OBJETIVOS ESPECÍFICOS

1. Modelar uma arquitetura de software em camadas, separando as responsabilidades de apresentação, orquestração e cognição.
2. Implementar um sistema multiagente, onde cada agente de IA possui uma especialidade e responsabilidade clara dentro de um fluxo de trabalho.
3. Integrar uma técnica de Geração Aumentada por Recuperação (RAG) para conectar os agentes a uma base de conhecimento de documentos legais.
4. Desenvolver um protótipo funcional focado no Direito do Consumidor para validar a arquitetura proposta.

1.4 ESTRUTURA DO TRABALHO

Este documento está organizado em cinco capítulos. O Capítulo 1 (este capítulo) apresenta o projeto. O Capítulo 2 explora a fundamentação teórica. O Capítulo 3 descreve a metodologia de desenvolvimento. O Capítulo 4 detalha a arquitetura e o desenvolvimento do protótipo. Por fim, o Capítulo 5 apresenta as conclusões e aponta direções para trabalhos futuros.

2 REVISÃO DA LITERATURA

Este capítulo aprofunda os conceitos teóricos que fundamentam a arquitetura de software proposta. Para um público multidisciplinar, é essencial que os pilares tecnológicos e de gestão sejam explicados de forma didática, estabelecendo uma base sólida para a compreensão das decisões de design detalhadas no Capítulo 4.

2.1 ARQUITETURAS DE AGENTES INTELIGENTES

2.1.1 AGENTES INTELIGENTES

O conceito de um agente inteligente é a unidade fundamental da Inteligência Artificial moderna. De acordo com a definição canônica, um agente é "qualquer coisa que pode ser vista como percebendo seu ambiente através de sensores e agindo sobre esse ambiente através de atuadores" (RUSSELL; NORVIG, 2021, p. 34). Um agente de IA, portanto, é um programa de computador que exibe esse comportamento, operando de forma autônoma para atingir metas pré-definidas.

Para tornar essa definição mais concreta, um agente inteligente deve exibir autonomia, aprendizado e cooperação. Um termostato, por exemplo, percebe a temperatura (sensor) e liga o aquecedor (atuador), mas não é inteligente, pois seu comportamento é fixo. Um agente de software inteligente, como o proposto neste trabalho, não apenas executa uma tarefa, mas o faz com um grau de independência, adaptando suas ações para atingir um objetivo. Existem diferentes arquiteturas para agentes, desde os simples agentes reativos (que respondem diretamente a estímulos) até agentes baseados em modelos (que mantêm um estado interno do mundo) e agentes baseados em objetivos (que agem para atingir metas). O modelo BDI, discutido a seguir, é uma forma avançada de agente baseado em objetivos (RUSSELL; NORVIG, 2021).

2.1.2 O MODELO COGNITIVO BDI (BELIEF-DESIRE-INTENTION)

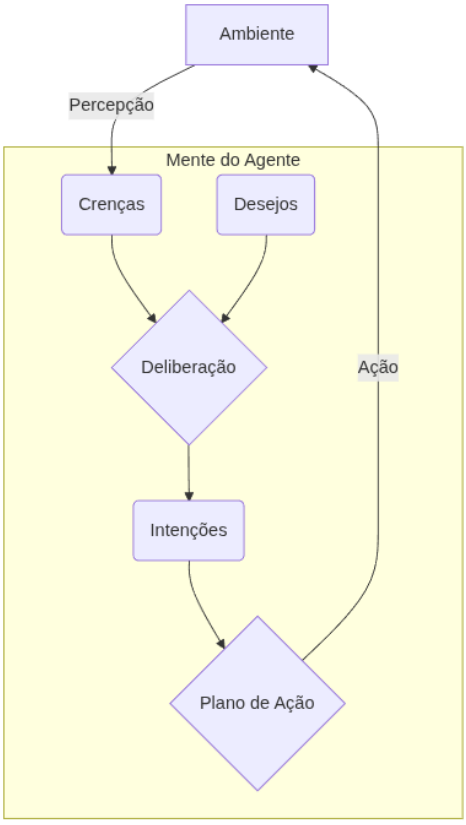
Para que um agente atue de forma racional e deliberada, é útil modelar seu processo de "raciocínio". O modelo BDI é uma arquitetura cognitiva que descreve o comportamento de um agente com base em três componentes mentais (RAO; GEORGEFF, 1995):

- Beliefs (Crenças): A visão de mundo do agente; o que ele acredita ser verdade. No sistema proposto, o backstory de um agente define suas crenças estáticas. As crenças dinâmicas são formadas pela consulta do usuário e pelo contexto retornado por suas ferramentas.

- Desires (Desejos): Os objetivos de alto nível do agente. O goal definido para um agente é uma representação direta de um desejo.
- Intentions (Intenções): O comprometimento do agente em seguir um plano para alcançar um desejo. O thought process de um agente, onde ele decide qual ferramenta usar, pode ser visto como a formação de uma intenção.

Este modelo é particularmente útil para sistemas complexos, pois permite que o comportamento do agente seja programado de forma mais abstrata e compreensível. Em vez de codificar uma série de comandos "se-então", o desenvolvedor define os objetivos (desejos) e o conhecimento (crenças) do agente, e o próprio agente delibera sobre a melhor forma de agir (intenções).

Figure 2 – Diagrama: Ciclo de raciocínio de um agente BDI. O agente percebe o mundo, atualiza suas crenças, delibera sobre seus desejos para formar intenções, cria um plano e age.



Fonte: Elaborado pelo autor (2025).

2.1.3 SISTEMAS MULTIAGENTE (SMA) E SUA JUSTIFICATIVA GERENCIAL

Um Sistema Multiagente (SMA) é um sistema composto por múltiplos agentes que interagem para resolver problemas que seriam difíceis ou impossíveis para um agente único solucionar (WOOLDRIDGE, 2009). A escolha por uma arquitetura SMA, em

detrimento de uma abordagem com um único agente monolítico, é uma decisão estratégica de gestão que responde diretamente às perguntas: "por que essa estrutura?" e "por que essa solução?". A justificativa se baseia em quatro pilares gerenciais:

- Escalabilidade e Custo de Expansão: Para um gestor de TI, a pergunta fundamental é como o sistema pode crescer. A arquitetura multiagente oferece uma resposta clara e de baixo custo. Para expandir o sistema para o Direito Trabalhista, por exemplo, não é necessário treinar um novo modelo gigante e generalista. A equipe pode desenvolver um novo Analista Trabalhista e uma nova base de conhecimento, reutilizando toda a estrutura de orquestração e os agentes de Roteamento e Comunicação existentes. Isso torna a expansão para novas áreas de negócio mais rápida e barata.

- Manutenibilidade e Custo Total de Propriedade (TCO): A clareza de responsabilidades de cada agente simplifica a manutenção. Quando um problema ocorre, é mais fácil diagnosticar qual agente falhou. A configuração dos agentes via Engenharia de Prompts (arquivos de texto) permite que ajustes finos no comportamento sejam feitos por analistas, sem a necessidade de envolver desenvolvedores de software para tudo, reduzindo o custo total de propriedade (TCO) da solução.

- Mitigação de Risco e Complexidade: Um único agente responsável por todas as tarefas (compreender, pesquisar, analisar, formatar) se torna um "ponto único de falha". A especialização de agentes isola os problemas. Se o agente Comunicador não está sendo empático o suficiente, a equipe pode ajustá-lo sem o risco de impactar a precisão da análise jurídica. Isso reduz o risco de regressões e simplifica os ciclos de teste.

- Otimização de Desempenho e Custo: A especialização permite otimizar o modelo de IA para cada tarefa. A Análise Técnica pode exigir um modelo de linguagem mais poderoso e caro, enquanto o Roteamento pode usar um modelo mais simples, rápido e barato. Essa granularidade permite ao gestor otimizar o balanço entre custo de API e qualidade da resposta, uma decisão puramente gerencial.

- Aplicação na Arquitetura Proposta: A arquitetura deste projeto, com sua equipe de agentes (Roteador, Analista Jurídico, Comunicador), materializa esses benefícios. A colaboração em um fluxo de trabalho demonstra não apenas a eficácia técnica da especialização, mas sua validade como uma decisão de gestão para construir uma solução robusta, sustentável e escalável.

A abordagem de agentes especializados pode ser vista como um análogo, no mundo da IA, à filosofia de **microsserviços** na engenharia de software. Em vez de construir uma aplicação monolítica, a arquitetura de microsserviços estrutura uma aplicação como uma coleção de serviços pequenos, autônomos e fracamente acoplados (NEWMAN, 2015). Da

mesma forma, a arquitetura multiagente deste trabalho decompõe um problema cognitivo complexo em tarefas menores, atribuídas a agentes especializados, ganhando em troca a mesma flexibilidade, escalabilidade e manutenibilidade.

2.2 TECNOLOGIAS DE PROCESSAMENTO DE LINGUAGEM

2.2.1 A ARQUITETURA TRANSFORMER

Os Grandes Modelos de Linguagem (LLMs) são a tecnologia central que impulsiona os agentes do sistema. Sua capacidade de compreender e gerar texto vem da arquitetura Transformer (VASWANI, 2017). Diferente de modelos anteriores que liam o texto em sequência, o Transformer processa todas as palavras de uma vez. Sua inovação fundamental é o mecanismo de autoatenção (self-attention), que permite ao modelo ponderar a importância de cada palavra em relação a todas as outras na frase, capturando o contexto de forma muito mais rica e eficaz. É essa capacidade de entender o contexto que permite aos LLMs realizar tarefas complexas de linguagem.

O mecanismo de autoatenção funciona calculando "pesos de atenção" para cada palavra em relação a todas as outras. Uma palavra como "ele" em "o advogado ajudou o cliente porque ele era experiente" terá sua atenção fortemente direcionada para "advogado", permitindo que o modelo resolva a ambiguidade. Essa capacidade de capturar dependências de longo prazo no texto é o que diferencia o Transformer de arquiteturas anteriores, como as Redes Neurais Recorrentes (RNNs).

2.2.2 BUSCA SEMÂNTICA, EMBEDDINGS E BANCOS DE DADOS VETORIAIS

O grande avanço dos LLMs foi sua capacidade de ir além da busca por palavra-chave, realizando uma busca semântica. Isso é possível através de Embeddings, que são representações vetoriais (listas de números) de textos (MIKOLOV, 2013). Textos com significados parecidos são posicionados próximos em um "espaço vetorial semântico". Para que essa busca seja eficiente, são utilizados Bancos de Dados Vetoriais, como o FAISS (JOHNSON et al., 2019), que são otimizados para encontrar rapidamente os vetores mais próximos de uma dada consulta.

Por exemplo, em um sistema de busca semântica, a consulta "posso devolver um produto que comprei online?" seria convertida em um vetor. Esse vetor seria então usado para encontrar, no banco de dados vetorial, os trechos de texto cujos vetores são mais próximos, como o Artigo 49 do CDC, que trata do "direito de arrependimento", mesmo que a palavra "arrependimento" não estivesse na pergunta original.

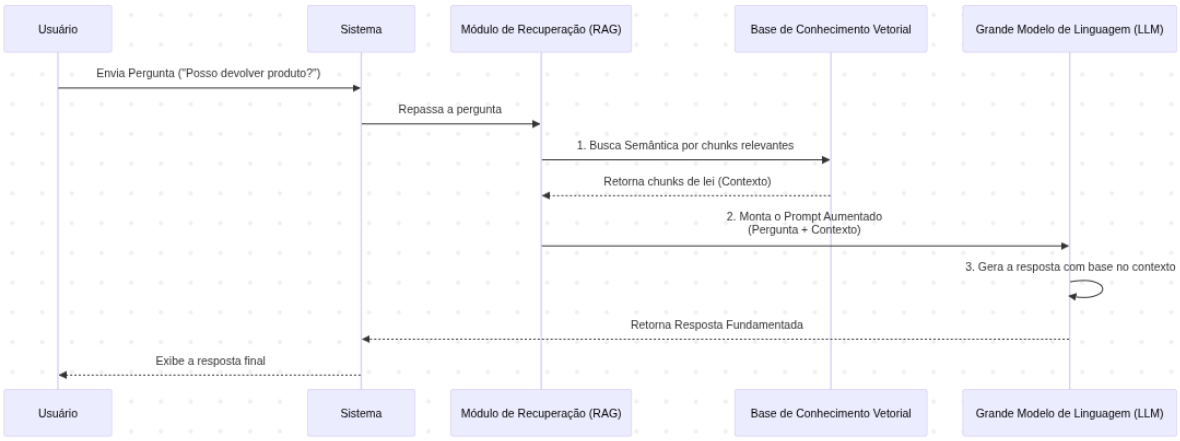
2.2.3 GERAÇÃO AUMENTADA POR RECUPERAÇÃO (RAG)

A RAG é uma técnica que torna a IA mais confiável ao forçá-la a basear suas respostas em um conjunto de documentos fornecidos, como em uma "prova com consulta" (LEWIS, 2020). O processo, ilustrado no Diagrama 2, combina a busca semântica (Recuperação) com a capacidade de geração de texto do LLM (Geração).

O principal motivador para o uso da RAG é mitigar o risco de **alucinações** dos LLMs. Uma alucinação ocorre quando o modelo gera informações que são factualmente incorretas, não relacionadas ao contexto ou simplesmente inventadas. Isso acontece porque os LLMs são, em sua essência, modelos probabilísticos treinados para prever a próxima palavra mais plausível em uma sequência, e não para acessar uma base de fatos. A RAG contorna esse problema ao "aterrar" (ground) o modelo em um contexto factual e explícito, instruindo-o a basear sua resposta apenas nos documentos recuperados.

É crucial distinguir a RAG de uma simples busca semântica. Uma busca semântica apenas encontra e retorna os trechos de texto mais relevantes de uma base de conhecimento. O resultado seria uma lista de artigos de lei, que, embora precisos, ainda exigiriam que o usuário os interpretasse. A RAG vai além: ela usa os resultados da busca semântica como contexto para que um modelo de linguagem generativo (o "G" da sigla) construa uma resposta coesa, contextualizada e em linguagem natural. Portanto, a RAG não apenas encontra a informação, mas a sintetiza em uma orientação clara, o que é essencial para o objetivo deste trabalho de democratizar a informação (LEWIS, 2020).

Figure 3 – Diagrama: Fluxo do processo de Geração Aumentada por Recuperação (RAG). O sistema recupera informações relevantes antes de gerar a resposta.



Fonte: Elaborado pelo autor (2025).

Aplicação no Sistema Proposto: A RAG é a espinha dorsal da confiabilidade do sistema. Ela garante que, ao responder sobre o direito de um consumidor, o agente especialista está consultando ativamente o texto do Código de Defesa do Consumidor, e não apenas "lembrando" de informações.

2.2.4 ENGENHARIA DE PROMPTS (PROMPT ENGINEERING)

A Engenharia de Prompts é a prática de projetar cuidadosamente as instruções (role, goal, backstory) dadas a um modelo de IA para controlar seu comportamento (LIU, 2023). Essa técnica é uma ferramenta de gestão crucial, pois permite que o comportamento dos agentes seja ajustado através de arquivos de texto, sem reprogramar o modelo.

Um prompt eficaz não é apenas uma pergunta, mas um conjunto de instruções que molda a "personalidade" e o foco do agente. Por exemplo, o prompt do "Agente Comunicador" não diz apenas "reescreva o texto", mas inclui diretrizes como "use uma linguagem empática", "evite jargões jurídicos" e "foque em ações práticas que o usuário pode tomar". Isso transforma um LLM genérico em um especialista em comunicação.

Do ponto de vista estratégico, a engenharia de prompts oferece uma alternativa mais ágil e de menor custo ao **fine-tuning** (ajuste fino). O fine-tuning é o processo de retreinar um LLM pré-treinado em um conjunto de dados específico para especializá-lo em uma tarefa. Embora poderoso, o fine-tuning exige grandes volumes de dados, alto custo computacional e expertise técnica. A engenharia de prompts, por outro lado, permite a especialização "em tempo de execução", oferecendo uma flexibilidade muito maior para prototipagem e ajustes iterativos, o que é ideal para o contexto deste projeto.

2.3 IA EXPLICÁVEL (XAI): ABRINDO A CAIXA-PRETA

2.3.1 O PROBLEMA DA OPACIDADE E A NECESSIDADE DE CONFIANÇA

Muitos modelos avançados de IA, especialmente em aprendizado profundo, funcionam como uma **"caixa-preta" (black box)**. Eles recebem dados de entrada e produzem uma saída (uma decisão ou previsão), mas o processo interno que leva a essa saída é tão complexo que é ininteligível para um observador humano. Em domínios de baixo risco, como a recomendação de filmes, isso pode ser aceitável. No entanto, em áreas de alto risco como o Direito, a medicina ou as finanças, a incapacidade de entender *por que* um sistema de IA tomou uma determinada decisão representa um risco inaceitável e uma barreira para a adoção (ARRIETA, 2020).

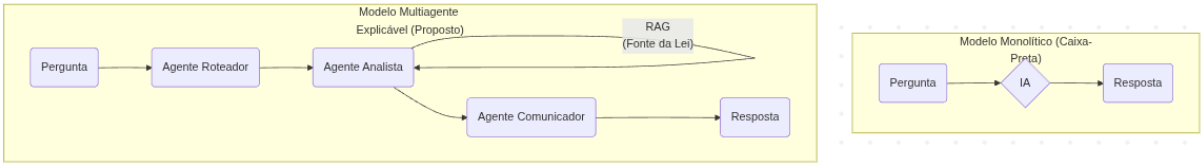
A **Inteligência Artificial Explicável (XAI)** é um campo de pesquisa e prática que visa resolver esse problema, desenvolvendo sistemas de IA que possam fornecer explicações claras e compreensíveis sobre suas decisões. O programa XAI da DARPA (Agência de Projetos de Pesquisa Avançada de Defesa dos EUA) define o objetivo como a criação de "parceiros mais confiáveis" para os humanos, promovendo a capacidade de entender, confiar e gerenciar a nova geração de sistemas de IA (Defense Advanced Research Projects Agency (DARPA), 2016). Um sistema explicável, segundo a DARPA, deve ser capaz de responder a perguntas como: "Por que você fez isso?", "Por que não outra coisa?" e "Quando você falha?".

2.3.2 A ARQUITETURA DESTE PROJETO COMO UMA SOLUÇÃO DE XAI

A arquitetura proposta neste trabalho foi deliberadamente projetada para ser explicável por natureza, transformando a XAI de um requisito a ser adicionado posteriormente em um pilar fundamental do design. A explicabilidade é alcançada em dois níveis complementares, que respondem a duas perguntas cruciais do usuário: "Com base em quê?" e "Como?".

A literatura de XAI distingue entre explicabilidade **global** (entender o comportamento geral do modelo) e explicabilidade **local** (entender por que o modelo tomou uma decisão específica para uma entrada particular) (ARRIETA, 2020). A arquitetura deste projeto se destaca ao fornecer uma forte explicabilidade local para cada consulta do usuário.

Figure 4 – Diagrama: Comparação entre um modelo de IA "caixa-preta" e a abordagem multiagente, que torna o processo de decisão transparente.



Fonte: Elaborado pelo autor (2025).

1. Transparência da Fonte (O quê?): Este é o primeiro nível de explicação, garantido pela técnica RAG. O sistema não apenas gera uma resposta, mas também cita a fonte legal específica (o artigo do Código de Defesa do Consumidor, por exemplo) que usou como base. Isso responde à pergunta: "Com base em que você está me dizendo isso?". Para um gestor, isso garante que a operação está ancorada em regras de negócio verificáveis, e não em conhecimento opaco da IA.

2. Transparência do Processo (Como?): Este é o segundo e mais profundo nível de explicação, possibilitado pela arquitetura multiagente. Ao invés de uma única IA "caixa-preta" que gera uma resposta, o sistema segmenta o raciocínio em etapas claras e auditáveis. É possível rastrear o fluxo: o Agente Roteador classificou a pergunta, o Agente Analista buscou a informação e formulou a lógica, e o Agente Comunicador traduziu o resultado para o usuário. Isso responde à pergunta: "Como você chegou a essa conclusão?". Para um gestor, essa transparência é crucial para depuração, otimização e, acima de tudo, para construir confiança na automação de processos complexos.

Aplicação no Sistema Proposto: A combinação de RAG e uma pipeline multiagente não é apenas uma escolha técnica, mas uma decisão de design focada em governança e confiança. Ela permite que o sistema seja auditado, que seus erros sejam diagnosticados com precisão (foi um erro de roteamento, de análise ou de comunicação?) e que a sua

operação seja compreendida por stakeholders não-técnicos, um requisito fundamental para a adoção de IA em ambientes corporativos.

2.4 CONCEITOS DO DOMÍNIO DE APLICAÇÃO

2.4.1 LEGAL DESIGN E UX WRITING

O Legal Design aplica os princípios do Design Thinking ao mundo jurídico, focando em criar serviços que sejam centrados no ser humano (HAGAN, 2022). No contexto deste projeto, isso se manifesta na busca pela empatia, que não é um mero detalhe, mas um requisito funcional crítico. Atingir esse nível de comunicação eficaz exige um esforço deliberado de UX Writing (a prática de escrever o texto visto pelos usuários em uma interface
) na criação da persona do communication$_agent, comtestesiterativosparaavaliararespostaemocional$

2.5 GESTÃO ESTRATÉGICA DE TECNOLOGIA

As decisões de arquitetura de software não ocorrem em um vácuo; elas estão intrinsecamente ligadas a objetivos de negócio e à gestão de recursos. Esta seção aborda os conceitos de gestão que fundamentam as escolhas feitas neste projeto.

2.5.1 CUSTO TOTAL DE PROPRIEDADE (TCO)

O Custo Total de Propriedade (TCO) é uma análise financeira projetada para avaliar o custo completo de um ativo de tecnologia ao longo de seu ciclo de vida (FERNANDES; ABREU, 2014). Ele vai além do preço de aquisição, englobando todos os custos diretos e indiretos, como implementação, manutenção, treinamento, suporte e operação (e.g., custos de API). A decisão por uma arquitetura multiagente e modular, conforme discutido na seção 2.1.3, é uma estratégia para otimizar o TCO. A facilidade de manutenção e a capacidade de expandir o sistema de forma incremental (adicionando novos agentes sem reestruturar o todo) reduzem os custos de longo prazo, tornando o investimento mais sustentável.

2.5.2 INDICADORES CHAVE DE DESEMPENHO (KPIS) E INTELIGÊNCIA DE NEGÓCIO (BI)

Indicadores Chave de Desempenho (KPIs) são valores mensuráveis que demonstram quão eficazmente uma organização está atingindo seus objetivos de negócio (ECKERSON, 2010). No contexto de TI, eles são usados para monitorar a saúde, a eficiência e o impacto dos sistemas. A Inteligência de Negócio (BI), por sua vez, refere-se ao processo de usar

tecnologias para transformar dados brutos em insights acionáveis que informam a tomada de decisão estratégica (TURBAN, 2011).

Um sistema como o proposto neste trabalho é projetado para ser mais do que uma ferramenta operacional; ele é um motor de geração de dados. Os KPIs propostos no framework de avaliação (Capítulo 4) — como as dúvidas mais frequentes dos usuários ou a taxa de sucesso da IA — são a matéria-prima para um processo de BI. Eles permitem que um gestor transforme o que tradicionalmente é um centro de custo (atendimento ao cliente) em um centro de inteligência, usando dados para guiar decisões sobre melhoria de produtos, estratégias de comunicação e alocação de recursos.

2.6 PRINCÍPIOS DE ARQUITETURA E DESENVOLVIMENTO DE SOFTWARE

2.6.1 ARQUITETURA EM CAMADAS E SEPARAÇÃO DE PREOCUPAÇÕES

A arquitetura em camadas é um padrão de design de software que organiza uma aplicação em camadas horizontais, cada uma com uma responsabilidade específica (e.g., Apresentação, Lógica de Negócio, Acesso a Dados) (FOWLER, 2002). Este padrão implementa o princípio fundamental da Separação de Preocupações (Separation of Concerns), que advoga pela divisão de um sistema em partes distintas com funcionalidades sobrepostas mínimas. A arquitetura em quatro camadas (Apresentação, Orquestração, Cognição, Conhecimento) adotada neste trabalho é uma aplicação direta desse princípio, resultando em um sistema mais modular, flexível e de fácil manutenção, onde cada camada pode ser modificada ou substituída com impacto mínimo sobre as outras.

2.6.2 PROTOTIPAGEM EVOLUTIVA

A prototipagem evolutiva é uma metodologia de desenvolvimento de software na qual um protótipo inicial é construído e, em seguida, refinado iterativamente com base no feedback dos usuários até se tornar o produto final (SOMMERVILLE, 2011). Diferente de modelos rígidos e sequenciais, como o modelo em cascata, essa abordagem é ideal para projetos inovadores onde os requisitos não são completamente conhecidos no início. Conforme descrito na metodologia (Capítulo 3), a adoção da prototipagem evolutiva permitiu a validação rápida de hipóteses sobre a interação do usuário e a eficácia dos agentes, reduzindo o risco de desenvolver uma solução que não atende às necessidades reais do público-alvo.

3 PROCEDIMENTOS METODOLÓGICOS

Este trabalho adotou uma abordagem dupla de metodologia, distinguindo a metodologia da pesquisa acadêmica da metodologia de desenvolvimento do artefato de software.

3.1 METODOLOGIA DA PESQUISA

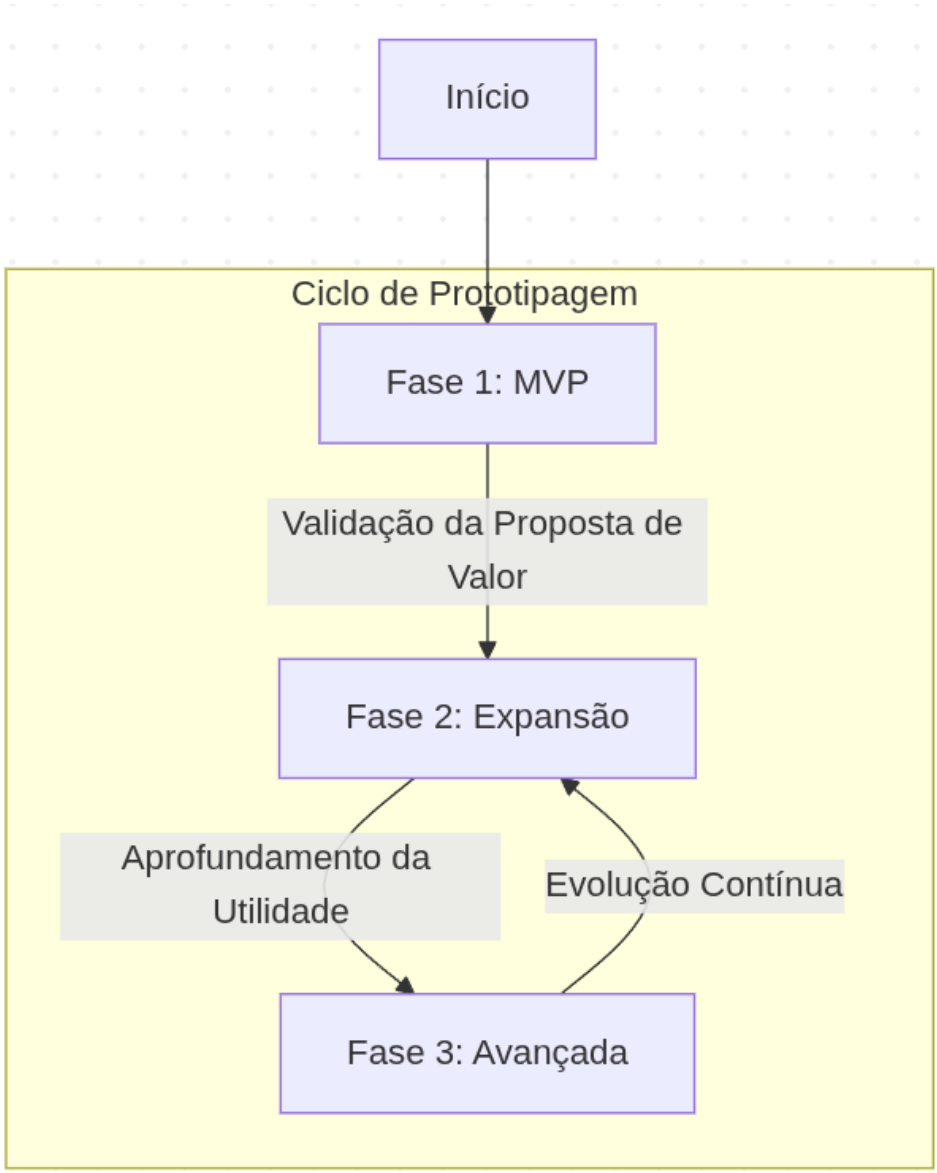
Quanto aos fins, esta pesquisa é classificada como aplicada, pois visa gerar conhecimentos para a aplicação prática e dirigida à solução de um problema específico (GIL, 2008). Quanto aos meios, a pesquisa é de natureza exploratória, buscando proporcionar maior familiaridade com o problema, e descritiva, descrevendo as características da solução proposta. Os procedimentos seguiram uma sequência lógica:

1. Revisão Bibliográfica: Investigação dos conceitos de Sistemas Multiagente, RAG, Legal Design e outros, para construir o embasamento teórico.
2. Modelagem da Arquitetura: Desenho da arquitetura de software em camadas e da pipeline de agentes, como solução para o problema de pesquisa.
3. Desenvolvimento do Protótipo: Implementação do artefato de software como prova de conceito e objeto de estudo.
4. Análise e Documentação: Descrição detalhada da arquitetura, dos componentes e dos resultados, consolidada neste documento.

3.2 METODOLOGIA DE DESENVOLVIMENTO DO SOFTWARE

Para a construção do protótipo, foi adotada uma metodologia de **prototipagem evolutiva**. Essa abordagem é ideal para projetos de inovação, onde os requisitos nem sempre são totalmente conhecidos no início. O processo consiste em construir uma versão inicial funcional (o protótipo), testá-la, aprender com ela e, em seguida, usar esse aprendizado para refinar e adicionar novas funcionalidades em ciclos (SOMMERVILLE, 2011). O roteiro de desenvolvimento foi planejado em fases, conforme ilustrado no diagrama abaixo.

Figure 5 – Diagrama: Metodologia de Prototipagem Evolutiva em Fases.



Fonte: Elaborado pelo autor (2025).

A seguir, detalhamos cada fase do ciclo de desenvolvimento do protótipo.

3.2.1 FASE 1: PROVA DE CONCEITO (POC) DA TECNOLOGIA RAG

O primeiro e mais crítico passo foi validar a hipótese fundamental do projeto: era tecnicamente viável usar a técnica de Geração Aumentada por Recuperação (RAG) para responder a perguntas com base no texto do Código de Defesa do Consumidor (CDC)? Esta fase focou exclusivamente na Camada de Conhecimento.

- **Objetivo:** Validar a eficácia da busca semântica e da geração de respostas fundamentadas.
- **Atividades:**

1. Preparação da Base de Conhecimento: O texto do CDC foi obtido e tratado, sendo dividido em trechos (chunks) para a vetorização.
2. Implementação do RAG: Um script simples em Python foi criado utilizando a biblioteca LangChain para orquestrar o processo de: a) receber uma pergunta, b) convertê-la em um embedding, c) usar o FAISS para encontrar os trechos mais relevantes do CDC, e d) passar esses trechos como contexto para um LLM da OpenAI gerar uma resposta.
3. Testes Iniciais: Foram realizados testes manuais com perguntas-chave para avaliar a precisão do contexto recuperado e a fidelidade da resposta gerada.

- **Resultado Esperado:** Confirmação de que a abordagem RAG era capaz de encontrar os artigos corretos do CDC e gerar respostas factualmente corretas, mitigando o risco de "alucinações". O sucesso desta fase foi o "sinal verde" para prosseguir com o desenvolvimento.

3.2.2 FASE 2: DESENVOLVIMENTO DO AGENTE ESPECIALISTA MONOLÍTICO

Uma vez validada a tecnologia RAG, o próximo passo foi encapsular essa lógica dentro de um agente autônomo. Nesta fase, o foco foi na Camada de Cognição, mas com um único agente "faz-tudo".

- **Objetivo:** Criar um único agente capaz de receber uma pergunta, usar a ferramenta RAG e gerar uma resposta final para o usuário.
- **Atividades:**
  1. Criação do Agente "Advogado Geral": Usando o framework CrewAI, foi criado um único agente. O prompt deste agente era complexo, instruindo-o a ser, ao mesmo tempo, um analista técnico e um bom comunicador.
  2. Integração da Ferramenta RAG: A lógica da PoC da Fase 1 foi transformada em uma "ferramenta" formal (tool) que o agente do CrewAI poderia invocar.
  3. Testes de Desempenho: Foram realizados testes para avaliar a qualidade das respostas geradas por este agente único.
- **Aprendizado e Iteração:** Os testes revelaram uma limitação fundamental da abordagem monolítica. O agente frequentemente tinha dificuldade em equilibrar as duas personas: ora gerava uma resposta tecnicamente precisa, mas excessivamente densa e jurídica; ora gerava uma resposta simples e amigável, mas que sacrificava detalhes técnicos importantes. Essa dificuldade em "vestir dois chapéus" ao mesmo tempo foi a principal justificativa para evoluir para uma arquitetura multiagente.

3.2.3 FASE 3: IMPLEMENTAÇÃO DA PIPELINE MULTIAGENTE

Com base no aprendizado da fase anterior, o sistema foi redesenhado para a arquitetura multiagente final, que é o coração deste trabalho.

- **Objetivo:** Implementar a pipeline de três estágios (Roteamento, Análise, Comunicação) para especializar as tarefas e melhorar a qualidade da resposta final.
- **Atividades:**
    1. Decomposição de Responsabilidades: O agente monolítico foi decomposto em três agentes especializados, conforme descrito em detalhes no Capítulo 4: o Roteador, o Analista Jurídico e o Comunicador.
    2. Criação da "Tripulação" (Crew): O arquivo ‘crew$_m$*anager.py*‘$foicriadoparaorquestraracol$

    2. Refinamento dos Prompts: Os prompts de cada agente foram reescritos para serem altamente focados em sua única responsabilidade, eliminando a ambiguidade da fase anterior.
    3. Testes Comparativos: Novos testes foram realizados e os resultados foram comparados com os da Fase 2. A melhora na qualidade, clareza e precisão da resposta final foi notável, validando a superioridade da abordagem multiagente.
- **Resultado Esperado:** Um protótipo funcional que implementa a arquitetura completa descrita no Capítulo 4, capaz de gerar respostas de alta qualidade de forma consistente e auditável.

3.2.4 JUSTIFICATIVA DA STACK DE TECNOLOGIA PARA O MVP

Uma decisão estratégica de gestão na fase de MVP (Mínimo Produto Viável) foi a seleção de uma stack de tecnologia focada na agilidade e no baixo custo, utilizando plataformas No-code/Low-code como o **Make.com** para orquestração de fluxos e o **Google Sheets** para armazenamento inicial de dados e logs.

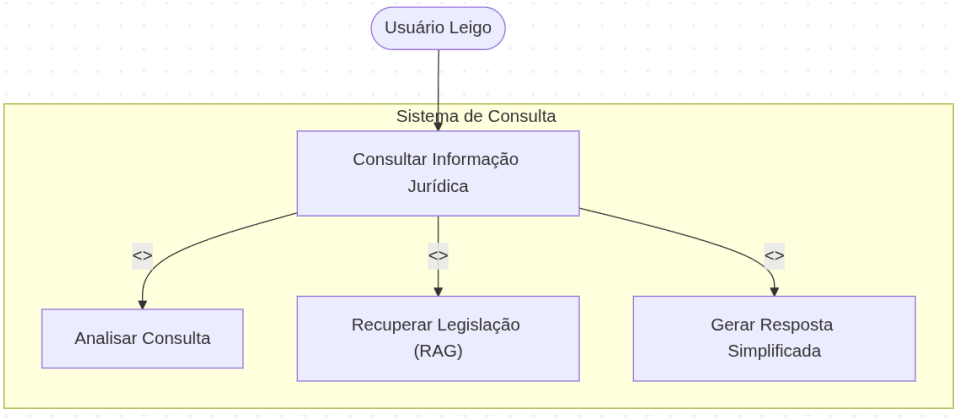
Essa abordagem, embora não seja escalável para produção, foi fundamental para a metodologia de prototipagem evolutiva. Ela permitiu a validação rápida de hipóteses de interface e fluxo de usuário sem o investimento inicial em uma infraestrutura de software complexa (como a construção de um front-end web e um banco de dados robusto). O núcleo da solução, a API com a lógica dos agentes, foi desenvolvido em Python utilizando CrewAI, FastAPI e LangChain/FAISS, garantindo que o "coração" do sistema fosse sólido e preparado para futuras evoluções, enquanto as "bordas" (interface e armazenamento) eram flexíveis e de baixo custo para permitir a experimentação ágil.

4 DESENVOLVIMENTO DO TRABALHO E APRESENTAÇÃO DE RESULTADOS

O principal resultado deste trabalho é o protótipo funcional que materializa a arquitetura proposta. Este capítulo aprofunda os detalhes técnicos do sistema, descrevendo como as camadas lógicas interagem, como os componentes de software executam suas funções e como a eficácia do sistema pode ser rigorosamente avaliada. O objetivo desta seção é ir além da descrição superficial, oferecendo um mergulho didático na implementação de cada componente, justificando as escolhas de tecnologia e design para que sirva como um guia para estudantes e profissionais de outras áreas.

Para contextualizar o desenvolvimento do ponto de vista do usuário, o diagrama a seguir modela o principal caso de uso do sistema. Ele ilustra a interação fundamental entre o ator ("Usuário Leigo") e a funcionalidade central do sistema, "Consultar Informação Jurídica".

Figure 6 – Diagrama de Caso de Uso principal do sistema.



Fonte: Elaborado pelo autor (2025).

O diagrama mostra que a principal interação do usuário (a consulta) é um processo composto. Para que o sistema entregue o valor esperado, ele internamente precisa executar um conjunto de subtarefas obrigatórias, representadas pela relação «include»:

- Analisar Consulta: O sistema primeiro precisa compreender a intenção e as entidades na pergunta do usuário.
- Recuperar Legislação (RAG): Em seguida, busca a informação relevante na base de conhecimento.
- Gerar Resposta Simplificada: Por fim, traduz a informação técnica para uma linguagem acessível.

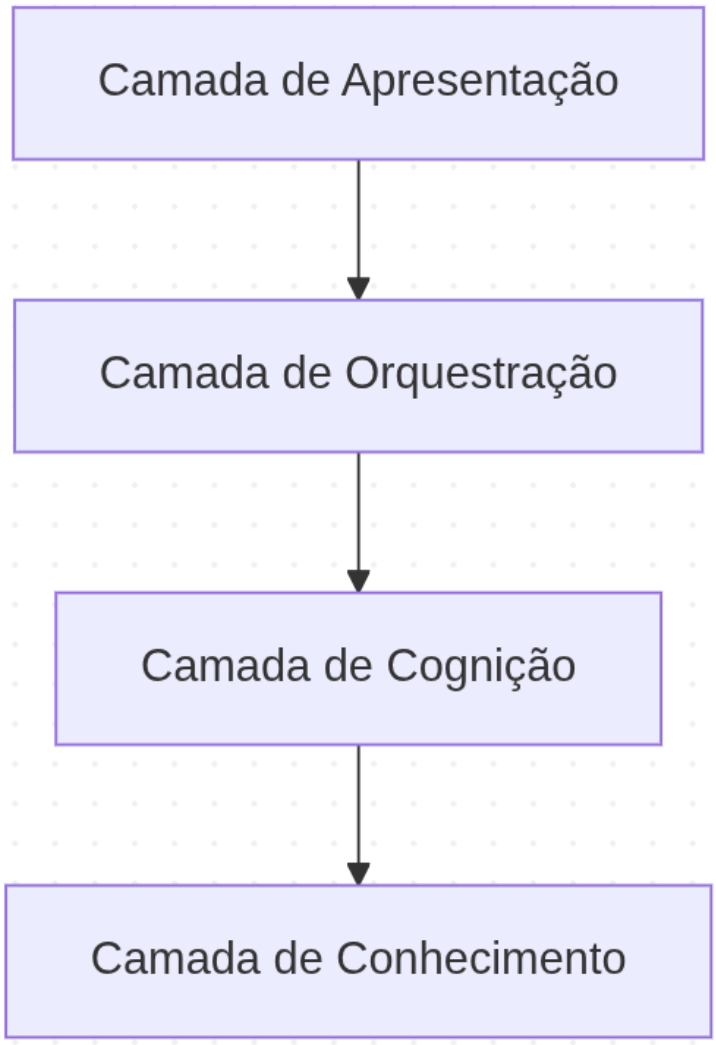
Este modelo de caso de uso serve como uma ponte entre a perspectiva do usuário e a arquitetura de software detalhada nas seções seguintes, que descreve como esses casos de uso são implementados.

As seções a seguir detalham as camadas do sistema e o framework de avaliação.

4.1 ARQUITETURA LÓGICA EM CAMADAS: UMA ANÁLISE APROFUNDADA

A arquitetura do sistema foi concebida em quatro camadas lógicas principais, garantindo a separação de preocupações (Separation of Concerns), a modularidade e a manutenibilidade, princípios fundamentais da Engenharia de Software. A seguir, cada camada é explorada em detalhes.

Figure 7 – Diagrama: Arquitetura Lógica em Camadas.



Fonte: Elaborado pelo autor (2025).

4.1.1 CAMADA DE APRESENTAÇÃO: A PORTA DE ENTRADA

A Camada de Apresentação é a porta de entrada do sistema, responsável por receber as requisições dos usuários e formatá-las para o processamento interno. No protótipo, esta camada é implementada como uma API RESTful utilizando o framework **FastAPI**.

A escolha pelo FastAPI não foi acidental. Para um projeto que envolve IA, a performance é crucial, e o FastAPI é um dos frameworks web Python mais rápidos disponíveis, graças ao seu suporte nativo a operações assíncronas. Além disso, ele oferece duas vantagens estratégicas para o desenvolvimento ágil:

- **Validação de Dados com Pydantic:** O FastAPI utiliza a biblioteca Pydantic para declarar os modelos de dados da API. Isso significa que toda requisição que chega ao sistema é automaticamente validada. Se um campo obrigatório estiver faltando ou se um tipo de dado estiver incorreto, o FastAPI retorna um erro claro e informativo ao cliente, antes mesmo que a requisição chegue à lógica de negócio. Isso torna a API mais robusta e segura.
- **Documentação Automática (Swagger UI):** O FastAPI gera automaticamente uma documentação interativa da API (usando os padrões OpenAPI e Swagger UI). Isso é imensamente valioso, pois permite que qualquer pessoa (incluindo outros desenvolvedores ou testadores) possa visualizar os endpoints disponíveis, seus parâmetros e até mesmo testá-los diretamente pelo navegador, sem a necessidade de ferramentas adicionais.

Sua função é expor um *endpoint* HTTP que aceita a pergunta do usuário, garantindo que a interação com o restante do sistema seja padronizada e segura.

4.1.2 CAMADA DE ORQUESTRAÇÃO: O CÉREBRO DA OPERAÇÃO

A Camada de Orquestração gerencia o fluxo de trabalho e a colaboração entre os agentes. Esta camada não executa o trabalho cognitivo em si, mas atua como um "maestro", definindo qual agente deve atuar em cada etapa e como a informação flui entre eles. O framework **CrewAI** é o principal componente aqui.

O CrewAI foi escolhido por sua abordagem intuitiva e poderosa para a criação de sistemas multiagente, baseada em três conceitos principais:

- **Agents:** São os trabalhadores da equipe. Cada agente é configurado com um ‘role‘ (papel), ‘goal‘ (objetivo), ‘backstory‘ (contexto) e, opcionalmente, ‘tools‘ (ferramentas) que ele pode usar. Essa configuração, feita via Engenharia de Prompts, transforma um LLM genérico em um especialista focado.

- **Tasks:** São as tarefas a serem executadas. Cada tarefa tem uma descrição clara e um agente designado para realizá-la. Crucialmente, as tarefas podem receber o *contexto* da tarefa anterior, criando uma cadeia de processamento.

- **Crew:** É a equipe formada pelos agentes e pelas tarefas. É na 'Crew' que se define o processo de execução. Para este projeto, foi utilizado o **Process.sequential**, que garante que as tarefas sejam executadas em uma ordem linear e previsível: a saída da Tarefa 1 se torna a entrada da Tarefa 2, e assim por diante. Essa linearidade é fundamental para garantir a transparência e a auditabilidade do processo de raciocínio, um dos pilares da XAI.

O CrewAI gerencia a complexidade de passar as informações entre os agentes e o LLM, permitindo que o desenvolvedor se concentre na lógica de negócio (definir os agentes e as tarefas) em vez de na infraestrutura de comunicação.

4.1.3 CAMADA DE COGNIÇÃO: OS ESPECIALISTAS EM AÇÃO

É nesta camada que o "raciocínio" de fato acontece. Cada agente é uma instância de um LLM configurado com um prompt específico para se tornar um especialista. A seguir, detalhamos os três agentes principais do protótipo:

4.1.3.1 AGENTE 1: O ROTEADOR (ROUTER AGENT)

- **Justificativa de Design:** Em um sistema projetado para escalar, é inviável ter um único agente que conheça todas as áreas do Direito. O Roteador atua como um "triador" inteligente. Sua única responsabilidade é analisar a pergunta do usuário e direcioná-la para o especialista mais adequado. Isso torna o sistema modular e fácil de expandir: para adicionar uma nova área (ex: Direito Trabalhista), basta criar um novo agente especialista e ensinar o Roteador a reconhecer perguntas sobre esse tópico.

- **Prompt (Exemplo Simplificado):**

  > **Role:** Roteador de Consultas Jurídicas
  > **Goal:** Analisar a pergunta do usuário e identificar a área do Direito correspondente, escolhendo o especialista mais qualificado da lista: [Advogado do Consumidor, Advogado Criminalista, Advogado Cível].
  > **Backstory:** Você é um assistente de triagem em um grande escritório de advocacia. Sua função é ler a pergunta de um novo cliente e encaminhá-la para o departamento correto, garantindo uma resposta rápida e eficiente. Você deve ser preciso e retornar apenas o nome do cargo do especialista escolhido.

4.1.3.2 AGENTE 2: O ANALISTA JURÍDICO (ADVOGADO DO CONSUMIDOR)

- **Justificativa de Design:** Este é o coração técnico da análise. Sua função é receber a pergunta do usuário (já validada pelo Roteador) e realizar uma análise aprofundada e precisa. Para garantir a confiabilidade e mitigar o risco de alucinações, este agente é o único que tem acesso à ferramenta de RAG.

- **Prompt (Exemplo Simplificado):**

  Role: Advogado Especialista em Direito do Consumidor
  Goal: Fornecer uma análise jurídica precisa e detalhada sobre a pergunta do usuário, baseando-se estritamente nas informações recuperadas da sua ferramenta de busca na legislação.
  Backstory: Você é um advogado experiente e meticuloso, especializado no Código de Defesa do Consumidor. Sua principal preocupação é a precisão. Você NUNCA responde com base em seu conhecimento geral; você sempre cita a fonte legal para cada afirmação que faz. Sua resposta deve ser técnica e completa, servindo como um parecer para outro profissional.

- **Ferramentas Associadas:** ‘rag$_{t}ool$‘($parabuscarnabasedeconhecimentodoCDC$).

4.1.3.3 AGENTE 3: O COMUNICADOR (COMMUNICATION AGENT)

- **Justificativa de Design:** A precisão técnica, por si só, não democratiza a informação. Um parecer jurídico denso pode ser tão inútil para um leigo quanto nenhuma informação. O Comunicador resolve este problema atuando como um "tradutor". Ele pega a análise técnica e densa do Analista Jurídico e a reescreve em uma linguagem simples, empática e acionável, focando nos princípios do Legal Design.

- **Prompt (Exemplo Simplificado):**

  Role: Especialista em Comunicação e Legal Design
  Goal: Transformar a análise jurídica técnica recebida em uma resposta clara, amigável e empática para um usuário leigo. O foco é empoderar o usuário, explicando seus direitos e os próximos passos que ele pode tomar.
  Backstory: Você é um especialista em UX Writing que trabalha em uma organização de defesa do consumidor. Sua paixão é traduzir "juridiquês" para o português claro. Você deve evitar jargões, usar analogias, estruturar a resposta com listas e negrito para facilitar a leitura e sempre adotar um tom tranquilizador e encorajador.

4.1.4 CAMADA DE CONHECIMENTO: A BASE DA VERDADE

A Camada de Conhecimento provê os recursos para a Camada de Cognição. Seus principais componentes são a base de conhecimento vetorial e as ferramentas que os agentes podem invocar.

O processo de criação e uso da ferramenta RAG pode ser dividido em duas fases:

4.1.4.1 FASE 1: INDEXAÇÃO (PROCESSO OFFLINE)

Antes que qualquer consulta possa ser feita, a base de conhecimento precisa ser construída. Este processo, executado apenas uma vez ou sempre que a legislação muda, segue os seguintes passos:

1. **Carregamento dos Documentos:** O texto completo do Código de Defesa do Consumidor é carregado a partir de um arquivo de texto.

2. **Divisão em Trechos (Chunking):** O documento é quebrado em trechos menores e sobrepostos (chunks). Essa etapa é crucial. Se os trechos forem muito grandes, a busca perde especificidade; se forem muito pequenos, o contexto pode ser perdido. Um tamanho de chunk de 500 a 1000 caracteres, com uma sobreposição de 100 a 200 caracteres, é um ponto de partida comum para garantir que as sentenças não sejam cortadas ao meio.

3. **Geração de Embeddings:** Cada chunk de texto é então passado por um modelo de embeddings (como os da OpenAI ou modelos open-source), que converte o texto em um vetor numérico.

4. **Armazenamento no Índice Vetorial:** Os vetores resultantes são armazenados em um índice do **FAISS (Facebook AI Similarity Search)**. O FAISS é uma biblioteca altamente otimizada para busca de similaridade em grandes conjuntos de vetores, permitindo encontrar os vizinhos mais próximos de um dado vetor de forma extremamente rápida.

4.1.4.2 FASE 2: RECUPERAÇÃO (PROCESSO ONLINE)

Quando um usuário faz uma pergunta, o seguinte fluxo acontece em tempo real:

1. **Vetorização da Consulta:** A pergunta do usuário passa pelo mesmo modelo de embeddings usado na indexação, gerando um vetor de consulta.

2. **Busca por Similaridade:** O FAISS é usado para comparar o vetor da consulta com todos os vetores no índice, retornando os 'k' trechos (chunks) cujos vetores são mais próximos (ou seja, mais semanticamente relevantes) à pergunta.

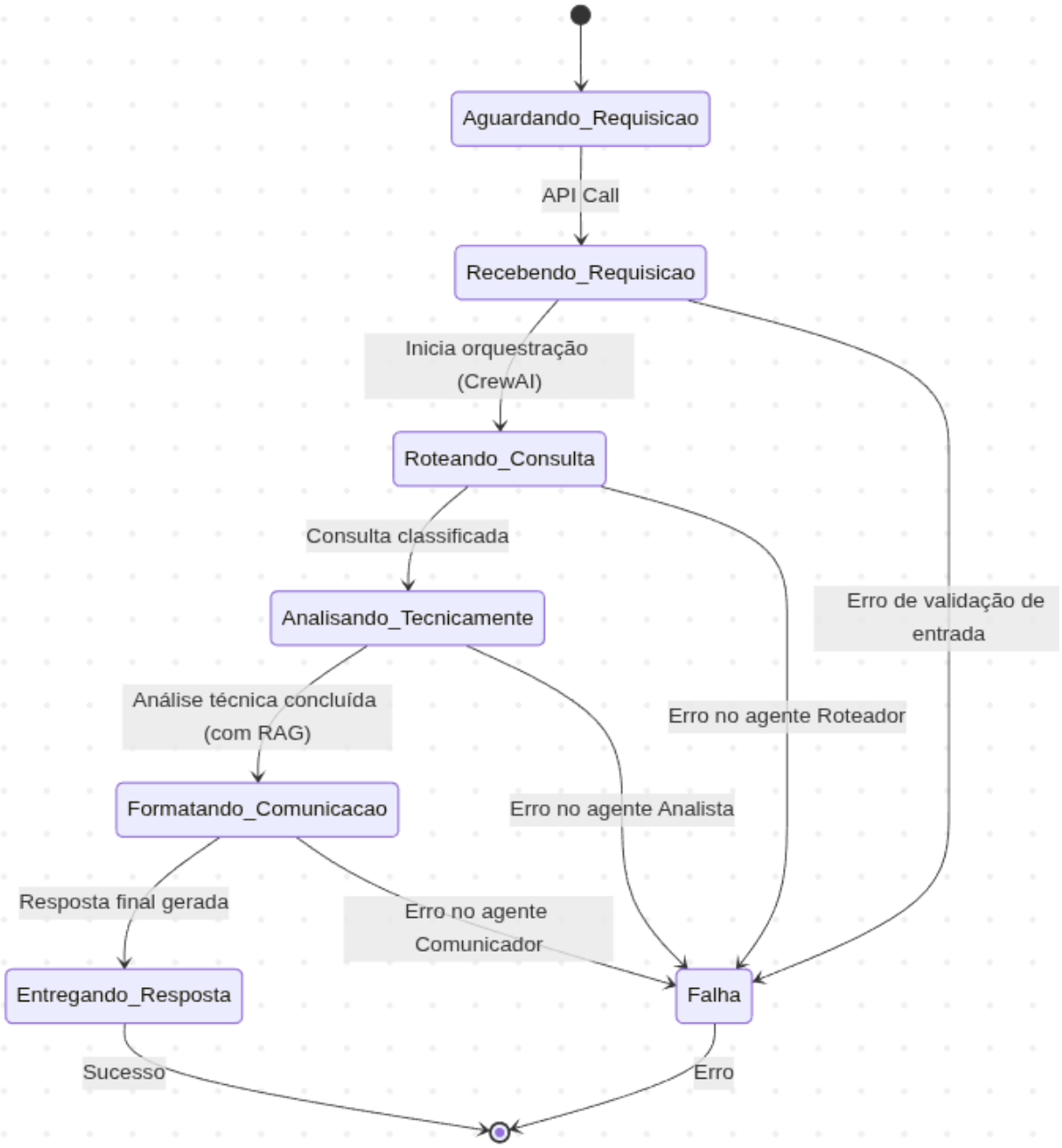
3. **Construção do Contexto:** Os textos desses 'k' trechos recuperados são concatenados para formar o contexto que será injetado no prompt do Agente Analista Jurídico.

Este processo de duas fases garante que o agente tenha acesso rápido e preciso ao conhecimento relevante para responder à consulta do usuário de forma fundamentada.

4.1.5 FLUXO DE EXECUÇÃO DA REQUISIÇÃO (DIAGRAMA DE ESTADOS)

Se a arquitetura em camadas descreve a estrutura estática do sistema, o diagrama de estados a seguir ilustra seu comportamento dinâmico. Ele modela o ciclo de vida de uma única consulta de usuário, desde o momento em que é recebida pela API até a entrega da resposta final, detalhando a transição entre as responsabilidades dos agentes.

Figure 8 – Diagrama de Estados do processamento de uma consulta.



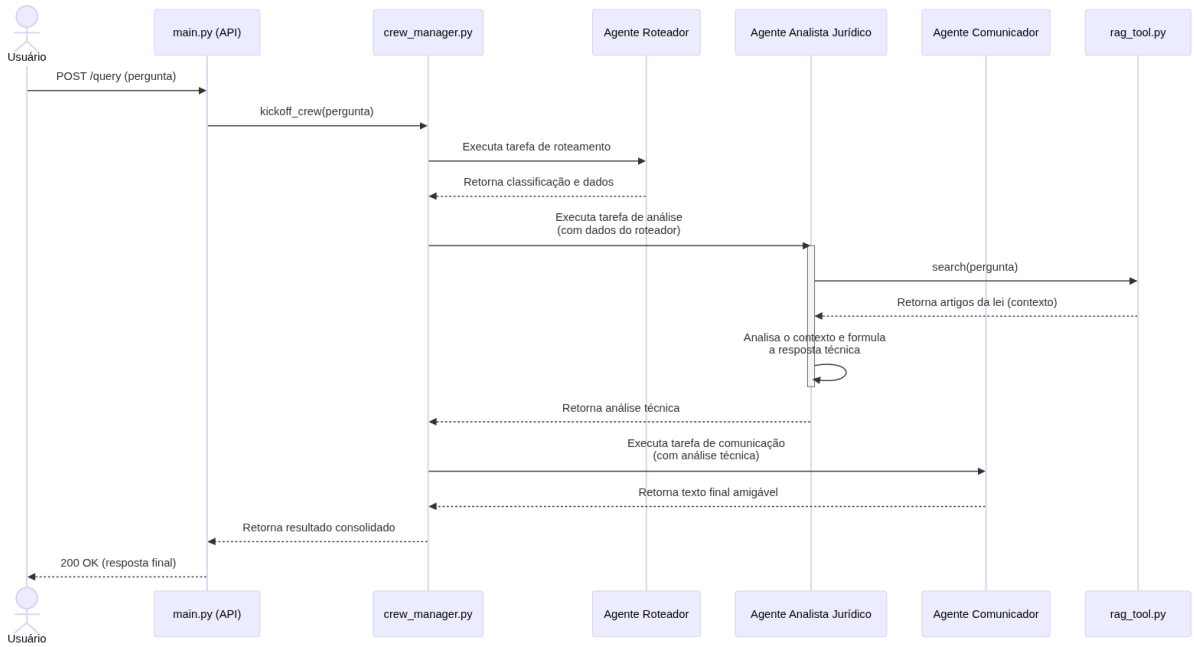
Fonte: Elaborado pelo autor (2025).

O diagrama evidencia o fluxo principal de sucesso, onde a requisição passa sequencialmente pelos agentes especializados. Além disso, ele contempla os estados de falha, que podem ocorrer em qualquer etapa do processo, desde uma validação de entrada mal-sucedida na camada de apresentação até um erro inesperado durante a execução de um dos agentes na camada de cognição.

4.2 COMPONENTES DO SISTEMA E MAPEAMENTO ARQUITETURAL

O fluxo de uma requisição através do sistema ilustra a interação entre os componentes de software e as camadas lógicas. O diagrama de sequência a seguir (Diagrama 8) detalha essa interação, desde a chegada da requisição até a devolução da resposta final.

Figure 9 – Diagrama de Sequência do fluxo de uma requisição no sistema.



Fonte: Elaborado pelo autor (2025).

Quando um usuário envia uma pergunta, o `main.py` (Camada de Apresentação) a recebe e invoca o `crew_manager.py` (Camada de Orquestração). Este, por sua vez, instancia a equipe de agentes (`core/agents/*.py`) e lhes atribui uma sequência de tarefas, conforme o fluxo acima. Durante a execução, um agente como o Analista Jurídico (Camada de Cognição) utiliza a ferramenta `rag_tool.py` (Camada de Conhecimento) para buscar artigos relevantes no Código de Defesa do Consumidor (`knowledge_base/`). O resultado final é então consolidado e retornado ao usuário.

Os componentes de software do protótipo foram mapeados para as camadas lógicas, conforme a tabela abaixo:

Table 1 – Descrição dos componentes e camadas da arquitetura

Componente	**Camada**	**Descrição**
`main.py`	Apresentação	Expõe a API FastAPI e gerencia as requisições de entrada.
`core/crew_manager.py`	Orquestração	Define a "tripulação" de agentes e a sequência de tarefas.
`core/agents/*.py`	Cognição	Implementação individual de cada agente especializado.
`core/tools/*.py`	Conhecimento	Ferramentas específicas que os agentes podem usar.
`core/rag_tool.py`	Conhecimento	Implementação da ferramenta de busca na base de conhecimento.
`knowledge_base/`	Conhecimento	Repositório dos documentos legais (ex: CDC).

Fonte: Elaborado pelo autor (2025).

4.3 FRAMEWORK DE AVALIAÇÃO: MÉTRICAS E EXEMPLOS PRÁTICOS

Para validar rigorosamente a eficácia do sistema para além da prova de conceito, um framework de avaliação que combina métricas de IA, indicadores de desempenho de TI e análise qualitativa é essencial. A avaliação objetiva é o que permite a transição de um protótipo para um serviço confiável.

4.3.1 MÉTRICAS DE QUALIDADE DA IA (BASEADAS EM GOLDEN DATASET)

Um Golden Dataset é um conjunto de dados de teste de alta qualidade, criado manualmente por especialistas, que serve como "gabarito" para avaliar o desempenho do sistema. Ele contém triplas de pergunta, contexto_ideal (o trecho da lei que deveria ser encontrado) e resposta_ideal. Com base nele, medimos métricas de ponta a ponta que avaliam a cadeia de RAG. A seguir, detalhamos cada métrica com exemplos práticos:

4.3.1.1 FAITHFULNESS (FIDELIDADE)

A resposta gerada se atém estritamente ao contexto da lei recuperado? Esta métrica penaliza "alucinações".

- **Pergunta do Usuário:** "Qual o prazo para o fornecedor consertar meu produto com defeito?"

- **Contexto Recuperado (Art. 18 do CDC):** "...não sendo o vício sanado no prazo máximo de trinta dias, pode o consumidor exigir, alternativamente e à sua escolha: a substituição do produto..."
- **Resposta com Alta Fidelidade:** "De acordo com o Artigo 18 do CDC, o fornecedor tem o prazo máximo de **30 dias** para consertar o produto."
- **Resposta com Baixa Fidelidade (Alucinação):** "O fornecedor tem *geralmente um mês, mas pode ser estendido para 45 dias em alguns casos*, para consertar o produto." (A informação de "45 dias" é uma alucinação, pois não estava no contexto recuperado).

4.3.1.2 ANSWER RELEVANCY (RELEVÂNCIA DA RESPOSTA)

A resposta, mesmo que fiel, responde diretamente à pergunta do usuário?

- **Pergunta do Usuário:** "Comprei um presente de Natal que veio com defeito. A loja precisa trocar imediatamente?"
- **Contexto Recuperado (Art. 18 do CDC):** (O mesmo do exemplo anterior, sobre o prazo de 30 dias para conserto).
- **Resposta Relevante:** "Entendo a urgência por ser um presente. No entanto, a lei (Art. 18 do CDC) geralmente dá ao fornecedor um prazo de até 30 dias para consertar o produto. A troca imediata só é obrigatória se o produto for essencial ou se o conserto comprometer sua qualidade."
- **Resposta Irrelevante (mas Fiel):** "O fornecedor tem o prazo máximo de 30 dias para consertar o produto." (A resposta é factualmente correta, mas não aborda a nuance da pergunta do usuário sobre a troca imediata de um presente).

4.3.1.3 CONTEXT PRECISION (PRECISÃO DO CONTEXTO)

O sistema recuperou os trechos de lei corretos e relevantes para a pergunta? Esta métrica avalia a eficácia do componente de busca (o "R" de RAG).

- **Pergunta do Usuário:** "Recebi uma cobrança em meu cartão de crédito por um serviço que cancelei. O que fazer?"
- **Contexto com Alta Precisão:** Recupera o Art. 42, Parágrafo único, do CDC, que trata da devolução em dobro de valores cobrados indevidamente.
- **Contexto com Baixa Precisão:** Recupera o Art. 49 do CDC, que trata do direito de arrependimento em compras online. (Embora seja um artigo do CDC, ele não tem relação com o problema de cobrança indevida).

4.3.2 AVALIAÇÃO QUALITATIVA DA EXPERIÊNCIA DO USUÁRIO

Além das métricas quantitativas, a avaliação da experiência do usuário é fundamental para medir o sucesso de um sistema centrado no ser humano. Uma abordagem padrão para isso é o **System Usability Scale (SUS)**, um questionário de 10 itens que fornece uma medida da usabilidade percebida do sistema (BROOKE, 1996).

A aplicação do SUS envolveria recrutar um grupo de usuários representativos do público-alvo, pedir que eles realizem tarefas específicas com o protótipo (ex: "Descubra seus direitos se um produto que você comprou online atrasar a entrega") e, em seguida, solicitar que respondam ao questionário SUS. O resultado seria uma pontuação de 0 a 100, oferecendo um benchmark claro da usabilidade do sistema e identificando pontos de atrito na interação que métricas de IA sozinhas não conseguem capturar.

4.3.3 AUDITORIA E EXPLICABILIDADE NA PRÁTICA: O LOG DE RASTREABILIDA

A arquitetura multiagente, além de seus benefícios de modularidade, é a chave para a implementação de um sistema auditável. A capacidade de rastrear o processo de decisão de uma IA é um dos objetivos centrais da XAI, pois transforma um modelo "caixa-preta" em um processo "caixa-de-vidro" (glass-box) (ARRIETA, 2020). No protótipo, isso é materializado através da geração de um **log de rastreabilidade** para cada consulta.

Frameworks como o CrewAI, ao executarem as tarefas de forma sequencial, permitem que as entradas e saídas de cada agente sejam capturadas. Um log de auditoria para uma consulta seria estruturado da seguinte forma:

```
--- INÍCIO DA AUDITORIA: ID da Requisição: 12345-ABCDE ---

[ETAPA 1: ROTEAMENTO]
- Agente: Router Agent
- Entrada (Pergunta do Usuário): "Comprei um celular pela internet e me arrependi.
  A loja pode se negar a aceitar a devolução porque eu já abri a caixa?"
- Saída (Decisão do Roteador): "Advogado do Consumidor"

[ETAPA 2: ANÁLISE TÉCNICA]
- Agente: Advogado do Consumidor
- Entrada (Contexto da Etapa 1): Pergunta do usuário.
- Ação: Uso da Ferramenta RAG com a consulta.
- Dados da Ferramenta (Contexto Recuperado):
  "Art. 49. O consumidor pode desistir do contrato, no prazo de 7 dias a contar
  de sua assinatura ou do ato de recebimento do produto ou serviço, sempre que a
  contratação de fornecimento de produtos e serviços ocorrer fora do
  estabelecimento comercial, especialmente por telefone ou a domicílio."
```

```
- Saída (Análise Técnica Gerada): "Com base no Art. 49 do CDC, o consumidor
  tem o direito de arrependimento em compras online no prazo de 7 dias. O fato
  de abrir a caixa não invalida esse direito, pois é necessário para avaliar o
  produto. O fornecedor deve aceitar a devolução e restituir os valores pagos."

[ETAPA 3: COMUNICAÇÃO]
- Agente: Communication Agent
- Entrada (Contexto da Etapa 2): A análise técnica do agente anterior.
- Saída (Resposta Final ao Usuário): "Olá! Entendo perfeitamente a sua dúvida...
  Você tem o direito de se arrepender e devolver o produto, mesmo que tenha
  aberto a caixa. Este é o chamado 'Direito de Arrependimento', garantido pelo
  Artigo 49 do Código de Defesa do Consumidor..."

--- FIM DA AUDITORIA ---
```

Este log detalhado é uma ferramenta de gestão de valor inestimável:

- **Depuração de Erros (Debugging):** Se a resposta final estiver incorreta, o log permite identificar exatamente onde a falha ocorreu. Foi um erro de roteamento? A ferramenta RAG recuperou o artigo errado? Ou o Agente Comunicador interpretou mal a análise técnica?
- **Otimização de Performance:** A análise dos logs pode revelar gargalos. Por exemplo, se o Agente Analista consistentemente leva muito tempo, pode ser um indicativo de que a ferramenta RAG precisa de otimização ou que o prompt do agente precisa ser mais direto.
- **Governança e Conformidade:** Em um ambiente corporativo, ter um registro auditável de por que uma determinada orientação foi fornecida a um cliente é fundamental para a gestão de riscos e para demonstrar a conformidade com as políticas internas e regulamentações externas.

4.3.4 INDICADORES CHAVE DE DESEMPENHO (KPIS) DE TI E OPERAÇÃO

Enquanto as métricas de IA avaliam a *qualidade* da resposta, os Indicadores Chave de Desempenho (KPIs) avaliam a *saúde e a eficiência* do serviço do ponto de vista da Gestão de TI. Eles são cruciais para garantir que o sistema seja sustentável, escalável e ofereça uma boa experiência ao usuário. Os seguintes KPIs são propostos:

Table 2 – Framework de Avaliação: Métricas e KPIs

Categoria KPI	**KPI Específico**	**Definição/Propósito**	**Frequência de Medição**
`Engajamento do Usuário`	Usuários Únicos Mensais	Número de usuários distintos que interagem com o bot. Mede o alcance e a adoção.	Mensal
	Tempo Médio de Interação	Duração média das conversas do usuário com o bot. Pode indicar a complexidade das questões ou o nível de engajamento.	Semanal
`Desempenho de TI`	Tempo de Resposta do Bot	Latência entre a pergunta e a resposta do bot. Crítico para a experiência do usuário.	Diário
	Disponibilidade (Uptime)	Percentual de tempo em que o serviço está operacional. Mede a confiabilidade da infraestrutura.	Mensal
	Taxa de Erro da IA	Frequência de falhas na compreensão da intenção do usuário ou erros na execução das tarefas dos agentes.	Semanal
`Sustentabilidade`	Custo por Interação (CPI)	Custo médio de cada interação (custos de API de LLM, infraestrutura). Essencial para a viabilidade financeira do projeto.	Mensal

Fonte: Elaborado pelo autor (2025).

4.3.5 VALOR ESTRATÉGICO DOS DADOS PARA A GESTÃO

É crucial entender que o framework de avaliação não serve apenas para validar a qualidade técnica do protótipo; ele transforma o sistema em uma ferramenta de inteligência de negócio para a tomada de decisão gerencial. Um gestor não está apenas implementando um "chatbot", mas sim um mecanismo de captura de insights valiosos diretamente do seu público-alvo. As interações dos usuários, agregadas e anonimizadas, geram um painel de controle estratégico que responde a perguntas cruciais para a gestão do negócio:

- Inteligência de Produto e Serviço: Quais são as dúvidas e reclamações mais frequentes dos consumidores? A análise desses dados pode revelar falhas de design em um produto, problemas na comunicação de uma oferta ou lacunas no serviço de pós-venda. A decisão de priorizar uma melhoria no produto X ou reescrever o manual do serviço Y pode ser diretamente informada por esses insights.

- Otimização Operacional: Em quais tópicos a IA demonstra maior dificuldade (baixa Answer Relevancy)? Isso não aponta apenas uma deficiência da IA, mas pode indicar áreas onde a própria documentação interna da empresa é confusa ou incompleta. A decisão de investir em treinamento para a equipe de suporte ou melhorar a base de conhecimento interna é uma consequência direta.

- Planejamento Financeiro e de TI: Qual o Custo por Interação e como ele se correlaciona com a complexidade da pergunta? A análise desses KPIs permite ao gestor de TI um planejamento orçamentário preciso, a otimização de custos de infraestrutura e a negociação de contratos com provedores de API de LLMs.

Dessa forma, o sistema proposto deixa de ser um mero canal de atendimento para se tornar um ativo estratégico. Ele gera um fluxo contínuo de dados que capacita o gestor a otimizar a operação, justificar investimentos e direcionar a evolução do negócio com base em evidências, não em suposições. A solução, portanto, é um motor para a tomada de decisões de gestão.

5 CONCLUSÕES

Este trabalho se propôs a enfrentar o complexo desafio da assimetria de informação no campo jurídico brasileiro por meio da tecnologia. A questão central não era apenas fornecer respostas, mas garantir que fossem, simultaneamente, tecnicamente precisas, fundamentadas em fontes confiáveis e comunicadas de forma a empoderar o cidadão leigo. A seguir, são apresentadas as conclusões sobre como a arquitetura proposta e o protótipo desenvolvido responderam a esse desafio.

5.1 EM RELAÇÃO AO OBJETIVO GERAL

O objetivo geral de projetar, desenvolver e documentar uma arquitetura de software funcional para democratizar a informação jurídica foi plenamente alcançado. A principal contribuição deste trabalho reside na demonstração de que a arquitetura multiagente não é apenas uma solução técnica elegante, mas uma decisão de gestão estratégica (WOOLDRIDGE, 2009). A especialização de tarefas, implementada através dos agentes, provou ser uma abordagem robusta para construir uma solução escalável e de manutenção simplificada. Mais importante, a arquitetura inerentemente explicável (XAI) do sistema, que oferece transparência tanto da fonte (via RAG) quanto do processo (via pipeline de agentes), estabelece um pilar para a confiança, governança e auditabilidade da solução, respondendo diretamente aos desafios de se criar uma solução de IA sustentável para um problema de negócio complexo (ARRIETA, 2020).

O protótipo funcional, focado no Direito do Consumidor, serviu como uma prova de conceito conclusiva, validando que esta abordagem é viável e que o sistema consegue entregar um resultado final coeso e de alta qualidade, superando os desafios de confiabilidade e complexidade delineados no problema de pesquisa.

5.2 EM RELAÇÃO AOS OBJETIVOS ESPECÍFICOS

O sucesso no alcance do objetivo geral foi sustentado pelo cumprimento de cada um dos cinco objetivos específicos:

1. Modelagem da arquitetura em camadas: A arquitetura foi efetivamente modelada em quatro camadas (Apresentação, Orquestração, Cognição e Conhecimento), conforme detalhado no Capítulo 4. Essa separação de responsabilidades provou ser fundamental para a organização, manutenibilidade e escalabilidade do código.

2. Implementação do sistema multiagente: O sistema foi implementado com sucesso utilizando o framework CrewAI. A criação de agentes distintos com papéis, objetivos e ferramentas próprias (Analista Jurídico, Comunicador) materializou a estratégia de "dividir para conquistar", sendo o principal diferencial da solução.

3. Integração da técnica RAG: A técnica de Geração Aumentada por Recuperação foi o pilar para garantir a confiabilidade das respostas. Ao forçar o Analista Jurídico a basear sua análise em trechos recuperados do Código de Defesa do Consumidor, o risco de "alucinações" da IA foi mitigado, tornando o sistema uma fonte de informação mais segura e transparente (XAI).

4. Desenvolvimento do protótipo funcional: O protótipo foi desenvolvido e validado, processando consultas sobre o Direito do Consumidor de ponta a ponta. Ele não apenas provou a viabilidade técnica da arquitetura, mas também serviu como um artefato tangível para a análise e documentação apresentadas neste trabalho.

5. Alinhamento com os ODS: O projeto foi conscientemente alinhado com os Objetivos de Desenvolvimento Sustentável 4, 10, 12 e 16. Esta conexão não foi apenas teórica, mas se manifesta na própria missão do sistema: educar cidadãos, reduzir desigualdades no acesso à justiça, promover o consumo responsável e fortalecer instituições, mesmo que de forma indireta.

5.3 LIMITAÇÕES DO ESTUDO

Apesar dos resultados positivos, é importante reconhecer as limitações inerentes a este trabalho. Primeiramente, o protótipo foi desenvolvido e testado em um escopo limitado, focado exclusivamente no Código de Defesa do Consumidor. A generalização de sua eficácia para outras áreas do Direito, embora arquiteturalmente possível, exigiria a curadoria de novas bases de conhecimento e a adaptação dos agentes especialistas.

Em segundo lugar, a avaliação da qualidade das respostas, embora baseada em um framework robusto proposto no Capítulo 4, ainda não foi submetida a um estudo em larga escala com usuários finais. A percepção de clareza, empatia e utilidade é subjetiva e requer validação qualitativa formal. Por fim, a dependência de APIs de LLMs de terceiros implica custos operacionais e uma certa perda de controle sobre a latência e a disponibilidade do modelo subjacente, fatores que seriam críticos em uma implantação em produção (SARTOR, 2021).

5.4 TRABALHOS FUTUROS

As conclusões e limitações deste estudo abrem um leque de oportunidades para a continuidade e evolução do projeto. A seguir, um roteiro detalhado para trabalhos futuros

é proposto, organizado em frentes de curto, médio e longo prazo.

5.4.1 CURTO PRAZO: VALIDAÇÃO E EXPANSÃO DO PROTÓTIPO

- **Implementação do Framework de Avaliação Quantitativa:** O primeiro passo é implementar o framework de avaliação descrito no Capítulo 4. Isso envolve a criação de um "Golden Dataset" robusto, com centenas de pares de perguntas e respostas ideais validadas por especialistas. Em seguida, um script de avaliação automatizado deve ser desenvolvido para calcular as métricas de Faithfulness, Answer Relevancy e Context Precision. Este processo transformará a avaliação de um exercício manual em um pipeline de integração contínua, permitindo que a qualidade do sistema seja monitorada a cada nova alteração.

- **Estudo de Usabilidade com Usuários Finais:** É crucial realizar um estudo formal de usabilidade com um grupo de usuários reais e leigos. Utilizando o método System Usability Scale (SUS), seria possível obter dados quantitativos sobre a percepção de facilidade de uso. Além disso, sessões de "think-aloud", onde os usuários verbalizam seus pensamentos enquanto interagem com o sistema, forneceriam insights qualitativos valiosos para identificar pontos de atrito e refinar a persona do Agente Comunicador, garantindo que a empatia e a clareza sejam eficazes na prática.

- **Desenvolvimento de uma Biblioteca de Recursos Acionáveis:** Para além da informação, o empoderamento do usuário vem da ação. Uma evolução natural do sistema seria a criação de uma ferramenta que, com base na orientação fornecida, gerasse documentos práticos para o usuário. Por exemplo, após orientar sobre o direito de arrerependimento, o sistema poderia oferecer a geração de um modelo de e-mail de notificação para a loja, já preenchido com os dados relevantes e citando o artigo de lei apropriado.

- **Implementação de um Dashboard de KPIs:** Os KPIs de TI e operação propostos devem ser implementados em um painel de controle (dashboard) em tempo real. Ferramentas como o Google Data Studio ou o Power BI poderiam ser conectadas aos logs do sistema para visualizar métricas como o tempo médio de resposta, a taxa de erro dos agentes e o custo por interação. Isso daria ao gestor de TI uma visão clara da saúde operacional e financeira do serviço.

5.4.2 MÉDIO PRAZO: EXPANSÃO DO DOMÍNIO E DA INTELIGÊNCIA

- **Expansão da Base de Conhecimento para Novas Áreas do Direito:** A modularidade da arquitetura multiagente foi projetada para facilitar a expansão. O próximo passo seria adicionar novas bases de conhecimento, como a Consolidação das Leis do Trabalho (CLT) ou partes do Código Civil. Para cada nova base, um novo

agente especialista (ex: "Advogado Trabalhista") seria criado, e o Agente Roteador seria treinado para identificar e direcionar as perguntas para o novo especialista, validando a escalabilidade da solução.

- **Desenvolvimento de um Agente de Análise de Documentos:** Uma funcionalidade de alto valor seria a criação de um novo tipo de agente capaz de analisar documentos enviados pelo usuário. Este "Agente Analista de Documentos" poderia receber o upload de um contrato de aluguel, uma fatura de cartão de crédito ou uma apólice de seguro e, utilizando técnicas de extração de informação, identificar cláusulas potencialmente abusivas, cobranças indevidas ou inconsistências, fornecendo uma análise preliminar e direcionada.

5.4.3 LONGO PRAZO: RUMO À PRODUÇÃO E À INTELIGÊNCIA ATIVA

- **Migração da Infraestrutura para um Ambiente de Produção:** A stack de tecnologia do MVP, focada em agilidade, precisaria ser migrada para uma infraestrutura robusta e escalável. Isso envolveria substituir o Make.com e o Google Sheets por um sistema de backend mais tradicional, com um banco de dados como o PostgreSQL ou Firestore para armazenar logs e dados de usuário, e um sistema de filas (como o RabbitMQ) para gerenciar as requisições de forma assíncrona, garantindo alta disponibilidade e resiliência.

- **Otimização de Custos com LLMs Open-Source:** A dependência de APIs pagas de LLMs é uma limitação estratégica. Uma linha de pesquisa futura seria explorar a viabilidade de substituir os modelos proprietários por modelos de linguagem open-source (como o Llama 3 ou o Mixtral) hospedados em infraestrutura própria. Isso exigiria um investimento inicial em hardware (GPUs), mas poderia reduzir drasticamente o custo por interação no longo prazo, tornando o serviço mais sustentável financeiramente.

- **Desenvolvimento de um "Agente Proativo":** A evolução final do sistema seria a transição de um modelo puramente reativo para um modelo proativo. Um "Agente Proativo" poderia monitorar, com a permissão do usuário, informações relevantes (como novas leis de proteção de dados ou recalls de produtos) e notificá-lo ativamente sobre direitos ou riscos que ele possa não conhecer, transformando o sistema de uma ferramenta de consulta em um verdadeiro assistente pessoal para a cidadania.

Em suma, este trabalho não apenas entregou uma solução funcional, mas também estabeleceu um alicerce sólido e um roteiro claro para futuras pesquisas e desenvolvimentos. A arquitetura de software multiagente, combinada com a confiabilidade da RAG e os princípios do Legal Design, provou ser uma abordagem promissora e de alto impacto para

utilizar a Inteligência Artificial como uma força para a democratização do acesso à justiça no Brasil.

REFERÊNCIAS

ARRIETA, A. B. e. a. *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges*. 2020. Information Fusion, 58, 82-115.

BROOKE, J. *SUS: A "'quick and dirty"' usability scale*. 1996. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, A. L. McClelland (Eds.), Usability Evaluation in Industry. Taylor & Francis.

Defense Advanced Research Projects Agency (DARPA). *Explainable Artificial Intelligence (XAI)*. 2016. Disponível em: <https://www.darpa.mil/program/explainable-artificial-intelligence>. Acesso em: DD mês. AAAA.

ECKERSON, W. W. *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*. 2nd. ed. [S.l.]: Wiley, 2010.

FERNANDES, A. A.; ABREU, V. F. *Implantando a Governança de TI: Da Estratégia à Gestão dos Processos e Serviços*. 4ª. ed. Rio de Janeiro: Brasport, 2014.

FOWLER, M. *Patterns of Enterprise Application Architecture*. [S.l.]: Addison-Wesley, 2002.

GIL, A. C. *Métodos e técnicas de pesquisa social*. 6. ed. São Paulo: Atlas, 2008.

HAGAN, M. *Law by Design*. Stanford, CA: Margaret Hagan, 2022.

JOHNSON, J.; DOUZE, M.; JéGOU, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, v. 7, n. 3, p. 535–47, 2019.

LEWIS, P. e. a. Retrieval-augmented generation for knowledge-intensive nlp tasks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2020. v. 33.

LIU, P. e. a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, v. 55, n. 9, p. 1–35, 2023.

MIKOLOV, T. e. a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

MINISTÉRIO DA JUSTIÇA E SEGURANÇA PÚBLICA. *Boletim Consumidor em Números 2023*. 2024. Secretaria Nacional do Consumidor.

NAÇÕES UNIDAS. *Transformando Nosso Mundo: A Agenda 2030 para o Desenvolvimento Sustentável*. 2015. Resolução A/RES/70/1.

NEWMAN, S. *Building Microservices: Designing Fine-Grained Systems*. [S.l.]: O'Reilly Media, 2015.

RAO, A. S.; GEORGEFF, M. P. Bdi agents: From theory to practice. In: *Proceedings of the First International Conference on Multi-Agent Systems*. São Francisco, CA: The MIT Press, 1995. p. 312–319.

RUSSELL, S. J.; NORVIG, P. *Inteligência Artificial: Uma Abordagem Moderna*. 4ª. ed. Rio de Janeiro: Gen LTC, 2021.

SARTOR, G. Artificial intelligence in law: An overview. In: *The Cambridge Handbook of Artificial Intelligence*. Cambridge: Cambridge University Press, 2021. p. 393–419.

SOMMERVILLE, I. *Software Engineering*. 9th. ed. [S.l.]: Addison-Wesley, 2011.

TURBAN, E. e. a. *Business Intelligence: A Managerial Approach*. 2nd. ed. [S.l.]: Prentice Hall, 2011.

VASWANI, A. e. a. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. v. 30.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. 2ª. ed. Chichester: John Wiley & Sons, 2009.

APÊNDICE A – CÓDIGO-FONTE DO ORQUESTRADOR

O código a seguir, extraído do arquivo `core/crew_manager.py`, ilustra como a "tripulação" de agentes é definida, como as tarefas são encadeadas e como o processo é iniciado. Este componente pertence à **Camada de Orquestração** e atua como o "maestro" que rege a colaboração entre os agentes.

```
"""
Gerencia a criação, configuração e execução da tripulação (Crew) de agentes da Crew

Este módulo importa todos os agentes, define a sequência de tarefas de 3 etapas
(Roteamento -> Análise Técnica -> Comunicação) e executa a tripulação.
"""

from crewai import Task, Crew, Process
from core.logger import logger

# --- Importação de todos os Agentes Modulares ---
from core.agents.router_agent import router_agent
from core.agents.advogado_consumidor import advogado_consumidor_agent
from core.agents.criminal_agent import criminal_agent
from core.agents.civil_agent import civil_agent
from core.agents.communication_agent import communication_agent # Novo agente!

# --- Definição das Tarefas em 3 Etapas ---

def create_crew_tasks(user_query: str) -> list[Task]:
    """Cria as tarefas para a tripulação com base na consulta do usuário."""

    # Tarefa 1: Roteamento
    # O router_agent analisa a consulta e decide qual especialista é o mais adequad
    routing_task = Task(
        description=f"Analise a seguinte consulta do usuário e decida qual especial
        expected_output="O nome do cargo (role) do agente especialista mais adequad
        agent=router_agent
    )
```

```
    # Tarefa 2: Análise Técnica
    # O agente especialista escolhido (via contexto) executa a análise aprofundada.
    # A resposta desta tarefa será a matéria-prima para a próxima etapa.
    technical_analysis_task = Task(
        description=f"Com base na consulta do usuário, forneça uma análise técnica
        expected_output="A análise técnica e detalhada, incluindo referências legai
        context=[routing_task], # O resultado do roteamento determina o agente
    )

    # Tarefa 3: Comunicação com o Usuário
    # O communication_agent "traduz" a análise técnica para uma linguagem simples.
    communication_task = Task(
        description="Reescreva a análise técnica a seguir em uma linguagem simples,
        expected_output="A resposta final, formatada e pronta para ser entregue ao
        agent=communication_agent,
        context=[technical_analysis_task] # Usa a saída da análise técnica como ent
    )

    return [routing_task, technical_analysis_task, communication_task]

# --- Montagem e Execução da Tripulação ---

def run_crew(user_query: str) -> str:
    """
    Monta e executa a tripulação para processar a consulta do usuário.
    """
    logger.info(f"Iniciando a execução da tripulação para a consulta: {user_query}"

    tasks = create_crew_tasks(user_query)

    # A lista de agentes inclui todos os nossos especialistas. O CrewAI selecionará
    # dinamicamente os agentes corretos para cada tarefa com base no contexto.
    crew = Crew(
        agents=[
            router_agent,
            advogado_consumidor_agent,
            criminal_agent,
            civil_agent,
```

```
            communication_agent
        ],
        tasks=tasks,
        process=Process.sequential, # As 3 tarefas são executadas em sequência
        verbose=2
    )

    result = crew.kickoff()

    logger.info(f"Execução da tripulação concluída. Resultado: {result}")
    return result
```

B CÓDIGO-FONTE DO AGENTE ANALISTA

O código a seguir, extraído do arquivo `core/agents/advogado_consumidor_agent.py`, ilustra a implementação de um agente especialista. Este componente pertence à **Camada de Cognição** e é responsável pela análise técnica da consulta do usuário, utilizando a ferramenta RAG para garantir a confiabilidade.

Note como o *prompt* (composto por 'role', 'goal' e 'backstory') é detalhado e específico, instruindo o LLM a atuar como um especialista meticuloso e a basear suas respostas estritamente nas fontes recuperadas.

```
"""
Implementação do Agente Especialista em Direito do Consumidor.
"""

from crewai import Agent
from core.tools.rag_tool import rag_tool

# Criação do Agente Advogado do Consumidor
advogado_consumidor_agent = Agent(
    role="Advogado Especialista em Direito do Consumidor",
    goal="""
        Fornecer uma análise jurídica precisa e detalhada sobre a pergunta
        do usuário, baseando-se estritamente nas informações recuperadas
        de sua ferramenta de busca na legislação (RAG).
    """,
    backstory="""
        Você é um advogado experiente e meticuloso, com profundo conhecimento
        do Código de Defesa do Consumidor (CDC). Sua principal preocupação é a
        precisão e a confiabilidade da informação. Você NUNCA responde com
        base em seu conhecimento geral pré-treinado; você sempre cita a fonte
        legal (artigo, parágrafo, etc.) para cada afirmação que faz.
        Sua resposta deve ser técnica, completa e estruturada, servindo como
        um parecer detalhado que será usado por outro profissional para
        comunicação com o cliente. Você deve extrair e apresentar todos os
        detalhes relevantes dos trechos da lei recuperados.
    """,
```

```
    tools=[rag_tool],
    allow_delegation=False,
    verbose=True,
    memory=False
)
```

C URL DO REPOSITÓRIO NO GITHUB

LINK DE ACESSO

O código-fonte e todos os artefatos relacionados a este trabalho estão disponíveis no seguinte repositório GitHub:

<https://github.com/MarcelDevBr/open-jus-architecture>

ANEXO A – EXEMPLOS DE INTERAÇÃO COM O SISTEMA

Esta seção ilustra o resultado prático do trabalho, apresentando interações completas com o protótipo. Os exemplos demonstram como a pipeline de agentes (Roteamento, Análise e Comunicação) processa diferentes tipos de perguntas e as transforma em orientações claras e acionáveis.

A.1 EXEMPLO 1: DIREITO DO CONSUMIDOR (PERGUNTA DIRETA)

Este exemplo demonstra o fluxo padrão para uma pergunta clara dentro do domínio inicial do protótipo.

PERGUNTA DO USUÁRIO:

> "Comprei um celular pela internet e me arrependi. A loja pode se negar a aceitar a devolução porque eu já abri a caixa?"

RESPOSTA GERADA PELO SISTEMA:

> Olá! Entendo perfeitamente a sua dúvida. É muito comum ficarmos inseguros sobre nossos direitos ao comprar online. Vamos esclarecer isso para você.
>
> **Você tem o direito de se arrepender e devolver o produto, mesmo que tenha aberto a caixa.**
>
> Este é o chamado **"Direito de Arrependimento"**, e ele está garantido pelo **Artigo 49 do Código de Defesa do Consumidor (CDC)**. Esse direito se aplica a todas as compras feitas fora de um estabelecimento comercial físico, como é o caso de compras pela internet, por telefone ou por catálogo.
>
> **O que a lei diz:**
>
> - **Prazo:** Você tem um prazo de **7 dias corridos** para desistir da compra, contados a partir da data em que recebeu o produto.
> - **Justificativa:** Você não precisa dar nenhuma justificativa para a loja. A simples insatisfação ou arrependimento já é motivo suficiente.

- **Caixa Aberta:** O fato de você ter aberto a embalagem para conferir o produto não invalida o seu direito. É seu direito avaliar o produto que recebeu. A loja só poderia recusar a devolução se ficasse comprovado que você usou o produto de forma indevida, causando danos a ele (mau uso).
- **Custos:** Todos os valores que você pagou, incluindo o frete, devem ser devolvidos. Os custos com a devolução do produto também são de responsabilidade da loja.

O que você pode fazer agora:

1. **Entre em contato com a loja** o mais rápido possível, formalmente (por e-mail ou chat), para informar sobre sua decisão de exercer o Direito de Arrependimento.
2. **Mencione o Artigo 49 do Código de Defesa do Consumidor**. Isso mostra que você conhece seus direitos.
3. **Siga as instruções da loja** para a devolução do produto.

Se a loja se recusar a aceitar a devolução, você pode procurar o Procon da sua cidade.

Lembre-se: esta é uma orientação inicial com base nas informações que você forneceu. Ela não substitui uma consulta jurídica com um advogado.

A.2 EXEMPLO 2: ROTEAMENTO PARA OUTRA ÁREA (DIREITO PENAL)

Este exemplo demonstra a capacidade do `router_agent` de identificar o especialista correto quando a pergunta foge do Direito do Consumidor.

PERGUNTA DO USUÁRIO:

"Fui acusado de um crime que não cometi, o que devo fazer? Estou desesperado."

RESPOSTA GERADA PELO SISTEMA:

Olá. Lamento que você esteja passando por uma situação tão difícil e angustiante. É compreensível que se sinta desesperado, mas manter a calma agora é fundamental.

A sua questão é muito séria e pertence à área do Direito Penal. A orientação correta neste momento é crucial.

Ação Imediata e Mais Importante:

- **Procure um Advogado Criminalista:** Você precisa, com a máxima urgência, de um advogado especializado em Direito Criminal. Este profissional é o único que pode analisar os detalhes do seu caso, garantir que seus direitos sejam respeitados e traçar a melhor estratégia de defesa.
- **Não fale sobre o caso:** Evite discutir os detalhes da acusação com qualquer pessoa que não seja seu advogado. Tudo o que você disser pode ser usado no processo.
- **Defensoria Pública:** Se você não tiver condições de contratar um advogado particular, procure a Defensoria Pública do seu estado. Eles oferecem assistência jurídica gratuita.

Neste momento, o passo mais seguro e correto é buscar assessoria jurídica qualificada imediatamente.

Lembre-se: esta é uma orientação emergencial e não substitui, de forma alguma, a consulta e o acompanhamento por um advogado.

A.3 EXEMPLO 3: PERGUNTA VAGA E RESPOSTA CAUTELOSA

Este exemplo mostra como o sistema reage a uma pergunta vaga, onde não é possível dar uma orientação específica sem mais detalhes, demonstrando um comportamento seguro.

PERGUNTA DO USUÁRIO:

"Tenho um problema com uma empresa. Eles não cumprem o que prometem. Posso processar?"

RESPOSTA GERADA PELO SISTEMA:

Olá! Entendo sua frustração com a empresa. É muito desgastante quando sentimos que nossos direitos não estão sendo respeitados.

Sua pergunta é importante, mas a resposta depende muito dos detalhes específicos do seu caso. Para saber se uma ação judicial (processo) é o melhor caminho, é preciso analisar alguns pontos.

Para te ajudar a entender melhor, considere as seguintes questões:

- **Qual foi a promessa?** O que a empresa prometeu exatamente? Isso estava em um contrato, anúncio ou conversa registrada? Ter provas do que foi combinado é muito importante.
- **O que aconteceu de fato?** Como a empresa descumpriu a promessa? O produto veio com defeito, o serviço não foi entregue, a cobrança foi errada?
- **Você já tentou resolver amigavelmente?** Você já entrou em contato com a empresa para tentar resolver? Guarde os protocolos e e-mails dessas tentativas. Muitas vezes, um acordo pode ser mais rápido e menos custoso que um processo.

Possíveis Caminhos:

1. **Consumidor.gov.br:** É uma plataforma online do governo onde você pode registrar uma reclamação formal. Muitas empresas respondem e resolvem os casos por lá.
2. **Procon:** O órgão de defesa do consumidor da sua cidade pode mediar um acordo com a empresa.
3. **Ação Judicial:** Se nenhuma das alternativas acima funcionar, você pode, sim, iniciar um processo. Para causas de menor valor, você pode procurar o Juizado Especial Cível (Pequenas Causas), onde, dependendo do valor, não é obrigatório ter um advogado.

A melhor pessoa para te dizer qual o caminho ideal é um advogado, que poderá analisar suas provas e a situação completa.

Lembre-se: esta é uma orientação geral para te ajudar a organizar as ideias. Ela não é uma recomendação legal definitiva.