UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

C.L.A.I.T.O.N - CHATBOT LEGAL ARTIFICIAL INTELLIGENCE TECHNOLOGY ON NETWORK

PEDRO HENRIQUE RORATTO

BAURU
2025

PEDRO HENRIQUE RORATTO

C.L.A.I.T.O.N - CHATBOT LEGAL ARTIFICIAL INTELLIGENCE TECHNOLOGY ON NETWORK

Proposta para Trabalho de Conclusão de Curso do Curso de Bacharelado em Sistemas de Informação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Câmpus Bauru.
Orientador: Prof. Dr. Clayton Reginaldo Pereira

BAURU
2025

R787c

Roratto, Pedro Henrique

C.L.A.I.T.O.N : CHATBOT LEGAL ARTIFICIAL INTELLIGENCE TECHNOLOGY ON NETWORK / Pedro Henrique Roratto. -- , 2025

51 p. : tabs., fotos

Trabalho de conclusão de curso (-) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências Farmacêuticas, Araraquara,

Orientador: Clayton Reginaldo Pereira

1. Inteligência artificial. 2. Processamento de linguagem natural (Computação). 3. Agentes inteligentes (Software). 4. Direito penal. 5. Jurisprudência. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Agradecimentos

Agradeço, primeiramente, a Deus pela força e inspiração. Ao meu orientador, Prof. Dr. Clayton Reginaldo Pereira, pela inestimável orientação, colaboração e direcionamento, que foram fundamentais para a concretização deste trabalho. Sua expertise e paciência foram pilares essenciais em cada etapa do projeto. Ao Prof. Dr. Carlo Jose Napolitano, pelos valiosos esclarecimentos e discussões aprofundadas na área jurídica, que enriqueceram significativamente a compreensão e o desenvolvimento do presente estudo. Aos amigos e colegas, pelo companheirismo e pelas trocas de conhecimento que tornaram a jornada acadêmica mais leve e produtiva, bem como possibilitaram a elaboração deste trabalho de conclusão de curso.

Resumo

O presente Trabalho de Conclusão de Curso apresenta o desenvolvimento do C.L.A.I.T.O.N (*Chatbot Legal Artificial Intelligence Technology on Network*), um assistente jurídico voltado ao Direito Penal brasileiro, fundamentado em técnicas de Inteligência Artificial e Processamento de Linguagem Natural. O sistema tem como objetivo oferecer respostas fundamentadas e contextualizadas a consultas jurídicas, integrando consulta semântica, recuperação de documentos e geração de linguagem natural com base em fontes verificáveis. A solução adota a arquitetura de Recuperação Aumentada por Geração, utilizando o LangChain como framework principal. Os documentos jurídicos, compostos por acórdãos do STF, do STJ, do TJSP e pela legislação penal, são processados e indexados no banco vetorial ChromaDB, por meio de *embeddings* da família E5. As respostas são geradas localmente com o Ollama (Llama 3), o que contribui para a privacidade e o desempenho do sistema. O sistema disponibiliza dois canais de acesso. A interface Web, desenvolvida em *Streamlit*, oferece recursos de explicabilidade e visualização de fontes. O bot para WhatsApp, integrado via *Twilio API*, atende ao uso cotidiano com praticidade. As respostas apresentam as fontes originais, com metadados e escores de relevância, promovendo transparência e auditabilidade. Os resultados obtidos demonstram a viabilidade da integração entre recuperação semântica, bancos vetoriais e modelos de linguagem de grande porte no domínio jurídico penal. O C.L.A.I.T.O.N contribui para a modernização da pesquisa e da prática jurídica, evidenciando o potencial da Inteligência Artificial no apoio à análise e à interpretação de normas e precedentes.

Palavras-chave: Inteligência Artificial; Direito Penal; Recuperação Aumentada por Geração; Chatbot Jurídico; ChromaDB; LangChain; Ollama.

Abstract

This undergraduate thesis presents the development of C.L.A.I.T.O.N (*Chatbot Legal Artificial Intelligence Technology on Network*), an intelligent legal assistant focused on Brazilian Criminal Law. The system is based on Artificial Intelligence and Natural Language Processing techniques and aims to provide grounded and contextualized answers to legal inquiries by integrating semantic retrieval, document search, and natural language generation using verifiable sources. The solution follows a Retrieval Augmented Generation architecture and uses the LangChain framework. Legal documents, including decisions from the Brazilian Supreme Court and the Superior Court of Justice and the Brazilian Penal Code, are processed and indexed in the ChromaDB vector database using E5 sentence-transformer embeddings. Answers are generated locally with Ollama (Llama 3), which enhances privacy and performance. The system provides two interaction channels. The Web interface, developed with *Streamlit*, offers explainability and source visualization. The WhatsApp bot, integrated via the *Twilio API*, supports practical everyday use. Each response includes the retrieved sources, metadata, and relevance scores, which ensures transparency and auditability. The results indicate the feasibility of integrating semantic retrieval components, vector databases, and large language models in the legal domain. C.L.A.I.T.O.N contributes to the modernization of legal research and practice by demonstrating the potential of Artificial Intelligence to support the analysis and interpretation of legal norms and precedents.

Keywords: Artificial Intelligence; Criminal Law; Retrieval Augmented Generation; Legal Chatbot; ChromaDB; LangChain; Ollama.

Sumário

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface* (Interface de Programação de Aplicações)
MVP	*Minimum Viable Product* (Produto Mínimo Viável)
BGE	Família de modelos de embeddings de texto BGE – *BAAI General Embeddings*
CLAITON	*Chatbot Legal Artificial Intelligence Technology on Network*
DJE	Diário da Justiça Eletrônico
DJEN	Diário da Justiça Eletrônico Nacional
E5	Família de modelos de embeddings de texto E5 – *Text Embeddings for Better Retrieval*
GECP	Gestor de Comunicações Processuais
IA	Inteligência Artificial
JSON	*JavaScript Object Notation*
LGPD	Lei Geral de Proteção de Dados
LLM	*Large Language Model* (Modelo de Linguagem de Grande Porte)
PACER	*Public Access to Court Electronic Records*
PDF	*Portable Document Format* (Formato Portátil de Documento)
PLN	Processamento de Linguagem Natural
RAG	*Retrieval-Augmented Generation* (Recuperação Aumentada por Geração)
RPA	*Robotic Process Automation* (Automação Robótica de Processos)
STF	Supremo Tribunal Federal
STJ	Superior Tribunal de Justiça
TJSP	Tribunal de Justiça do Estado de São Paulo
VLex	Plataforma jurídica VLex (repositório de jurisprudência)

1 Introdução

O ecossistema jurídico contemporâneo é marcado por elevada complexidade informacional e por um volume crescente de documentos, decisões e textos normativos. No contexto brasileiro, os dados do relatório Justiça em Números, publicado anualmente pelo Conselho Nacional de Justiça (CNJ), evidenciam um cenário de elevada litigiosidade, com dezenas de milhões de processos em tramitação no Poder Judiciário (Conselho Nacional de Justiça, 2025). Em meio a esse volume, advogados e demais profissionais do Direito demandam acesso ágil, preciso e contextualizado a jurisprudências e à legislação vigente para embasar sua atuação técnica e a tomada de decisão. Entretanto, a multiplicidade de fontes, a heterogeneidade dos formatos e a necessidade de atualização contínua tornam a pesquisa jurídica uma atividade intensiva em tempo e suscetível a lacunas informacionais, impactando diretamente a eficiência e a qualidade do trabalho. Essa realidade tem impulsionado o interesse por abordagens quantitativas e tecnológicas, como a jurimetria e a análise de dados jurídicos, para lidar com a sobrecarga informacional (Jusbrasil, 2025).

Nesse cenário, a IA, em especial os LLMs combinados à estratégia de RAG, tem se mostrado promissora para mitigar gargalos de busca, sumarização e contextualização de conhecimento jurídico. No Brasil, já se observa o uso de IA e jurimetria na estruturação de grandes volumes de processos judiciais e na extração de padrões a partir de decisões, apoiando a gestão do contencioso e a avaliação de risco (MANDALITI, 2021). A abordagem RAG permite que um modelo de linguagem seja orientado por uma base externa confiável e continuamente atualizada, reduzindo alucinações e ancorando as respostas em evidências verificáveis. Esse paradigma é particularmente relevante para o domínio jurídico, no qual acurácia, rastreabilidade das fontes e segurança informacional são requisitos essenciais.

No âmbito internacional, fornecedores de soluções de *legal tech* vêm explorando RAG para construir assistentes jurídicos mais confiáveis, capazes de combinar LLMs com bases de decisões, contratos e normas cuidadosamente curadas (JU, 2023). Também se observa o uso de RAG em escritórios de advocacia de menor porte, com foco na consulta a documentos internos, redução de alucinações e controle de custos por meio de modelos executados localmente (GOTTLIEB, 2023). Essas experiências reforçam a pertinência de arquiteturas que integrem recuperação de informação jurídica e geração de linguagem natural em fluxos de trabalho já consolidados na prática profissional.

Este trabalho propõe o desenvolvimento de um assistente jurídico conversacional integrado ao *WhatsApp* e uma interface Web, com foco em Direito Penal brasileiro (especificamente para este MVP: embriaguez, estelionato, furto, homicídio, lesão corporal, porte para consumo, receptação, roubo e tráfico). A solução emprega uma arquitetura RAG para consultar, de forma

unificada, duas fontes estruturantes: (i) jurisprudência, decisões e precedentes, e (ii) a legislação penal brasileira. O objetivo é fornecer respostas rápidas, embasadas e acompanhadas de referências, apoiando a pesquisa jurídica cotidiana e a tomada de decisão. Ao privilegiar um canal amplamente difundido entre profissionais (*WhatsApp* e Web), busca-se reduzir barreiras de adoção, aproximando o fluxo de trabalho do advogado de uma interface acessível e familiar, sem abrir mão de critérios de precisão e confiabilidade. Dessa forma, o protótipo responde diretamente a problemas mapeados na prática forense: excesso de informação dispersa, dificuldade de localizar precedentes relevantes em curto espaço de tempo e necessidade de verificar, de forma ágil, a legislação aplicável ao caso concreto.

Do ponto de vista técnico, a proposta contempla um *pipeline* de processamento e indexação textual que inclui extração e saneamento de documentos, enriquecimento de metadados (por exemplo, tribunal, tipo penal, data, número do processo), segmentação em unidades consultáveis (*chunks*), geração de *embeddings* e armazenamento em base vetorial. A etapa de recuperação utiliza a busca por similaridade para compor, juntamente com a consulta do usuário, o contexto a ser fornecido ao LLM, que elabora a resposta final, citando as fontes e apresentando os trechos relevantes. Esse desenho dialoga com práticas recentes em sistemas RAG jurídicos, que enfatizam a importância de um bom estágio de recuperação e de dados de alta qualidade para reduzir respostas imprecisas e aumentar a confiabilidade percebida pelo usuário (MANDALITI, 2021; JU, 2023; GOTTLIEB, 2023). Busca-se, assim, conciliar desempenho, transparência e auditabilidade, ao mesmo tempo em que se oferecem ganhos de produtividade em consultas rotineiras e estudos de caso.

A relevância desta proposta reside em três eixos principais: (a) utilidade prática, redução do tempo de pesquisa e apoio imediato à tomada de decisão em um ambiente marcado por grande volume de processos e informações dispersas (Conselho Nacional de Justiça, 2025); (b) rigor, respostas contextualizadas e referenciadas, ancoradas em jurisprudência e legislação estruturadas em uma base vetorial consultável; e (c) conformidade, alinhamento com princípios de segurança e privacidade previstos na LGPD, por meio da possibilidade de execução local dos componentes críticos e do controle sobre os dados indexados. Adicionalmente, ao adotar componentes abertos e passíveis de execução local (por exemplo, *embeddings* e LLM por inferência local), a solução favorece escalabilidade, portabilidade e controle de custos, aproximando-se de estratégias já defendidas na literatura técnica para adoção responsável de IA no ambiente jurídico (GOTTLIEB, 2023).

2 Soluções Existentes

Durante a fase de pesquisa do presente Trabalho de Conclusão de Curso, o escopo foi orientado para a consulta estruturada a jurisprudências e a legislação penal, com ênfase no Direito Penal brasileiro, por meio de um assistente conversacional com arquitetura RAG.

2.1 Jusbrasil (Jurisprudência)

O Jusbrasil disponibiliza um amplo repositório de decisões de diferentes tribunais brasileiros, acessível em . Trata-se de uma solução consolidada e amplamente utilizada, útil para consultas exploratórias e levantamento inicial de precedentes. Entretanto, para fluxos que demandam refinamento temático e filtragem mais específica (por exemplo, por tipo penal, metadados processuais, tribunal, período e contexto fático), o processo pode tornar-se relativamente trabalhoso e menos direcionado. Na prática, o usuário tende a iterar manualmente entre filtros e resultados, o que eleva o custo cognitivo e o tempo de curadoria. Além disso, a extração de trechos e a padronização de metadados para reuso em pipelines de RAG exigem etapas adicionais de saneamento e estruturação, que nem sempre são triviais a partir da interface pública.

2.2 Juridico.ai

A plataforma Juridico.ai (Juridico.ai, 2025) posiciona-se como uma solução de automação e suporte à pesquisa jurídica baseada em IA. Embora ofereça funcionalidades que convergem com a necessidade de acelerar a obtenção de informações jurídicas, o nível de transparência sobre fontes, estratégias de recuperação e granularidade dos metadados nem sempre é explicitado publicamente, o que pode limitar a auditabilidade desejada em contextos acadêmicos e de validação. Em termos de adequação ao presente projeto, a proposta aqui descrita enfatiza a construção de uma base vetorial sob controle do pesquisador, a anotação de metadados jurídicos relevantes (por exemplo, tribunal, classe e número do processo, data e tipo penal) e a citação direta das fontes recuperadas, com vistas à rastreabilidade e à avaliação objetiva do desempenho.

2.3 e-Xyon

Fundada em 2001, a e-Xyon é uma empresa brasileira que desenvolve soluções tecnológicas para a gestão jurídica. A plataforma integrada utiliza *Robotic Process Automation* (RPA), *workflows* e IA para automatizar e otimizar processos em departamentos jurídicos e escritórios de advocacia. Dentre as soluções oferecidas, destacam-se: (i) Jurimetria, que aplica *big data* e IA para identificar padrões e tendências em decisões judiciais (e-Xyon, 2024c); (ii) Captura

Antecipada 4.0, com extração de dados da petição inicial e cadastro no sistema de gestão (e-Xyon, 2024a); e (iii) GECP, que organiza as notificações dos DJEs e do DJEN (e-Xyon, 2024b). Embora robusta para a gestão e automação processual, a proposta deste TCC concentra-se na recuperação e resposta assistida por RAG para consultas penais, com foco em citações e contexto.

2.4 Juriscraper

Desenvolvida pelo *Free Law Project, a Juriscraper* é uma biblioteca de código aberto escrita em Python, destinada à coleta automatizada de dados jurídicos do sistema judiciário dos Estados Unidos. Ela extrai opiniões de tribunais federais de apelação, cortes estaduais de última instância, argumentos orais e conteúdos do sistema PACER (Free Law Project, 2024). Apesar do foco no ecossistema norte-americano, a Juriscraper evidencia a viabilidade técnica de *scraping* jurídico em larga escala e inspira práticas de coleta, normalização e automação que são pertinentes à construção de bases de conhecimento para pipelines de RAG. No contexto brasileiro, desafios como heterogeneidade de portais, padronização de metadados e conformidade legal reforçam a necessidade de curadoria e saneamento cuidadosos na fase de ingestão de dados.

Tabela 1 – Comparativo simplificado de soluções jurídicas existentes

Solução	**Principal característica**	**Limitações para o TCC**	**País**
Jusbrasil	Repositório de jurisprudência e busca textual	Exige saneamento para RAG; busca menos direcionada	Brasil
Juridico.ai	Automação e IA para pesquisa jurídica	Baixa transparência de fontes e metadados; controle limitado	Brasil
e-Xyon	Gestão jurídica e automação processual	Foco em gestão, não em RAG para Direito Penal	Brasil
Juriscraper	Coleta automatizada de dados jurídicos (*scraping*)	Foco nos EUA; não oferece RAG pronto	EUA

Fonte: Elaborado pelo autor.

3 Objetivos

3.1 Objetivo Geral

Desenvolver um chatbot jurídico inteligente, integrado ao WhatsApp, capaz de fornecer respostas precisas e embasadas em jurisprudências e na legislação penal brasileira, utilizando técnicas de Recuperação Aumentada por Geração (RAG) e modelos de linguagem avançados, visando auxiliar advogados na pesquisa e consulta de informações legais.

3.2 Objetivos Específicos

- Coletar, processar e organizar documentos jurídicos do âmbito penal, incluindo jurisprudências e artigos da legislação penal brasileira, realizando sanitização textual e extração de metadados relevantes (tribunal, tipo penal, data, número do processo, entre outros);
- Implementar o *pipeline* de indexação vetorial utilizando a biblioteca ChromaDB (CHROMA, 2024), com embeddings gerados por modelos da HuggingFace baseados em E5 (WANG et al., 2024a), de forma a viabilizar buscas semânticas eficientes;
- Desenvolver a arquitetura de Recuperação Aumentada por Geração (RAG), integrando a base vetorial ao modelo de linguagem hospedado localmente por meio do Ollama, garantindo respostas contextualizadas e embasadas em fontes jurídicas verificáveis;
- Construir uma interface web com Streamlit (INC., 2024a) para consultas e visualização de resultados, acompanhada de um bot para interação via WhatsApp (INC., 2024b), ampliando o acesso à ferramenta em diferentes meios;
- Implementar mecanismos de citação automática das fontes jurídicas em cada resposta, incluindo metadados e trechos extraídos dos documentos originais, promovendo transparência e rastreabilidade;
- Avaliar o desempenho do sistema com métricas de Recuperação de Informação, como precisão, *recall* e F1-score (métrica de avaliação de modelos de machine learning que combina a precisão e o recall em um único número, sendo calculado como a média harmônica desses dois valores), além da pertinência semântica das respostas geradas;
- Garantir conformidade com a LGPD, assegurando a anonimização de informações sensíveis e a segurança na manipulação dos dados jurídicos;

- Documentar a arquitetura, os fluxos e os resultados obtidos, assegurando a reprodutibilidade do projeto e sua escalabilidade para futuras expansões (como inclusão de novos ramos do Direito).

4 Metodologia

4.1 Visão Geral da Metodologia

Esta seção descreve a metodologia empregada para conceber, implementar e validar o C.L.A.I.T.O.N como um sistema de RAG especializado em Direito Penal brasileiro, com ênfase em precisão jurídica, explicabilidade e auditabilidade. O trabalho abrangeu o desenho completo do *pipeline* de PLN, a curadoria e sanitização de documentos jurídicos, a engenharia de recuperação dual (jurisprudência e legislação), a construção de *prompts* com instruções específicas para o domínio, além da implementação de interfaces web e mensageria. No escopo penal, considerando a amplitude normativa e jurisprudencial, foram priorizados nove tipos penais de alta incidência prática: embriaguez, estelionato, furto, homicídio, lesão corporal, porte para consumo, receptação, roubo e tráfico. Essa delimitação viabiliza um ciclo iterativo de avaliação e melhorias, mantendo cobertura suficiente para demonstrar o valor do sistema e as decisões de engenharia.

4.1.1 Coleta de Dados Jurídicos: Jurisprudências via VLex

Antes da etapa de processamento de linguagem natural, foi necessário constituir uma base de decisões judiciais representativa dos principais crimes penais contemplados neste trabalho. Para isso, utilizou-se a plataforma VLex, um repositório jurídico amplamente empregado por profissionais e pesquisadores, que concentra jurisprudências, legislação e doutrina.

A coleta concentrou-se na seção de jurisprudências da plataforma, selecionando tribunais diretamente relacionados aos crimes estudados: TJSP, STJ (5ª e 6ª Turmas e 3ª Seção) e STF. Esses órgãos foram escolhidos por sua relevância prática e teórica na consolidação de entendimentos em Direito Penal, bem como pela frequência com que aparecem em pesquisas e decisões de referência.

O processo de coleta seguiu, em linhas gerais, as etapas abaixo:

1. Definição de palavras-chave: para cada tipo penal de interesse (embriaguez, estelionato, furto, homicídio, lesão corporal, porte para consumo, receptação, roubo e tráfico), foram definidos termos de busca e combinações de palavras-chave na VLex, com o objetivo de localizar decisões diretamente relacionadas a esses crimes.

2. Filtragem por tribunal e órgão julgador: na interface da VLex, aplicaram-se filtros para restringir os resultados aos tribunais selecionados (TJSP, STJ e STF) e, no caso do STJ, às 5ª e 6ª Turmas e à 3ª Seção, órgãos tradicionalmente responsáveis pelo julgamento de matérias criminais.

3. Seleção manual de decisões: dentre os resultados retornados, foram selecionadas decisões que apresentassem fundamentação minimamente robusta, com discussão do tipo penal e das circunstâncias do caso concreto, de modo a enriquecer a base com conteúdo adequado para o RAG.

4. Download em PDF: para cada jurisprudência selecionada, realizou-se o download do respectivo arquivo em formato PDF. A VLex disponibiliza cada decisão em um PDF individual, já com uma formatação padronizada, o que facilita etapas posteriores de extração automática de texto.

Ao final desse processo, foi constituído um conjunto de aproximadamente 405 decisões em PDF, distribuídas entre os diferentes tribunais e tipos penais selecionados. Essa coleta não pretende esgotar o universo de decisões possíveis, mas formar um corpus inicial, coerente e representativo para experimentação do *pipeline* de RAG, permitindo validar as heurísticas de recuperação semântica, a qualidade da sanitização e a integração com o legislação penal.

4.1.2 PLN, Sanitização e Extração de Metadados

O sucesso de um sistema RAG em Direito depende diretamente da qualidade do texto indexado e de seus metadados. Implementou-se, portanto, um *pipeline* dedicado de sanitização e estruturação de documentos jurídicos, codificado em `sanitaze.py`. Os PDFs de acórdãos obtidos na etapa anterior passam por extração de conteúdo, remoção de artefatos de OCR (cabeçalhos, rodapés, paginações, trechos repetitivos), normalização de caracteres e divisão em *chunks* com janelas sobrepostas para manter coesão semântica. Em paralelo, extrai-se e consolida-se um conjunto de metadados (tribunal, número do processo, data, tipo penal, relator e chaves adicionais) que habilitam filtros e auditoria de respostas.

O *pipeline* de sanitização implementa as seguintes etapas principais:

1. Extração de texto bruto: utiliza-se PyPDF2 para leitura do conteúdo textual dos PDFs, com tratamento de exceções para documentos corrompidos ou protegidos.

2. Remoção de artefatos: aplicam-se expressões regulares e heurísticas para eliminar:

    - Cabeçalhos e rodapés repetitivos (identificados por padrões de tribunal, data e numeração);
    - Marcadores de paginação e quebras artificiais de linha;
    - Caracteres especiais resultantes de OCR mal-sucedido;
    - Trechos duplicados (comum em documentos jurídicos digitalizados).

3. Normalização textual: padronização de espaços em branco, conversão de aspas tipográficas para formato ASCII, remoção de hifens de quebra de linha e unificação de codificação para UTF-8.

4. Extração de metadados: mediante padrões específicos, identifica-se automaticamente tribunal, número do processo, data do julgamento e possíveis tipos penais, a partir de expressões regulares e listas de crimes de interesse. Essa etapa é implementada em uma função específica de extração de metadados no arquivo `sanitaze.py`, cuja estrutura é apresentada em imagem própria neste trabalho, destacando o uso de expressões regulares para localizar campos-chave dentro do texto das decisões.

A partir dessa função de extração, cada decisão passa a possuir uma camada adicional de informação estruturada, essencial para filtragem e explicabilidade dos resultados apresentados ao usuário.

Em seguida, realiza-se a segmentação em *chunks*, na qual o texto é dividido em blocos de tamanho aproximado, com sobreposição, a fim de preservar a continuidade lógica de trechos relevantes. A estratégia adotada consiste em criar blocos de cerca de 500 tokens com sobreposição de 50 tokens entre eles. Essa abordagem reduz o risco de "cortar" raciocínios jurídicos ao meio e melhora a qualidade das respostas do RAG, uma vez que cada *chunk* tende a conter um argumento relativamente completo (por exemplo, a fundamentação de um determinado ponto da decisão).

Por fim, aplicam-se heurísticas de validação e controle de qualidade, descartando *chunks* muito curtos (com poucos caracteres) ou contendo grande quantidade de caracteres especiais, típicos de falhas de extração. Esses filtros contribuem para que a base vetorial contenha, majoritariamente, trechos juridicamente úteis, mitigando ruídos no processo de recuperação semântica.

A legislação penal é representado em *JSON* estruturado, com campos de artigo, capítulo, título e corpo normativo. Essa granularidade viabiliza indexação fina, citações precisas e composição de contexto com recortes normativos pertinentes à consulta. A indexação de ambas as fontes (jurisprudência e legislação) é realizada por scripts específicos (`create_db_jurisprudencia.py` e `create_db_cp.py`), que aplicam o mesmo *backbone* de *embeddings*.

4.1.3 Construção do Banco Vetorial e Heurísticas de Recuperação

A camada de recuperação utiliza ChromaDB com coleções separadas para jurisprudência e legislação, mantendo isolamento lógico e permitindo ajustes de estratégia por fonte. Adotou-se como padrão *embeddings E5* (via *sentence-transformers*) por apresentarem excelente relação entre custo computacional e qualidade semântica no português. As principais heurísticas são:

- Dual retrieve: consulta simultânea às duas coleções, para compor contexto híbrido norma–precedente.
- Top-K balanceado: seleção típica de `k_juris = 3` e `k_lei = 2` (ajustável), controlando a representatividade de cada fonte.

- Ordenação por *score*: fusão e ordenação por similaridade, preservando diversidade de fontes e priorizando trechos mais pertinentes.
- Filtros por metadados: restrições opcionais por tribunal, período, tipo penal e classe processual, elevando precisão e auditabilidade.

O objetivo dessas heurísticas é maximizar o *recall* de evidências relevantes sem sacrificar a concisão do contexto entregue ao LLM. A Figura 1 exibe o trecho do código responsável por executar a estratégia dual e ordenar os resultados por similaridade.

4.1.4 Concepção do RAG e Decisões de Projeto

A arquitetura se estrutura em torno de um retrieval dual que opera, em paralelo, sobre duas coleções vetoriais: *jurisprudência* e *legislação penal*. Essa opção nasce de um requisito substantivo do domínio jurídico: respostas tecnicamente corretas devem estar, simultaneamente, ancoradas na base normativa (legislação penal) e na base de precedentes (jurisprudência dos tribunais superiores), pois é na interseção entre norma e interpretação que se constrói a fundamentação robusta. A camada de geração utiliza LLM local via Ollama, privilegiando controle do ambiente, privacidade e previsibilidade de custos; o LangChain é empregado como *framework* de orquestração para compor o contexto, padronizar instruções e gerenciar a interação com o modelo.

Para garantir transparência técnica e reprodutibilidade, o projeto inclui, nesta seção, recortes do código-fonte que materializam as principais decisões. As figuras a seguir apresentam trechos centrais do *backend* de RAG: carregamento das coleções, recuperação dual com fusão por similaridade, formatação de contexto com metadados e template do *prompt*. Esses trechos (gerados como imagens a partir do código) integram a documentação do método e sustentam, com evidência concreta, as escolhas descritas.

Figura 1 – Recuperação dual em coleções ChromaDB (jurisprudência e legislação) com *top-K* balanceado e reordenação por similaridade. O `dual_retrieve` consolida as fontes com metadados, permitindo fusão explicável.

```python
def load_vectorstores():
    juris = Chroma(
        persist_directory=CHROMA_PATH,
        embedding_function=EMBEDDINGS,
        collection_name=JURIS_COLLECTION
    )
    lei = Chroma(
        persist_directory=CHROMA_PATH,
        embedding_function=EMBEDDINGS,
        collection_name=LEI_COLLECTION
    )
    return juris, lei

def dual_retrieve(question: str, k_juris=3, k_lei=2) -> List[Dict]:
    juris, lei = load_vectorstores()
    docs_juris = juris.similarity_search_with_score(question, k=k_juris)
    docs_lei = lei.similarity_search_with_score(question, k=k_lei)

    # Normalize results into a list of dicts: content, meta, score, origem
    results = []
    for doc, score in docs_juris:
        results.append({
            "content": doc.page_content,
            "metadata": doc.metadata,
            "score": float(score),
            "origem": "jurisprudencia"
        })
    for doc, score in docs_lei:
        results.append({
            "content": doc.page_content,
            "metadata": doc.metadata,
            "score": float(score),
            "origem": "legislacao"
        })
    # Opcional: reordenar por score ascendente
    results.sort(key=lambda x: x["score"])
    return results
```

4.1.5 Engenharia de Prompt e Geração Local

No módulo `rag_core.py`, a função de resposta executa: validação da entrada, recuperação dual, formatação do contexto e construção do prompt com instruções de sistema, pergunta do usuário e trechos selecionados. Para reduzir variação e favorecer reprodutibilidade, a gera-

ção ocorre via Ollama com modelos *Llama*-based, usualmente com temperatura baixa e `top-p` controlado. O formato de saída inclui resposta, citações, metadados e escores de similaridade, facilitando explicabilidade e auditoria.

No módulo `rag_core.py`, a função de resposta executa: validação da entrada, recuperação dual, formatação do contexto e construção do prompt com instruções de sistema, pergunta do usuário e trechos selecionados. Para reduzir variação e favorecer reprodutibilidade, a geração ocorre via Ollama com modelos *Llama*-based, com parâmetros cuidadosamente ajustados para o domínio jurídico.

A escolha dos hiperparâmetros de geração foi orientada por testes empíricos e pela natureza do domínio:

- Temperatura = 0.1: valor baixo que prioriza determinismo e previsibilidade nas respostas. No contexto jurídico, onde precisão terminológica e fidedignidade às fontes são essenciais, temperaturas elevadas aumentariam o risco de variações semânticas indesejadas e de ”alucinações”(geração de conteúdo não suportado pelos documentos recuperados). A temperatura de 0.1 mantém o modelo próximo às distribuições de probabilidade mais altas, favorecendo respostas consistentes e tecnicamente corretas.

- Top-p = 0.9: implementa *nucleus sampling*, restringindo a amostragem aos tokens que acumulam 90% da massa de probabilidade. Esse valor equilibra diversidade lexical suficiente para respostas naturais com controle sobre tokens de baixa probabilidade que poderiam introduzir imprecisões. Em testes, valores de top-p acima de 0.95 ocasionalmente geraram construções verbosas ou tangenciais; valores abaixo de 0.85 tornaram as respostas excessivamente repetitivas.

- Num_ctx = 4096: define a janela de contexto efetiva do modelo. Esse valor foi dimensionado para acomodar: (i) as instruções de sistema (400 tokens), (ii) o contexto formatado com trechos recuperados (2500–3000 tokens, equivalente aos 6000 caracteres do limite de `format_contexts`), e (iii) a pergunta do usuário e margem para a resposta gerada (600–1000 tokens). A escolha de 4096 tokens representa o limite prático do modelo Llama 3 3B utilizado, maximizando a quantidade de evidências jurídicas disponíveis ao LLM sem comprometer desempenho ou causar truncamento indesejado.

Esses parâmetros, em conjunto, configuram um regime de geração conservador e controlado, adequado para aplicações onde a confiabilidade e a rastreabilidade das respostas são prioritárias. O formato de saída inclui resposta, citações, metadados e escores de similaridade, facilitando explicabilidade e auditoria.

A Figura 2 documenta a rotina de formatação e truncamento do contexto (com cabeçalhos padronizados “[Fonte i]”, id, origem e *score*). A Figura 3 apresenta o *template* de instruções de

sistema e de *prompt* que estabelece o comportamento esperado do modelo (não inventar, citar fontes, explicar termos complexos de forma breve). Por fim, a Figura 4 mostra a chamada HTTP ao servidor do Ollama com parâmetros de geração controlados e a Figura 5 sintetiza o *pipeline* ponta a ponta de resposta.

Figura 2 – Formatação e truncamento de contexto com metadados e cabeçalhos “[Fonte i]”, preservando explicabilidade e controle de janela efetiva.

```python
def format_contexts(chunks: List[Dict], max_chars: int = 6000) -> Tuple[str, List[Dict]]:
    formatted = []
    used = []
    total = 0
    for i, ch in enumerate(chunks, start=1):
        doc_id = ch["metadata"].get("id") or ch["metadata"].get("source") or ch["metadata"].get("file")
or "doc"
        titulo = ch["metadata"].get("titulo") or ch["metadata"].get("title") or doc_id
        header = f"[Fonte {i}] id={doc_id}, origem={ch['origem']}, score={ch['score']:.4f}, titulo=
{titulo}\n"
        block = header + ch["content"].strip() + "\n"
        if total + len(block) > max_chars:
            break
        formatted.append(block)
        used.append(ch)
        total += len(block)
    return "\n".join(formatted), used
```

Figura 3 – Instruções de sistema e *template* de *prompt* explicitando diretrizes de fidedignidade, estilo e citação de fontes.

```
SYSTEM_INSTRUCTIONS = """
Você é um assistente jurídico especializado em Direito Penal brasileiro.

### Objetivo:
Responder perguntas sobre crimes e infrações penais de forma **correta, objetiva e compreensível até para
leigos**.

### Diretrizes:
- Baseie-se apenas nas leis e jurisprudências brasileiras.
- **Não invente** artigos, súmulas ou decisões.
- Se os "Contextos Recuperados" não forem suficientes, diga claramente que não é possível concluir.
- Quando houver **termos jurídicos difíceis**, explique-os **brevemente em linguagem simples**, antes da
resposta jurídica.
- Depois dessa explicação, apresente a **resposta técnica resumida** (3 a 6 frases), de forma direta e
impessoal.
- Cite as fontes ao final no formato: [Fonte {N} – {source_meta}].
- Evite opiniões pessoais e especulações.

### Estrutura sugerida da resposta:
1. (Opcional) Explicação simples de termos jurídicos difíceis.
2. Resposta jurídica objetiva e fundamentada.
3. Fontes citadas no formato indicado.
"""

PROMPT_TEMPLATE = """
[SISTEMA]
{system_instructions}

[PERGUNTA DO USUÁRIO]
{question}

[CONTEXTOS RECUPERADOS]
{contexts}

[INSTRUÇÕES DE SAÍDA]
- Responda em português do Brasil.
- Seja conciso, técnico e juridicamente preciso.
- Cite as fontes no final no formato [Fonte N – {{source_meta}}].
"""
```

Figura 4 – Chamada ao Ollama (inferência local) com temperatura baixa, `top-p` e `num_ctx` ajustados para estabilidade e previsibilidade.

```python
OLLAMA_MODEL = os.getenv("OLLAMA_MODEL") #llama3.2:3b-instruct-q8_0
TEMPERATURE = float(os.getenv("OLLAMA_TEMPERATURE")) #0.1
NUM_CTX = int(os.getenv("OLLAMA_NUM_CTX")) #4096
TOP_P = float(os.getenv("OLLAMA_TOP_P")) #0.9


def call_ollama(prompt: str, model: str = OLLAMA_MODEL) -> str:
    url = f"{OLLAMA_URL}/api/generate"
    payload = {
        "model": model,
        "prompt": prompt,
        "options": {
            "temperature": TEMPERATURE,
            "top_p": TOP_P,
            "num_ctx": NUM_CTX
        },
        "stream": False
    }
    r = requests.post(url, json=payload, timeout=180)
    r.raise_for_status()
    data = r.json()
    return data.get("response", "").strip()
```

Figura 5 – *Pipeline* de resposta: recuperação dual, construção do contexto, composição do *prompt*, geração local e retorno de fontes com escores.

```python
def answer_question(question: str) -> Tuple[str, List[Dict]]:
    try:
        # Retrieve
        retrieved = dual_retrieve(question, k_juris=K_JURIS, k_lei=K_LEI)

        if not retrieved:
            return "Não encontrei informações relevantes sobre isso. Pode reformular a pergunta?",
[]
        contexts_str, used = format_contexts(retrieved)
        prompt = PROMPT_TEMPLATE.format(
            system_instructions=SYSTEM_INSTRUCTIONS,
            question=question.strip(),
            contexts=contexts_str if contexts_str else "(nenhum contexto recuperado)"
        )
        response = call_ollama(prompt)

        fontes = []
        for ch in used:
            meta = ch["metadata"]
            fontes.append({
                "titulo": meta.get("titulo") or meta.get("title") or meta.get("id") or "Documento",
                "id": meta.get("id") or meta.get("source") or meta.get("file") or "N/A",
                "origem": ch["origem"],
                "score": ch["score"],
                "text": ch["content"]
            })

        return response, fontes

    except Exception as e:
        print(f"[ERRO em answer_question] {e}")
        import traceback
        traceback.print_exc()
        return "Erro ao processar sua pergunta. Tente novamente.", []
```

4.1.6 Construção das Interfaces e Justificativas

A decisão de implementar duas interfaces distintas, web e WhatsApp, deriva de uma análise das necessidades e contextos de uso do público-alvo: estudantes de Direito, pesquisadores e profissionais da área jurídica.

A interface web (`streamlit_app.py`) foi concebida para cenários de pesquisa aprofundada e auditoria. Streamlit foi escolhido por sua capacidade de prototipagem rápida, facilidade de manutenção e adequação a aplicações de ciência de dados. A interface oferece:

- Pré-visualização de trechos recuperados: antes da geração final, o usuário visualiza os documentos selecionados pelo retrieval, com *scores* de similaridade, permitindo avaliar a pertinência das fontes.
- Explicabilidade granular: cada resposta é acompanhada de metadados completos (tribunal, processo, data, tipo penal), facilitando a verificação e citação em trabalhos acadêmicos.

- Acesso ao texto integral: expansão dos trechos para visualização do documento completo, essencial para análise jurídica detalhada.
- Filtros avançados: parâmetros de busca por tribunal, período, tipo penal e outros metadados, refinando a recuperação conforme o interesse do usuário.

Essa interface privilegia transparência e controle, valores fundamentais quando se trata de informação jurídica, onde a origem e a confiabilidade das fontes são tão importantes quanto o conteúdo da resposta.

O bot WhatsApp (`whatssap_bot.py`), por sua vez, foi projetado para consultas rápidas e ubiquidade. A escolha do WhatsApp como canal se justifica por:

- Penetração e familiaridade: o WhatsApp é a plataforma de mensageria mais utilizada no Brasil, tornando o acesso ao assistente imediato e sem necessidade de instalação de aplicativos adicionais.
- Contexto de uso móvel: profissionais e estudantes frequentemente necessitam de consultas jurídicas em trânsito, em audiências ou durante estudos fora do ambiente de escritório/biblioteca.
- Baixa fricção: a interação por mensagem de texto reduz barreiras de entrada e permite uso assíncrono, onde o usuário envia a pergunta e recebe a resposta quando conveniente.

A integração via Twilio API implementa um fluxo de processamento assíncrono: o *webhook* confirma o recebimento da mensagem imediatamente (evitando *timeout* da API do WhatsApp) e dispara uma *thread* em *background* que executa o *pipeline* RAG completo. Quando a resposta é gerada, o sistema a envia de volta ao usuário no mesmo canal. Essa arquitetura garante responsividade percebida, mesmo com latências de 35–65 segundos no processamento.

A complementaridade das interfaces reflete uma compreensão de que diferentes contextos de uso demandam diferentes affordances: a web para análise e auditoria; o WhatsApp para agilidade e acessibilidade. Ambas compartilham o mesmo *backend* RAG, garantindo consistência nas respostas e facilitando manutenção e evolução do sistema.

Essa dualidade também serve a um propósito pedagógico e de validação: a interface web permite ciclos rápidos de teste e refinamento durante o desenvolvimento, enquanto o bot WhatsApp demonstra a viabilidade de aplicação prática e a escalabilidade da solução para contextos reais de uso.

4.1.7 Avaliação dos Modelos de Embeddings

Dentro da metodologia deste trabalho, foi conduzida uma etapa específica de avaliação dos modelos de *embeddings* utilizados no pipeline de RAG do sistema C.L.A.I.T.O.N. O objetivo

foi comparar, de forma quantitativa, diferentes alternativas de representação vetorial de textos para o domínio do Direito Penal brasileiro e, a partir disso, definir a configuração adotada na versão final do sistema.

4.1.7.1 Procedimento de Teste

Para essa avaliação, foi criado um conjunto de 9 perguntas jurídicas, cobrindo temas centrais de Direito Penal (por exemplo, homicídio doloso x culposo, princípio da insignificância, roubo qualificado etc.). Para cada pergunta, foi definido manualmente um gabarito contendo:

- uma resposta esperada em texto;
- uma lista de artigos relevantes da legislação penal brasileira;
- uma lista de trechos de jurisprudência relevantes, identificados por *chunk IDs*.

Essas informações foram armazenadas em um arquivo CSV (`perguntas_gabarito.csv`), utilizado pelo *script* de testes (`test.py`). Para cada pergunta, o sistema executou a função de *dual retrieval*, que consulta simultaneamente as coleções de legislação e de jurisprudência no ChromaDB, recuperando os k documentos mais similares de cada base (com $k_{\text{lei}} = 3$ e $k_{\text{juris}} = 3$).

Os identificadores dos documentos recuperados foram então comparados com o gabarito, separadamente para:

- legislação: campo `artigo` (número do artigo da legislação penal);
- jurisprudência: campo `chunk_id` (identificador do trecho da decisão).

A partir dessa comparação, foram calculadas, para cada pergunta, as métricas padrão de recuperação de informação (MANNING; RAGHAVAN; SCHÜTZE, 2008; POWERS, 2011):

- Precisão@k: proporção de documentos relevantes entre os documentos recuperados;
- *Recall*@k: proporção de documentos relevantes que foram efetivamente recuperados;
- F1-Score: média harmônica entre precisão e *recall*.

Os testes foram executados para três modelos de *embeddings* e dois modelos de LLM, usando sempre o mesmo conjunto de perguntas e o mesmo banco vetorial (com a lógica de *dual retrieval* já implementada no módulo `rag_core.py`). As métricas foram agregadas como médias sobre as 9 perguntas de teste.

4.1.7.2 Modelos Avaliados

Foram comparados três modelos de *embeddings*:

1. intfloat/multilingual-e5-small (WANG et al., 2024b): modelo multilíngue compacto da família E5, otimizado para tarefas de recuperação semântica, utilizado como *baseline* do sistema;
2. intfloat/multilingual-e5-base (WANG et al., 2024b): versão maior do mesmo modelo, com mais parâmetros e, em tese, maior capacidade de representação;
3. neuralmind/bert-base-portuguese-cased (SOUZA; NOGUEIRA; LOTUFO, 2020): modelo BERTimbau, pré-treinado especificamente em português brasileiro, não otimizado originalmente para *embeddings* de sentença, mas bastante utilizado no contexto nacional.

Para isolar o efeito dos *embeddings*, cada experimento foi repetido com dois modelos de LLM (LLaMA 3B e LLaMA 8B). As métricas apresentadas a seguir são médias entre esses dois LLMs para cada modelo de *embedding*.

4.1.7.3 Resultados e Análise

A Tabela 2 resume os resultados obtidos, apresentando as métricas médias de precisão, *recall* e F1-Score, separadas para legislação e jurisprudência, bem como o tempo médio de geração por resposta.

Tabela 2 – Comparação entre modelos de *embeddings* no sistema C.L.A.I.T.O.N (média entre LLaMA 3B e 8B, $n = 9$ perguntas).

Embedding	P@3 Lei	R@3 Lei	F1 Lei	P@3 Juris	R@3 Juris	F1 Juris	Tempo (s)
E5-Small (multilíngue)	0,39	**0,67**	**0,48**	1,00	1,00	1,00	**13,56**
E5-Base (multilíngue)	0,39	0,59	0,45	1,00	1,00	1,00	13,76
BERT-PT (português brasileiro)	0,17	0,22	0,19	1,00	1,00	1,00	14,69

Fonte: Elaborado pelo autor com base nos experimentos realizados.

De forma objetiva, os principais pontos observados foram:

- Jurisprudência: todos os modelos de *embeddings* atingiram desempenho máximo (precisão, *recall* e F1 iguais a 1,00) na recuperação de trechos de jurisprudência. Isso indica que, para o conjunto de teste utilizado, a tarefa de localizar as decisões relevantes era relativamente simples para todos os modelos avaliados.
- Legislação: a principal diferença entre os modelos aparece na recuperação de artigos da legislação penal. O E5-Small apresentou o melhor equilíbrio entre precisão e *recall*, com F1 de 0,48 e *recall* de 0,67, recuperando a maior parte dos artigos considerados relevantes

no gabarito. O E5-Base obteve desempenho um pouco inferior (F1 de 0,45 e *recall* de 0,59), sugerindo que o aumento de tamanho do modelo nem sempre se traduz em ganho efetivo de recuperação nesse domínio específico. Já o BERT-PT apresentou desempenho significativamente menor em legislação (F1 de 0,19), recuperando, em média, apenas 22% dos artigos relevantes.

- Tempo: os três modelos apresentaram tempos médios de geração de resposta muito próximos (entre 13 e 15 segundos), com uma ligeira vantagem para o E5-Small. Como o tempo total é dominado pela etapa de geração do LLM, a escolha do *embedding* não impactou de forma decisiva no tempo de resposta.

4.1.7.4 Decisão de Modelo

Com base nesses resultados, a configuração adotada na versão final do CLAITON prioriza:

- o modelo de *embedding* intfloat/multilingual-e5-small, por apresentar o melhor F1-Score em legislação, maior *recall* de artigos relevantes e bom desempenho em jurisprudência, com custo computacional reduzido;
- o LLM LLaMA 3B quantizado (Q8), que, nos experimentos, obteve métricas de recuperação idênticas ao modelo LLaMA 8B, porém com tempo de geração de resposta muito menor, sendo mais adequado para o uso interativo em interface web e bot de WhatsApp.

Assim, as decisões de modelo foram guiadas diretamente pelos dados obtidos nesta etapa de testes, buscando um equilíbrio entre qualidade de recuperação, tempo de resposta e viabilidade computacional para o contexto.

4.1.8 Instrumentação e Reprodutibilidade

O repositório documenta o workflow de ponta a ponta no `README.md`: sanitização de PDFs, indexação das coleções, execução das interfaces e configurações do ambiente (`.env`). A estrutura de pastas (`dados_sanitizados/`, `vectordb/`, `chromadb/`) e os scripts de criação de base asseguram reprodutibilidade e manutenção. Essa engenharia operacional permite atualização incremental das coleções, inclusão de novos tipos penais e portabilidade da solução.

Em síntese, a metodologia integra PLN aplicado ao domínio jurídico, recuperação semântica em bases vetoriais, geração local com LLM e interfaces complementares para produzir respostas explicáveis, auditáveis e útil ao estudo e à prática do Direito Penal. Para além do funcionamento, o projeto destaca-se por decisões que priorizam confiabilidade e transparência, valores essenciais em ambientes jurídicos.

4.2 Fluxo do Sistema

O sistema proposto, denominado C.L.A.I.T.O.N, opera como um assistente jurídico baseado em RAG com dois canais de interação: (i) WhatsApp e (ii) interface web. Em ambos os casos, as respostas são embasadas por uma base de conhecimento composta por jurisprudências e pela legislação penal brasileira, com citações e metadados. A seguir, descrevem-se os fluxos.

4.2.1 Fluxo via WhatsApp

1. Interação Inicial (WhatsApp): O usuário envia uma pergunta por WhatsApp. A mensagem é recebida por um conector de mensageria e encaminhada ao *backend* do sistema.

2. Processamento da Consulta: A entrada passa por saneamento e normalização. Em seguida, é convertida em vetor de consulta por um modelo de *embeddings*.

3. Recuperação (RAG): O vetor de consulta é utilizado para busca por similaridade na base vetorial (ChromaDB), indexada com jurisprudências e artigos dlegislação penal enriquecidas com metadados.

4. Geração de Resposta: Os trechos recuperados são concatenados ao *prompt* do LLM (inferência local via Ollama) que gera a resposta. São mantidas as citações e metadados relevantes.

5. Entrega: A resposta é enviada ao usuário via WhatsApp, com links/identificadores das fontes e, quando aplicável, pontuações de similaridade.

4.2.2 Fluxo via Interface Web

1. Interação Inicial (Web): O usuário acessa uma interface em *browser* para realizar consultas. O formulário permite parâmetros adicionais (por exemplo, tribunal, intervalo de datas, tipo penal).

2. Processamento e Recuperação: A consulta é vetorizada e enviada ao mecanismo de busca semântica (ChromaDB). O usuário pode visualizar a prévia dos documentos recuperados antes da geração final da resposta.

3. Geração e Explicabilidade: O LLM elabora a resposta a partir do contexto recuperado. A interface exibe as citações, metadados, trechos relevantes e indicadores (por exemplo, *similarity score*).

4. Exploração e Auditoria: O usuário pode expandir para visualizar o texto integral das decisões e artigos citados, além de exportar a resposta e referências.

Esse desenho em dois fluxos preserva coerência arquitetural (mesmo *pipeline* RAG), ao mesmo tempo em que explora capacidades específicas de cada canal: agilidade e ubiquidade no WhatsApp; filtragem avançada, visualização e auditoria na interface web.

4.3 Funcionalidades

4.3.1 Funcionalidades via WhatsApp

- Perguntas em linguagem natural: Consulta direta sobre temas de Direito Penal, jurisprudências e artigos da legislação penal.
- Respostas embasadas: Retorno com citações das fontes recuperadas, incluindo metadados essenciais.
- Acompanhamento de contexto: Manutenção de histórico curto da conversa para preservar o tema da sessão.

4.3.2 Funcionalidades via Web

- Filtros avançados: Parâmetros de consulta por tribunal, período, tipo penal, classe, número do processo e outros metadados.
- Pré-visualização de resultados: Lista de trechos recuperados com *similarity score* antes da geração final da resposta.
- Explicabilidade e auditoria: Exibição das citações com metadados, trechos destacados e acesso ao texto integral.
- Exportação: Opção de exportar respostas e referências para apoio à pesquisa.

4.4 Tecnologias Utilizadas

As tecnologias foram selecionadas para favorecer reprodutibilidade, controle de custos (preferência por execução local) e transparência das fontes.

- Python (FOUNDATION, 2024): Linguagem principal para ingestão, processamento, indexação e orquestração do *pipeline* RAG.
- LangChain (DEVELOPERS, 2024): Orquestração do fluxo RAG (retrieval, *prompting*, *routing* e formatação de respostas).
- Embeddings (E5) (WANG et al., 2024a): Geração de *embeddings* multilingues (família E5) via Hugging Face para busca semântica.

- Sentence-Transformers (REIMERS; GUREVYCH, 2019): *Framework* para uso prático dos modelos de *embeddings*.
- ChromaDB (CHROMA, 2024): Banco de dados vetorial para armazenamento dos vetores e metadados, com *indexes* por coleção (jurisprudência e legislação).
- LLM local (Ollama) (OLLAMA, 2024): Inferência local do modelo de linguagem, integrando o contexto recuperado para geração da resposta.
- Interface Web (Streamlit) (INC., 2024a): Front-end simples para consultas com filtros, visualização de resultados, explicabilidade e exportação.
- Integração WhatsApp (Twilio) (INC., 2024b): Conector de mensageria para envio/recebimento de mensagens no canal WhatsApp.
- Pré-processamento de PDFs (PyPDF2): Extração de texto de decisões em PDF para posterior saneamento e indexação.

4.5 Resultados

Para avaliar a eficácia do C.L.A.I.T.O.N como assistente jurídico em Direito Penal, consideraram-se cenários que exigem articulação entre norma e precedente (p. ex., distinção entre furto e roubo; requisitos de estelionato; tipicidade em porte para consumo; hipóteses de lesão corporal leve/gravíssima). Os resultados observados, em ciclos de teste com usuários-alvo (estudantes e profissionais), indicaram:

- Pertinência jurídica e fundamentação: as respostas se mostraram aderentes aos trechos recuperados, com boa integração entre artigos da legislação penal e decisões paradigmáticas, evidenciando o valor do retrieval dual para qualidade da fundamentação.
- Explicabilidade e auditoria: a apresentação de *scores* de similaridade e metadados das fontes (id, origem, título) facilitou a validação do conteúdo e a confiança do usuário na resposta.
- Cobertura balanceada de fontes: em consultas onde a resposta dependia tanto de interpretação jurisprudencial quanto de leitura normativa, a fusão por *score* ofereceu equilíbrio útil; ajustes de `k_juris` e `k_lei` mostraram-se eficientes para cenários distintos.
- Desempenho prático: a latência ponta a ponta permaneceu, em média, entre 35 s e 65 s, variando conforme o tamanho dos contextos e a carga computacional local; o uso de Ollama contribuiu para previsibilidade e controle.

- Redução de alucinações: as instruções de sistema (*prompting*) e a limitação do contexto com cabeçalhos “[Fonte i]” colaboraram para reduzir trechos não suportados diretamente pelas fontes recuperadas.

- Tratamento de ambiguidades jurídicas: em cenários de múltiplo enquadramento típico, especialmente na distinção entre porte de drogas para consumo pessoal e tráfico de entorpecentes, o sistema apresentou desempenho misto. Em perguntas mais diretas, baseadas em hipóteses factuais simples, o C.L.A.I.T.O.N conseguiu recuperar dispositivos legais e precedentes pertinentes, apontando critérios usuais (quantidade, forma de acondicionamento, circunstâncias da prisão, antecedentes, dentre outros). Entretanto, em consultas mais abertas ou formuladas de maneira ambígua, observou-se certa oscilação na priorização de teses, ora inclinando-se a interpretar como porte, ora como tráfico, evidenciando que a desambiguação fina entre enquadramentos concorrentes ainda depende, em grande medida, da intervenção crítica do usuário e de futuros ajustes nos critérios de recuperação e sumarização.

Em síntese, considerando as limitações computacionais do ambiente local de desenvolvimento e os resultados obtidos nos ciclos de teste, a performance do C.L.A.I.T.O.N pode ser considerada satisfatória, ainda que claramente passível de aprimoramentos. As decisões de engenharia adotadas mostraram-se adequadas: o retrieval dual contribuiu para a qualidade da fundamentação jurídica; a formatação explicável do contexto favoreceu a auditoria e a confiança do usuário; e a inferência local permitiu maior governança de custos e de privacidade. Ao mesmo tempo, os testes indicaram oportunidades de evolução, como a otimização de latência, o refinamento dos parâmetros de recuperação e a exploração de modelos mais eficientes. Esses achados sustentam a utilidade do C.L.A.I.T.O.N como apoio à pesquisa e ao estudo no âmbito do Direito Penal brasileiro, ao mesmo tempo em que evidenciam o seu potencial de melhoria contínua e de extensão a novos tipos penais e a outros ramos do Direito.

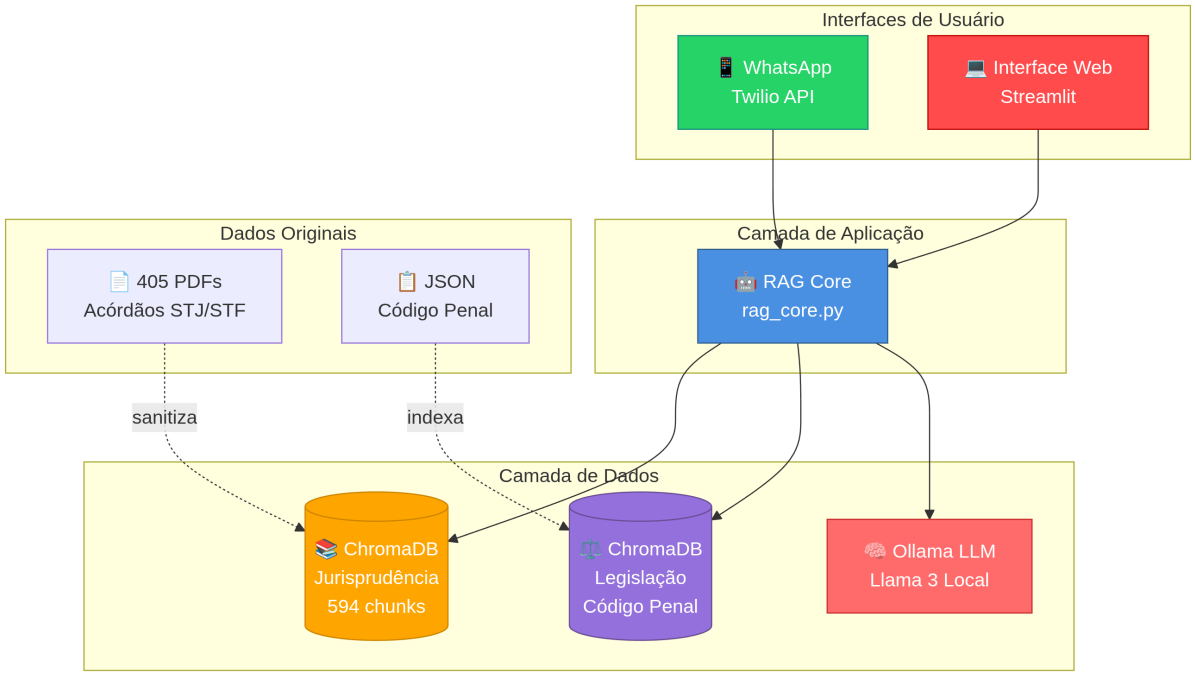
5 Diagramas e Fluxo

Esta seção apresenta a arquitetura geral do sistema e os fluxos de interação pelos dois canais disponíveis (WhatsApp e Web), além do pipeline interno de RAG. Os diagramas visam oferecer uma visão sintética e auditável do funcionamento do C.L.A.I.T.O.N.

5.1 Arquitetura Geral

A Figura 6 ilustra a arquitetura de alto nível com três camadas principais: (i) interfaces de usuário (WhatsApp via Twilio e interface Web em Streamlit); (ii) camada de aplicação (núcleo RAG); e (iii) camada de dados (duas coleções no ChromaDB, jurisprudência e legislação, e o LLM local via Ollama). A ingestão de dados contempla a sanitização de acórdãos em PDF e a indexação estruturada da legislação penal em JSON.

Figura 6 – Arquitetura geral do C.L.A.I.T.O.N: canais de interação, núcleo RAG e camadas de dados.



Fonte: Elaborado pelo autor.

5.2 Fluxo de Interação via WhatsApp

As Figuras 7, 8 e 9 apresentam o fluxo assíncrono de atendimento no WhatsApp, dividido em três etapas complementares.

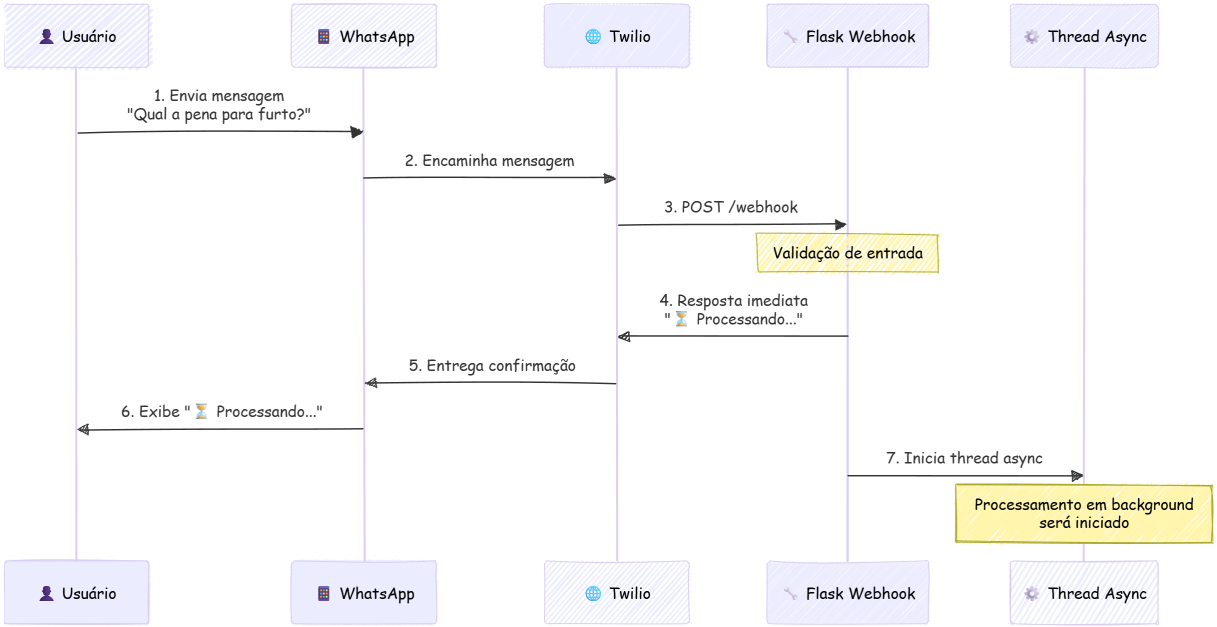
Na primeira etapa (Figura 7), a mensagem enviada pelo usuário é recebida pelo WhatsApp, encaminhada à Twilio e entregue ao *webhook* Flask. Após a validação da entrada, o sistema confirma imediatamente ao usuário que a solicitação está em processamento e inicia uma *thread* em *background*, evitando *timeouts* no canal de mensageria.

Na segunda etapa (Figura 8), essa *thread* em *background* executa o núcleo do pipeline RAG: chamada à função `answer_question()`, realização de `dual_retrieve` nas duas coleções vetoriais (jurisprudência e legislação com `k_juris=3` e `k_lei=2`), retorno de cinco documentos com seus respectivos *scores* de similaridade, formatação do contexto com metadados via `format_contexts()` e, por fim, envio do *prompt* estruturado ao LLM local através de `call_ollama()`.

Na terceira etapa (Figura 9), o LLM conclui a inferência local e retorna a resposta gerada ao `RAG Core`. Este, por sua vez, encapsula a resposta junto com as fontes recuperadas (incluindo metadados completos: tribunal, processo, data, tipo penal e *scores* de similaridade), preparando o pacote completo para envio. A *thread* em *background* recebe esse pacote estruturado e se prepara para transmiti-lo via Twilio API.

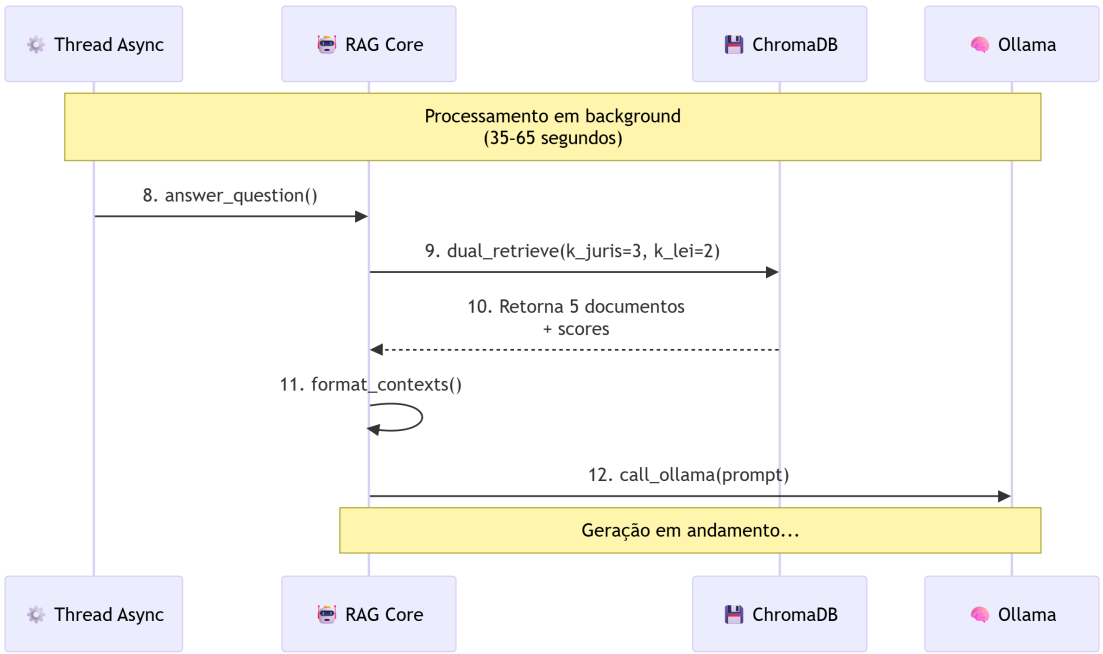
Na quarta etapa (Figura 10), a *thread* em *background* envia a resposta completa via Twilio API, que a entrega ao WhatsApp para exibição ao usuário final. O usuário recebe a resposta fundamentada acompanhada das fontes com metadados e, quando aplicável, dos *scores* de similaridade associados aos trechos recuperados. O tempo total do fluxo varia tipicamente entre 35 e 65 segundos, dependendo da complexidade da consulta e da carga computacional local.

Figura 7 – Fluxo de atendimento via WhatsApp – Parte 1: da mensagem do usuário até o disparo do processamento assíncrono.



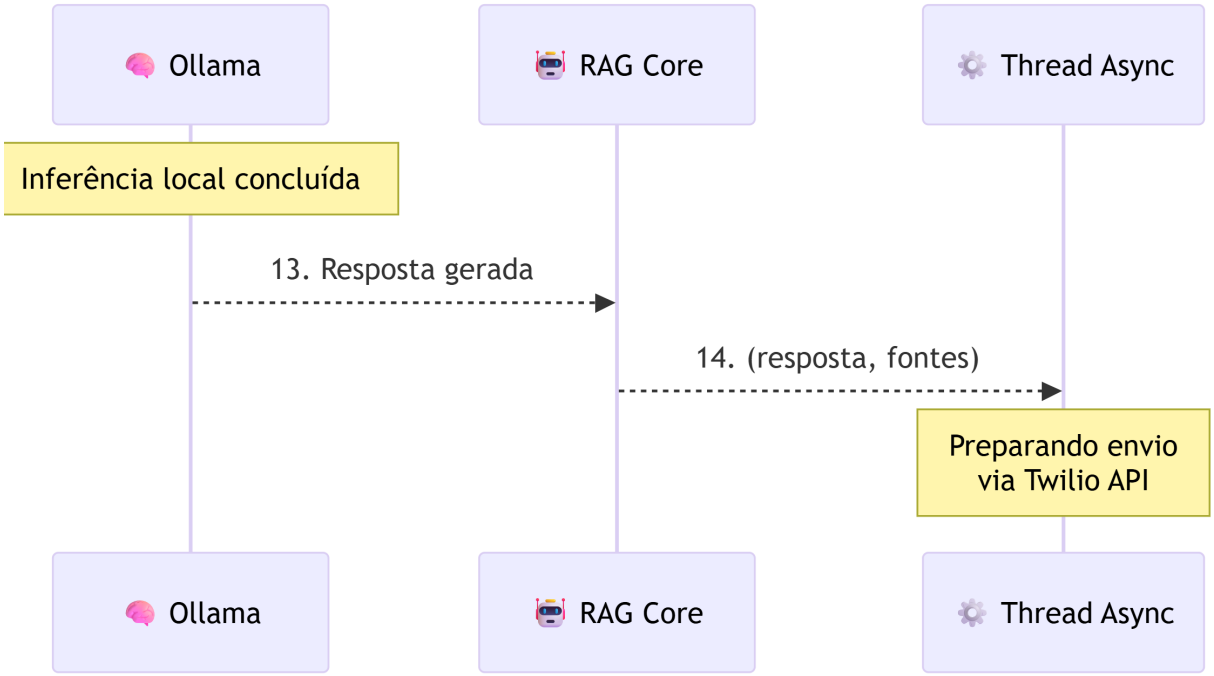
Fonte: Elaborado pelo autor.

Figura 8 – Fluxo de atendimento via WhatsApp – Parte 2: execução do pipeline RAG (recuperação dual, formatação de contexto e chamada ao LLM).



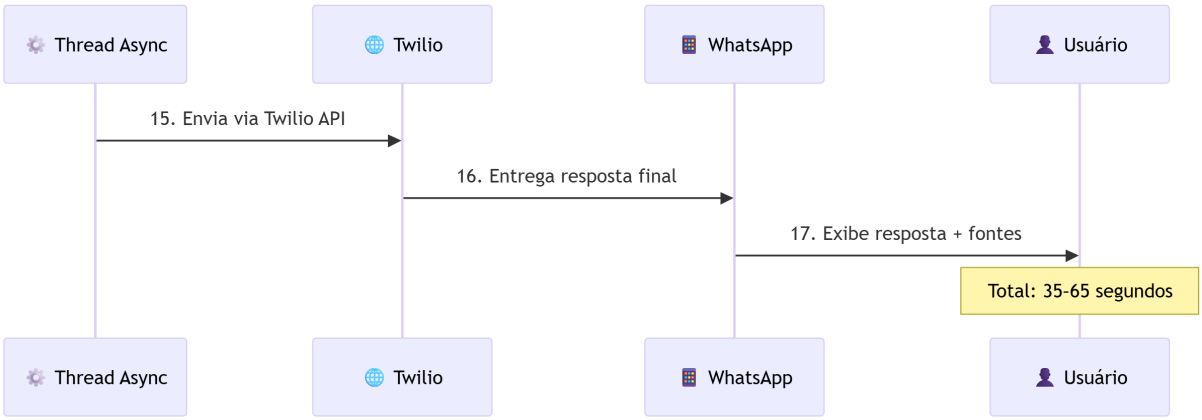
Fonte: Elaborado pelo autor.

Figura 9 – Fluxo de atendimento via WhatsApp – Parte 3: conclusão da inferência pelo LLM e encapsulamento da resposta com fontes.



Fonte: Elaborado pelo autor.

Figura 10 – Fluxo de atendimento via WhatsApp – Parte 4: envio da resposta completa ao usuário com fontes e metadados.



Fonte: Elaborado pelo autor.

5.3 Fluxo de Interação via Interface Web

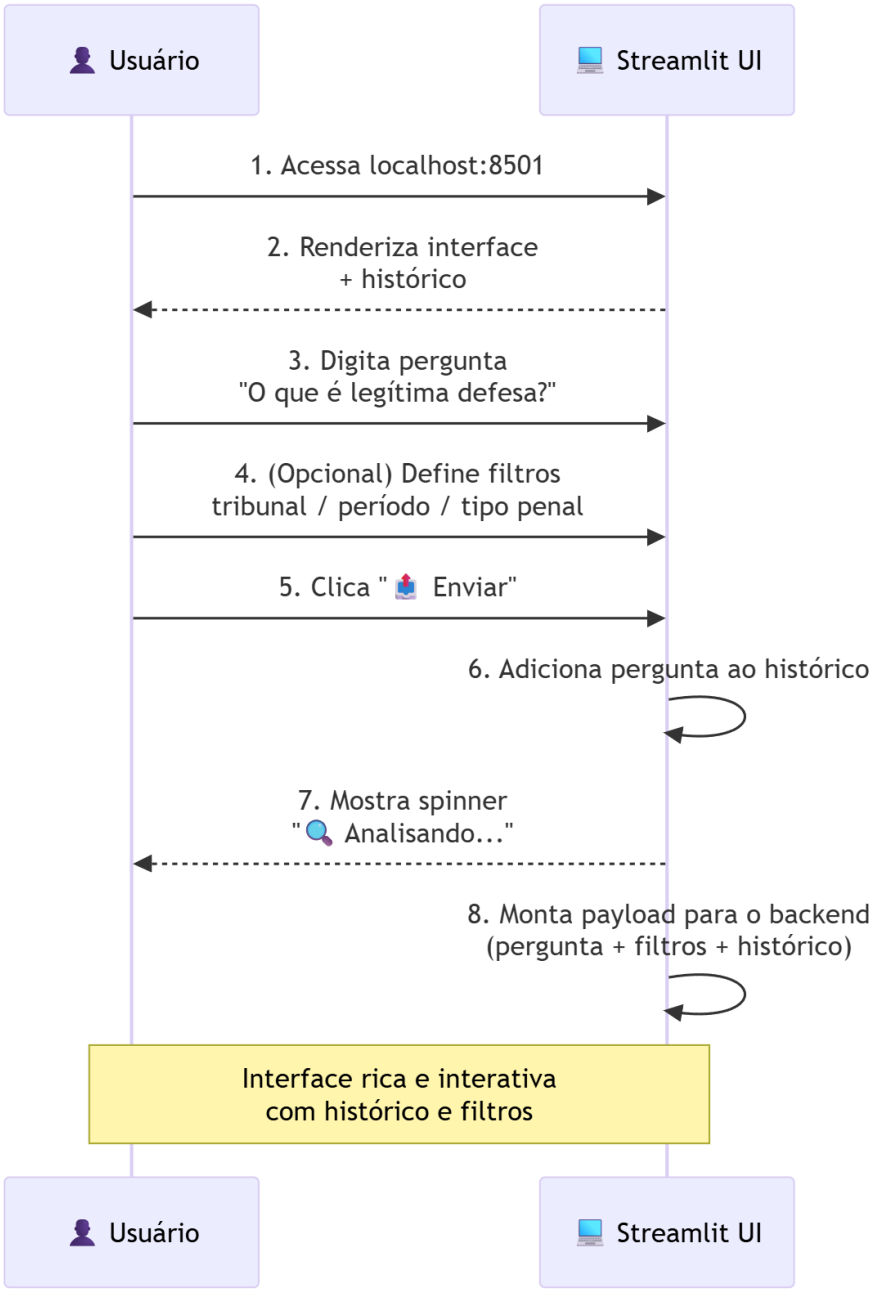
As Figuras 11, 12 e 13 mostram o fluxo de uso na interface Web construída em Streamlit, dividido em três etapas complementares.

Na primeira etapa (Figura 11), o usuário acessa a aplicação (em ambiente local, via `localhost:8501`), e a interface renderiza tanto os componentes de entrada quanto o histórico de interação da sessão corrente. O usuário digita sua pergunta, podendo opcionalmente aplicar filtros (por tribunal, período, tipo penal etc.), e aciona o envio. A interface registra a nova mensagem no histórico da sessão e exibe um indicador de processamento (*spinner*), sinalizando que o sistema está analisando a consulta. Nessa etapa, o Streamlit também consolida os parâmetros relevantes (pergunta, filtros selecionados e contexto anterior) em um *payload* que será enviado ao backend.

Na segunda etapa (Figura 12), o backend recebe a chamada, executando o pipeline RAG completo: realização de `dual_retrieve()` nas coleções vetoriais de jurisprudência e legislação (com `k_juris=3` e `k_lei=2`), ordenação dos documentos por *score* de similaridade, formatação do contexto com metadados via `format_contexts()` e chamada ao LLM local por meio de `call_ollama()`. Ao término da inferência, o núcleo RAG retorna ao Streamlit a resposta em linguagem natural, acompanhada dos trechos recuperados, metadados e *scores* de relevância. O tempo típico dessa etapa varia entre 15 e 35 segundos, dependendo da complexidade da consulta e da carga computacional.

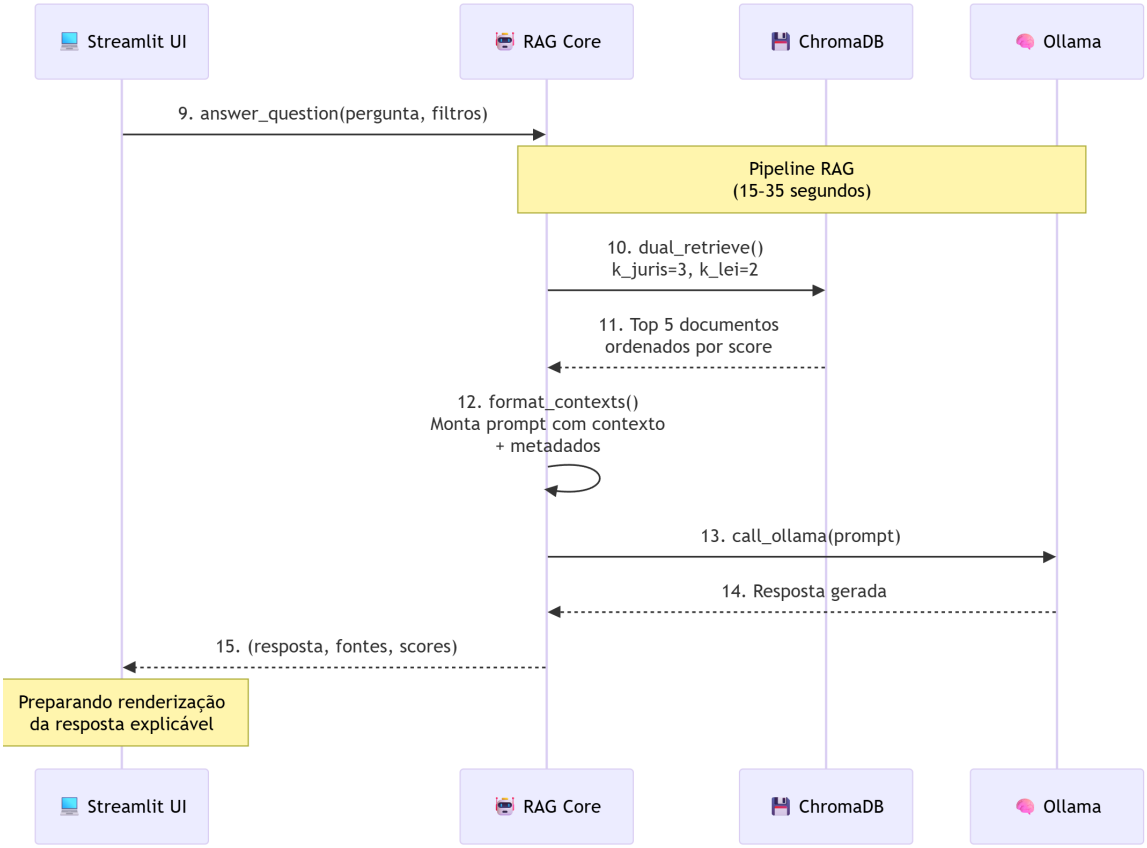
Na terceira etapa (Figura 13), a interface Streamlit renderiza a mensagem do assistente jurídico, estruturando a apresentação com seção expandível de fontes, exibição de *scores* de similaridade e acesso ao texto integral dos documentos recuperados. O usuário pode então auditar as referências utilizadas, expandir trechos para visualização completa, copiar conteúdo relevante e interagir com os metadados apresentados (tribunal, processo, data, tipo penal). O histórico é mantido na sessão, favorecendo a continuidade do contexto em consultas subsequentes e permitindo que o usuário construa uma linha de raciocínio jurídico ao longo de múltiplas perguntas relacionadas.

Figura 11 – Fluxo de atendimento via Web (Streamlit) – Parte 1: interação do usuário com a interface, filtros e histórico.



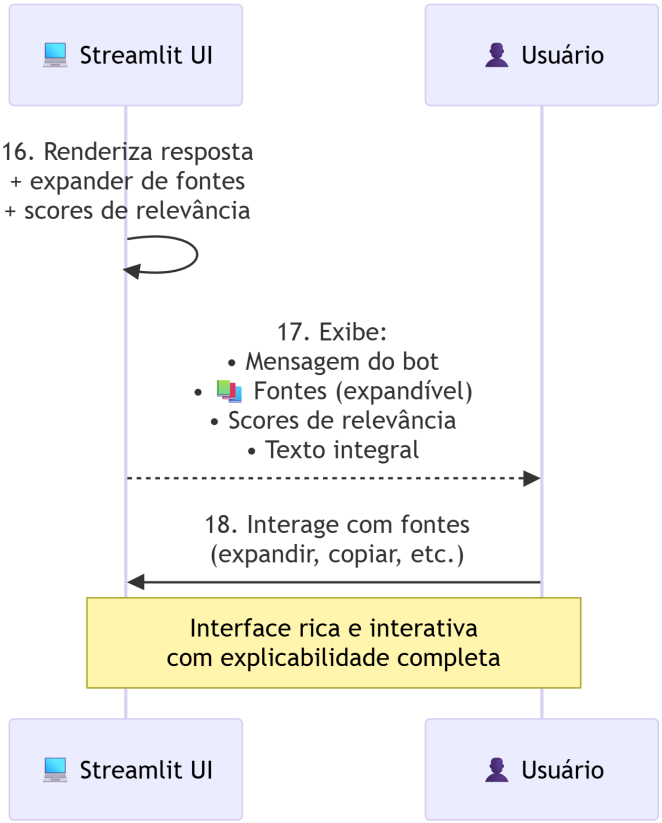
Fonte: Elaborado pelo autor.

Figura 12 – Fluxo de atendimento via Web (Streamlit) – Parte 2: execução do pipeline RAG no backend e retorno da resposta com fontes.



Fonte: Elaborado pelo autor.

Figura 13 – Fluxo de atendimento via Web (Streamlit) – Parte 3: renderização da resposta explicável e interação do usuário com as fontes.



Fonte: Elaborado pelo autor.

5.4 Pipeline Interno de RAG

As Figuras 14, 15, 16 e 17 apresentam, de forma detalhada, o funcionamento interno da *pipeline* RAG utilizada pelo CLAITON, desde a entrada da pergunta até a geração da resposta explicável com fontes estruturadas.

Na primeira etapa (Figura 14), a *pipeline* recebe a pergunta do usuário e aplica uma validação mínima de qualidade, exigindo um comprimento de pelo menos 10 caracteres. Caso a pergunta seja considerada muito curta, o sistema retorna imediatamente uma mensagem de erro, orientando o usuário a reformular a consulta. Se a validação for bem-sucedida, é disparada a função `dual_retrieve`, responsável por acionar, em paralelo, as buscas nas coleções vetoriais de jurisprudência e de legislação.

Na segunda etapa (Figura 15), a função `dual_retrieve` executa duas consultas independentes ao ChromaDB: uma na coleção de jurisprudência, retornando os três acórdãos mais similares com seus respectivos *scores*, e outra na coleção da legislação penal, recuperando dois artigos com maior similaridade. Em seguida, os resultados são mesclados e ordenados em um único conjunto, com base no *score* de relevância. Caso nenhuma evidência seja encontrada

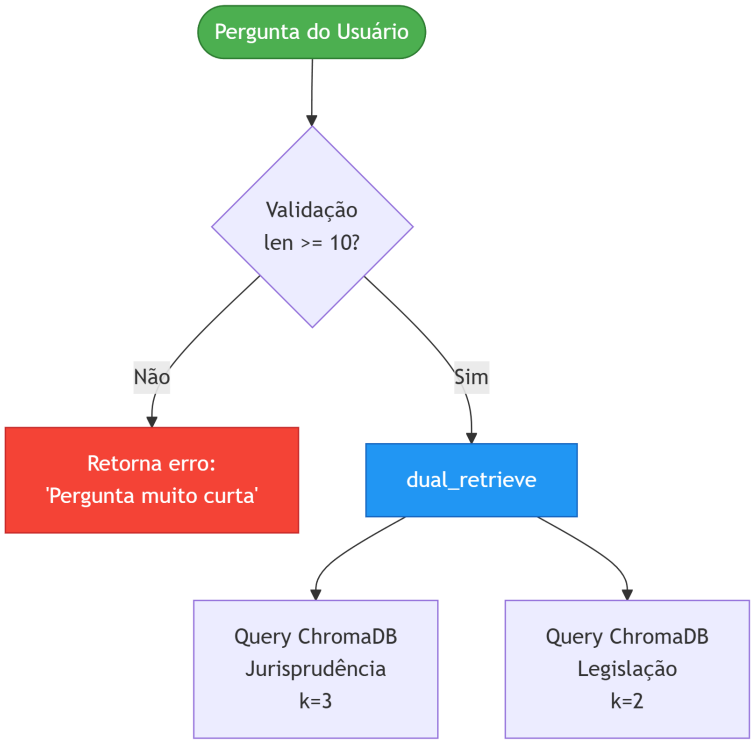
acima de um limiar aceitável, a *pipeline* retorna um erro do tipo “Nenhum resultado”, evitando gerar respostas sem respaldo documental. Se houver resultados, o fluxo prossegue para a fase de formatação de contexto e geração da resposta.

Na terceira etapa (Figura 16), a *pipeline* inicia a preparação do contexto para o modelo de linguagem. A partir dos documentos recuperados, já ordenados por relevância, a função `format_contexts` seleciona e organiza os trechos mais pertinentes, incorporando metadados relevantes (como tribunal, número do processo, artigo da legislação penal, data, entre outros). Em seguida, é construído o *prompt* final, combinando instruções de sistema (definindo o papel de assistente jurídico penal), a pergunta do usuário e até cinco contextos textuais derivados dos documentos recuperados. Esse *prompt* enriquecido é então enviado à função `call_ollama`, que realiza a inferência no modelo Llama 3 em ambiente local.

Por fim, na quarta etapa (Figura 17), a resposta gerada pelo modelo de linguagem é pós-processada pelo núcleo RAG. A *pipeline* formata as fontes utilizadas, estruturando-as em um formato adequado para apresentação na interface: título do documento, origem (tribunal ou legislação), *score* de similaridade e o texto integral ou trecho relevante. O resultado final devolvido ao cliente (Streamlit ou bot de WhatsApp) contém, portanto, a resposta em linguagem natural acompanhada de um conjunto de referências auditáveis, o que reforça a rastreabilidade e a confiabilidade das informações fornecidas pelo assistente jurídico.

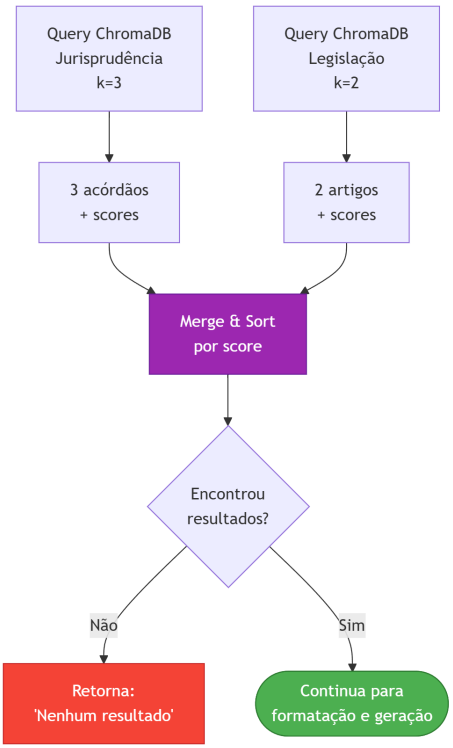
O fluxo contempla tratamento de erros em pontos críticos (validação de entrada e ausência de resultados), garantindo robustez operacional e transparência no processo de recuperação e geração.

Figura 14 – Pipeline RAG – Parte 1: validação da pergunta do usuário e disparo da busca dual.



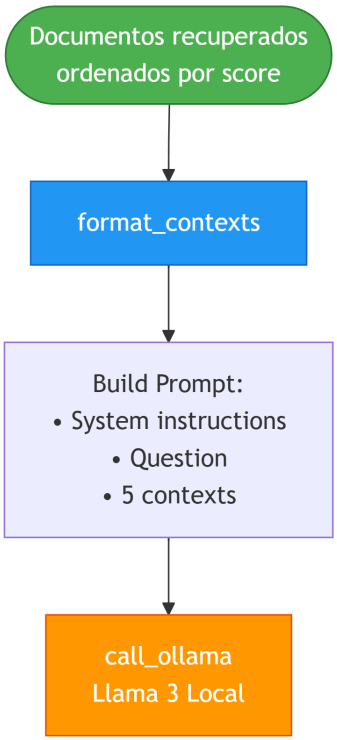
Fonte: Elaborado pelo autor.

Figura 15 – Pipeline RAG – Parte 2: fusão, ordenação dos resultados e checagem de existência de evidências.



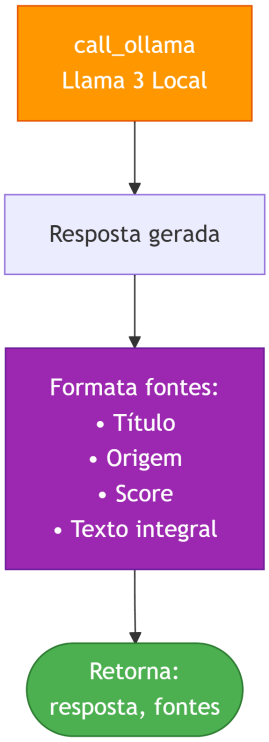
Fonte: Elaborado pelo autor.

Figura 16 – Pipeline RAG – Parte 3: formatação dos contextos recuperados e construção do *prompt* para o LLM.



Fonte: Elaborado pelo autor.

Figura 17 – Pipeline RAG – Parte 4: geração da resposta e estruturação das fontes explicáveis retornadas ao cliente.



Fonte: Elaborado pelo autor.

6 Conclusão

O presente trabalho apresentou o C.L.A.I.T.O.N, um assistente jurídico inteligente voltado ao Direito Penal brasileiro, fundamentado em uma arquitetura de *Retrieval-Augmented Generation* (RAG). O sistema integra a recuperação de informações jurídicas, provenientes tanto da jurisprudência quanto da legislação, com a geração de linguagem natural por modelos de aprendizado profundo, oferecendo respostas fundamentadas, contextualizadas e acompanhadas de suas fontes.

A proposta se apoiou na utilização de ChromaDB como banco de dados vetorial, permitindo a indexação semântica de documentos, e na adoção de modelos de embeddings E5 para representação textual eficiente. O processamento e a organização dos dados jurídicos possibilitaram a criação de duas coleções vetoriais, uma voltada à jurisprudência, composta por centenas de acórdãos, e outra dedicada à legislação penal estruturada a partir da legislação penal. O componente de inferência, baseado no LLM local Ollama (Llama 3), garantiu autonomia e controle sobre a geração das respostas, sem dependência de serviços externos.

A implementação do C.L.A.I.T.O.N, com dupla interface, Web (Streamlit) e WhatsApp (Twilio), ampliou a acessibilidade do sistema, permitindo consultas tanto em contexto de pesquisa aprofundada quanto em uso cotidiano. Pela interface Web, o usuário tem acesso a uma experiência mais analítica, com visualização de fontes, *scores* de similaridade e texto integral. No canal WhatsApp, a interação privilegia a agilidade, viabilizando respostas rápidas e contextualizadas.

Do ponto de vista técnico, o sistema demonstrou que a integração entre componentes de recuperação semântica, bases vetoriais e modelos de linguagem de grande porte pode ser realizada de forma eficaz em ambiente local, com eficiência aceitável e resultados interpretáveis. A estrutura proposta também se mostrou escalável e flexível, favorecendo futuras expansões, como o acréscimo de novas bases ou a substituição do modelo de linguagem principal.

6.1 Experiência de Desenvolvimento e Desafios Enfrentados

Durante o processo de desenvolvimento, algumas etapas se mostraram particularmente desafiadoras. A mais complexa delas foi, sem dúvida, a obtenção e sanitização dos dados em formato PDF. As jurisprudências, muitas vezes em estrutura inconsistente, exigiram a criação de rotinas robustas para extração e limpeza de texto, bem como para padronização e enriquecimento de metadados. Esse trabalho manual e automatizado foi fundamental para garantir a qualidade das informações armazenadas na base vetorial.

Outro grande desafio foi o equilíbrio entre eficiência, rapidez e precisão nas respostas. Testaram-se diferentes modelos de *embeddings* e configurações no Ollama, resultando na escolha

do modelo E5 por combinar boa performance semântica e tempo de resposta tolerável. Esse processo de tentativa e erro foi crucial para encontrar o ponto ótimo entre robustez jurídica e viabilidade operacional.

Houve, entretanto, aspectos que não atingiram plenamente as expectativas. Um ponto que se mostrou limitado foi a ausência de um mecanismo de persistência de sessão, a capacidade do sistema de "lembrar" perguntas anteriores e construir respostas contextuais ao longo do diálogo. Essa funcionalidade teria tornado o assistente mais fluido e humanizado, permitindo continuidade na interação, especialmente relevante em consultas jurídicas multilinhas. Apesar de idealizada, sua implementação exigiria alterações mais profundas na arquitetura do pipeline e na gestão de memória do modelo local, o que foi adiado para versões futuras.

Em contrapartida, entre os principais acertos destacam-se: a implementação eficaz do retrieval dual (jurisprudência + legislação), que garantiu respostas mais ricas e equilibradas; e o sucesso das integrações técnicas, tanto com a camada de mensageria via Twilio quanto com a interface Web através do Streamlit. Essas integrações demonstraram a versatilidade da arquitetura e a possibilidade concreta de aplicar o C.L.A.I.T.O.N tanto em ambientes acadêmicos quanto em usos práticos.

6.2 Lições Aprendidas e Possíveis Melhorias

Ao longo do desenvolvimento, foi possível compreender a importância da curadoria dos dados e do desenho criterioso dos *prompts* para o sucesso de sistemas de RAG. A função de dual retrieval, aliada à orquestração via LangChain, mostrou-se uma escolha eficiente para conciliar precisão técnica com explicabilidade, que se tornou um valor essencial no contexto jurídico.

Para resolver os problemas identificados, sobretudo a falta de memória de sessão, seria necessário implementar uma camada de armazenamento de contexto (por exemplo, Redis ou SQLite) em conjunto com ajustes na lógica de construção do *prompt*, de forma a reaproveitar perguntas anteriores. Outro ponto de melhoria envolve a otimização de desempenho, seja pela quantização dos modelos locais, seja pela adoção de técnicas de pré-carregamento de vetores ou paralelização das consultas às coleções.

Também se observou que nem todas as funcionalidades puderam ser testadas exaustivamente devido a limitações de tempo e recursos. Gostaria-se, por exemplo, de ter testado comparativamente diferentes famílias de embeddings (como BGE e Instructor) e explorado novos métodos de reordenação semântica dos resultados (*re-ranking*) para aprimorar a precisão das respostas.

6.3 Melhorias Futuras

Como perspectivas futuras, planeja-se ampliar a robustez do banco de dados jurídico, incorporando novas fontes (doutrinas e tribunais estaduais) e explorando atualizações periódicas automáticas. Também se pretende testar novos modelos de embeddings e estratégias de RAG híbridas, buscando o equilíbrio ideal entre custo computacional, tempo de resposta e qualidade semântica.

Outra vertente importante será o investimento em eficiência operacional, visando reduzir a latência e permitir que o sistema se comporte de modo mais responsivo em consultas complexas. A implementação de um gerenciamento de sessão permitirá ao C.L.A.I.T.O.N manter o histórico recente de interações, adaptando as respostas de modo contínuo e aperfeiçoando a experiência do usuário.

Por fim, apesar das limitações enfrentadas, o trabalho alcançou seus objetivos principais e abriu caminho para avanços concretos na aplicação de métodos de IA no campo jurídico. A jornada de desenvolvimento do C.L.A.I.T.O.N demonstrou que, com curadoria adequada e decisões de arquitetura bem fundamentadas, é possível construir sistemas inteligentes, explicáveis e eficientes, que traduzam o potencial das técnicas de inteligência artificial em ferramentas úteis e confiáveis para o Direito.

APÊNDICE A – Protótipo do Assistente Jurídico CLAITON

Este apêndice apresenta o protótipo do assistente jurídico C.L.A.I.T.O.N, desenvolvido como parte deste trabalho de conclusão de curso. O sistema implementa uma *pipeline* de RAG para consultas em Direito Penal brasileiro, utilizando jurisprudência e a legislação penal. O C.L.A.I.T.O.N oferece interfaces de interação via Web (Streamlit) e WhatsApp.

O código-fonte completo, scripts de processamento de dados, implementação da *pipeline* RAG, componentes de interface e instruções detalhadas para configuração e execução estão disponíveis no repositório GitHub:

<https://github.com/pedrororatto/claiton-app>

O repositório está organizado para facilitar a compreensão da arquitetura do projeto e a replicação do ambiente de desenvolvimento. Recomenda-se clonar o repositório e seguir as instruções contidas no arquivo ‘README.md‘ para explorar o projeto.

Referências

CHROMA. *ChromaDB: The AI-native open-source embedding database*. 2024. Disponível em: <https://www.trychroma.com/>. 15, 33

Conselho Nacional de Justiça. *Justiça em Números*. 2025. <https://www.cnj.jus.br/pesquisas-judiciarias/justica-em-numeros/>. Relatório anual do Conselho Nacional de Justiça. Acesso em: 01 nov. 2025. 11, 12

DEVELOPERS, L. *LangChain: Build applications with LLMs through composability*. 2024. Disponível em: <https://langchain.readthedocs.io/>. 32

e-Xyon. *Captura Antecipada 4.0*. 2024. Disponível em: <https://www.e-xyon.com.br/captura-antecipada-40>. 14

e-Xyon. *Gestor de Comunicações Processuais*. 2024. Disponível em: <https://www.e-xyon.com.br/gecp-gestor-e-xyon>. 14

e-Xyon. *Jurimetria preditiva: antecipando tendências jurídicas usando big data e IA*. 2024. Disponível em: <https://www.e-xyon.com.br/post/jurimetria-preditiva-ia>. 13

FOUNDATION, P. S. *Python Programming Language*. 2024. Disponível em: <https://www.python.org/>. 32

Free Law Project. *Juriscraper Legal Scraping Toolkit*. 2024. Disponível em: <https://free.law/projects/juriscraper>. 14

GOTTLIEB, K. Building reliable rag pipelines for small and medium law firms. *Journal of Legal Technology*, v. 12, n. 1, p. 23–39, 2023. Estudo técnico sobre RAG com foco em escritórios de advocacia de pequeno e médio porte. 11, 12

INC., S. *Streamlit: Turn data scripts into shareable web apps*. 2024. Disponível em: <https://streamlit.io/>. 15, 33

INC., T. *Twilio WhatsApp API*. 2024. Disponível em: <https://www.twilio.com/whatsapp>. 15, 33

JU, J. Retrieval-augmented generation for legal assistants: Architectures and challenges. In: *Proceedings of the Conference on AI for Law*. [S.l.: s.n.], 2023. p. 101–115. Trabalho técnico sobre uso de RAG em assistentes jurídicos. 11, 12

Juridico.ai. *Juridico.ai: Plataforma de automação e suporte à pesquisa jurídica*. 2025. Acesso em: 01 nov. 2025. Disponível em: <https://juridico.ai/>. 13

Jusbrasil. *Jurimetria e dados no Direito brasileiro*. 2025. Artigo sobre jurimetria e uso de dados no ecossistema jurídico brasileiro. Acesso em: 01 nov. 2025. Disponível em: <https://www.jusbrasil.com.br/>. 11

MANDALITI, R. Inteligência artificial, processos judiciais e jurimetria no brasil. *Revista de Direito e Novas Tecnologias*, v. 5, n. 2, p. 45–62, 2021. Discussão sobre aplicação de IA e jurimetria na gestão de grandes volumes de processos judiciais no Brasil. 11, 12

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN 978-0-521-86571-5. 28

OLLAMA. *Ollama: Run LLMs locally*. 2024. Disponível em: <https://ollama.com/>. 33

POWERS, D. M. W. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, v. 2, n. 1, p. 37–63, 2011. 28

REIMERS, N.; GUREVYCH, I. *Sentence-Transformers*. 2019. Acesso em: 07 fev. 2025. Disponível em: <https://www.sbert.net/>. 33

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: pretrained BERT models for Brazilian Portuguese. In: *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23*. [S.l.]: Springer, 2020. p. 403–417. 29

WANG, L.; YANG, N.; HUANG, X.; YANG, L.; MAJUMDER, R.; WEI, F. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024. 15, 32

WANG, L.; YANG, N.; HUANG, X.; YANG, L.; MAJUMDER, R.; WEI, F. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024. Disponível em: <https://arxiv.org/abs/2402.05672>. 29