UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Ciência da Computação

Mikael Hernandes de Jesus Filgueiras Barros

Desenvolvimento de um Agente Inteligente para Tomada de Contas Especial no TCE-MA

São Luís

2025

Mikael Hernandes de Jesus Filgueiras Barros

Desenvolvimento de um Agente Inteligente para Tomada de Contas Especial no TCE-MA

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profª. Drª Simara Vieira da Rocha

São Luís

2025

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Filgueiras Barros, Mikael Hernandes de Jesus.
Desenvolvimento de um Agente Inteligente para Tomada de Contas Especial no TCE-MA / Mikael Hernandes de Jesus Filgueiras Barros. - 2025.
51 f.

Orientador(a): Simara Vieira da Rocha.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2025.

1. Inteligência Artificial. 2. Processamento de Linguagem Natural. 3. Large Language Models. 4. Tribunal de Contas. 5. Tomada de Contas Especial. I. Rocha, Simara Vieira da. II. Título.

Mikael Hernandes de Jesus Filgueiras Barros

Desenvolvimento de um Agente Inteligente para Tomada de Contas Especial no TCE-MA

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho _____ em São Luís, 18 agosto de 2025:

Profª. Drª Simara Vieira da Rocha
Orientador

Prof. Msc. Carlos Eduardo Portela Serra de Castro
Examinador

Prof. Dr. Anselmo Cardoso de Paiva
Examinador

São Luís
2025

Agradecimentos

Agradeço, em primeiro lugar e de forma especial, a minha mãe Neuzanil Filgueiras, por ser a base de tudo me proporcionando amor, carinho e educação. Em seguida, às minhas irmãs, ao meu irmão, aos meus avós e ao meu pai pelo incentivo aos meus estudos, pelo apoio constante aos meus sonhos e pela motivação que sempre me ofereceram. Sem vocês, esta jornada não seria possível.

A minha namorada, Ana Beatriz, por estar ao meu lado em todos os momentos e ser uma fonte de inspiração, sempre me incentivando a buscar o meu melhor.

A Prof[a]. Dr[a]. Simara, não apenas pela valiosa orientação na elaboração deste trabalho, mas por todo o aprendizado proporcionado durante a graduação. Sua disponibilidade e incentivo foram fundamentais para a conclusão desta etapa.

A todos os professores da UFMA, pelo conhecimento compartilhado que foi a base para minha formação profissional e para a realização deste trabalho.

Aos líderes de equipe do setor de desenvolvimento no TCE-MA, com especial destaque a Bruno, que como um verdadeiro mentor compartilhou seu conhecimento e me auxiliou não somente em projetos mas também me inspirou a cultivar a busca constante pela excelência profissional.

A Leidiane, por toda a dedicação em sanar dúvidas e auxiliar nas mais diversas questões acadêmicas.

A Felipe e Pedro por terem sido meus companheiros de jornada, sempre prontos a ajudar e compartilhar conhecimento, tornando essa experiência ainda mais enriquecedora.

Aos amigos de curso, pela companhia, pelo apoio e pelos momentos de descontração ao longo de toda a graduação. A presença de vocês foi essencial para tornar esta jornada mais leve.

"Não fiques em terreno plano.
Não subas muito alto.
O mais belo olhar sobre o mundo
Está a meia encosta."

Friedrich Nietzsche, em *"A Gaia Ciência"*

Resumo

O Tribunal de Contas do Estado do Maranhão (TCE-MA) enfrenta desafios no processo de Tomada de Contas Especial, instrumento fundamental para apuração de danos ao erário. A análise manual de extensos documentos e a complexidade normativa envolvida demandam considerável tempo dos auditores, representando um desafio significativo para o controle externo. Este trabalho apresenta o desenvolvimento de um agente inteligente baseado em Processamento de Linguagem Natural (PLN) e *Large Language Models* (LLMs) para auxiliar os auditores na análise documental e elaboração de relatórios técnicos. A metodologia empregada envolveu o levantamento de requisitos junto aos auditores do TCE-MA, o desenvolvimento do agente utilizando a técnica *Retrieval-Augmented Generation* (RAG), e a implementação de uma arquitetura integrada que combina o modelo Gemini do Google, PostgreSQL para armazenamento de dados, extensão pgvector para *embeddings*, armazenamento de objetos na AWS S3 e construção de *workflows* via N8N. A solução foi integrada à infraestrutura existente do tribunal através de uma interface web desenvolvida em Next.js com autenticação via Keycloak. Os resultados demonstraram redução no tempo necessário para elaboração de relatórios, diminuindo de vários dias para algumas horas de trabalho efetivo. Além do ganho de produtividade, observou-se melhoria na padronização dos relatórios gerados, garantindo conformidade com todos os requisitos normativos. O trabalho demonstra o potencial transformador da inteligência artificial na modernização de processos de controle governamental, contribuindo para maior eficiência na fiscalização de recursos públicos.

Palavras-chave: Inteligência Artificial. Processamento de Linguagem Natural. Large Language Models. Tribunal de Contas. Tomada de Contas Especial. Retrieval-Augmented Generation.

Abstract

The Court of Accounts of the State of Maranhão (TCE-MA) faces challenges in the Special Accounts Review process, a fundamental instrument for investigating damage to public funds. The manual analysis of extensive documents and the normative complexity involved demand considerable time from auditors, representing a significant challenge for external control. This work presents the development of an intelligent agent based on Natural Language Processing (NLP) and Large Language Models (LLMs) to assist auditors in document analysis and technical report preparation. The methodology employed involved requirements gathering with TCE-MA auditors, agent development using the Retrieval-Augmented Generation (RAG) technique, and implementation of an integrated architecture combining Google's Gemini model, PostgreSQL for data storage, pgvector extension for embeddings, object storage in AWS S3, and N8N-based workflow construction. The solution was integrated into the court's existing infrastructure through a web interface developed in Next.js with Keycloak authentication. Results demonstrated a reduction in the time required for report preparation, decreasing from several days to a few hours of effective work. Beyond productivity gains, improvements were observed in the standardization of generated reports, ensuring compliance with all regulatory requirements. This work demonstrates the transformative potential of artificial intelligence in modernizing government control processes, contributing to greater efficiency in public resource oversight.

Keywords: Artificial Intelligence. Natural Language Processing. Large Language Models. Court of Accounts. Special Accounts Review. Retrieval-Augmented Generation.

Lista de Ilustrações

Lista de Abreviaturas e Siglas

ACID	*Atomicidade, Consistência, Isolamento e Durabilidade*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
CC	*Ciência da Computação*
CORS	*Cross-Origin Resource Sharing*
DOM	*Document Object Model*
GPT	*Generative Pre-trained Transformer*
HTTP	*Hypertext Transfer Protocol*
IA	*Inteligência Artificial*
JSON	*JavaScript Object Notation*
JSX	*JavaScript XML*
JWT	*JSON Web Tokens*
LLM	*Large Language Model*
LSTM	*Long Short-Term Memory*
OAuth	*Open Authorization*
OCR	*Optical Character Recognition*
PLN	*Processamento de Linguagem Natural*
RAG	*Retrieval-Augmented Generation*
REST	*Representational State Transfer*
RNN	*Redes Neurais Recorrentes*
S3	*Simple Storage Service*
SEO	*Search Engine Optimization*
SGBD	*Sistema de Gerenciamento de Banco de Dados*

SQL	*Structured Query Language*
SSG	*Static Site Generation*
SSR	*Server Side Rendering*
TCE	*Tomada de Contas Especial*
TCE-MA	*Tribunal de Contas do Estado do Maranhão*
TCU	*Tribunal de Contas da União*
UFMA	*Universidade Federal do Maranhão*
XML	*Extensible Markup Language*

Sumário

1 Introdução

A administração pública brasileira enfrenta desafios crescentes na fiscalização e controle do uso de recursos públicos. Os Tribunais de Contas, como órgãos de controle externo, desempenham papel fundamental na garantia da transparência e eficiência na gestão dos recursos públicos, conforme estabelecido pela Constituição Federal de 1988 (BRASIL, 1988). No contexto do Estado do Maranhão, o Tribunal de Contas do Estado (TCE-MA) é responsável por julgar as contas de qualquer pessoa física ou jurídica que utilize, arrecade, guarde, gerencie ou administre recursos públicos estaduais ou municipais.

Um dos instrumentos dessa fiscalização é a Tomada de Contas Especial (TCE) que é o último processo administrativo instaurado quando há indícios de dano ao erário. Este processo caracteriza-se pela complexidade documental e necessidade de análise minuciosa de contratos, comprovantes de gastos, pareceres técnicos e extensa legislação aplicável. Os auditores responsáveis pela elaboração dos relatórios técnicos enfrentam um volume considerável de documentos que demandam análise detalhada, pesquisa normativa e correlação de informações obtidas de múltiplas fontes (CORREGEDORIA GERAL DO ESTADO DO MARANHÃO, 2023).

O processo tradicional de análise e elaboração de relatórios de Tomada de Contas Especial no TCE-MA caracteriza-se pela análise manual de extensos volumes documentais consumindo vários dias de trabalho. Outro ponto que merece destaque é que essa análise está sujeita a limitações humanas na identificação de padrões e correlações, e pode resultar em inconsistências na padronização dos relatórios produzidos. Essas limitações impactam diretamente a eficiência do controle externo e na apuração de irregularidades no uso de recursos públicos.

Os recentes avanços em Inteligência Artificial, particularmente no campo do Processamento de Linguagem Natural (PLN) e das *Large Language Models* (LLMs), apresentam oportunidades promissoras para otimização de processos que envolvem análise intensiva de documentos. Tecnologias como *Retrieval-Augmented Generation* (RAG) permitem que sistemas de IA acessem bases de conhecimento específicas, combinando capacidades generativas com informações contextuais precisas. O avanço dessas tecnologias viabiliza o desenvolvimento de agentes inteligentes capazes de auxiliar em tarefas complexas de análise documental.

Neste trabalho será abordado o desenvolvimento de um agente inteligente baseado em Processamento de Linguagem Natural e *Large Language Models* para auxiliar os auditores do Tribunal de Contas do Estado do Maranhão na análise de documentos e elaboração de relatórios técnicos no processo de Tomada de Contas Especial. Também

serão apresentados todos os conceitos fundamentais necessários para compreensão da solução desenvolvida, assim como teste e avaliação da aplicação em uso real dos auditores do TCE-MA.

1.1 Objetivos

Desenvolver um agente inteligente usando técnicas de Processamento de Linguagem Natural e *Large Language Models* para auxiliar os auditores do Tribunal de Contas do Estado do Maranhão na análise de documentos e elaboração de relatórios técnicos no processo de Tomada de Contas Especial.

1.1.1 Objetivos Específicos

- Analisar o processo atual de elaboração de relatórios de Tomada de Contas Especial no TCE-MA, identificando as principais oportunidades de otimização;
- Analisar as principais tecnologias de Inteligência Artificial, Processamento de Linguagem Natural e *Large Language Models*;
- Desenvolver um agente inteligente capaz de auxiliar os auditores fiscais do TCE-MA na elaboração de relatórios técnicos no processo de Tomada de Contas Especial;
- Fazer uma avaliação da solução desenvolvida.

1.2 Organização do Trabalho

Este trabalho está organizado em quatro capítulos. Além da introdução, o capítulo 2 abordará a fundamentação teórica necessária para a construção do Agente Inteligente. O capítulo 3 fará a descrição da metodologia proposta e da avaliação dos resultados e por fim, o capítulo 4 apresentará as considerações finais e sugestões de trabalho futuro.

2 Fundamentação Teórica

Este Capítulo apresenta os conceitos e fundamentos teóricos utilizados no desenvolvimento deste trabalho. Inicialmente, aborda-se o processo de Tomada de Contas Especial e o papel do Tribunal de Contas. Em seguida, são apresentadas as tecnologias empregadas no desenvolvimento da solução, incluindo Processamento de Linguagem Natural (PLN), *Large Language Models* (LLMs), tecnologias de armazenamento de dados e ferramentas de integração de sistemas.

2.1 O Tribunal de Contas e a Tomada de Contas Especial

A fiscalização do uso de recursos públicos constitui um elemento fundamental do Estado Democrático de Direito. Essa atividade garante que os princípios de legalidade, transparência, moralidade e eficiência sejam respeitados na administração pública, conforme estabelecido no artigo 37 da Constituição Federal de 1988 (BRASIL, 1988). Para assegurar essa fiscalização, foram criados mecanismos de controle que impedem o uso inadequado de verbas públicas e garantem o equilíbrio entre os poderes Executivo, Legislativo e Judiciário.

O Tribunal de Contas da União (TCU) e seus equivalentes estaduais e municipais desempenham papel central nesse sistema de controle. A Constituição Federal de 1988 define o TCU como órgão independente, com autonomia orçamentária e administrativa, responsável por auxiliar tecnicamente o Poder Legislativo no exercício do controle externo (BRASIL, 1988). Embora não integre qualquer dos poderes, o referido tribunal possui competências próprias estabelecidas nos artigos 71 e 73 da Constituição.

No Estado do Maranhão, o Tribunal de Contas do Estado (TCE-MA) exerce funções similares. A Constituição Estadual, em seu artigo 50, parágrafo único (MARANHÃO, 1989), e a Lei Estadual nº 8.258/2005, em seu artigo 7º, inciso VII (MARANHÃO, 2005), estabelecem sua competência para julgar as contas de qualquer pessoa física ou jurídica que utilize, arrecade, guarde, gerencie ou administre recursos públicos estaduais ou municipais. Isso inclui recursos repassados através de convênios, acordos, ajustes e instrumentos similares.

A Tomada de Contas Especial (TCE) é um processo administrativo formal instaurado quando há indícios de dano ao erário público. Caracteriza-se como procedimento excepcional, com rito próprio, destinado a apurar responsabilidades, quantificar danos e obter o ressarcimento dos valores devidos (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2017b). O processo busca identificar os responsáveis pelo dano e garantir a recuperação dos recursos públicos desviados ou mal utilizados.

O TCE-MA possui competência para instaurar o processo de Tomada de Contas Especial de ofício, conforme previsto no artigo 13, §1º, da Lei Estadual nº 8.258/2005 (MARANHÃO, 2005) e no artigo 5º, §2º, inciso I, da Instrução Normativa TCE/MA nº 50/2017 (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2017b). Além da instauração, o tribunal também é responsável pela análise e julgamento desses processos, exercendo papel fundamental na proteção do patrimônio público.

Para padronizar e otimizar a tramitação desses processos, a Corregedoria Geral do Estado do Maranhão desenvolveu um manual que especifica todos os documentos necessários para o julgamento (CORREGEDORIA GERAL DO ESTADO DO MARANHÃO, 2023). Entre os documentos exigidos estão notas de empenho, ordens bancárias, pareceres jurídicos e financeiros, relatórios de auditoria e informações detalhadas fornecidas pelos órgãos de controle interno. Essa documentação extensa e complexa requer análise técnica especializada para identificação de irregularidades e quantificação de danos.

O volume e a complexidade dos documentos envolvidos na Tomada de Contas Especial representam um desafio significativo para os auditores. A análise manual desses materiais demanda tempo considerável e está sujeita a limitações humanas na identificação de padrões e correlações entre diferentes documentos. Nesse contexto, o desenvolvimento de soluções baseadas em Processamento de Linguagem Natural e *Large Language Models* apresenta-se como alternativa viável para otimizar a análise documental, aumentar a precisão na identificação de irregularidades e acelerar o processo de controle externo.

2.2 Conceitos Computacionais e Tecnologias Utilizadas

2.2.1 Agentes de Inteligência Artificial

Um agente como uma entidade que interage com seu ambiente, percebendo informações por meio de sensores e executando ações por meio de atuadores. Essa visão estabelece o ciclo fundamental de percepção-ação, caracterizando os agentes de IA como sistemas capazes de realizar tarefas com objetivos específicos de forma autônoma (RUSSELL; NORVIG, 2021).

Essa conceituação pode ser expandida ao identificar quatro propriedades essenciais dos agentes inteligentes: autonomia, reatividade, proatividade e habilidade social. A autonomia permite que operem sem intervenção humana direta, tomando decisões baseadas em seu estado interno e percepções. A habilidade social possibilita interação efetiva com outros agentes e humanos através de linguagens de comunicação apropriadas. A reatividade garante percepção contínua do ambiente e resposta oportuna às mudanças. Por fim, a proatividade manifesta-se no comportamento orientado a objetivos, permitindo que os agentes tomem iniciativas para alcançar suas metas (WOOLDRIDGE; JENNINGS, 1995).

Essas propriedades fundamentais têm sido potencializadas pelos avanços recentes em *Large Language Models* (LLMs), tecnologia que será detalhada na Seção 2.2.3. As LLMs revolucionaram a implementação de agentes ao fornecer capacidades avançadas de raciocínio e compreensão contextual, permitindo que as propriedades de autonomia e habilidade social sejam expressas de forma mais natural e efetiva. Atualmente, as LLMs servem como componentes centrais dos agentes modernos, sendo aprimoradas com módulos especializados para memória persistente, planejamento de tarefas complexas, utilização dinâmica de ferramentas e interação sofisticada com o ambiente.

2.2.2 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN), do inglês *Natural Language Processing*, é uma área da computação dedicada à compreensão e manipulação da linguagem humana por sistemas computacionais. Essa disciplina combina um conjunto de métodos, como conhecimentos de linguística, ciência da computação e matemática para desenvolver algoritmos capazes de interpretar, analisar e gerar texto automaticamente (EISENSTEIN, 2018).

O PLN utiliza diversas técnicas fundamentais da computação, incluindo expressões regulares, tokenização, *embeddings*, *machine learning*, reconhecimento óptico de caracteres, redes neurais e análise estatística. Essas técnicas permitem o desenvolvimento de aplicações como tradução automática, sistemas de perguntas e respostas, recuperação de informações, classificação de textos, *Retrieval-Augmented Generation* (RAG), *chatbots*, sistemas de diálogo e reconhecimento de fala (JURAFSKY; MARTIN, 2025).

O Reconhecimento Óptico de Caracteres, do inglês *Optical Character Recognition* (OCR), é uma técnica fundamental no processamento de documentos pois consegue digitalizar texto impresso e imagens para que possam ser manipulados por máquinas (ISLAM; ISLAM; NOOR, 2017). No contexto de órgãos públicos, o OCR torna-se indispensável uma vez que para realizar análises com base em técnicas de PLN torna-se necessário que os dados estejam em formato textual e grande parte dos documentos armazenados e escaneados por essas entidades, são salvos como imagens.

A tokenização é outra uma etapa fundamental do pré-processamento em PLN. Ela divide o texto em unidades menores chamadas *tokens*, que podem ser palavras, caracteres ou frases, dependendo da estratégia adotada. A qualidade da tokenização impacta diretamente o desempenho das técnicas subsequentes, pois determina como o texto será representado numericamente e processado pelos algoritmos (JURAFSKY; MARTIN, 2025).

Os *embeddings* constituem uma das técnicas mais importantes do PLN moderno. Eles representam palavras, frases ou documentos como vetores numéricos em espaços multidimensionais, permitindo que algoritmos de *machine learning* processem texto

matematicamente. Nessa representação vetorial, elementos linguísticos com significados similares ficam próximos no espaço matemático. Por exemplo, palavras como "gato"e "felino"terão vetores próximos, enquanto "gato"e "matemática"estarão distantes. Essa propriedade é fundamental para sistemas RAG, onde a similaridade vetorial determina quais informações são recuperadas para auxiliar na geração de respostas.

O desenvolvimento de arquiteturas baseadas nessas técnicas permitiu criar ferramentas capazes de processar e gerar texto de forma cada vez mais natural. As primeiras arquiteturas importantes foram as Redes Neurais Recorrentes (RNNs) e *Long Short-Term Memory* (LSTMs), que processavam texto sequencialmente. Embora representassem avanços significativos, essas arquiteturas enfrentavam limitações com textos longos devido a problemas de memória e propagação de gradiente (JURAFSKY; MARTIN, 2025).

Um marco revolucionário no PLN foi a introdução da arquitetura *Transformer* (VASWANI et al., 2017) no desenvolvimentode de modelos de . Diferentemente das arquiteturas anteriores que processavam texto sequencialmente, o *Transformer* analisa todas as palavras simultaneamente através do mecanismo de *self-attention*. Este mecanismo funciona como um sistema de foco inteligente, permitindo que o modelo determine quais palavras são mais relevantes para compreender cada parte do texto, independentemente da distância entre elas na frase ou texto.

Por exemplo, na frase "O gato, que estava assustado no telhado, miou alto", um modelo baseado em *Transformer* consegue conectar instantaneamente "gato"e "miou", mesmo com várias palavras entre eles. Essa capacidade de capturar relações de longo alcance tornou possível o processamento eficiente de textos extensos e complexos.

A combinação da arquitetura *Transformer* com treinamento em grandes volumes de dados resultou em modelos capazes de gerar texto com precisão e coerência sem precedentes. Esses avanços levaram ao desenvolvimento dos *Large Language Models* (LLMs), que representam o estado da arte em processamento de linguagem natural (JURAFSKY; MARTIN, 2025).

2.2.3 *Large Language Models* (LLMs)

Os *Large Language Models* (LLMs) são modelos de linguagem caracterizados por sua capacidade de compreender e gerar texto coerente e contextualmente relevante. Treinados em vastos volumes de dados textuais de diversas áreas do conhecimento, esses modelos podem realizar múltiplas tarefas sem necessidade de treinamento específico para cada aplicação (JURAFSKY; MARTIN, 2025).

Os LLMs podem ser classificados segundo diferentes critérios. Em relação ao tamanho, variam desde modelos compactos com alguns bilhões de parâmetros até modelos massivos com centenas de bilhões ou trilhões de parâmetros. Modelos maiores geralmente

apresentam capacidades mais avançadas, incluindo melhor compreensão contextual, raciocínio mais complexo e maior criatividade na geração de texto.

Quanto à disponibilização, existem modelos proprietários e *open source*. Modelos proprietários como *ChatGPT* (OPENAI, 2025), *Gemini* (GOOGLE, 2025b) e *Claude* (ANTHROPIC, 2025) são mantidos por empresas privadas e acessados via APIs ou interfaces *web* próprias. Modelos *open source* como *Llama* (META, 2025b) e *Mistral* (MISTRAL AI, 2025) têm código aberto, permitindo modificações e hospedagem em infraestrutura própria.

Há também classificação por especialização. Modelos generalistas podem realizar diversas tarefas de linguagem natural, enquanto modelos especializados são otimizados para domínios específicos como programação, medicina ou análise financeira. Alguns modelos são multimodais, processando não apenas texto, mas também imagens, áudio e outros tipos de dados.

Atualmente, os principais modelos proprietários oferecem interfaces amigáveis e APIs robustas para integração com aplicações. Eles apresentam desempenho similar em muitas tarefas, com pequenas variações em aspectos específicos. Os modelos *open source* têm ganhado destaque por permitirem personalização completa, embora exijam infraestrutura robusta para hospedagem, resultando em custos operacionais significativos.

Apesar de suas capacidades avançadas, as LLMs apresentam limitações importantes. Seu conhecimento está restrito à data de finalização do treinamento, não podem acessar informações específicas de organizações e ocasionalmente geram informações incorretas, fenômeno conhecido como "alucinação". Para superar essas limitações, é possível aperfeiçoar os resultados a partir de duas técnicas, uma sendo o *fine-tuning*, que consiste em um retreinamento da LLM em cima de dados próprios, e a outra através da técnica de *Retrieval-Augmented Generation* (RAG), que consiste em fornecer um maior contexto baseado em um conjunto de dados próprio para a LLM sem modificá-la (LEWIS et al., 2021).

O *Retrieval-Augmented Generation* (RAG) é uma técnica que combina a capacidade generativa dos LLMs com sistemas de recuperação de informações. Essa abordagem permite que os modelos acessem conhecimento externo atualizado durante a geração de respostas, superando as limitações do conhecimento estático adquirido no treinamento (LEWIS et al., 2021).

O funcionamento do RAG ocorre em duas etapas principais. Na etapa de recuperação (*retrieval*), o sistema busca informações relevantes em uma base de dados externa usando técnicas de busca por similaridade baseadas em *embeddings*. Na etapa de geração (*generation*), as informações recuperadas são fornecidas como contexto adicional para o LLM, que as utiliza para gerar respostas mais precisas e atualizadas.

O funcionamento do RAG ocorre em duas etapas principais. Na etapa de recuperação (*retrieval*), o sistema busca informações relevantes em uma base de dados externa usando técnicas de busca por similaridade baseadas em *embeddings*. Para viabilizar essa busca eficiente, documentos extensos são previamente segmentados em fragmentos menores denominados *chunks*. Esses segmentos são porções de texto com tamanho otimizado, tipicamente entre 200 e 1000 tokens que preservam contexto semântico suficiente para serem compreensíveis isoladamente. Cada *chunk* é convertido em um vetor numérico através de modelos de *embedding*, permitindo que o sistema identifique rapidamente os fragmentos mais relevantes para uma consulta específica através de cálculos de similaridade vetorial. Na etapa de geração (*generation*), as informações recuperadas dos *chunks* mais relevantes são fornecidas como contexto adicional para a LLM, que as utiliza para gerar respostas mais precisas e atualizadas.

A arquitetura *Retrieval-Augmented Generation* pode ser implementada de diferentes formas, desde sistemas simples com documentos estáticos até implementações sofisticadas que integram múltiplas fontes de dados em tempo real. Os componentes essenciais para esse sistema, incluem um processo de vetorização para converter documentos em *embeddings*, uma base de dados vetorial para armazenamento, um mecanismo de recuperação para identificar conteúdo relevante e integração com uma LLM para geração das respostas finais (LEWIS et al., 2021).

As principais vantagens dessa técnica incluem acesso a informações atualizadas sem retreinamento do modelo, capacidade de trabalhar com conhecimento específico de domínios ou organizações, redução significativa de alucinações através do uso de fontes confiáveis e rastreabilidade das fontes utilizadas nas respostas.

Para implementar sistemas com esse processo, é fundamental compreender as tecnologias de gerenciamento e armazenamento de dados que sustentam essas soluções.

2.2.4 Gerenciamento de Dados e Armazenamento de Objetos

A implementação de sistemas RAG e outras aplicações de IA requer infraestrutura robusta de gerenciamento de dados. Essa seção apresenta as principais tecnologias utilizadas, desde bancos de dados relacionais tradicionais até soluções especializadas para dados vetoriais e armazenamento de objetos.

2.2.4.1 Bancos de Dados Relacionais

Os bancos de dados relacionais são predominantes para gerenciamento de dados estruturados. Eles organizam informações em tabelas com relacionamentos definidos através de chaves primárias e estrangeiras (ELMASRI; NAVATHE, 2018). Esses sistemas

garantem integridade dos dados através das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), essenciais para aplicações críticas.

Para implementar e gerenciar esses bancos de dados, foram desenvolvidos *softwares* especializados conhecidos como Sistemas de Gerenciamento de Banco de Dados (SGBDs). Eles oferecem funcionalidades como controle de acesso, processamento de consultas, gerenciamento de transações concorrentes, *backup* e recuperação. Entre as principais soluções do mercado estão Oracle Database (ORACLE, 2025b), Microsoft SQL Server (MICROSOFT, 2025), MySQL (ORACLE, 2025a) e PostgreSQL (POSTGRESQL, 2025).

A comunicação com SGBDs é padronizada através da Structured Query Language (SQL), uma linguagem declarativa para manipulação de dados. O SQL oferece comandos para consultas (SELECT), inserção (INSERT), atualização (UPDATE) e exclusão (DELETE), além de comandos para definição de estruturas (CREATE TABLE, ALTER TABLE). Essa padronização permite que conhecimentos em SQL sejam transferíveis entre diferentes SGBDs.

O TCE-MA adota o PostgreSQL (POSTGRESQL, 2025), que destaca-se como um SGBD *open source* robusto e confiável. Suas principais características incluem conformidade com padrões SQL, ausência de custos de licenciamento, comunidade ativa de desenvolvedores e grande extensibilidade através de tipos de dados, funções e operadores customizados.

O PostgreSQL oferece recursos avançados como suporte nativo para tipos de dados modernos (JSON, XML), diversos tipos de índices para otimização de *performance*, ferramentas de replicação e particionamento para alta disponibilidade e compatibilidade com múltiplas plataformas e linguagens de programação. Essas características o tornam ideal para aplicações que exigem *performance*, segurança e personalização.

A extensibilidade do PostgreSQL é fundamental para sua adaptação a novas tecnologias. Através de extensões, ele pode incorporar funcionalidades além de um sistema relacional tradicional, incluindo suporte para dados vetoriais essenciais para aplicações de IA.

2.2.4.2 Bancos de Dados Vetoriais

Com o avanço das técnicas de PLN, surgiu a necessidade de gerenciar dados baseados em características semânticas, não apenas em valores exatos. Os bancos de dados vetoriais são especializados em armazenar e consultar *embeddings* - representações numéricas de dados em espaços multidimensionais.

Essas representações capturam o significado e relações contextuais dos dados. Itens semanticamente similares ficam "próximos"no espaço vetorial, propriedade crucial

para sistemas RAG. A relevância contextual é determinada pela proximidade vetorial, permitindo recuperação eficiente de informações pertinentes.

Embora bancos relacionais tradicionais não sejam otimizados para operações vetoriais, a extensibilidade do PostgreSQL permite adicionar essa funcionalidade através de extensões especializadas como o *pgvector* (PGVECTOR, 2025).

O *pgvector* (PGVECTOR, 2025) é uma extensão *open source* que transforma o PostgreSQL em um banco de dados vetorial. Ela permite armazenar, indexar e realizar consultas de similaridade em *embeddings* diretamente no PostgreSQL, eliminando a necessidade de uma plataforma vetorial separada.

Com o *pgvector*, é possível utilizar operadores para calcular distâncias entre vetores e criar índices otimizados para busca de dados mais próximos. Essa funcionalidade é essencial para aplicações que dependem de análise de similaridade, como sistemas RAG, recomendação e busca semântica.

A integração do *pgvector* com PostgreSQL combina a robustez de um SGBD relacional maduro com capacidades avançadas de processamento vetorial, simplificando a arquitetura de aplicações baseadas em processamento de linguagem natural.

2.2.4.3 Armazenamento de Objetos

O armazenamento de objetos é projetado para gerenciar grandes volumes de dados não estruturados como documentos, imagens, vídeos e arquivos de *log*. Diferentemente de sistemas de arquivos hierárquicos, organiza dados como objetos únicos em estrutura plana, identificados por chaves únicas e acompanhados de metadados.

Os metadados incluem informações como tamanho, datas de criação e modificação, tipo de conteúdo, versão e etiquetas personalizadas. Isso permite busca e gerenciamento eficiente sem depender de hierarquias de diretórios. Os objetos são organizados em *containers* ou *buckets*, acessíveis via URLs HTTP através de APIs REST.

O mercado oferece diversas soluções, lideradas por provedores de nuvem como *Amazon S3* e *Google Cloud Storage*. Existem também alternativas *on-premises* e *open source* como *MinIO*. A escolha depende de fatores como localização dos dados, custos operacionais e necessidades de integração.

O TCE-MA utiliza o *Amazon S3*, que é a solução de armazenamento de objetos da *Amazon Web Services* (AWS). Projetado para oferecer durabilidade, disponibilidade e escalabilidade praticamente ilimitadas, permite armazenar e recuperar qualquer volume de dados via *web*.

O S3 organiza dados em objetos dentro de *buckets*, com cada objeto identificado por chave única. Essa estrutura plana otimiza o acesso direto aos dados. Como API REST,

todas as operações são realizadas via requisições HTTP, facilitando integração com diversas aplicações e serviços.

2.2.5 Integração de Sistemas e Automação

O desenvolvimento de agentes inteligentes requer um ambiente tecnológico que possibilite integrar todas as técnicas e conceitos discutidos anteriormente. Dentre as principais tecnologias atuais, que permitem essa implementação, destaca-se o Python (PYTHON SOFTWARE FOUNDATION, 2025), que é a linguagem predominante nesse domínio por disponibilizar bibliotecas e *frameworks* robustos para IA e PLN, além de ser de fácil aprendizado e possuir grande comunidade. Entre essas bibliotecas, destacam-se:

- Pandas (PANDAS, 2025): biblioteca essencial para manipulação de dados;
- Requests (REITZ, 2025): biblioteca para comunicação HTTP;
- Skicit-learn (SCIKIT-LEARN, 2025): biblioteca que reune ferramentas voltadas para aprendizado de máquina e analise de dados;
- FastAPI (MONTAñO, 2018): *framework back-end* para construção de APIs REST;
- LangChain (LANGCHAIN, 2025) e LlamaIndex (LLAMAINDEX, 2025): *frameworks* especializados para construção de sistemas RAG que fornecem componentes pré-construídos para processamento de documentos, vetorização, recuperação e orquestração de consultas;

A comunicação entre sistemas inteligentes utiliza principalmente APIs com o padrão de arquitetura Representational State Transfer (REST) (FIELDING, 2000), que permitem integração padronizada via HTTP independentemente da linguagem ou plataforma. Essa padronização é fundamental para arquiteturas distribuídas que integram múltiplos provedores de IA, bases de dados e serviços externos.

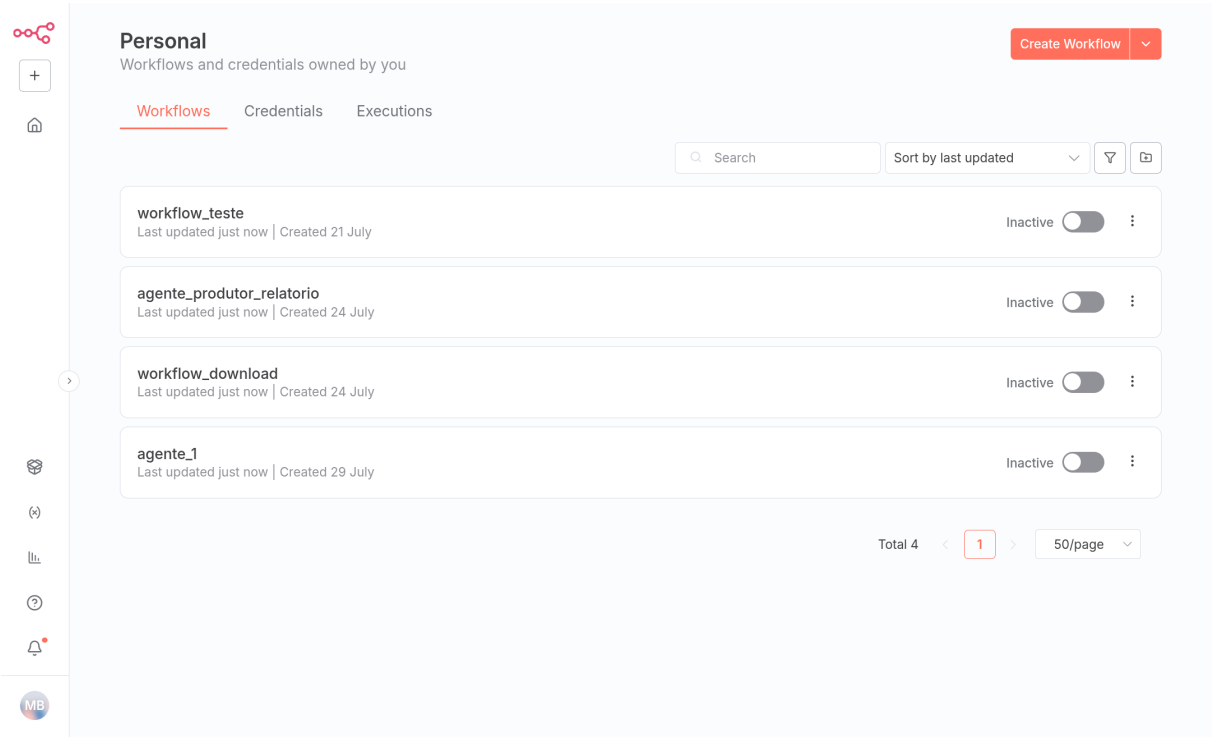
As implementações tradicionais baseadas em código oferecem controle total mas apresentam desafios relacionados à complexidade de desenvolvimento, manutenção de autenticações, tratamento de erros e monitoramento. A partir dessas dificuldades surgiram alternativas que democratizam o desenvolvimento de integrações através de plataformas *no-code* e *low-code* como N8N (N8N, 2025) e Make (CELONIS, INC., 2025) com interfaces visuais intuitivas e de fácil configuração.

O N8N (N8N, 2025) é uma dessas plataformas e se destina a automação de fluxos de trabalhos (*workflows*) baseada em nós (*nodes*). Inicialmente foi desenvolvido para automação de processos mas expandiu seu escopo para desenvolvimento de *chatbots*, análise de dados e implementação de agentes de IA. Ademais, oferece versão *open source* para instalação local e versão paga em nuvem com recursos empresariais adicionais.

Esse sistema funciona através de fluxos visuais compostos por nós interconectados. Cada nó representa uma ação específica: requisições HTTP, processamento JSON, interação com bancos de dados ou comunicação com serviços externos. Para funções customizadas, disponibiliza nós de código que executam *scripts* Python ou JavaScript diretamente no fluxo.

A interface do N8N é intuitiva, permitindo criar *workflows* facilmente a partir do botão *Create Workflow* que pode ser visto no canto superior direito da tela inicial do sistema na Figura 1. Os *workflows* são criados a partir da união de nós que são facilmente adicionados a partir de botões com o símbolo "+". O sistema disponibiliza nós com integrações prontas para os mais diversos objetivos como pode ser visto na Figura 2. Após a escolha do tipo, cada nó é configurado através de formulários que definem parâmetros, autenticações e lógica condicional.

Figura 1 – Tela inicial do N8N.

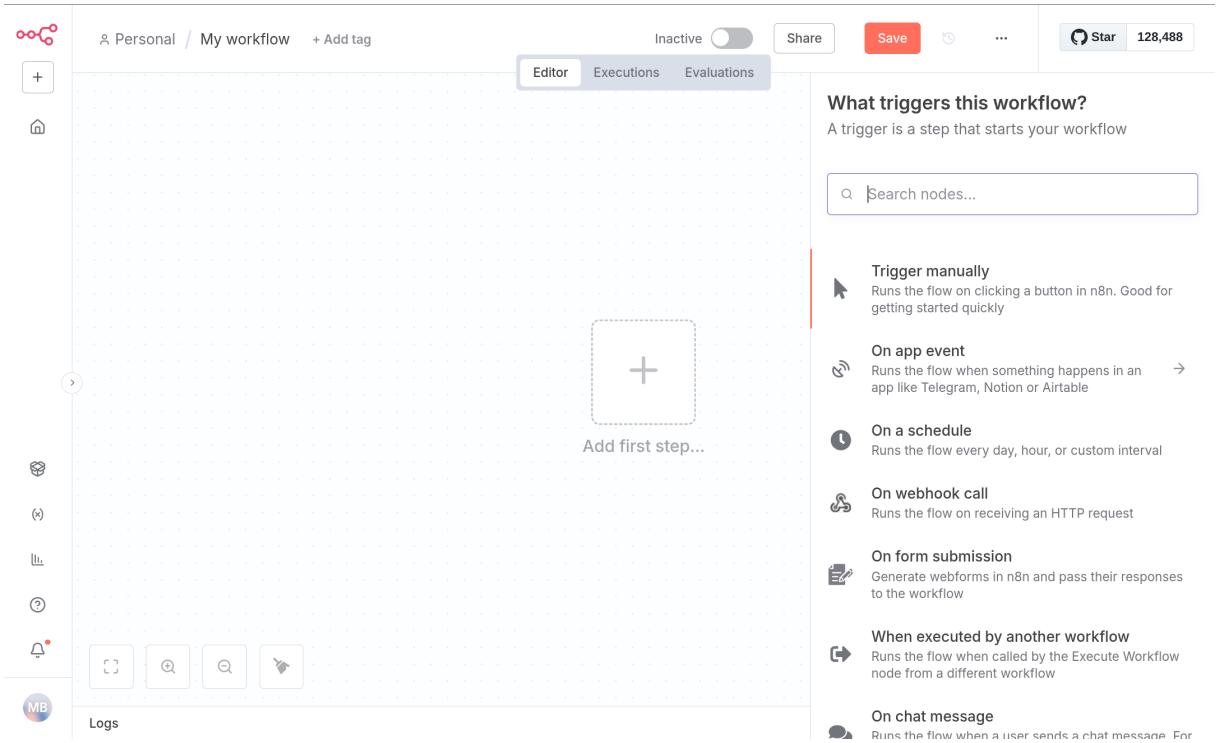


Fonte: acervo do autor.

Os *workflows* podem ser iniciados manualmente, por agendamento ou automaticamente via nós do tipo webhook. A arquitetura modular permite que exista fluxos aninhados, promovendo reutilização e organização hierárquica. Os fluxos podem ser expostos como *endpoints* HTTP, transformando o N8N em um *back-end* com construção visual para aplicações.

O sistema de gerenciamento de credenciais centraliza autenticações para diferentes serviços como ser visto na Figura 3, incluindo chaves de API e *tokens OAuth*. O N8N

Figura 2 – Criação de nó no N8N.



Fonte: acervo do autor.

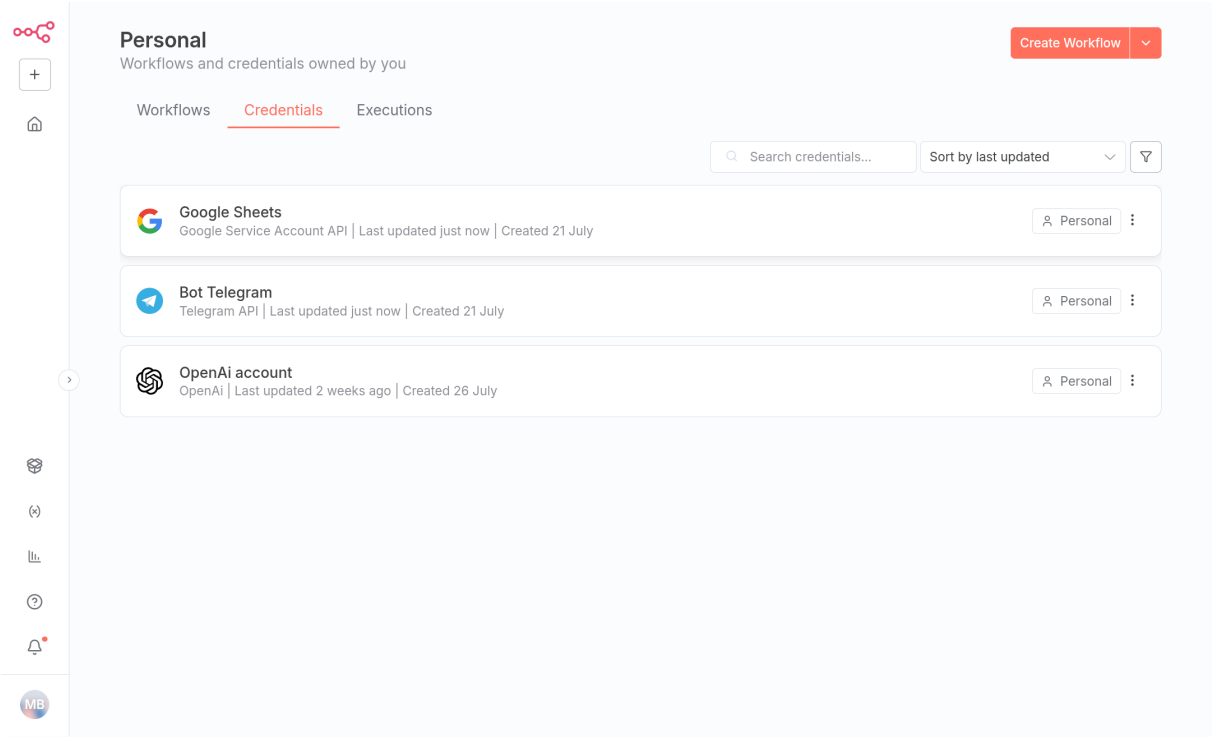
oferece conectores nativos para provedores de LLMs como OpenAI, Anthropic e Google AI, além de integração com bancos vetoriais (pgvector), armazenamento de objetos (AWS S3, Google Drive) e ferramentas de comunicação (Telegram, WhatsApp).

2.2.6 Interface de Usuário

Agentes de IA conversacionais podem ter interfaces através de diferentes tecnologias. Plataformas de mensagens como WhatsApp (WHATSAPP, 2025) e Telegram (TELEGRAM, 2025b) oferecem alcance imediato através de bases de usuários estabelecidas. O WhatsApp Business API (WHATSAPP BUSINESS, 2025) permite criar *chatbots*, mas apresenta restrições de personalização e custos com base no uso . O Telegram oferece uma API mais flexível com suporte a comandos customizados e processamento de mídias diversas além de opção com uso gratuito (TELEGRAM, 2025a).

Outra forma de implementar esses agentes é através de interfaces *web* próprias que proporcionam controle total sobre a experiência do usuário, permitindo customização completa e implementação de funcionalidades específicas sem limitações de plataformas externas. Para desenvolvimento *web*, utiliza-se *frameworks* baseados na linguagem JavaScript (ECMA INTERNATIONAL, 2025) como React (META PLATFORMS, INC., 2025), Vue.js (VUE.JS, 2025) e Angular (GOOGLE, 2025a), que oferecem componentes reutilizáveis e renderização otimizada.

Figura 3 – Gerenciamento de credenciais no N8N.



Fonte: acervo do autor.

A escolha de tecnologias adequadas para desenvolvimento *front-end* deve considerar fatores como performance de renderização, facilidades para consumo de APIs REST, capacidades de otimização para mecanismos de busca, além de disponibilidade de ferramentas que acelerem o desenvolvimento como bibliotecas de componentes. Nesse contexto, *frameworks* baseados em React oferecem vantagens significativas devido seu ambiente maduro, extensa documentação e disponibilidade de bibliotecas especializadas.

React é uma biblioteca JavaScript desenvolvida em 2013 pela equipe de desenvolvedores da rede social Facebook (META, 2025a). Foi elaborado para construção de interfaces baseadas em componentes reutilizáveis. Utiliza o *Virtual Document Object Model* (DOM) para otimizar atualizações da interface, resultando em desempenho superior, além disso também trouxe o JSX (*JavaScript XML*) que permite escrever elementos HTML diretamente no código JavaScript, além do sistema de *hooks*, que simplifica o gerenciamento de estado em componentes funcionais.

O Next.js é um *framework* React desenvolvido pela Vercel que adiciona funcionalidades avançadas como renderização do lado do servidor, do inglês *Server Side Rendering* (SSR), geração de sites estáticos, do inglês Static Site Generation (SSG) e *API routes* integradas. O SSR melhora o *Search Engine Optimization* (SEO) e tempo de carregamento inicial renderizando páginas no servidor. O SSG gera páginas HTML estáticas durante o *build* para maior velocidade. O *API routes* cria *endpoints* HTTP

automaticamente baseados na estrutura de pastas do projeto, agilizando o desenvolvimento do roteamento de APIs. Essas funcionalidades abstraem configurações complexas do React, proporcionando uma experiência de desenvolvimento otimizada com *hot reload* e otimizações de *performance* (NEXT.JS, 2025).

A autenticação de usuários em aplicações *web* pode ser implementada através de diferentes estratégias, desde simples *JSON Web Tokens* (JWT) até sistemas completos de gerenciamento de identidade. O Next.js oferece integração com bibliotecas como *NextAuth.js* (AUTH.JS, 2025), que simplifica a implementação de autenticação com suporte a provedores externos como Google, Facebook e GitHub, além de autenticação baseada em credenciais.

Além de extensões, também existem sistemas completos totalmente dedicados a autenticação. Um deles é o Keycloak (KEYCLOAK, 2025), sistema *open source* responsável pelo gerenciamento de identidade e acesso que fornece uma camada de segurança unificada capaz de centralizar autenticação e autorização para múltiplas aplicações e serviços.

O Keycloak implementa os principais padrões de segurança da indústria, incluindo OAuth 2.0, que é utilizado pelo NextAuth.js. Essa conformidade com padrões de autenticação e disponibilidade de uma API REST para comunicação permite integração transparente com aplicações modernas e sistemas legados.

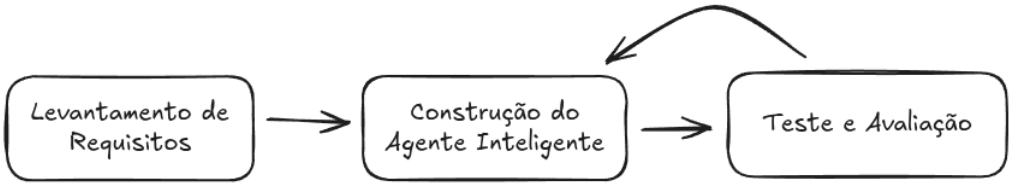
A arquitetura baseada em um serviço dedicado a autenticação oferece vantagens significativas para ambientes governamentais: auditoria completa de acessos, conformidade com políticas de segurança institucionais, gestão centralizada de credenciais e possibilidade de integração futura com outros sistemas do tribunal através do mesmo provedor de identidade.

A combinação de todas essas tecnologias citadas nessa fundamentação, desde o processamento de linguagem natural até as interfaces de usuário seguras, forma a base tecnológica para o desenvolvimento do agente inteligente proposto que visa auxiliar o TCE-MA na análise de processos de Tomada de Contas Especial. O próximo Capítulo abordará a metodologia utilizada na construção desse agente.

3 Metodologia

Neste Capítulo são descritos os procedimentos necessários para a construção do agente inteligente que visa auxiliar os auditores do TCE-MA no processo de Tomada de Contas Especial, mais especificamente na elaboração de relatórios sobre irregularidades do uso do dinheiro público. A Figura 4 apresenta as etapas da metodologia proposta, que são: Levantamento de Requisitos, Construção do Agente Inteligente, Teste e Avaliação.

Figura 4 – Etapas da metodologia proposta.



Fonte: acervo do autor.

3.1 Levantamento de Requisitos

Inicialmente, foi realizada uma análise detalhada do processo atual de elaboração de relatórios de Tomada de Contas Especial no TCE-MA. Essa análise teve como objetivo compreender o fluxo de trabalho dos auditores, identificar os principais gargalos operacionais e definir os requisitos para o desenvolvimento do agente inteligente.

Para essa análise, foram realizadas reuniões com a equipe de tecnologia da informação do TCE-MA e, principalmente, com um auditor fiscal que já utilizava grandes modelos de linguagem (LLMs) em suas atividades diárias.

O processo identificado consiste em quatro etapas principais executadas sequencialmente pelos auditores: recebimento e análise preliminar do processo, análise documental detalhada, pesquisa da legislação aplicável e elaboração do relatório técnico.

A primeira etapa envolve o recebimento e análise preliminar do processo, em que o auditor recebe documentos e realiza uma leitura inicial para compreender a natureza da irregularidade investigada. Durante essa análise, identifica-se o tipo de problema, verifica-se a competência do tribunal e avalia-se a documentação disponível.

A segunda etapa consiste na análise documental detalhada, na qual o auditor examina individualmente cada documento constante nos autos, realizando leituras e releituras de contratos, documentos oficiais, comprovantes de gastos e ofícios. Durante essa análise minuciosa, o auditor extrai e anota as informações relevantes como nomes dos

envolvidos, valores, comprovantes de pagamentos e datas, organizando os dados coletados manualmente e desenvolvendo controles próprios para acompanhar o progresso da análise.

Em seguida, o auditor realiza a pesquisa normativa para identificar e analisar toda a legislação aplicável ao caso. Essa pesquisa abrange múltiplas fontes normativas, tais como:

- Constituição do Estado do Maranhão, arts. 51, inciso II, e 172, inciso II (MARANHÃO, 1989);
- Lei Orgânica do Tribunal de Contas do Estado do Maranhão, art. 13 (MARANHÃO, 2005);
- Regimento Interno do Tribunal de Contas do Estado do Maranhão, arts. 174, 175 e 177 (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2005);
- Instrução Normativa TCE/MA n° 50, de 30 de agosto de 2017 (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2017b);
- Decisão Normativa TCE n° 28, de 6 de dezembro de 2017, que altera a IN TCE/MA 50/2017 e dá outras providências (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2017a);
- Decisão Normativa TCE n° 38, de 21 de outubro de 2020, que estabelece em R$ 100.000,00 o valor histórico do dano causado ao erário a partir do qual o controlado fica obrigado a enviar ao Tribunal de Contas do Estado do Maranhão a tomada de contas especial respectiva (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2020);
- Resolução TCE/MA n° 383, de 26 de abril de 2023, que regulamenta, no âmbito do Tribunal de Contas do Estado do Maranhão, a prescrição para o exercício das pretensões punitiva e de ressarcimento (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2023);
- Resolução TCE/MA n° 406, de 14 de agosto de 2024, que altera a Resolução TCE/MA n° 383, de 26 de abril de 2023 (TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO, 2024).

A etapa final desse processo consiste na elaboração do relatório técnico, que representa a consolidação de todo o trabalho de análise realizado. O auditor estrutura um documento que atende aos padrões técnicos e normativos estabelecidos, articulando os fatos apurados, a legislação aplicável e as conclusões alcançadas. Este documento

é estabelcido no Manual de Procedimentos para Tomada de Contas Especial do TCE-MA (CORREGEDORIA GERAL DO ESTADO DO MARANHÃO, 2023), que orienta a elaboração de relatórios técnicos e estabelece os requisitos mínimos para sua estruturação.

Durante a análise do processo, observou-se como um auditor do TCE-MA já explorava o potencial dos grandes modelos de linguagem (LLMs) disponíveis comercialmente para auxiliar em suas tarefas diárias. Este profissional desenvolveu uma abordagem própria que consistia em elaborar prompts detalhados e submeter essas informações, juntamente com os arquivos do processo, a LLMs (GPT e Gemini). O retorno dessas interações gerava um checklist de informações cruciais para a tomada de contas especial, que ele utilizava para preencher a mesma planilha Excel do processo padrão. Somente após a validação desse checklist, o auditor solicitava à LLM a geração de um protótipo do relatório com base em um modelo também encaminhado para a IA, que servia como ponto de partida para sua revisão final. Essa experiência demonstrou o potencial transformador da inteligência artificial no fluxo de trabalho dos auditores.

A solução foi então concebida tendo como base esse processo já testado pelo auditor com as LLMs. A ideia inicial foi automatizar esse processo através da infraestrutura tecnológica do próprio Tribunal, buscando as informações do processo diretamente de sua base de dados, além de fornecer uma ferramenta de chatbot administrada pela própria instituição, eliminando a necessidade do uso de ferramentas externas para consumo das LLMs.

Para o desenvolvimento da solução, as tecnologias foram selecionadas considerando as práticas atuais do mercado, a compatibilidade com a infraestrutura existente e a proficiência técnica já estabelecida no referido Tribunal.

Para a interface de usuário, já havia um *hub* de agentes inteligentes desenvolvido pelo TCE-MA, que utiliza o *framework* Nextjs com autenticação por parte do Keycloak (fundamentados na Seção 2.2.6). Dessa forma decidiu-se aproveitar essa interface e apenas integrar o agente nela, permitindo que os auditores acessassem o agente inteligente diretamente através da interface *web* já existente.

Para a orquestração das interações com os grandes modelos de linguagem, foram analisadas alternativas de desenvolvimento *back-end* com Python, incluindo Langchain e LlamaIndex, assim como a ferramenta N8N (discutidos na Seção 2.2.5). Optou-se pelo N8N, devido à sua interface visual extremamente intuitiva e facilidade de uso e, além disso, o Tribunal já utilizar essa ferramenta. O N8N já disponibiliza diversas integrações nativas para automações e desenvolvimento com LLMs, eliminando a necessidade de desenvolvimento customizado para essas funcionalidades específicas. Além disso, por ser uma solução open-source que pode ser hospedada internamente, garante maior controle sobre a infraestrutura.

No contexto atual, diversas opções de LLMs estão disponíveis no mercado, incluindo GPT-4 da OpenAI, Claude da Anthropic, Gemini do Google e modelos open-source como Llama e Deepseek (apresentados na Seção 2.2.3). A escolha feita para o projeto foi o Gemini devido o TCE-MA já possuir contrato ativo com o Google Workspace para email institucional e ferramentas de produtividade, o que facilitou o acesso ao modelo e acelerou o processo de implementação.

Para o armazenamento e gerenciamento dos dados, apesar de existirem alternativas eficientes como MySQL ou SQLServer, a escolha feita foi o PostgreSQL (Seção 2.2.4.1). Essa escolha foi motivada pela infraestrutura já consolidada no Tribunal, que possui ampla base de dados histórica nessa tecnologia, além de sua robustez e confiabilidade comprovadas. Para as funcionalidades de busca semântica e armazenamento de embeddings, optou-se pela extensão pgvector (Seção 2.2.4.2), que permite integrar capacidades vetoriais ao PostgreSQL existente, evitando a necessidade de gerenciar sistemas de banco de dados adicionais.

Para o armazenamento de documentos e arquivos do sistema, foram consideradas soluções de *cloud storage*, como AWS S3 e Google Drive (Seção 2.2.4.3). A opção pelo AWS S3 baseou-se na experiência prévia do Tribunal com essa tecnologia em outros projetos, garantindo aproveitamento do conhecimento técnico existente.

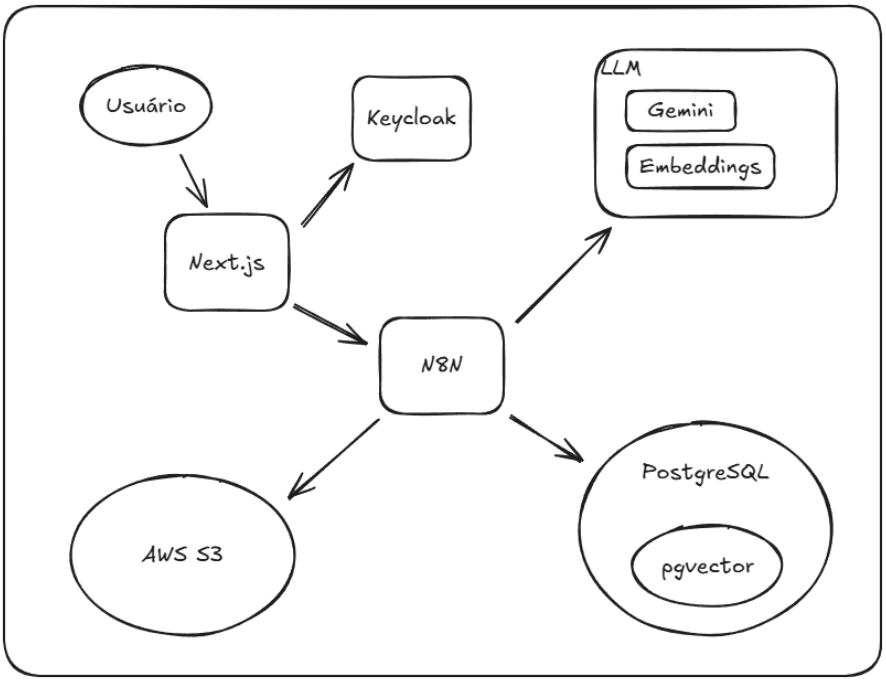
A arquitetura do agente inteligente proposto, conforme ilustrado na Figura 5, inicia-se com a interação do usuário que acessa o *front-end* (Next.js), onde a autenticação é realizada através do Keycloak. Após a validação, o usuário pode inserir informações e documentos relevantes ao caso. O fluxo de trabalho é orquestrado pelo N8N, que atua como uma API, sendo responsável por gerenciar e direcionar os dados, salvando os documentos na AWS S3, enquanto as informações estruturadas e suas representações vetoriais *(embeddings)* são persistidas em um banco de dados PostgreSQL com a extensão pgvector. Por fim, o N8N interage com as LLMs do Google (Gemini), fornecendo o contexto necessário para que o agente de IA analise as informações e gere insights que auxiliem o auditor.

3.2 Construção do Agente Inteligente

A segunda etapa da metodologia proposta iniciou-se pelo desenvolvimento do núcleo de inteligência com integração dos sistemas existentes e bases de dados, posteriormente para a interface de usuário.

Primeiramente, configurou-se as credenciais dos serviços que seriam utilizados a partir do N8N que utiliza credenciais centralizadas, configuradas para garantir segurança e facilitar o desenvolvimento e manutenção. As credenciais são cadastradas a partir do menu "*Credentials*", onde são armazenadas de forma segura e acessíveis a todos os

Figura 5 – Arquitetura do agente inteligente proposto.



Fonte: acervo do autor.

workflows da conta conectada ao orquestrador. As principais credenciais configuradas para o desenvolvimento desse agente foram a AWS, Google Gemini API e PostgreSQL.

Com as credenciais cadastradas, tornou-se possível a criação dos "nós"de integração dessas tecnologias citadas que o N8N disponibiliza. Após isso, iniciou-se a criação de fluxos responsáveis pela gerência de documentos e pelo agente. A própria interface do N8N facilitou significativamente o desenvolvimento e a validação dessas funcionalidades. A seguir, serão apresentados os principais *workflows* desenvolvidos, começando pela manipulaçao de arquivos, passando para o agente e por fim a interface de usuário, conforme está definido na Figura 6.

Figura 6 – *Ordem da construção do agente.*

Fonte: acervo do autor.

3.2.1 Implementação da Manipulação de Arquivos

Para as funcionalidades de manipulação de arquivos enviados pelo usuário, foram implementado *workflows* configurados como *endpoints* HTTP. Essa implementação consiste em realizar as operações de upload, download, listar e deletar, dessa forma gerenciando

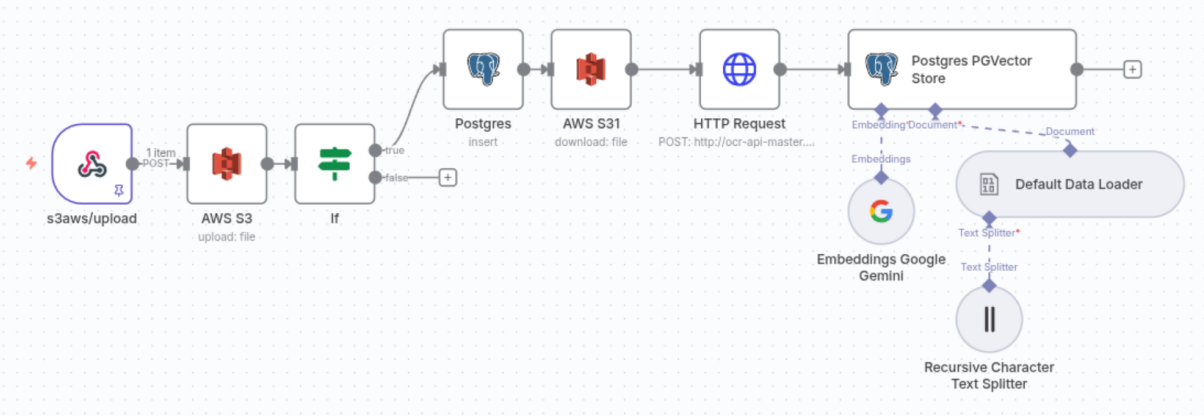
os documentos no armazenamento em um bucket S3 da AWS, assim como o controle de dados em um uma base PostgreSQL.

Para criar *endpoints* no N8N, deve-se criar um *workflow* e adicionar o primeiro nó como um Webhook e a partir disso configurar os parâmetros desse *endpoint* como método HTTP, *path*, autenticação, quando deve responder, o que deve retornar, assim como configurações de CORS e código de resposta, tudo através de cliques e poucas linhas de configuração. Os nós seguintes vão variar conforme as ações e objetivos desejados. Um *workflow* pode ser criado a partir do botão *"Create Workflow"* que aparece na interface do N8N. A seguir será apresentado como cada *workflow/endpoint* do agente foi feito.

3.2.1.1 *Workflow* - *Upload* de Arquivo

O *upload* de arquivo foi implementado através de um *workflow* chamado "*Upload File and Vectorize*". Esse *endpoint* possui uma sequência de sete nós interconectados, sendo o último um nó com 3 subnós, como mostra a Figura 7. Essa sequência de nós foi projetada para receber um arquivo enviado pelo usuário, armazená-lo no bucket S3 da AWS, inserir os dados do arquivo na base de dados PostgreSQL e, por fim, baixar o arquivo novamente para realizar a vetorização e armazenamento dos embeddings no banco de dados.

Figura 7 – *Workflow "Upload File and Vectorize".*



Fonte: acervo do autor.

Cada nó foi definido da seguinte forma:

1. ***Webhook* (s3aws/*upload*)**: Nó que define o ponto de entrada do *workflow*, criando um *endpoint* HTTP para receber requisições de *upload* de arquivos. Configura o método POST, *path* da URL e parâmetros de resposta.

2. **AWS S3 (*Upload*)**: Nó que realiza o *upload* do arquivo para o *bucket* S3 da AWS. Define o nome do *bucket*, estrutura de pastas para organização dos arquivos e credenciais de acesso ao serviço.

3. ***If* (Verificação)**: Nó de controle de fluxo que verifica se o *upload* para o S3 foi bem-sucedido. Avalia o status de retorno da operação anterior e decide se o *workflow* deve continuar para as próximas etapas.

4. **Postgres (Insert)**: Nó que registra as informações do arquivo na base de dados PostgreSQL. Insere dados como nome do arquivo, *path* do documento no S3 e identificador do chat na tabela de controle de arquivos.

5. **AWS S3 (*download:file*)**: Nó que baixa o arquivo recém enviado do *bucket* S3 para processamento local. Essa etapa é necessária para que o arquivo possa ser enviado ao serviço de OCR (definido na fundamentação de RAG na Seção 2.2.3).

6. **HTTP *Request* (OCR)**: Nó que envia o arquivo para o microsserviço já existente no TCE-MA de OCR. Configura a URL do serviço, método HTTP e formato de envio do arquivo para extração de texto.

7. **Postgres PGVector *Store***: Nó que armazena os *embeddings* do texto no banco de dados com a extensão pgvector. Coordena o processo de vetorização conectando-se aos subnós de fragmentação de texto e geração de *embeddings*.

8. ***Embeddings* Google Gemini**: Subnó que define o modelo de inteligência artificial usado para converter texto em representações vetoriais. Configura o modelo embeddings-001 do Google Gemini e as credenciais de acesso à API.

9. ***Recursive Character Text Splitter***: Subnó responsável por dividir o texto em fragmentos menores (*chunks*) para otimizar o processamento. Define o tamanho dos fragmentos e sobreposição entre eles para manter contexto.

10. ***Default Data Loader***: Subnó que prepara e estrutura os dados do texto para armazenamento no banco vetorial. Define o formato dos dados e metadados que serão recebidos.

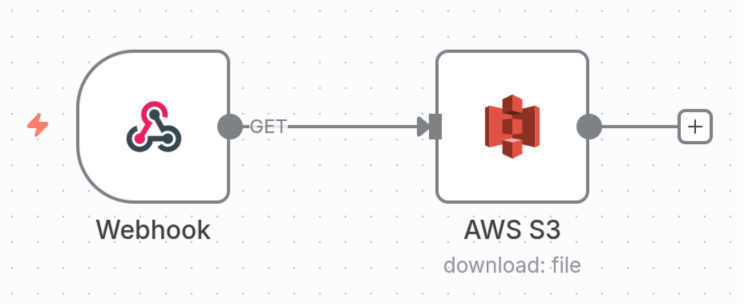
Todo este *workflow* resulta no *upload* seguro, processamento via OCR, fragmentação textual e vetorização dos arquivos enviados pelo usuário.

3.2.1.2 *Workflow - Download* de Arquivo

O *endpoint* de *download* de arquivo possui estrutura mais simples, contendo apenas dois nós, como mostra a Figura 8.

1. ***Webhook***: Nó que define o *endpoint* para requisições de *download* de arquivos. Configura o *path* da URL, tipo de resposta (dados binários) e políticas de CORS para acesso pela interface *web*.

Figura 8 – Workflow "Download de Arquivo".



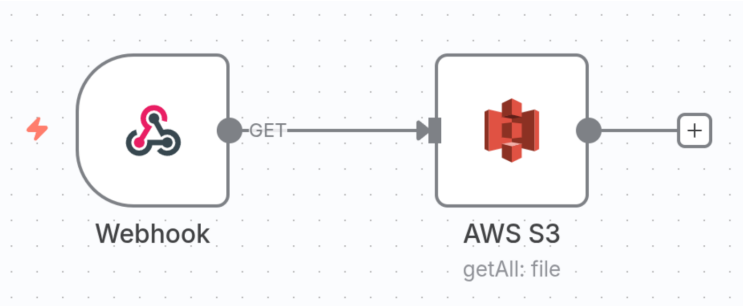
Fonte: acervo do autor.

2. **AWS S3**: Nó que realiza o *download* do arquivo específico do *bucket* S3. Utiliza os parâmetros nome do *bucket* e caminho do arquivo recebidos na requisição para localizar e retornar o conteúdo.

3.2.1.3 *Workflow* - Listagem de Arquivos

O *endpoint* de listagem de arquivos, similar ao de *download*, possui apenas dois nós, como mostra a Figura 9.

Figura 9 – Workflow "Listagem de Arquivos".



Fonte: acervo do autor.

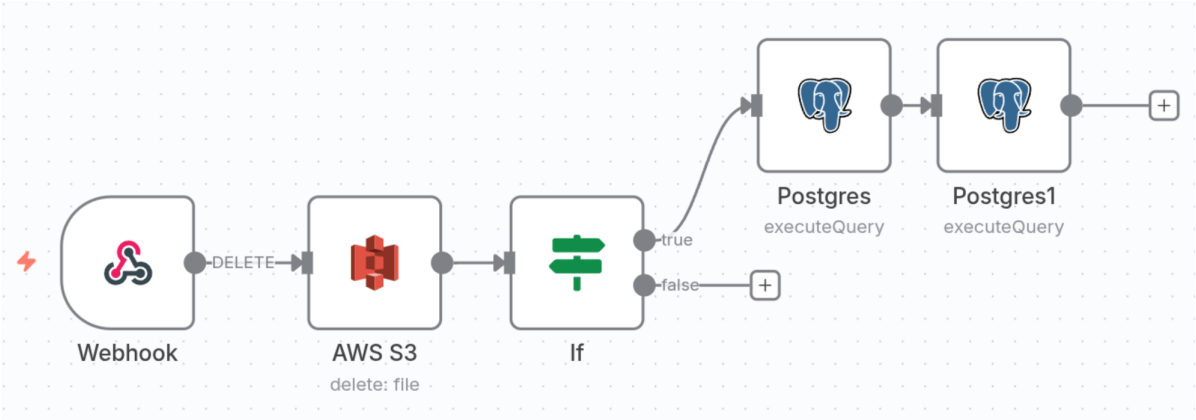
1. **Webhook**: Nó que define o *endpoint* para listagem de arquivos. Configura o método HTTP GET e parâmetros de resposta para retornar a lista de arquivos de uma conversa específica.

2. **AWS S3**: Nó que executa a operação de listagem de objetos no *bucket* S3. Configura o *bucket* de origem e filtros por pasta para retornar apenas os arquivos de uma conversa específica.

3.2.1.4 *Workflow* - Exclusão de Arquivo

O *endpoint* de exclusão de arquivo possui a sequência de 5 nós, como mostra a Figura 10 e é responsável por fazer a exclusão de um arquivo específico tanto no *bucket* S3 quanto na base de dados PostgreSQL. Segue a sequência de nós do *workflow*:

Figura 10 – Workflow "Exclusão de Arquivo".



Fonte: acervo do autor.

1. ***Webhook***: Nó que define o *endpoint* para requisições de exclusão de arquivos. Configura o método HTTP DELETE e parâmetros para receber o identificador do arquivo a ser removido.

2. **AWS S3**: Nó que realiza a exclusão física do arquivo no *bucket* S3. Utiliza a operação DELETE com o caminho completo do arquivo para remoção permanente do armazenamento.

3. **Verificação**: Nó de controle que verifica o sucesso da operação de exclusão no S3. Avalia o status de retorno para decidir se deve prosseguir com a limpeza dos dados no banco de dados.

4. **Consulta PostgreSQL**: Nó que executa comando SQL para remover os *embeddings* relacionados ao arquivo da tabela vetorial. Utiliza consulta baseada no ID do arquivo para limpeza dos dados de busca semântica.

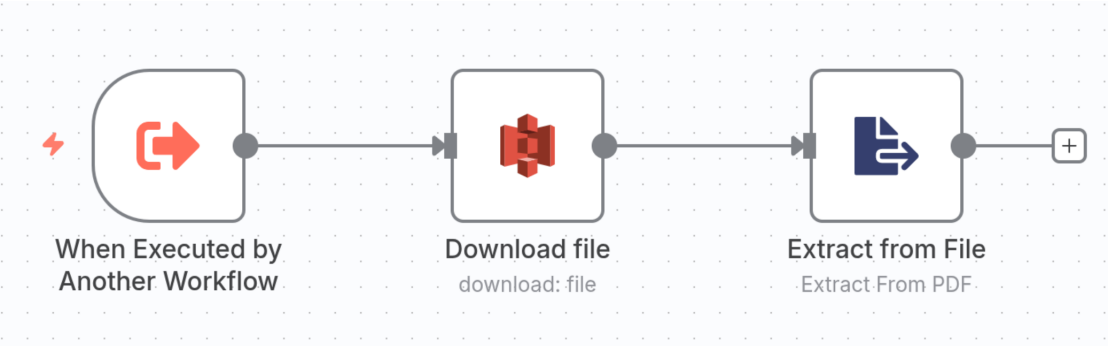
5. **Delete PostgreSQL**: Nó que remove o registro do arquivo da tabela de controle, finalizando o processo de exclusão completa. Executa comando SQL DELETE baseado no ID do arquivo.

3.2.2 Implementação do Agente Inteligente

Com base na listagem de leis, normativos e modelos de relatórios fornecidos pelo auditor entrevistado e essenciais para a elaboração do relatório de Tomada de Contas

Especial, foi criado um *bucket* na AWS S3 para armazenar esses documentos. A partir desse armazenamento, foram desenvolvidos três *workflows* com estrutura idêntica para automatizar a obtenção e extração do conteúdo desses documentos. Esses fluxos são responsáveis por baixar os arquivos do *bucket* S3 e extrair o texto contido neles, permitindo que o agente tenha acesso às informações necessárias para exercer suas funções. Cada *workflow* é composto por três nós sequenciais: o primeiro nó é do tipo "*When Executed by Another Workflow*", que permite o *workflow* ser invocado por outro fluxo; o segundo nó é do tipo AWS S3 com operação configurada para *Download*, responsável por baixar os documentos do armazenamento de objetos com base no *path* do arquivo desejado; e o terceiro nó é do tipo "*Extract from file*", que extrai o conteúdo textual dos arquivos baixados e retorna essas informações para o *workflow* que originou a chamada. Essa estrutura pode ser observada na Figura 11.

Figura 11 – Estrutura padrão dos *workflows* de obtenção de leis, normativos e modelos de relatórios.



Fonte: acervo do autor.

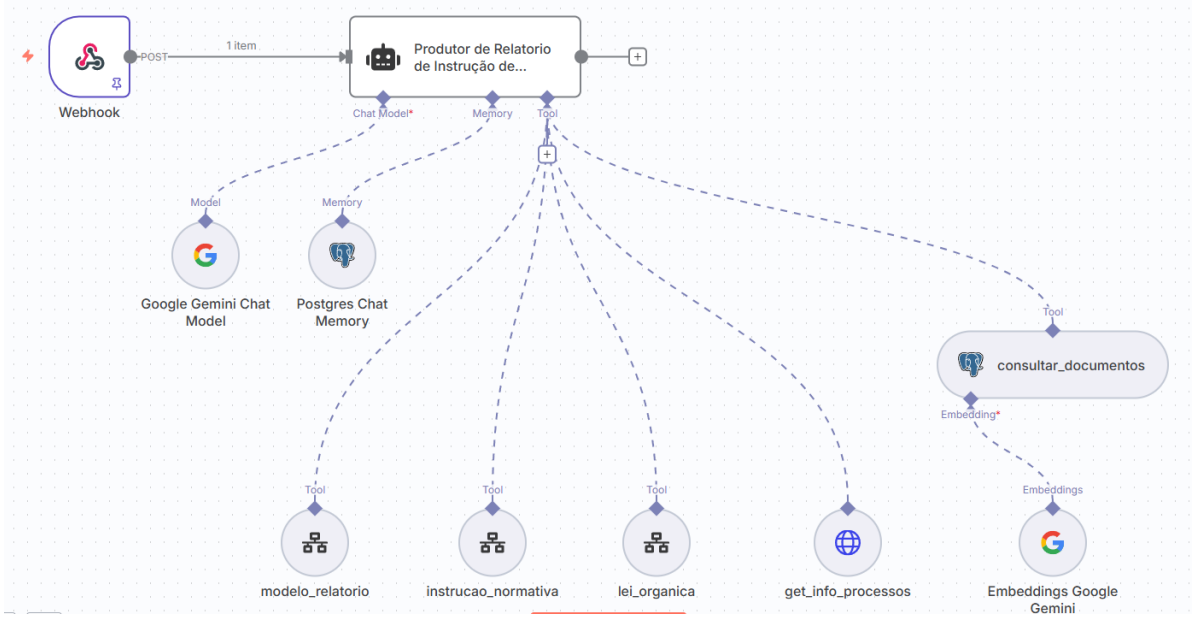
Com as operações de manipulação de arquivos e ferramentas para obtenção de documentos devidamente implementadas, tornou-se possível implementar o agente propriamente dito. Este agente inteligente foi implementado no *workflow* "AGENTE - TOMADA ESPECIAL DE CONTAS"e personalizado com um *prompt* específico, que foi contruído com base no *prompt* que o auditor entrevistado na fase de Levantamento de Requisitos (Seção 3.1)já utilizava.

O *workflow* do agente é composto por dois nós principais conectados, onde o primeiro é um *webhook* para definir o *endpoint* de acesso e o outro é o agente de IA que possui subnós como o modelo de linguagem (LLM), a memória e as ferramentas. Pode-se analisar a sua composição na Figura 12.

O agente é composto pelos seguintes nós:

1. ***Webhook***: Nó que define o *endpoint* principal do agente inteligente. Configurado com método POST para receber mensagens dos usuários e repassar para o agente.

Figura 12 – *Workflow* "Agente Inteligente - Tomada Especial de Contas".



Fonte: acervo do autor.

2. **Produtor de Relatório - *Agent***: Nó principal que orquestra todo o funcionamento do agente inteligente. Integra *prompt* especializado, modelo de linguagem, memória e ferramentas, permitindo a interação com o usuário e a geração de relatórios técnicos. Este nó é responsável por receber as mensagens do usuário, processar as informações e gerar respostas estruturadas.

O *prompt* foi feito em formato Markdown e implementa um protocolo rigoroso que exige a execução sequencial de todas as ferramentas antes da geração do relatório final, garantindo fundamentação legal completa e coleta abrangente de dados.

O agente central possui conexões especializadas com os seguintes nós auxiliares:

1. **Google Gemini *Chat Model***: Nó que configura o modelo de inteligência artificial principal do sistema. Define o uso do modelo Gemini 2.5 Pro *Preview* para processamento de linguagem natural e geração de respostas.

2. **Postgres *Chat Memory***: Nó que gerencia a memória conversacional do agente. Armazena e recupera o histórico de mensagens de cada conversa, configurando janela de contexto e identificação por *chat*.

3. **get_info_processos**: Nó responsável por consumir um *endpoint* HTTP já existente que consulta informações básicas sobre o processo informado pelo usuário na aplicação de processos internos do TCE-MA.

4. **modelo_relatorio**: Nó que fornece acesso ao modelo padrão de relatório de Tomada de Contas Especial, fornecendo contexto para que o agente possa gerar relatórios estruturados.

5. **instrucao_normativa**: Nó que retorna instruções normativas específicas, contribuindo para a fundamentação legal das respostas do agente.

6. **lei_organica**: Nó que fornece acesso à Lei Orgânica do TCE-MA, permitindo que o agente consulte artigos e parágrafos relevantes para fundamentar suas respostas.

7. **consultar_documentos**: Nó de busca semântica nos documentos vetorizados enviados pelo usuário, conectado ao módulo de *embeddings* do Google Gemini.

8. **Embeddings Google Gemini**: Nó módulo de *embeddings* conectado à ferramenta de consulta de documentos para realizar buscas vetoriais.

Todas as ferramentas estão conectadas ao nó central que é do tipo *AI Agent* através de conexões do tipo *Tool*, permitindo que o agente acesse cada uma conforme sua lógica de *prompt* determinar.

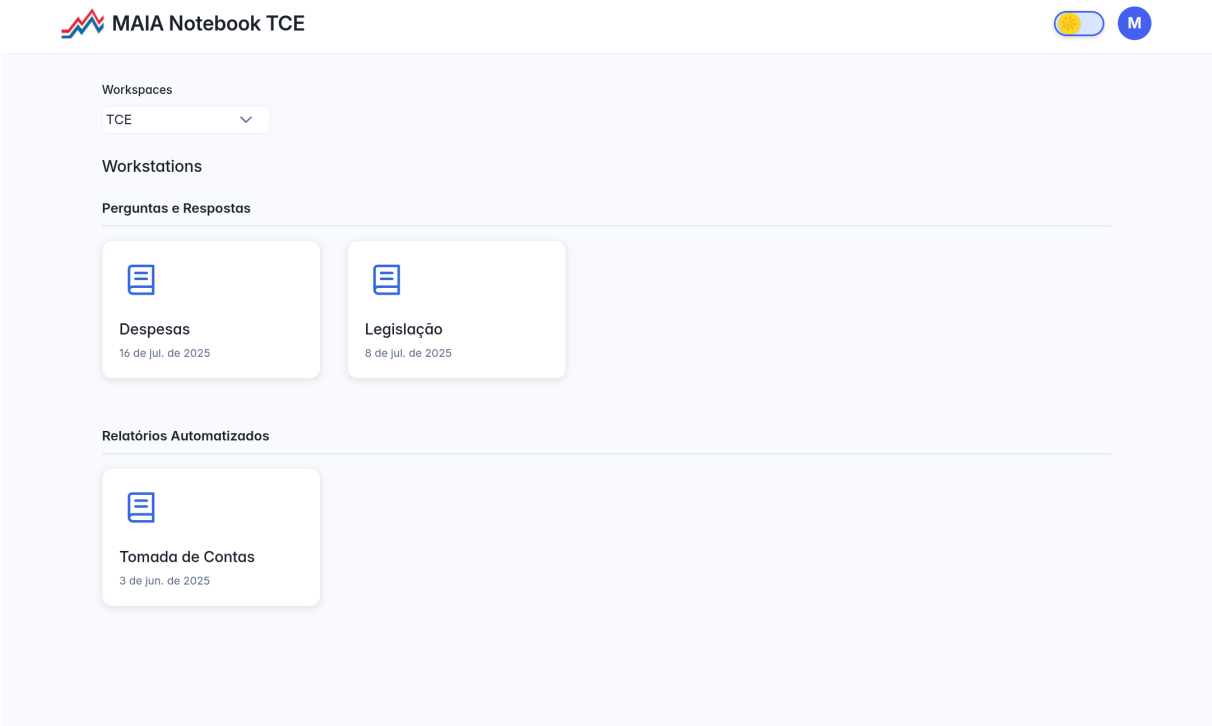
Para testes durante o desenvolvimento o próprio N8N fornece um nó de *chat* para conversar com o modelo diretamente, permitindo que o agente seja testado e validado antes de ser integrado à interface de usuário. Este nó pode ser utilizado para enviar mensagens e receber respostas do agente, facilitando o processo de depuração e ajuste fino do comportamento do agente.

3.2.3 Desenvolvimento da Interface de Usuário

Em seguida, incorporou-se o agente na interface de usuário que já estava implementada no TCE-MA. Essa aplicação, desenvolvida em Next.js e feita priorizando usabilidade, serve como ponto de contato principal entre os auditores e os agentes inteligentes do Tribunal. A página inicial desse *hub* que lista os agentes e possibilita a seleção deles pode ser vista na Figura 13.

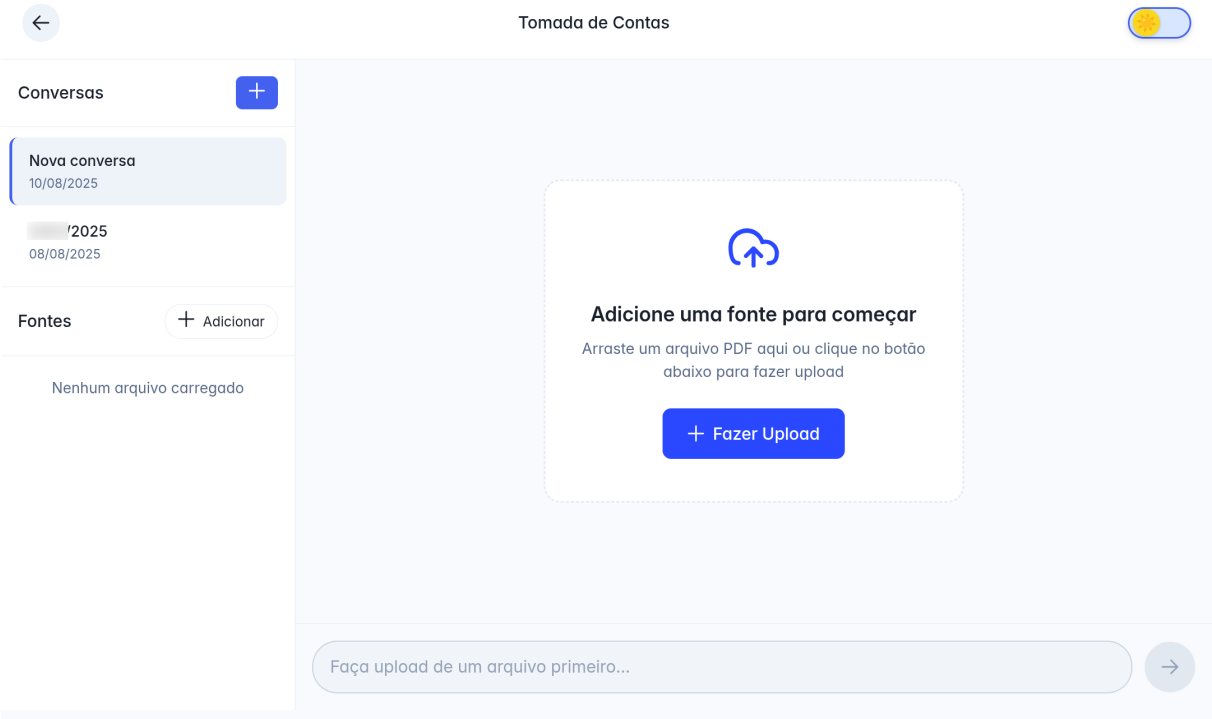
A interface foi projetada com foco na simplicidade e funcionalidade, integrando-se aos *endpoints* criados no N8N para disponibilizar todas as funcionalidades do agente inteligente. A interface permite que o auditor interaja por meio de um *chat* ao lado direito da tela, realizando o *upload* dos documentos pertinentes, informando o número do processo ou informações necessárias e então visualizando os resultados gerados pelo agente. O auditor pode criar uma ou mais conversas assim como fazer *upload* de uma ou mais fontes (documentos) do lado esquerdo da tela, sendo cada conversa dedicada a um processo específico. Essas possibilidades de interações podem ser analisadas na Figura 14.

Figura 13 – Tela inicial do hub de agentes do TCE-MA.



Fonte: acervo do autor.

Figura 14 – Interface de interação com o Agente Tomada de Contas Especial.



Fonte: acervo do autor.

O *hub* de agentes já possuía um *template* de *chat* previamente desenvolvido, que foi adaptado especificamente para o agente de Tomada de Contas Especial. Essa interface existente foi desenvolvida priorizando a usabilidade, permitindo que os auditores do TCE-MA naveguem facilmente entre as funcionalidades e interajam de forma intuitiva com o agente.

As principais adaptações realizadas no template envolveram a implementação do consumo dos *endpoints* HTTP criados para o gerenciamento de arquivos e a integração com o agente inteligente. Essas modificações permitiram que a interface fornecesse as funcionalidades de realizar o gerenciamento de documentos e interação com o agente. A adaptação garantiu que todos os padrões de design já existentes na aplicação fossem mantidos, garantindo a experiência de usuário fluida e intuitiva.

O processo completo de interação do auditor com o agente a partir da interface será apresentado na seção a seguir.

3.3 Teste e Avaliação

Após a implementação do agente inteligente, foram realizados testes práticos para validar sua funcionalidade e avaliar os benefícios proporcionados aos auditores do TCE-MA. Essa seção apresenta os resultados dos testes realizados, demonstrando o fluxo de trabalho do sistema através da interação do usuário final (auditor) com a interface gráfica, analisando as melhorias obtidas em relação ao processo tradicional. O processo exemplificativo utilizado mostra *prints* de exemplos reais de caso de uso do sistema. Por essa razão, mostramos com censura de informações sensíveis (borrado nas imagens) relacionadas aos processos, dessa forma mantendo a confidencialidade dos dados do TCE-MA.

A validação da ferramenta foi conduzida simulando o fluxo de trabalho real de um auditor durante o processo de Tomada de Contas Especial. O processo iniciou-se com o acesso à plataforma que começa validando se o usuário está autenticado, caso não esteja, ele é encaminhado para tela de *login*.

A Figura 15 apresenta a interface de *login* da plataforma dedicada a autenticação de todas as aplicações do TCE-MA, o Keycloak.

Após a autenticação bem-sucedida, o usuário é direcionado para a tela de seleção de agentes do hub já existente no TCE-MA, conforme já ilustrado na Figura 13. Essa interface permite ao auditor escolher entre diferentes agentes especializados disponíveis no sistema, incluindo o agente de Tomada de Contas Especial.

Ao clicar no botão do agente Tomada de Contas, o usuário acessa a interface principal do agente, também já ilustrada anteriormente na Figura 14, onde pode visualizar as conversas já criadas com o agente ou criar uma nova. Ao criar uma nova conversa, o

Figura 15 – Tela de login da aplicação.

Fonte: acervo do autor.

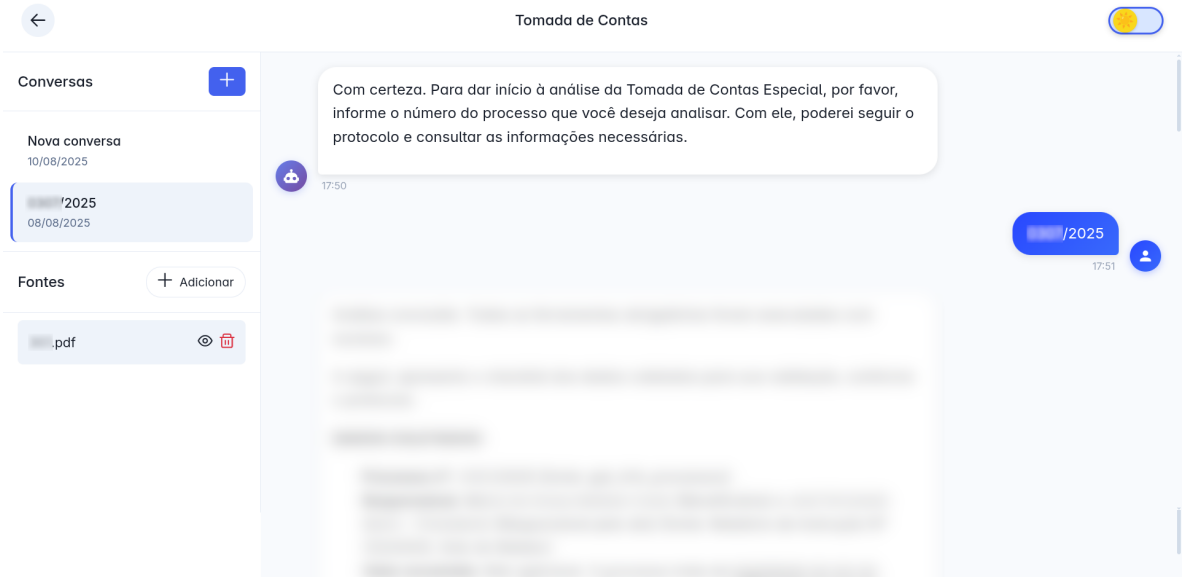
sistema solicita ao usuário o *upload* dos documentos relacionados ao processo de Tomada de Contas Especial. Essa funcionalidade permite que o auditor carregue múltiplos arquivos simultaneamente, que serão processados automaticamente pelo sistema.

Após o *upload* dos documentos, é liberado para auditor a opção de mandar mensagens escritas para o agente através do *chat*. Nessa etapa, o agente solicita que o usuário forneça o número do processo para que ele possa acessar as informações pertinentes armazenadas na base de dados do TCE-MA. O auditor pode digitar o número do processo diretamente no campo de mensagem, iniciando a consulta. Assim que o usuário envia o número do processo, a aplicação altera o nome da conversa ativa para o número do processo informado, facilitando a identificação e gerenciamento das conversas no menu a esquerda da tela. O usuário também pode fornecer outros documentos adicionais através do botão "+ Adicionar", localizado no menu lateral esquerdo, no inicio da listagem de fontes já enviadas. Todas essas interações podem ser vistas na Figura 16.

Em seguida o agente realiza as consultas internas e analisa os documentos enviados pelo usuário, assim como as leis e normativos, então gera um *checklist* de informações sobre o processo para que o auditor valide e responda se o agente deve ou não continuar para a etapa de elaboração de relatório. A Figura 17 demonstra essa etapa.

Após a validação do *checklist* pelo auditor, o agente avança para a etapa final de geração do relatório técnico. A Figura 18 ilustra o resultado dessa etapa, apresentando um modelo estruturado que atende aos padrões normativos do TCE-MA e incorpora todos os elementos identificados durante a análise automatizada. O auditor pode então revisar esse relatório e realizar ajustes finais antes de sua formalização.

Figura 16 – Tela de interação com o agente por meio do chat após o usuário fornecer fonte/documento.



Fonte: acervo do autor.

Os testes realizados evidenciaram significativos benefícios em relação ao processo tradicional de elaboração de relatórios de Tomada de Contas Especial. O principal ganho observado relaciona-se à redução do tempo necessário para a análise e elaboração de relatórios.

No processo tradicional, conforme identificado durante o levantamento de requisitos, um auditor necessitava de vários dias para analisar completamente um processo de média complexidade, incluindo a leitura detalhada de documentos, pesquisa normativa e elaboração do relatório técnico. Com a implementação do agente inteligente, esse tempo foi reduzido para aproximadamente 4 a 6 horas de trabalho efetivo do auditor.

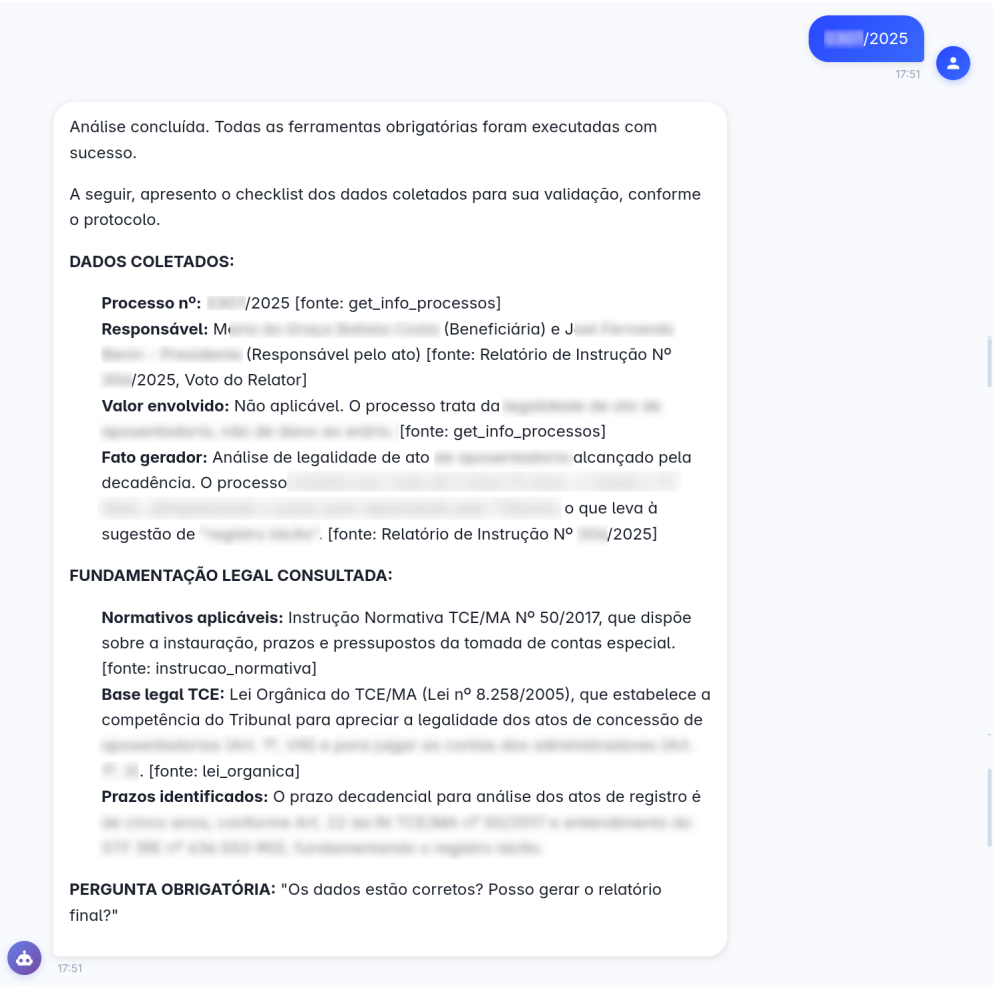
Essa redução de tempo representa um ganho de produtividade, permitindo que os auditores dediquem mais tempo à análise crítica dos casos e à tomada de decisões estratégicas.

Além da redução temporal, observou-se melhoria na padronização dos relatórios gerados. O agente inteligente assegura que todos os elementos normativos obrigatórios sejam contemplados no documento final, reduzindo a possibilidade de omissões que poderiam comprometer a qualidade técnica do trabalho.

Durante o processo de desenvolvimento e de validação, foram identificadas oportunidades de melhoria que podem ser implementadas em versões futuras do sistema. A principal limitação observada refere-se à atualização de normativos e legislação aplicável.

Atualmente, quando há alterações na legislação pertinente às Tomadas de Contas Especiais, os auditores não podem realizar a atualização diretamente no sistema. Essas

Figura 17 – Resposta do agente com *checklist* de verificação.



Fonte: acervo do autor.

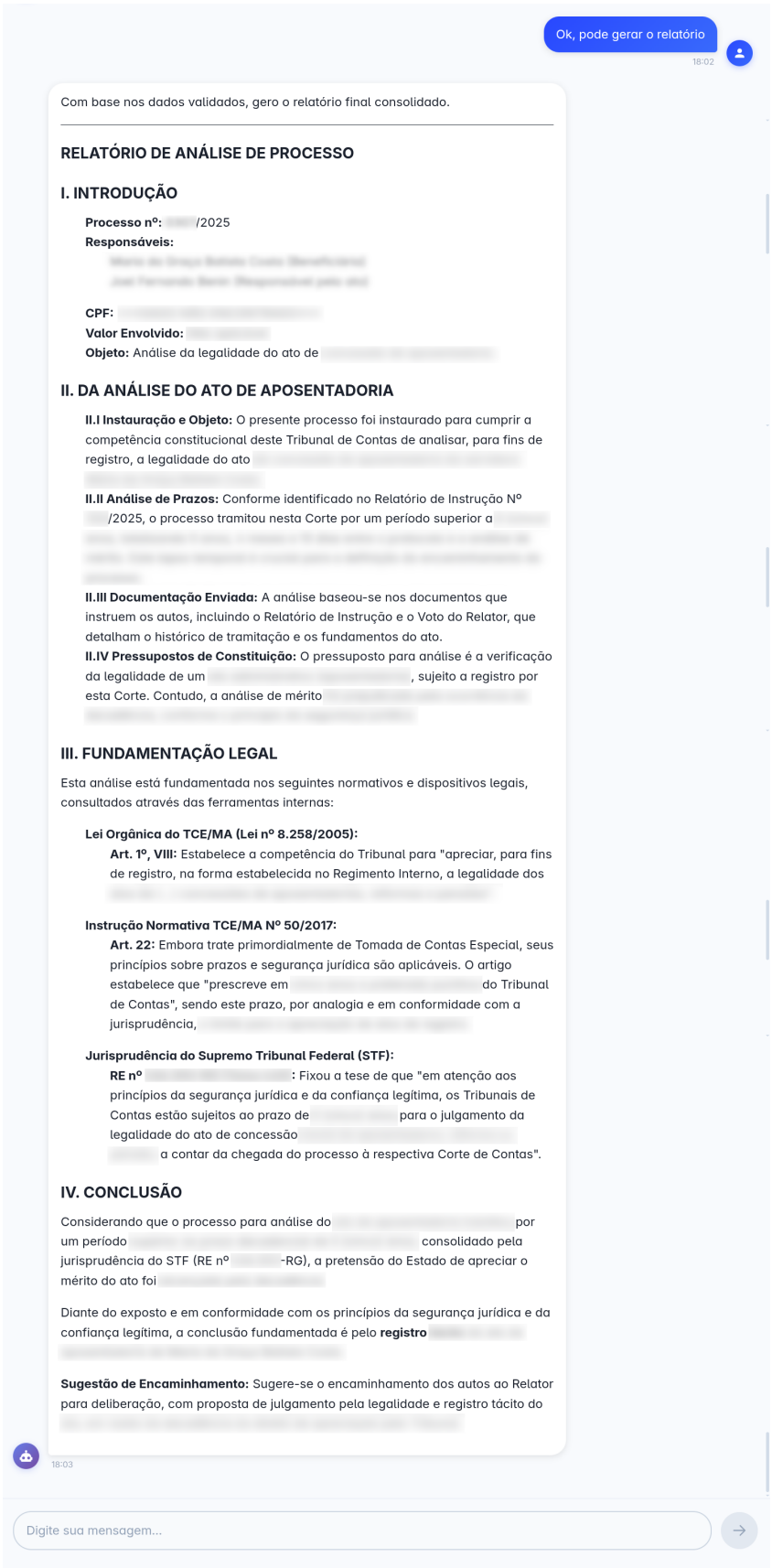
modificações devem ser solicitadas à equipe de desenvolvimento, que procede com as alterações nos workflows do N8N e nas bases de conhecimento do agente. Essa dependência técnica pode impactar a agilidade na incorporação de mudanças normativas.

Outra necessidade identificada foi o gerenciamento dos prompts utilizados no agente. Atualmente, os prompts estão fixados nos workflows do N8N, limitando a capacidade de ajustes finos no comportamento do agente sem intervenção técnica.

Uma melhoria significativa seria a implementação de uma interface administrativa que permita aos próprios auditores ou gestores do TCE-MA atualizar a base de conhecimento normativo do sistema e os prompts utilizados pelo agente. Essa funcionalidade poderia incluir:

- Interface para adição, edição e remoção de normativos e legislação aplicável assim como a atualização de prompts;
- Sistema de versionamento das alterações;

Figura 18 – Relatório gerado após validação do checklist.



Fonte: acervo do autor.

- Mecanismo de aprovação para mudanças críticas na base de conhecimento;
- Histórico de modificações realizadas.

A implementação de um editor de prompts acessível aos usuários especializados permitiria refinamento contínuo das instruções fornecidas ao agente, adaptação do comportamento do sistema conforme a experiência de uso, criação de prompts especializados para diferentes tipos de processos com base na expertise dos auditores, e a possibilidade de personalização de respostas para casos específicos.

O presente Capítulo apresentou a metodologia empregada na construção do agente inteligente para a Tomada de Contas Especial. A seguir, no Capítulo 4, serão feitas as considerações finais, bem como algumas sugestões de trabalhos futuros.

4 Conclusão

Este trabalho teve como objetivo a construção de um agente inteligente para auxiliar na elaboração de relatórios técnicos de Tomada de Contas Especial no Tribunal de Contas do Estado do Maranhão.

A aplicação desenvolvida teve como propósito fundamental otimizar o processo de análise documental e geração de relatórios padronizados, reduzindo o tempo necessário para essas atividades, assim como melhorando a padronização e qualidade dos documentos produzidos pelos auditores fiscais do TCE-MA.

Para o desenvolvimento da solução, foram exploradas as mais recentes tecnologias envolvendo a criação de agentes, destacando-se o Processamento de Linguagem Natural (PLN) e os *Large Language Models* (LLMs) com técnicas de Retrieval-Augmented Generation (RAG) e armazenamento vetorial. A plataforma N8N foi utilizada como orquestrador central, integrando diversos serviços e agilizando a construção do agente inteligente.

O sistema implementado integrou com sucesso as tecnologias exploradas, incluindo o LLM Gemini como cerebro do agente, a técnica RAG para fornecimento de conhecimento específico do domínio, o PostgreSQL para armazenamento de dados relacionais, assim como sua extensão pgvector para *embeddings*, e a plataforma N8N para orquestração de *workflows*. A arquitetura desenvolvida demonstrou-se robusta e eficiente, permitindo integração harmoniosa com a infraestrutura tecnológica existente no TCE-MA.

Os resultados obtidos evidenciaram ganhos expressivos de produtividade. O tempo necessário para análise e elaboração de relatórios foi reduzido de vários dias no processo tradicional para aproximadamente 4 a 6 horas com o auxílio do agente inteligente, representando um ganho de produtividade significativo. Além da redução temporal, observou-se melhoria significativa na padronização dos relatórios gerados, garantindo que todos os elementos normativos obrigatórios fossem contemplados nos documentos finais. Com isso, o objetivo geral proposto foi plenamente alcançado.

Quanto aos objetivos específicos propostos, todos foram alcançados com êxito. O processo atual de elaboração de relatórios manuais foi analisado e as oportunidades de otimização identificadas. As principais tecnologias de IA aplicáveis ao contexto foram estudadas e selecionadas e em seguida o agente inteligente foi desenvolvido com sucesso, integrando capacidades de análise documental automatizada e geração de relatórios padronizados. Por fim, a validação prática da aplicação comprovou melhorias no processo executado pelos auditores fiscais, tanto em termos de tempo quanto de qualidade dos relatórios produzidos.

Apesar dos resultados promissores observou-se algumas limitações na solução como a falta de atualizações dos normativos e resoluções aplicáveis, assim como gerenciamento direto dos prompts utilizados no agente por parte dos auditores. Essas funcionalidades acabaram ficando limitadas por parte do time técnico de desenvolvimento do agente.

Com base nessas limitações, destacam-se como trabalhos futuros a implementação de uma interface administrativa que possibilite a gerência da base normativa e de *prompts*, assim como controle de permissões pelos próprios auditores , eliminando a dependência da equipe técnica para manutenções rotineiras nesses requisitos. Essas implementações futuras representam oportunidades para evolução da solução.

Por fim, este trabalho demonstra que a união de expertise técnica humana com agentes de IA pode transformar significativamente processos complexos de análise documental no setor público, abrindo caminho para uma ampla modernização dos órgãos de controle e contribuindo para uma administração pública mais eficiente e transparente.

Referências

ANTHROPIC. *Claude*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://claude.ai>. Citado na página 17.

AUTH.JS. *Auth.js: authentication for the Web*. 2025. Acesso em: 8 ago. 2025. Disponível em: <https://authjs.dev/>. Citado na página 25.

BRASIL. *Constituição da República Federativa do Brasil de 1988*. 1988. Acesso em: 10 fev. 2025. Disponível em: <http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm>. Citado 2 vezes nas páginas 11 e 13.

CELONIS, INC. *Make | Automation you can see, flex, and scale*. 2025. Acesso em: 05 ago. 2025. Disponível em: <https://www.make.com>. Citado na página 21.

CORREGEDORIA GERAL DO ESTADO DO MARANHÃO. *Manual de Procedimentos para Tomada de Contas Especial*. São Luís: [s.n.], 2023. Acesso em: 8 ago. 2025. Disponível em: <https://stc.ma.gov.br/uploads/stc/docs/MANUAL-2021-COMPLETO.pdf>. Citado 3 vezes nas páginas 11, 14 e 28.

ECMA INTERNATIONAL. *ECMAScript Language Specification*. [S.l.], 2025. Standard ECMA-262, 15th Edition. Acesso em: 14 ago. 2025. Disponível em: <https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>. Citado na página 23.

EISENSTEIN, J. *Natural Language Processing*. MIT Press, 2018. Acesso em: 06 ago. 2025. Disponível em: <https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf>. Citado na página 15.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados*. 7. ed. [S.l.]: Pearson Education do Brasil, 2018. ISBN 9786550110512. Citado na página 18.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, Irvine, 2000. Acesso em: 13 ago. 2025. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Citado na página 21.

GOOGLE. *Angular | Introduction | What is Angular?* 2025. Acesso em: 13 ago. 2025. Disponível em: <https://angular.dev/overview>. Citado na página 23.

GOOGLE. *Gemini*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://gemini.google.com>. Citado na página 17.

ISLAM, N.; ISLAM, Z.; NOOR, N. *A Survey on Optical Character Recognition System*. 2017. ArXiv:1710.05703. Acesso em: 14 ago. 2025. Disponível em: <https://arxiv.org/abs/1710.05703>. Citado na página 15.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3. ed. [s.n.], 2025. Acesso em: 06 ago. 2025. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>. Citado 2 vezes nas páginas 15 e 16.

KEYCLOAK. *Keycloak | Open Source Identity and Access Management*. 2025. Acesso em: 8 ago. 2025. Disponível em: <https://www.keycloak.org/>. Citado na página 25.

LANGCHAIN. *LangChain*. 2025. Acesso em: 05 ago. 2025. Disponível em: <https://www.langchain.com>. Citado na página 21.

LEWIS, P.; PEREZ, E.; PIKTUS, A.; PETRONI, F.; KARPUKHIN, V.; GOYAL, N.; KüTTLER, H.; LEWIS, M.; YIH, W.-t.; ROCKTäSCHEL, T.; RIEDEL, S.; KIELA, D. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. ArXiv:2005.11401. Acesso em: 01 ago. 2025. Disponível em: <https://arxiv.org/abs/2005.11401>. Citado 2 vezes nas páginas 17 e 18.

LLAMAINDEX. *LlamaIndex | Redefine document workflows with AI Agents*. 2025. Acesso em: 05 ago. 2025. Disponível em: <https://www.llamaindex.ai>. Citado na página 21.

MARANHÃO. [Constituição(1989)], *Constituição do Estado do Maranhão*. 1989. Acesso em: 17 jun. 2025. Disponível em: <https://www.al.ma.leg.br/arquivos/constituicaoma.pdf>. Citado 2 vezes nas páginas 13 e 27.

MARANHÃO. *Lei n. 8.258, de 6 de junho de 2005. Dispõe sobre a Lei Orgânica do Tribunal de Contas do Estado do Maranhão, e dá outras providências*. 2005. Acesso em: 17 jun. 2025. Disponível em: <https://app.tcema.tc.br/publicacao/#/documentohtml/8200?compilado=true>. Citado 3 vezes nas páginas 13, 14 e 27.

META. *facebook | O Facebook ajuda você a se conectar e compartilhar com as pessoas que fazem parte da sua vida*. 2025. Acesso em: 13 ago. 2025. Disponível em: <https://www.facebook.com/>. Citado na página 24.

META. *Llama*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://www.llama.com/>. Citado na página 17.

META PLATFORMS, INC. *React | The library for web and native user interfaces*. 2025. Acesso em: 13 ago. 2025. Disponível em: <https://react.dev>. Citado na página 23.

MICROSOFT. *Microsoft SQL Server*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://www.microsoft.com/pt-br/sql-server>. Citado na página 19.

MISTRAL AI. *Mistral*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://mistral.ai/>. Citado na página 17.

MONTAñO, S. R. *FastAPI*. 2018. Acesso em: 05 ago. 2025. Disponível em: <https://github.com/tiangolo/fastapi>. Citado na página 21.

N8N. *n8n | Flexible AI workflow automation for technical teams*. 2025. Acesso em: 28 jul. 2025. Disponível em: <https://n8n.io/>. Citado na página 21.

NEXT.JS. *Next.js*. 2025. Acesso em: 06 ago. 2025. Disponível em: <https://nextjs.org>. Citado na página 25.

OPENAI. *ChatGPT*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://chat.openai.com>. Citado na página 17.

ORACLE. *MySQL*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://www.mysql.com/>. Citado na página 19.

ORACLE. *Oracle Database*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://www.oracle.com/database/>. Citado na página 19.

PANDAS. *pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language*. 2025. Acesso em: 05 ago. 2025. Disponível em: <https://pandas.pydata.org>. Citado na página 21.

PGVECTOR. *pgvector | Open-source vector similarity search for Postgres*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://github.com/pgvector/pgvector>. Citado na página 20.

POSTGRESQL. *PostgreSQL*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://www.postgresql.org/>. Citado na página 19.

PYTHON SOFTWARE FOUNDATION. *Python*. 2025. Acesso em: 03 ago. 2025. Disponível em: <https://www.python.org>. Citado na página 21.

REITZ, K. *Requests | HTTP for Humans*. 2025. Acesso em: 05 ago. 2025. Disponível em: <https://requests.readthedocs.io/en/latest/>. Citado na página 21.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4. ed. Hoboken, NJ: Pearson, 2021. Citado na página 14.

SCIKIT-LEARN. *scikit-learn | User Guide*. 2025. Acesso em: 13 ago. 2025. Disponível em: <https://scikit-learn.org/stable/user_guide.html>. Citado na página 21.

TELEGRAM. *Perguntas Frequentes*. 2025. Acesso em: 06 ago. 2025. Disponível em: <https://telegram.org/faq>. Citado na página 23.

TELEGRAM. *Telegram*. 2025. Acesso em: 06 ago. 2025. Disponível em: <https://telegram.org>. Citado na página 23.

TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO. *Regimento Interno do Tribunal de Contas do Estado do Maranhão*. 2005. Acesso em: 17 jun. 2025. Disponível em: <https://app.tcema.tc.br/publicacao/#/documentohtml/8207?compilado=true>. Citado na página 27.

TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO. *Decisão Normativa TCE n. 28, de 6 de dezembro de 2017. Altera a Instrução Normativa TCE/MA nº 50, de 30 de agosto de 2017, e dá outras providências*. 2017. Acesso em: 14 de ago. 2025. Disponível em: <https://app.stc.ma.gov.br/legisla/consulta/publicacao/5051>. Citado na página 27.

TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO. *Instrução Normativa TCE/MA n. 50, de 30 de agosto de 2017. Dispõe sobre medidas administrativas para elisão de dano e sobre instauração, pressupostos de constituição, quantificação do débito, conclusão e encaminhamento de tomada de contas especial para julgamento pelo Tribunal de Contas do Estado do Maranhão e disciplina o instituto da decadência*. 2017. Acesso em: 17 jun. 2025. Disponível em: <https://app.tcema.tc.br/publicacao/#/documentohtml/834?compilado=true>. Citado 3 vezes nas páginas 13, 14 e 27.

TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO. *Decisão Normativa TCE n. 38, de 21 de outubro de 2020. Atende ao disposto no art. 10, inciso I, da Instrução Normativa TCE/MA nº 50, de 30 de agosto de 2017, fixando o valor histórico do dano causado ao erário a partir do qual o controlado fica obrigado a enviar ao Tribunal de Contas do Estado do Maranhão a tomada de contas especial respectiva.* 2020. Acesso em: 17 jun. 2025. Disponível em: <https://app.tcema.tc.br/publicacao/#/documentohtml/7083?compilado=true>. Citado na página 27.

TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO. *Resolução TCE/MA n. 383, de 26 de abril de 2023. Regulamenta, no âmbito do Tribunal de Contas do Estado do Maranhão, a prescrição para o exercício das pretensões punitiva e de ressarcimento e dá outras providências.* 2023. Acesso em: 17 jun. 2025. Disponível em: <https://app.tcema.tc.br/publicacao/#/documentohtml/17678?compilado=true>. Citado na página 27.

TRIBUNAL DE CONTAS DO ESTADO DO MARANHÃO. *Resolução TCE/MA n. 406, de 14 de agosto de 2024. Altera a Resolução TCE/MA n° 383, de 26 de abril de 2023, que regulamenta, no âmbito de Tribunal de Contas do Estado de Maranhão, a prescrição para o exercício das pretensões punitiva e de ressarcimento e dá outras providências.* 2024. Acesso em: 17 jun. 2025. Disponível em: <https://app.tcema.tc.br/publicacao/#/documentohtml/23397?compilado=true>. Citado na página 27.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. *Advances in Neural Information Processing Systems*, v. 30, p. 5998–6008, 2017. Citado na página 16.

VUE.JS. *Vue.js | The Progressive JavaScript Framework.* 2025. Acesso em: 13 ago. 2025. Disponível em: <https://vuejs.org>. Citado na página 23.

WHATSAPP. *WhatsApp | Mensagens e chamadas simples, fiáveis, privadas e gratuitas*, disponíveis em todo o mundo.* 2025. Acesso em: 06 ago. 2025. Disponível em: <https://www.whatsapp.com/?lang=pt_PT>. Citado na página 23.

WHATSAPP BUSINESS. *WhatsApp Business | Use todo o potencial da API do WhatsApp para oferecer experiências incríveis a um grande número de clientes.* 2025. Acesso em: 06 ago. 2025. Disponível em: <https://business.whatsapp.com/products/business-platform>. Citado na página 23.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, v. 10, n. 2, p. 115–152, 1995. Citado na página 14.