

Universidade de Brasília – UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Engenharia de Software

Sistema de apoio à análise de processos organizacionais: Uma abordagem com inteligência artificial e controle de contexto

Autor: Miguel Matos Costa de Frias Barbosa

Orientador: Prof. Dr. Sergio Antônio Andrade de Freitas

Brasília, DF

2025



Miguel Matos Costa de Frias Barbosa

Sistema de apoio à análise de processos organizacionais: Uma abordagem com inteligência artificial e controle de contexto

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Orientador: Prof. Dr. Sergio Antônio Andrade de Freitas

Brasília, DF

2025

Miguel Matos Costa de Frias Barbosa

Sistema de apoio à análise de processos organizacionais: Uma abordagem com inteligência artificial e controle de contexto/ Miguel Matos Costa de Frias Barbosa. – Brasília, DF, 2025-

90 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Sergio Antônio Andrade de Freitas

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologias em Engenharia – FCTE , 2025.

1. *Large Language Models*. 2. *Model Context Protocol*. I. Prof. Dr. Sergio Antônio Andrade de Freitas . II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Sistema de apoio à análise de processos organizacionais: Uma abordagem com inteligência artificial e controle de contexto

CDU 02:141:005.6

Miguel Matos Costa de Frias Barbosa

Sistema de apoio à análise de processos organizacionais: Uma abordagem com inteligência artificial e controle de contexto

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 16 de julho de 2025 – Data da aprovação do trabalho:

Prof. Dr. Sergio Antônio Andrade de Freitas
Orientador

Prof. Dr. André Luiz Peron Martins Lanna
Convidado 1

Dr. Waldeyr Mendes Cordeiro da Silva
Convidado 2

Brasília, DF
2025

Resumo

A análise de processos organizacionais, especialmente no setor público, enfrenta desafios críticos relacionados à repetitividade e padronização, resultando em ineficiência operacional e comprometimento da rastreabilidade das decisões. As abordagens atuais frequentemente carecem de ferramentas integradas capazes de aproveitar o conhecimento contido em documentos históricos e no contexto legislativo para auxiliar os analistas. Este trabalho apresenta o desenvolvimento do "SEI Expert", um sistema de software que integra modelos de linguagem de grande escala (LLMs), aprendizado de máquina e protocolos de controle de contexto para otimizar a análise de processos.

A solução implementada apresenta uma arquitetura robusta com uma camada de inteligência artificial especializada, utilizando um validador semântico e um orquestrador para garantir precisão contextual e mitigar alucinações por meio da técnica *Retrieval-Augmented Generation* (RAG).Este trabalho detalha o desde o planejamento ao desenvolvimento do sistema, os resultados demonstram a entrega de um produto mínimo viável (MVP) funcional, capaz de ingerir dados processuais, fornecer respostas fundamentadas com rastreabilidade de fontes e reduzir a carga cognitiva dos analistas, promovendo maior transparência e padronização na tomada de decisão organizacional.

Palavras-chave: Inteligência Artificial. *Large Language Models* (LLMs). *Retrieval-Augmented Generation* (RAG). Análise de Processos. *Model Context Protocol* (MCP).

Abstract

The analysis of organizational processes, particularly in the public sector, faces critical challenges related to repetitiveness and standardization, leading to operational inefficiency and compromised decision traceability. Current approaches often lack integrated tools capable of leveraging knowledge contained in historical documents and legislative contexts to assist analysts. This work presents the development of "SEI Expert", a software system that integrates Large Language Models (LLMs), Machine Learning, and context control protocols to optimize process analysis.

The implemented solution features a robust architecture with a specialized Artificial Intelligence layer, utilizing a semantic validator and an orchestrator to ensure contextual accuracy and mitigate hallucinations through the Retrieval-Augmented Generation (RAG) technique. This work details the entire lifecycle, from planning to system development. The results demonstrate the delivery of a functional Minimum Viable Product (MVP) capable of ingesting process data, providing grounded responses with source traceability, and reducing the analysts' cognitive load, thereby promoting greater transparency and standardization in organizational decision-making.

Keywords: Artificial Intelligence. Large Language Models (LLMs). Retrieval-Augmented Generation (RAG). Process Analysis. Model Context Protocol (MCP).

Lista de ilustrações

Figura 1 – Etapas da engenharia de requisitos. 26
Figura 2 – Visão em espiral dos processos da engenharia de requisitos. 27
Figura 3 – Desenvolvimentos orientado a um plano e ágil. 29
Figura 4 – Composição de um processo organizacional. 31
Figura 5 – Diagrama de Venn representando subáreas da IA 32
Figura 6 – Exemplo de *clusters* 34
Figura 7 – Ilustração representando um modelo de *deep learning* 35
Figura 8 – Modelo de arquitetura *transformer* 39
Figura 9 – Visualização dos componentes do RAG 39
Figura 10 – Uso de ferramentas com e sem MCP 40
Figura 11 – Interação do usuário com o SEI e IA 43
Figura 12 – Diagrama da arquitetura geral do sistema. 44
Figura 13 – Representação dos níveis do sistema. 47
Figura 14 – Diagrama da camada de IA 51
Figura 15 – Protótipo da tela de gerenciamento de legislações 54
Figura 16 – Protótipo da tela de gerenciamento de processos 55
Figura 17 – Protótipo da tela de *chatbot* 55
Figura 18 – Protótipo da tela de cadastro 56
Figura 19 – Diagrama da Arquitetura do Sistema 68
Figura 20 – Planejamento original 70
Figura 21 – Planejamento original 72
Figura 22 – Tela de *login* do sistema 75
Figura 23 – Fluxo de recuperação de senha 76
Figura 24 – Tela de cadastro com seleção de perfil (*role*) 77
Figura 25 – Visão geral do *dashboard* administrativo 77
Figura 26 – *Dashboard* com a interface de ajuda ativa 78
Figura 27 – Listagem geral de processos importados 78
Figura 28 – Visualização de processos com sistema de *tags* coloridas 79
Figura 29 – Detalhe do processo aberto com lista de arquivos 80
Figura 30 – Interface de listagem da base de legislação 80
Figura 31 – Legislação categorizada com *tags* semânticas 81
Figura 32 – Visualização detalhada do texto da lei 81
Figura 33 – Tela inicial do assistente virtual inteligente 82
Figura 34 – *Chatbot* realizando análise detalhada de um processo 83
Figura 35 – Análise com citação direta de lei anexada ao contexto 83
Figura 36 – Visão geral do painel administrativo 84

Figura 37 – Gerenciamento de usuários e permissões 85
Figura 38 – Métricas detalhadas e auditoria do sistema 85
Figura 39 – Tela de configurações de perfil do usuário 86

Lista de tabelas

Tabela 1 – Cronograma de Atividades - TCC 1 22
Tabela 2 – Cronograma de Atividades - TCC 2 23
Tabela 3 – Requisitos funcionais do sistema no nível 1 58
Tabela 4 – Requisitos não funcionais do sistema no nível 1 59
Tabela 5 – Requisitos funcionais do sistema no nível 2 59
Tabela 6 – Requisitos não funcionais do sistema no nível 2 60
Tabela 7 – Requisitos funcionais do sistema no nível 3 60
Tabela 8 – Requisitos não funcionais do sistema no nível 3 60
Tabela 9 – Requisitos funcionais do sistema no nível 4 61
Tabela 10 – Requisitos não funcionais do sistema no nível 4 61
Tabela 11 – Tecnologias e Ferramentas Utilizadas 69

Lista de abreviaturas e siglas

TCC	Trabalho de Conclusão de Curso
MCP	*Model Context Protocol*
IA	Inteligência artificial
ML	*Machine learning*
PLN	Processamento de linguagem natural
BERT	*Bidirectional Encoder Representations from Transformers*
GPT	*Generative Pre-trained Transformer*
LLM	*Large Language Model*
RAG	*Retrieval Augmented Generation*
API	*Application Programming Interface*
SEI	Sistema Eletrônico de Informações
SOAP	*Simple Object Access Protocol*
WSDL	*Web Services Description Language*
RBAC	*Roel-Based Access Control*

Sumário

1 INTRODUÇÃO . . . 19
1.1 Contextualização . . . 19
1.2 Motivação . . . 19
1.3 O Problema . . . 19
1.4 Objetivos . . . 20
1.4.1 Objetivo Geral . . . 20
1.4.2 Objetivos Específicos (Desenvolvimento e Implementação) . . . 20
1.4.2.1 Implementação da Arquitetura Fullstack . . . 20
1.4.2.2 Construção da Camada de Inteligência Artificial . . . 20
1.4.2.3 Desenvolvimento do Módulo de Gestão e Acesso . . . 20
1.4.2.4 Desenvolvimento das Interfaces Funcionais . . . 21
1.4.2.5 Implementação do Mecanismo de Ingestão de Dados . . . 21
1.4.2.6 Execução e Gestão Ágil do Projeto . . . 21
1.5 Metodologia . . . 21
1.5.1 Cronograma . . . 22
1.6 Estruturação do trabalho . . . 22

2 FUNDAMENTAÇÃO TEÓRICA . . . 25
2.1 Engenharia de Software . . . 25
2.1.1 Engenharia de requisitos . . . 25
2.1.2 Metodologias de desenvolvimento de *software* . . . 28
2.1.3 Design e arquitetura de *software* . . . 29
2.1.4 Prototipação . . . 30
2.2 Gerenciamento e Análise de Processos Organizacionais . . . 30
2.3 Inteligência artificial . . . 32
2.3.1 Aprendizagem de máquina . . . 33
2.3.1.1 *Clustering* (Agrupamento) . . . 33
2.3.2 Aprendizagem profunda . . . 34
2.3.3 Processamento de linguagem natural . . . 36
2.3.4 Modelos de linguagem de grande escala . . . 37
2.3.5 Geração Aumentada por Recuperação . . . 38
2.3.6 *Model context protocol* (MCP) . . . 40
2.4 Sistema Eletrônico de Informações . . . 41
2.4.1 Integração via API . . . 42

3 PLANEJAMENTO DO SEI EXPERT . . . 43

3.1 Introdução . . . 43
3.2 Arquitetura do sistema . . . 44
3.3 Funcionalidades detalhadas por níveis de desenvolvimento . . . 46
3.3.1 Nível 1 - Base . . . 47
3.3.2 Nível 2 - Contextualização e rastreabilidade . . . 48
3.3.3 Nível 3 - Padronização . . . 49
3.3.4 Nível 4 - Recomendação e gestão . . . 49
3.4 Perfis de usuários do sistema . . . 50
3.5 Camada de inteligência artificial . . . 51
3.5.1 Validador semântico . . . 51
3.5.2 Orquestrador MCP . . . 52
3.5.2.1 Processamento da requisição . . . 52
3.5.2.2 Processamento da resposta . . . 53
3.5.3 Recomendador inteligente . . . 53
3.5.4 Modelos de linguagem de grande escala . . . 53
3.6 Prototipação . . . 54
3.6.1 Gerenciamento de legislações . . . 54
3.6.2 Gerenciamento de processos . . . 54
3.6.3 Interface do *chatbot* . . . 55
3.6.4 Cadastro no sistema . . . 55
3.7 Requisitos do sistema . . . 57
3.7.1 Nível 1 - Base . . . 57
3.7.2 Nível 2 - Contextualização e rastreabilidade . . . 57
3.7.3 Nível 3 - Padronização . . . 59
3.7.4 Nível 4 - Recomendação e gestão . . . 61
3.8 Planejamento . . . 62
3.8.1 Épicos . . . 62
3.8.1.1 Épico 1: Gestão de Acesso e Perfis de Usuário . . . 62
3.8.1.2 Épico 2: Gerenciamento de dados . . . 63
3.8.1.3 Épico 3: Desenvolvimento do *chatbot* . . . 64
3.8.1.4 Épico 4: Desenvolvimento da camada de IA . . . 65

4 DESENVOLVIMENTO DO SEI EXPERT . . . 67
4.1 Engenharia de Software . . . 67
4.2 Visão da Arquitetura Implementada . . . 67
4.3 Gestão de Configuração e Tecnologias . . . 68
4.4 Evolução do Desenvolvimento . . . 69
4.4.1 Planejamento Original . . . 69
4.4.2 Execução Realizada . . . 71
4.5 Infraestrutura e processo de entrega . . . 72

4.5.1 Versionamento de código (Git/GitHub) . . . 72
4.5.2 Ambiente de implantação . . . 73
4.6 Limitações e dívidas técnicas . . . 73

5 O SEI EXPERT . . . 75
5.1 Controle de acesso e identidade . . . 75
5.1.1 Autenticação e recuperação . . . 75
5.1.2 Cadastro com definição de papéis (*Roles*) . . . 76
5.2 *Dashboard* e visão geral . . . 76
5.3 Módulo de processos e gestão documental . . . 77
5.3.1 Listagem e organização . . . 78
5.3.2 Sistema de *tags* e categorização . . . 79
5.3.3 Detalhamento e arquivos . . . 79
5.4 Base de conhecimento (legislação) . . . 79
5.4.1 Gestão de normas . . . 79
5.4.2 Classificação por *tags* . . . 80
5.4.3 Visualização de conteúdo legal . . . 81
5.5 O *Chatbot* e análise . . . 82
5.5.1 Interface de conversação . . . 82
5.5.2 Análise contextual de processos . . . 82
5.5.3 Referência legislativa . . . 82
5.6 Administração e configuração . . . 83
5.6.1 Painéis administrativos . . . 84
5.6.2 Configurações de perfil . . . 84

6 CONCLUSÃO E TRABALHOS FUTUROS . . . 87
6.1 Síntese dos resultados . . . 87
6.2 Limitações e adaptações . . . 87
6.3 Trabalhos futuros . . . 88

REFERÊNCIAS . . . 89

1 Introdução

1.1 Contextualização

Os processos organizacionais, especialmente no âmbito do setor público, caracterizam-se por sua complexidade e volume de informações, demandando abordagens para otimizar sua análise e tomada de decisão (DUMAS et al., 2018). Um desafio recorrente enfrentado por analistas de processos é o tempo despendido em tarefas repetitivas e na consulta a extensos acervos de legislações e históricos de casos anteriores. Esta dinâmica pode resultar em inconsistências, dificultando a manutenção da padronização e a clareza na justificação das decisões (WESKE, 2019). Neste cenário, a inteligência artificial, particularmente através de modelos de linguagem de grande escala, técnicas de aprendizado de máquina e protocolos de contexto de modelo, emerge como uma tecnologia para aprimorar essas atividades.

1.2 Motivação

A crescente demanda por eficiência e transparência na administração pública exige soluções que reduzam o tempo gasto em tarefas repetitivas e minimizem erros humanos (DUMAS et al., 2018). A adoção de sistemas inteligentes pode contribuir para a padronização das análises, facilitar o uso de informações e auxiliar o processo de decisão. O avanço de modelos de linguagem, quando combinado com protocolos de interoperabilidade como o *model context protocol* (MCP), oferece uma padronização na integração de ferramentas e fontes de dados. O que permite à inteligência artificial invocar essas ferramentas e utilizá-las de forma contextualizada.

1.3 O Problema

A análise de processos, muitas vezes manual e dependente do conhecimento individual do analista, está propensa a ineficiências e falta de padronização, onde decisões podem variar entre processos semelhantes (WESKE, 2019). Um dos desafios que emergem é a subutilização do conhecimento organizacional, pois analistas despendem tempo em processos similares já solucionados por carecerem de mecanismos para recuperar o conhecimento de precedentes.

Além disso, a dependência da experiência individual leva à falta de uniformidade e padronização, resultando em uma variabilidade nas decisões, o que dificulta manter um padrão de qualidade e conformidade. Isso se soma à dificuldade de fornecer uma justifi-

cativa e rastreabilidade, uma vez que a fundamentação das decisões exige que o analista consulte manualmente fontes, como legislações e documentos, tornando a auditoria e o rastreamento tarefas demoradas.

Também destaca-se a dificuldade em gerenciar o grande volume de informações envolvido na análise de processos organizacionais, especialmente em instituições públicas que lidam com inúmeros documentos e legislações correlacionadas (DUMAS et al., 2018). Esse excesso de dados eleva ainda mais a complexidade e o tempo das análises.

Por fim, a carência de mecanismos que identifiquem e recomendem especialistas com base em suas experiências e decisões anteriores impede a distribuição das análises entre os profissionais mais capacitados em determinados tipos de processos,o que contribui para a sobrecarga de analistas.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo deste trabalho é projetar e desenvolver o nível 1 do sistema SEI Expert, uma solução de software que utiliza modelos de linguagem de grande escala (LLMs) e controle de contexto para auxiliar na análise de processos organizacionais. O trabalho visa diminuir a complexidade e a repetitividade presente na análise de processos, identificando perfis e recomendando decisões.

1.4.2 Objetivos Específicos (Desenvolvimento e Implementação)

1.4.2.1 Implementação da Arquitetura Fullstack

Construir e integrar a arquitetura de software composta por um *frontend* reativo em React, um *backend* em FastAPI e uma camada de serviços de inteligência artificial, utilizando contêineres Docker para garantir a portabilidade e escalabilidade do sistema.

1.4.2.2 Construção da Camada de Inteligência Artificial

Desenvolver e codificar o Validador Semântico e o Orquestrador *Model Context Protocol* (MCP), implementando a técnica de *Retrieval-Augmented Generation* (RAG) para permitir análises processuais fundamentadas e contextualizadas.

1.4.2.3 Desenvolvimento do Módulo de Gestão e Acesso

Materializar o sistema de permissões baseado em papéis (RBAC), implementando os perfis de Analista, Especialista e Gestor, garantindo a segurança e o controle de acesso às funcionalidades de análise de processos.

1.4.2.4 Desenvolvimento das Interfaces Funcionais

Codificar as interfaces de usuário responsivas, transformando os requisitos de usabilidade em componentes funcionais que permitam a interação via *chat* inteligente e a visualização clara de documentos e legislações.

1.4.2.5 Implementação do Mecanismo de Ingestão de Dados

Desenvolver o *parser* para processamento de arquivos `.zip`, permitindo a extração, vetorização e armazenamento de documentos processuais no banco de dados *pgvector* para consultas semânticas.

1.4.2.6 Execução e Gestão Ágil do Projeto

Executar o desenvolvimento do software seguindo a metodologia ágil, realizando ciclos semanais de *sprints* para permitir o refinamento contínuo das funcionalidades e a entrega de um Produto Mínimo Viável (MVP) operacional.

1.5 Metodologia

O processo metodológico para o desenvolvimento do trabalho foi estruturado nas seguintes etapas:

Revisão bibliográfica e fundamentação Teórica: A primeira etapa consistiu no estudo do tema e das tecnologias envolvidas. Foram revisadas literaturas sobre gerenciamento de processos de negócio (BPM), inteligência artificial, modelos de linguagem amplos (do inglês, *Large Language Models* - LLMs), aprendizado de máquina (do inglês, *Machine Learning* - ML), e protocolos de interoperabilidade como o *Model Context Protocol* (MCP).

Definição do problema e dos requisitos: A partir da fundamentação teórica e de discussões semanais com a orientação, o problema de pesquisa foi definido. O levantamento de requisitos funcionais e não funcionais foi realizado através de técnicas de *brainstorming* e introspecção, onde o orientador atuou como cliente e especialista para validar as necessidades do sistema. Os requisitos foram então organizados em quatro níveis e priorizados utilizando a metodologia MoSCoW.

Design e concepção do artefato: Etapa onde a solução foi projetada. Foi definida uma arquitetura de sistema, detalhando seus componentes, como a interface com o usuário, o *backend*, a base de dados e a camada de IA. Houve a preparação para uma futura integração com sistemas externos, como o SEI, envolvendo o estudo do manual de desenvolvimento de módulos do referido sistema para garantir a viabilidade da proposta.

Demonstração e Avaliação Inicial: Para demonstrar a viabilidade e os conceitos da proposta durante a primeira fase (TCC 1), foi realizada a prototipação de média fidelidade das principais interfaces de usuário. Esta etapa permitiu uma avaliação do design e dos fluxos de interação, validando a usabilidade e a coerência da solução proposta.

Construção e Desenvolvimento: Para a etapa de concretização do artefato de software referente ao nível 1 do sistema planejado, adotou-se uma metodologia experimental e incremental, fundamentada nos princípios do *Lean Software Development* e práticas do *Scrum*. O desenvolvimento foi fracionado em ciclos semanais (*Sprints*), permitindo a adaptação do escopo,especialmente na camada de inteligência artificial, conforme os desafios técnicos e as necessidades de engenharia emergiam durante a execução.

1.5.1 Cronograma

As Tabelas 1 e 2 apresentam, respectivamente, o cronograma de atividades realizadas na primeira fase (Planejamento) e na segunda fase (Desenvolvimento) do trabalho. A primeira fase (TCC 1) focou no planejamento e modelagem, enquanto a segunda fase (TCC 2) dedicou-se ao desenvolvimento e implementação do sistema.

Mês	**Atividades Realizadas**
Abril	Análise inicial do tema e escopo; Definição do problema de pesquisa e levantamento bibliográfico inicial.
Maio	Estruturação da proposta de solução; Elaboração dos diagramas de arquitetura inicial.
Junho	Definição do conceito de MCP (*Model Context Protocol*); Detalhamento dos requisitos funcionais e não funcionais.
Julho	Escrita da monografia do TCC 1; Revisão da proposta e entrega final.

Tabela 1 – Cronograma de Atividades - TCC 1

Fonte: Autor.

1.6 Estruturação do trabalho

A estrutura do presente trabalho se divide nos seguintes capítulos:

1. **Introdução:** Apresenta a contextualização, motivação, problema, objetivos, metodologia e estrutura do trabalho.
2. **Fundamentação Teórica:** Reúne conceitos sobre gerenciamento e análise de processos organizacionais, inteligência artificial, aprendizado de máquina, processamento de linguagem natural, modelos de linguagem de grande escala, RAG e *Model Context Protocol*.

Mês	**Atividades Realizadas**
Agosto	Revisão da arquitetura para microsserviços; Configuração do ambiente de desenvolvimento (Docker, Repositórios).
Setembro	Implementação da infraestrutura base; Desenvolvimento do módulo de identidade (Login/Auth) e Banco de Dados.
Outubro	Desenvolvimento do *frontend* e gestão de processos; Implementação da ingestão de arquivos (*parser* .zip).
Novembro	**Foco em IA:** Desenvolvimento do Validador Semântico e Orquestrador MCP; Integração com OpenAI e ajustes de RAG.
Dezembro	Integração final e testes do MVP;

Tabela 2 – Cronograma de Atividades - TCC 2

Fonte: Autor.

3. **Planejamento do SEI Expert:** Descreve a arquitetura planejada, funcionalidades previstas, perfis de usuários, camada de IA, requisitos e os protótipos desenhados inicialmente para o sistema.

4. **Desenvolvimento do SEI Expert:** Detalha o processo de engenharia de software, a gestão de configuração, as tecnologias utilizadas e a narrativa da execução técnica, confrontando o planejamento inicial com a realidade das *sprints*.

5. **O Sistema SEI Expert:** Apresenta o produto final desenvolvido, detalhando as interfaces, o fluxo de uso real e as funcionalidades operacionais entregues.

6. **Conclusão:** Apresenta o encerramento do trabalho, destacando as contribuições, o comparativo entre o planejado e o executado, as limitações técnicas encontradas e as possibilidades de trabalhos futuros.

2 Fundamentação Teórica

Este capítulo estabelece as bases teóricas necessárias para a compreensão do sistema proposto. O objetivo é fornecer uma base conceitual para os tópicos abordados, dentre eles a engenharia de software, o gerenciamento de processos organizacionais, as tecnologias de inteligência artificial como aprendizagem de máquina, aprendizagem profunda, processamento de linguagem natural, modelos de linguagem de grande escala e *model context protocol* (MCP), e o Sistema Eletrônico de Informações (SEI).

2.1 Engenharia de Software

A engenharia de *software* é uma disciplina de engenharia que se preocupa com todos os aspectos da produção de *software*, desde a análise de requisitos até a manutenção contínua do sistema. Ela envolve a aplicação sistemática e organizada de métodos e técnicas para desenvolver *software* de maneira eficiente e confiável. A disciplina não considera apenas o código, mas todo o ambiente de documentação associada e os dados de configuração necessários para o funcionamento correto do sistema. Onde um sistema real é, muitas vezes, composto por diversos programas e seus arquivos de configuração (SOMMERVILLE, 2007).

A importância desta disciplina consiste na dependência, cada vez maior, da sociedade em sistemas de *software*, priorizando a capacidade de produzir sistemas confiáveis de forma rápida. E também, a longo prazo, a utilização de métodos e técnicas da engenharia de *software* para o desenvolvimento de sistemas é mais economicamente viável em vez de abordagens de programação pessoal (SOMMERVILLE, 2007). Isso se justifica porque, para a maioria dos sistemas, a maior parte dos custos está associada às alterações e à evolução do software após sua implementação.

As seções a seguir detalharão alguns conceitos de engenharia de *software* que são necessários para o entendimento do trabalho. Serão abordados os conceitos de engenharia de requisitos, design e arquitetura de *software*, metodologias de desenvolvimento de *software* e prototipação.

2.1.1 Engenharia de requisitos

A Engenharia de Requisitos é o processo de descobrir, analisar, documentar e verificar os serviços que um sistema deve fornecer e as restrições sob as quais deve operar, configurando os requisitos de um sistema. Esses requisitos refletem as necessidades do cliente para um sistema que serve determinado propósito. Este é um processo importante na

engenharia de *software*, pois erros podem levar a custos de retrabalho se forem descobertos em fases posteriores do desenvolvimento (SOMMERVILLE, 2007).



Figura 1 – Etapas da engenharia de requisitos.

Fonte: Reinehr (2020)

As etapas da engenharia de requisitos, vistas na figura 1, não se limitam a um modelo de desenvolvimento específico, elas são atividades que devem ser realizadas no desenvolvimento do produto, independentemente do ciclo de vida utilizado (REINEHR, 2020). As atividades do desenvolvimento de requisitos são detalhadas a seguir.

1. Elicitação: É a atividade onde os engenheiros de software trabalham diretamente com os clientes e usuários finais do sistema. O objetivo é descobrir informações sobre o domínio da aplicação, os serviços que o sistema deve fornecer, o desempenho necessário, as restrições de *hardware*, entre outros. Esta atividade envolve pessoas, ou *stakeholders*, que incluam não apenas os usuários finais, mas pessoas que possam ter uma influência sobre os requisitos do sistema (SOMMERVILLE, 2007).

2. Análise: Tem como objetivo identificar e resolver possíveis conflitos. Esses conflitos podem surgir tanto das diferentes visões dos *stakeholders* quanto de incompatibilidades entre os próprios requisitos. Durante esta etapa, requisitos de alto nível, expressos como objetivos de negócio, são decompostos em níveis detalhados para as etapas subsequentes do desenvolvimento. Além disso, atributos importantes são definidos para cada requisito, como seu tipo e prioridade (REINEHR, 2020).

3. Especificação: É a etapa dedicada a representar os requisitos de usuário e de sistema em um documento, de forma que eles possam perdurar ao longo do tempo para serem

verificados e validados posteriormente. Essa atividade envolve o uso de diversos formatos, como textos, diagramas e tabelas, para registrar de forma clara o que o sistema deve fazer, com o nível de detalhamento desta etapa dependendo do contexto do projeto (REINEHR, 2020).

4. Validação: É o processo de verificar se os requisitos realmente definem o sistema que o cliente deseja. Durante o processo de validação, diferentes verificações devem ser realizadas para garantir a qualidade, inclui checar a validade, a consistência, a completude, o realismo e a verificabilidade. Para realizar essas verificações, são utilizadas técnicas como a prototipagem, revisão de requisitos e a geração de casos de teste (SOMMERVILLE, 2007).

5. Gerenciamento: O gerenciamento de requisitos é o processo contínuo de controlar as mudanças nos requisitos de um sistema durante todo ciclo de vida do produto. Seu objetivo é estabelecer formas de rastrear os requisitos e facilitar a análise de impacto de mudanças de decisão. Para que a análise de impacto de uma mudança seja precisa, deve-se manter a rastreabilidade, ou seja, conhecer a fonte de cada requisito, suas relações com outros artefatos do projeto. Essa rastreabilidade permite estimar os recursos e o tempo necessários para realizar uma mudança e identificar quais artefatos precisam ser alterados (SOMMERVILLE, 2007).

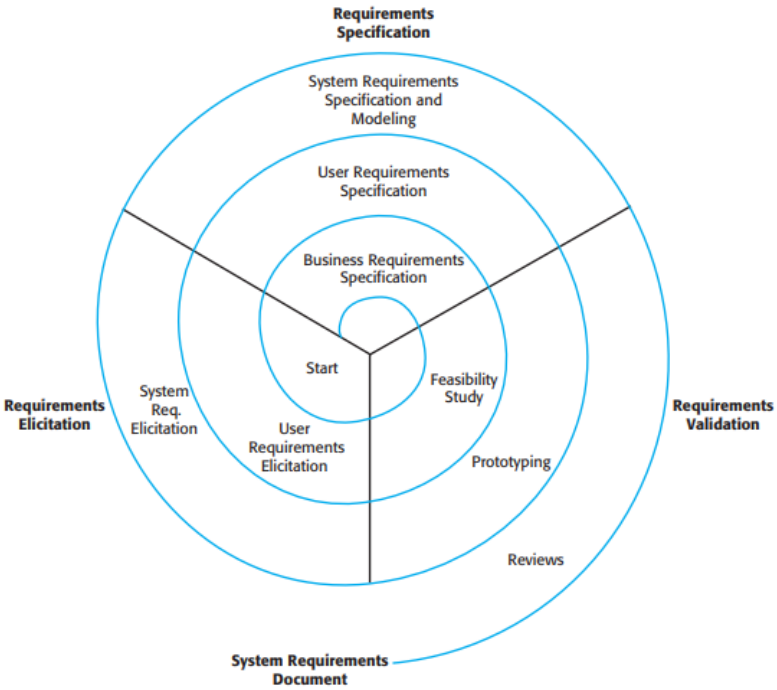


Figura 2 – Visão em espiral dos processos da engenharia de requisitos.

Fonte: Sommerville (2007)

Enquanto a separação conceitual das atividades de engenharia de requisitos (elicitação, análise, especificação e validação) é útil para entender o propósito de cada, na prática, o processo não ocorre linearmente. Para representar a dinâmica dessas atividades, Sommerville (2007) propõe uma visão em espiral. A figura 2 representa a engenharia de requisitos como um processo iterativo no qual as atividades são intercaladas. O processo começa no centro da espiral e evolui para fora. O resultado final deste processo é a criação de um documento de requisitos do sistema.

2.1.2 Metodologias de desenvolvimento de *software*

O desenvolvimento de *software* envolve um conjunto de atividades associadas que levam a um produto de *software*. Embora existam muitos modelos de desenvolvimento de *software* diferentes, eles geralmente incorporam quatro atividades que são comuns a todos: especificação, *design* e implementação, validação e evolução (SOMMERVILLE, 2007).

1. Especificação de *software*: Atividade onde deve ser definida as restrições e funcionalidades.

2. *Design* e implementação de *software*: Ocorre o desenvolvimento do *software* que atenda as especificações.

3. Validação de *software*: É a atividade em que o *software* deve ser validado para garantir o que o cliente quer.

4. Evolução de *software*: Atividade onde o software deve evoluir para atender as mudanças de necessidade do cliente.

Abordagens tradicionais, como o modelo em cascata, apresentam essas atividades como fases sequenciais e distintas. Em contrapartida, uma abordagem mais moderna e atualmente a mais comum é o desenvolvimento incremental. Neste modelo, as atividades são intercaladas, e o sistema é desenvolvido em uma série de versões (incrementos), com cada incremento adicionando novas funcionalidades (SOMMERVILLE, 2007).

O desenvolvimento incremental pode ser **orientado a um plano** , onde os incrementos do sistema são identificados previamente, ou pode ser **ágil**, onde os incrementos iniciais são identificados e o desenvolvimento dos incrementos posteriores depende do progresso e das prioridades do cliente. A diferença entre as duas está em como elas integram as atividades do processo de software, ilustrada pela figura 3, a abordagem orientada a um plano identifica estágios separados, com documentos formais dos resultados de cada etapa, em que os resultados são utilizados como base para planejar a etapa seguinte. Em contraste, as abordagens ágeis consideram o *design* e a implementação como as principais atividades, incorporando outras atividades, como o levantamento de requisitos e os testes,

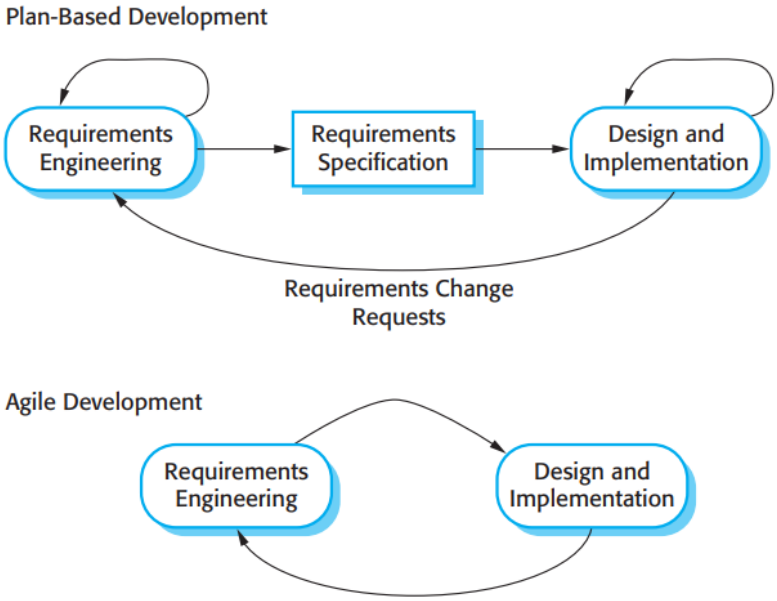


Figura 3 – Desenvolvimentos orientado a um plano e ágil.

Fonte: Sommerville (2007)

diretamente nelas. Dessa forma, a iteração ocorre através das atividades, permitindo que os requisitos e o projeto sejam desenvolvidos juntos, em vez de separadamente.

2.1.3 Design e arquitetura de *software*

O design e arquitetura são focados em compreender como um sistema deve ser organizado e em projetar sua estrutura geral. Esta fase atua como um elo entre a engenharia de requisitos e o design, pois identifica os componentes estruturais do sistema e as relações entre eles. O resultado do processo de design de arquitetura é um modelo arquitetural que descreve como o sistema é organizado como um conjunto de componentes.

A definição da arquitetura afeta os atributos de qualidade de um sistema, como desempenho, robustez, distributividade e manutenibilidade. Enquanto os componentes implementam os requisitos funcionais, são as decisões arquiteturais ,a forma como esses componentes são organizados e se comunicam, que determinam predominantemente se o sistema pode atender aos seus requisitos não funcionais. O design de arquiteturas de software pode ser abordado em dois níveis de abstração (SOMMERVILLE, 2007):

- *Architecture in the small*: Refere-se à arquitetura de programas individuais, focando em como um programa é decomposto em componentes.
- *Architecture in the large*: Diz respeito à arquitetura de sistemas complexos, que incluem múltiplos sistemas, programas e componentes, muitas vezes distribuídos.

A escolha de uma estrutura arquitetural deve depender diretamente dos requisitos não funcionais. Se a manutenibilidade é a prioridade, a arquitetura deve ser projetada com componentes modulares, que possam ser alterados facilmente. Normalmente, modelos arquiteturais são representados através de diagramas de blocos simples, que apresentam uma visão de alto nível da estrutura do sistema, facilitando a comunicação. Embora informais, esses diagramas são eficazes durante o processo de design para discutir o sistema com *stakeholders* e para o planejamento do desenvolvimento (SOMMERVILLE, 2007).

2.1.4 Prototipação

A prototipação consiste na criação de uma versão inicial de um sistema. O objetivo principal de um protótipo não é ser um produto final, mas sim uma ferramenta para demonstrar conceitos, e obter um entendimento mais aprofundado sobre o problema e suas possíveis soluções. Um protótipo de *software* pode ser empregado em diferentes fases do processo de desenvolvimento para antecipar e refinar requisitos e decisões de design. Segundo Sommerville (2007) suas principais aplicações são:

- **Na engenharia de requisitos:** Um protótipo é uma ferramenta para a elicitação e validação dos requisitos do sistema. Ao interagir, os usuários podem ter novas ideias para funcionalidades e identificar erros nas especificações iniciais.
- **No processo de design do sistema:** A prototipação pode ser utilizada para explorar certas soluções de *software* e para apoiar o design da interface do usuário.

O processo de desenvolvimento de um protótipo deve seguir etapas bem definidas, começando pelo estabelecimento de objetivos. Estes objetivos podem variar, desde validar a interface do usuário até demonstrar a viabilidade técnica de uma aplicação. A etapa seguinte envolve a definição da funcionalidade do protótipo, decidindo quais recursos incluir e quais deixar de fora para acelerar a entrega e reduzir custos. E a etapa de avaliação que permite que os *stakeholders* interajam com o protótipo e forneçam *feedbacks* (SOMMERVILLE, 2007).

2.2 Gerenciamento e Análise de Processos Organizacionais

Segundo Dumas et al. (2018) uma organização, seja pública ou privada, precisa lidar com uma grande quantidade de processos. Um processo organizacional é uma estrutura que vai além de uma sequência de atividades, ele é composto por pontos de decisão que afetam a maneira como o processo é executado. Além disso, um processo também envolve a interação de atores, que podem ser humanos, organizações ou sistemas de software, e o manuseio de objetos, tanto físicos (como equipamentos) quanto não físicos (como

documentos eletrônicos), podendo ser observado na imagem 4. Esse conjunto de fatores gera um resultado que, se positivo, tenha valor para, no mínimo, um cliente (DUMAS et al., 2018).

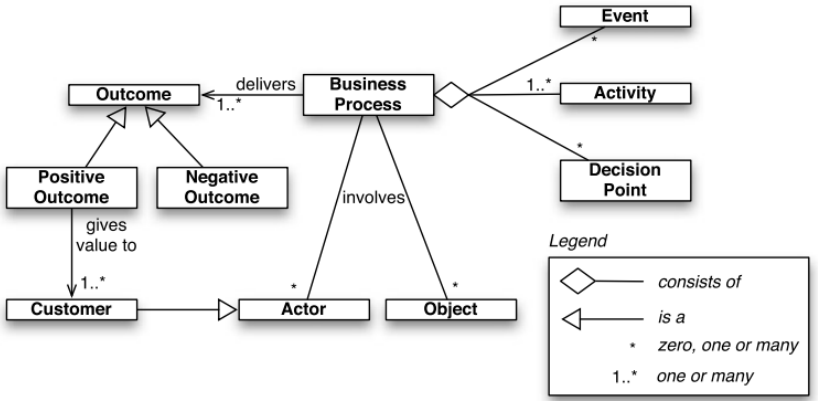


Figura 4 – Composição de um processo organizacional.

Fonte: Dumas et al. (2018)

O gerenciamento de processos de negócio (do inglês, *Business Proccess Management* - BPM) é uma abordagem de gestão que trata os processos de uma organização como ativos estratégicos que precisam ser modelados, analisados, aprimorados e gerenciados para agregar valor e alcançar os objetivos estratégicos da organização. A ideia do BPM é que, para uma organização ser eficiente e competitiva, ela precisa ter um entendimento de como seu trabalho é realizado. Em vez de focar apenas no resultado final, o BPM foca em como o resultado é alcançado, ou seja, no processo em si (DUMAS et al., 2018). Para fazer isso, o BPM é tipicamente estruturado em um ciclo de vida contínuo, que inclui as seguintes fases:

1. Identificação e descoberta de processos: É a fase inicial onde os processos na organização são identificados e mapeados. O resultado é um modelo do processo como ele é atualmente (o modelo *as-is*).

2. Análise de processos: Nesta fase, o modelo *as-is* é analisado para identificar problemas, gargalos e ineficiências.

3. Redesenho de processos: Com base na análise, são propostas mudanças para melhorar o processo. O resultado é um novo modelo de processo, o *to-be*, que representa como o processo deveria funcionar.

4. Implementação de processos: Onde ocorre execução das mudanças propostas no redesenho, isso pode incluir mudanças organizacionais e mudanças técnicas.

5. Monitoramento e controle de processos: Uma vez que o processo está em execução, ele é monitorado para garantir que está atingindo os objetivos esperados. Os dados coletados nesta fase servem de entrada para um novo ciclo..

Desta maneira, o BPM fornece uma estrutura para que as organizações possam superar a subjetividade e a inconsistência da análise manual. Contudo, em instituições de grande porte e no setor público, esses processos tendem a se tornar complexos e burocráticos, envolvendo vastas quantidades de documentos, regulamentos e interações humanas. Analisá-los manualmente com o objetivo de otimizá-los e padronizá-los é uma tarefa lenta, subjetiva e difícil de escalar. A ausência de ferramentas adequadas para apoiar os analistas compromete a produtividade, aumenta a inconsistência nas decisões e dificulta a justificativa das ações adotadas, reforçando a necessidade de abordagens mais instrumentadas e inteligentes (DUMAS et al., 2018).

2.3 Inteligência artificial

Segundo Russell e Norvig (2020) o campo de inteligência artificial (IA) foca na criação de agentes que percebem o ambiente e atuam para maximizar suas chances de sucesso. Essa busca se ramifica em subáreas, como ilustrado na figura 5. Dentre as principais áreas, destacam-se o aprendizado de máquina, que permite aos sistemas aprenderem com dados, o processamento de linguagem natural, que habilita a compreensão e geração da linguagem humana, a visão computacional, para a interpretação de imagens e vídeos, e a robótica, que integra a inteligência em agentes físicos (RUSSELL; NORVIG, 2020).

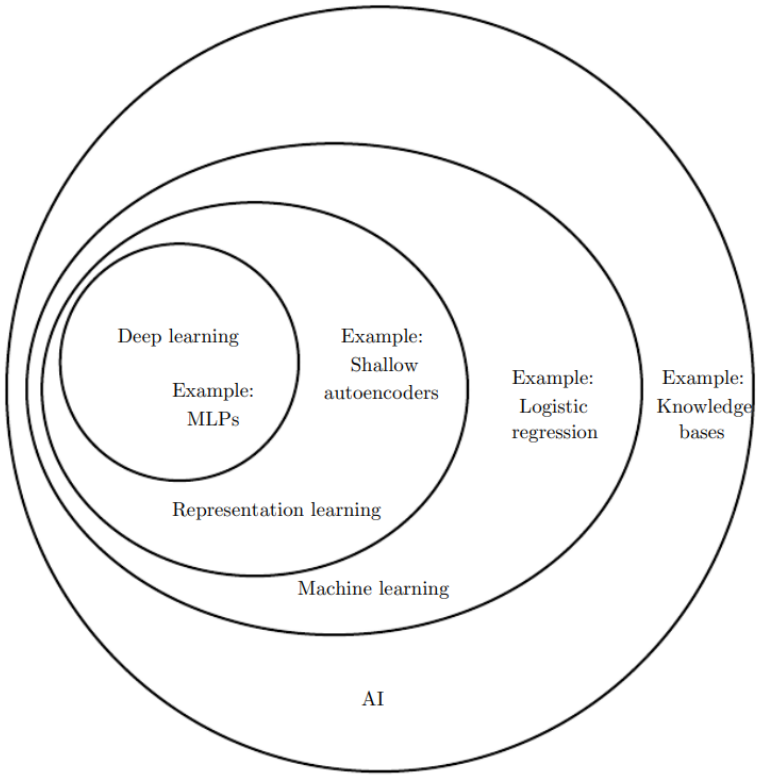


Figura 5 – Diagrama de Venn representando subáreas da IA

Fonte: Goodfellow, Bengio e Courville (2016)

2.3.1 Aprendizagem de máquina

O Aprendizado de máquina é um subcampo da inteligência artificial que se concentra no desenvolvimento de algoritmos que permitem aos computadores aprender a partir de dados sem serem explicitamente programados para cada tarefa específica (RUSSELL; NORVIG, 2020). Em essência, em vez de desenvolver o agente para cada cenário específico, no aprendizado de máquina fornecemos ao agente uma grande quantidade de dados e ele aprende os padrões, relações e estruturas inerentes a esses dados para realizar tarefas como predição, classificação ou reconhecimento de padrões (MITCHELL, 1997). Fazer o agente aprender é o meio de contornar o fato de que não é possível, para o desenvolvedor, prever as situações em que o agente vai se encontrar, sendo necessário o uso do aprendizado para fazer com que o agente se adapte a mudanças, justamente tendo como objetivo generalizar esse aprendizado para novos dados não vistos anteriormente (RUSSELL; NORVIG, 2020).
Existem três tipos de *feedback* que definem os três principais métodos de aprendizado:

A **aprendizagem não supervisionada**, nesse método o agente aprende padrões, mesmo que não seja fornecida uma indicação externa, sendo a aprendizagem realizada por meio da descoberta de regularidades nos dados de entrada, o principal conceito utilizado para descobrir essas regularidades é o de similaridade, que é definido por uma medida de distância. A atividade mais comum desse método de aprendizado é o agrupamento, ou *clustering* em inglês, que detecta possíveis agrupamentos dos dados de entrada (RUSSELL; NORVIG, 2020; ROCHA; CORTEZ; NEVES, 2008).

A **aprendizagem por reforço**, onde exige a presença de uma indicação externa, embora não seja dada a resposta correta, a aprendizagem ocorre através de "punições"e "recompensas"mediante as respostas fornecidas pelo modelo em certas situações (ROCHA; CORTEZ; NEVES, 2008).

A **apredizagem supervisionada**, onde há a presença de uma indicação externa, ou seja, é dada uma resposta correta para cada situação. A aprendizagem acontece através de casos de treino compostos por uma entrada e por uma saída desejada(ROCHA; CORTEZ; NEVES, 2008).

2.3.1.1 *Clustering* (Agrupamento)

O *clustering*, ou agrupamento, é uma técnica de aprendizagem não supervisionada. O objetivo principal do *clustering* é particionar um conjunto de dados em grupos, ou *clusters* do inglês, de forma que os itens dentro de um mesmo *cluster* sejam similares entre si, e itens em *clusters* diferentes sejam distintos (MITCHELL, 1997).

No *clustering* não há rótulos pré-definidos ou saídas corretas fornecidas nos dados de treinamento. Em vez disso, o algoritmo explora a estrutura inerente aos dados

para descobrir agrupamentos. A similaridade entre os itens é definida usando uma medida de distância (por exemplo, distância euclidiana) em um espaço de características. O *clustering* é amplamente utilizado para análise exploratória de dados, detecção de anomalias, segmentação de clientes, entre outras aplicações onde se busca encontrar padrões ou categorias nos dados.(RUSSELL; NORVIG, 2020)

Desta maneira, cada item de dado é representado como um vetor em um espaço de características. O objetivo é particionar o conjunto de dados em k *clusters*, como na figura 6, de modo a minimizar uma função de custo. Uma das funções de custo mais vistas é a soma dos erros quadrados, que corresponde à soma das distâncias euclidianas ao quadrado de cada ponto de dado até o centroide do *cluster* ao qual ele foi atribuído. Ao minimizar essa função, o algoritmo busca criar *clusters* que sejam compactos e ao mesmo tempo distintos uns dos outros. (RUSSELL; NORVIG, 2020)

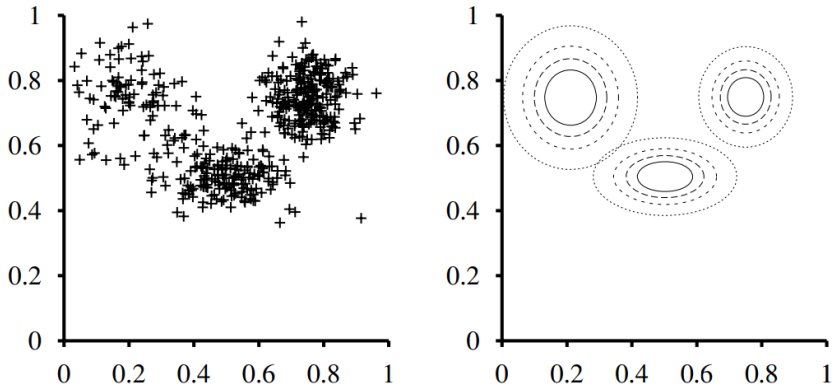


Figura 6 – Exemplo de *clusters*

Fonte: Russell e Norvig (2020)

2.3.2 Aprendizagem profunda

A aprendizagem profunda (do inglês, *deep learning*) é uma forma de aprendizado de máquina, observado na figura 5. Os modelos de aprendizagem profunda são construídos a partir de redes neurais artificiais. Uma rede neural é um modelo matemático inspirado no cérebro, composto por unidades de processamento conectadas, chamadas neurônios, que são organizadas em uma série de camadas. Cada neurônio realiza uma computação, ele recebe um conjunto de valores de entrada, calcula uma soma ponderada desses valores, adiciona um viés e aplica uma função de ativação ao resultado. Essa função de ativação determina a saída do neurônio, que servirá como entrada para os neurônios da camada seguinte. (GOODFELLOW; BENGIO; COURVILLE, 2016)

A profundidade refere-se ao uso de múltiplas camadas organizadas de forma hierárquica, incluindo uma camada de entrada, camadas ocultas e uma camada de saída. Essa arquitetura é o que possibilita o aprendizado, no qual cada camada aprende repre-

sentações progressivamente mais abstratas dos dados. Por exemplo, na figura 7 a camada de entrada recebe os *pixels* brutos da imagem, enquanto as camadas ocultas extraem gradualmente padrões, como bordas e contornos, partes de objetos e, por fim, suas identidades (GOODFELLOW; BENGIO; COURVILLE, 2016). Ou seja, cada camada aprende a transformar a representação de dados da camada anterior em uma representação mais complexa e abstrata, permitindo que o modelo realize tarefas complexas que antes exigiam conhecimento detalhado e específico para cada problema. (JURAFSKY; MARTIN, 2025).

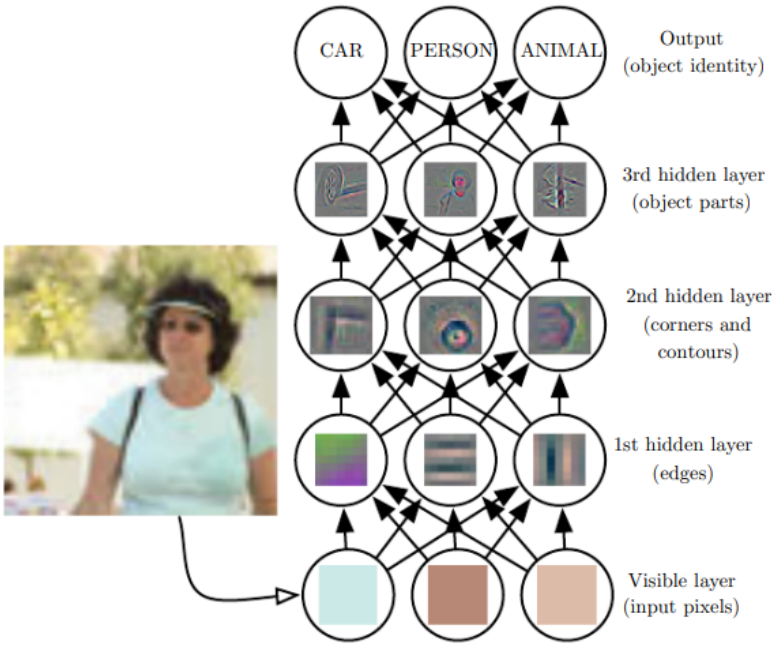


Figura 7 – Ilustração representando um modelo de *deep learning*

Fonte: Goodfellow, Bengio e Courville (2016)

Em suma, essa estrutura em camadas define a aprendizagem profunda, uma forma de aprendizado de máquina que permite aos computadores aprenderem a partir da experiência e entenderem as coisas em uma hierarquia de conceitos. A essência do *deep learning* está nessa hierarquia, onde conceitos mais complexos são aprendidos com base em conceitos mais simples.

2.3.3 Processamento de linguagem natural

O processamento de linguagem natural (PLN) é um campo interdisciplinar da ciência da computação e da inteligência artificial que tem como objetivo fundamental capacitar as máquinas a processar, compreender e gerar a linguagem humana (JURAFSKY; MARTIN, 2009). A linguagem, sendo o principal veículo para a comunicação e registro de conhecimento, é um desafio para a inteligência artificial, uma vez que sua compreensão exige um grande conhecimento sobre o mundo. Com a crescente digitalização da informação e a explosão de dados textuais disponíveis, o PLN se tornou indispensável para analisar e extrair significado de grandes volumes de texto de forma automática e eficiente (MANNING; SCHüTZE, 1999; RUSSELL; NORVIG, 2020).

O funcionamento do PLN pode ser compreendido em diferentes níveis de análise linguística. Em um nível fundamental, a análise morfológica consiste na estrutura das palavras e seus componentes, como raízes, prefixos e sufixos. Em seguida, a análise sintática examina a estrutura gramatical das sentenças, que busca determinar como as palavras se combinam para formar orações e frases válidas. A um nível mais profundo, a análise semântica foca na extração do significado das palavras e sentenças, lidando com ambiguidades para construir uma representação do sentido do texto. Por fim, a análise pragmática considera o contexto amplo, incluindo a intenção do falante e a forma como as sentenças se conectam para formar um texto coeso(JURAFSKY; MARTIN, 2009).

Historicamente, as abordagens de PLN eram dominadas por sistemas simbólicos que dependiam de conjuntos de regras linguísticas elaboradas manualmente. A partir da década de 1990, o PLN passou por uma transformação com a ascensão dos métodos estatísticos. Essas técnicas utilizam coleções de textos, conhecidas como corpora, para aprender padrões e probabilidades da linguagem de forma empírica. Esta mudança permitiu o desenvolvimento de sistemas com melhor desempenho em tarefas reais (MANNING; SCHüTZE, 1999). Atualmente, o campo é definido por modelos de *deep learning,* como a arquitetura Transformer, que possui uma capacidade de capturar as complexas relações presentes nos dados linguísticos (JURAFSKY; MARTIN, 2025).

A arquitetura Transformer representou uma mudança no PLN ao abandonar a natureza sequencial das redes neurais em favor de um modelo baseado inteiramente em mecanismos de atenção (VASWANI et al., 2017). Um transformer é uma rede neural com uma estrutura específica chamada *self-attention*. Esta estrutura pode ser pensada como

uma maneira de construir representações contextuais do significado de um símbolo ao integrar informações dos símbolos vizinhos, ajudando o modelo a entender como esses símbolos se relacionam em grandes extensões.(JURAFSKY; MARTIN, 2025)

O sucesso e a eficiência dessa arquitetura a tornaram a fundação para os atuais modelos de linguagem de grande escala, como o BERT (do inglês, Bidirectional Encoder Representations from Transformers) e a família de modelos GPT (do inglês, Generative Pre-trained Transformer), que revolucionaram o campo (VASWANI et al., 2017).

2.3.4 Modelos de linguagem de grande escala

Modelos de linguagem de grande escala (LLMs) são modelos de linguagem neurais baseados na arquitetura *transformer*, que são pré-treinados em vastas quantidades de dados textuais. O termo "grande escala"refere-se tanto à grande quantidade de dados de treino quanto ao número massivo de parâmetros que esses modelos possuem. O objetivo de uma LLM, descrevendo de maneira simples, é prever o próximo *token* em uma sequência, dada a sequência de palavras anteriores (JURAFSKY; MARTIN, 2025). Essa tarefa, quando realizada em grande escala, permite que os modelos aprendam uma grande quantidade de informações sobre a linguagem.

Os *transformers* foram essenciais para o desenvolvimento das LLMs. Antes, modelos recorrentes eram comuns, mas tinham dificuldade em lidar com dependências de longo alcance no texto e eram mais lentos para treinar devido à sua natureza sequencial (VASWANI et al., 2017). Para que seja possível entender melhor os *transformers*, primeiro será necessário explicar os mecanismos de atenção e a arquitetura.

O mecanismo de atenção é projetado para permitir que o modelo pondere a importância relativa de diferentes *tokens* em uma sequência ao gerar uma representação para cada um deles. Essencialmente, ele funciona mapeando uma (*query*) a um conjunto de pares chave-valor (*key-value*), onde a saída é uma média ponderada dos valores, e o peso de cada valor é calculado com base na compatibilidade entre a *query* e sua chave correspondente. A arquitetura utiliza o componente *multi-head attention*, que executa o processo de atenção em paralelo múltiplas vezes. Quando este mecanismo é aplicado para relacionar *tokens* dentro da mesma sequência, é chamado de auto-atenção (self-attention), que permite ao *transformer* capturar dependências complexas entre palavras, independentemente de sua distância na frase.

A arquitetura é composta por um codificador (*encoder*) e um decodificador (*decoder*), que estão representados pela figura 8. O *encoder* consiste em uma pilha de camadas idênticas, sendo que cada camada é composta por duas subcamadas. A primeira subcamada é um mecanismo de *multi-head self-attention* que, como descrito anteriormente, permite a cada *token* da sequência de entrada se relacionar a todos os demais, capturando

dependências contextuais. A segunda subcamada é uma rede *feed-forward*, aplicada a cada posição da sequência, com o objetivo de transformar não linearmente as representações obtidas na etapa anterior. Cada uma dessas subcamadas é acompanhada por uma conexão residual que soma sua entrada com sua saída, seguida de uma normalização por camadas. Ao final das camadas do *encoder*, obtém-se uma representação da sequência de entrada, na forma de uma matriz de vetores, para cada *token*. Esses vetores codificam tanto o conteúdo semântico quanto as relações de dependência entre os *tokens*, e são repassados como entrada para o *decoder*, que utilizará para gerar a sequência de saída.(VASWANI et al., 2017; TUNSTALL; WERRA; WOLF, 2022)

O *decoder*, que também é composto por uma pilha de camadas idênticas, contém três subcamadas. A primeira é uma subcamada de *masked multi-head attention* que, diferentemente do *encoder*, aplica uma máscara para impedir que um token da saída "olhe"posições futuras, garantindo que a predição de cada *token* dependa apenas dos anteriores já gerados. A segunda subcamada é um mecanismo de atenção cruzada (*cross-attention*), implementado por uma *multi-head attention* que utiliza como *queries* as saídas da subcamada anterior e como *keys* e *values* as representações geradas pelo *encoder*. Essa subcamada é fundamental para que o *decoder* possa consultar a entrada codificada, permitindo que cada *token* gerado na saída utilize informações de todos os *tokens* da entrada para sua predição. A terceira subcamada é uma rede *feed-forward* idêntica à utilizada no *encoder* (VASWANI et al., 2017).

Ao final da pilha de camadas do *decoder*, a saída é um vetor de pontuações numéricas (*logits*). Para transformar essa representação vetorial em um *token* de saída, o resultado passa por uma camada linear, que atua como um classificador, seguida por uma função de ativação *softmax*. A função *softmax* converte as pontuações numéricas em uma distribuição de probabilidade sobre todo o vocabulário conhecido pelo modelo. O *token* com a maior probabilidade é então selecionado como saída, este processo é executado de forma regressiva, a palavra gerada em um passo é alimentada como entrada para o próximo passo do *decoder*, que então prediz o próximo *token* da sequência, continuando até que um *token* de "fim de sequência"seja gerado. (VASWANI et al., 2017; TUNSTALL; WERRA; WOLF, 2022)

2.3.5 Geração Aumentada por Recuperação

A Geração Aumentada por Recuperação (do inglês, *Retrieval Augmented Generation* - RAG) é uma técnica que combina os pontos fortes dos modelos de linguagem com a recuperação de informação de uma fonte de conhecimento externa. Em vez de depender apenas do conhecimento aprendido durante o treinamento, os modelos RAG recuperam documentos ou passagens relevantes de um corpus grande e usam essas informações recuperadas como contexto adicional ao gerar uma resposta, visto na figura 9 (LEWIS et al.,

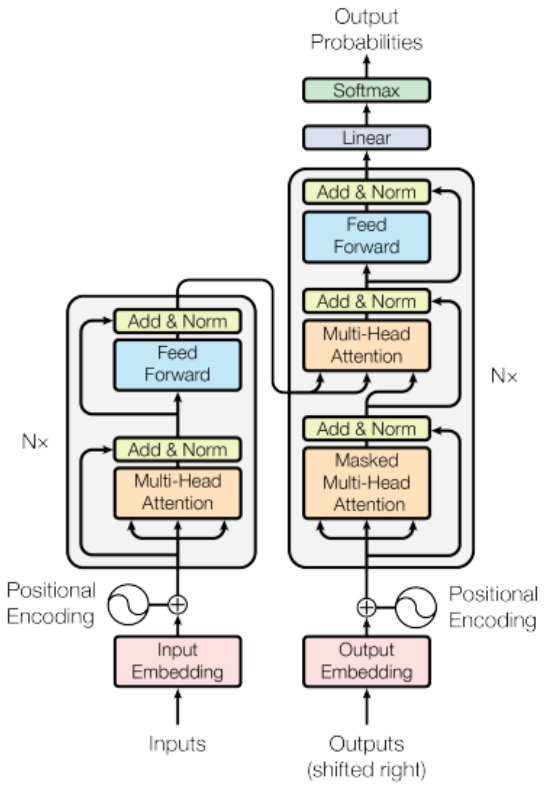


Figura 8 – Modelo de arquitetura *transformer*

Fonte: Vaswani et al. (2017)

2020).

O modelo RAG proposto por Lewis et al. (2020) possui dois componentes principais:

Um recuperador (*retriever*), utiliza um modelo de recuperação de densidade que, a partir de uma pergunta ou prompt, busca em um índice de vetores e retorna um conjunto de passagens de texto consideradas relevantes. Um gerador (*generator*), usa um grande modelo de linguagem pré-treinado que recebe a pergunta original e o conteúdo das passagens recuperadas para gerar a resposta final.

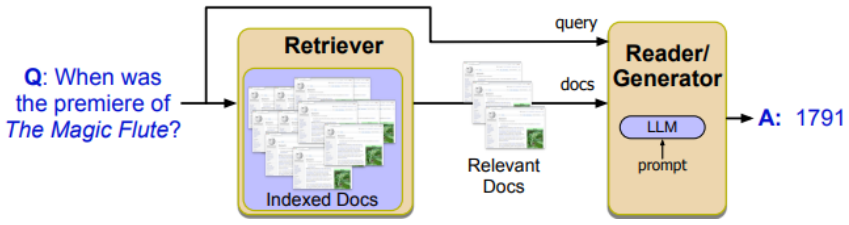


Figura 9 – Visualização dos componentes do RAG

Fonte: Jurafsky e Martin (2025)

A ideia é que ao recuperar informações, o modelo de linguagem pode se concentrar na tarefa de sintetizar uma resposta com base nessas informações, em vez de ter

que memorizar e recordar fatos de seus próprios parâmetros, podendo levar a erros ou informações desatualizadas (JURAFSKY; MARTIN, 2025).

2.3.6 *Model context protocol* (MCP)

O *model context protocol* é um protocolo de comunicação projetado para facilitar a interoperabilidade e o compartilhamento de contexto entre agentes de IA. Ele surge para solucionar a ausência de uma abordagem padronizada em sistemas multiagentes (do inglês, Multi-agent systems), para que agentes possam trocar e utilizar informações contextuais de forma eficiente e segura (HOU et al., 2025).

O MCP é um protocolo aberto, desenvolvido inicialmente pela Anthropic entre 2023 e 2024, com o objetivo de padronizar como modelos de IA acessam dados, utilizam ferramentas e mantêm o contexto. Sua criação foi motivada pelos desafios enfrentados na integração de modelos de linguagem em sistemas complexos, onde havia dificuldade em manter o contexto de forma consistente e eficiente. O MCP permite que diferentes agentes e modelos compartilhem informações contextuais de maneira estruturada, promovendo interoperabilidade e facilitando a integração com fontes de dados, APIs e ferramentas externas , como observado na figura 10 (KRISHNAN, 2025).

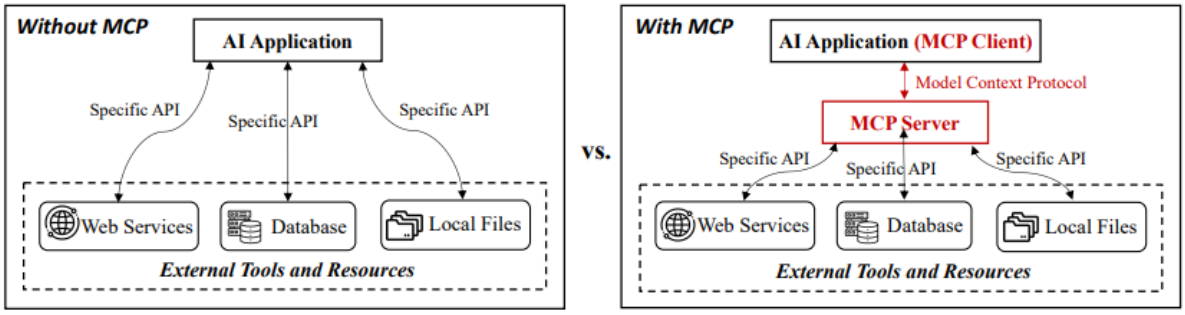


Figura 10 – Uso de ferramentas com e sem MCP

Fonte: Hou et al. (2025)

A arquitetura do MCP é composta por três componentes: o MCP *Host*, o MCP *Client* e o MCP *Server*, os componentes são definidos da seguinte forma:

MCP *Host*: É a aplicação de IA que fornece o ambiente de execução para as tarefas, servindo como o iniciador de todas as interações no MCP. Ele é responsável por gerenciar as conexões com os MCP *Servers* e por orquestrar os fluxos de trabalho de acordo com o protocolo (EHTESHAM et al., 2025).

MCP *Client*: Atua como um intermediário dentro do *host*, gerenciando a comunicação entre a aplicação e um ou mais MCP *Servers*. Ele é responsável por iniciar requisições, consultar as funções disponíveis e receber as respostas que descrevem as capacidades do servidor (HOU et al., 2025).

MCP *Server*: Habilita o acesso a sistemas externos e oferece três capacidades fundamentais:

- *Tools* (Ferramentas): Permitem ao servidor invocar serviços e APIs externas para executar ações no lugar de modelos de IA, como obter dados em tempo real ou realizar análises específicas. Quando o cliente faz um pedido, o servidor identifica a ferramenta apropriada e retorna o resutado. Diferente de chamadas de função tradicionais, as ferramentas do MCP permitem que o modelo selecione a ação apropriada com base no contexto (HOU et al., 2025).

- Resources (Recursos): Fornecem acesso a conjuntos de dados estruturados e não estruturados que o servidor pode expor aos modelos de IA. Quando modelos de IA requisitam dados específicos, o servidor recupera e processa a informação relevante, permitindo que o modelo faça decisões baseadas em dados (HOU et al., 2025).

- Prompts (Modelos de Prompt): São *templates* e fluxos de trabalho predefinidos que o servidor mantém para otimizar e padronizar as respostas da IA, garantindo consistência e eficiência em tarefas repetitivas (HOU et al., 2025).

O MCP se diferencia por superar as abordagens anteriores de gerenciamento de contexto. Ele incorpora práticas como:

***Prompt engineering*:** Abordagem que insere informações contextuais diretamente no prompt do modelo. O MCP retira a flexibilidade dos prompts, mas oferece uma gestão controlada de quais contextos devem ser incluídos e quando (KRISHNAN, 2025).

RAG: a maioria das implementações de RAG é focada apenas em recuperação de documentos e não possui protocolos padronizados para integrar diferentes tipos de dados. O MCP generaliza o conceito de RAG, permitindo a incorporação de múltiplos tipos de contexto, com prompts padronizados que simplificam a integração (KRISHNAN, 2025).

***Frameworks* de agentes com ferramentas:** Esses *frameworks* conectam modelos a ferramentas externas, mas geralmente implementam integrações específicas de cada framework. O MCP padroniza essas integrações, oferecendo protocolos que funcionam em diferentes ambientes e *frameworks* (KRISHNAN, 2025).

2.4 Sistema Eletrônico de Informações

O Sistema Eletrônico de Informações (SEI) é uma plataforma de gestão de processos e documentos eletrônicos desenvolvida pelo Tribunal Regional Federal da 4ª Região (TRF4). Sua finalidade é promover a eficiência administrativa e a sustentabilidade através da eliminação do uso de papel como suporte físico para documentos institucionais e do

compartilhamento do conhecimento. Disponibilizado como software público, o SEI é adotado por um grande número de órgãos da administração pública brasileira, tornando-se importante na modernização da gestão processual (TRF4, 2018).

A arquitetura do SEI prevê duas formas de integração e extensão de suas funcionalidades: o desenvolvimento de módulos internos e o uso de serviços externos. Os módulos são funcionalidades novas acopladas ao sistema principal do SEI, compartilhando a mesma base de dados e com acesso direto às suas funcionalidades internas. Já para a comunicação com sistemas externos, o SEI disponibiliza um conjunto de *web services* que funcionam como uma interface de programação de aplicações (API) (TRF4, 2018).

2.4.1 Integração via API

Para permitir que sistemas externos interajam com seus dados e funcionalidades, o SEI expõe um conjunto de *web services* que seguem o protocolo SOAP (*Simple Object Access Protocol*). A estrutura e as operações disponíveis nesta API são descritas através de arquivos WSDL (*Web Services Description Language*), que funcionam como um contrato detalhando como a comunicação deve ser realizada (TRF4, 2018).

O acesso à API é realizado por meio de uma autenticação em nível de sistema. Um sistema externo que deseja se conectar precisa ser previamente cadastrado no SEI para obter credenciais, que devem ser enviadas no cabeçalho de cada requisição SOAP para validar o acesso. As operações disponibilizadas pela API são divididas em dois serviços, cada um com seu respectivo arquivo WSDL (TRF4, 2018):

- **SeiPort.wsdl:** Oferece um conjunto de operações para o gerenciamento de processos. Através deste serviço, um sistema externo pode, por exemplo, gerar novos procedimentos, consultar o estado e os dados de um processo existente, listar as unidades organizacionais cadastradas e adicionar documentos a um processo.
- **SeiDocumentoPort.wsdl:** Foca nas operações relacionadas a documentos dentro dos processos. Este serviço permite a um sistema externo consultar o conteúdo de um documento, assinar um documento eletronicamente, cancelar um documento e incluir um arquivo em um documento.

Essa arquitetura de integração via API permite que sistemas externos, como portais ou novas aplicações inteligentes, possam interagir com o SEI, automatizando tarefas e criando funcionalidades com as base de dados e os fluxos de trabalho já gerenciados pelo sistema.

3 Planejamento do SEI Expert

Este capítulo apresenta o planejamento realizado para o desenvolvimento do SEI Expert. São abordadas a arquitetura proposta para o sistema, a modelagem dos perfis de usuário, o planejamento em quatro níveis de desenvolvimento, a especificação dos requisitos funcionais e não funcionais, a prototipação e o cronograma estipulado.

3.1 Introdução

O gerenciamento de processos de negócio oferece uma estrutura para a otimização de fluxos de trabalho, mas a etapa de análise de processos enfrenta dificuldades quando realizada manualmente. Sabendo que grande parte desses fluxos opera dentro de plataformas como o SEI, foi estudada a possibilidade de integração por meio de sua API. A dependência do conhecimento do analista, a subutilização do histórico de casos e a dificuldade em garantir a aplicação uniforme de legislações resultam em erros, ineficiências e falta de padronização, configurando o problema que este trabalho se propôs a resolver.

Para ilustrar a interação planejada do analista com a inteligência virtual, tem-se a figura 11, que mostra um usuário interagindo com uma inteligência artificial e com um sistema de informação, o SEI. O processo começa com o usuário entendendo as tarefas a serem feitas e coletando documentos e informações para apresentar a IA. A partir disso, a inteligência artificial atua como um assistente, consultando e identificando padrões nas informações apresentadas. Essa assistência mantém o papel do usuário como central, mas amplia suas capacidades com o apoio da IA.

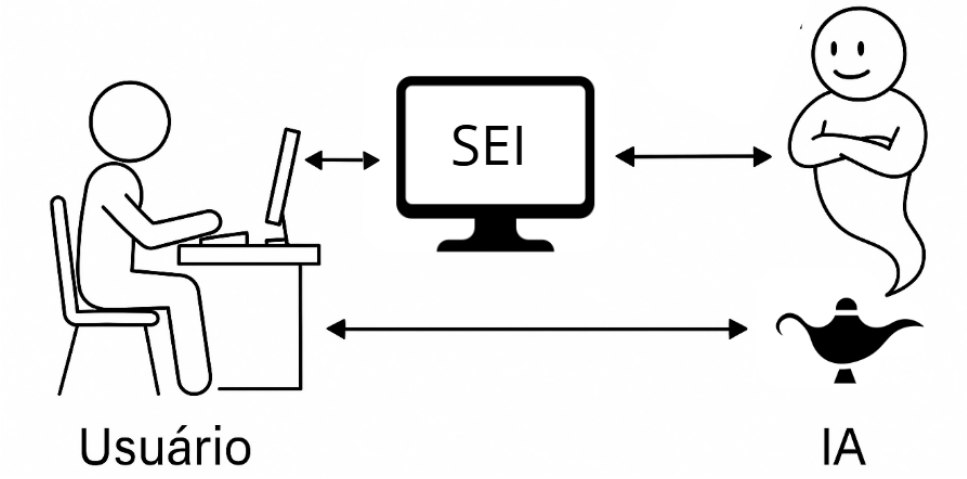


Figura 11 – Interação do usuário com o SEI e IA

Fonte: Autor

A solução proposta visava continuar com esse objetivo, porém formalizando essa interação com a IA de maneira a ampliar sua capacidade de assistência. A solução foi concebida como um sistema de software que atuaria como apoio para o analista, com base em LLMs, protocolos de contexto e ML. A interação do usuário com o sistema foi projetada para ser integrada ao seu fluxo de trabalho. O analista, após autenticar-se na plataforma, realizaria o *upload* de um processo. A partir daí, a principal ferramenta de interação seria um *chatbot*, onde o usuário poderia, em linguagem natural, fazer requisições.

A interação foi desenhada para começar quando o analista enviasse uma requisição não diretamente a uma LLM. A camada de IA validaria essa requisição e, em seguida, construiria um contexto, agregando informações do perfil do usuário, do conteúdo do processo e de documentos relevantes recuperados da base de leis e da base de casos similares. Com esse contexto completo, formular-se-ia um *prompt* para envio à LLM presente na camada. A resposta gerada seria então validada novamente antes de ser apresentada ao usuário.

3.2 Arquitetura do sistema

A arquitetura do sistema foi concebida de forma a facilitar o desenvolvimento e a escalabilidade, conforme visto na figura 12, consistindo em 4 componentes e uma camada de IA. Os 4 componentes definidos foram o *frontend*, o *backend*, um possível componente de API de dados e o banco de dados. Além desses componentes, também foi planejada a camada de IA consistindo em 5 módulos: o validador semântico, uma LLM específica, uma LLM principal, o recomendador inteligente e outro módulo destinado a componentes futuros, junto com o orquestrador MCP que seria o componente principal dessa camada.

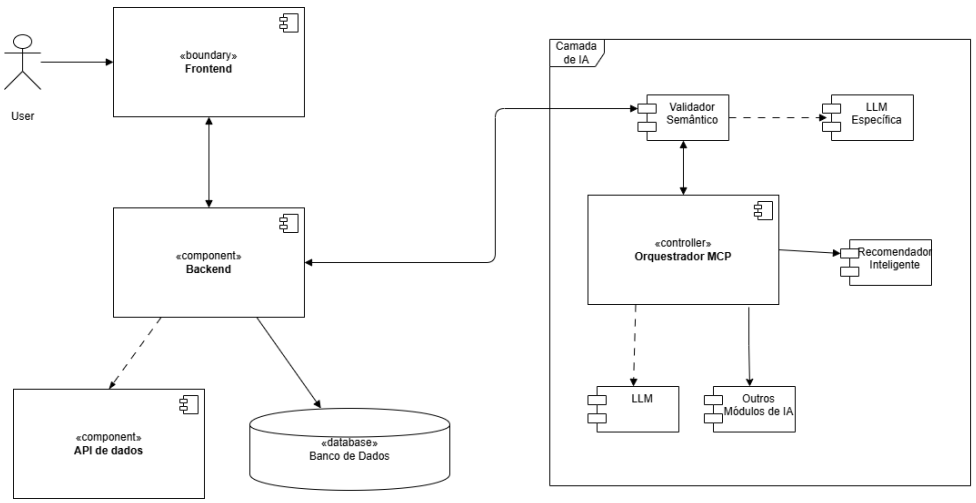


Figura 12 – Diagrama da arquitetura geral do sistema.

Fonte: Autor

- **Frontend:** Camada responsável pela interação do usuário. Incluindo o gerenciamento de sessões de usuário através de autenticação e de um controle de acesso baseado em papéis (do inglês, *Role-Based Access Control*) que adaptaria as funcionalidades disponíveis de acordo com o perfil do usuário logado (analista, especialista, gestor, administrador). A interface permitiria a visualização e o *upload* de processos e documentos (inicialmente de maneira manual), a busca e o gerenciamento da base de legislações, e o compartilhamento de *templates* de processos. Na interação com o *chatbot* a interface enviaria a requisição do usuário e o contexto do processo para o *backend* e aguardaria o retorno da resposta já validada pela camada de IA.

- **Backend:** Componente da lógica da aplicação, planejado para tratar todas as requisições, validando a autenticação e autorização do usuário para cada operação. Suas responsabilidades incluíam o gerenciamento de processos, envolvendo o controle de estado e as regras de negócio associadas aos documentos. Coordenaria as comunicações, recebendo uma requisição da interface e decidindo para onde encaminhá-la, se para a base de dados, ou se para a camada de inteligência artificial. Além disso, seria encarregado do registro de *logs*, como acessos, modificações de dados e as interações com a IA, garantindo a rastreabilidade e a segurança do sistema.

- **Base de dados:** Este componente ficaria responsável pelo armazenamento de todos os dados da aplicação. Seu escopo incluía: a gestão dos perfis de usuários, armazenando suas informações e, fundamentalmente, as permissões associadas a cada papel para a implementação do RBAC, o armazenamento de documentos e processos analisados, incluindo tanto os arquivos brutos quanto seus metadados (tipo, data, status) e as relações entre eles (como a associação de uma lei a um tipo de processo), e o armazenamento dos *logs* de interações com a IA. Os *logs* incluiriam o histórico completo de cada requisição com a camada de IA, contendo o *prompt* final enviado à LLM, a resposta bruta, a resposta pós-processada, e outros dados relevantes, servindo como base para auditoria e rastreabilidade.

- **API de dados:** Este componente representaria a interface do sistema para comunicação com fontes de dados externas. A arquitetura foi projetada de forma extensível para poder consumir APIs de sistemas externos, sendo o Sistema Eletrônico de Informações (SEI) o principal alvo de integração.

- **Camada de inteligência artificial:** Esta camada foi projetada como um ecossistema onde haveria a interação entre o sistema e os módulos de IA, transformando requisições genéricas em respostas contextualizadas. Ela seria composta por:

  - **Validador semântico:** Atuaria como a porta da camada de IA, tendo uma função bidirecional. Na entrada, pré-processaria a requisição do usuário, refinando perguntas ambíguas antes de passá-las ao orquestrador MCP, e iden-

tificaria a requisição. Na saída, realizaria uma validação na resposta gerada pela IA, verificando a coerência, a conformidade com as regras de negócio e detectando inconsistências antes de retorná-la ao *backend*. Para isso, poderia interagir com uma LLM específica e mais leve.

- **Orquestrador MCP:** Seria o orquestrador da camada, com a responsabilidade de gerenciar todo o ciclo de vida da interação: receber a identificação da tarefa do validador, construir o contexto dinamicamente (usando perfil de usuário, documentos, leis), aplicar *prompts*, acionar o módulo de IA correto, e realizar o processamento inicial da resposta.
- **Recomendador inteligente:** Módulo que utilizaria técnicas de Aprendizado de Máquina, focado em análises que vão além da geração de texto. Utilizaria técnicas como *clustering*, analisando dados de comportamento e decisão dos analistas para agrupar perfis por similaridade e sugerir o especialista mais adequado para analisar novos processos, otimizando a alocação de tarefas.
- **LLM(s):** A arquitetura previa o uso de múltiplos modelos de linguagem para diferentes tarefas. Haveria ao menos uma LLM mais poderosa para as tarefas complexas de análise e geração de texto orquestradas pelo orquestrador MCP, e outra, mais leve, dedicada às tarefas de validação e refinamento dentro do validador semântico.
- **Outros módulos de inteligência artificial:** A arquitetura previa espaço para futuras expansões e adição de novos módulos de inteligência artificial.

Em suma, o fluxo básico envolveria o usuário interagindo com a interface, cujas requisições seriam processadas pelo *backend*. Este, por sua vez, poderia interagir com a camada de inteligência artificial, acionando o validador que daria início as atividades da camada de IA. A resposta, após ser validada, seria encaminhada ao *backend* e, em seguida, à interface do usuário.

3.3 Funcionalidades detalhadas por níveis de desenvolvimento

O desenvolvimento do sistema proposto foi estruturado através de quatro níveis. Esta abordagem, alinhada com os princípios do desenvolvimento incremental, estabeleceu uma ordem de dependência das funcionalidades, onde cada nível serviria como alicerce para o seguinte, garantindo um progresso com entregas de valor. A figura 13 ilustra esta estratégia através de uma pirâmide, ressaltando que os níveis seriam dependentes dos seus sucessores para a implementação mas funcionais desde a base. A base representa as funcionalidades essenciais e ao caminho do topo encontram-se as capacidades mais avançadas do sistema. O nível 1 (Base) estabeleceu os fundamentos do sistema, com a

arquitetura e a interação básica via *chatbot*. O nível 2 (Contextualização e rastreabilidade) introduziria as recomendações de casos similares e de legislações pertinentes. Em seguida, o nível 3 (Padronização) focaria na padronização, trazendo a possibilidade de criar, editar e compartilhar *templates* de processos. Por fim, o nível 4 (Recomendação e gestão), no topo da pirâmide, utilizaria aprendizado de máquina para fornecer recomendações de especialistas e visualização de desempenho. Os níveis são detalhados a seguir:

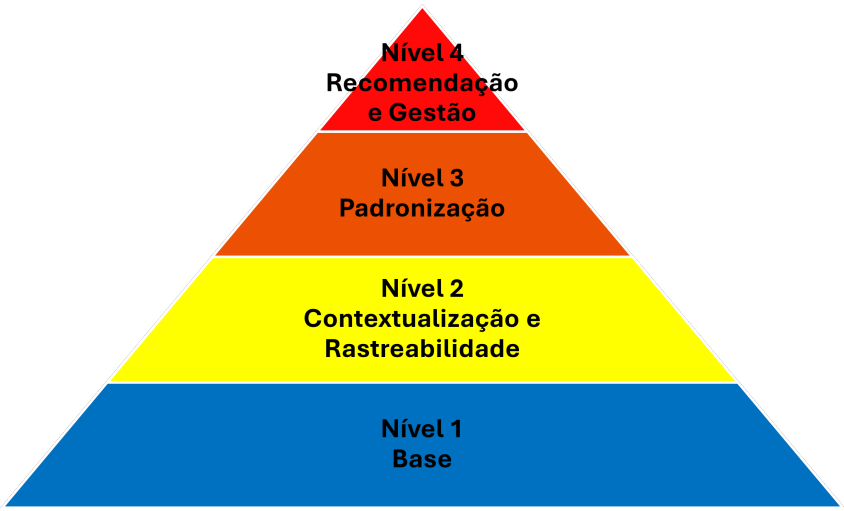


Figura 13 – Representação dos níveis do sistema.

Fonte: Autor

3.3.1 Nível 1 - Base

O nível 1 (Base) representou a base do projeto, responsável por estabelecer a arquitetura do sistema e implementar as funcionalidades para a interação do analista com a inteligência artificial. O objetivo desta etapa era desenvolver um sistema mínimo, que realizasse os principais fluxos de interação e servisse como alicerce para os níveis sequentes que agregariam mais complexidade. As funcionalidades deste nível foram definidas em:

Gerenciamento de Usuários e Perfis: incluía funcionalidades para o cadastro de novos usuários e sua vinculação a perfis predefinidos. A gestão do sistema seria uma atribuição do perfil de administrador, definido por permissões iniciais através de um modelo de controle de acesso baseado em papéis (do inglês, *Role-Based Access Control* - RBAC). Foram planejados perfis de analistas, utilizando uma base de dados com atributos estáticos (cargo, setor), dinâmicos (tipos de processos analisados) e um "atributo LLM"(contendo *templates* e legislações associadas) para contextualizar as interações do analista com a IA.

Base de dados: Neste nível, foi planejada uma base de dados permitindo o *upload* e gerenciamento de dois tipos de conteúdo:

- **Legislações:** O sistema teria uma funcionalidade para armazenar, gerenciar e buscar documentos legais. Ao interagir, especialistas poderiam associar essas leis a processos, criando uma relação que seria utilizada no nível 2.

- **Processos:** Usuários poderiam realizar o *upload* de processos para análise, inicialmente por meio de arquivos contendo os documentos pertinentes.

Interação com IA via *chatbot*: A interface permitiria que o usuário submetesse um processo carregado no sistema para análise e fizesse requisições em linguagem natural. Todas as interações seriam mediadas pelo orquestrador MCP, que contextualizaria a requisição do usuário com informações do sistema, como o perfil do analista e as legislações vinculadas ao processo, garantindo que as respostas, como resumos de documentos ou esclarecimentos, fossem relevantes.

Sistema base: Neste nível o sistema permitiria ao usuário realizar *login*, acessar seu perfil, visualizar os processos carregados, interagir com o *chatbot* e gerenciar a base de legislações e processos. Este protótipo serviria para validar o fluxo de interação e a comunicação entre os diferentes componentes da arquitetura.

3.3.2 Nível 2 - Contextualização e rastreabilidade

O nível 2 introduziria uma atuação sobre dados de processos anteriores e de legislações, que seriam utilizados como referência na análise dos processos. Neste nível, utilizar-se-ia técnicas de RAG e de análise de dados para identificar e referenciar processos anteriores semelhantes e legislações pertinentes. Um objetivo deste nível era o desenvolvimento das justificativas e da rastreabilidade das sugestões, que deveriam ser acompanhadas de explicações e referências a documentos anteriores ou normas, auxiliando o analista a confiar e revisar as informações com maior rapidez. As funcionalidades do nível 2 foram definidas como:

Recomendações de casos similares e legislações: Baseava-se na implementação de uma base de casos similares e de legislações pertinentes. Esta base de dados seria populada com informações de processos já analisados pelo sistema, incluindo os documentos, as decisões tomadas, os pareceres gerados e a legislação aplicada. O sistema recomendaria decisões, encaminhamentos ou trechos de pareceres baseados nesses casos anteriores e legislações pertinentes.

Justificativas explícitas e rastreabilidade de fontes: Para aumentar a confiança do analista com a interação, toda recomendação ou análise significativa seria acompanhada por justificativas explícitas. O sistema apresentaria as leis e os casos similares que foram utilizados como base para a sua recomendação. Isso permitiria ao analista compreender as fontes e revisar informações com maior rapidez.

Fluxo de validação e aprimoramento contínuo: O nível 2 introduziria um fluxo de validação, onde usuários com perfis de especialista e gestor teriam acesso à funcionalidade de validar as decisões anteriores armazenadas no sistema para fins de auditoria e gestão da qualidade.

3.3.3 Nível 3 - Padronização

Neste nível, o foco seria a padronização através de *templates*. O sistema permitiria que especialistas criassem, editassem e compartilhassem *templates* de processos, baseados em análises anteriores bem-sucedidas. Seriam implementadas funcionalidades para busca e reutilização desses *templates* por outros analistas, além de mecanismos de edição colaborativa com histórico de versões e controle de acesso para garantir a segurança no compartilhamento. As funcionalidades deste nível foram definidas como:

Gerenciamento de *templates*: O sistema proveria uma interface dedicada para que usuários com o perfil de especialista pudessem criar e editar *templates*. Estes *templates* poderiam ser criados do zero ou baseados em análises anteriores bem-sucedidas. Incluiria também o histórico de versão, que permitiria aos usuários com permissão (como especialistas e gestores) visualizarem e compararem diferentes versões de um mesmo *template*, garantindo a rastreabilidade e qualidade das alterações.

Compartilhamento e controle de acesso: Para segurança dos dados compartilhados, o sistema implementaria um controle de acesso. Especialistas poderiam compartilhar *templates* com regras de acesso específicas, definindo quais usuários ou grupos poderiam visualizar ou utilizar cada *template*. Este mecanismo seria gerenciado pelo modelo de controle de acesso baseado em papéis (RBAC), assegurando que apenas os perfis autorizados pudessem criar, modificar e aprovar os *templates*.

Busca e reutilização de *templates*: Introduziria a capacidade de buscar e reutilizar *templates* existentes. A busca poderia ser realizada por palavras-chave, tipo de processo, ou outros dados. O sistema permitiria uma visualização clara do conteúdo e função de cada *template*, ajudando o usuário a escolher o mais adequado para o seu caso.

3.3.4 Nível 4 - Recomendação e gestão

O Nível 4 introduziria funcionalidades de aprendizado de máquina para identificação de perfis de especialistas com base em seus comportamentos, decisões e dados do sistema, visando recomendar o analista mais adequado para novos processos. O sistema também previa visualizações gráficas dos grupos de especialistas e seu desempenho, e permitiria uma evolução dinâmica dos perfis ao longo do tempo. As funcionalidades deste nível foram definidas como:

Identificação de perfis: A funcionalidade consistiria na identificação de perfis de

especialistas através de técnicas de ML, executadas pelo recomendador inteligente. Utilizando algoritmos de *clustering*, o sistema analisaria os dados de uso armazenados na base de dados (como tipos de processos analisados, tempo médio, decisões tomadas e *templates* utilizados) para agrupar analistas com base nos seus comportamentos e especializações.

Recomendação especialistas: Com base nos perfis identificados, o sistema ofereceria uma funcionalidade de recomendação direcionada aos gestores. Ao receber um novo processo, o sistema sugeriria o especialista mais adequado para a tarefa, considerando a área temática e os perfis identificados pelos modelos de ML.

Visualização de desempenho e perfis: Gestores poderiam visualizar os agrupamentos de especialistas, comparar o desempenho entre eles e acompanhar métricas individuais e de equipe, como tempo médio de análise por tipo de processo e volume de trabalho, através de *dashboards* com visualização gráfica.

Evolução dos perfis: A arquitetura previa a evolução dos perfis de usuário. Isso garantiria que os agrupamentos e as recomendações permanecessem relevantes e se adaptassem às mudanças e ao foco de trabalho dos analistas ao longo do tempo, ocorrendo através da alimentação contínua dos modelos de ML.

3.4 Perfis de usuários do sistema

O sistema foi planejado para implementar um modelo RBAC. O RBAC é um método de gerenciamento de permissões onde o acesso aos recursos do sistema não é atribuído diretamente a usuários individuais, mas sim a perfis predefinidos. Cada usuário seria então associado a um ou mais perfis, herdando automaticamente as permissões daquele papel. Esta abordagem simplificaria a administração, aumentaria a segurança e garantiria que os usuários tivessem acesso apenas às funcionalidades necessárias para executar suas funções. O sistema contaria com os seguintes perfis de usuário principais:

- **Analista:** Usuário que realiza a análise direta de processos. Suas funcionalidades incluiriam visualizar processos, utilizar o *chatbot* para obter resumos e orientações, consultar documentos legais e utilizar *templates* criados previamente.

- **Especialista:** Profissional com maior autonomia e conhecimento. Possuiria todas as permissões do analista, além da capacidade de validar e refinar respostas da inteligência artificial, criar *templates* de processos e adicionar e gerenciar legislações.

- **Gestor:** Responsável pela coordenação de equipes e processos. Supervisionaria os perfis e o desempenho dos analistas, acompanharia decisões e justificativas, visualizaria agrupamentos de especialistas e aprovaria ou rejeitaria *templates*.

- **Administrador:** Perfil responsável pela infraestrutura, configuração do sistema, gerenciamento de usuários e perfis, configuração da camada de IA, controle de acesso à APIs de dados externos e garantia da rastreabilidade e segurança do sistema.

3.5 Camada de inteligência artificial

Esta camada tinha como foco a execução das tarefas de inteligência artificial, sendo responsável por transformar as requisições dos usuários em respostas contextualmente corretas. A arquitetura desta camada, visualizada na figura 14, englobava componentes como o validador semântico, o orquestrador MCP, o recomendador inteligente e as LLMs, sendo concebida para garantir não apenas a eficácia do processamento, mas também a escalabilidade futura do sistema.

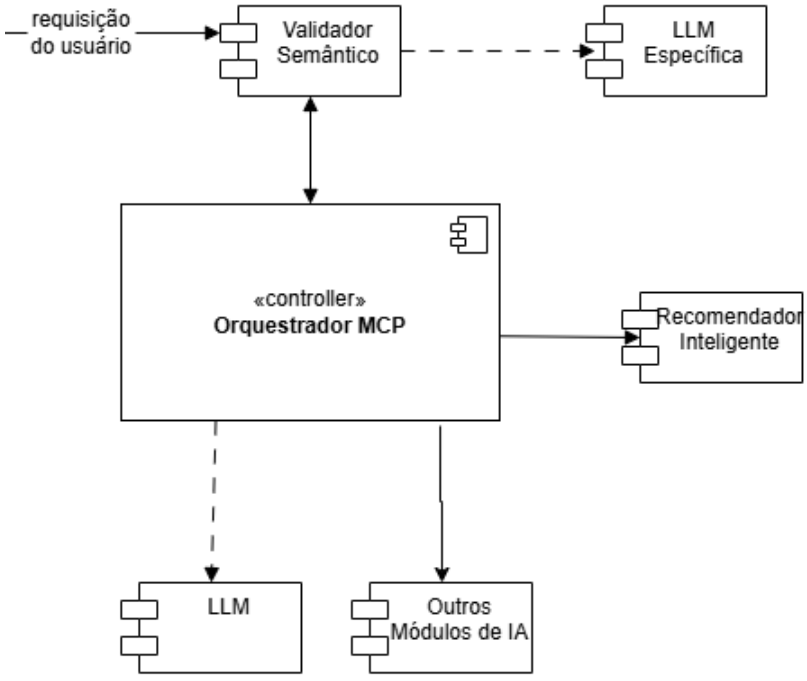


Figura 14 – Diagrama da camada de IA

Fonte: Autor

A seguir são apresentadas mais informações sobre cada um dos componentes citados.

3.5.1 Validador semântico

O Validador Semântico atuaria como um gestor de qualidade para as interações seguintes. Sua função seria bidirecional, processando tanto as requisições que entram na

camada quanto as respostas que saem dela, garantindo maior precisão e conformidade em todo o fluxo. E também identificaria com o auxílio dessa LLM, a requisição do usuário.

Na etapa de entrada, ao receber uma requisição do *backend*, o validador semântico analisaria a consulta do usuário para corrigir ambiguidades e aprimorar sua clareza, utilizando uma LLM específica e mais leve para essa tarefa. Isso asseguraria que a requisição encaminhada ao orquestrador fosse bem desenvolvida, otimizando o processamento subsequente.

Na etapa de saída, após o MCP e a LLM principal gerarem uma resposta, o validador semântico realizaria uma verificação final. Sua responsabilidade nesta fase consistiria em assegurar a qualidade textual antes que a resposta fosse enviada de volta ao *backend* e, consequentemente, ao usuário. Este componente seria fundamental para aumentar a confiabilidade do sistema e diminuir os riscos associados à geração de informações incorretas.

3.5.2 Orquestrador MCP

Sua função seria padronizar, contextualizar e controlar toda interação entre o sistema e seus módulos. Esta abordagem visava superar as limitações das interações diretas com LLMs, que podem resultar em respostas genéricas e inconsistentes. O orquestrador MCP atuaria, portanto, como um tradutor que recebe requisições e as converte em instruções precisas para a inteligência artificial. O fluxo de trabalho do orquestrador foi desenhado para ser bidirecional, abrangendo tanto o processamento da requisição (entrada) quanto o processamento da resposta (saída):

3.5.2.1 Processamento da requisição

Preparação da Tarefa: Ao receber uma solicitação do validador semântico, o orquestrador primeiro receberia a tarefa identificada (e.g., resumo de documento, análise de processo, recomendação de especialista) e a prepararia, isso permitiria que ele selecionasse os recursos adequados para cada tarefa.

Construção de Contexto: O orquestrador coletaria todas as informações contextuais necessárias. Isso incluiria dados do perfil do usuário (cargo, setor), o conteúdo do processo ou documento em análise, e normas ou legislações relevantes recuperadas da base de dados do sistema.

Definição do Prompt: Com o contexto coletado, utilizar-se-ia estruturas de *prompt* pré-definidas e otimizadas para cada tipo de tarefa, para montar a instrução final que seria enviada à LLM. Além do conteúdo, ele também definiria os parâmetros de comportamento da LLM, como o tamanho da resposta, o formato e o estilo de linguagem.

Acionamento dos Módulos de Execução: Com o *prompt* finalizado, o MCP acionaria o(s) módulo(s) apropriado(s), que poderia ser a LLM ou o recomendador inteligente, dependendo da tarefa identificada.

3.5.2.2 Processamento da resposta

A resposta bruta gerada pela LLM seria recebida pelo orquestrador MCP, que realizaria um processamento inicial. Isso incluiria formatar a resposta, adicionar referências e registrar *logs* da interação. Em seguida, a resposta processada seria enviada ao validador semântico para a verificação final antes de ser retornada ao *backend*.

3.5.3 Recomendador inteligente

O recomendador inteligente foi definido como o módulo focado na aplicação de técnicas de aprendizado de máquina para extrair padrões e fornecer sugestões que vão além da análise textual das LLMs. Este componente seria acionado pelo orquestrador MCP quando a tarefa identificada envolvesse a necessidade de análise e/ou recomendação de perfis. As funcionalidades deste módulo incluiriam:

Agrupamento de Perfis e Processos: Utilizando algoritmos de *clustering*, o recomendador analisaria dados de múltiplos analistas (como os tipos de processos analisados e o tempo médio de análise) e de processos, para identificar e agrupar perfis e processos com características e especializações semelhantes.

Sugestão de Analistas e Processos: Com base nos perfis agrupados e nas características de um novo processo, o gestor poderia pedir uma sugestão de distribuição de tarefas, otimizando o gerenciamento dos analistas. Além disso, também faria a sugestão de processos semelhantes e de legislações pertinentes.

Para executar suas funções, este módulo consumiria dados do sistema, especificamente os **logs** de interação e os perfis de usuário, para alimentar seus modelos de aprendizado. O resultado das análises seria retornado ao MCP para que a informação fosse integrada ao modelo de resposta.

3.5.4 Modelos de linguagem de grande escala

Os Modelos de Linguagem de Grande Escala seriam responsáveis pela compreensão de linguagem natural. A arquitetura proposta adotava uma abordagem que previa o uso de múltiplos modelos para diferentes finalidades. As LLMs seriam utilizadas via API e sua interação seria mediada pelo orquestrador MCP. A atuação das LLMs no sistema começaria na validação semântica, onde uma LLM específica seria utilizada pelo validador semântico para tarefas de verificação de coerência e aprimoramento de requisições, atuando no controle de qualidade. Além disso, as LLMs gerariam análises e respostas, e

o orquestrador enviaria *prompts* contextualizados para as LLMs a fim de gerar resumos e análises de documentos, respostas para as perguntas dos analistas e outras necessidades.

3.6 Prototipação

Os protótipos a seguir foram desenvolvidos com base nas funcionalidades abordadas no nível 1, representando uma visualização de média fidelidade das principais telas e fluxos de interação do sistema. O objetivo destes protótipos era demonstrar como as funcionalidades do sistema interagiriam com o usuário. As telas ilustram a camada de *frontend*.

3.6.1 Gerenciamento de legislações

A figura 15 apresenta a interface de gestão de legislações. O usuário poderia realizar buscas, visualizar documentos já cadastrados ou inserir novas legislações por meio do botão *upload*. A navegação seria feita por pastas, o que melhoraria a organização das informações.

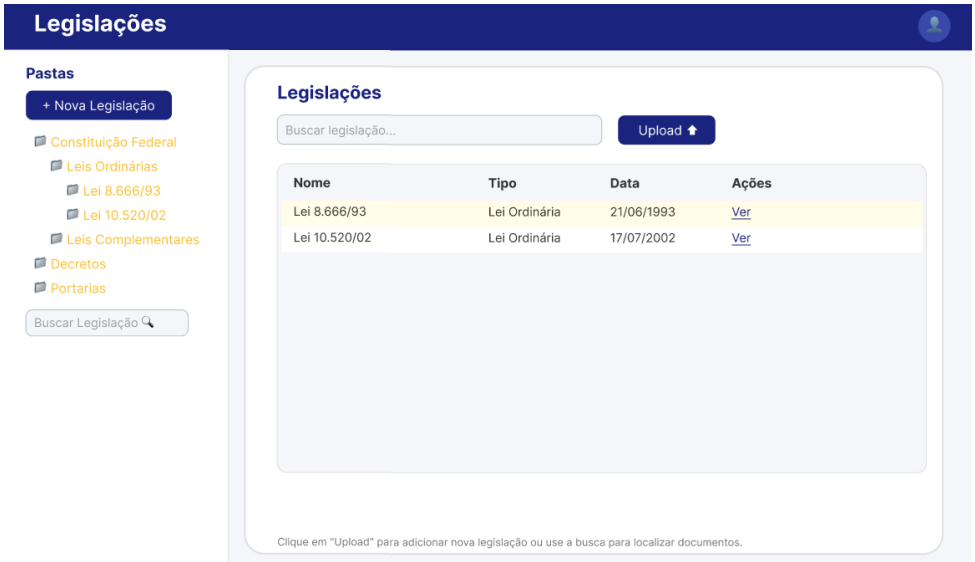


Figura 15 – Protótipo da tela de gerenciamento de legislações

Fonte: Autor

3.6.2 Gerenciamento de processos

A figura 16 mostra a interface dedicada ao gerenciamento de processos. Nela, os processos seriam exibidos com nome, tipo e data de cadastro. O sistema também ofereceria acesso rápido a *templates* compartilhados (nível 3).

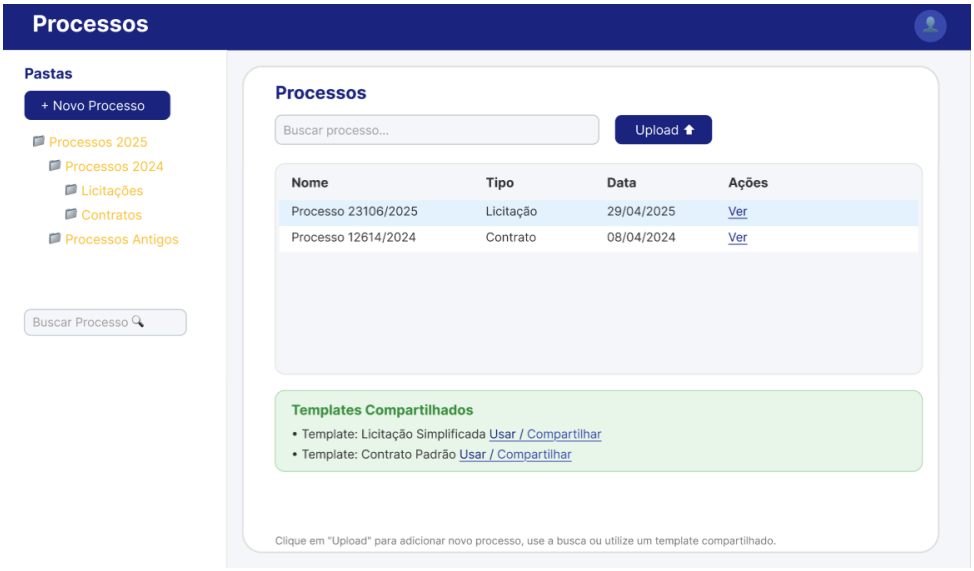


Figura 16 – Protótipo da tela de gerenciamento de processos

Fonte: Autor

3.6.3 Interface do *chatbot*

A Figura 17 representa a interface do *chatbot*. À esquerda e à direita da tela, seriam listadas legislações e processos anexadas pelo usuário. Ao centro, encontrava-se a área de interação textual.



Figura 17 – Protótipo da tela de *chatbot*

Fonte: Autor

3.6.4 Cadastro no sistema

A figura 18 exibe a tela de cadastro de novos usuários, onde seriam solicitados dados essenciais. Além disso, o usuário deveria informar qual perfil representava seu papel

no sistema: analista, especialista ou gestor. Essa distinção de perfis permitiria aplicar regras de acesso específicas conforme o tipo de atuação do usuário.



Figura 18 – Protótipo da tela de cadastro

Fonte: Autor

3.7 Requisitos do sistema

Os requisitos do sistema foram levantados através de uma abordagem interativa, utilizando como técnicas o *brainstorming* e a introspecção nas reuniões semanais de orientação. Nessas reuniões, foram realizadas introspecções para elicitação de requisitos, onde o orientador atuou como cliente e especialista. Esta elicitação de requisitos teve como propósito compreender as necessidades dos clientes e os objetivos do sistema, para então definir e documentar suas funcionalidades e restrições. Para organizar a complexidade, os requisitos foram estruturados de acordo com os quatro níveis de desenvolvimento do sistema e priorizados de acordo com o método MoSCoW, que propôs definir o que seria essencial para uma primeira versão funcional. Assim os requisitos foram priorizados em: *Must Have (M), Should Have (S), Could Have (C), Won't Have (W)*, de acordo com a metodologia.

3.7.1 Nível 1 - Base

A seguir são apresentadas as tabelas 3 e 4, que contém os requisitos elicitados para o nível 1 (Base). A tabela 3 contém os requisitos funcionais do sistema e a tabela 4 os não funcionais, os requisitos foram identificados por um ID, descritos e priorizados com as letras M, S, C e W que representam, respectivamente, *Must Have, Should Have, Could Have e Wont't Have*.

Os requisitos definidos para o nível 1 visavam garantir uma base funcional para o sistema. A implementação dos requisitos essenciais (*Must Have*) asseguraria que o sistema fosse capaz de gerenciar usuários, armazenar dados, e oferecer uma interação contextualizada com a IA através do *chatbot* e do *MCP*.

3.7.2 Nível 2 - Contextualização e rastreabilidade

A tabela a 5 contém os requisitos funcionais do nível 2 (Contextualização e rastreabilidade), identificados, descritos e priorizados.

A tabela a 6 contém os requisitos não funcionais do nível 2 (Contextualização e rastreabilidade), identificados, descritos e priorizados.

Os requisitos definidos para o nível 2 representam uma melhoria na contextualização e na rastreabilidade, ao garantir a comparação com processos similares e a apresentação de justificativas rastreáveis. Estes requisitos visam aumentar diretamente a confiança do analista no sistema.

ID	**Descrição**	**Prioridade**
RF01	O sistema deve permitir o cadastro do usuário	M
RF02	O sistema deve permitir o login do usuário	M
RF03	O sistema deve fornecer a possibilidade de recuperar a senha do usuário	S
RF04	O sistema deve permitir que o usuário altere suas informações de cadastro	S
RF05	O *chatbot* deve ser capaz de fornecer ajuda sobre as funcionalidades do sistema	C
RF06	O sistema deve ter um *chatbot* funcional e contextualizado	M
RF07	O sistema deve permitir o *upload* e busca de documentos legais	M
RF08	O sistema deve permitir o *upload* de processos no formato .zip	M
RF09	O sistema deve fornecer uma interface para o usuário acessar os processos	M
RF10	O sistema deve fornecer uma interface para o usuário acessar as legislações	M
RF11	O sistema deve registrar dados das interações do usuário com o *chatbot*	M
RF12	O sistema deve fornecer informações das interações do usuário com o *chatbot*	M
RF13	O sistema deve permitir que um usuário administrador cadastre novos usuários na plataforma	M
RF14	O sistema deve restringir o acesso a funcionalidades com base no perfil de acesso do usuário logado (RBAC)	M
RF15	O sistema deve armazenar todas as informações necessárias para seu funcionamento básico.	M
RF16	O usuário deve poder visualizar e editar seus próprios dados de perfil após o login	C
RF17	O sistema deve possuir uma barra de navegação que permita ao usuário navegar pela interface	M
RF18	O sistema deve ser apresentado no idioma português	M
RF19	O sistema deve oferecer modo escuro e claro, com configuração manual e sincronização automática com o SO	W
RF20	A Camada de IA deve identificar a requisição do usuário para acionar o fluxo correto no MCP.	M
RF21	A Camada de IA deve coletar e agregar dados de múltiplas fontes do sistema para construir o contexto do prompt.	M
RF22	O Validador Semântico deve pré-processar e refinar requisições ambíguas do usuário antes de encaminhá-las ao MCP.	M
RF23	O Validador Semântico deve analisar a resposta da LLM para verificar a coerência e a conformidade com as regras de negócio.	M

Tabela 3 – Requisitos funcionais do sistema no nível 1

Fonte: Autor

ID	Descrição	Prioridade
RNF01	O sistema deve ter uma interface responsiva e de fácil uso.	M
RNF02	Todo e qualquer conteúdo gerado pela IA deve ser claramente identificado como tal.	C
RNF03	A terminologia utilizada na interface deve ser consistente com a do domínio de processos organizacionais.	M
RNF04	O sistema deve fornecer *feedback* visual rápido para as ações do usuário.	M
RNF05	O sistema deve apresentar respostas não demoradas do *chatbot*.	S
RNF06	As operações de CRUD na base de dados devem ser completadas em um tempo viável.	M
RNF07	As interações e dados do sistema devem ser armazenados de forma segura, com criptografia para dados sensíveis em repouso.	C
RNF08	O Controle de Acesso Baseado em Papéis (RBAC) deve ser validado no *backend* a cada requisição.	M
RNF09	O tempo de processamento interno da Camada de IA não deve impactar na análise do usuário.	S
RNF10	Dados internos da camada de IA devem ser registrados.	M

Tabela 4 – Requisitos não funcionais do sistema no nível 1

Fonte: Autor

ID	Descrição	Prioridade
RF24	O sistema deve armazenar de forma estruturada os dados relevantes e resultados de um processo analisado	M
RF25	O *chatbot* deve ser capaz de apresentar casos similares e legislações pertinentes em sua justificativa	M
RF26	O *chatbot* deve apresentar em todas as suas análises justificativas rastreáveis	S
RF27	Usuários com perfil "Especialista"devem ter uma interface para corrigir e validar uma resposta gerada pela IA	S
RF28	O sistema deve permitir a consulta e visualização de decisões de processos já validados	S
RF29	O *log* de interações da IA deve incluir as fontes de informação (IDs de documentos, leis ou casos similares) utilizadas para gerar a resposta	M
RF30	O sistema deve referenciar ativamente casos similares e legislações pertinentes às requisições do usuário	S

Tabela 5 – Requisitos funcionais do sistema no nível 2

Fonte: Autor

3.7.3 Nível 3 - Padronização

A tabela a 7 contém os requisitos funcionais do nível 3 (Padronização), identificados, descritos e priorizados.

A tabela a 8 contém os requisitos não funcionais do nível 3 (Padronização), iden-

ID	Descrição	Prioridade
RNF11	A busca e apresentação de casos similares na interface devem ser concluídas em um tempo que não interrompa o fluxo de análise do usuário	M
RNF12	A interface deve apresentar as justificativas e as referências para as sugestões da IA	M
RNF13	O sistema deve garantir que todas as fontes citadas em uma justificativa sejam válidas	M
RNF14	O acesso à base de casos similares deve ser estritamente controlado pelas regras de permissão do sistema	M
RNF15	O sistema deve ter um mecanismo para monitorar a precisão das recomendações.	S

Tabela 6 – Requisitos não funcionais do sistema no nível 2

Fonte: Autor

ID	Descrição	Prioridade
RF31	O sistema deve permitir que um perfil especialista crie e edite um *template* de processo	M
RF32	O sistema deve permitir que um perfil especialista compartilhe um *template* de processo	M
RF33	O sistema deve manter um histórico de versões para cada *template*	S
RF34	A interface do *chatbot* deve permitir que o usuário busque e aplique um *template*	M
RF35	O sistema deve permitir que a IA utilize o conteúdo de um *template* selecionado	S
RF36	O sistema deve ter uma interface onde um perfil gestor pode aprovar ou rejeitar novos *templates*	C

Tabela 7 – Requisitos funcionais do sistema no nível 3

Fonte: Autor

tificados, descritos e priorizados.

ID	Descrição	Prioridade
RNF16	A interface de criação e edição de *templates* deve ser intuitiva	M
RNF17	A busca por *templates* na base de conhecimento deve retornar resultados rapidamente	M
RNF18	As regras de compartilhamento e acesso aos *templates* (RBAC) devem ser seguidas	M

Tabela 8 – Requisitos não funcionais do sistema no nível 3

Fonte: Autor

Os requisitos do nível 3 introduzem uma estrutura para a gestão de *templates*. Essas funcionalidades facilitam a padronização e diminuiriam o trabalho manual. A priorização

dos requisitos contribuiria para o desenvolvimento do nível.

3.7.4 Nível 4 - Recomendação e gestão

A tabela a 9 contém os requisitos funcionais do nível 4 (Recomendação e gestão), identificados, descritos e priorizados.

ID	Descrição	Prioridade
RF37	O sistema deve processar periodicamente os *logs* de atividade para alimentar os modelos de ML	S
RF38	O sistema deve apresentar ao gestor uma interface para visualizar as recomendações de analistas	M
RF39	O sistema deve ser capaz de utilizar técnicas de ML para a recomendação de analistas e processos similares	M
RF40	O sistema deve fornecer um *dashboard* com métricas de desempenho dos analistas ao gestor	S
RF41	O sistema deve permitir que os perfis dos analistas sejam atualizados dinamicamente à medida que novos dados são coletados	S

Tabela 9 – Requisitos funcionais do sistema no nível 4

Fonte: Autor

A tabela a 10 contém os requisitos não funcionais do nível 4 (Recomendação e gestão), identificados, descritos e priorizados.

ID	Descrição	Prioridade
RNF19	A geração da recomendação de especialista para um novo processo deve ser concluída em um tempo aceitável	S
RNF20	O *dashboard* de visualização de perfis e desempenho (Nível 4) deve ser responsiva.	S
RNF21	As visualizações gráficas e os *dashboards* de desempenho (Nível 4) devem ser intuitivos [...]	M
RNF22	A recomendação de especialista deve ser apresentada com uma justificativa clara [...]	M
RNF23	Os critérios utilizados pelo modelo de ML para recomendação de especialistas devem ser auditáveis [...]	S

Tabela 10 – Requisitos não funcionais do sistema no nível 4

Fonte: Autor

Os requisitos do nível 4 representam uma maior utilização de IA no sistema, com a aplicação de técnicas de ML para a recomendação de especialistas e o monitoramento de desempenho, contribuindo para a automatização do contexto no sistema e para o gerenciamento de recursos. De forma geral, os requisitos mapeados ao longo dos quatro

níveis representavam uma evolução do sistema, indo da disponibilização de funcionalidades básicas até a incorporação de técnicas de inteligência artificial.

3.8 Planejamento

A presente seção detalha o planejamento realizado para a fase de implementação do sistema. Para garantir um desenvolvimento estruturado e alinhado aos princípios ágeis, o desenvolvimento do trabalho foi organizado em épicos. A seguir, são apresentados os épicos definidos para a construção do Nível 1 do sistema, que era o nível alvo de desenvolvimento, e cronogramas para entender o desenvolvimento.

3.8.1 Épicos

Para organizar o desenvolvimento das funcionalidades do nível 1 os requisitos foram agrupados em três épicos. Um Épico representa um corpo de trabalho que entrega uma funcionalidade de ponta a ponta, e que pode ser decomposto em histórias de usuário menores para ser trabalhado em *sprints*. Cada épico foi elaborado de modo a cobrir um conjunto de requisitos funcionais e não funcionais do Nível 1.

3.8.1.1 Épico 1: Gestão de Acesso e Perfis de Usuário

Descrição: Este épico engloba todas as funcionalidades necessárias para gerenciar a identidade, o acesso e as permissões dos usuários no sistema. O objetivo é criar um ambiente onde cada usuário tem uma experiência de acordo com seu papel na organização. A implementação correta deste épico foi um pré-requisito para todas as outras funcionalidades do sistema. Requisitos associados:

- RF01: O sistema deve permitir o cadastro do usuário - Funcionalidade de entrada para novos usuários, permitindo que criem um cadastro para acessar a plataforma.
- RF02: O sistema deve permitir o login do usuário - Mecanismo de autenticação para que usuários já cadastrados possam acessar o sistema.
- RF03: O sistema deve fornecer a possibilidade de recuperar a senha do usuário - Funcionalidade de usabilidade para que os usuários possam redefinir suas senhas de forma autônoma.
- RF04: O sistema deve permitir que o usuário altere suas informações de cadastro - Garante que os usuários tenham controle sobre alguns de seus próprios dados.
- RF13: O sistema deve permitir que um usuário administrador cadastre novos usuários na plataforma - Centraliza a criação de contas em um perfil com privilégios de controle do sistema.

- RF14: O sistema deve restringir o acesso a funcionalidades com base no perfil de acesso do usuário logado (RBAC) - Garante que cada tipo de usuário (Analista, Especialista, Gestor, Administrador) tenha acesso apenas às ferramentas e informações pertinentes à sua função.

- RF15: O sistema deve armazenar todas as informações necessárias para seu funcionamento básico - Refere-se à necessidade de um local para armazenar os perfis de usuário, os processos, as legislações e os logs de interação

- RF16: O usuário deve poder visualizar e editar seus próprios dados de perfil após o login - Complementa o RF04, detalhando que, após a autenticação, o usuário terá uma interface específica para gerenciar suas próprias informações pessoais.

3.8.1.2 Épico 2: Gerenciamento de dados

Descrição: Este épico foca na criação da infraestrutura para que o sistema possa ser alimentado com o conhecimento necessário para as análises da IA. Ele abrange as funcionalidades que permitem aos usuários carregar e organizar dois tipos de conteúdo: os processos que são analisados e os documentos legais servem de contexto para as respostas da LLM. Requisitos associados:

- RF07: O sistema deve permitir o *upload* e busca de documentos legais - Mecanismo para que um usuário possa submeter e buscar um documento legal, que será utilizado por funções do sistema.

- RF08: O sistema deve permitir o *upload* de processos no formato .zip - Mecanismo para que um analista submeta um processo completo, que pode conter múltiplos documentos, para análise. O formato .zip simplifica a submissão de um conjunto de arquivos em uma única ação.

- RF09: O sistema deve fornecer uma interface para o usuário acesse os processos permitidos - Garante que o usuário tenha uma área de trabalho onde possa visualizar, selecionar e gerenciar os processos aos quais tem permissão de acesso.

- RF10: O sistema deve fornecer uma interface para o usuário acesse as legislações - Garante que o usuário tenha uma área de trabalho onde possa visualizar, selecionar e gerenciar os documentos legais aos quais tem permissão de acesso

- RF17: O sistema deve possuir uma barra de navegação que permita ao usuário navegar pela interface - É um requisito fundamental de usabilidade que garante ao usuário a capacidade de transitar de forma clara e intuitiva entre as principais seções do sistema.

- RNF01: O sistema deve ter uma interface responsiva e de fácil uso - Este requisito de qualidade assegura que as interfaces sejam intuitivas para o usuário alvo e funcionem corretamente.

- RNF03: As terminologias utilizadas na interface devem ser consistentes com a do domínio de processos organizacionais - Este requisito reduz a curva de aprendizado e facilita a utilização do sistema.

- RNF04: O sistema deve fornecer *feedback* visual rápido para as ações do usuário - Para ações como o upload de um arquivo ou a execução de uma busca, a interface deve fornecer uma resposta visual imediata (e.g., uma barra de progresso, um ícone de carregamento) para informar ao usuário que sua ação foi recebida e está sendo processada.

- RNF06: As operações de CRUD na base de dados devem ser completadas em um tempo viável - Este requisito de desempenho garante que operações fundamentais sejam executadas rapidamente para não prejudicar a fluidez do trabalho do usuário.

3.8.1.3 Épico 3: Desenvolvimento do *chatbot*

Descrição: Este épico é responsável por implementar a principal interface de interação entre o analista e a inteligência artificial. O objetivo é desenvolver um *chatbot* mais contextualizado, onde as interações fossem orquestradas pela camada de IA. Este épico também inclue a necessidade de registrar e apresentar o histórico das análises, garantindo a rastreabilidade. Requisitos Funcionais Associados:

- RF06: O sistema deve ter um *chatbot* funcional e contextualizado - Define que o sistema deve fornecer uma interface de conversação que seja contextualizada.

- RF11: O sistema deve registrar dados das interações do usuário com o *chatbot* - Para garantir a rastreabilidade cada interação deve ser salva e associada ao processo em análise

- RF12: O sistema deve fornecer informações das interações do usuário com o *chatbot* - O usuário deve poder reabrir a análise de um processo e ver todo o histórico de conversas com a IA.

- RF05: O *chatbot* deve ser capaz de fornecer ajuda sobre as funcionalidades do sistema - O *chatbot* deve atuar como um mecanismo de ajuda para a utilização do próprio sistema

- RNF05: O sistema deve apresentar respostas não demoradas do *chatbot* - A latência entre a pergunta do usuário e a resposta da IA deve ser suficientemente baixa para

que a interação seja fluida e não interrompa o raciocínio do analista, garantindo que a ferramenta seja produtiva.

- RNF07: As interações e dados do sistema devem ser armazenados de forma segura, com criptografia para dados sensíveis em repouso - Este requisito de segurança garante que o conteúdo dos processos analisados e outras informações sejam protegidos contra acesso não autorizado no banco de dados.

- RNF02: Todo e qualquer conteúdo gerado pela IA deve ser claramente identificado como tal - Para garantir a transparência e a responsabilidade, a interface do *chatbot* deve diferenciar visualmente as mensagens do usuário das respostas geradas pela IA.

3.8.1.4 Épico 4: Desenvolvimento da camada de IA

Descrição: Este épico foca no desenvolvimento da camada de inteligência artificial. O objetivo é construir os componentes de *backend* responsáveis por receber uma requisição, orquestrar a interação com os diversos módulos de IA e produzir uma resposta que fosse contextualizada, validada e confiável. Este épico não lida com a interface do usuário, mas sim com toda a lógica de processamento. Requisitos Associados:

- RF20: A camada de IA deve identificar a requisição do usuário para acionar o fluxo correto no MCP - Garante que a camada de IA possa interpretar a intenção do usuário para que o MCP selecione e execute a cadeia de processamento e os *prompts* adequados para aquela tarefa específica.

- RF21: A camada de IA deve coletar e agregar dados de múltiplas fontes do sistema para construir o contexto do *prompt* - A camada de IA buscará informações como o perfil do usuário, o conteúdo do processo em análise e as legislações pertinentes para injetá-las no *prompt*.

- RF22: O validador semântico deve pré-processar e refinar requisições ambíguas do usuário antes de encaminhá-las ao MCP - Atuando na entrada, este requisito garante que o validador semântico funcione como um filtro de qualidade. Ele pode corrigir ou esclarecer as solicitações do usuário, assegurando que o MCP receba uma instrução clara e otimizada, o que melhora a precisão de todo o fluxo de IA.

- RF23: O validador semântico deve analisar a resposta da LLM para verificar a coerência e a conformidade com as regras de negócio - O validador Semântico inspeciona a respostada LLM para garantir que ela seja coerente para o usuário.

- RNF05: As estruturas de *prompt* do sistema devem ser baseadas em padrões reutilizáveis (MCPs) - Define a lógica de interação com os módulos de IA.

- RNF08: O controle de acesso baseado em Papéis (RBAC) deve ser validado no *backend* a cada requisição - Este requisito de segurança determina que a camada de IA deve operar respeitando as permissões do usuário.
- RNF09 : O tempo de processamento interno da camada de IA não deve impactar na análise do usuário - Este requisito de desempenho foca na eficiência da própria camada. O tempo que o MCP e o validador semântico levam para processar a requisição (excluindo o tempo de espera da API da LLM) deve ser mínimo, para não impactar a experiência do usuário.
- RNF10 : Dados internos da camada de IA devem ser registrados - O sistema deve manter logs de cada etapa dentro da camada de IA.

4 Desenvolvimento do SEI Expert

Este capítulo descreve o processo de desenvolvimento do SEI Expert. A execução do projeto seguiu os princípios da engenharia de *software* ágil, caracterizada pela adaptação do planejamento inicial frente aos desafios emergentes. O foco principal foi validar precocemente os componentes mais críticos da arquitetura, especificamente a integração entre a gestão processual e os recursos de Inteligência Artificial.

4.1 Engenharia de Software

A abordagem adotada seguiu o modelo de desenvolvimento evolutivo. Conforme definido no capítulo no capítulo 3, o trabalho foi dividido em *Sprints* com o objetivo de implementar os requisitos funcionais (RF) e não funcionais (RNF) levantados na etapa de elicitação.

A gestão do projeto buscou atender aos épicos definidos:

- **Épico 1:** Gestão de Acesso e Perfis.
- **Épico 2:** Gerenciamento de Dados.
- **Épico 3:** Desenvolvimento do *Chatbot*.
- **Épico 4:** Desenvolvimento da Camada de IA.

Durante a execução, aplicou-se o princípio *Lean* de "Criar Conhecimento"e "Adiar o Compromisso"(decidir o mais tarde possível com base em fatos). Isso foi crucial quando se identificou que a complexidade dos requisitos de IA (especificamente o RF20 e RF21) exigiria ajustes na arquitetura e execução planejadas.

4.2 Visão da Arquitetura Implementada

A arquitetura do SEI Expert foi desenhada como um sistema *Fullstack*, onde a complexidade do processamento de dados e IA é gerenciada por um *backend* em *Python*, enquanto a interatividade é garantida por um *frontend* reativo.

O sistema é composto por três blocos lógicos principais:

- ***Frontend* (Interface do Usuário):** Construído no ecossistema *React* com *TypeScript*. Utiliza *Material UI (MUI v5)* para uma interface padronizada e *TanStack*

Query para gerenciamento eficiente de estado e cache, garantindo uma experiência fluida ao usuário.

- ***Backend* (API e Lógica de IA):** Diferente de arquiteturas tradicionais que separam lógica de negócio e serviço de dados, o *backend* foi unificado em *Python* utilizando o *framework FastAPI*. Esta escolha permitiu integrar nativamente as bibliotecas de manipulação de dados e IA (como processamento de PDFs e vetores) diretamente na API, reduzindo a latência de microsserviços.

- **Camada de Dados e Vetores:** A persistência utiliza *PostgreSQL* na versão 16. O diferencial arquitetural é o uso da extensão *pgvector*, que transforma o banco de dados relacional em um banco vetorial, permitindo armazenar não apenas dados estruturados (usuários, processos), mas também *embeddings* para busca semântica.

A Figura 19 ilustra a arquitetura e componentes do SEI Expert conforme implementada. Ela demonstra a divisão clara entre a camada de apresentação (*Frontend*), a lógica de negócio unificada com a Inteligência Artificial (*Backend* em FastAPI/Python).

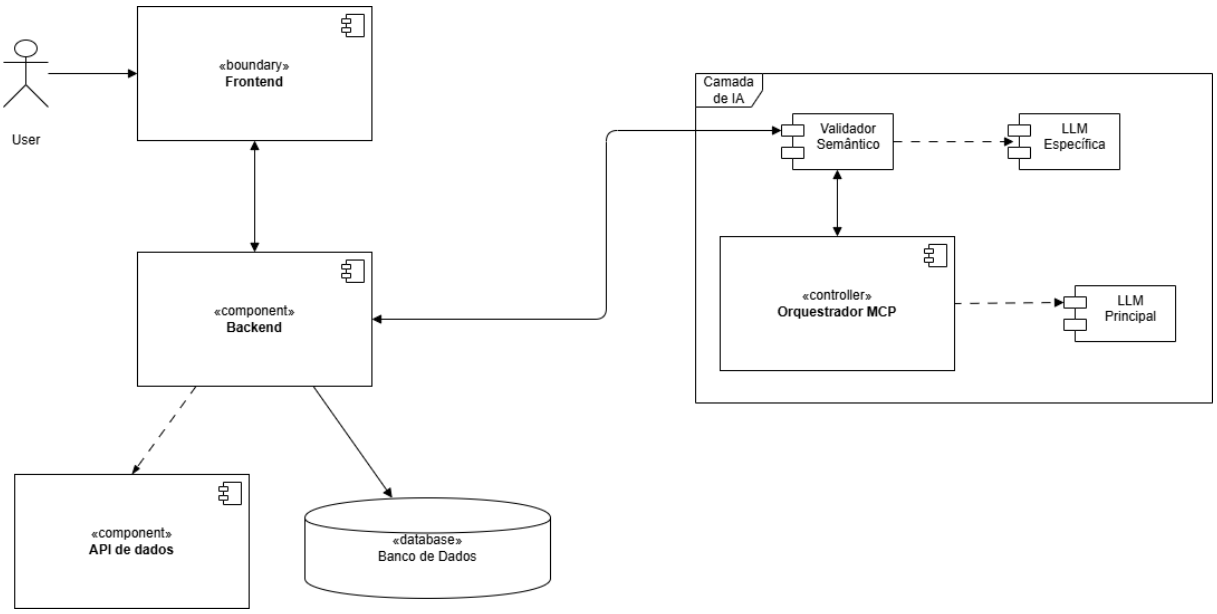


Figura 19 – Diagrama da Arquitetura do Sistema

Fonte: Autor.

4.3 Gestão de Configuração e Tecnologias

A seleção tecnológica priorizou o desempenho assíncrono e a capacidade de processamento de linguagem natural. A Tabela 11 detalha a *stack* utilizada.

Camada	**Tecnologia**	**Função Principal**
Frontend	React 18 + TypeScript	*Framework* principal e tipagem estática.
	Vite (SWC)	Ferramenta de *build* e compilação rápida.
	Material UI (MUI v5)	Biblioteca de componentes visuais e *Data Grid*.
	TanStack Query	Gerenciamento de estado do servidor e cache.
	React Hook Form + Zod	Gerenciamento e validação de formulários.
	Axios	Cliente HTTP para comunicação com a API.
Backend	FastAPI (Python)	*Framework web* de alta performance (ASGI).
	Uvicorn / Gunicorn	Servidores de aplicação assíncronos.
	SQLAlchemy 2.0+	ORM com suporte a *asyncio*.
	Pydantic	Validação de esquemas de dados.
	Python-Jose / Passlib	Autenticação JWT e *hash* de senhas (bcrypt).
	Alembic	Gerenciamento de migrações de banco de dados.
IA e Dados	OpenAI SDK	Integração com LLMs.
	LangChain	Orquestração de fluxos inteligentes, combinando LLMs, busca vetorial e ferramentas externas.
	PyPDF2 / pdfplumber	Extração de texto de documentos legais.
	pgvector	Armazenamento e busca de vetores (*embeddings*).
	BeautifulSoup4	*Parsing* e limpeza de dados HTML.
Infra	PostgreSQL 16	Banco de dados relacional e vetorial.
	Docker / Compose	Containerização e orquestração.

Tabela 11 – Tecnologias e Ferramentas Utilizadas

Fonte: Autor.

A arquitetura híbrida detalhada acima, combinando um *backend* em Node.js para a lógica e um serviço dedicado em *Python* para a inteligência artificial, foi determinante para implementação. Essa segregação permitiu isolar as dependências de processamento de dados e vetorização (como *pgvector* e *LangChain*) da camada de aplicação *web*, garantindo que operações de IA não impactassem a responsividade da interface do usuário. Além disso, o uso de contêineres *Docker* assegurou a consistência do ambiente de desenvolvimento entre as linguagens, mitigando problemas de compatibilidade e facilitando a orquestração dos microsserviços durante as etapas de validação.

4.4 Evolução do Desenvolvimento

A gestão do projeto seguiu uma abordagem adaptativa. Para evidenciar as decisões de engenharia tomadas durante o percurso, esta seção apresenta o contraste entre o cronograma original planejado e a execução real, onde a complexidade da camada de inteligência artificial exigiu uma reestruturação dos componentes finais.

4.4.1 Planejamento Original

O cronograma inicial foi desenhado para cobrir todo o semestre letivo, dividido em 9 *Sprints* quinzenais (com exceção da última), onde cada período focaria na entrega de um ou mais componentes específicos (C1 a C10) que juntos compõem os épicos planejados, conforme definido no capítulo 3.

A estrutura planejada foi:

- **Sprint 1 (18/08 – 31/08): Componente C1 (Base).** Configuração do ambiente, repositório, CI/CD e *healthcheck*.
- **Sprint 2 (01/09 – 14/09): Componente C2 (Identidade).** Implementação de cadastro, login e gestão de sessão de usuários.
- **Sprint 3 (15/09 – 28/09): Componente C3 (Acesso e Admin).** Definição da matriz de papéis (RBAC) e painel administrativo.
- **Sprint 4 (29/09 – 12/10): Componente C4 (Ingestão).** Upload de arquivos e modelagem das tabelas de processos e leis.
- **Sprint 5 (13/10 – 26/10): Componente C5 (Descoberta).** Interfaces de busca, listagem e detalhe de documentos.
- **Sprint 6 (27/10 – 09/11): Componente C6 (Chat).** Implementação da interface visual do chat.
- **Sprint 7 (10/11 – 23/11): Componente C7 (Memória).** Histórico de conversas e auditoria.
- **Sprint 8 (24/11 – 07/12): Componente C8 (IA Mínima) e C9 (Integração SEI).** Implementação básica da lógica de IA e conector com o sistema externo.
- **Sprint 9 (08/12 – 15/12): Componente C10 (Entrega Final).** Testes E2E, documentação e revisão final.

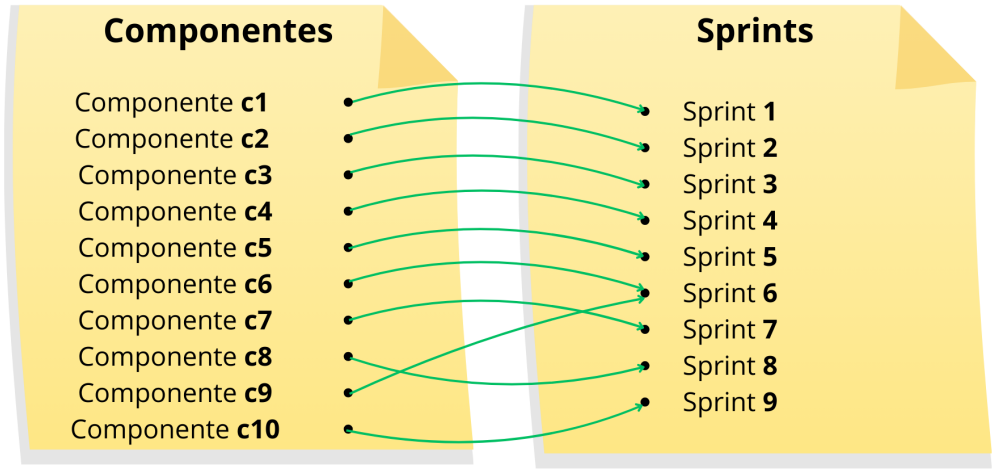


Figura 20 – Planejamento original

Fonte: Autor

4.4.2 Execução Realizada

Durante a execução, as *Sprints* 1 a 5 seguiram o planejamento previsto, estabelecendo a infraestrutura do sistema. No entanto, ao atingir a *Sprint* 6, identificou-se que o componente **C8 (IA Mínima)** era insuficiente para garantir a qualidade das respostas. A complexidade da orquestração de IA exigia que este componente fosse quebrado em microsserviços especializados.

Dessa forma, a execução real a partir de outubro divergiu do plano original: o componente de integração C9 foi descontinuado (substituído apenas pela ingestão de arquivos), os componentes de Chat e Memória (C6, C7) foram adiantados na *sprint* 5 e com isso o componente c8 foi re-planejado em um novo ciclo de 5 *sprints* focado exclusivamente na engenharia da Camada de IA.

O cronograma executado para a finalização do sistema foi:

- ***Sprints* 1 a 7 (18/08 – 26/10):** Executadas conforme o planejado, apenas com um adiantamento das *sprints* 6 e 7 terminadas juntas da *sprint* 5. Com isso foi entregado a Base, Identidade, Acesso, Ingestão de Dados, Memória e Histórico.
- ***Sprint* 1 de IA (28/10 – 03/11): Foco: Validador Semântico (Entrada).** Foi implementado um componente dedicado a classificar a intenção da requisição do usuário antes de acionar qualquer modelo, otimizando custos e precisão.
- ***Sprint* 2 de IA (04/11 – 10/11): Foco: Orquestrador MCP.** Implementação do *Model Context Protocol*. Nesta etapa, a lógica de "memória"foi integrada diretamente ao orquestrador para permitir a construção dinâmica de contexto com base nos documentos ingeridos.
- ***Sprint* 3 de IA (11/11 – 17/11): Foco: Geração (LLM Principal).** Integração efetiva com o modelo GPT-4o-mini. O foco foi a engenharia de *prompt* para garantir respostas estritas em JSON, viabilizando a renderização correta no *frontend*.
- ***Sprint* 4 de IA (18/11 – 24/11): Foco: Validador Semântico (Saída).** Criação de uma camada de verificação pós-geração, garantindo que as citações legislativas, referências a processos e todas as requisições fossem válidas e completas antes da exibição ao usuário.
- ***Sprint* 5 de IA (25/11 – 01/12): Foco: Integração e MVP.** Convergência do *frontend* (Chat) com o novo serviço de *backend* de IA, consolidando as funcionalidades de histórico e interação em tempo real.

Essa reestruturação transformou o componente relacionado ao desenvolvimento da camada de IA em outros componentes, elevando a capacidade de desenvolvimento e

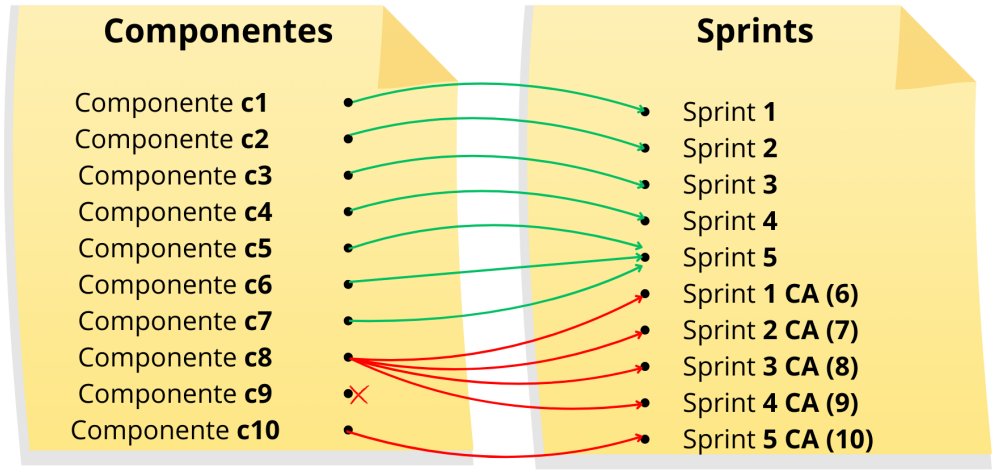


Figura 21 – Planejamento original

Fonte: Autor

tempo hábil para o desenvolvimento da camada de IA. Essa reestruturação foi possível já que as *sprints* 7 e 8 foram adiantadas durante a *sprint* 5. Então as semanas restantes foram re-planejadas com o foco no desenvolvimento da camada de IA.

4.5 Infraestrutura e processo de entrega

Para garantir a integridade do código e a disponibilidade do sistema, foram adotadas práticas de versionamento e implantação contínua.

4.5.1 Versionamento de código (Git/GitHub)

Todo o código fonte do projeto foi gerido utilizando o sistema de controle de versão *Git*, hospedado em um repositório privado no *GitHub*. A estrutura do repositório seguiu o padrão de *Monorepo*, onde tanto o *frontend* quanto os serviços de *backend* coexistem no mesmo versionamento, facilitando a sincronização entre as alterações de contrato de API e as interfaces de usuário. O código-fonte está disponível em https://github.com/cedis-unb/SEIexpert

O fluxo de trabalho baseou-se em *commits* atômicos e descritivos, permitindo a rastreabilidade de cada funcionalidade implementada, desde a configuração inicial do ambiente até os ajustes na engenharia de *prompt*.

4.5.2 Ambiente de implantação

A estratégia de implantação foi dividida para atender aos requisitos específicos de cada camada da aplicação:

- ***Frontend*:** O cliente *web* em React foi implantado na plataforma **Vercel**. A escolha deveu-se à sua integração nativa com o ecossistema Vite/React e à capacidade de *Continuous Deployment* (CD). A cada *push* na branch principal (`main`) do repositório, a Vercel automaticamente dispara o processo de *build*, executa as otimizações de ativos estáticos e atualiza a versão em produção em questão de minutos.
- ***Backend* e IA:** Para lidar com processos de longa duração, como o processamento de arquivos `.zip` e a vetorização de documentos , e garantir a consistência das bibliotecas *Python* de inteligência artificial, os serviços foram desenvolvidos seguindo os princípios de **aplicações nativas em nuvem**. Durante o desenvolvimento e validação local, utilizou-se **Docker** para garantir o isolamento de dependências (como drivers de manipulação de PDF e o banco de dados vetorial). Em produção, a arquitetura foi implantada em ambiente PaaS (*Platform as a Service*) no Azure, utilizando o *App Service* para a execução da API e o *Azure Database for PostgreSQL Flexible Server* para a persistência.

4.6 Limitações e dívidas técnicas

A comparação entre o planejamento idealizado e a execução real resultou em algumas dívidas técnicas conscientes, assumidas para viabilizar a entrega do MVP:

- **Integração simulada com o SEI:** Conforme mencionado, a integração direta via API (SOAP) foi substituída apenas pela importação de arquivos `.zip` processados localmente. Embora funcional para análise, existe apenas uma etapa manual feita pelo usuário.
- **Processamento Síncrono:** No planejamento, idealizava-se um sistema de filas para ingestão. No realizado apenas um PDF é ingerido por vez. A migração para filas de tarefas (ex: *Celery*) foi mapeada como melhoria futura.
- **Infraestrutura de Vetores:** A necessidade não prevista de um banco de dados vetorial (*pgvector*) para o RAG aumentou a complexidade da infraestrutura em comparação ao banco relacional padrão planejado inicialmente.

5 O SEI Expert

Este capítulo apresenta os resultados do desenvolvimento do SEI Expert. A seguir, o sistema é detalhado através de suas interfaces, demonstrando o fluxo de uso, desde a autenticação com controle de acesso até a interação com a inteligência artificial, evidenciando o tratamento de *tokens* e a gestão documental.

A aplicação opera em ambiente *web*, integrando o *frontend* em React com os serviços de *backend* e IA.

5.1 Controle de acesso e identidade

A entrada no sistema implementa os requisitos de segurança e segregação de funções (RBAC) definidos.

5.1.1 Autenticação e recuperação

A Figura 22 exibe a tela de *login*. O sistema utiliza *tokens* JWT (*JSON Web Tokens*) para gerenciar a sessão. Caso o usuário esqueça suas credenciais, o fluxo de recuperação (Figura 23) permite o envio de instruções para o e-mail cadastrado.

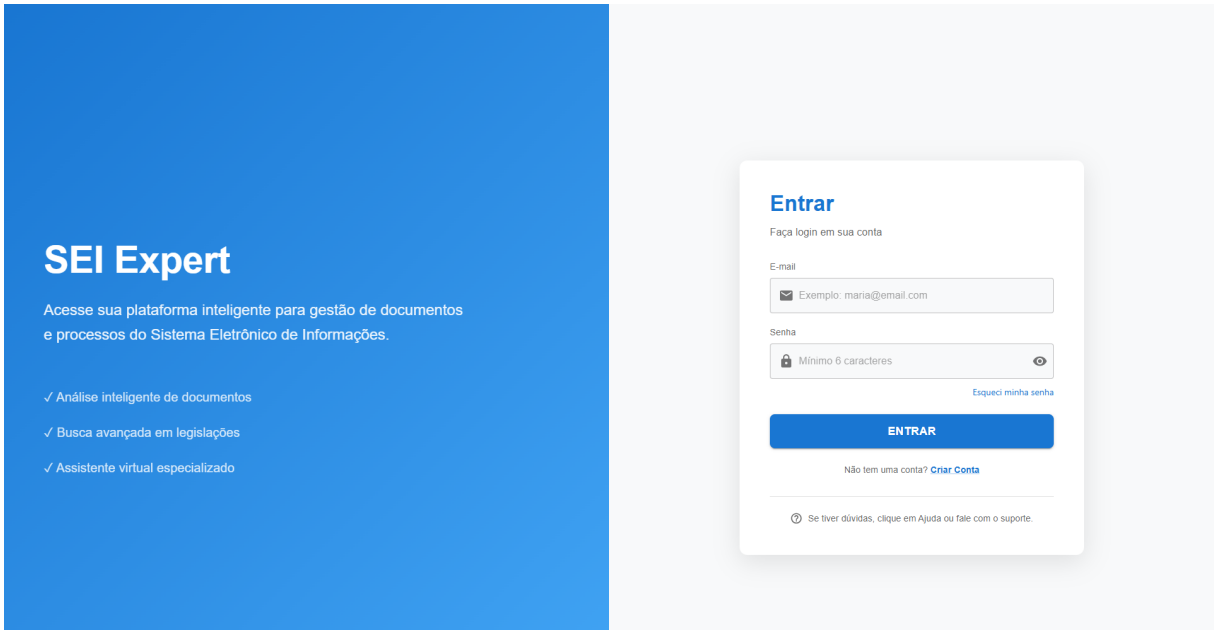


Figura 22 – Tela de *login* do sistema

Fonte: Autor.

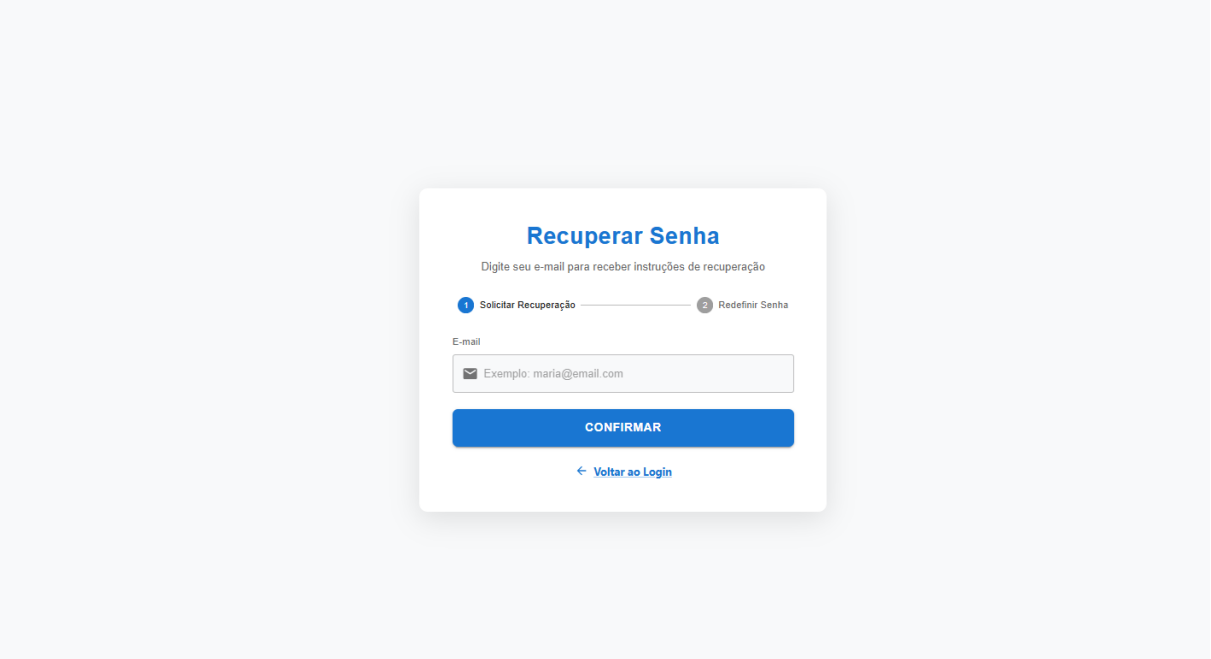


Figura 23 – Fluxo de recuperação de senha

Fonte: Autor.

5.1.2 Cadastro com definição de papéis (*Roles*)

Um ponto da implementação é a atribuição de perfis no momento do cadastro. Conforme mostra a Figura 24, o usuário deve selecionar sua função (analista, especialista ou gestor).

Esta escolha define as *roles* no banco de dados:

- **Analista:** acesso padrão para submissão e análise de processos.
- **Especialista:** permissão adicional para gerenciar a base de conhecimento (leis).
- **Gestor:** acesso a alguns painéis administrativos e métricas de uso.

5.2 *Dashboard* e visão geral

Ao acessar o sistema, o usuário é recebido pelo *dashboard* (Figura 25). Esta tela centraliza as métricas principais, como processos analisados recentemente e atalhos para as funções de ingestão.

Para auxiliar na navegação, o sistema conta com um componente de *chat* de ajuda, que pode ser acionado para resolver dúvidas do sistema (Figura 26).

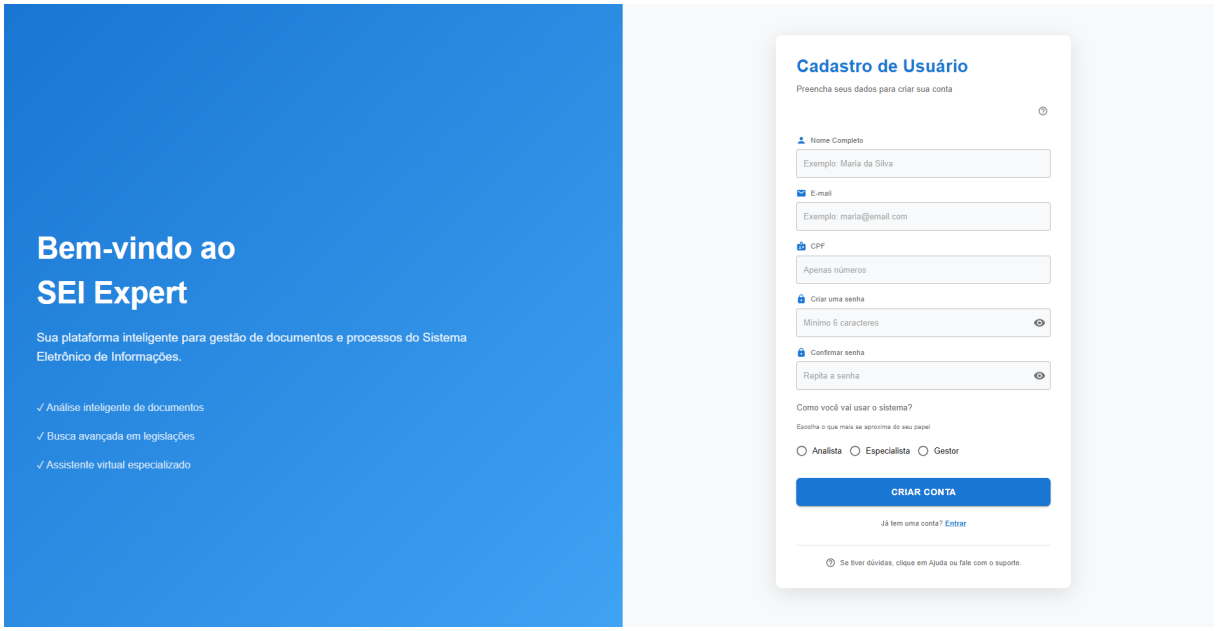


Figura 24 – Tela de cadastro com seleção de perfil (*role*)

Fonte: Autor.

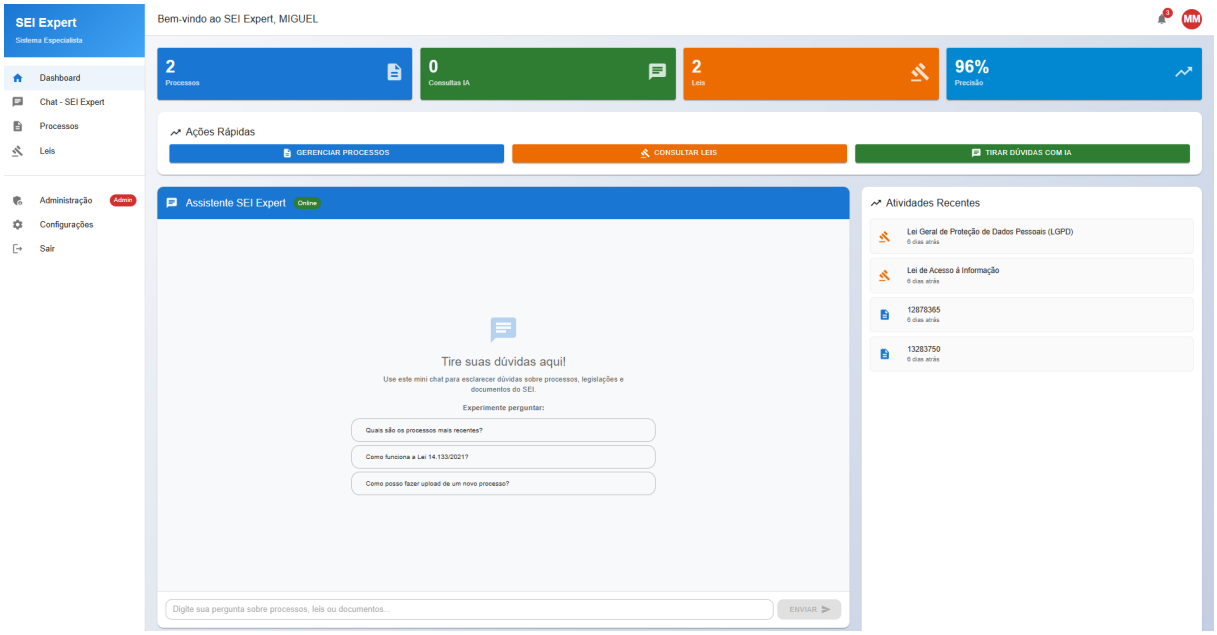


Figura 25 – Visão geral do *dashboard* administrativo

Fonte: Autor.

5.3 Módulo de processos e gestão documental

Este módulo materializa a funcionalidade de ingestão e organização dos dados processuais, apresentando diferentes níveis de visualização.

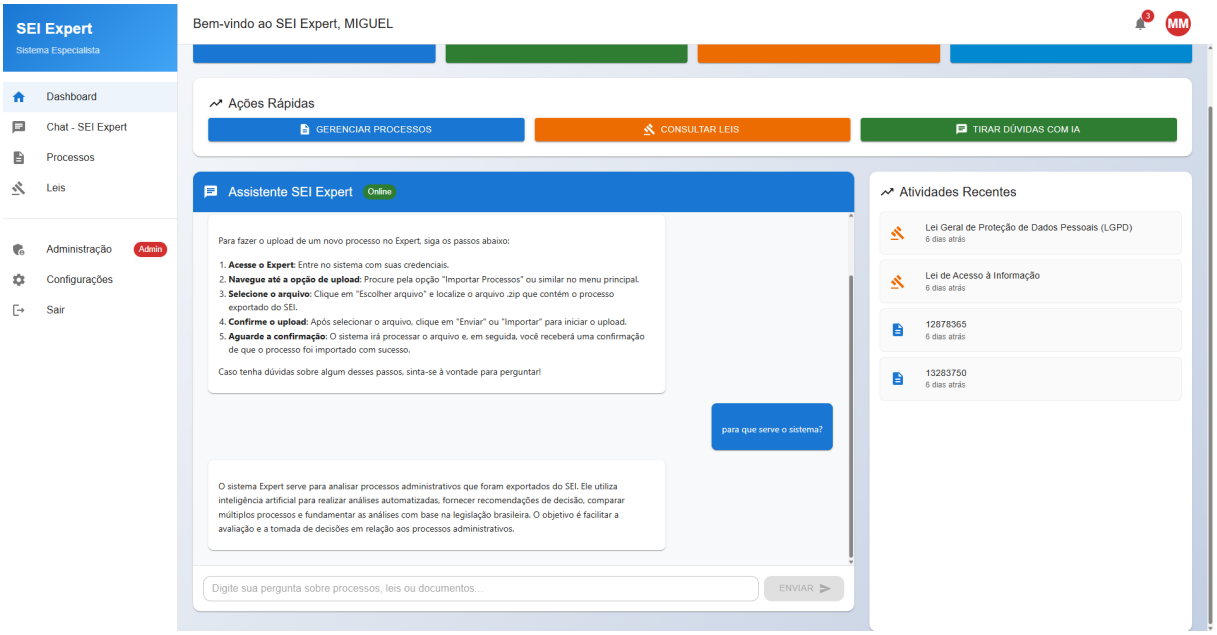


Figura 26 – *Dashboard* com a interface de ajuda ativa

Fonte: Autor.

5.3.1 Listagem e organização

A Figura 27 apresenta a listagem dos processos importados no sistema, permitindo uma visão dos processos ingeridos pelo sistema.

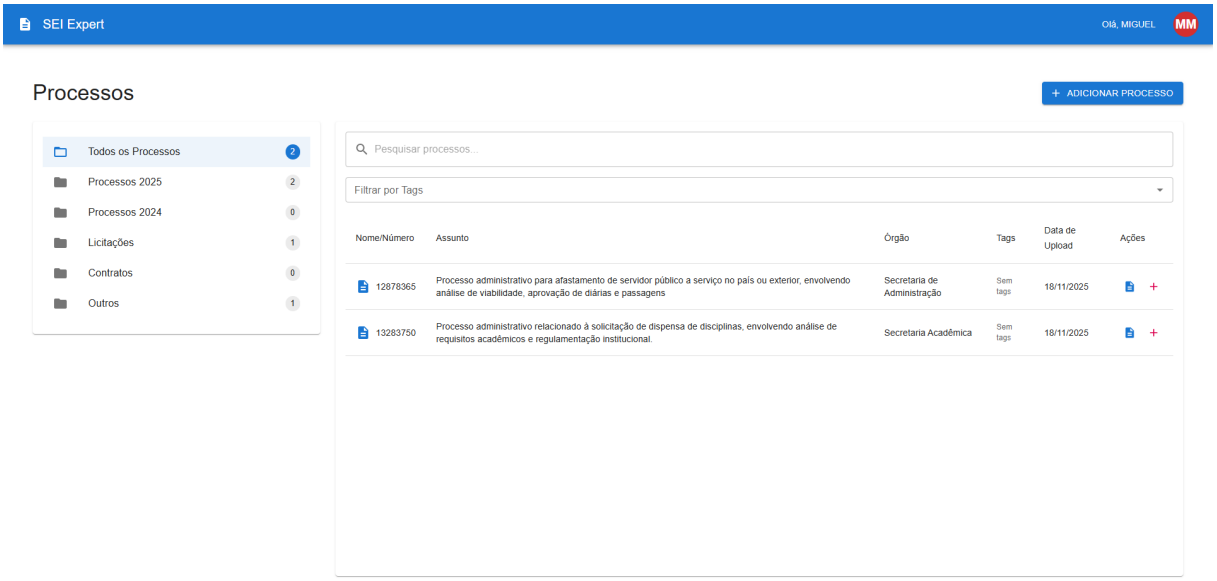


Figura 27 – Listagem geral de processos importados

Fonte: Autor.

5.3.2 Sistema de *tags* e categorização

Para facilitar a triagem, foi implementado um sistema de etiquetas. A Figura 28 demonstra como as *tags* (ex: "Licitação", "Urgente") auxiliam na identificação e filtragem dos processos.

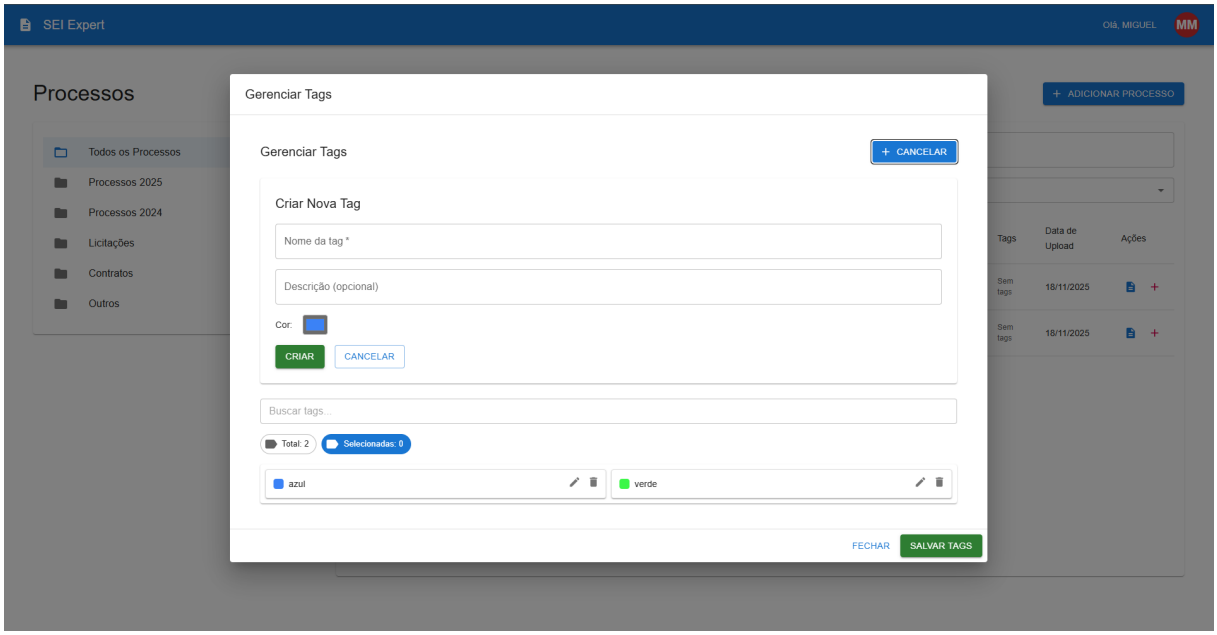


Figura 28 – Visualização de processos com sistema de *tags* coloridas

Fonte: Autor.

5.3.3 Detalhamento e arquivos

Ao selecionar um processo, o usuário acessa a visão detalhada (Figura 29). Nesta tela, é possível visualizar a árvore de documentos extraída do arquivo `.zip` e realizar o *download* para conferência.

5.4 Base de conhecimento (legislação)

O módulo de legislação alimenta a base de conhecimento da IA (RAG), permitindo a gestão das normas que embasarão as análises.

5.4.1 Gestão de normas

A Figura 30 exibe a interface de gerenciamento de leis, onde o especialista pode cadastrar novos textos legais e regulamentos.

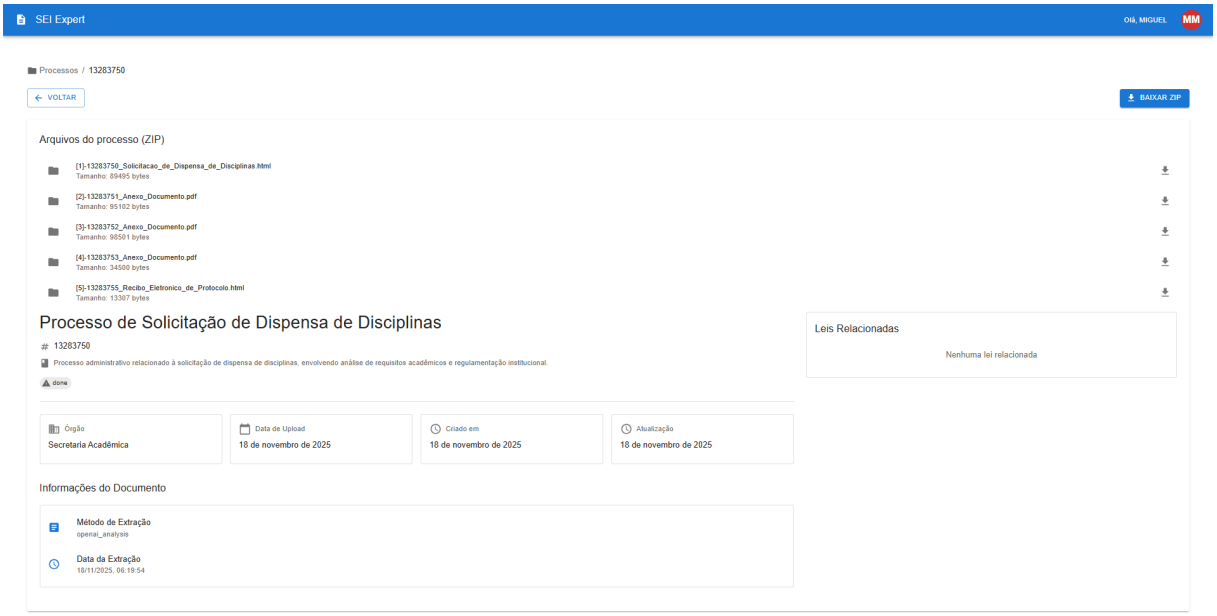


Figura 29 – Detalhe do processo aberto com lista de arquivos

Fonte: Autor.

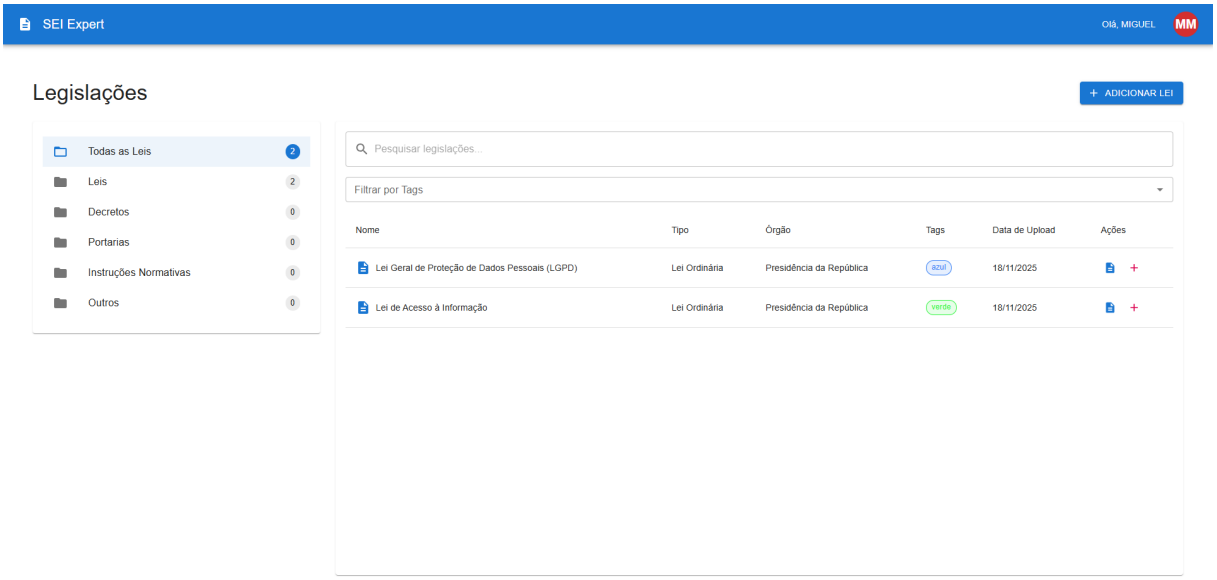


Figura 30 – Interface de listagem da base de legislação

Fonte: Autor.

5.4.2 Classificação por *tags*

Assim como nos processos, as leis são categorizadas. A Figura 31 mostra a aplicação de *tags* às normas, o que permite que haja uma filtragem e acesso rápido à lei desejada.

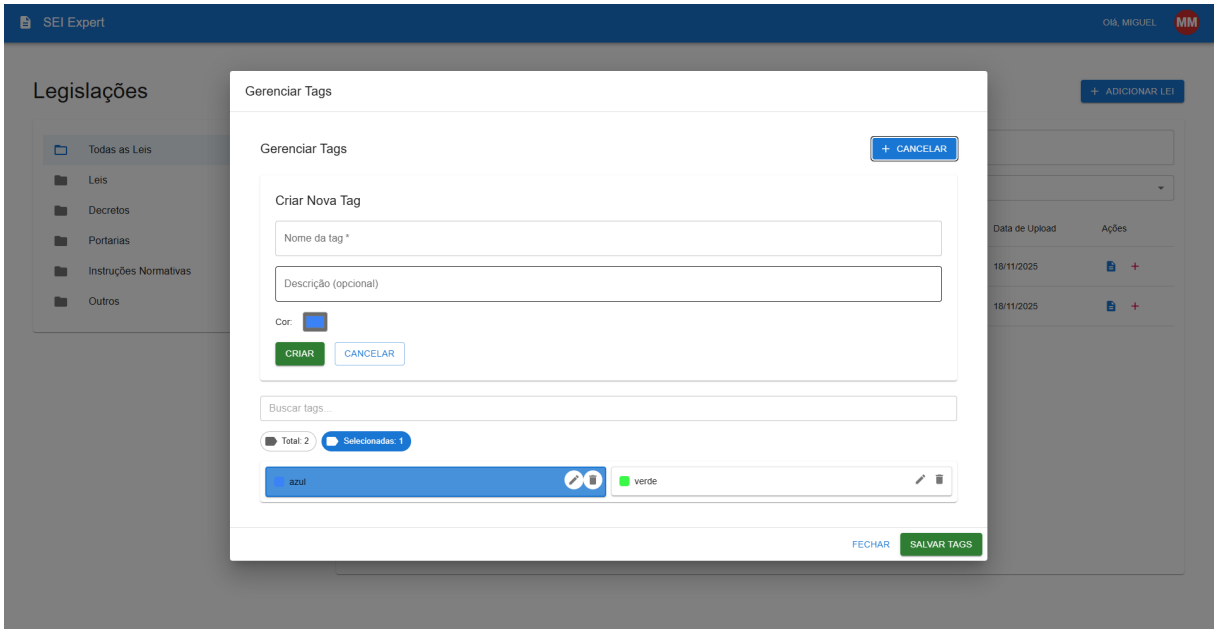


Figura 31 – Legislação categorizada com *tags* semânticas

Fonte: Autor.

5.4.3 Visualização de conteúdo legal

A Figura 32 demonstra a visualização expandida de uma lei, permitindo que o usuário confira as informações que poderá ser utilizado pela inteligência artificial.

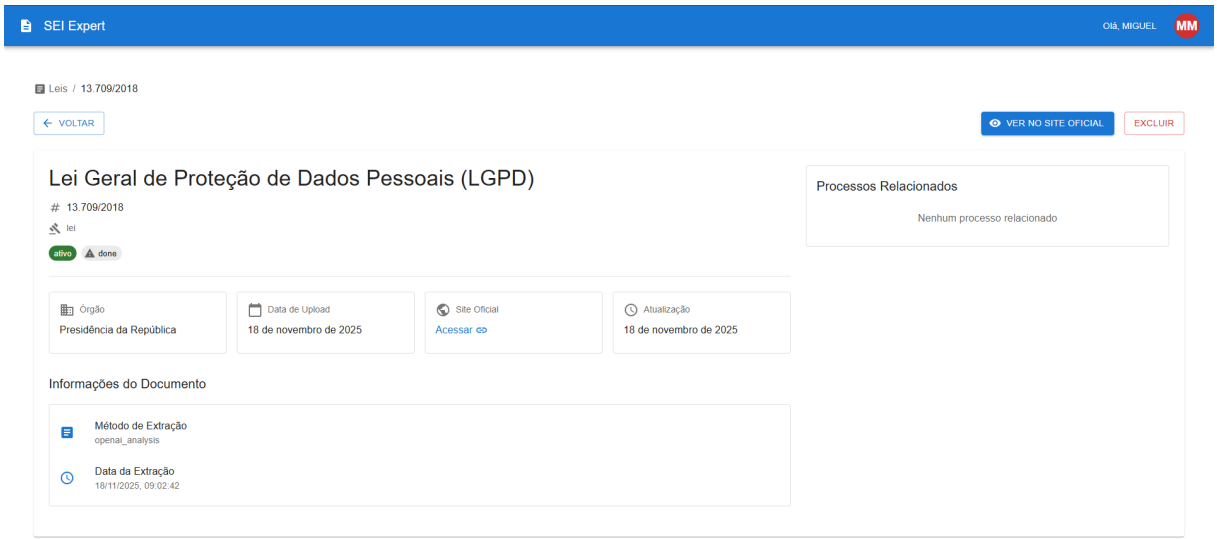


Figura 32 – Visualização detalhada do texto da lei

Fonte: Autor.

5.5 O *Chatbot* e análise

A interface de conversação atua como o principal canal de comunicação entre o usuário e o sistema, operacionalizando as técnicas de engenharia de *prompt* e de uso *tokens* para sumarizar a conversa por exemplo.

5.5.1 Interface de conversação

A Figura 33 apresenta a tela inicial do *chat*, pronta para receber os comandos do analista.

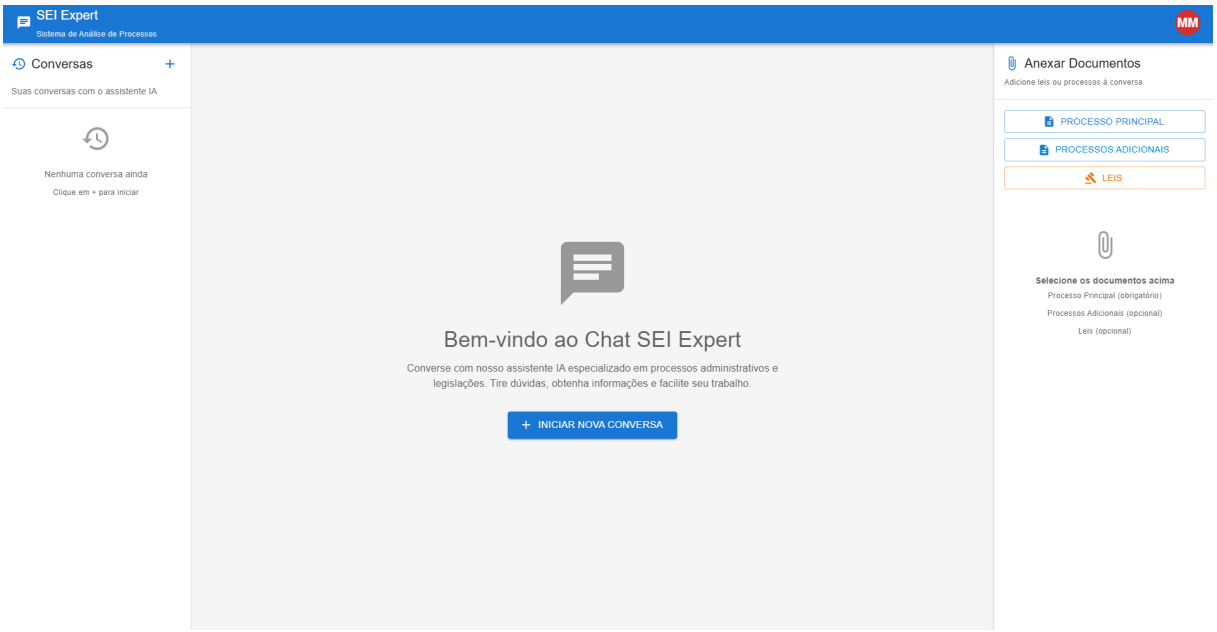


Figura 33 – Tela inicial do assistente virtual inteligente

Fonte: Autor.

5.5.2 Análise contextual de processos

Na Figura 34, observa-se o sistema realizando uma análise complexa. A IA utiliza o contexto dos documentos do processo para gerar uma análise pertinente e útil para o analista.

5.5.3 Referência legislativa

Um diferencial técnico é a capacidade de citar e anexar leis específicas. A Figura 35 mostra a IA utilizando uma lei anexada ao contexto..

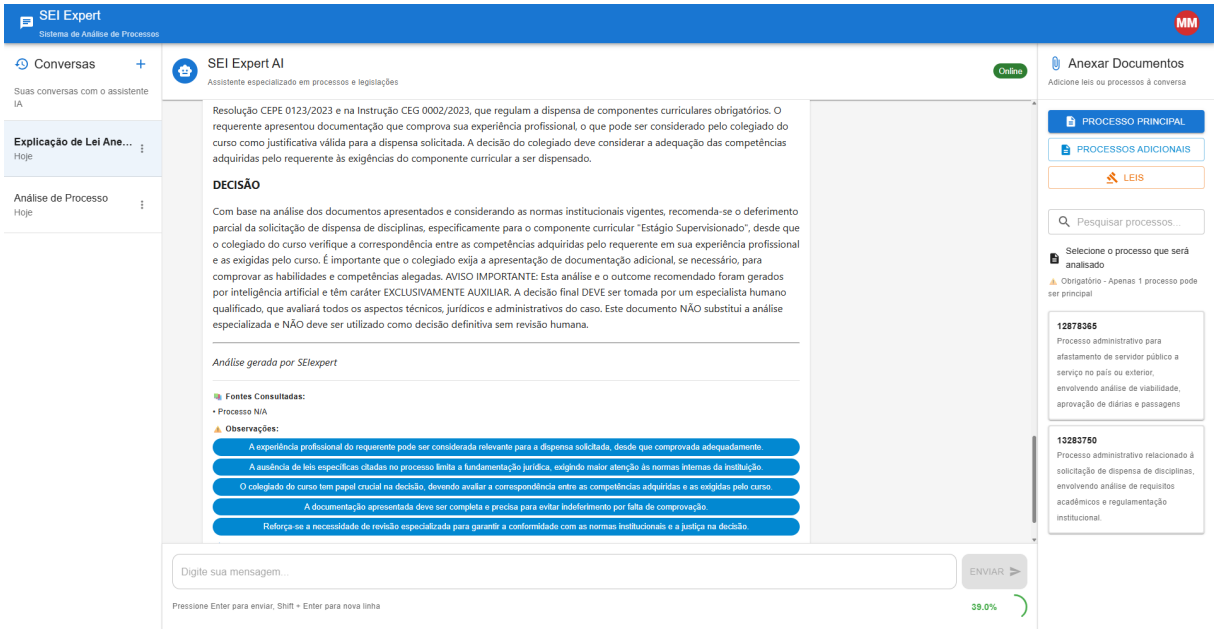


Figura 34 – *Chatbot* realizando análise detalhada de um processo

Fonte: Autor.

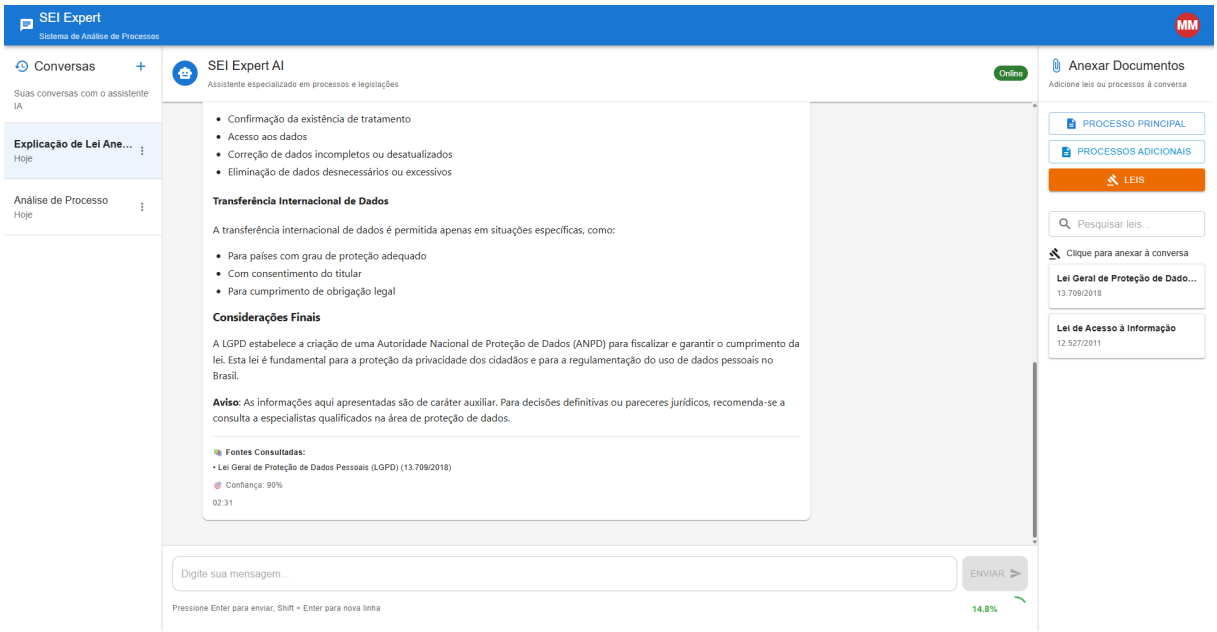


Figura 35 – Análise com citação direta de lei anexada ao contexto

Fonte: Autor.

5.6 Administração e configuração

Para os administradores, o sistema oferece ferramentas de controle e auditoria.

5.6.1 Painéis administrativos

A Figura 36 apresenta a visão geral da administração, com métricas de usuários e atividades recentes.

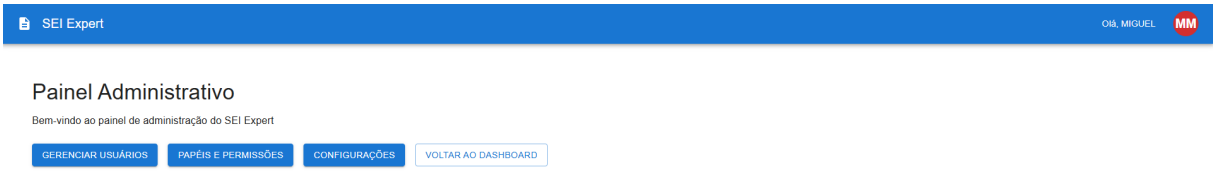


Figura 36 – Visão geral do painel administrativo

Fonte: Autor.

A gestão de usuários é detalhada na Figura 37, onde o administrador pode alterar permissões e visualizar o *status* de cada conta.

Métricas adicionais de desempenho do sistema e *logs* de auditoria são visualizados na Figura 38, garantindo a rastreabilidade das ações.

5.6.2 Configurações de perfil

Por fim, a Figura 39 mostra a tela de configurações pessoais, onde o usuário pode gerenciar seus dados e preferências de notificação.

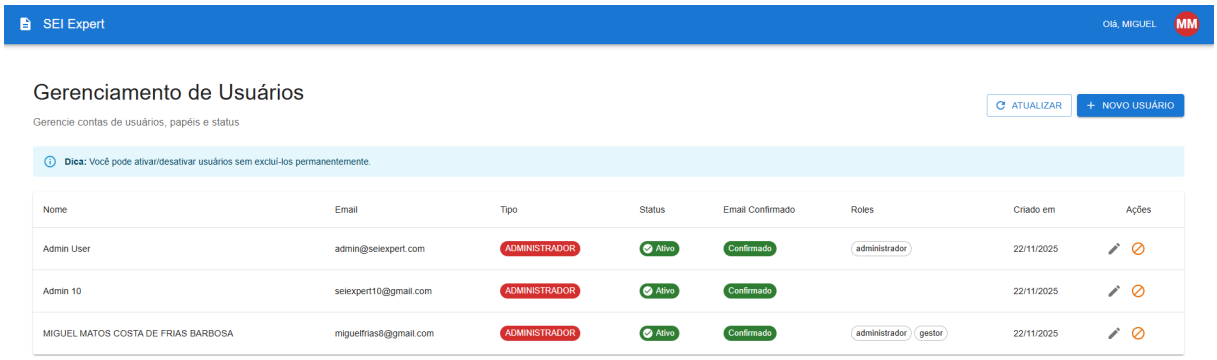


Figura 37 – Gerenciamento de usuários e permissões

Fonte: Autor.

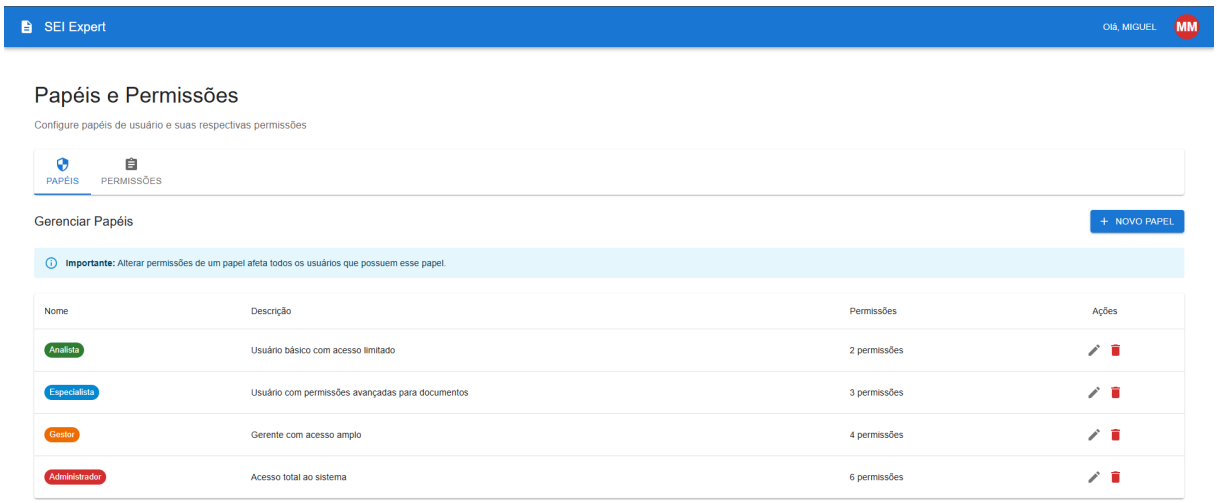


Figura 38 – Métricas detalhadas e auditoria do sistema

Fonte: Autor.

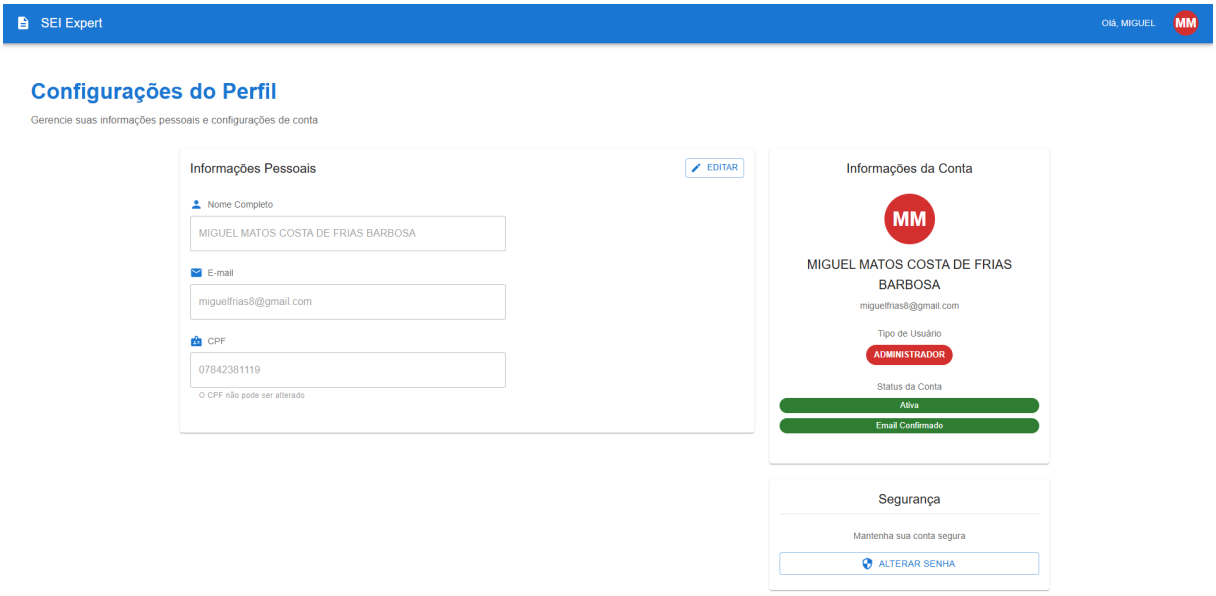


Figura 39 – Tela de configurações de perfil do usuário

Fonte: Autor.

6 Conclusão e trabalhos futuros

O desenvolvimento deste trabalho teve como propósito propor e implementar um sistema voltado à análise e gestão de processos organizacionais. A solução entregue, denominada SEI Expert, incorporou modelos de linguagem de grande escala (LLMs) e protocolos de contexto em uma ferramenta funcional.

Ao longo da execução, a abordagem metodológica ágil permitiu adaptar o escopo inicial. O resultado foi a construção de uma arquitetura integrando um *frontend* com microsserviços em Python e Node.js, capazes de processar linguagem natural com precisão jurídica.

6.1 Síntese dos resultados

Entre os principais resultados obtidos, destaca-se a efetivação da camada de inteligência artificial. Diferente de uma integração simples via API, foi desenvolvido um ecossistema composto por um validador semântico e um orquestrador. Essa engenharia permitiu que o sistema não apenas "conversasse", mas compreendesse a intenção do analista e utilizasse a técnica RAG (*Retrieval-Augmented Generation*) para fundamentar respostas em leis e documentos reais.

A prototipação inicial evoluiu para um produto de software (MVP) operacional que integrou os fluxos administrativos e a inteligência artificial. A implementação da interface de *chat* demonstrou eficácia ao mitigar a complexidade inerente à análise de processos, oferecendo ao analista uma ferramenta capaz de sintetizar grandes volumes de informação jurídica com agilidade.

6.2 Limitações e adaptações

É importante ressaltar as adaptações realizadas em relação ao planejamento original. A integração direta com a API do Sistema Eletrônico de Informações (SEI) via serviços SOAP, embora arquitetada, não está contida na implementação final, sendo substituída por um mecanismo de ingestão via arquivos (*upload* de pacotes `.zip`).

Esta decisão não comprometeu a validação da lógica de análise: o sistema demonstrou capacidade de processar a estrutura de dados do SEI (árvore de documentos).

6.3 Trabalhos futuros

O SEI Expert estabelece uma base sólida para evoluções futuras determinadas por níveis em seu planejamento, que já incluem muitos pontos de continuidade do trabalho. Mas para enfatizar alguns desses pontos, sugerem-se as seguintes frentes:

- **Conector Nativo SEI:** Implementação do módulo de conexão direta via API/SOAP eliminando a necessidade de *upload* manual.
- **Refinamento do RBAC:** Evolução do sistema de permissões para permitir o compartilhamento granular de análises entre usuários específicos, e não apenas por perfil de acesso.
- **Processamento Assíncrono:** Adoção de filas de mensagens (ex: RabbitMQ ou Celery) para o processamento de processos volumosos, melhorando a experiência do usuário durante a ingestão de dados.
- ***Feedback* por Reforço (RLHF):** Implementação de um mecanismo onde o feedback dos analistas nas respostas da IA sirvam para realizar o ajuste fino (*fine-tuning*) do modelo de linguagem específico para o domínio administrativo.

Conclui-se que o trabalho atingiu seus objetivos gerais e específicos, entregando uma ferramenta que demonstra, na prática, como a engenharia de software moderna pode ser aplicada para modernizar e dar eficiência à gestão pública.

Referências

DUMAS, M. et al. *Fundamentals of Business Process Management*. 2nd. ed. [S.l.]: Springer, 2018. Citado 5 vezes nas páginas 19, 20, 30, 31 e 32.

EHTESHAM, A. et al. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279v2*, May 2025. Citado na página 40.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. Citado 3 vezes nas páginas 32, 34 e 35.

HOU, X. et al. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM*, 2025. Disponível em: <https://arxiv.org/abs/2503.23278v2>. Citado 2 vezes nas páginas 40 e 41.

JURAFSKY, D.; MARTIN, J. H. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Second. [S.l.]: Pearson Prentice Hall, 2009. Citado na página 36.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. Third edition draft. [S.l.: s.n.], 2025. Draft of January 12, 2025. Copyright ©2024. Citado 5 vezes nas páginas 35, 36, 37, 39 e 40.

KRISHNAN, N. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *arXiv preprint arXiv:2504.21030*, 2025. Disponível em: <https://arxiv.org/abs/2504.21030v1>. Citado 2 vezes nas páginas 40 e 41.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, v. 33, p. 9459–9474, 2020. Citado na página 39.

MANNING, C. D.; SCHüTZE, H. *Foundations of Statistical Natural Language Processing*. [S.l.]: MIT Press, 1999. Citado na página 36.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997. Citado na página 33.

REINEHR, S. *Engenharia de requisitos*. Porto Alegre: SAGAH, 2020. ISBN 978-65-5690-067-4. Citado 2 vezes nas páginas 26 e 27.

ROCHA, M.; CORTEZ, P.; NEVES, J. M. *ANÁLISE INTELIGENTE DE DADOS - Algoritmos e Implementação em JAVA*. [S.l.]: LIDEL, 2008. Citado na página 33.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4. ed. Upper Saddle River, NJ: Prentice Hall, 2020. ISBN 9780134610993. Citado 4 vezes nas páginas 32, 33, 34 e 36.

SOMMERVILLE, I. *Software Engineering*. 9th. ed. USA: Addison-Wesley Publishing Company, 2007. ISBN 978-0-321-31379-9, 0-321-31379-8. Citado 6 vezes nas páginas 25, 26, 27, 28, 29 e 30.

TRF4. *Manual de Desenvolvimento de Módulos para o SEI*. [S.l.], 2018. Versão 3.0.0. Disponível em: <https://softwarepublico.gov.br/social/articles/0005/4736/SEI_Manual_Desenvolvimento_de_Modulos.pdf>. Citado na página 42.

TUNSTALL, L.; WERRA, L. von; WOLF, T. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. Revised edition. Sebastopol, CA: O'Reilly Media, 2022. ISBN 978-1-098-13679-6. Citado na página 38.

VASWANI, A. et al. Attention is all you need. *Advances in Neural Information Processing Systems*, v. 30, p. 5998–6008, 2017. Citado 4 vezes nas páginas 36, 37, 38 e 39.

WESKE, M. *Business Process Management: Concepts, Languages, Architectures*. 3rd. ed. [S.l.]: Springer, 2019. Citado na página 19.