LICITA.AI: Uma aplicação de Inteligência Artificial para análise de editais e apoio à participação em contratações públicas

Trabalho de Conclusão de Curso

Alda Monique Gonçalves de Alcantara Santos

Grinaldo Lopes de Oliveira
Orientador

Instituto Federal da Bahia - IFBA
Curso de Análise e Desenvolvimento de Sistemas
Campus Salvador

Salvador, Bahia, Brasil
Julho de 2026

Sumário

RESUMO

Este Trabalho de Conclusão de Curso apresenta o desenvolvimento do Licita.AI, uma plataforma web destinada à análise de editais e ao apoio à participação em contratações públicas, utilizando recursos de Inteligência Artificial. A sanção da Lei nº 14.133/2021 (Nova Lei de Licitações e Contratos Administrativos) introduziu diretrizes que adicionam novos desafios para os licitantes, sobretudo para pequenas e médias empresas. Para mitigar esse cenário, a solução proposta possui uma arquitetura de monólito modular desenvolvida em Python com o framework FastAPI, interface visual em React com TypeScript e a aplicação de técnicas de Retrieval-Augmented Generation (RAG) integradas aos modelos da OpenAI. A plataforma consome dados públicos por meio da API do Portal Nacional de Contratações Públicas (PNCP) para realizar o monitoramento e a extração automatizada de editais publicados. A partir dessa coleta, o sistema automatiza a análise documental e notifica os usuários sobre a aderência entre seus perfis institucionais e os certames mais recentes. Como diferencial, a plataforma realiza uma análise automatizada de compatibilidade que confronta o perfil institucional do licitante com as exigências e os requisitos de habilitação de cada edital, apoiada em modelos de linguagem. Como resultado, obteve-se um protótipo funcional de apoio à participação em certames, oferecendo recursos inteligentes para consulta e processamento de grandes volumes de informação. Por fim, a plataforma conta com suporte bilíngue (português e inglês) e integração com serviços de notificação por e-mail e WhatsApp.

Palavras-chave: Licitações. Inteligência Artificial. Portal Nacional de Contratações Públicas. Python. React.

ABSTRACT

This Undergraduate Thesis presents the development of Licita.AI, a web platform designed for bidding document analysis and support for participation in public procurement using Artificial Intelligence features. The enactment of Law No. 14,133/2021 (the New Public Procurement and Administrative Contracts Law) introduced guidelines that add new challenges for bidders, especially for small and medium-sized enterprises. To mitigate this scenario, the proposed solution features a modular monolith architecture developed in Python with the FastAPI framework, a visual interface in React with TypeScript, and the application of Retrieval-Augmented Generation (RAG) techniques integrated with OpenAI models. The platform consumes public data through the National Public Procurement Portal (PNCP) API to perform the automated monitoring and extraction of published bidding documents. Based on this collection, the system automates document analysis and notifies users about the adherence between their institutional profiles and the latest procurement processes. As a differentiating factor, the platform performs an automated compatibility analysis that cross-references the bidder's institutional profile with the requirements and qualification criteria of each bidding document, supported by language models. As a result, a functional prototype to support participation in procurement processes was achieved, offering intelligent resources for querying and processing large volumes of information. Finally, the platform features bilingual support (Portuguese and English) and integration with email and WhatsApp notification services.

Keywords: Public procurement. Artificial Intelligence. National Public Procurement Portal. Python. React.

1 Visão Geral

O Licita.AI é uma aplicação web que utiliza recursos de inteligência artificial e integração com dados abertos do governo federal para análise de contratações públicas. Sua proposta principal é capturar editais por meio do Portal Nacional de Contratações Públicas (PNCP) e, com base no perfil do usuário (pessoa jurídica ou física), avaliar por meio de Inteligência Artificial a compatibilidade entre as exigências do certame e a área de atuação descrita. A aplicação busca ampliar a competitividade no ramo de licitações, diminuir os esforços de leitura de documentos extensos, além de democratizar a linguagem, convertendo a linguagem jurídica para uma linguagem mais acessível.

A solução adota arquitetura de monólito modular em FastAPI, com ingestão de dados via API oficial do PNCP, integração com o modelo GPT-4o mini e técnica de Geração Aumentada por Recuperação (RAG), e *front-end* em React com TypeScript. Sua motivação decorre da Lei nº 14.133/2021, a nova Lei de Licitações e Contratos Administrativos, que surge também como uma tentativa de modernização e simplificação dos procedimentos licitatórios, historicamente marcados por complexidade e burocracia (PEREIRA, 2024).

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma plataforma web que, integrada ao Portal Nacional de Contratações Públicas (PNCP) e apoiada por técnicas de inteligência artificial, automatize a busca, a análise e a avaliação de compatibilidade de certames com o perfil do usuário, reduzindo as barreiras operacionais e de linguagem que dificultam a participação no processo licitatório, sobretudo de micro e pequenas empresas e de profissionais autônomos.

1.1.2 Objetivos Específicos

- Integrar a aplicação à API pública do PNCP para a obtenção automatizada de editais de licitação;
- Permitir a estruturação do perfil do usuário, utilizado como referência para a avaliação de compatibilidade;
- Empregar modelos de linguagem para interpretar editais, evidenciar requisitos e prazos e estimar a aderência ao perfil cadastrado;
- Implementar um assistente conversacional baseado na técnica de Geração Aumentada por Recuperação (RAG), com respostas fundamentadas no conteúdo dos editais e rastreabilidade das fontes;
- Assegurar o isolamento de dados entre os usuários por meio de arquitetura *multi-tenant*.

1.2 Declaração do Problema

O processo licitatório brasileiro representa uma das principais vias de contratação pública. Estudos do Instituto de Pesquisa Econômica Aplicada estimam que o mercado de compras governamentais brasileiro representa, em média, 12,5% do Produto Interno Bruto nacional (RIBEIRO; INÁCIO JÚNIOR, 2019). Apenas em 2025, as contratações homologadas no âmbito do Portal Nacional de Contratações Públicas (PNCP) totalizaram cerca de R$ 1 trilhão, dos quais R$ 272,6 bilhões corresponderam a processos de micro e pequenas empresas (BRASIL, 2026). Para empresas fornecedoras de bens e serviços ao governo, participar de licitações pode significar acesso a contratos relevantes e crescimento sustentável. No entanto, a complexidade operacional envolvida nesse processo representa uma barreira significativa.

A promulgação da Lei nº 14.133, de 1º de abril de 2021, substituiu o marco regulatório anterior, vigente desde 1993, introduzindo novos procedimentos, modalidades e exigências (BRASIL, 2021). A transição normativa gerou um cenário de adaptação, agravado pelo volume contínuo de normas complementares, instruções normativas e acórdãos do Tribunal de Contas da União (TCU) que demandam atualização constante por parte de fornecedores.

Do ponto de vista operacional, o monitoramento de editais publicados no Portal Nacional de Contratações Públicas (PNCP) exige dedicação considerável de tempo e recursos humanos. A análise de cada edital envolve leitura de documentos extensos, identificação de requisitos técnicos e habilitatórios, verificação de prazos e interpretação de cláusulas redigidas em linguagem jurídica densa, o que frequentemente inviabiliza a participação de empresas com estrutura reduzida.

A motivação para o desenvolvimento deste trabalho parte da identificação de uma lacuna entre as ferramentas disponíveis e as necessidades reais do setor. Soluções de busca tradicionais e modelos de linguagem generalistas não são suficientes para atuar como agentes especializados no domínio licitatório. Os recentes avanços em modelos de linguagem de grande escala e na técnica de Geração Aumentada por Recuperação (RAG) viabilizam a construção de sistemas capazes de compreender bases documentais jurídicas e responder perguntas em linguagem natural com rastreabilidade das fontes.

1.3 Proposta de Solução de Software

A plataforma Licita.AI constitui uma solução de software desenvolvida para reduzir as barreiras operacionais e jurídicas enfrentadas por licitantes participantes de processos licitatórios no Brasil. A solução integra quatro módulos funcionais complementares, que abrangem o ciclo de identificação, análise e acompanhamento de oportunidades em licitações.

O primeiro módulo destina-se à definição do perfil do usuário licitante, realizada mediante o preenchimento de um formulário. São informados dados como o segmento de atuação, os principais produtos ou serviços ofertados e o porte, a partir dos quais se constrói um perfil estruturado, utilizado como referência para a avaliação de compatibilidade das oportunidades identificadas.

O segundo módulo é responsável pela busca e pela análise de editais, mediante integração

com a API pública do PNCP. A pesquisa de editais é realizada com base em filtros como data, modalidade e localidade. Para um edital selecionado, o sistema aciona um modelo de linguagem que interpreta o documento, evidencia seus principais requisitos e prazos e avalia sua compatibilidade com o perfil previamente cadastrado.

O terceiro módulo disponibiliza um chat fundamentado na técnica de Geração Aumentada por Recuperação (RAG), por meio do qual o usuário formula perguntas em linguagem natural acerca de um edital específico. As respostas baseiam-se no conteúdo do próprio edital, permitindo o esclarecimento de dúvidas relativas a requisitos, condições e prazos, sem que seja necessária a leitura integral do documento.

O quarto módulo é responsável pelas notificações. O sistema seleciona os editais recém-publicados, avalia sua aderência ao perfil do licitante e envia os compatíveis por e-mail e, quando configurado, por aplicativo de mensagens (WhatsApp), dispensando a consulta manual frequente à plataforma.

A plataforma é concebida como solução *multi-tenant*, possibilitando que diferentes licitantes utilizem o sistema com isolamento completo de dados, o que a torna adequada tanto ao uso interno quanto à comercialização como serviço.

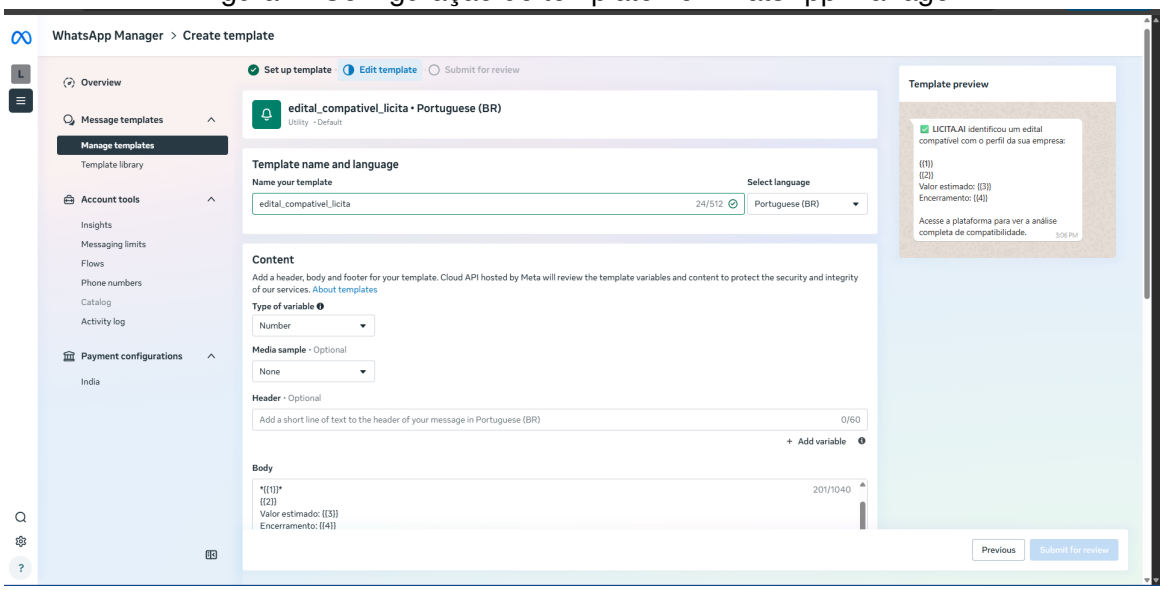
1.4 Tecnologias Adotadas

A plataforma é desenvolvida em Python 3.12 com o framework FastAPI no *backend*, selecionado por seu alto desempenho na construção de APIs RESTful. O *frontend* é construído em React 19 com TypeScript, utilizando o Vite como ferramenta de *build* para otimizar o tempo de inicialização e o *hot reloading*. A persistência de dados é gerenciada pelo PostgreSQL, que atua tanto no armazenamento relacional convencional quanto na gestão de vetores de *embeddings* para o módulo de chat, viabilizada pela extensão pgvector.

O pipeline de Geração Aumentada por Recuperação (RAG) foi implementado de forma própria. O fluxo compreende a segmentação dos editais em trechos, sua vetorização e a busca por similaridade semântica, e utiliza os modelos da OpenAI: o text-embedding-3-small para a vetorização dos textos e o GPT-4o mini para a geração das respostas finais. A segurança e o controle de acesso são implementados via autenticação por tokens JWT, com as credenciais dos usuários protegidas em banco de dados por meio de algoritmos de *hashing* seguros (bcrypt).

Referente à comunicação com os usuários, a plataforma integra dois serviços externos responsáveis pelo envio de notificações. O Resend é empregado para o disparo de mensagens transacionais por correio eletrônico, acessado por meio de sua API HTTP, abrangendo a recuperação de senha, a confirmação de alteração de credenciais e o envio de alertas-resumo de editais compatíveis com o perfil do licitante. Para a notificação por mensageria instantânea, utiliza-se a API de Nuvem do WhatsApp (Meta WhatsApp Cloud API), que permite o envio de mensagens baseadas em modelos previamente aprovados (*templates*) aos usuários que manifestaram consentimento e cadastraram número de telefone (Figura 1). Em conjunto, esses serviços sustentam o módulo de notificações proativas, que alerta os licitantes sobre novas oportunidades aderentes ao seu perfil sem exigir consulta manual à plataforma.

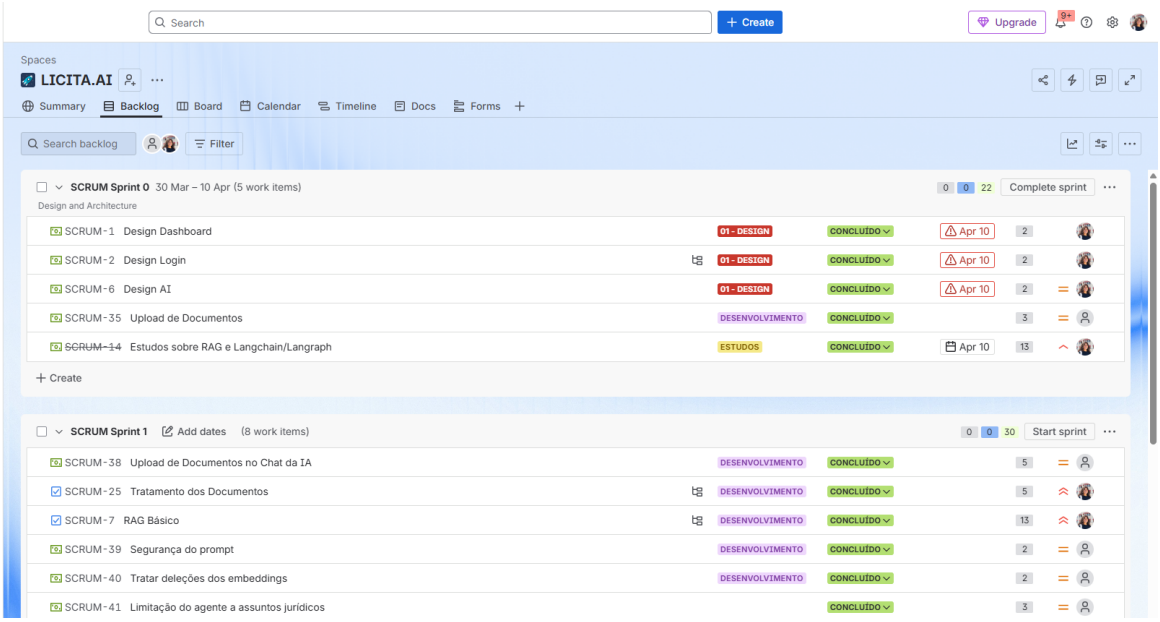
Figura 1: Configuração do template no WhatsApp Manager.



Fonte: Elaborado pela autora (2026).

A infraestrutura da aplicação é integralmente containerizada com Docker e Docker Compose, garantindo a paridade entre os ambientes de desenvolvimento e produção. O controle de versão e os pipelines de integração e entrega contínuas (CI/CD) são centralizados no GitLab, enquanto a gestão do *backlog* e o acompanhamento das tarefas do projeto foram conduzidos na ferramenta Jira, da Atlassian, na qual o trabalho foi organizado em tarefas (*issues*) distribuídas em um quadro visual, com base em metodologia ágil, o que conferiu rastreabilidade ao progresso do desenvolvimento (Figura 2). Por fim, o *deploy* é distribuído estrategicamente: o *backend* é hospedado na plataforma Fly.io e o banco de dados PostgreSQL na plataforma Neon (PostgreSQL *serverless*), ao passo que o *frontend* é implantado na Vercel, assegurando baixa latência e alta disponibilidade na entrega da interface.

Figura 2: Quadro de gerenciamento do projeto Licita.AI na ferramenta Jira.



Fonte: Elaborado pela autora (2026).

1.5 Trabalhos Relacionados

O emprego de inteligência artificial em processos licitatórios já constitui uma realidade no Brasil, com diferentes soluções em operação que combinam automação, monitoramento de editais e assistência baseada em modelos de linguagem. Entre as iniciativas institucionais, destaca-se o sistema ALICE (Análise de Licitações e Editais), desenvolvido pela Controladoria-Geral da União (CGU) em conjunto com o Tribunal de Contas da União (TCU), que analisa automaticamente editais em busca de indícios de irregularidades. Por seu propósito fiscalizatório, contudo, é restrito aos órgãos de controle, não se destinando aos fornecedores.

No âmbito acadêmico e empresarial, a startup Licitei, originada na Universidade Federal de Juiz de Fora, disponibiliza a funcionalidade “Pergunte ao Edital”, que utiliza IA generativa para responder a dúvidas sobre licitações. A plataforma comercial LicitaIA, por sua vez, oferece monitoramento de pregões, agentes de IA para análise de editais e geração automática de documentos, como impugnações e propostas. No segmento de plataformas consolidadas, o ConLicitação atua como referência em captação e gestão de editais, monitorando milhares de fontes oficiais e agregando forte componente humano de consultoria jurídica especializada. Outras soluções, como Effecti e Alerta Licitação, além de assessorias jurídicas tradicionais, também atuam nesse mercado, cada qual com distintos níveis de automação e suporte.

O Licita.AI compartilha o objetivo comum às demais soluções de simplificar a participação em licitações. Sua principal diferenciação consiste na integração de três componentes em um único ambiente:

1. Análise automatizada de compatibilidade entre o perfil institucional do licitante e os requisitos de cada edital;

2. Assistente conversacional (LicitaChat) baseado em recuperação aumentada por geração (RAG), fornecendo respostas fundamentadas diretamente no edital;

3. Arquitetura multi-tenant com isolamento completo de dados, garantindo confidencialidade entre usuários.

Esta combinação oferece um nível de integração e acessibilidade ainda pouco explorado pelas plataformas concorrentes.

2 Requisitos

Os requisitos funcionais foram modularizados logicamente em sete partes para compreensão do fluxo.

2.1 Requisitos Funcionais

Autenticação e Gestão de Conta

- **RF01** O sistema deve permitir o cadastro de usuários, distinguindo o tipo Empresa (Pessoa Jurídica) ou Pessoa Física.
- **RF02** O sistema deve autenticar o usuário por e-mail e senha, emitindo uma credencial de acesso (token JWT) para a sessão.
- **RF03** O sistema deve permitir a recuperação de senha, enviando ao e-mail do usuário um link de redefinição com validade temporária.
- **RF04** O sistema deve notificar por e-mail a confirmação de alteração de senha.
- **RF05** O sistema deve permitir o encerramento de sessão (logout).
- **RF06** O sistema deve permitir ao usuário configurar os dados de notificação, incluindo telefone e consentimento para recebimento de alertas.

Perfil do Usuário

- **RF07** O sistema deve permitir criar e editar o perfil da empresa ou pessoa, contemplando identificação, porte, segmento, produtos ou serviços, abrangência geográfica, capacidade técnica e experiência.
- **RF08** O sistema deve calcular e exibir a completude do perfil, apresentando percentual de preenchimento e itens pendentes.
- **RF09** O sistema deve adaptar os campos do perfil conforme o tipo de usuário, utilizando CNPJ e CNAE para empresas e CPF para pessoas físicas.

Busca de Editais no PNCP

- **RF10** O sistema deve permitir buscar editais no PNCP por período, UF (opcional) e

modalidade (obrigatória).

- **RF11** O sistema deve oferecer filtros rápidos por modalidade, reexecutando a busca filtrada.
- **RF12** O sistema deve paginar os resultados da busca.
- **RF13** O sistema deve permitir filtrar os resultados por palavra-chave relacionada ao objeto do edital.
- **RF14** O sistema deve permitir visualizar os detalhes de um edital em janela modal, incluindo objeto, órgão responsável, modalidade, valor estimado, localização, datas, status e número de controle.
- **RF15** O sistema deve disponibilizar hiperligação para o edital original no PNCP.

Análise de Editais por Inteligência Artificial

- **RF16** O sistema deve gerar um parecer de análise de edital, contemplando uma seção de compatibilidade com o perfil do usuário.
- **RF17** O sistema deve permitir analisar um edital em PDF enviado pelo usuário, mediante extração automática de texto.
- **RF18** O sistema deve permitir exportar o parecer gerado em formato PDF.

LicitaChat (Chat sobre Editais com RAG)

- **RF19** O sistema deve permitir adicionar um edital ao contexto do chat, a partir do PNCP ou de arquivo PDF, realizando sua ingestão (segmentação e vetorização).
- **RF20** O sistema deve responder a perguntas em linguagem natural sobre o edital, fundamentando as respostas em trechos recuperados do documento (RAG).
- **RF21** O sistema deve persistir o histórico das conversas, permitindo retomá-las.
- **RF22** O sistema deve permitir listar e remover editais do contexto do chat.
- **RF23** O sistema deve oferecer um assistente de dúvidas gerais sobre licitações, restrito a esse domínio e sem persistência de histórico.
- **RF24** O sistema deve permitir exportar a conversa em formato PDF.

Notificações de Oportunidades

- **RF25** O sistema deve registrar o consentimento (opt-in) do usuário para recebimento de alertas.
- **RF26** O sistema deve executar a triagem de editais compatíveis com o perfil do licitante.
- **RF27** O sistema deve enviar um e-mail-resumo consolidando os editais compatíveis encontrados.

- **RF28** O sistema deve enviar alertas por WhatsApp contendo os editais compatíveis encontrados aos usuários que cadastraram telefone.

Painel Inicial (Dashboard)

- **RF29** O sistema deve exibir indicadores reais, incluindo quantidade de editais indexados, editais compatíveis, completude do perfil e estado das notificações.
- **RF30** O sistema deve apresentar uma tabela de editais recentes do PNCP com estimativa de compatibilidade em relação ao perfil do usuário.
- **RF31** O sistema deve exibir o progresso do perfil e oferecer atalho para completá-lo ou editá-lo.
- **RF32** O sistema deve disponibilizar, no painel inicial, o assistente de dúvidas gerais (RF23).
- **RF33** O sistema deve oferecer ações rápidas para busca de editais, acesso ao chat, envio de PDF e configuração de notificações.

2.2 Requisitos Não-Funcionais

- **RNF01** O sistema deve ser hospedado em infraestrutura de nuvem, com ambientes segregados de homologação e produção.
- **RNF02** As consultas ao PNCP devem implementar mecanismo de timeout e tratamento de exceções para garantir a continuidade da aplicação em caso de indisponibilidade do serviço externo.
- **RNF03** A recuperação semântica de informações deve utilizar indexação vetorial por meio do pgvector e cálculo de similaridade por distância de cosseno.
- **RNF04** A interface deve ser implementada como uma aplicação web compatível com os principais navegadores modernos, em ambiente desktop.
- **RNF05** As informações referentes aos editais devem ser obtidas por meio da API oficial do Portal Nacional de Contratações Públicas (PNCP).
- **RNF06** Toda comunicação entre cliente e servidor deve utilizar o protocolo HTTPS com suporte a TLS.
- **RNF07** As senhas dos usuários devem ser armazenadas utilizando algoritmo de hash criptográfico (bcrypt), não sendo permitido o armazenamento em texto puro.
- **RNF08** A autenticação deve ser stateless e baseada em tokens JWT assinados, com prazo de expiração configurável.
- **RNF09** Credenciais, chaves de API e demais segredos da aplicação devem ser armazenados em variáveis de ambiente e não podem ser versionados no código-fonte.

- **RNF10** Todas as entidades persistidas devem possuir um identificador de tenant (tenant_id), e todas as consultas ao banco de dados devem aplicar filtragem por tenant para garantir isolamento *multi-tenant*.
- **RNF11** O tratamento de dados pessoais deve observar os princípios estabelecidos pela Lei Geral de Proteção de Dados (LGPD), incluindo consentimento explícito para notificações.
- **RNF12** A arquitetura da aplicação deve seguir separação em camadas, permitindo baixo acoplamento e alta coesão entre os módulos do sistema.
- **RNF13** O código-fonte deve seguir padrões de desenvolvimento e engenharia de software, incluindo análise estática, formatação automática e revisão de código.
- **RNF14** Os processos de build, teste e deploy devem ser automatizados por meio de pipelines de Integração Contínua e Entrega Contínua (CI/CD).
- **RNF15** O back-end deve ser conteinerizado utilizando Docker, garantindo portabilidade e reprodutibilidade entre ambientes.
- **RNF16** Falhas em serviços externos, como PNCP, provedores de IA e serviços de e-mail, não devem interromper a execução da aplicação, devendo ser tratadas por mecanismos de contingência e degradação controlada.

2.3 Matriz de Rastreabilidade

A matriz apresentada no Quadro 1 relaciona os objetivos específicos aos requisitos, aos principais casos de uso e às formas de verificação adotadas. A rastreabilidade foi construída em nível macro, agrupando os requisitos que participam do mesmo objetivo funcional.

Tabela 1: Matriz de rastreabilidade entre objetivos, requisitos, casos de uso e testes.

Objetivo	**Requisitos**	**Caso de uso relacionado**	**Verificação realizada**
Obter editais do PNCP	RF10–RF15; RNF02; RNF05	Buscar e visualizar editais	Testes manuais de filtros, paginação, visualização de detalhes, acesso à fonte original e tratamento de indisponibilidade
Estruturar o perfil do licitante	RF07–RF09; RF31	Criar e editar perfil	Testes automatizados de completude e máscaras, além de testes manuais dos fluxos de cadastro e edição
Analisar a compatibilidade por IA	RF16–RF18	Analisar edital	Testes automatizados da montagem do *prompt* e validação manual e exploratória dos pareceres gerados
Responder com base no edital	RF19–RF24; RNF03	Ingerir edital e utilizar o LicitaChat	Testes automatizados de segmentação e testes manuais de ingestão, recuperação de trechos e geração de respostas
Notificar oportunidades aderentes	RF25–RF28	Configurar e executar notificações	Testes da heurística de triagem e verificação manual dos envios nos provedores configurados
Isolar os dados dos usuários	RNF10; RNF11	Autenticar e acessar recursos do *tenant*	Testes automatizados de autenticação (hash de senha e tokens JWT)

Fonte: Elaborado pela autora (2026).

3 Design

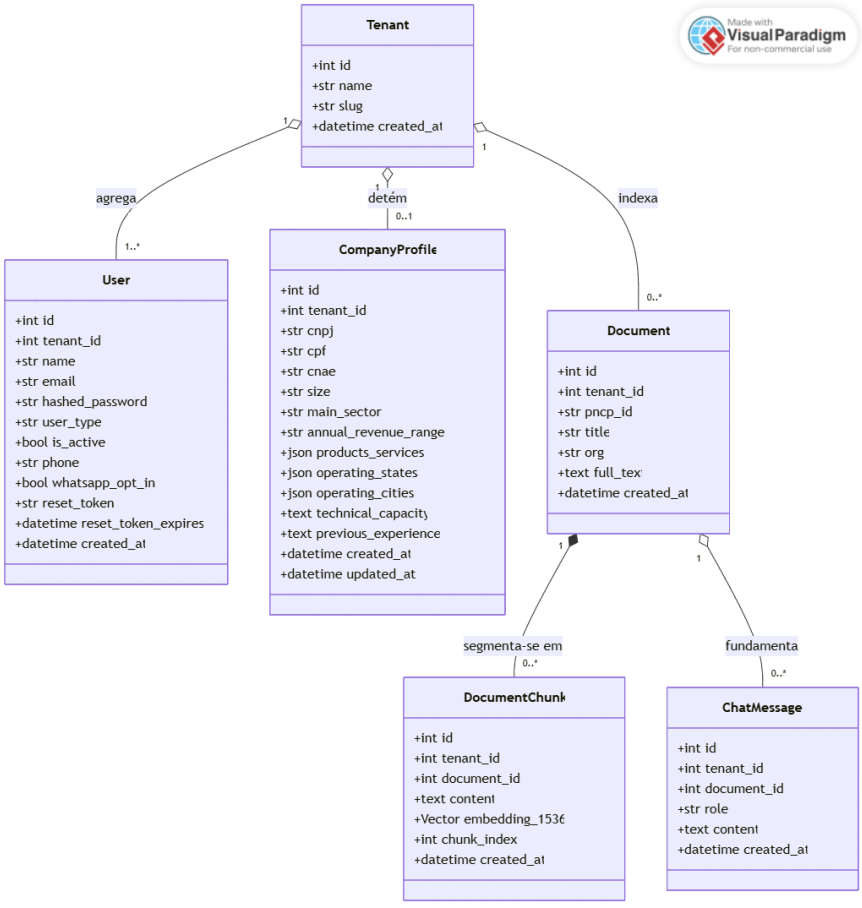
3.1 Projeto UML

A modelagem do sistema foi documentada por meio de diagramas da *Unified Modeling Language* (UML), que descrevem tanto a estrutura estática quanto o comportamento dinâmico da solução. As subseções a seguir apresentam os diagramas elaborados, acompanhados de sua respectiva descrição.

3.1.1 Diagrama de Classe

A Figura 3 apresenta as seis classes que compõem o modelo de dados da plataforma. A classe *Tenant* representa a organização ou o licitante e constitui a unidade de isolamento do sistema. A ela vinculam-se a classe *User*, que armazena as contas de acesso, e a classe *CompanyProfile*, que reúne as informações comerciais utilizadas na avaliação de compatibilidade. As classes *Document*, *DocumentChunk* e *ChatMessage* sustentam o mecanismo de Geração Aumentada por Recuperação: *Document* corresponde ao edital persistido; *DocumentChunk* armazena os trechos do edital acompanhados de seu vetor de *embedding*, do tipo *vector*, provido pela extensão pgvector; e *ChatMessage* registra o histórico das conversas.

Figura 3: Diagrama de Classe.



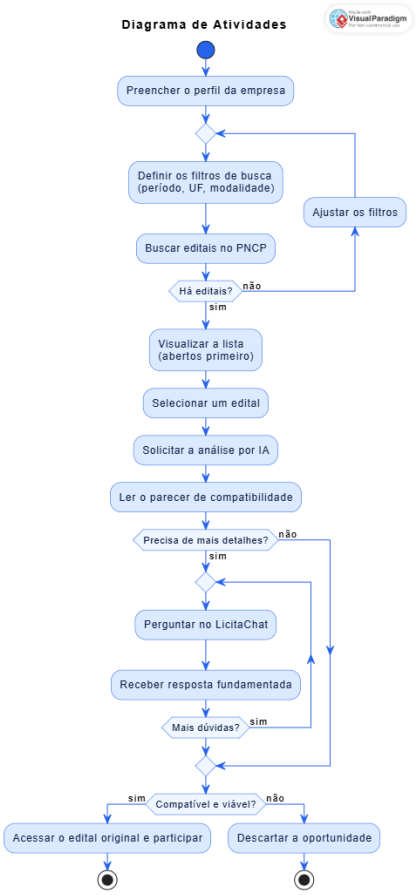
Fonte: Elaborado pela autora (2026).

Quanto aos relacionamentos, uma organização agrega zero ou mais usuários e possui, no máximo, um perfil, em razão da restrição de unicidade sobre o atributo tenant_id. Cada edital é segmentado em diversos trechos, em uma relação de composição com exclusão em cascata, de modo que a remoção do edital elimina automaticamente os trechos associados. Destaca-se, ainda, a presença do atributo tenant_id nas entidades subordinadas, recurso que assegura o isolamento dos dados entre os diferentes licitantes em todas as consultas ao banco de dados.

3.1.2 Diagrama de Atividades da principal atividade do sistema

O diagrama de atividades descreve o fluxo da principal atividade do sistema, a identificação e a análise de uma oportunidade de licitação. A Figura 4 ilustra o percurso do usuário desde o preenchimento do perfil e a busca de editais, passando pela análise por Inteligência Artificial e pelo esclarecimento de dúvidas no LicitaChat, até a decisão de participar ou descartar a oportunidade. As estruturas de decisão representam os pontos de ramificação do fluxo, enquanto o laço de repetição expressa o refinamento iterativo dos filtros de busca.

Figura 4: Diagrama de Atividades.

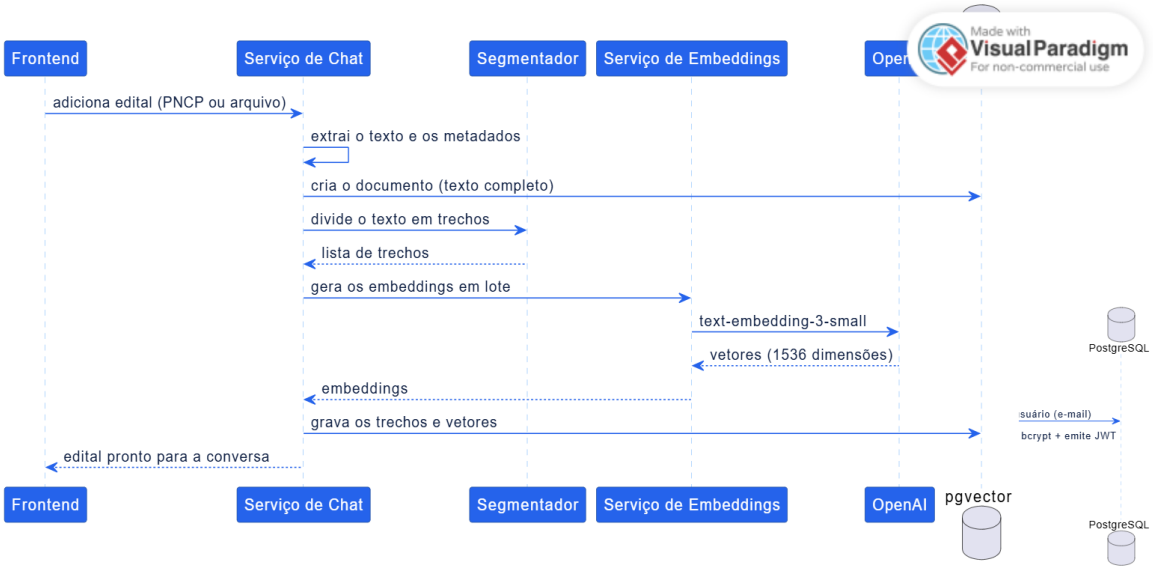


Fonte: Elaborado pela autora (2026).

3.1.3 Diagrama de Sequência (ingestão de edital para o RAG)

Este diagrama de sequência detalha a interação entre os componentes durante a ingestão de um edital no mecanismo de Geração Aumentada por Recuperação. A Figura 5 apresenta a ordem das mensagens trocadas: a extração do texto, sua segmentação em trechos, a geração dos vetores de *embedding* pela OpenAI e o armazenamento no pgvector, ao final do qual o edital fica disponível para consulta.

Figura 5: Diagrama de Sequência (ingestão de edital para o RAG).

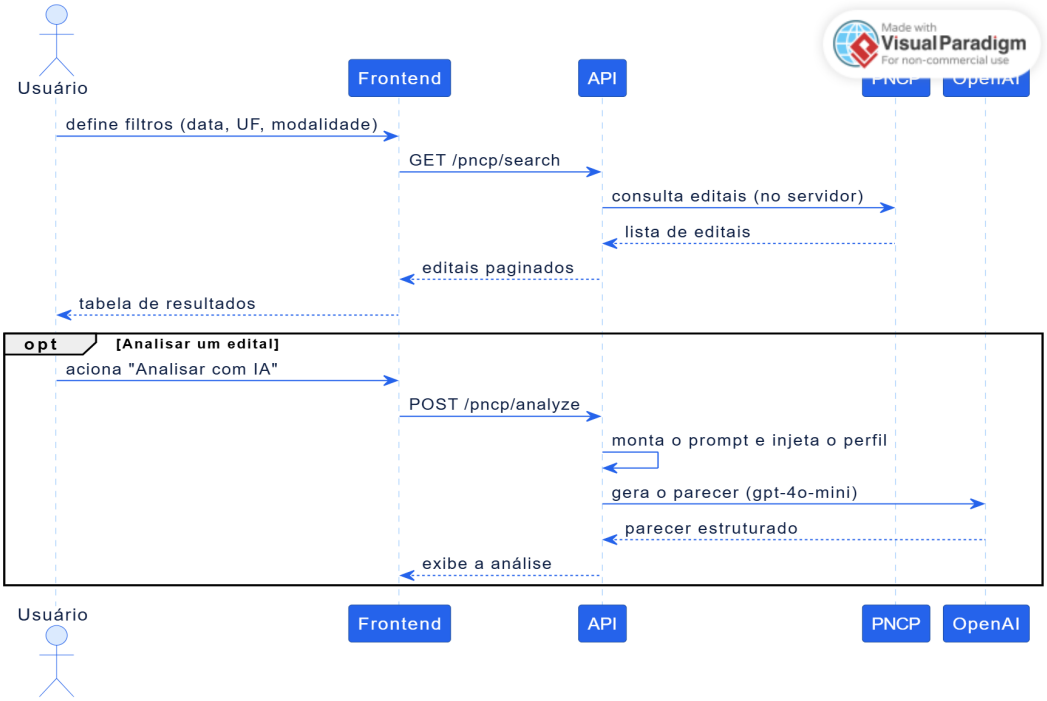


Fonte: Elaborado pela autora (2026).

3.1.4 Diagrama de Sequência (busca e análise de edital)

Este diagrama de sequência representa o fluxo de busca de editais no PNCP e a posterior análise por Inteligência Artificial. A Figura 6 evidencia que a consulta ao PNCP é intermediada pelo servidor e que, ao ser solicitada a análise, o sistema constrói o *prompt* com o perfil do licitante e aciona o modelo de linguagem, que devolve um parecer estruturado.

Figura 6: Diagrama de Sequência (busca e análise de edital).

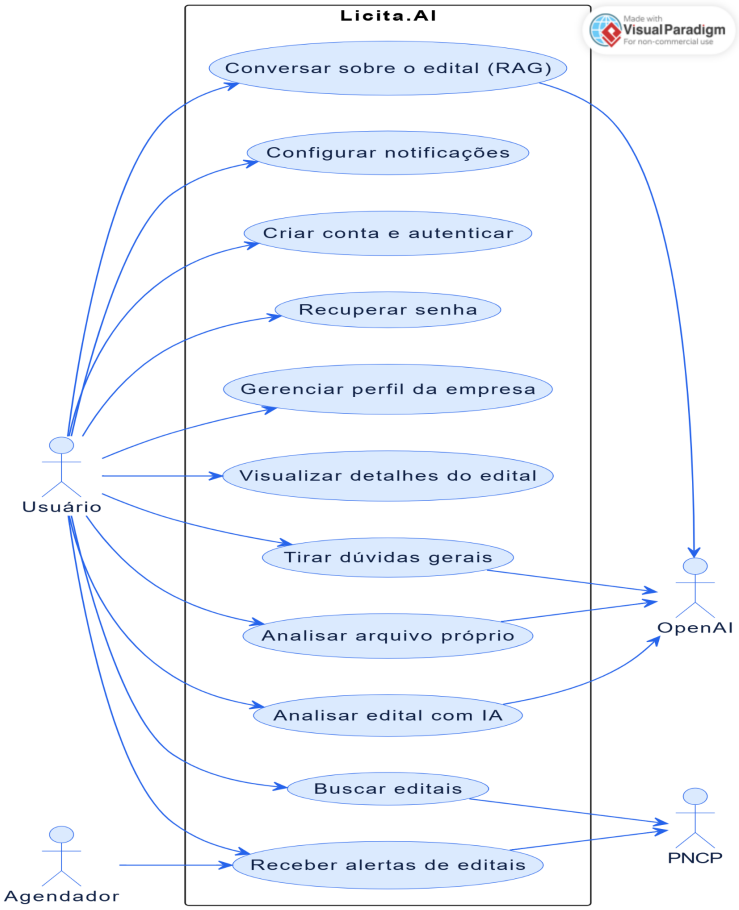


Fonte: Elaborado pela autora (2026).

3.1.5 Diagrama de Caso de Uso

O diagrama de casos de uso sintetiza as funcionalidades do sistema sob a perspectiva dos atores. A Figura 7 apresenta o usuário licitante como ator principal e suas interações com os casos de uso, além do agendador, responsável por disparar as notificações, e dos sistemas externos PNCP e OpenAI, que participam, respectivamente, da busca e da análise.

Figura 7: Diagrama de Caso de Uso.



Fonte: Elaborado pela autora (2026).

3.1.6 Diagrama de Componentes

O diagrama de componentes representa a organização interna da aplicação e as dependências entre as suas partes. A Figura 8 evidencia a arquitetura em camadas adotada no servidor, na qual cada componente possui responsabilidade bem delimitada. No cliente, a aplicação de página única (SPA) consome a interface de programação por meio de requisições REST. No servidor, a camada de apresentação (routers) recebe as requisições e delega o processamento à camada de negócio (services); a camada de persistência (models) abstrai o acesso ao banco por meio do mapeamento objeto-relacional; e os componentes transversais de núcleo respondem pela segurança, pela configuração e pela sessão de banco de dados. O diagrama explicita, ainda, as dependências externas: o PostgreSQL com pgvector e os serviços do PNCP, da OpenAI e dos provedores de notificação, todos acessados exclusivamente pela camada de negócio.

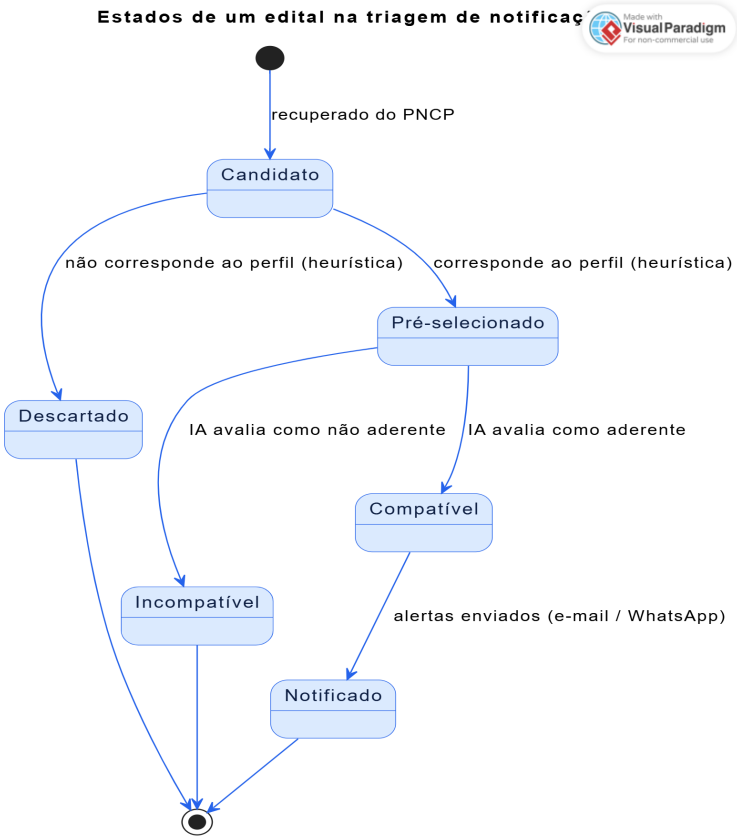
Figura 8: Diagrama de Componentes.

Fonte: Elaborado pela autora (2026).

3.1.7 Diagrama de Estados (ciclo de vida de um edital na triagem)

O diagrama de estados modela o ciclo de vida de um edital no processo de triagem de oportunidades comunicadas ao licitante. A Figura 9 descreve essa progressão. Ao ser recuperado do PNCP, o edital assume o estado *Candidato*. Em seguida, é submetido a um filtro heurístico, baseado na correspondência entre as palavras-chave do perfil e o objeto do edital: caso não haja correspondência, transita para *Descartado*; caso haja, passa para *Pré-selecionado*. Os editais pré-selecionados são então avaliados por um modelo de linguagem, que determina sua aderência ao perfil e os conduz ao estado *Compatível* ou *Incompatível*. Por fim, os editais compatíveis alcançam o estado *Notificado*, quando os alertas são enviados por e-mail e, quando aplicável, por WhatsApp. Essa modelagem corresponde diretamente ao funil de triagem do sistema, no qual a quantidade de editais se reduz a cada etapa.

Figura 9: Diagrama de Estados (ciclo de vida de um edital na triagem).



Fonte: Elaborado pela autora (2026).

3.2 Visão Arquitetural

O Licita.AI foi estruturado como um monolito modular: o *back-end* é uma única unidade de implantação, mas dividido internamente em módulos de responsabilidade bem definida, com baixo acoplamento e alta coesão. Esse monolito é organizado em camadas (*layered architecture*) e exposto sob o paradigma cliente-servidor, no qual uma interface web independente consome uma API REST.

A opção pelo monolito modular, em vez de microsserviços, decorre da escala do projeto. Microsserviços trariam complexidade operacional considerável (comunicação em rede entre serviços, consistência distribuída e monitoramento independente), custos que não se justificariam neste contexto. O monolito modular preserva os ganhos de organização e manutenibilidade, mantendo a simplicidade de um único *deploy* e de uma única base de dados transacional. Além disso, a modularização interna deixa o sistema preparado para a extração de módulos em serviços autônomos, caso a evolução futura exija. Internamente, o *back-end* segue uma arquitetura em três camadas, somada a um conjunto de componentes transversais:

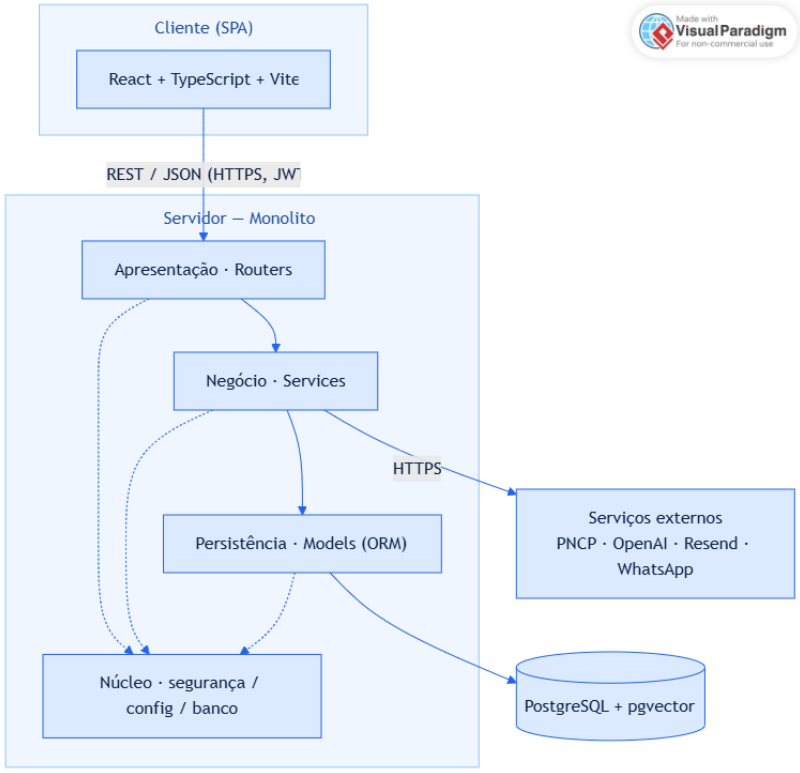
- **Apresentação (HTTP/REST):** os routers recebem as requisições, validam os dados de entrada, aplicam a autenticação e delegam a lógica às camadas inferiores;
- **Negócio (serviços):** concentram as regras do domínio, como a ingestão de editais, a

segmentação de texto, a geração de *embeddings*, a busca semântica, a interação com a IA e a triagem de notificações;

- **Persistência (modelos):** abstraem o acesso ao banco por meio do mapeamento objeto-relacional (ORM);
- **Núcleo (transversal):** segurança (autenticação e *hashing*), configuração e gerência de sessão do banco, utilizados por todas as camadas.

Essa separação garante que mudanças na forma de acesso aos dados não afetem a regra de negócio, e que a regra de negócio não dependa de detalhes do protocolo HTTP.

Figura 10: Arquitetura de Camadas.



Fonte: Elaborado pela autora (2026).

3.2.1 Tecnologias por Componente

O Quadro 2 detalha as tecnologias adotadas em cada componente da solução.

3.2.2 Integração com Serviços Externos

Uma característica central da arquitetura é tratar as integrações externas exclusivamente no lado do servidor. O cliente nunca se comunica diretamente com o PNCP, a OpenAI ou os serviços de notificação: todas as chamadas passam pelo *back-end*. Esse desenho centraliza os segredos (chaves de API), padroniza o tratamento de erros e, no caso do PNCP, contorna restrições de segurança do portal, cujo firewall de aplicação bloqueia requisições que não partam de um servidor legítimo.

Tabela 2: Detalhamento das tecnologias adotadas.

Componente	Tecnologias
Interface (cliente)	React, TypeScript, Vite, Tailwind CSS, React Router, Axios, i18n
API (servidor)	Python 3.12, FastAPI, Uvicorn, Pydantic, SQLAlchemy (ORM)
Segurança	JWT (tokens de acesso), bcrypt (hashing de senhas)
Persistência	PostgreSQL com a extensão pgvector
IA / RAG	OpenAI text-embedding-3-small e GPT-4o mini
Integrações externas	API do PNCP, Resend (e-mail), WhatsApp Cloud API
Infraestrutura	Docker, GitLab (CI/CD), Fly.io, Neon, Vercel

Fonte: Elaborado pela autora (2026).

Na versão atual, os recursos de geração e de vetorização dependem diretamente da API da OpenAI. Quando o provedor está indisponível, o sistema trata a exceção e preserva o funcionamento dos módulos que não dependem da IA, mas a análise e o LicitaChat ficam temporariamente indisponíveis. Não foram implementados, no escopo do protótipo, cache persistente de respostas, política de novas tentativas com espera exponencial, fila para reprocessamento ou troca automática de provedor.

3.2.3 Geração Aumentada por Recuperação (RAG)

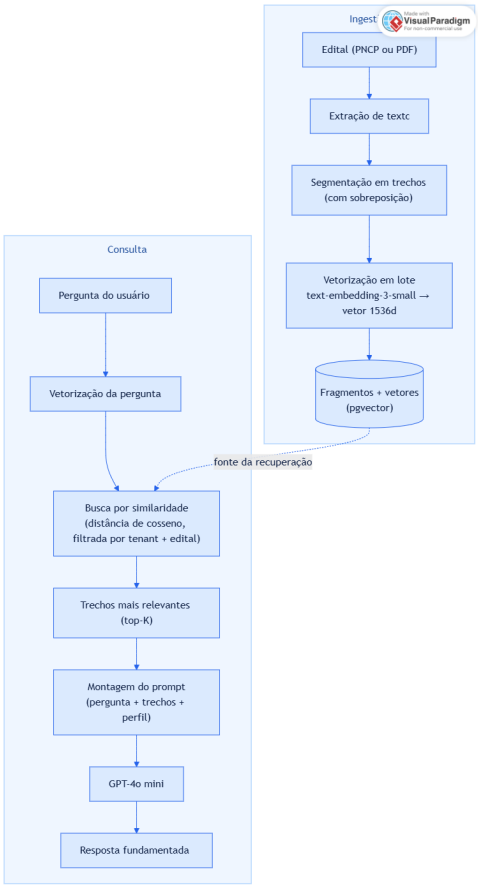
O módulo de chat fundamenta-se na técnica de Geração Aumentada por Recuperação (*Retrieval-Augmented Generation*, RAG), cujo objetivo é gerar respostas baseadas no conteúdo do próprio edital, e não apenas no conhecimento prévio do modelo de linguagem. Ao fornecer ao modelo apenas os trechos efetivamente recuperados do documento, busca-se reduzir o risco de alucinações (respostas plausíveis, porém incorretas) e aumentar a rastreabilidade das informações prestadas. O subsistema divide-se em dois fluxos: a ingestão, executada uma vez por edital, e a consulta, executada a cada pergunta.

Ingestão. Quando um edital é adicionado ao contexto, seu texto é obtido e submetido às seguintes etapas: segmentação (*chunking*), em que o texto é dividido em trechos menores com sobreposição parcial; vetorização (*embedding*), em que cada trecho é convertido em um vetor de 1.536 dimensões pelo modelo text-embedding-3-small, processado em lote; e armazenamento, em que os trechos e seus vetores são persistidos no PostgreSQL com a extensão pgvector, já associados ao respectivo tenant.

Consulta. A cada pergunta formulada pelo usuário, executa-se: a vetorização da pergunta pelo mesmo modelo de *embedding*; a busca por similaridade, em que o pgvector recupera os trechos mais próximos pela distância de cosseno, restringindo a busca ao tenant e ao edital; a montagem do contexto, em que os trechos mais relevantes são combinados com a pergunta e com o perfil do licitante, compondo o *prompt*; e a geração da resposta pelo modelo GPT-4o mini, fundamentada nos trechos recuperados, com registro da interação para permitir a retomada do diálogo.

A Figura 11 sintetiza os dois fluxos acima.

Figura 11: Pipeline de Geração Aumentada por Recuperação (RAG).



Fonte: Elaborado pela autora (2026).

Uma decisão relevante foi armazenar os dados relacionais e os vetores no mesmo sistema de banco de dados, por meio da extensão pgvector, em vez de adotar um banco vetorial dedicado. Essa escolha simplifica a operação e permite que a busca semântica e o isolamento entre os licitantes ocorram sob a mesma transação. De forma análoga, optou-se por uma implementação própria da segmentação de texto, sem o uso de bibliotecas de orquestração de maior porte, em razão da restrição de memória do ambiente de implantação. Os modelos text-embedding-3-small e GPT-4o mini foram selecionados também por sua adequação econômica ao protótipo. A Figura 12 apresenta o consumo observado durante parte do desenvolvimento, mas não constitui uma projeção de custo em escala.

Figura 12: Painel de consumo dos modelos da OpenAI durante parte do desenvolvimento.

Fonte: Elaborado pela autora (2026).

3.3 Modelo de Banco de Dados

A camada de persistência da plataforma foi implementada sobre o PostgreSQL, Sistema Gerenciador de Banco de Dados Relacional (SGBDR) de código aberto, provisionado na modalidade *serverless* por meio do serviço Neon. A adoção de uma infraestrutura *serverless* justifica-se pelo perfil de carga variável característico de um protótipo acadêmico, eliminando a necessidade de aprovisionamento e manutenção manual de servidores de banco de dados e adequando o custo operacional à demanda efetiva. A decisão arquitetural mais determinante desta camada consistiu em unificar, em um único SGBDR, os dados relacionais e os dados vetoriais exigidos pelo mecanismo de Geração Aumentada por Recuperação (*Retrieval-Augmented Generation*, RAG). Para tanto, habilitou-se a extensão pgvector, que incorpora ao PostgreSQL o tipo de dado *vector* e os operadores de cálculo de distância entre vetores. Uma alternativa seria empregar um banco de dados vetorial dedicado (por exemplo, Pinecone ou Weaviate); essa opção, no entanto, foi descartada por introduzir um componente adicional de infraestrutura, com custo de operação e de sincronização, sem benefício proporcional na escala do projeto. A unificação permite que dados relacionais e *embeddings* sejam manipulados na mesma transação e submetidos ao mesmo modelo de isolamento, simplificando a consistência e a segurança.

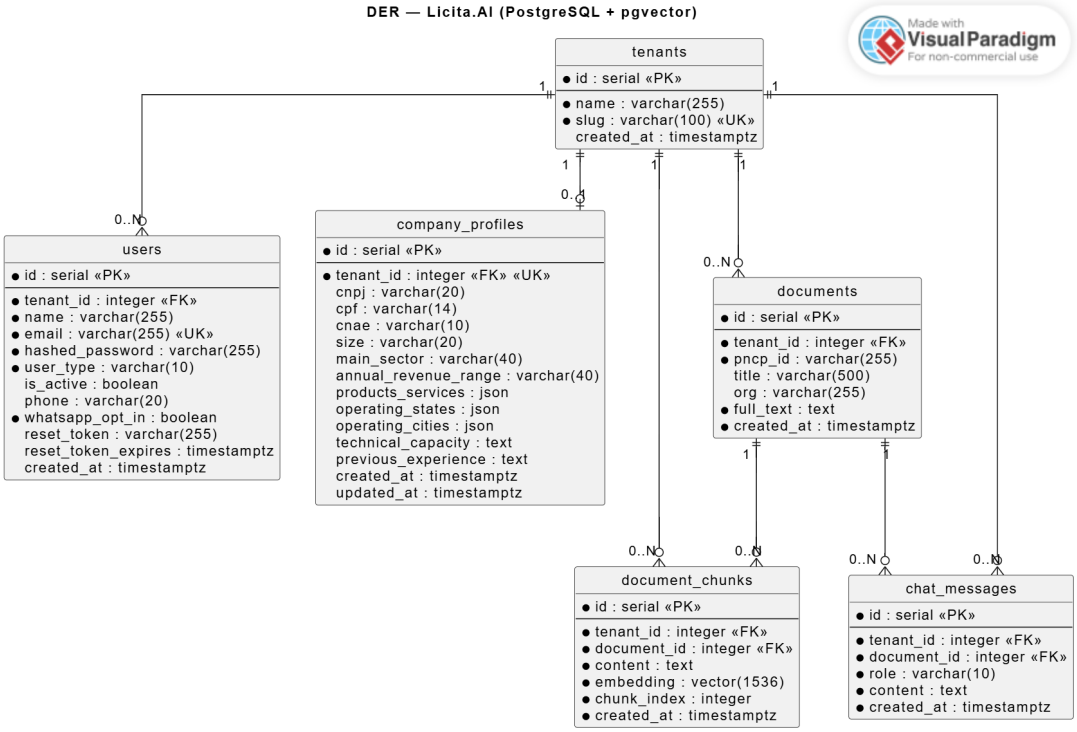
3.3.1 Princípio Estruturante: o Isolamento Multi-Tenant

O modelo de dados organiza-se em torno do princípio de isolamento *multi-tenant*, no qual cada organização (tenant) constitui uma fronteira lógica de dados. Toda entidade do sistema referencia o atributo tenant_id, e toda consulta é obrigatoriamente filtrada por ele, de modo que nenhum usuário acesse informações pertencentes a outra organização. Por decisão de privacidade, adotou-se a política de um usuário por organização: ainda que dois indivíduos pertençam à mesma empresa, seus documentos, conversas e perfis permanecem segregados, opção que prioriza a confidencialidade sobre o compartilhamento.

3.3.2 Disposição Atual do Esquema

O esquema vigente compõe-se de seis relações, organizadas em dois agrupamentos funcionais, conforme ilustra o diagrama entidade-relacionamento (Figura 13).

Figura 13: Diagrama Entidade-Relacionamento.

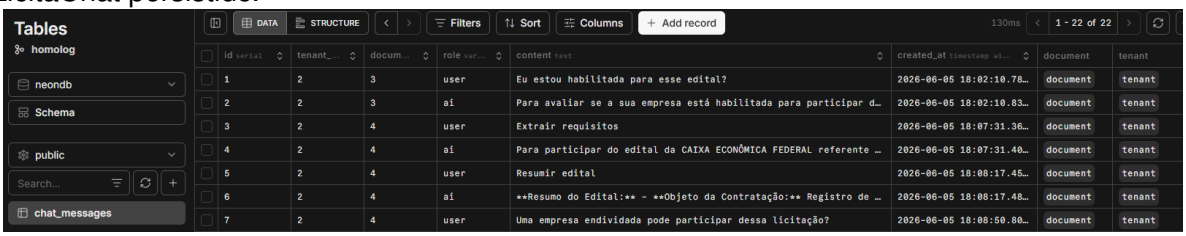


Fonte: Elaborado pela autora (2026).

(a) Cadastro e identidade. A relação *tenants* representa a unidade de isolamento, identificada por um *slug* único gerado a partir do nome acrescido de um sufixo aleatório, o que evita colisões. A relação *users* armazena as contas humanas, vinculadas a um *tenant* por chave estrangeira; nela, a senha é persistida exclusivamente sob a forma de hash criptográfico (bcrypt), nunca em texto puro, e o atributo *user_type* discrimina pessoa jurídica de pessoa física. A relação *company_profiles*, em associação de cardinalidade um-para-no-máximo-um com *tenants* (restrição UNIQUE sobre *tenant_id*), consolida a identidade comercial do licitante, reunindo os atributos posteriormente fornecidos à IA para a avaliação de compatibilidade.

(b) Núcleo do RAG. A relação *documents* persiste os editais (oriundos do PNCP ou de inserção manual), preservando o texto integral em atributo do tipo *text*. A relação *document_chunks* armazena os fragmentos textuais desses editais acompanhados de seu respectivo *embedding*, um vetor de 1.536 dimensões compatível com o modelo text-embedding-3-small, sobre o qual a busca semântica opera por distância de cosseno. Por fim, a relação *chat_messages* registra o histórico das conversas, vinculando cada mensagem ao documento em discussão e permitindo a retomada do diálogo entre sessões. O esquema descrito foi efetivamente implementado e populado no ambiente de homologação. A Figura 14 apresenta a relação *chat_messages* com o histórico de conversas do LicitaChat persistido, evidenciando o vínculo de cada mensagem ao respectivo documento e *tenant*.

Figura 14: Relação chat_messages no PostgreSQL (Neon), com o histórico de conversas do LicitaChat persistido.



Fonte: Elaborado pela autora (2026).

3.4 Afastamentos Deliberados da Normalização

O modelo incorpora três desnormalizações controladas, justificadas por requisitos de desempenho, segurança e simplicidade, reconhecendo-se que, em sistemas de leitura intensiva e dados semiestruturados, a normalização estrita nem sempre constitui o critério ótimo. Os riscos clássicos da desnormalização, sobretudo as anomalias de atualização, são mitigados pela natureza dos dados envolvidos: aqueles redundantes são imutáveis após a escrita (gravados uma única vez na ingestão e nunca alterados) ou de baixa criticidade e reescritos integralmente a cada edição (o perfil). Em suma, troca-se pureza relacional por desempenho de leitura, simplicidade operacional e robustez do isolamento.

3.4.1 Atributos multivalorados em colunas JSON (afastamento da 1ª Forma Normal)

Os campos *products_services*, *operating_states* e *operating_cities* armazenam listas em colunas do tipo *json*. Tais dados são lidos integralmente para compor o contexto submetido à IA, jamais consultados elemento a elemento, o que torna desnecessária a decomposição em tabelas associativas e dispensa junções.

3.4.2 Redundância de tenant_id (dependência transitiva, afastamento da 3ª Forma Normal)

As relações *document_chunks* e *chat_messages* mantêm *tenant_id* próprio, ainda que este seja derivável por meio de *document_id*. A duplicação é proposital: confere desempenho às consultas vetoriais (que filtram o *tenant* sem necessidade de junção com *documents*) e segurança ao isolamento (o predicado de filtragem torna-se explícito e local, reduzindo o risco de vazamento por junção omitida), em um padrão consagrado em arquiteturas multi-tenant.

3.4.3 Duplicação do texto integral em relação aos fragmentos (dado derivado)

O texto do edital coexiste em *documents.full_text* e, fracionado, em *document_chunks.content*. A retenção do texto íntegro habilita o refracionamento com estratégias distintas sem nova obtenção do documento original, além de assegurar rastreabilidade.

4 Testes de Software

4.1 Projeto de Testes

A estratégia de testes orientou-se pelo modelo da pirâmide de testes, no qual uma base de testes unitários automatizados sobre a lógica determinística do sistema é complementada por verificação estática de código e por testes manuais e exploratórios dos fluxos integrados. A automação foi incorporada à esteira de integração contínua, mantida no GitLab, de modo que a suíte é executada a cada alteração submetida ao repositório, prevenindo regressões antes da promoção do código. Como parte expressiva do sistema depende de serviços externos não determinísticos (a API do PNCP, os modelos de linguagem da OpenAI e os provedores de notificação), adotou-se o critério de concentrar os testes unitários na lógica de negócio determinística, ou seja, aquela cujo resultado é previsível e independe de chamadas de rede.

Esses testes foram implementados com o framework pytest, no backend, e com o Vitest, no frontend. As unidades verificadas dessa forma incluem, no backend, as rotinas de segurança (geração e verificação de hash de senha e de tokens de autenticação), a segmentação de texto do pipeline de RAG (tamanho dos trechos, sobreposição e casos-limite), a montagem dos prompts enviados ao modelo de linguagem e as funções de triagem de editais (formatação de valores, heurística de compatibilidade e extração de palavras-chave); e, no frontend, as funções puras de utilidade, como as máscaras de documento e o cálculo de completude do perfil.

- **Código acoplado a dependências externas:** Para os componentes que dependem de serviços externos, em vez de simular integrações completas, verificou-se o comportamento defensivo, por exemplo, o retorno seguro dos serviços de notificação quando inexistem credenciais configuradas, garantindo que falhas externas não comprometam a estabilidade da aplicação.

- **Verificação estática e segurança:** Complementarmente aos testes dinâmicos, a esteira executa análise estática, compreendendo a padronização de estilo com o ESLint, a verificação de tipos com o TypeScript e a exigência de que o build de produção seja bem-sucedido como condição de promoção. Incorporam-se, ainda, as análises automatizadas de SAST e de detecção de segredos, alinhadas a práticas de DevSecOps.

- **Testes manuais e exploratórios:** Os fluxos ponta a ponta, como cadastro e autenticação, recuperação de senha, busca de editais, análise por IA, LicitaChat e disparo de notificações, são validados de forma manual e exploratória no ambiente de homologação, por dependerem de serviços externos cuja resposta não é determinística.

5 Implantação

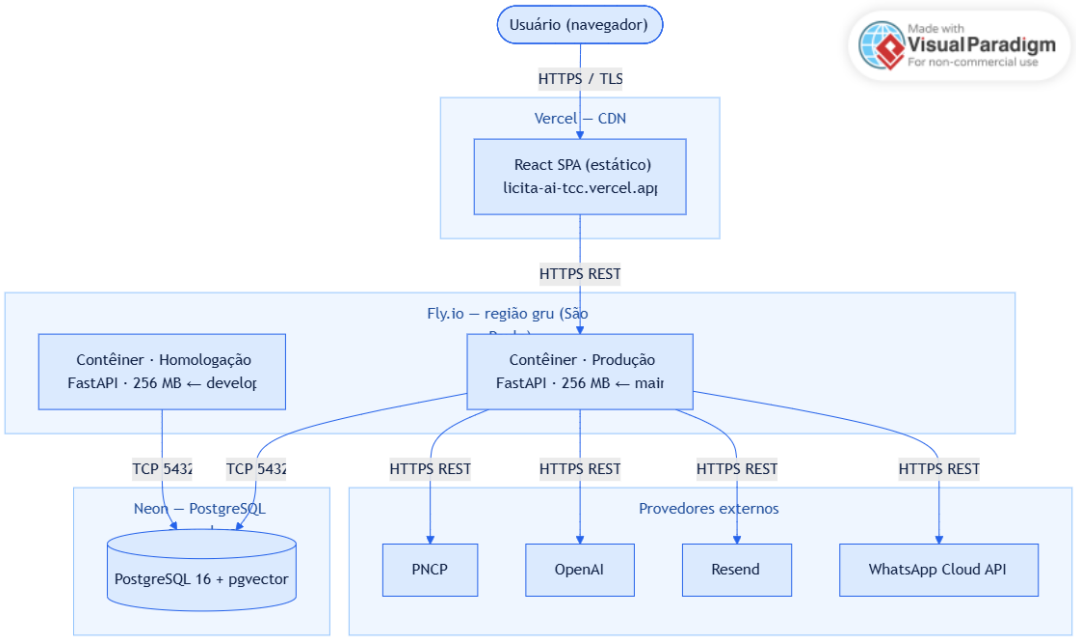
A implantação do Licita.AI foi concebida segundo o modelo de integração e entrega contínuas (CI/CD), no qual cada alteração enviada ao repositório aciona automaticamente uma sequência padronizada de verificação e publicação. O objetivo é garantir que apenas código validado quanto a estilo, testes e compilação chegue aos ambientes de execução, reduzindo a intervenção

manual e os erros dela decorrentes.

5.1 Arquitetura de Implantação

Embora logicamente monolítico, o sistema é fisicamente distribuído entre provedores de nuvem especializados, conforme a natureza de cada componente: a interface estática é servida pela CDN da Vercel; a API, conteinerizada com Docker, executa na Fly.io; e o banco de dados reside na Neon, em modalidade *serverless*. A Figura 15 apresenta essa topologia. Foram definidos dois ambientes de execução, isolados entre si. O de homologação recebe publicações automáticas a cada alteração na ramificação develop e foi, na prática, o ambiente utilizado durante todo o desenvolvimento, em razão da necessidade de consumir a API do PNCP a partir de um endereço de datacenter. O de produção recebe publicações manuais a partir da ramificação main, mediante acionamento explícito, o que funciona como uma barreira de controle antes da liberação ao usuário.

Figura 15: Arquitetura de implantação.



Fonte: Elaborado pela autora (2026).

5.2 Plataforma de Hardware e Software

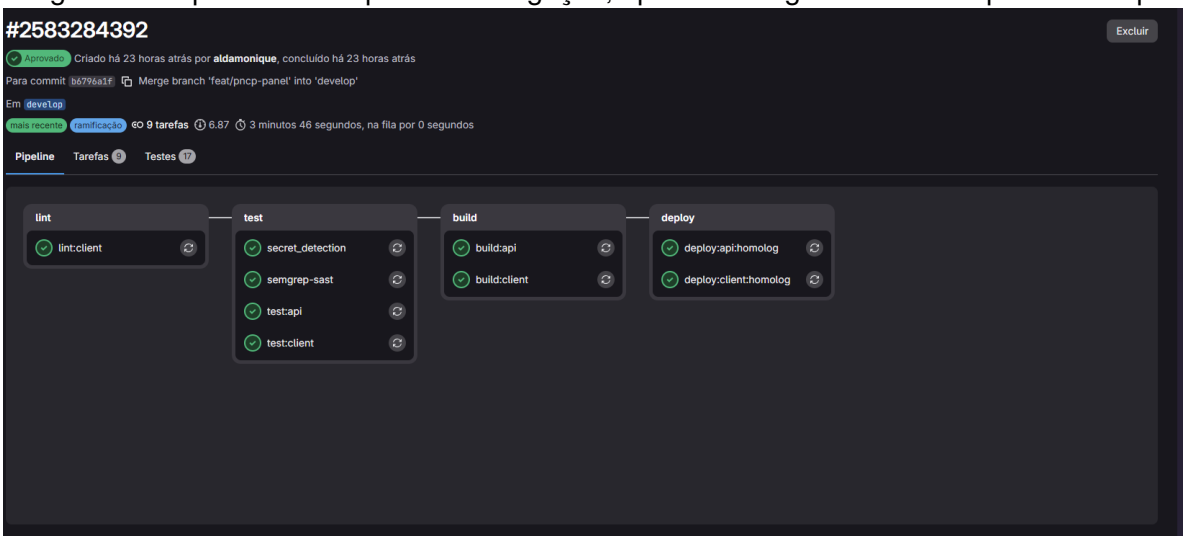
O back-end é distribuído como imagem de container Docker, executando sobre o runtime Python 3.12 com o servidor de aplicação Uvicorn. A persistência exige PostgreSQL 16 com a extensão pgvector habilitada, responsável por armazenar tanto os dados relacionais quanto os vetores de embeddings. O front-end é compilado em artefatos estáticos e servido por uma rede de distribuição de conteúdo (CDN). A operação depende, ainda, de credenciais de serviços externos fornecidas por variáveis de ambiente, a saber, a conexão com o banco, a chave da API da OpenAI, a chave de assinatura dos tokens de autenticação, as credenciais dos provedores de e-mail e de WhatsApp e o segredo do endpoint interno de notificações. Para o usuário final, o único requisito é um navegador web moderno. A natureza conteineri-

zada e serverless da solução reduz os requisitos de hardware. O contêiner do back-end opera com 256 MB de memória RAM e processador compartilhado, na região GRU (São Paulo); o banco de dados é provisionado em modalidade serverless, dispensando dimensionamento manual; e a interface, por ser estática, não impõe carga de processamento ao servidor.

5.3 Esteira de Integração e Entrega Contínuas (CI/CD)

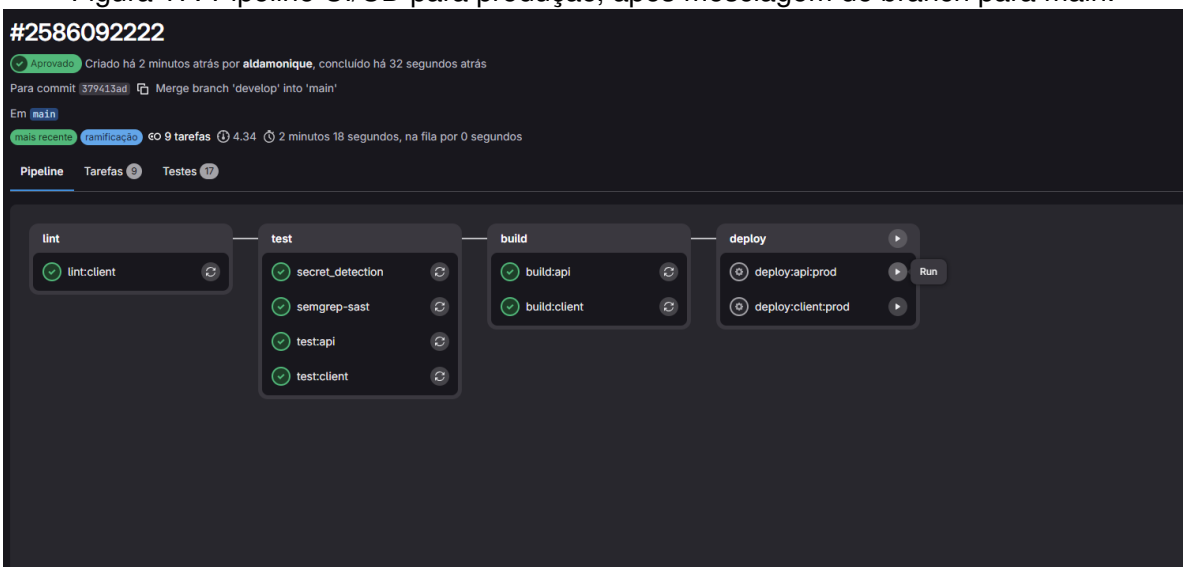
A publicação é orquestrada pelo GitLab CI/CD, organizado em quatro estágios sequenciais (lint, test, build e deploy), de modo que a falha em qualquer etapa interrompe a promoção do código. No estágio de lint, realiza-se a análise estática do front-end; no de test, executam-se os testes automatizados do front-end e do back-end; no de build, geram-se os artefatos de entrega (o pacote estático da interface e a imagem Docker do back-end, esta publicada em um registro de contêineres); e, no de deploy, os artefatos são promovidos aos ambientes. Construir a imagem no estágio de build, e não no de deploy, assegura que o mesmo artefato testado seja o efetivamente publicado (princípio do build once, deploy many). As Figuras 16 e 17 apresentam a execução da esteira nos ambientes de homologação e de produção, respectivamente, e a Figura 18 exibe o registro da publicação da API na Fly.io.

Figura 16: Pipeline CI/CD para homologação, após mesclagem de branch para develop.



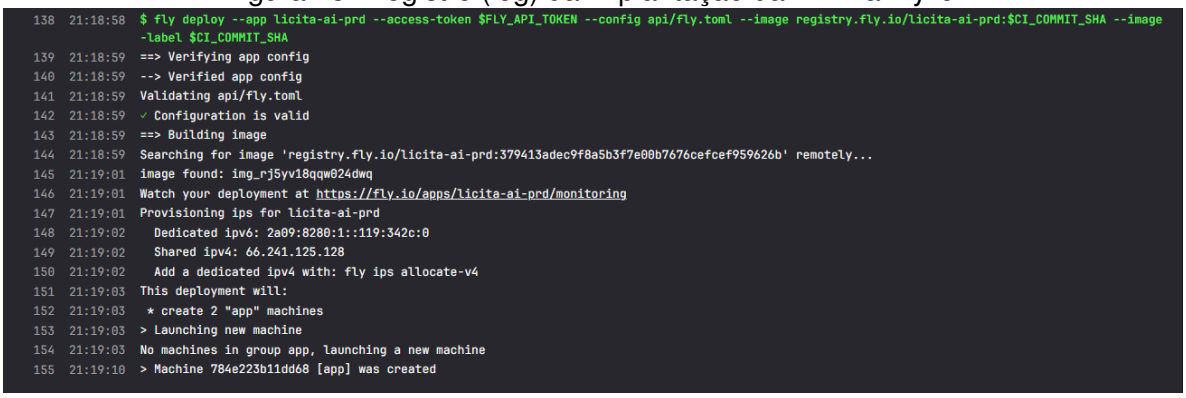
Fonte: Elaborado pela autora (2026).

Figura 17: Pipeline CI/CD para produção, após mesclagem de branch para main.



Fonte: Elaborado pela autora (2026).

Figura 18: Registro (log) da implantação da API na Fly.io.



Fonte: Elaborado pela autora (2026).

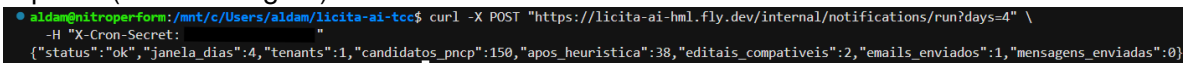
5.4 Gestão de Segredos

Em conformidade com o princípio de separação entre código e configuração, nenhum dado sensível é versionado no repositório: os segredos são injetados em tempo de execução e distribuídos em duas camadas: as variáveis consumidas pela aplicação (conexão com o banco, chaves de API e segredo de assinatura) e as variáveis usadas pela própria esteira (credenciais de publicação na Fly.io, na Vercel e no registro de contêineres). Essa separação garante que a aplicação só tenha acesso ao que lhe é pertinente e que as credenciais de publicação permaneçam restritas ao ambiente de CI/CD. A esteira incorpora, ainda, verificações automatizadas de segurança, a análise estática de vulnerabilidades (SAST) e a detecção de segredos, alinhadas a práticas de DevSecOps.

5.5 Operação do Módulo de Notificações

O processo de triagem e envio de alertas é acionado por uma requisição HTTP POST autenticada ao endpoint interno /internal/notifications/run, protegido por um segredo transmitido no cabeçalho X-Cron-Secret. Durante o desenvolvimento e a validação, o acionamento foi realizado manualmente, conforme ilustra a Figura 19, que apresenta a requisição e a respectiva resposta, o ”funil”de triagem, com o total de editais candidatos, os aprovados pelo filtro heurístico, os confirmados pela IA e as notificações enviadas.

Figura 19: Acionamento manual do módulo de notificações via requisição POST e respectiva resposta (funil de triagem).



Fonte: Elaborado pela autora (2026).

5.6 Versionamento

O controle de versão do código-fonte é feito com Git, hospedado no GitLab, que também centraliza os pipelines de CI/CD. A estratégia de ramificação é baseada em ambientes: a ramificação develop direciona-se à homologação (publicação automática) e a main à produção (publicação manual, mediante aprovação).

A numeração das versões segue o Versionamento Semântico (Semantic Versioning, SemVer), no formato MAJOR.MINOR.PATCH, em que cada componente comunica a natureza da mudança: incrementa-se o MAJOR em alterações incompatíveis com versões anteriores; o MINOR na adição de funcionalidades retrocompatíveis; e o PATCH em correções de defeitos. A aplicação encontra-se na versão 0.3.0; o número MAJOR igual a zero indica, por convenção, um sistema em fase inicial de desenvolvimento, cuja interface ainda pode evoluir entre versões MINOR. A versão é declarada uma única vez em cada projeto (nos arquivos pyproject.toml e package.json) e exposta em tempo de execução pela API, por meio de um endpoint de verificação de saúde (/health), o que permite confirmar qual versão está publicada em cada ambiente.

Além da versão semântica, voltada à leitura humana, cada imagem de contêiner do back-end recebe uma dupla etiquetagem orientada à rastreabilidade automática: o identificador do commit (SHA), imutável, que vincula com exatidão o artefato em execução ao ponto correspondente do histórico, e o nome da ramificação, etiqueta legível que aponta para a versão mais recente daquele fluxo. A etiquetagem por commit viabiliza o retorno a versões anteriores (rollback) em caso de falha, justamente por essa imutabilidade. Na publicação do back-end, a imagem construída e testada é recuperada do registro, reetiquetada e enviada ao registro do provedor de hospedagem, que então a executa, empregando-se, nesse processo, um token de acesso temporário, de curta validade, em vez da credencial permanente, como boa prática de segurança.

Em resumo, o sistema combina duas camadas de identificação de versão: a semântica (MAJOR.MINOR.PATCH), que comunica a evolução funcional, e a técnica (SHA do commit),

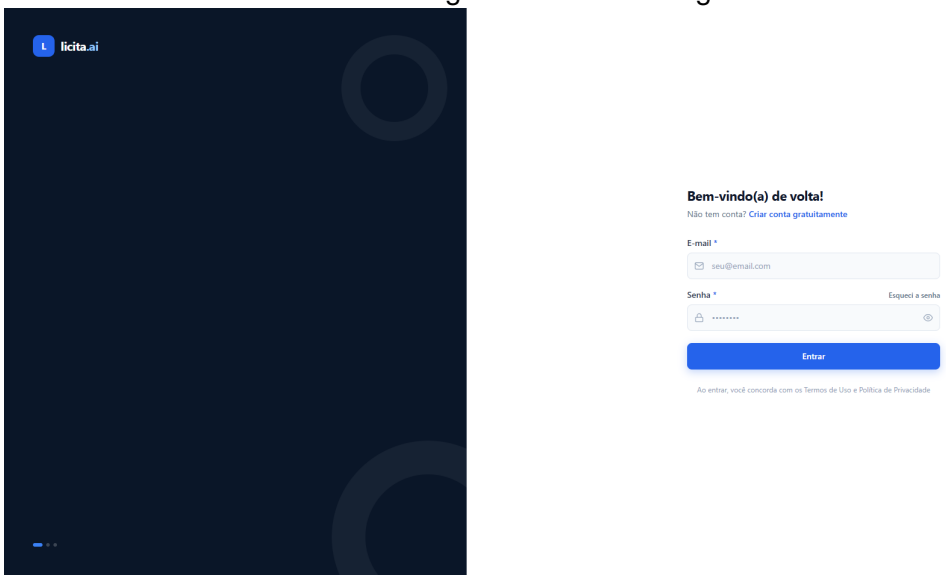
que garante rastreabilidade e reversibilidade no ciclo de implantação.

6 Manual do Usuário

6.1 Autenticação (login)

O usuário informa o e-mail e a senha previamente cadastrados. A autenticação é validada pelo servidor, que emite uma credencial temporária de acesso (token), mantida apenas durante a sessão.

Figura 20: Tela de Login.



Fonte: Elaborado pela autora (2026).

6.2 Criação de conta (registro)

Na página de Login, o usuário seleciona a opção ”Criar conta gratuitamente”. Em seguida, informa nome, organização, e-mail e senha, e seleciona o tipo de cadastro, Empresa ou Pessoa física. Concluída essa etapa, o sistema apresenta uma segunda etapa, de caráter opcional, destinada ao preenchimento do perfil. O usuário pode respondê-la imediatamente ou optar por pulá-la, completando-a posteriormente pela página de perfil. Recomenda-se, contudo, realizar esse preenchimento o quanto antes, uma vez que o perfil é determinante para a qualidade das análises de compatibilidade produzidas pela plataforma.

Figura 21: Tela de Registro da aplicação (Parte 1).

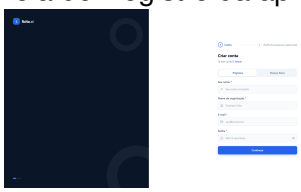


Figura 22: Tela de Cadastro do Licitante (Parte 2).

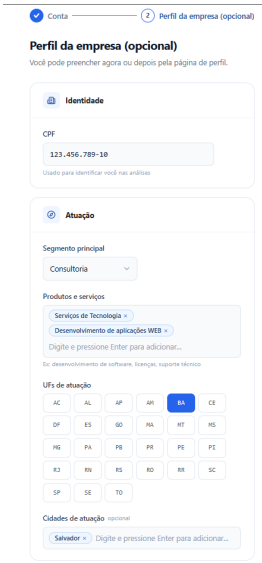


Figura 23: Tela de Cadastro do Licitante (Parte 2).

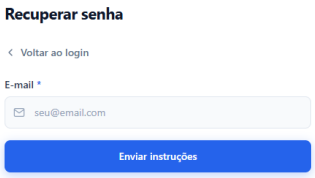
Fonte: Elaborado pela autora (2026).

6.3 Recuperação de senha

Na tela de acesso, o usuário seleciona a opção “Esqueci a senha” e informa o e-mail. O sistema envia, ao endereço informado, uma mensagem com link de redefinição, válido por tempo limitado (uma hora). Ao acessar o link, o usuário define a nova senha e a confirma. Por segurança, uma mensagem de confirmação da alteração é enviada em seguida.

Figura 24: Tela de Recuperação de senha.

Figura 25: E-mail recebido após solicitação de recuperação de senha.

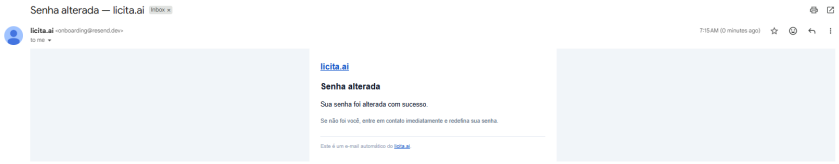


Fonte: Elaborado pela autora (2026).

Figura 26: Tela de Definição da nova senha.

Figura 27: Formulário para definição da nova senha.

Figura 28: E-mail enviado após recuperação da senha.

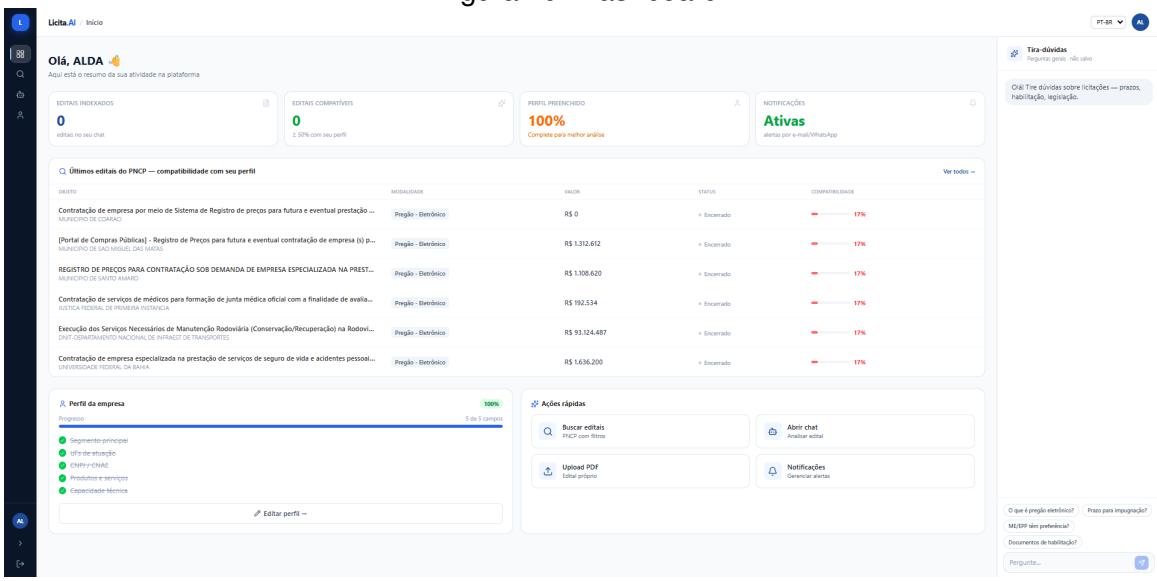


Fonte: Elaborado pela autora (2026).

6.4 Painel inicial (Dashboard)

Após a autenticação, o usuário é direcionado ao painel inicial, que apresenta uma síntese da atividade na plataforma.

Figura 29: Dashboard.



Fonte: Elaborado pela autora (2026).

O painel compõe-se dos seguintes elementos:

- **Indicadores (cartões de estatística):**
  - **Editais indexados:** quantidade de editais carregados no chat do usuário;
  - **Editais compatíveis:** número de editais recentes do PNCP com compatibilidade igual ou superior a 50% em relação ao perfil;
  - **Perfil preenchido:** percentual de completude do perfil;
  - **Notificações:** estado de consentimento para alertas.
- **Tabela "Últimos editais do PNCP, compatibilidade com o perfil":** relação de editais recentes, ordenados por grau de compatibilidade, com representação visual (barra colorida) do percentual estimado.
- **Cartão "Perfil da empresa":** barra de progresso e lista de verificação dos campos essenciais, com atalho para edição.
- **Ações rápidas:** atalhos para busca, chat, envio de PDF e configuração de notificações.
- **Assistente "Tira-dúvidas":** canal lateral de perguntas gerais sobre licitações.

6.5 Perfil da empresa

O preenchimento do perfil é determinante para a qualidade das análises, sendo recomendável realizá-lo durante o próprio cadastro.

6.5.1 Campos do perfil

Tabela 3: Campos do perfil.

Campo	Descrição
Identificação (CNPJ/CNAE ou CPF)	Documento e classificação econômica.
Porte e segmento	Classificação (MEI/ME/EPP e demais) e setor de atuação.
Produtos e serviços	Lista do que o licitante oferece.
Abrangência geográfica	Estados (UF) e municípios de atuação.
Capacidade técnica	Certificações, equipe e atestados.
Experiência prévia	Órgãos públicos já atendidos.

Fonte: Elaborado pela autora (2026).

6.5.2 Procedimento

O usuário acessa a área de Perfil, pela barra lateral, pelo cartão do painel ou pelo ícone de usuário, e preenche os campos. Nos campos do tipo lista (produtos, UFs), cada item é adicionado ao digitar o termo e pressionar a tecla Enter. Concluído o preenchimento, o usuário seleciona a opção ”Salvar perfil”. A barra de progresso indica a completude; quando todos os campos essenciais estão preenchidos, o botão de ação do painel inicial passa a oferecer a função ”Editar perfil”.

Figura 30: Barra de progresso do perfil.

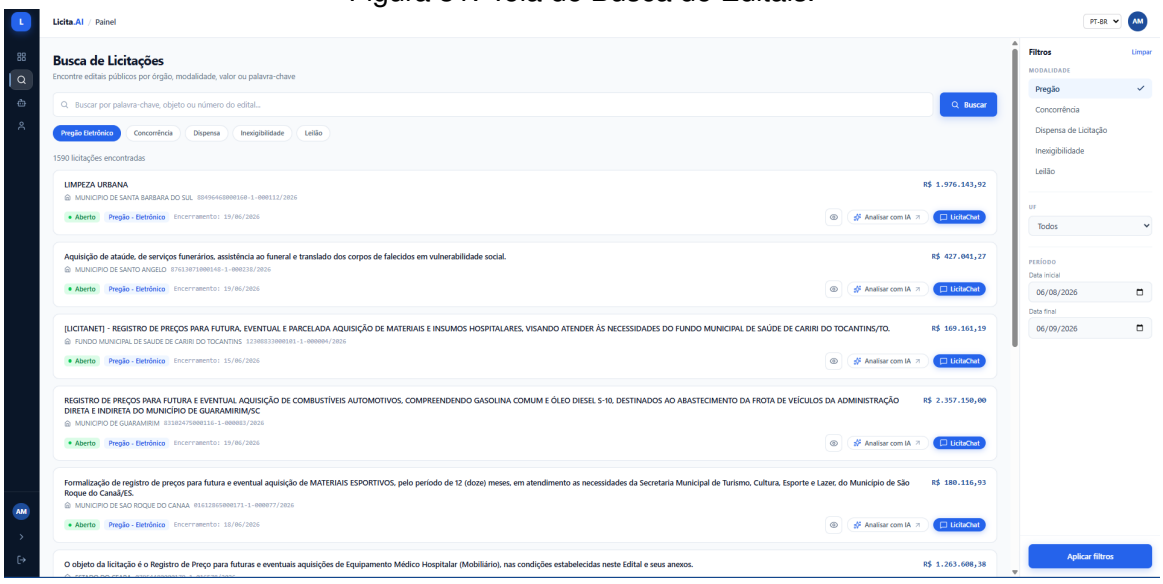
Fonte: Elaborado pela autora (2026).

6.6 Busca de editais no PNCP

6.6.1 Procedimento de consulta

O usuário acessa o Painel e define os filtros de busca: o período (data inicial e final de publicação), a UF (opcional) e a modalidade, que é um campo obrigatório, por exigência da API do PNCP. Ao acionar a opção ”Buscar”, os resultados são exibidos em lista paginada, com objeto, órgão, valor estimado e prazos.

Figura 31: Tela de Busca de Editais.

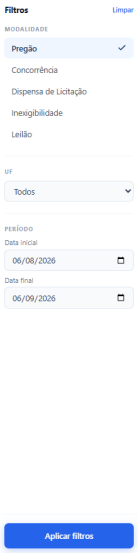


Fonte: Elaborado pela autora (2026).

6.6.2 Filtros rápidos

A interface disponibiliza atalhos por modalidade, que reexecutam a busca já restrita à modalidade selecionada, agilizando a triagem.

Figura 32: Filtros rápidos.

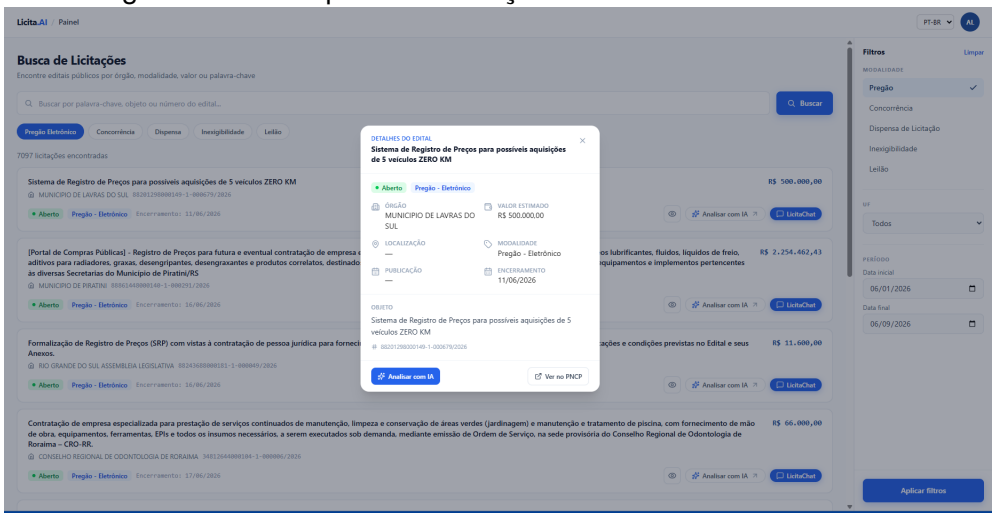


Fonte: Elaborado pela autora (2026).

6.6.3 Visualização dos detalhes de um edital

Para inspecionar um edital sem sair da lista de resultados, o usuário pode abrir uma janela modal de detalhes, acionada de duas formas: ao clicar sobre o título do edital ou sobre o ícone de visualização no respectivo cartão. A janela apresenta, de forma consolidada, os dados oficiais da contratação: objeto, órgão responsável, modalidade, valor estimado, localização (município e UF), datas de publicação e de encerramento, status e número de controle no PNCP. A partir dessa janela, o usuário pode prosseguir para a análise por IA ou acessar o edital original no PNCP, por meio de hiperligação direta à fonte oficial.

Figura 33: Modal para visualização dos detalhes de um edital.



Fonte: Elaborado pela autora (2026).

6.7 Análise de compatibilidade por IA

A análise por IA produz um parecer textual estruturado sobre um edital específico, considerando o perfil do usuário.

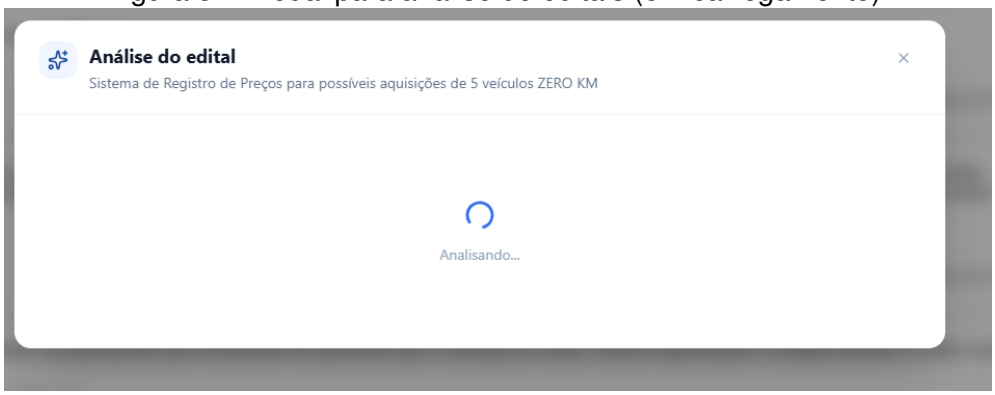
6.7.1 Uso Responsável

O parecer possui caráter informativo e de apoio à leitura, não substituindo a análise integral do edital, a conferência dos documentos oficiais ou a orientação de profissional qualificado. Modelos de linguagem podem omitir requisitos, interpretar cláusulas de forma inadequada ou produzir informações sem sustentação no documento. Antes de decidir participar de um certame, enviar uma proposta ou adotar providência jurídica, o usuário deve conferir os prazos, as exigências e as condições diretamente na fonte oficial. O Licita.AI não substitui a decisão final do licitante, tampouco garante habilitação, classificação ou êxito no processo.

6.7.2 A partir da busca

Em um resultado de busca, o usuário seleciona a opção ”Analisar com IA”e aguarda o processamento. O sistema apresenta um parecer contendo a síntese do objeto, os dados e prazos relevantes e uma seção de compatibilidade, que confronta os requisitos do edital com o perfil informado. O parecer pode ser exportado em PDF ou encaminhado ao chat para aprofundamento.

Figura 34: Modal para análise de editais (em carregamento).



Fonte: Elaborado pela autora (2026).

Figura 35: Modal para análise de editais (parte 1).

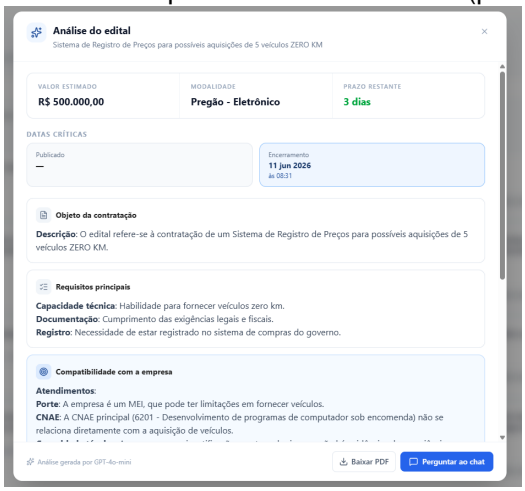
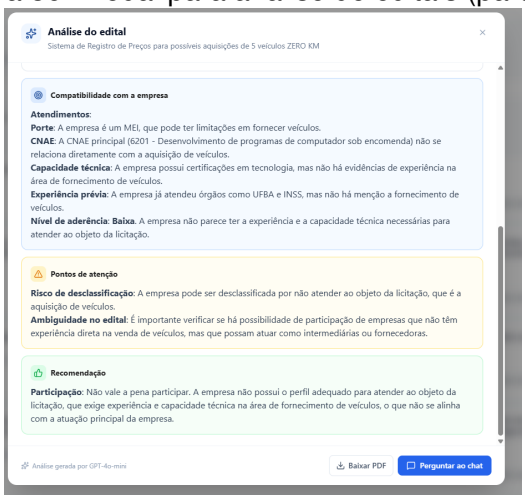
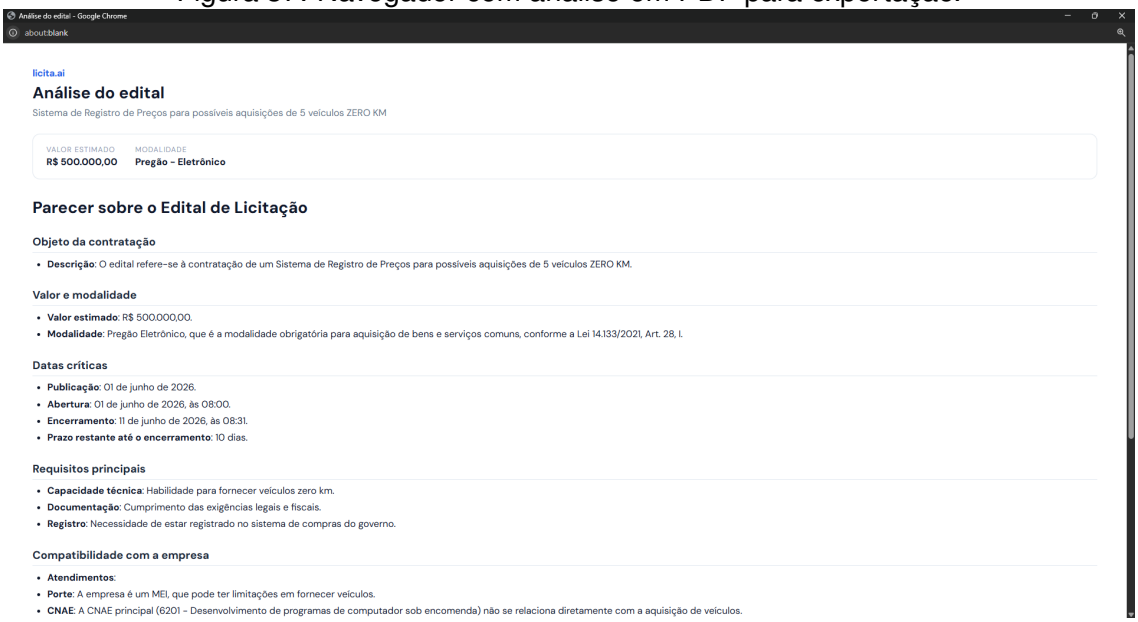


Figura 36: Modal para análise de editais (parte 2).



Fonte: Elaborado pela autora (2026).

Figura 37: Navegador com análise em PDF para exportação.



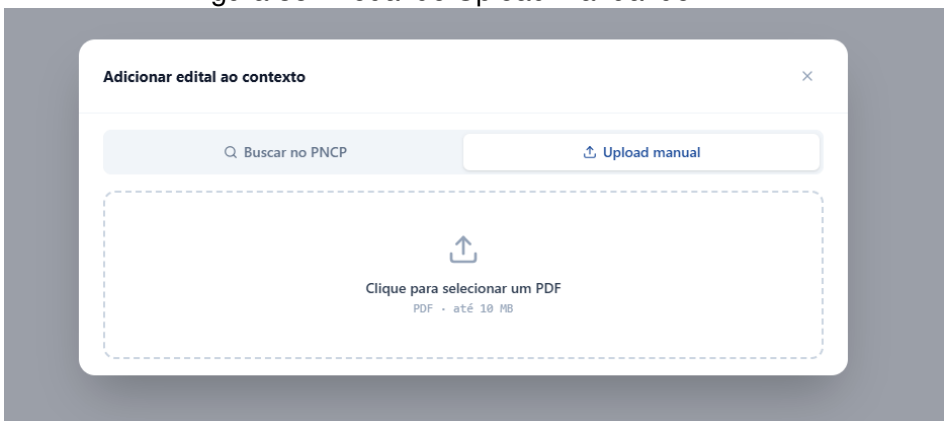
Fonte: Elaborado pela autora (2026).

6.7.3 A partir de um arquivo (PDF)

Caso o edital não esteja no PNCP, é possível enviar o PDF diretamente. O sistema extrai o texto e procede à análise. Aplicam-se restrições de formato (arquivo PDF com texto selecionável) e tamanho (10 MB).

Importante: Documentos digitalizados como imagem (PDF escaneado sem camada textual) não permitem extração automática de texto e, portanto, não podem ser analisados por esta via.

Figura 38: Modal de Upload manual de PDF.



Fonte: Elaborado pela autora (2026).

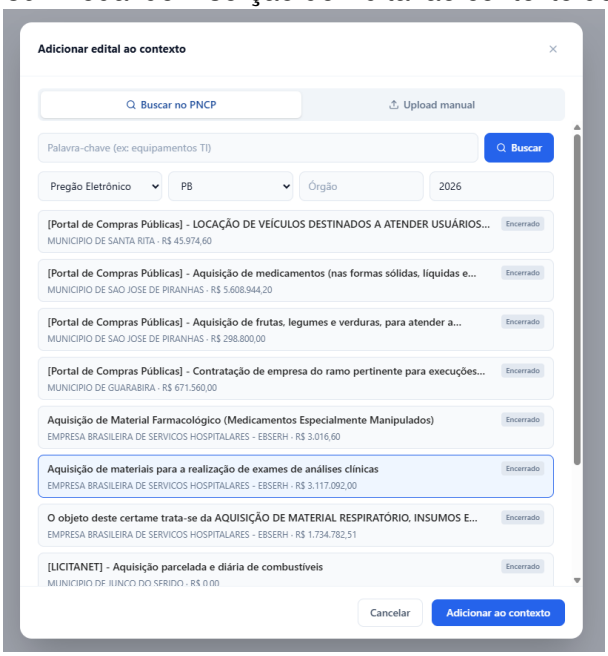
6.8 Chat sobre editais (RAG)

O módulo de chat permite dialogar com o conteúdo de um edital, obtendo respostas fundamentadas no próprio documento.

6.8.1 Adição de um edital ao contexto

O usuário acessa o LicitaChat e seleciona a opção ”Adicionar novo edital”, escolhendo a origem, busca no PNCP ou upload de PDF. Ao confirmar, o sistema realiza a ingestão do documento.

Figura 39: Modal de Inserção de Edital ao contexto do Chat.

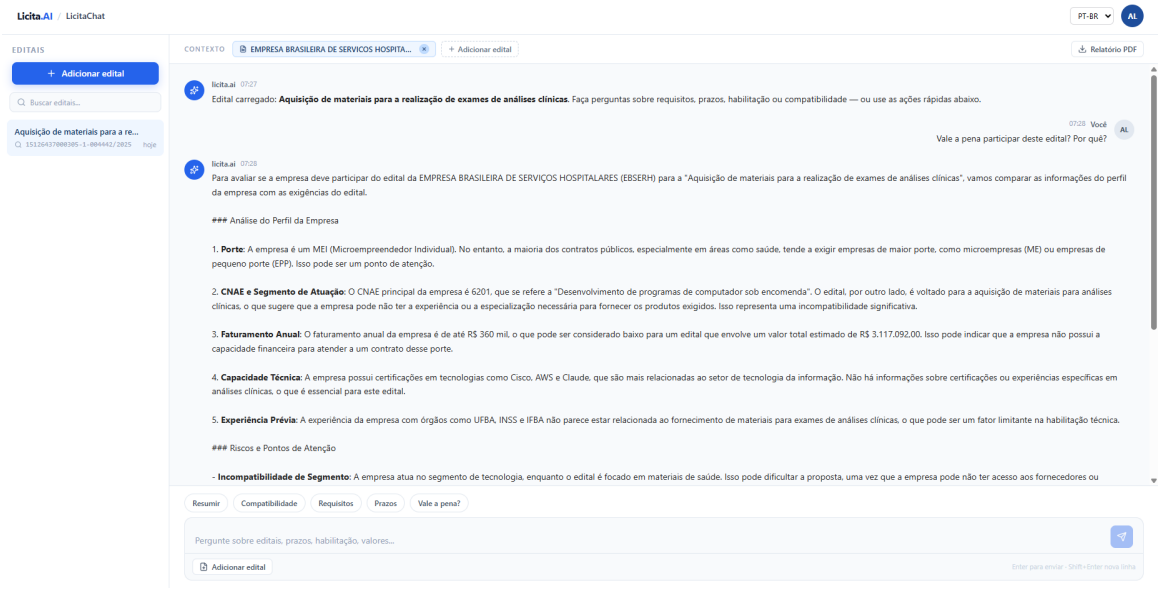


Fonte: Elaborado pela autora (2026).

6.8.2 Formulação de perguntas

Com o edital selecionado, o usuário formula perguntas em linguagem natural (por exemplo, ”Qual o prazo para impugnação?”ou ”Quais os documentos de habilitação exigidos?”). O sistema recupera os trechos mais pertinentes do edital e os utiliza para compor a resposta, preservando o histórico da conversa para consulta posterior.

Figura 40: Chat para análise de editais.

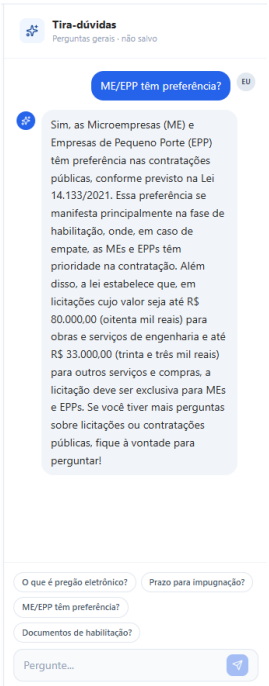


Fonte: Elaborado pela autora (2026).

6.8.3 Assistente de dúvidas gerais

No painel inicial, o assistente lateral ”Tira-dúvidas”responde a questões gerais sobre licitações e aspectos jurídicos (conceitos, prazos, prerrogativas de ME/EPP). Diferentemente do chat por edital, esse canal não preserva histórico e não se vincula a um documento específico.

Figura 41: Assistente Tira-Dúvidas.

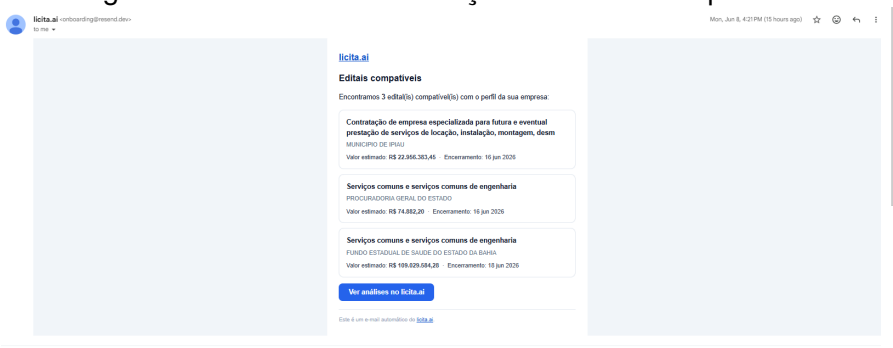


Fonte: Elaborado pela autora (2026).

6.9 Notificações de oportunidades

A plataforma pode alertar proativamente o usuário sobre editais recém-publicados compatíveis com o perfil do licitante.

Figura 42: E-mail com notificação de editais compatíveis.



Fonte: Elaborado pela autora (2026).

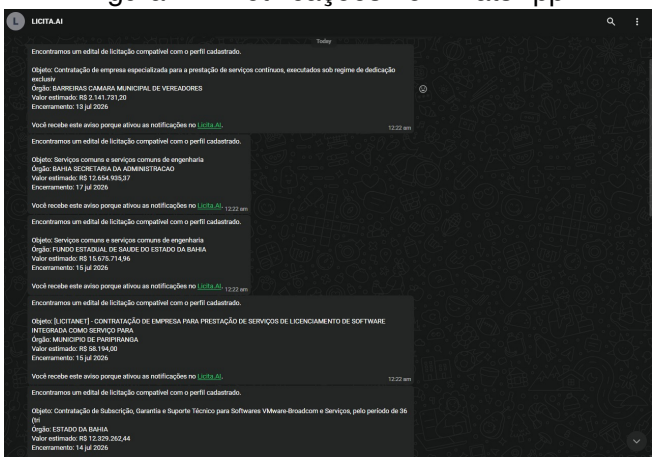
6.9.1 Ativação

A ativação dos alertas ocorre na área de Perfil, na seção de Notificações, mediante a manifestação do consentimento (opt-in) para o recebimento de notificações. Para o canal de WhatsApp, é necessário o cadastro do telefone no formato internacional, com código do país e DDD. O canal de e-mail é utilizado por padrão.

Figura 43: Seção de Notificações.

Fonte: Elaborado pela autora (2026).

Figura 44: Notificações no WhatsApp.



Fonte: Elaborado pela autora (2026).

7 Considerações Finais

O desenvolvimento do Licita.AI materializou a viabilidade técnica de aplicar Inteligência Artificial generativa como suporte à decisão no ecossistema de compras públicas. O projeto exigiu a orquestração de múltiplas disciplinas da engenharia de software, integrando uma arquitetura cliente-servidor, estratégias de persistência de dados relacionais e vetoriais e a aplicação de Processamento de Linguagem Natural (PLN) por meio do padrão Retrieval-Augmented Generation (RAG). Esta seção apresenta a capacitação complementar realizada, os impedimentos e limitações encontrados e as perspectivas de evolução do sistema.

7.1 Capacitação e Fundamentação Tecnológica

Para apoiar tecnicamente a construção do sistema, sobretudo no que se refere à aplicação de Inteligência Artificial generativa, foram realizados dois cursos complementares na plataforma Udemy, cujos certificados constam nos anexos deste trabalho.

O primeiro, ”LangChain e LangGraph: Crie Agentes de IA com LLMs e RAGs”, ministrado por Fernando Amaral, forneceu a fundamentação sobre modelos de linguagem de grande porte (LLM), agentes e a técnica de geração aumentada por recuperação (RAG).

O segundo, ”AI Engineer Production Track: Deploy LLMs & Agents at Scale”, ministrado por Ligency e Ed Donner, aprofundou a construção e a implantação de aplicações baseadas em modelos de linguagem em escala de produção. Essa formação foi determinante para as decisões relativas ao subsistema de RAG e à integração com a API da OpenAI, fornecendo a base conceitual que sustentou a implementação própria do pipeline de recuperação, posteriormente adequada às restrições do ambiente de implantação.

7.2 Impedimentos e Limitações

Durante o desenvolvimento, foram identificados alguns impedimentos técnicos que influenciaram decisões de projeto e de operação.

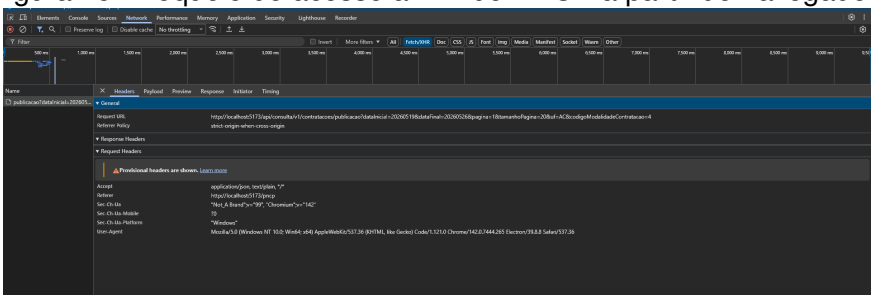
7.2.1 Limitações do PNCP

O principal impedimento envolveu as características da API do Portal Nacional de Contratações Públicas. A API é protegida por um firewall de aplicação (WAF) que bloqueia requisições originadas de ambientes de desenvolvimento local, por interpretá-las como tráfego suspeito. A Figura 45 evidencia esse comportamento: uma requisição feita a partir do ambiente local à API do PNCP permanece sem resposta, exibindo apenas cabeçalhos provisórios. A esse bloqueio somam-se a alta latência de resposta e instabilidades intermitentes do próprio portal, que chegou a retornar erros de comunicação com o seu banco de dados (Figura 46).

A solução adotada foi estruturar o sistema de modo que as consultas ao PNCP fossem realizadas exclusivamente pelo servidor, hospedado na nuvem (Fly.io), cujo endereço de datacenter é reconhecido e aceito pelo portal. Como consequência direta, o back-end precisou ser mantido em ambiente de homologação durante todo o desenvolvimento, já que a execução local impediria o acesso à busca de editais, que é a funcionalidade central do sistema.

Cabe ressaltar, ainda, que não foi implementado tratamento específico para eventuais mudanças no esquema ou no contrato da API do PNCP, como alteração de campos, modalidades ou formatos de resposta, cuja evolução futura poderia impactar diretamente a integração vigente e exigiria acompanhamento e adaptação contínuos.

Figura 45: Bloqueio do acesso à API do PNCP a partir do navegador.



Fonte: Elaborado pela autora (2026).

Figura 46: Instabilidade na API do PNCP.

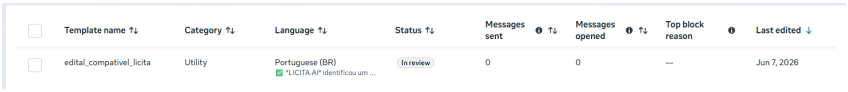
Fonte: Elaborado pela autora (2026).

7.2.2 Restrições secundárias em serviços de terceiros

Outras restrições foram observadas em serviços de terceiros e no ambiente de implantação. A limitação de memória do contêiner (256 MB) impediu o uso de bibliotecas de maior porte no *pipeline* de RAG, levando à implementação manual da segmentação de texto. O serviço de e-mail Resend, sem verificação de domínio próprio, restringe as entregas ao endereço do titular da conta. Já o WhatsApp Business exige aprovação prévia dos modelos de mensagem pela

Meta, o que limitou o uso imediato desse canal de notificação durante os testes (Figura 47). Ademais, foi necessária a aquisição de um chip telefônico vinculado a uma conta cadastrada na Meta, a fim de viabilizar a configuração e os testes da integração com a WhatsApp Cloud API. Nenhum desses fatores impediu a demonstração das funcionalidades propostas, mas delimitam o escopo de operação do protótipo e apontam para as melhorias descritas na seção de Trabalhos futuros.

Figura 47: Modelo de mensagem edital_compativel_licita em análise pela Meta (status In review).



Fonte: Elaborado pela autora (2026).

7.2.3 Validação da IA e avaliação experimental

A principal limitação científica do trabalho é a ausência de uma avaliação sistemática da qualidade das respostas produzidas pelos modelos de linguagem. O protótipo foi validado funcionalmente, mas não houve comparação com pareceres de especialistas em licitações, construção de um conjunto de respostas de referência ou medição de taxa de acerto e de alucinação. O uso de RAG, a restrição do contexto ao edital e a rastreabilidade dos trechos são mecanismos de mitigação, mas não constituem prova de correção.

Também não foi realizado um estudo quantitativo do impacto da solução. Não foi medida a redução do tempo de leitura, o aumento da identificação de oportunidades, a diferença entre análise manual e assistida por IA ou a percepção de usuários reais. Assim, o trabalho demonstra que a arquitetura e os fluxos são tecnicamente viáveis, mas não permite afirmar, com base experimental, o grau de eficiência ou de efetividade da plataforma.

7.2.4 Dependência tecnológica

A implementação atual utiliza exclusivamente modelos de inteligência artificial fornecidos pela OpenAI para geração de respostas e embeddings. Embora o tratamento de exceções impeça que falhas do provedor comprometam a integridade da aplicação, os recursos de IA permanecem indisponíveis durante períodos de indisponibilidade do serviço. Não foram implementadas estratégias de cache persistente, política de novas tentativas, fila de reprocessamento, comutação automática de provedor ou suporte a múltiplos modelos de IA.

Os dados de consumo coletados durante o desenvolvimento confirmaram a viabilidade operacional do protótipo, porém não foram formalizados em análise de custos conforme requisitos de edital, nem em projeções de consumo para volumes elevados de documentos. Consequentemente, o comportamento financeiro em cenários de escala permanece como questão em aberto.

7.2.5 Limites jurídicos e de uso

As análises geradas pelo sistema possuem exclusivamente caráter de apoio informacional e não constituem parecer jurídico. Uma interpretação incorreta dos resultados pode influenciar decisões estratégicas relativas a prazos de participação, critérios de habilitação ou elegibilidade para participação no certame. Por esse motivo, os resultados devem ser confrontados com o edital original e, quando necessário, submetidos à revisão de profissional qualificado com competência legal específica.

O protótipo não implementa validação jurídica automática, mecanismo formal de aprovação humana, trilha completa de auditoria das decisões tomadas, nem oferece garantia de ausência de erros computacionais ou de interpretação. Esses limites técnicos e processuais impedem que o usuário utilize os resultados gerados pela inteligência artificial como fundamento exclusivo e suficiente para uma decisão de participação em processo licitatório.

7.3 Trabalhos futuros

A arquitetura modular adotada favorece a evolução incremental do sistema. A prioridade metodológica é validar a qualidade do módulo de IA através de um protocolo de avaliação estruturado. Para isso, será necessário construir um conjunto de editais anotados por especialistas, que permitirá medir correção, completude, fidelidade às fontes e ocorrência de alucinações. Além disso, estudos com empresas participantes de licitações deverão comparar a análise manual e a análise assistida, mensurando tempo de processamento, utilidade percebida, identificação de oportunidades e confiança nas respostas.

Para fundamentar essas decisões, propõe-se comparar diferentes modelos de linguagem e estratégias de implantação, incluindo ofertas da OpenAI, Claude, Gemini e alternativas de código aberto executadas localmente ou em nuvem, sob os critérios de qualidade, latência, privacidade, custo e facilidade de substituição. Técnicas de ajuste fino podem ser exploradas após a consolidação de uma base validada, sem comprometer a recuperação contínua de conteúdo atualizado dos editais. Para documentos digitalizados, propõe-se incorporar OCR e mecanismos de avaliação da qualidade do texto extraído.

A escalabilidade operacional requer investimentos em infraestrutura, observabilidade e confiabilidade. Propõe-se a instrumentação de consumo e custos por etapa, testes de carga e concorrência, cache persistente de resultados, política de novas tentativas com backoff exponencial, filas de reprocessamento, limitação de taxa e suporte a múltiplos provedores. Paralelamente, devem ser implementados agendador de tarefas, migrações com Alembic e interface responsiva para completar a automação da busca, triagem e disparo de notificações.

Como expansões funcionais, sugere-se a geração automática de documentos licitatórios (propostas, declarações, impugnações e recursos) a partir do edital e perfil do licitante, acompanhada de verificação automatizada de habilitação que confronte requisitos do edital com documentos da empresa e produza checklist de aptidão. O aproveitamento do histórico de contratações do PNCP permitiria estimar faixas de preço vencedor e probabilidade de êxito, apoiando a decisão de participação. Nas integrações, seria relevante implementar comunicação completa com WhatsApp Business em conta corporativa verificada e adotar

um modelo multi-tenant colaborativo, permitindo que uma empresa validada gerencie vários usuários com controle de papéis e permissões.

7.4 Conclusão

O presente trabalho alcançou o objetivo de demonstrar que técnicas de Inteligência Artificial generativa e de busca semântica podem ser aplicadas à análise de contratações públicas de forma viável e documentada. A partir da motivação trazida pela Lei nº 14.133/2021, que ampliou a complexidade do processo licitatório, desenvolveu-se o Licita.AI, uma plataforma web que integra, em um único ambiente, a busca de editais no Portal Nacional de Contratações Públicas, a análise automatizada de compatibilidade por Inteligência Artificial, um assistente de conversação fundamentado na técnica de Geração Aumentada por Recuperação (RAG) e o envio de notificações sobre oportunidades aderentes ao perfil do licitante.

A construção da solução articulou competências de engenharia de software, desde a concepção de um monólito modular e a modelagem UML até a persistência conjunta de dados relacionais e vetoriais, a integração com serviços externos, os testes automatizados e a implantação contínua. O resultado é um protótipo funcional e documentado, com potencial para reduzir barreiras operacionais e de linguagem, sobretudo para micro e pequenas empresas e profissionais autônomos. Como esse impacto não foi medido experimentalmente, a conclusão limita-se à viabilidade técnica observada, sem atribuir percentuais de economia de tempo, precisão ou aumento de oportunidades.

A principal contribuição científico-tecnológica do trabalho reside no projeto e na implementação documentada de uma arquitetura integrada voltada ao licitante, combinando perfil institucional, análise de compatibilidade, recuperação semântica com rastreabilidade, isolamento multi-tenant e notificações. A opção pelo monólito modular demonstra ser possível manter separação de responsabilidades e capacidade de evolução sem introduzir prematuramente a complexidade operacional de microsserviços.

As limitações mais relevantes, contudo, não se restringem às dependências externas relativas à API do PNCP e a serviços de terceiros: incluem também a ausência de validação por especialistas, de métricas quantitativas, de estudo com usuários, de ensaio de custos e de testes de carga. Reconhecer esses limites evita tratar as respostas da IA como verdade estabelecida ou como parecer jurídico, e estabelece uma agenda clara de continuidade.

Dessa forma, mais do que entregar uma ferramenta isolada, o trabalho constitui uma base extensível e devidamente documentada para pesquisas e evoluções futuras, cuja adoção em decisões reais, no entanto, deve permanecer sempre acompanhada de conferência das fontes e supervisão humana.

Agradecimentos

Agradeço primeiramente a Deus, por ter me concedido serenidade, paciência e um caminho repleto de alegrias e oportunidades.

À minha mãe, à minha avó Alda e à minha tia Simone, por serem fontes inesgotáveis de amor, carinho e segurança.

À minha avó Alda, pelo primeiro livro que me deu aos 4 meses de idade, com o nobre intuito de me fazer leitora e, futuramente, estudante, além de sempre incentivar a minha educação.

À minha mãe, que dedicou todo o seu tempo e esforço à minha formação. Agradeço por me ensinar a ter gana para vencer cada um dos desafios. Sou grata pelo amor incondicional, pelos abraços, pela clareza nos momentos difíceis e pelos nossos laços de amizade.

À minha tia-mãe, Simone, que guiou meus primeiros passos em um computador, alimentando minha curiosidade, fazendo sopas, me ninando e sendo aconchego.

Ao professor Henrique Caribé, que confiou em mim para a minha primeira atuação em instrução de programação, me fazendo inspirar um carinho especial pela área.

Ao meu orientador, Grinaldo Lopes, pelo incentivo, pela motivação e pela orientação ao longo da realização deste trabalho.

Aos meus amigos, que me acompanharam durante toda essa trajetória, acolhendo, ouvindo e trocando experiências. Em especial, a Vinícius Farias e a Pedro Matos, que tantas vezes me auxiliaram na volta para casa, garantindo a minha segurança.

Ao IFBA, por ter me proporcionado um caminho repleto de oportunidades, desde o ensino técnico em Automação Industrial até o ensino superior.

Por fim, agradeço aos professores do IFBA pela dedicação, pelo incentivo ao aprendizado e por terem contribuído, de tantas maneiras, para a formação da minha base de conhecimento.

À memória do meu tio Amilton,
que guardo para sempre em meu coração
e que me inspira a ter a mesma alegria.

Referências

1 BRASIL. **Lei nº 14.133, de 1º de abril de 2021**: institui a nova Lei de Licitações e Contratos Administrativos. Diário Oficial da União, Brasília, DF, 2021. Disponível em: ⟨https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2021/lei/l14133.htm⟩. Acesso em: 17 maio 2026.

2 BRASIL. Ministério da Gestão e da Inovação em Serviços Públicos. **Ano de 2025 é marcado pela inovação no mercado de compras públicas brasileiro**. Brasília, 2026. Disponível em: ⟨https://www.gov.br/gestao/pt-br/assuntos/noticias/2026/janeiro/ano-de-2025-e-marcado-pela-inovacao-no-mercado-de-compras-publicas-brasileiro⟩. Acesso em: 17 maio 2026.

3 CONLICITAÇÃO. **Plataforma de licitações ConLicitação**. Disponível em: ⟨https://conlicitacao.com.br⟩. Acesso em: 2 maio 2026.

4 EFFECTI. **Plataforma de licitações Effecti**. Disponível em: ⟨https://effecti.com.br⟩. Acesso em: 2 maio 2026.

5 LICITAIA. **Plataforma LicitaIA**: sistema all-in-one de licitações com inteligência artificial. Disponível em: ⟨https://www.licitaia.app⟩. Acesso em: 2 maio 2026.

6 LICITEI. **Plataforma Licitei**. Universidade Federal de Juiz de Fora, Juiz de Fora, 2024. Disponível em: ⟨https://licitei.com.br⟩. Acesso em: 2 maio 2026.

7 PEREIRA, Caio César Queiroz. As inovações da Lei 14.133/2021 e seu impacto no ambiente jurídico brasileiro. **Revista FT**, v. 28, n. 134, maio 2024. Disponível em: ⟨https://revistaft.com.br/as-inovacoes-da-lei-14-133-2021-e-seu-impacto-no-ambiente-juridico-brasileiro/⟩. Acesso em: 15 maio 2026.

8 PORTAL NACIONAL DE CONTRATAÇÕES PÚBLICAS. **PNCP**. Disponível em: ⟨https://pncp.gov.br⟩. Acesso em: 2 maio 2026.

9 RIBEIRO, Cássio Garcia; INÁCIO JÚNIOR, Edmundo. **O mercado de compras governamentais brasileiro (2006-2017)**: mensuração e análise. Brasília: IPEA, 2019. (Texto para Discussão, n. 2476). Disponível em: ⟨https://repositorio.ipea.gov.br/handle/11058/9315⟩. Acesso em: 17 maio 2026.

10 TRIBUNAL DE CONTAS DA UNIÃO. **Sistema ALICE**: análise de licitações e editais. Brasília, 2018. Disponível em: ⟨https://portal.tcu.gov.br⟩. Acesso em: 2 maio 2026.

11 TRIBUNAL DE CONTAS DA UNIÃO. **Acórdãos e jurisprudência sobre licitações e contratos**. Brasília, 2024. Disponível em: ⟨https://pesquisa.apps.tcu.gov.br⟩. Acesso em: 2 maio 2026.

Anexo A – Certificados de cursos complementares

Figura 48: Certificado do curso AI Engineer Production Track: Deploy LLMs & Agents at Scale.

Fonte: Udemy (2026).

Figura 49: Certificado do curso LangChain e LangGraph: Crie Agentes de IA com LLMs e RAGs.


udemy

Certificate no: UC-9a7c4458-707a-42f4-8ce8-460820481997
Certificate url: ude.my/UC-9a7c4458-707a-42f4-8ce8-460820481997
Reference Number: 0004

CERTIFICATE OF COMPLETION

LangChain e LangGraph: Crie Agentes de IA com LLMs e RAGs

Instructors Fernando Amaral

Alda Monique Gonçalves de Alcantara Santos

Date April 7, 2026
Length 5.5 total hours

Fonte: Udemy (2026).