Documentação de Projeto

para o sistema

Cognita

Versão 6.0

Projeto de sistema elaborado pelo aluno Lucas Hemétrio Teixeira e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo dos professores Danilo de Quadros Maia Filho, Leonardo Vilela Cardoso, Raphael Ramos Dias Costa, orientação acadêmica do professor Cleiton Silva Tavares e orientação de TCC II do professor (a ser definido no próximo semestre).

14/12/2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Atividade 3	24/09/25	Introdução, Persona, Casos de Uso, Histórias de Usuário, Protótipos Iniciais	1.0
Atividade 4	13/10/25	Correções/Ajustes Atividade 3, Diagramas de Sequência do Sistema, Diagrama de Classes, Diagrama de Comunicação	2.0
Atividade 5	02/11/25	Correções/Ajustes Atividade 4, Diagrama de Arquitetura, Diagrama de Estados, Diagrama de Componentes, Diagrama de Implantação, Glossário, Property Graph Model	3.0
Atividade 6	16/11/25	Correções/Ajustes Atividade 5, Cronograma	4.0

Atividade 7	30/11/25	Correções/Ajustes Atividade 6, Casos de Testes	5.0
Atividade 8	14/12/25	Adicionar teste relacionado a IA	6.0

1. Introdução

Este documento agrega a elaboração e a revisão dos modelos de domínio e de projeto para o sistema Cognita. Ele serve como o *blueprint* técnico e funcional que guiará o desenvolvimento do *software*, detalhando sua arquitetura, comportamento e estrutura de dados.

O sistema Cognita nasce para solucionar uma lacuna fundamental no ecossistema de ferramentas de estudo digital: a fragmentação entre aplicativos de anotação de alta fidelidade, que tratam documentos como objetos isolados, e as plataformas de Gestão de Conhecimento Pessoal (PKM - *Personal Knowledge Management*), que, embora foquem na conexão de ideias, oferecem uma experiência básica de interação com os materiais-fonte.

A proposta central do Cognita é unificar esses dois mundos, criando uma plataforma coesa para o estudo ativo. O objetivo é transformar um repositório passivo de informações em uma base de conhecimento dinâmica e interconectada, onde a inovação chave reside na construção de um grafo de conhecimento de forma automática, utilizando técnicas de Processamento de Linguagem Natural (PLN) para descobrir e visualizar conexões latentes entre os conceitos estudados pelo usuário.

A referência principal para a descrição geral do problema, domínio, escopo e requisitos do sistema é o Documento de Visão, que acompanha este documento. Anexo a este documento também se encontra o Glossário, que define os termos técnicos e conceituais utilizados ao longo do projeto.

2. Modelos de Usuário e Requisitos

Esta seção é dedicada à modelagem dos usuários e dos requisitos funcionais do sistema Cognita. O objetivo é definir quem são os usuários, quais são seus objetivos e como eles interagirão com a plataforma para alcançá-los. A análise começa com a identificação e descrição dos atores do sistema, seguida pela elaboração de modelos de usuário mais detalhados, como personas, para aprofundar o entendimento de suas necessidades e contextos. Por fim, os requisitos são formalizados através de casos de uso e histórias de usuário, que descrevem as funcionalidades do sistema sob a perspectiva do usuário final.

2.1 Descrição de Atores

Nesta subseção é apresentada a descrição de cada um dos atores que interagem com o sistema. Para o sistema Cognita, foi identificado um ator principal que engloba o perfil de usuário para o qual a plataforma foi projetada.

- Ator: Estudante / Pesquisador:
    - Descrição: Este ator representa o usuário final do sistema, englobando um espectro de aprendizes sérios, como estudantes universitários (graduação e pós-graduação),

pesquisadores acadêmicos, concurseiros e autodidatas. A característica principal deste ator é a necessidade de gerenciar, analisar e sintetizar um volume significativo de material de estudo complexo, predominantemente em formato de texto (artigos em PDF, livros digitais, anotações).

- Objetivos no Sistema: O objetivo primário do Estudante / Pesquisador é transformar sua biblioteca digital de um repositório estático para uma base de conhecimento ativa e interconectada. Ele busca uma ferramenta que não apenas centralize seus materiais, mas que o auxilie ativamente na descoberta de conexões conceituais latentes, na identificação de temas centrais e na visualização da estrutura de seu próprio conhecimento.
- Interações com o Sistema: As principais interações deste ator com o Cognita incluem: realizar o upload e gerenciamento de documentos PDF; criar anotações manuscritas e digitadas diretamente sobre os documentos e em notas independentes; utilizar a busca global para encontrar informações em toda a sua base de conhecimento (incluindo texto de imagens e caligrafia); e, crucialmente, explorar o grafo de conhecimento gerado automaticamente para navegar e compreender as relações entre os diferentes tópicos de estudo.

2.2 Modelos de Usuários

Para aprofundar a compreensão sobre o ator "Estudante / Pesquisador", foi desenvolvida uma persona, detalhada na Tabela 1. A decisão de focar em uma única persona é estratégica: o arquétipo "Ana Oliveira, a Mestranda Organizada" foi escolhido por representar o ponto de maior complexidade dentro do espectro do ator definido. Ela encapsula tanto as necessidades de organização e aprendizado de um estudante quanto as de síntese e descoberta de um pesquisador. Ao projetar o sistema para atender às suas necessidades exigentes, garantimos que as funcionalidades essenciais para outros usuários no espectro (como graduandos ou concurseiros) também sejam contempladas.

	**Biografia e Comportamento** Ana Oliveira é uma mestranda dedicada à História Contemporânea. Sua rotina é intensa, dividida entre aulas, grupos de estudo e, principalmente, a pesquisa para sua dissertação. Ela lida com um volume massivo de artigos acadêmicos, livros digitalizados e anotações de fontes primárias, todos em formato PDF. Ana é tecnologicamente adepta e valoriza ferramentas que otimizam seu tempo. Ela já tentou de tudo: usa o GoodNotes em seu tablet para a leitura e anotação inicial dos PDFs, pela excelente experiência com a caneta, mas depois se vê forçada a copiar e colar manualmente seus insights para o Obsidian, onde tenta construir um mapa de suas ideias. Ela se sente constantemente frustrada por esse
Idade: 28 anos	
Ocupação: Mestranda e Pesquisadora	

	fluxo de trabalho desconectado e pela sensação de que existem conexões importantes em sua pesquisa que ela ainda não conseguiu enxergar.
Objetivos - Centralizar todos os seus materiais de pesquisa em um único lugar. - Descobrir conexões inesperadas entre autores e conceitos de diferentes artigos. - Otimizar seu tempo de revisão, podendo pesquisar instantaneamente em todas as suas anotações, incluindo as manuscritas. - Sentir que tem o controle total sobre sua base de conhecimento, e não que suas ideias estão espalhadas em arquivos e pastas.	**Frustrações** - Ter que usar dois ou três aplicativos diferentes para completar um ciclo de estudo (leitura, anotação e conexão). - Perder tempo procurando a fonte original de uma anotação ou citação que fez semanas atrás. - Suas anotações manuscritas serem "dados mortos", impossíveis de serem pesquisados globalmente. - A visão em grafo do Obsidian ser útil, mas depender 100% de seu esforço manual para criar cada link.

Tabela 1. Persona Ana Oliveira

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta subseção detalha as interações funcionais entre o ator e o sistema Cognita. Primeiramente, é apresentado um modelo de casos de uso que descreve as principais funcionalidades do sistema de forma macro, conforme ilustrado na Figura 1. Em seguida, as histórias de usuário, detalhadas na Tabela 2, traduzem essas funcionalidades em requisitos granulares e centrados nas necessidades da persona definida.

2.3.1 Diagrama de Caso de Uso

O diagrama de casos de uso para o sistema Cognita, apresentado na Figura 1, é centrado no único ator definido, o Estudante / Pesquisador. Este ator interage com um conjunto de funcionalidades principais que representam o núcleo da proposta de valor da plataforma.

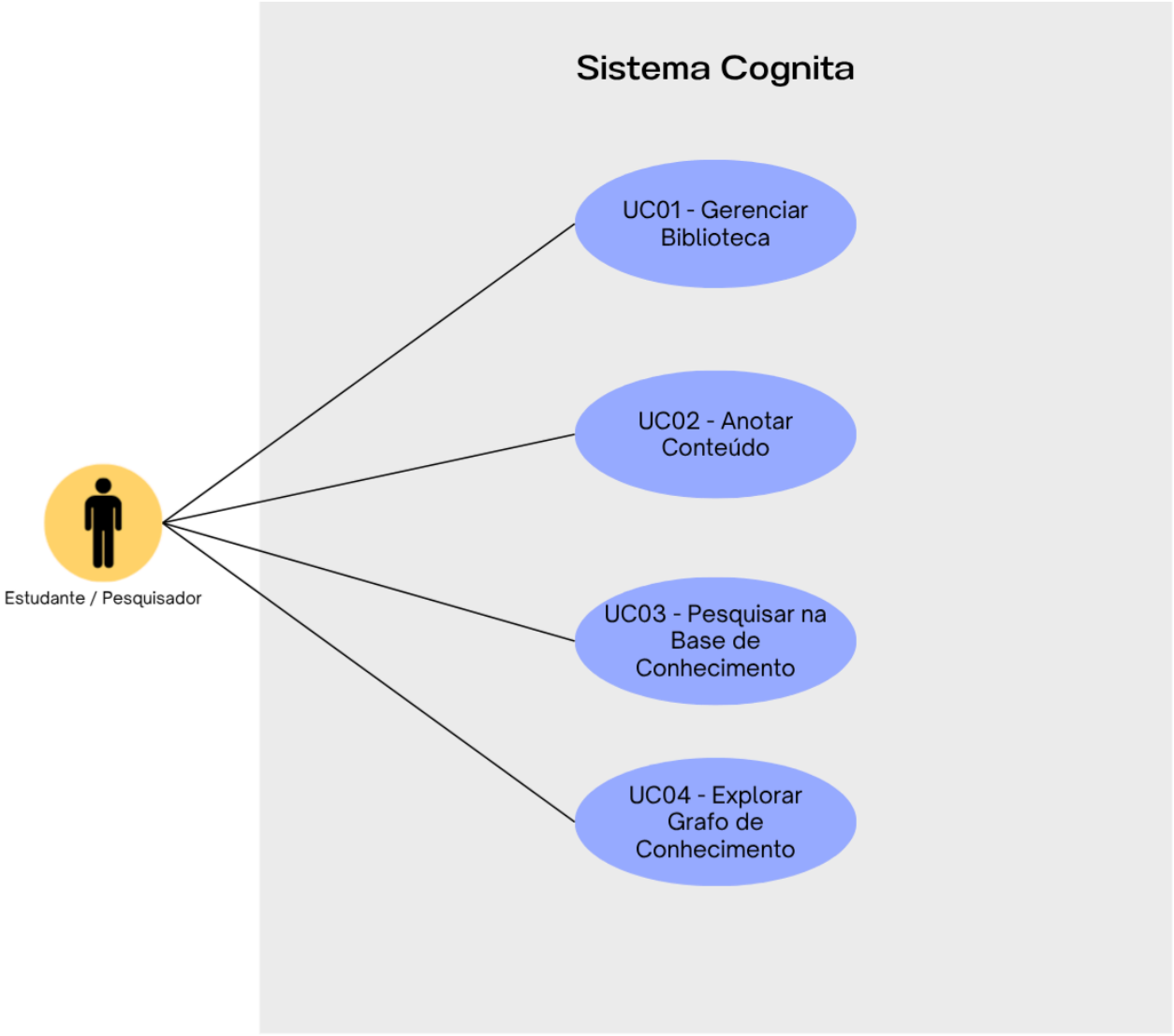

Figura 1. Diagrama de Caso de Uso

2.3.2 Histórias de Usuário

As histórias de usuário a seguir, apresentadas na Tabela 2, são formuladas a partir da perspectiva da persona "Ana Oliveira" e estão agrupadas por épicos que correspondem às funcionalidades centrais do sistema.

Caso de Uso	ID	Historia

Gerenciar Biblioteca	US 01	Como Ana, a mestranda, eu quero fazer o upload de múltiplos artigos em PDF de uma só vez, para que eu possa rapidamente adicionar todo o material de um novo tópico de pesquisa à minha biblioteca.
	US 02	Como Ana, eu quero organizar meus documentos em projetos, para que eu possa separar os materiais de diferentes matérias ou capítulos da minha dissertação.
Anotar Conteúdo	US 03	Como Ana, eu quero fazer anotações manuscritas diretamente nas margens de um PDF usando minha caneta *stylus*, para que eu possa capturar ideias da mesma forma que faria em papel.
	US 04	Como Ana, eu quero criar notas independentes que não estão atreladas a nenhum documento, para que eu possa registrar *brainstormings* ou ideias gerais.
	US 05	Como Ana, eu quero que minhas anotações sejam salvas automaticamente e estejam disponíveis offline, para que eu possa estudar em qualquer lugar sem me preocupar em perder meu trabalho.
Pesquisa na Base de Conhecimento	US 06	Como Ana, eu quero que o sistema processe automaticamente o texto de PDFs digitalizados e de minhas anotações manuscritas, para que todo o conteúdo da minha biblioteca se torne pesquisável.
	US 07	Como Ana, eu quero realizar uma busca por uma palavra-chave e ver resultados de todos os meus documentos e notas (digitadas e manuscritas), para que eu possa encontrar rapidamente qualquer informação que já estudei.
Explorar Grafo de Conhecimento	US 08	Como Ana, eu quero ter uma visão geral que me mostre os tópicos principais dos meus estudos e como eles se ligam, para que eu possa entender o quadro geral de um assunto complexo sem me perder nos detalhes.
	US 09	Como Ana, ao ver uma conexão entre dois tópicos, eu quero poder clicar nela e ser levada instantaneamente para a frase ou anotação exata em cada documento, para que eu possa relembrar e validar o contexto original daquela ligação.

Tabela 2. Histórias de Usuário

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Esta subseção apresenta os Diagramas de Sequência do Sistema (DSS) para os principais casos de uso, ilustrando a ordem cronológica das interações entre o ator "Estudante / Pesquisador" e o sistema Cognita. Para cada operação de sistema iniciada pelo ator em um DSS, é detalhado um Contrato de Operação, que formaliza as responsabilidades do sistema, especificando as pré-condições necessárias para a operação e as pós-condições que garantem seu resultado.

Os diagramas a seguir detalham os fluxos de interação mais complexos e centrais para a proposta de valor do sistema Cognita, focando nos comportamentos assíncronos e nas interações ricas em dados. Fluxos de CRUD mais simples, como a criação de um novo projeto (US 02), foram omitidos por não apresentarem complexidade dinâmica significativa, estando sua lógica já representada no Diagrama de Classes. Da mesma forma, o fluxo para criar notas independentes (US 04) é coberto pelo diagrama de 'Fazer Anotação Manuscrita' (Figura 3), pois o comportamento dinâmico da interação é idêntico.

2.4.1 DSS – Realizar Upload de Documento

O diagrama apresentado na Figura 2, modela o fluxo de interação para as histórias de usuário US 01 e US 06. A modelagem é unificada em um único diagrama pois a ação do usuário de realizar o *upload* de um arquivo (solicitarUpload) é o gatilho para duas funcionalidades interligadas: a gestão do arquivo na biblioteca (atendendo à US 01) e o início do processamento assíncrono de conteúdo (OCR/HCR) pelo sistema para tornar esse arquivo pesquisável (atendendo à US 06). A Tabela 3 descreve o contrato do fluxo apresentado.

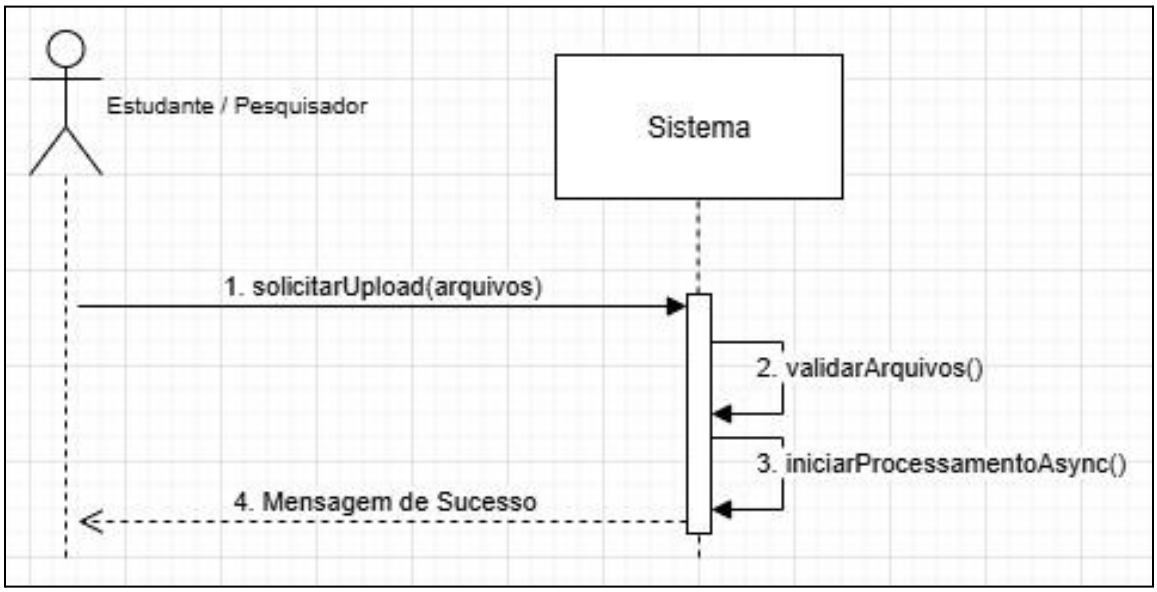

Figura 2. DSS para Upload de Documento

Contrato	*Upload* de Documento
Operação	solicitarUpload(arquivos)

Referências cruzadas	Histórias de Usuário: US 01, US 06
Pré-condições	O ator "Estudante / Pesquisador" deve estar autenticado no sistema
Pós-condições	O documento é adicionado à biblioteca do usuário com o estado "Processando". O sistema inicia o processamento do conteúdo do documento em segundo plano. O usuário recebe uma confirmação do *upload*.

Tabela 3. Contrato Upload de Documento

2.4.2 DSS – Fazer Anotação Manuscrita

Este diagrama, apresentado na Figura 3, modela o fluxo de interação para as histórias de usuário US 03 e US 05. O fluxo único atende a duas necessidades complementares: a sequência de eventos de desenho representa a funcionalidade de anotação manuscrita (atendendo à US 03), enquanto a etapa salvarTracoLocal, que ocorre ao final do gesto, é a implementação técnica que garante que a anotação seja salva automaticamente e esteja disponível offline (atendendo à US 05). A Tabela 4 descreve o contrato do fluxo apresentado.

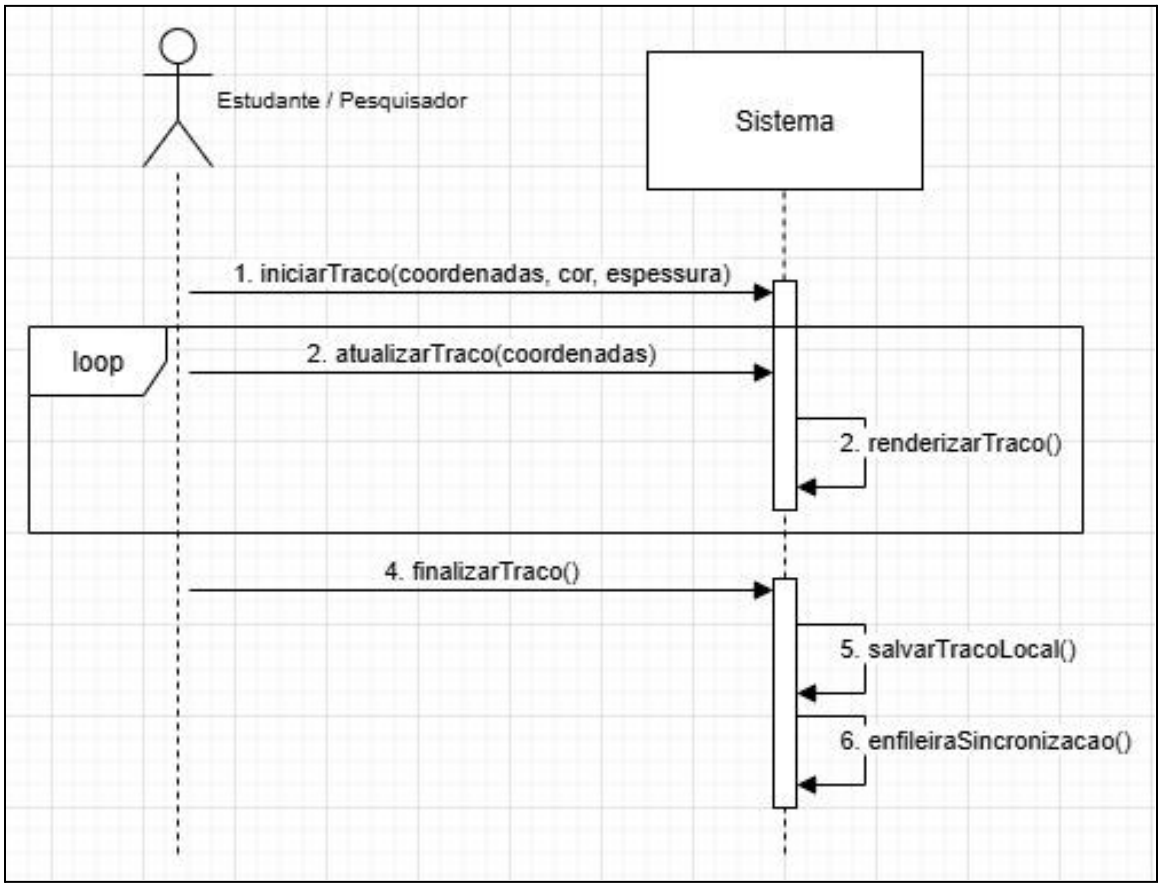

Figura 3. DSS para Fazer Anotação Manuscrita

Contrato	Anotação Manuscrita
Operação	iniciarTraco()
Referências cruzadas	Histórias de Usuário: US 03, US 05
Pré-condições	Um Document ou uma nota deve estar aberto na interface de anotação.
Pós-condições	O traço manuscrito é exibido na tela. Ao finalizar o traço, a anotação é salva permanentemente no banco de dados local. A anotação é enfileirada para sincronização com a nuvem.

Tabela 4. Contrato Anotação Manuscrita

2.4.3 DSS – Realizar Busca Global

Este diagrama, apresentado na Figura 4, modela o fluxo de interação para a história de usuário US 07, onde o usuário busca por um termo em toda a sua base de conhecimento. A Tabela 5 descreve o contrato do fluxo apresentado.

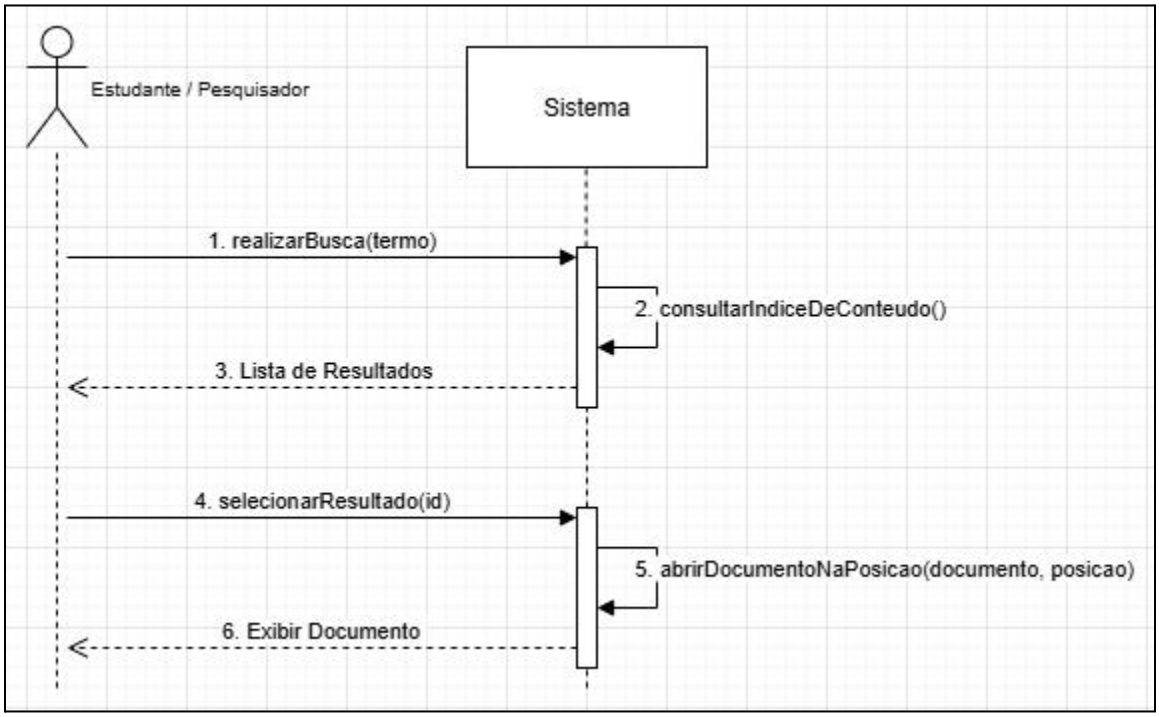

Figura 4. DSS para Realizar Busca Global

Contrato	Realizar Busca Global
Operação	realizarBusca(termo: string), selecionarResultado(id: string)
Referências cruzadas	História de Usuário: US 07
Pré-condições	O ator "Estudante / Pesquisador" deve estar autenticado no sistema
Pós-condições	Uma lista de resultados da busca é exibida para o usuário.

	O documento correspondente é aberto na localização exata do resultado, com o termo destacado.

Tabela 5. Contrato Realizar Busca Global

2.4.4 DSS – Explorar Conexão no Grafo

Este diagrama, apresentado na Figura 5, modela a jornada interativa de descoberta de conhecimento, que abrange as histórias de usuário US 08 e US 09. A primeira parte do fluxo, onde o usuário solicita e o sistema exibe o grafo, atende à necessidade de "ter uma visão geral" (US 08). A segunda parte, que detalha a ação de clicar em uma conexão para ver o contexto e navegar até a fonte, representa a principal ação de exploração (US 09). O DSS, portanto, modela a experiência completa, desde a visualização macro até o "mergulho" em um *insight* específico. A Tabela 6 descreve o contrato do fluxo apresentado.

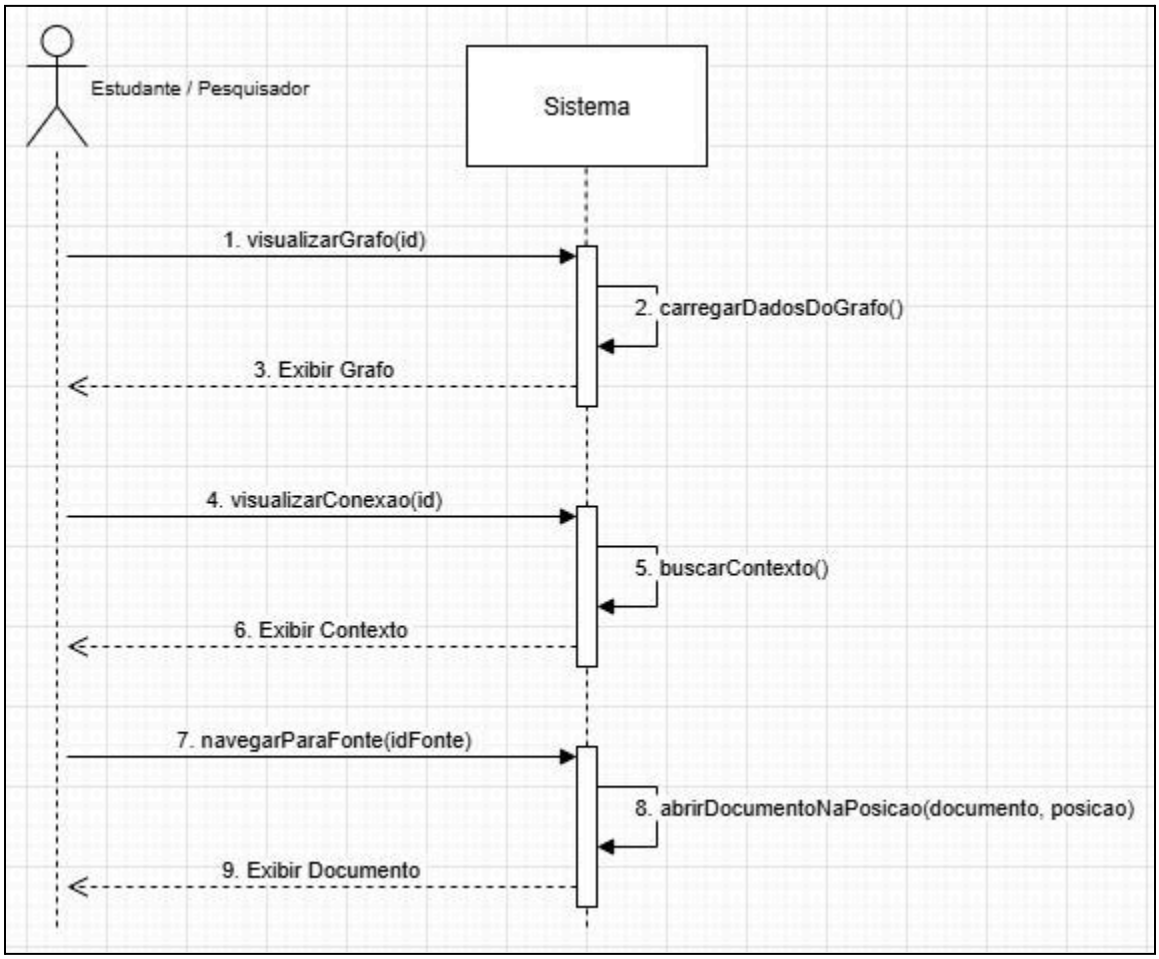

Figura 5. DSS para Explorar Conexão no Grafo

Contrato	Explorar Conexão no Grafo
Operação	visualizarGrafo(id)

Referências cruzadas	Histórias de Usuário: US 08, US 09
Pré-condições	O ator "Estudante / Pesquisador" deve estar autenticado no sistema
Pós-condições	O grafo de conhecimento do projeto é exibido de forma interativa na tela. O contexto da conexão selecionada (trechos de texto das fontes) é exibido para o usuário. O documento original é aberto na localização exata da fonte selecionada.

Tabela 6. Contrato Explorar Conexão no Grafo

3. Modelos de Projeto

Esta seção é dedicada à apresentação dos modelos de projeto que definem a arquitetura e o *design* detalhado do sistema Cognita. O objetivo é fornecer um conjunto de *blueprints* técnicos que descrevem o *software* a partir de diferentes perspectivas, servindo como um guia essencial para a fase de implementação. Serão apresentados os diagramas UML (*Unified Modeling Language*) que ilustram tanto a estrutura estática quanto o comportamento dinâmico do sistema. A modelagem começa com o Diagrama de Classes, que representa a espinha dorsal da arquitetura. Em seguida, serão detalhados os Diagramas de Sequência, Comunicação, Arquitetura, Estados, Componentes e Implantação utilizados para descrever a organização física e a distribuição do *software*.

3.1 Diagrama de Classes

O Diagrama de Classes, apresentado na Figura 6, é a principal representação da estrutura estática do sistema Cognita. Ele descreve as entidades fundamentais do domínio, seus atributos e operações, e os relacionamentos que existem entre elas, formando a base sobre a qual todas as funcionalidades serão construídas.

O modelo é centrado na classe User, que representa o ator do sistema. A partir dela, estabelece-se uma hierarquia de posse através de relações de Composição (representadas pelo losango preto), que indicam um forte vínculo de ciclo de vida: se o "todo" é destruído, a "parte" também é. Assim, um User possui múltiplos Projects, que por sua vez compõem tanto os Documents (arquivos PDF) quanto as Annotations independentes. Esta modelagem permite que um projeto funcione como um "bloco de notas", contendo anotações que não estão atreladas a nenhum documento específico, atendendo à história de usuário US 04. Da mesma forma, um Document também compõe suas próprias Annotations, que representam os destaques e notas manuscritas feitas sobre o PDF.

A inovação central do sistema, o grafo de conhecimento, é representada por um conjunto de classes interligadas. A classe KnowledgeNode modela os conceitos abstratos extraídos pelo sistema (ex: "Semana de Arte Moderna"), funcionando como os nós do grafo. A relação semântica entre dois KnowledgeNodes é descrita pela classe de associação KnowledgeEdge, que representa as arestas do grafo com um rótulo específico (ex: "ocorreu_em"). A conexão crucial entre o conteúdo concreto e

os conceitos abstratos é realizada pela classe Mention. Ela representa a ocorrência específica de um KnowledgeNode em uma localização exata e seu ciclo de vida depende inteiramente de sua fonte, sendo composta por um Document ou por uma Annotation. Esta classe é a ponte que permite a navegação contextual da visualização do grafo para a fonte original da informação, uma funcionalidade essencial do Cognita.

Para garantir a consistência e a segurança de tipo, o modelo também utiliza enumerações (<<enumeration>>) para atributos com um conjunto fixo de valores, como DocumentStatus, AnnotationType e NodeType.

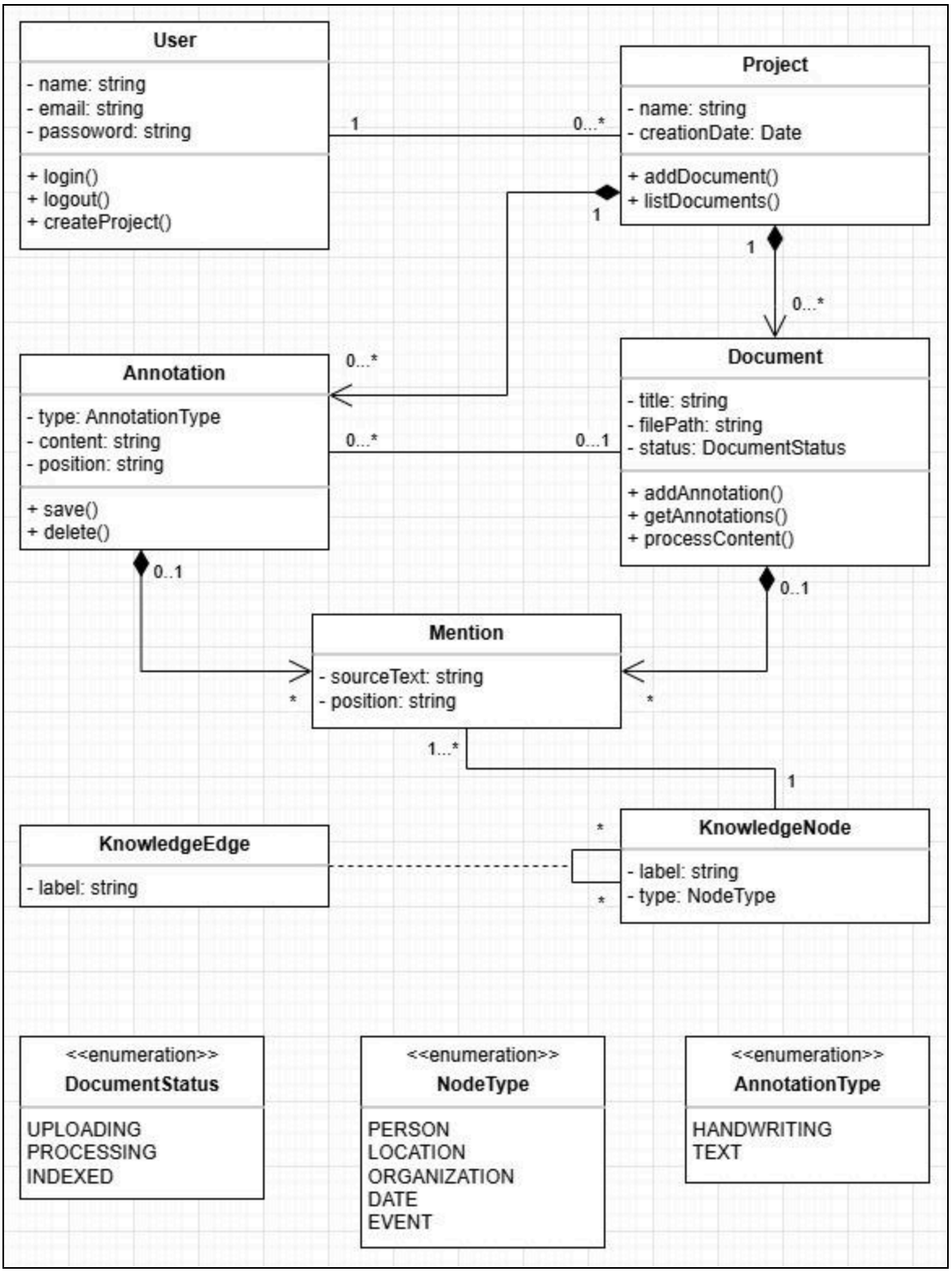

Figura 6. Diagrama de Classes do Sistema Cognita

3.2 Diagramas de Sequência

Os Diagramas de Sequência desta seção oferecem uma visão "caixa-branca" do sistema Cognita, detalhando como os objetos internos colaboram para realizar os principais casos de uso. Diferente dos Diagramas de Sequência do Sistema (DSS), que focam na fronteira do sistema, estes diagramas aprofundam a modelagem, ilustrando a troca de mensagens entre os componentes da interface, os controladores, os serviços de *back-end* e as camadas de persistência de dados.

3.2.1 Diagrama de Sequência – Fazer Anotação Manuscrita

A Figura 7 detalha a interação para as histórias de usuário US 03 e US 05. O diagrama ilustra a arquitetura offline-first do sistema. A interação começa com o ator desenhando na Interface Usuário, que repassa os eventos para o :ControllerAnotacao. Dentro do fragmento de *loop*, o controlador comanda a renderização do traço. Ao finalizar o gesto (quando o usuário levanta a caneta), o controlador primeiro salva a anotação no :Banco de Dados (Local) e, em seguida, enfileira a sincronização com o :APIServidor de forma assíncrona, garantindo que a interface permaneça sempre fluida e responsiva, independentemente da conexão de rede.

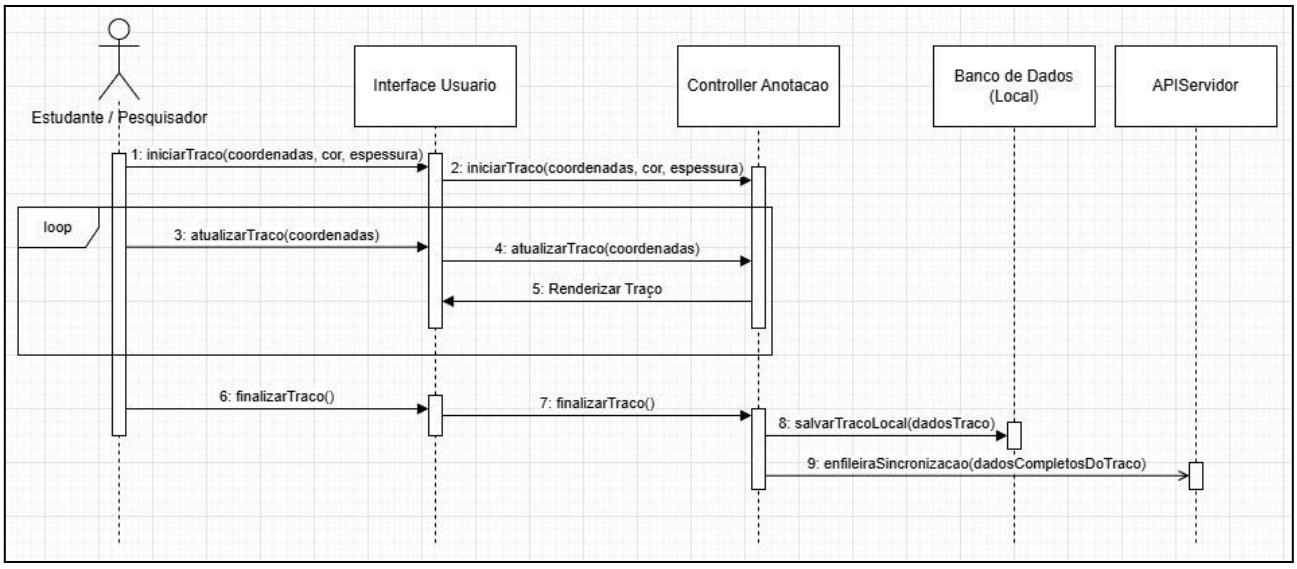

Figura 7. Diagrama de Sequência - Fazer Anotação Manuscrita

3.2.2 Diagrama de Sequência - Realizar Upload de Documento

A Figura 8 modela o fluxo para as histórias de usuário US 01 e US 06. Este diagrama evidencia a arquitetura de processamento assíncrono do sistema. Após o usuário iniciar o *upload*, o :ControllerBiblioteca envia os arquivos para o :APIServidor. O servidor, por sua vez, cria um registro inicial do documento no banco de dados com o estado "PROCESSANDO" e imediatamente enfileira a tarefa de processamento pesado (OCR/PLN) para o :ProcessadorConteudo. Uma confirmação é retornada ao usuário sem esperar o término do processamento, garantindo que a aplicação não trave durante operações demoradas.

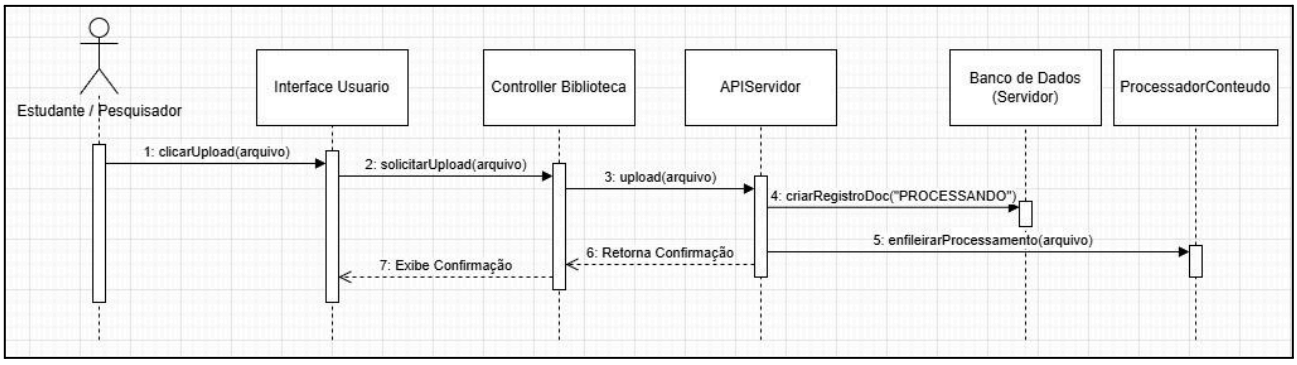

Figura 8. Diagrama de Sequência - Realizar Upload de Documento

3.2.3 Diagrama de Sequência – Realizar Busca Global

A Figura 9 ilustra a jornada completa do usuário para a história US 07. O fluxo é dividido em duas fases. Na primeira, o usuário digita um termo, que é enviado através das camadas de controle até o :APIServidor. O servidor consulta um índice de conteúdo otimizado no banco de dados — e não o texto bruto dos documentos — para retornar uma lista de resultados. Na segunda fase, ao selecionar um resultado, o controlador comanda a interface para abrir o documento na posição exata da ocorrência, completando o ciclo de busca e recuperação da informação.

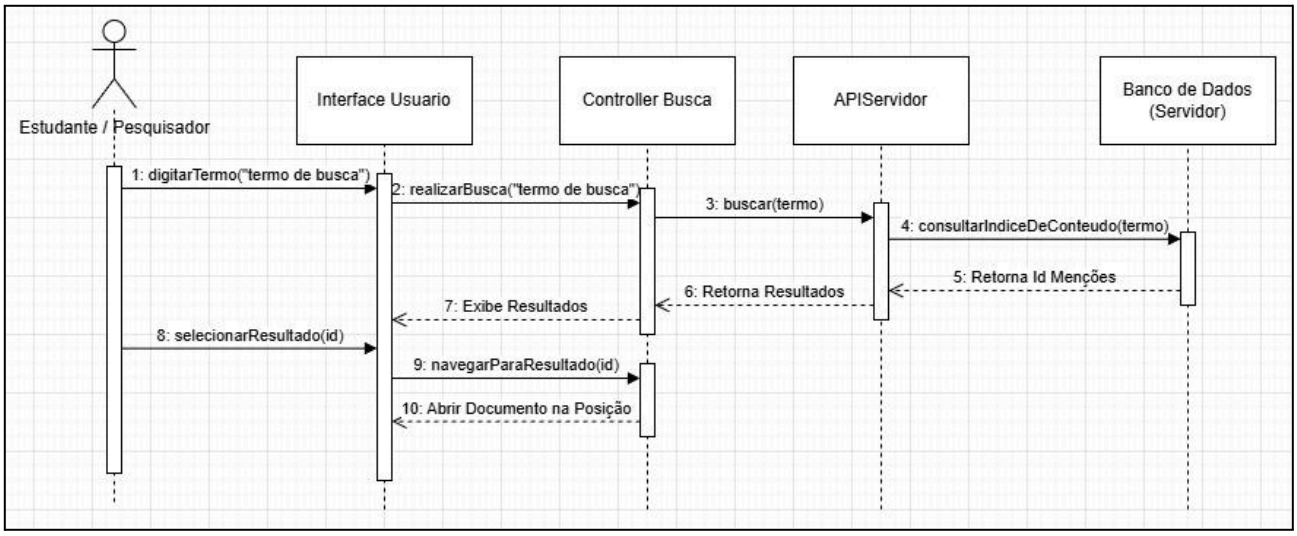

Figura 9. Diagrama de Sequência - Realizar Busca Global

3.2.4 Diagrama de Sequência – Explorar Conexão no Grafo

A Figura 10 detalha a interação com a funcionalidade mais inovadora do Cognita, correspondente às histórias US 08 e US 09. O diagrama modela a jornada de descoberta em três fases: primeiro, o usuário solicita a visualização do grafo de um projeto, e o sistema busca os nós e arestas no banco de dados para exibi-lo. Segundo, ao clicar em uma conexão, o sistema busca as "Menções" associadas para exibir o painel de contexto. Terceiro, ao selecionar uma fonte no painel de contexto, o sistema navega o usuário diretamente para a localização exata daquela informação no documento original, cumprindo a principal proposta de valor do projeto.

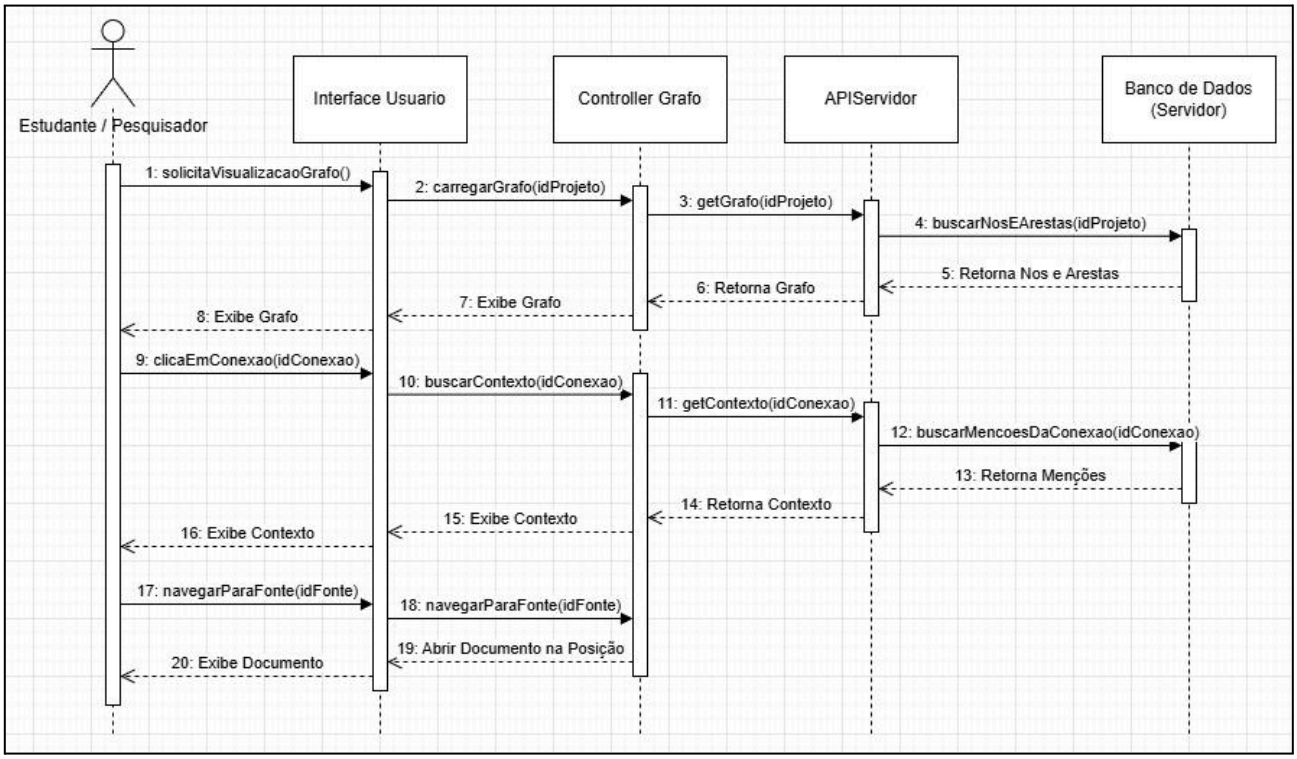

Figura 10. Diagrama de Sequência – Explorar Conexão no Grafo

3.3 Diagramas de Comunicação

Os Diagramas de Comunicação oferecem uma visão alternativa das interações modeladas nos Diagramas de Sequência. Enquanto os diagramas de sequência enfatizam a ordem temporal das mensagens, os diagramas de comunicação focam na estrutura das interações e nos relacionamentos (links) entre os objetos participantes.

Eles utilizam a mesma base de informações dos Diagramas de Sequência, mas representam o fluxo através de uma numeração sequencial das mensagens trocadas ao longo dos links que conectam os objetos. A seguir, são apresentados os diagramas de comunicação para os principais casos de uso do sistema Cognita.

3.3.1 Diagrama de Comunicação – Fazer Anotação Manuscrita

A Figura 11 ilustra a colaboração entre os objetos para as histórias de usuário US 03 e US 05. O diagrama destaca os links de comunicação necessários para o processo de criação de uma anotação manuscrita, desde a captura do gesto na interface até a persistência local e o enfileiramento assíncrono para o *back-end*, evidenciando a arquitetura *offline-first*.

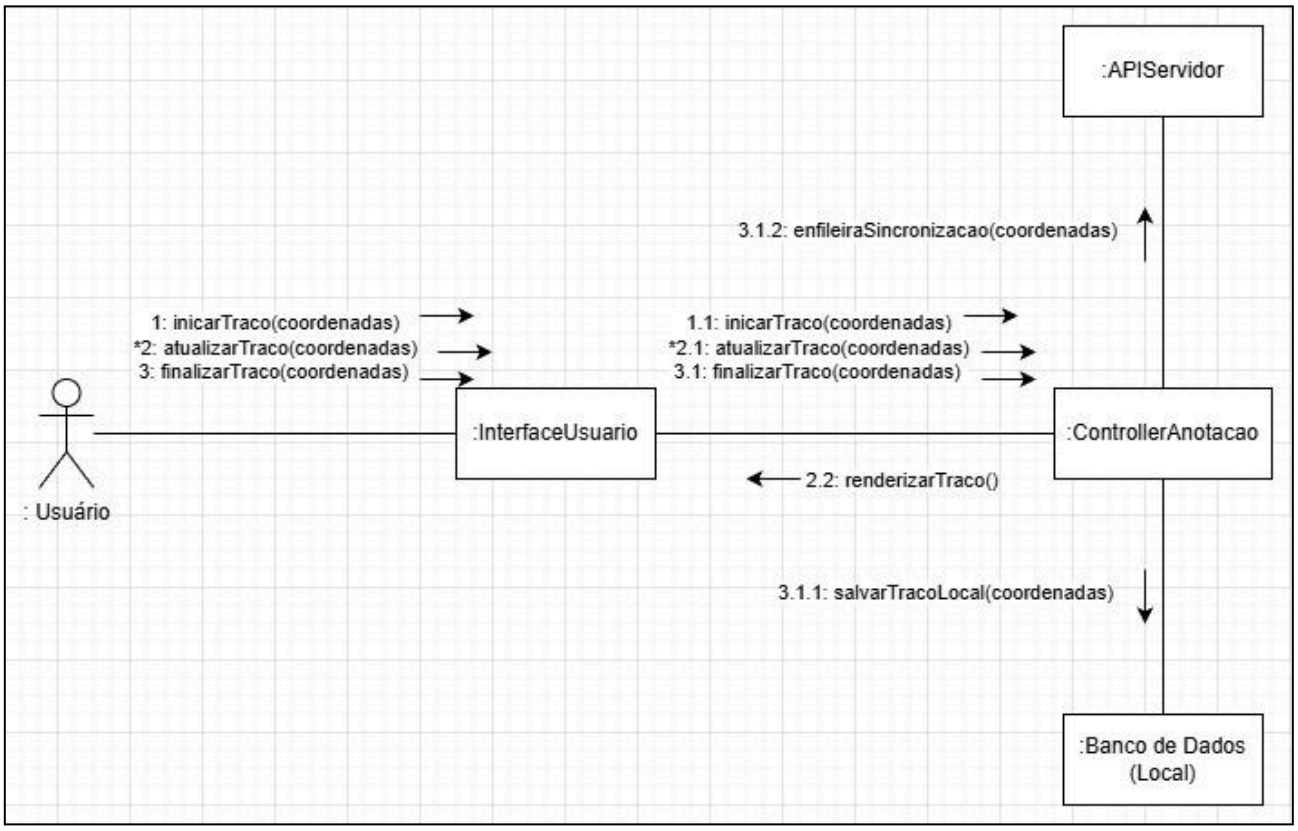

Figura 11. Diagrama de Comunicação – Fazer Anotação Manuscrita

3.3.2 Diagrama de Comunicação – Realizar Upload de Documento

A Figura 12 representa a colaboração de objetos para as histórias de usuário US 01 e US 06. O diagrama mostra como o cliente (:ControllerBiblioteca) se comunica apenas com a porta de entrada do servidor (:APIServidor), que por sua vez orquestra as ações internas de salvar o registro no banco de dados e enfileirar a tarefa de processamento para o :ProcessadorConteudo de forma assíncrona.

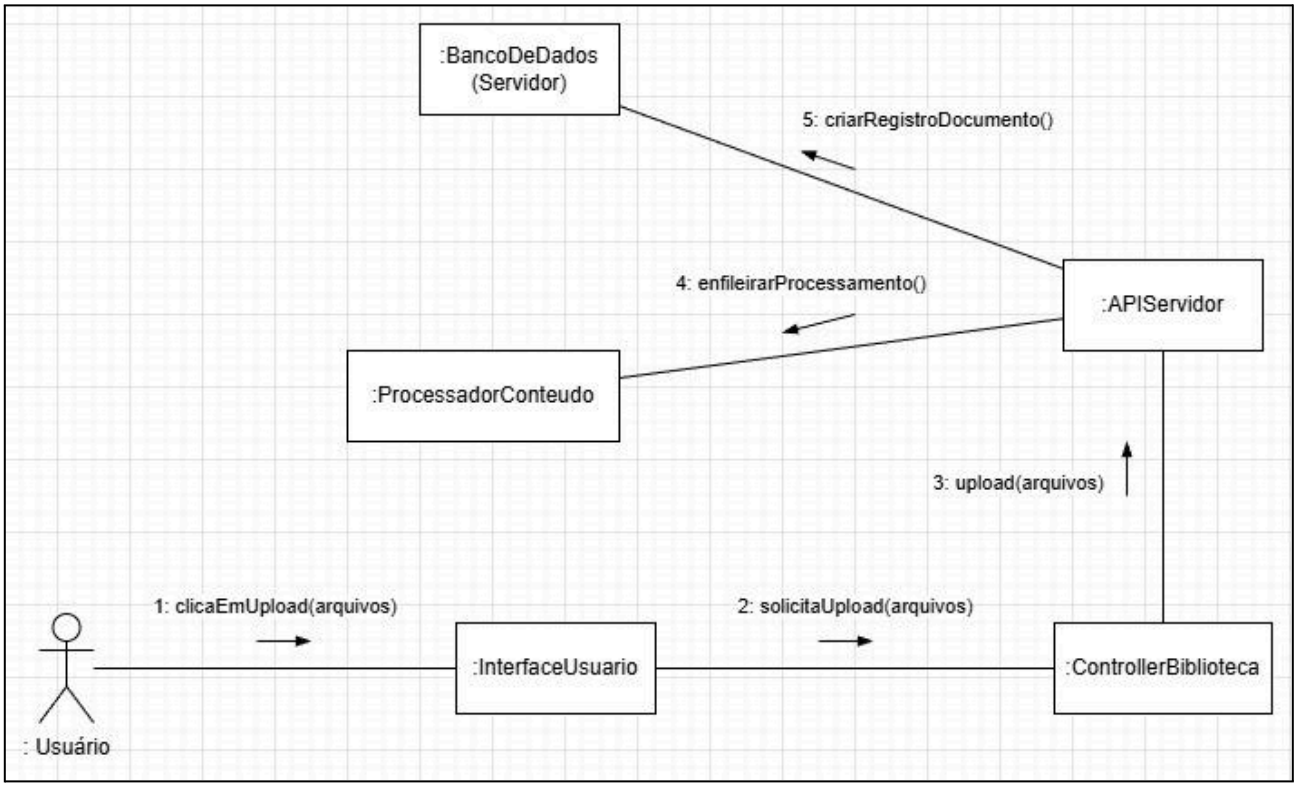

Figura 12. Diagrama de Comunicação – Realizar Upload de Documento

3.3.3 Diagrama de Comunicação – Realizar Busca Global

A Figura 13 detalha a colaboração para a história de usuário US 07. O diagrama foca nos links entre os componentes para realizar a busca completa. Ele mostra a cadeia de comunicação, desde a consulta do usuário na interface até a chamada à API, que consulta o índice de conteúdo no banco de dados e, em uma segunda fase, como a seleção de um resultado na interface aciona o controlador para exibir o documento na posição correta.

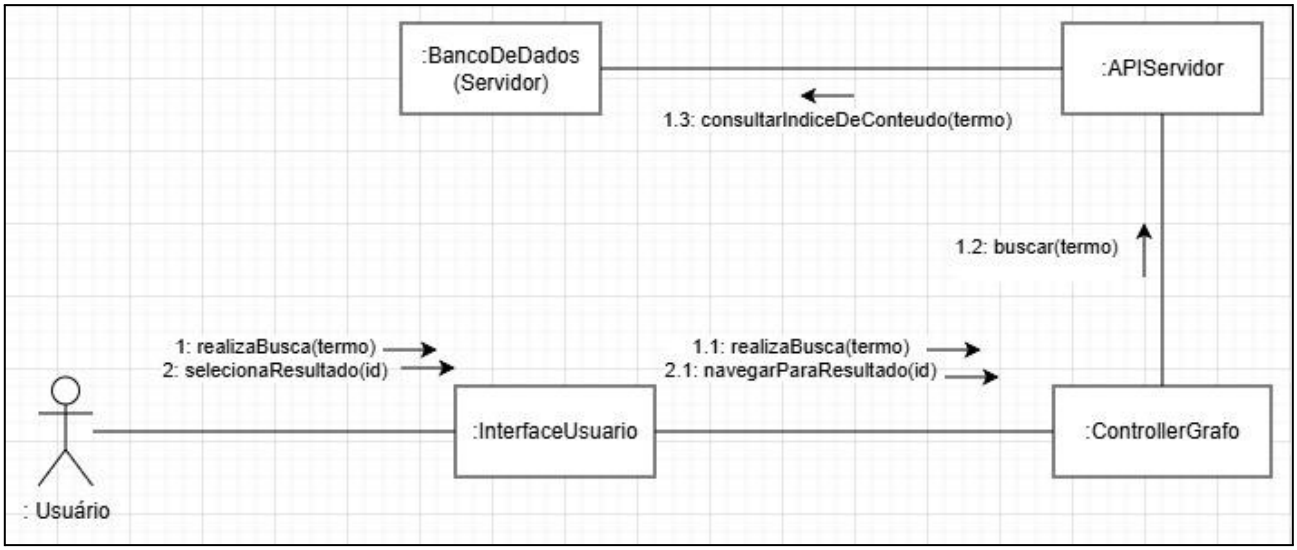

Figura 13. Diagrama de Comunicação – Realizar Busca Global

3.3.4 Diagrama de Comunicação – Explorar Conexão no Grafo

A Figura 14 ilustra a colaboração de objetos para as histórias de usuário US 08 e US 09, que representam a principal funcionalidade do sistema. O diagrama mostra os *links* necessários para a jornada completa do usuário: primeiro, a busca dos dados do grafo; segundo, a busca pelo contexto de uma conexão específica; e terceiro, a navegação para a fonte original no documento, demonstrando a complexa orquestração entre os componentes para entregar o *insight* ao usuário.

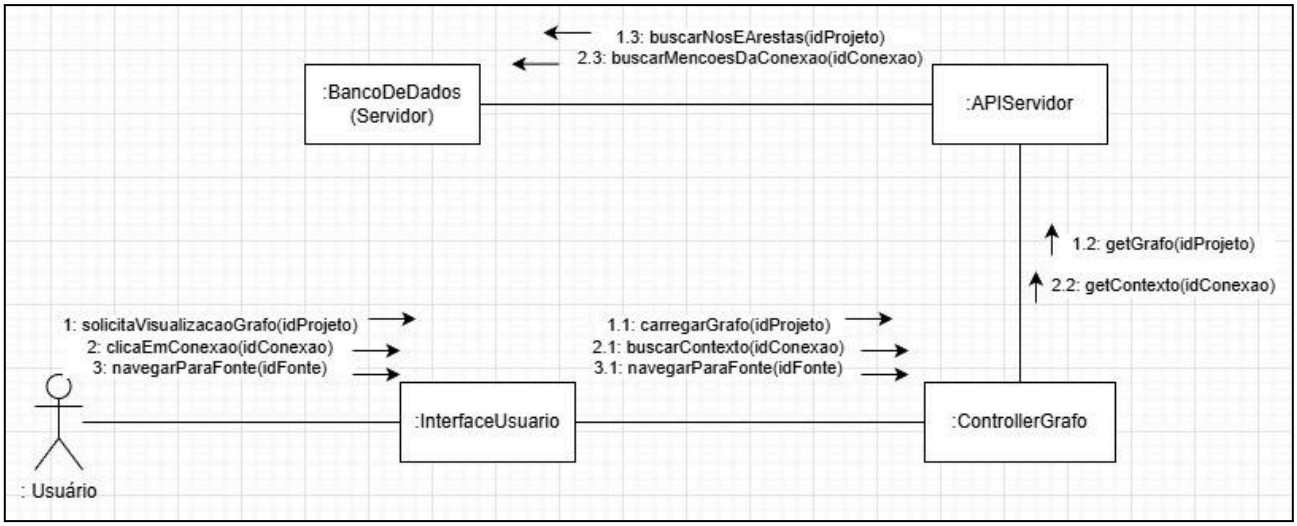

Figura 14. Diagrama de Comunicação – Explorar Conexão no Grafo

3.4 Arquitetura

A arquitetura do sistema Cognita foi projetada para atender a requisitos críticos de performance, escalabilidade e confiabilidade, garantindo uma experiência de usuário fluida e responsiva, especialmente nas funcionalidades de anotação. O modelo arquitetural escolhido é uma Arquitetura Cliente-Servidor em Camadas, com uma ênfase particular em um design *Offline-First* para a aplicação cliente.

A arquitetura é dividida em dois macro componentes principais: a Aplicação Cliente, que roda no *tablet* do usuário, e o Servidor *Backend*, que centraliza a lógica de negócio pesada e a persistência de dados.

O cliente, desenvolvido em React Native, é o coração da interação do usuário e foi projetado seguindo o padrão *Offline-First*. Esta decisão é estratégica e visa garantir que a aplicação seja 100% funcional mesmo sem uma conexão com a internet, um requisito essencial para um estudante que pode estar em locais com conectividade instável. A performance da anotação manuscrita, que precisa ser instantânea, é o principal beneficiário desta abordagem.

O cliente é estruturado em três camadas lógicas:

- Camada de Apresentação (UI): Responsável por toda a renderização da interface do usuário, incluindo o canvas de anotação e a visualização do grafo. Ela captura os gestos do usuário e os repassa para a camada de controle.
- Camada de Controle (Lógica da Aplicação): Onde residem os "Controladores" (ControllerAnotacao, ControllerBiblioteca, etc.). Esta camada orquestra o fluxo de dados, gerencia o estado da aplicação e toma as decisões lógicas, como comandar a renderização de um traço ou iniciar uma busca.

- Camada de Persistência Local: Utiliza um banco de dados embarcado no dispositivo para salvar imediatamente todas as ações do usuário (anotações, criação de projetos, etc.). Esta camada é a fonte da verdade para a aplicação enquanto ela está em uso, garantindo que os dados sejam carregados e salvos de forma quase instantânea.

O servidor, *backend,* é responsável pelas operações que exigem processamento intensivo, persistência de dados centralizada e segurança. Ele também segue uma arquitetura em camadas para garantir a separação de responsabilidades e a manutenibilidade.

- *Gateway* de API (APIServidor): É a única porta de entrada para o *backend*. Ele expõe uma API REST para o cliente, validando todas as requisições e orquestrando as chamadas para os serviços internos.
- Camada de Serviço: Contém a lógica de negócio principal. É aqui que as requisições são processadas. Uma decisão arquitetural chave foi separar os serviços síncronos (como busca de dados) dos assíncronos.
- Processador de Conteúdo Assíncrono (ProcessadorConteudo): Um serviço especializado e desacoplado que opera em uma fila de tarefas. Ele é responsável por todas as operações pesadas e demoradas, como o processamento de OCR/HCR e a análise de PLN para a construção do grafo de conhecimento. Esta separação garante que a API principal permaneça sempre responsiva.
- Camada de Dados: Gerencia a comunicação com o banco de dados centralizado na nuvem, onde todos os dados dos usuários são armazenados de forma segura e persistente.

Como vantagens desta escolha de arquitetura:

- Performance e Responsividade Superior: A abordagem *Offline-First* no cliente garante que a interface nunca trave esperando por respostas da rede, proporcionando uma experiência de anotação natural e sem latência.
- Confiabilidade e Disponibilidade: O sistema é totalmente funcional *offline*. A sincronização com a nuvem ocorre em segundo plano, funcionando como um *backup* e permitindo o uso em múltiplos dispositivos, no caso da aplicação avançar no desenvolvimento.
- Escalabilidade: A separação do ProcessadorConteudo em um serviço assíncrono permite que o *backend* escale de forma independente. Se o número de uploads aumentar, apenas este serviço precisa de mais recursos, sem impactar a performance da API principal.
- Manutenibilidade: A divisão clara em camadas e componentes, tanto no cliente quanto no servidor, facilita a manutenção, os testes e a evolução do sistema.

A Figura 15 ilustra a arquitetura do Cognita dividida em dois pacotes principais: Cliente e Servidor. O pacote Cliente contém os sub-pacotes que representam suas camadas lógicas (UI, Controladores, PersistenciaLocal). O pacote Servidor contém os sub-pacotes que representam seus componentes internos (API, Servicos, Processadores, Dados). A seta de dependência tracejada mostra a única via de comunicação permitida: o pacote Cliente depende exclusivamente do pacote API do servidor, que atua como um gateway, encapsulando toda a complexidade do backend.

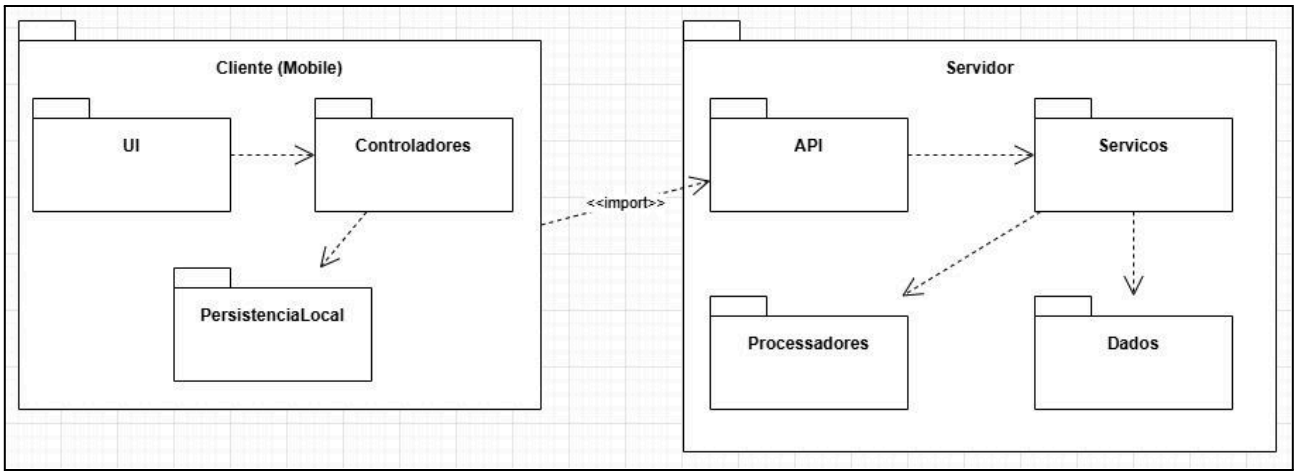

Figura 15. Diagrama de Pacotes

3.5 Diagramas de Estados

Para o sistema Cognita, o objeto mais relevante para este tipo de modelagem é a classe Document, devido ao seu ciclo de vida complexo que envolve o processamento assíncrono de conteúdo. A Figura 16 ilustra a máquina de estados para um objeto Document, desde o momento de seu *upload* até a finalização de seu processamento e indexação.

O diagrama mostra que, após ser criado, um Document entra no estado Aguardando Processamento. Quando o serviço de *backend* (ProcessadorConteudo) inicia a tarefa, o documento transita para o estado Processando Conteúdo. Este estado possui atividades internas (*do* /), que representam as operações contínuas de OCR, HCR e análise de PLN. Se todas as operações forem concluídas com êxito, o evento processamentoConcluidoComSucesso leva o documento para o estado final Indexado. Ao entrar (*entry* /) neste estado, seu status é atualizado e a interface do usuário é notificada. Caso ocorra qualquer erro, o evento erroNoProcessamento leva o documento para o estado Falha no Processamento, onde ações de registro de erro são executadas.

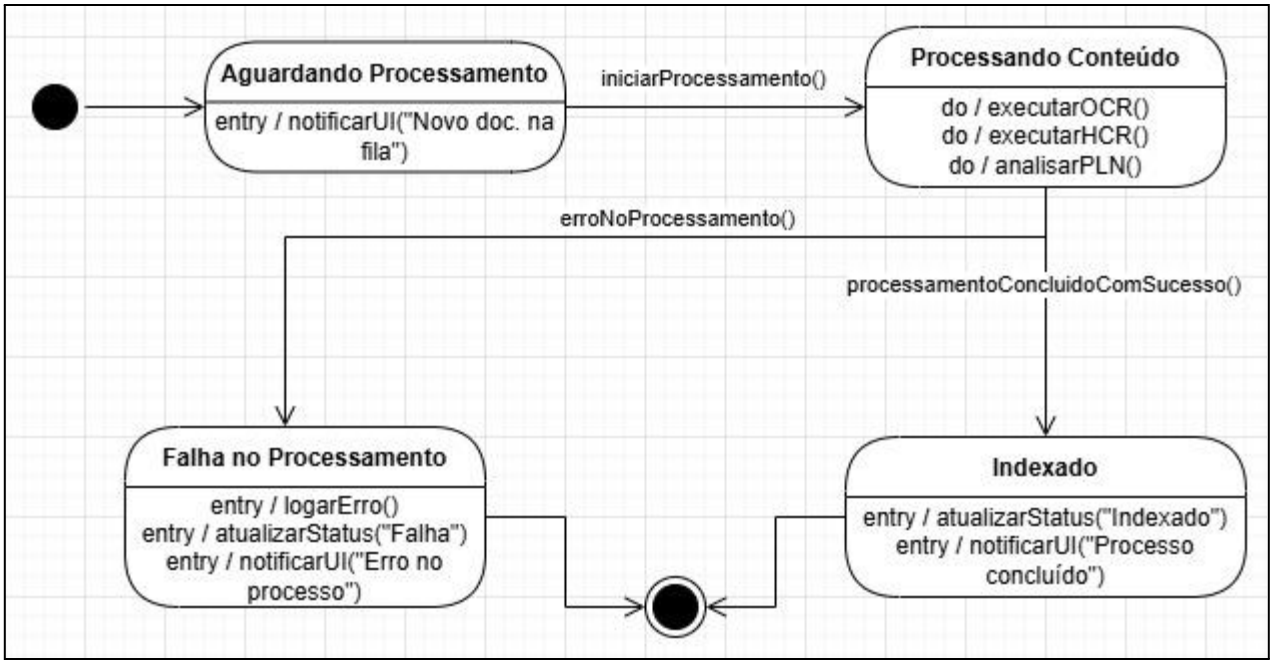

Figura 16. Diagrama de Estados

3.6 Diagrama de Componentes e Implantação

Esta seção finaliza a modelagem de projeto, focando na estrutura física e na distribuição dos artefatos de *software*. O Diagrama de Componentes, em particular, move a representação da arquitetura de uma visão puramente lógica (como no Diagrama de Pacotes) para uma visão mais concreta, que descreve os componentes de *software* reais, suas interfaces e as dependências entre eles

3.6.1 Diagrama de Componentes

O Diagrama de Componentes, apresentado na Figura 17, ilustra a arquitetura física do sistema Cognita. Ele detalha os principais componentes de *software* que constituem a aplicação cliente e o servidor *backend*, bem como as tecnologias e serviços de terceiros dos quais o sistema depende para operar.

O diagrama é dividido em dois grandes blocos: o Cliente e o Servidor.

É importante notar que a escolha de componentes externos, como o Google Cloud Storage e a Vision API, é guiada pela disponibilidade de um plano *free-tier* que atende às restrições de custo do projeto. Ciente de que estes planos estão sujeitos a limites e mudanças, a arquitetura do *backend* foi projetada para ser tecnologicamente agnóstica. Este design garante que qualquer um desses componentes possa ser substituído no futuro por uma alternativa, se necessário, com impacto mínimo na lógica de negócio central do sistema.

No lado do Cliente, temos o componente principal Aplicação Cliente, que será implementado em React Native. Este componente expõe a Interface de Usuário (UI) para o ator. Para suportar a arquitetura *offline-first*, ele depende diretamente do componente WatermelonDB, que representa o banco de dados local responsável pela persistência de dados no dispositivo do usuário.

No lado do Servidor, a arquitetura é composta por um componente central, o Servidor Backend (API), que expõe uma REST API como sua interface principal. Este componente é o único ponto de contato para o cliente, encapsulando toda a lógica de negócio e as interações com os demais serviços. Para executar suas funções, o Servidor Backend depende de um conjunto de componentes especializados:

- Armazenamento de Objetos (Google Cloud Storage): Componente responsável por armazenar de forma segura e escalável os arquivos PDF enviados pelos usuários. A escolha pelo Google Cloud Storage foi estratégica para manter a coesão do ecossistema de nuvem, uma vez que o sistema já depende de outros serviços da Google Cloud, como a Vision API.
- API de Visão (Google Cloud Vision): Um serviço externo do qual o *backend* depende para realizar as tarefas de Reconhecimento de Escrita à Mão (HCR) e Reconhecimento Óptico de Caracteres (OCR).
- Banco de Dados de Grafo (Neo4j): Componente central de persistência do servidor, responsável por armazenar todos os dados estruturados do sistema, incluindo usuários, projetos e, crucialmente, os nós e arestas do grafo de conhecimento.
- Modelos de PLN (Hugging Face): Representa a dependência do *backend* com um *pipeline* de modelos de Processamento de Linguagem Natural para realizar a construção do grafo. Para a tarefa de Reconhecimento de Entidades Nomeadas (NER), será utilizado um modelo pré-treinado da família BERT, como o marcosgg/bert-large-pt-ner-enamex, que é otimizado para o português. Para a tarefa de Extração de Relações (RE), a abordagem será utilizar um Grande Modelo de Linguagem (LLM) pré-treinado para português, através de técnicas de *prompting*, para extrair as relações semânticas entre as entidades identificadas.

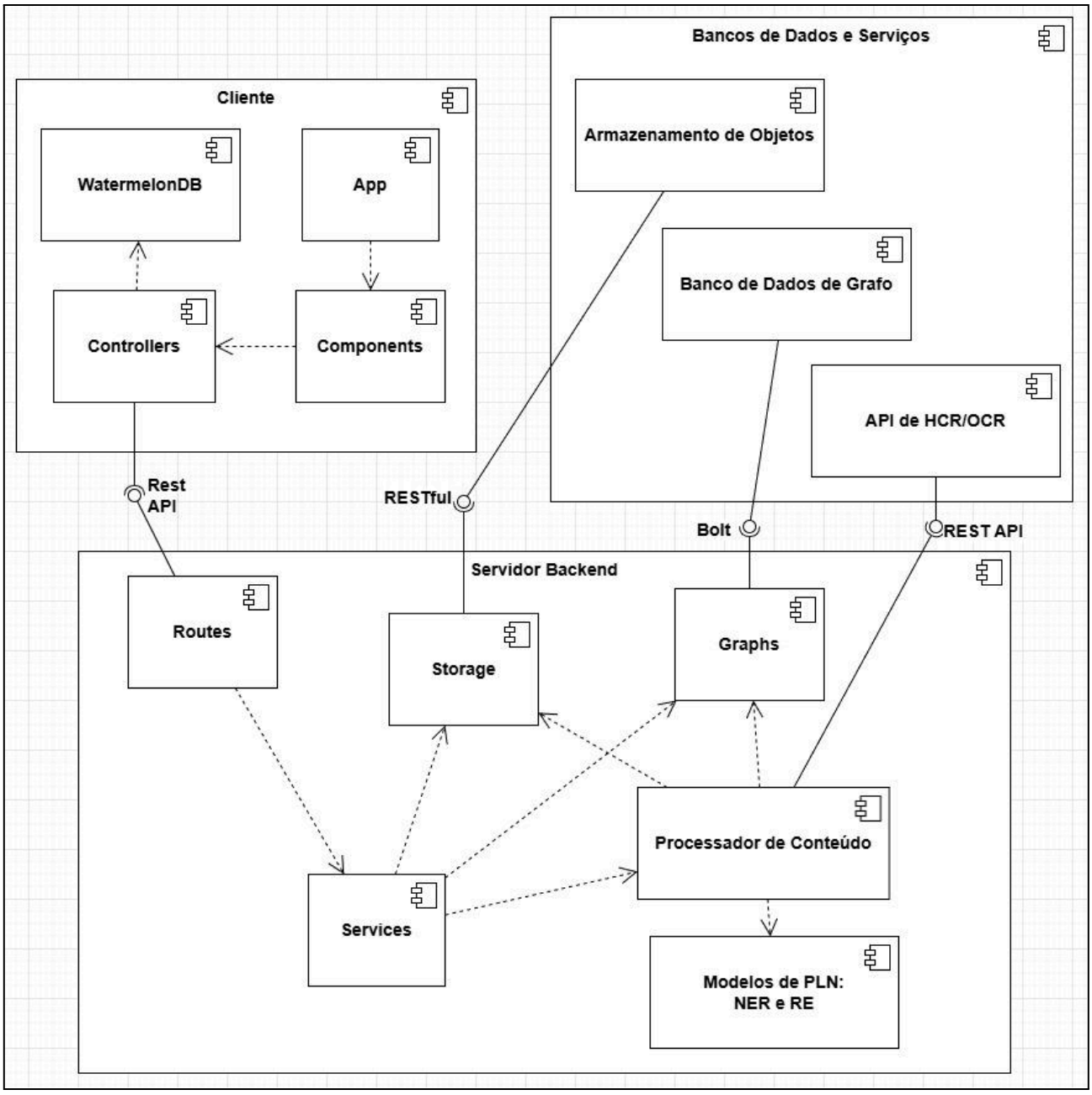

Figura 17. Diagrama de Componentes

3.6.2 Diagrama de Implantação

O Diagrama de Implantação complementa o Diagrama de Componentes ao modelar a topologia física do sistema. Ele ilustra como os artefatos de *software* (os componentes executáveis) são alocados nos nós de *hardware* ou ambientes de execução, e como esses nós se comunicam entre si.

A Figura 18 apresenta a arquitetura de implantação do sistema Cognita. Uma decisão de design importante neste diagrama é a representação dos serviços de nuvem de forma abstrata, focando em seu papel arquitetural em vez de um provedor específico. Isso reforça a independência tecnológica do sistema, onde a implementação concreta é um detalhe que pode ser alterado sem impactar a arquitetura de implantação geral.

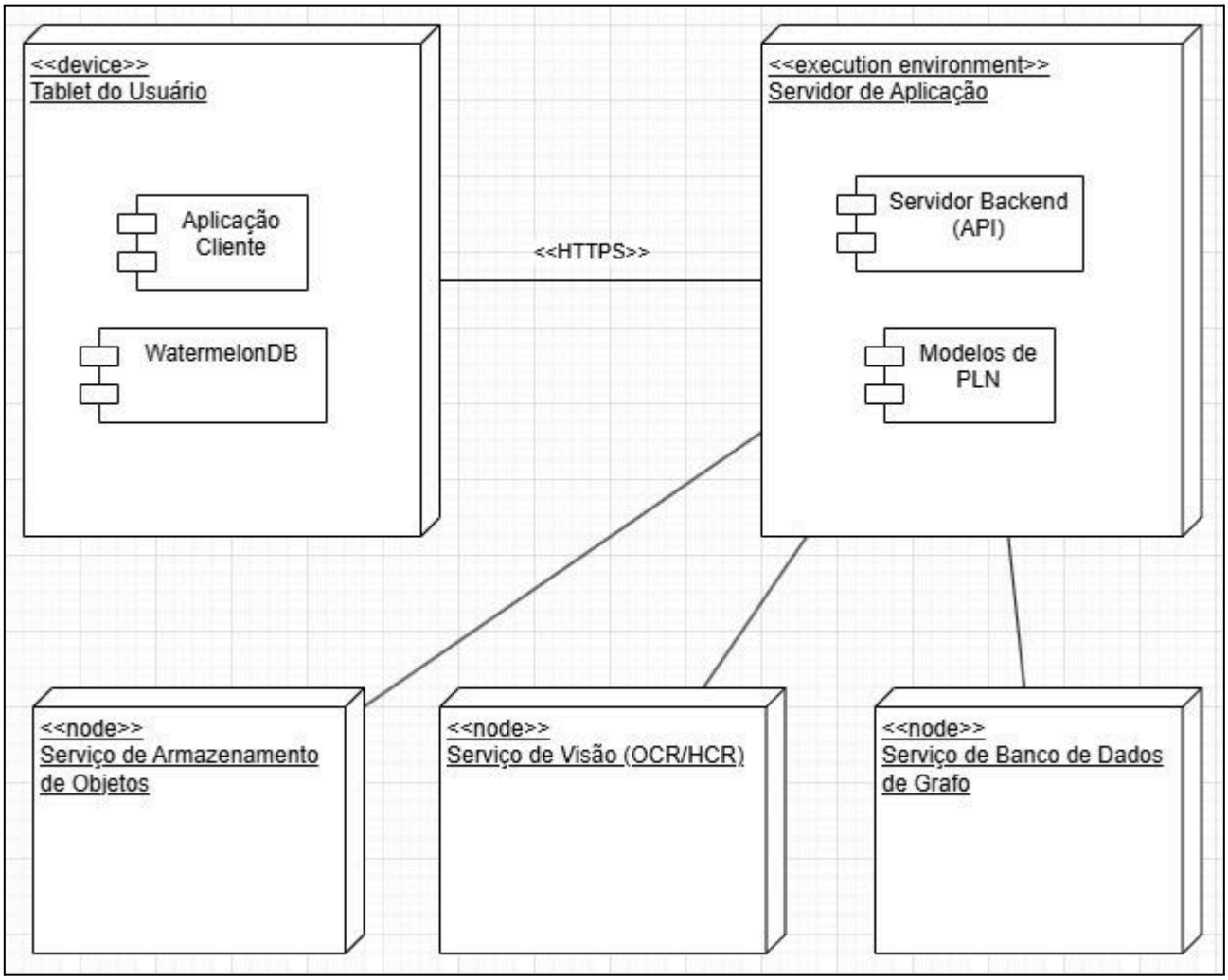

Figura 18. Diagrama de Implantação

4. Projeto de Interface com Usuário

Esta seção apresenta os wireframes de baixa fidelidade que esboçam a estrutura visual e o fluxo de navegação do sistema Cognita. O objetivo destes esboços é ilustrar as principais telas com as quais o ator "Estudante / Pesquisador" irá interagir, demonstrando como as funcionalidades descritas nos casos de uso e histórias de usuário se manifestam na interface. O design foi concebido com foco em uma experiência minimalista e funcional, otimizada para a interação em tablets.

4.1 Esboço das Interfaces Comuns a Todos os Atores

As interfaces a seguir representam o fluxo principal do usuário, desde a autenticação até a interação com as funcionalidades centrais de anotação e visualização do grafo de conhecimento.

As telas de autenticação, mostradas na Figura 19 e Figura 20, foram projetadas para serem simples e diretas, garantindo um acesso seguro e rápido à plataforma.

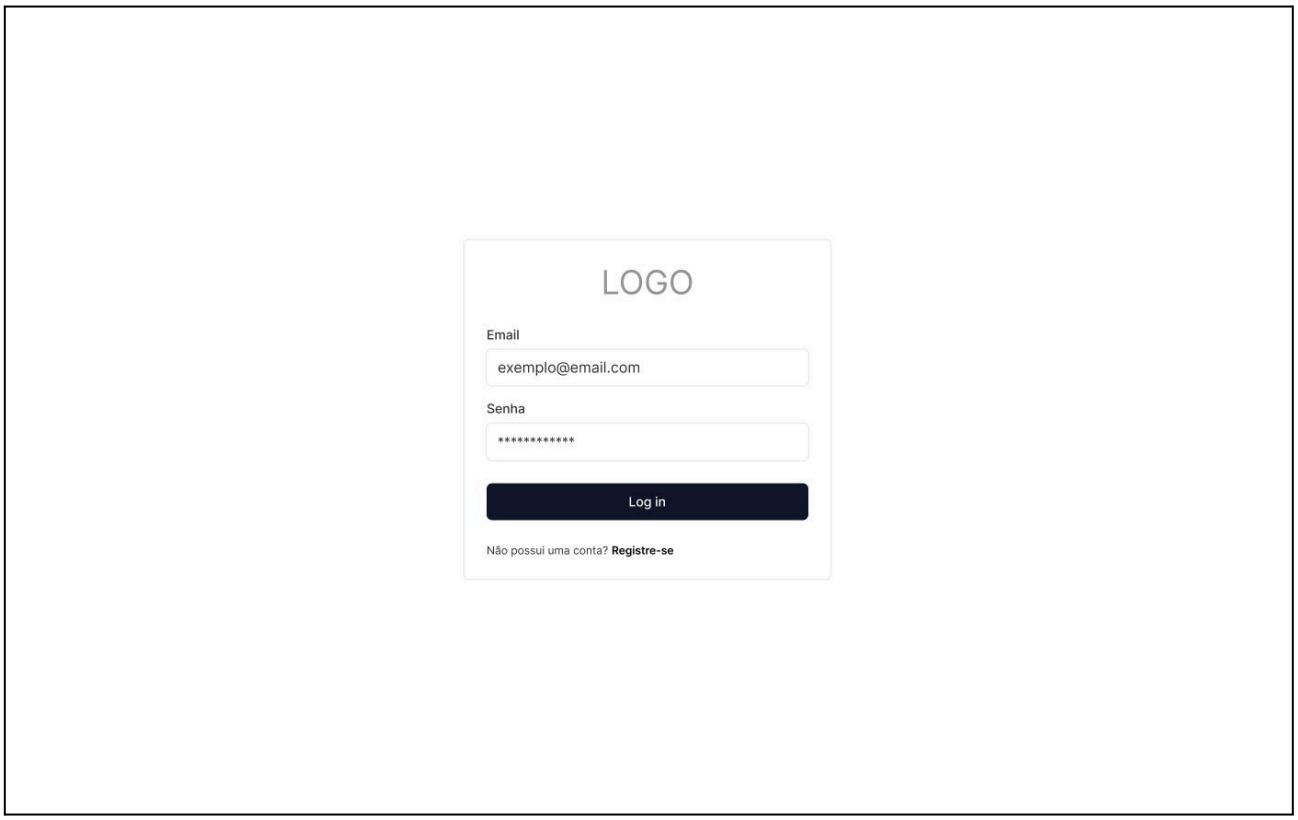

Figura 19. Tela de Login

A Figura 19 ilustra a tela de login, o ponto de entrada para usuários já cadastrados no sistema. A interface solicita apenas as credenciais essenciais (email e senha) para manter o foco na ação de acesso.

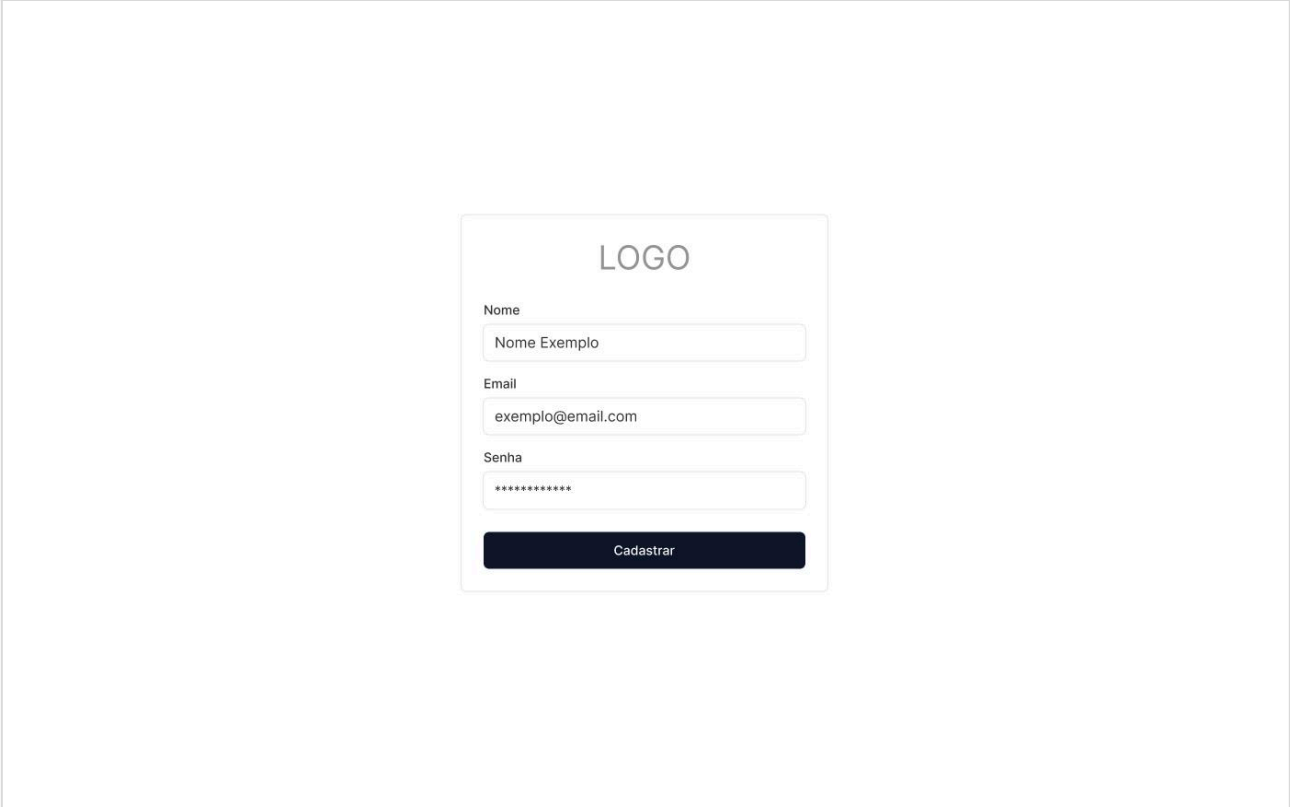

Figura 20. Tela de Cadastro

A Figura 20 apresenta a tela de cadastro para novos usuários. O formulário solicita as informações mínimas necessárias para a criação de uma conta: nome, email e senha.

Uma vez autenticado, o usuário tem acesso à área principal do sistema. A interface é organizada em um leiaute de múltiplos painéis para facilitar a navegação e a interação com os conteúdos, como demonstrado nas figuras subsequentes.

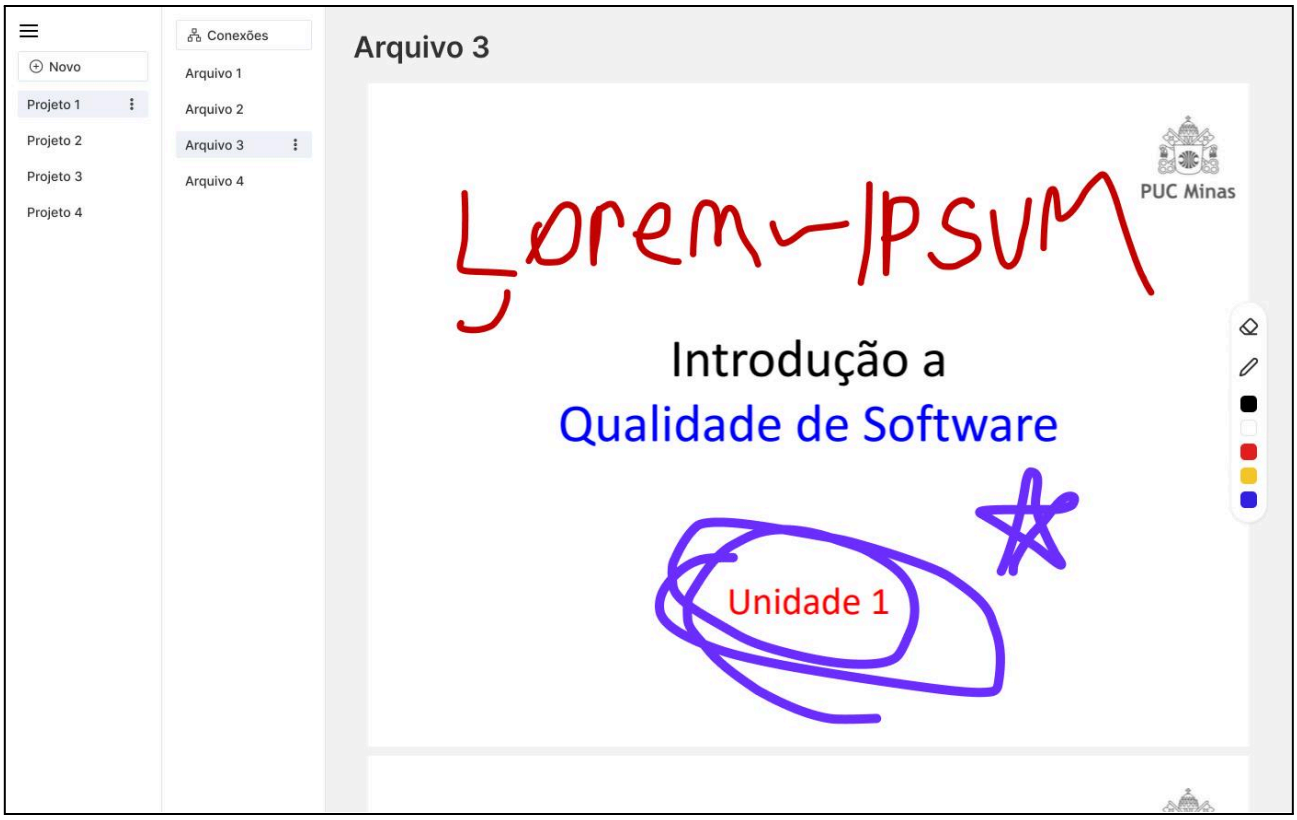

Figura 21. Tela de anotação em pdf

A Figura 21 exibe uma das telas centrais do Cognita: o leitor de PDF com funcionalidades de anotação. O painel mais à esquerda permite a navegação entre os diferentes "Projetos" do usuário. O segundo painel lista os arquivos contidos no projeto selecionado. A área principal, à direita, é dedicada à visualização do documento, onde o usuário pode realizar anotações manuscritas diretamente sobre o conteúdo com uma caneta *stylus*, como exemplificado pelos traços e destaques na imagem.

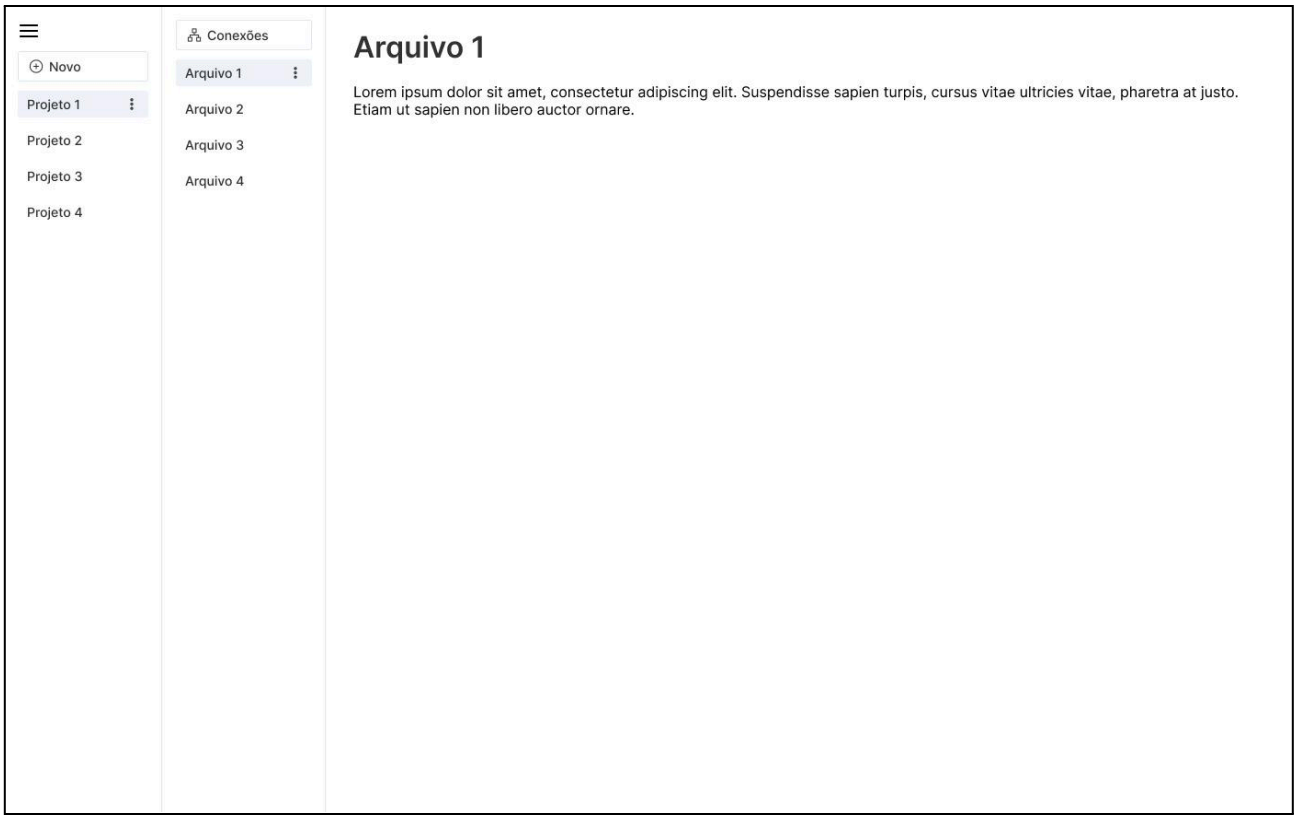

Figura 22. Tela de anotação digitada

A Figura 22 mostra a interface para uma nota simples contendo texto digitado. Esta tela demonstra a capacidade do sistema de suportar anotações textuais tradicionais.

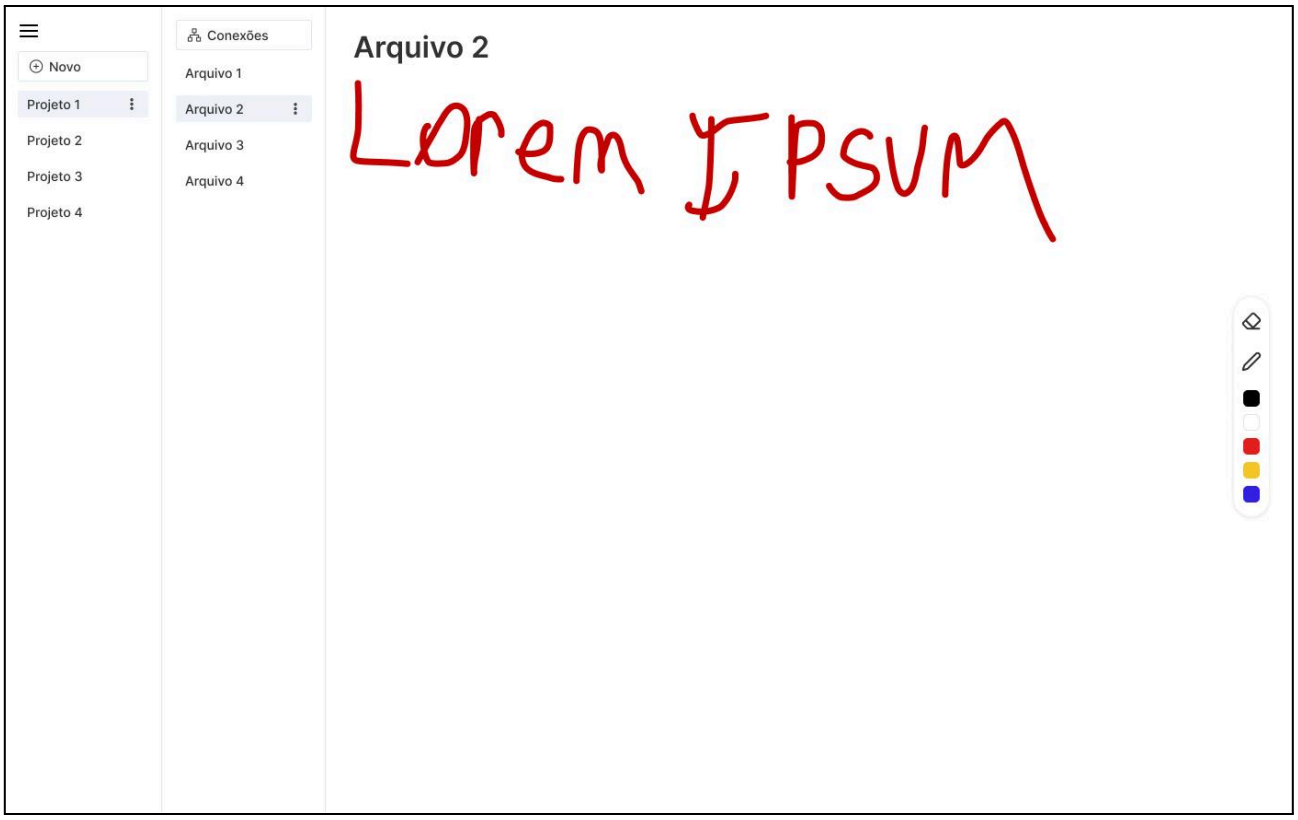

Figura 23. Tela de anotação manuscrita

Complementando a funcionalidade anterior, a Figura 23 ilustra a criação de uma nota com conteúdo puramente manuscrito. Esta funcionalidade é essencial para a proposta de valor do sistema, permitindo que o usuário capture ideias de forma livre e natural.

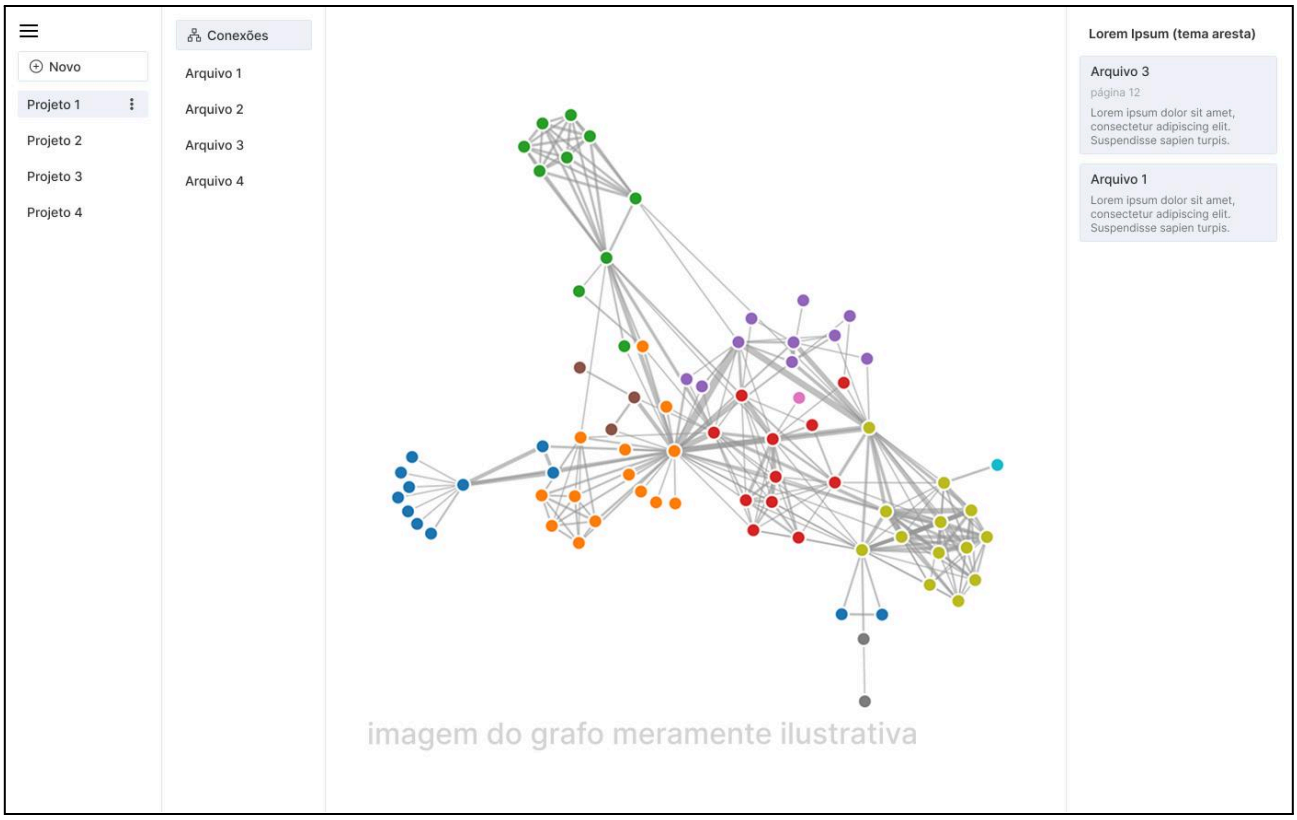

Figura 24. Tela de conexões (grafo)

A Figura 24 apresenta o *wireframe* da funcionalidade mais inovadora do Cognita. A área central exibe o grafo de conhecimento gerado automaticamente, com os nós representando conceitos e as arestas, suas relações. O painel esquerdo mantém a navegação entre projetos. Crucialmente, o painel direito, intitulado "Contexto", exibe informações detalhadas sobre o nó ou a conexão selecionada no grafo, como o trecho de texto original, o nome do arquivo e a página, cumprindo o requisito de fornecer contexto sem que o usuário precise sair da visualização.

5. Glossário e Modelos de Dados

Esta seção apresenta o glossário de termos do sistema e detalha o modelo de dados que define a estrutura de persistência da informação.

5.1 Glossário

O glossário a seguir define os principais atributos de dados com os quais o usuário interage, seja fornecendo-os como entrada ou visualizando-os como saída do sistema. O objetivo é padronizar a terminologia e esclarecer o formato e a descrição de cada campo reconhecido pelo usuário, conforme apresentado na Tabela 7.

Atributo	Formato	Descrição
Nome do Usuário	Texto	Nome completo do usuário, utilizado para identificação e personalização da conta. É uma entrada de dados durante o cadastro.
Email do Usuário	Email	Endereço de e-mail do usuário, que serve como identificador único para login e para comunicação do sistema.
Senha	Senha	Credencial de segurança definida pelo usuário para acesso à sua conta. O sistema armazena este dado de forma criptografada.
Nome do Projeto	Texto	Nome descritivo que o usuário atribui a um projeto para agrupar documentos e notas relacionados a um mesmo tema ou matéria.
Data de Criação do Projeto	Data	Data em que o projeto foi criado. É uma saída de dados gerada automaticamente pelo sistema e pode ser exibida na interface.
Arquivo PDF	Arquivo (.pdf)	O arquivo em formato PDF que o usuário seleciona em seu dispositivo para fazer upload para a sua biblioteca no Cognita.
Título do Documento	Texto	O nome do arquivo PDF que foi importado, exibido na lista de documentos de um projeto.
Status do Documento	Texto (Enum)	Indicação visual do estado de processamento de um documento (ex: "Processando", "Indexado", "Falha"). É uma saída de dados do sistema.
Conteúdo da Anotação	Texto / Vetor	O conteúdo da anotação criada pelo usuário. Pode ser um texto digitado, um destaque sobre um PDF ou os dados vetoriais de um traço manuscrito.
Cor da Anotação	Cor (Hex)	A cor escolhida pelo usuário para uma ferramenta de anotação, como um marca-texto ou uma caneta.
Espessura da Anotação	Numérico	A espessura do traço escolhida pelo usuário para uma ferramenta de anotação, como a caneta.
Termo de Busca	Texto	A palavra ou frase que o usuário digita no campo de busca para encontrar informações em sua base de conhecimento.
Lista de Resultados	Lista	A lista de trechos de texto e suas fontes (documento, página) que o sistema exibe como resultado de uma busca.

Rótulo do Conceito	Texto	O nome da entidade ou conceito (ex: "Semana de Arte Moderna") extraído pelo sistema e exibido como um "nó" na visualização do grafo.
Contexto da Conexão	Texto	O trecho de texto original de um documento ou anotação que gerou uma menção a um conceito, exibido ao interagir com o grafo.

Tabela 7. Glossário

5.2 Modelo de Dados – Property Graph Model

Dada a escolha arquitetural de utilizar um banco de dados de grafo nativo (Neo4j) para a persistência dos dados do servidor, um Diagrama de Entidade-Relacionamento (DER) tradicional não é o modelo mais adequado. Em seu lugar, utilizamos o *Property Graph Model*, que é o modelo conceitual nativo para bancos de dados de grafo e representa a estrutura de dados de forma mais fiel à sua implementação.

O *Property Graph Model* descreve os dados em termos de Nós, Relações e Propriedades:

- Nós (*Nodes*): Representam as entidades do sistema. Cada nó pode ter um ou mais Rótulos (*Labels*) que definem seu tipo ou papel no grafo.
- Relações (*Relationships*): Representam as conexões semânticas e direcionadas entre nós. Cada relação possui um único Tipo que descreve a natureza da conexão.
- Propriedades (*Properties*): São pares de chave-valor que armazenam os dados e podem ser atribuídos tanto aos Nós quanto às Relações.

A Figura 25 ilustra o *Property Graph Model* para o sistema Cognita, mostrando como as classes definidas no Diagrama de Classes são mapeadas para nós e como seus relacionamentos são representados.

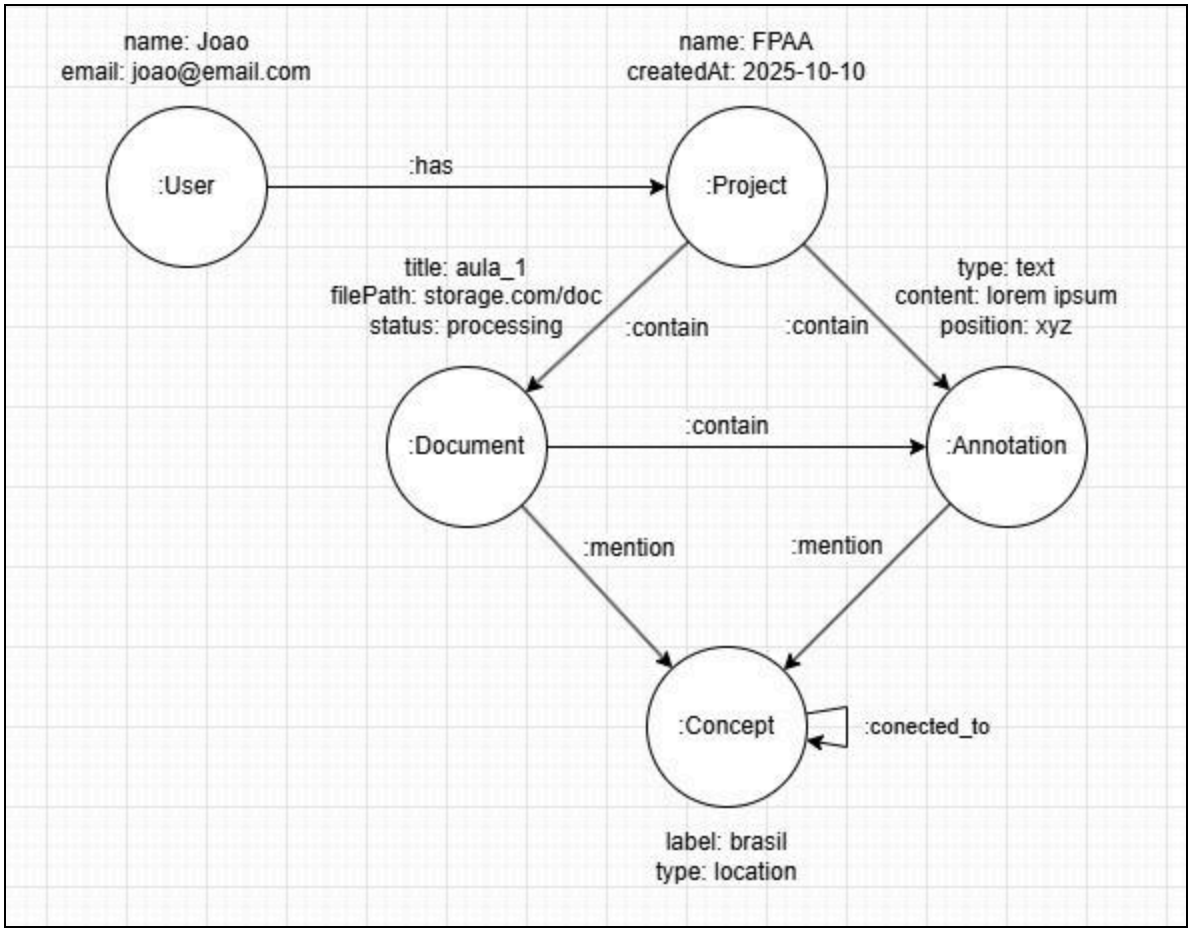

Figura 25. *Property Graph Model*

6. Casos de Teste

Esta seção descreve a estratégia de validação e verificação do *software* Cognita, dividida em testes de aceitação (focados no usuário e requisitos) e testes de integração (focados na arquitetura e comunicação entre componentes).

6.1 Plano de Teste de Aceitação

O objetivo dos testes de aceitação é validar se as funcionalidades desenvolvidas atendem às necessidades do usuário final conforme definido no Documento de Visão. A Tabela 8 apresenta os casos de teste mapeados para as principais necessidades do sistema.

Necessidade (Doc. Visão)	ID	Descrição do Caso de Teste	Entrada de Dados (Ação)	Saída Esperada / Comportamento

1. Gestão da Biblioteca	TA-01	*Upload* de arquivo PDF válido	Selecionar um arquivo qualquer com texto e confirmar *upload*.	O sistema deve exibir o arquivo na lista "Meus Documentos" com status "Processando".
	TA-02	Tentativa de *upload* de formato inválido	Selecionar arquivo "Imagem.png" ou "Texto.docx".	O sistema deve exibir mensagem de erro: "Formato não suportado. Apenas PDF." e não iniciar o *upload*.
	TA-03	Visualização de documento processado	Clicar em um documento com status "Pronto".	O documento deve abrir no visualizador pronto para leitura.
2. Anotação de Conteúdo	TA-04	Criação de anotação manuscrita	Usar a *stylus* para desenhar um círculo vermelho sobre um parágrafo.	O traço vermelho deve ser renderizado instantaneamente sobre o PDF e persistir após fechar e reabrir o documento.
	TA-05	Criação de nota de texto (adesiva)	Clicar e segurar em uma área vazia e digitar "Importante para a tese".	Um ícone de nota deve aparecer no local; ao clicar, o texto digitado deve ser exibido.
	TA-06	Remoção de anotação	Selecionar a ferramenta borracha e passar sobre um traço existente.	O traço deve ser removido da visualização e do banco de dados local.
3. Busca Global	TA-07	Busca por termo presente no texto	Digitar "Redes Neurais" na barra de busca global.	O sistema deve listar todos os documentos e anotações que contêm o termo, indicando a página da ocorrência.
	TA-08	Busca por termo em anotação manuscrita	Digitar uma palavra que foi escrita à mão (ex: "Revisar") em uma nota.	O sistema deve encontrar a anotação manuscrita (validando o HCR) e levar o usuário até ela.

	TA-09	Navegação para resultado da busca	Clicar no 2º item da lista de resultados.	O sistema deve abrir o documento correspondente e rolar automaticamente até a página onde o termo foi encontrado.
4. Conexão de Conhecimento (Grafo)	TA-10	Visualização do Grafo de Conhecimento	Acessar a aba "Grafo" de um projeto.	O sistema deve renderizar uma rede de nós (conceitos) e arestas (relações) interativos.
	TA-11	Exploração de Conexão	Clicar na linha (aresta) que liga "IA" e "Ética".	Um painel lateral deve abrir exibindo o trecho original do texto que fundamenta essa conexão ("Contexto").
	TA-12	Filtragem do Grafo	Selecionar o filtro "Apenas Pessoas" na visualização.	O grafo deve se atualizar para mostrar apenas nós classificados como entidades do tipo "Pessoa".

Tabela 8. Casos de Teste de Aceitação

6.2 Plano de Teste de Integração

Os testes de integração visam garantir que os subsistemas e serviços externos do Cognita comuniquem-se corretamente. Dada a arquitetura distribuída, o foco é validar as interfaces entre o Cliente, o Servidor e os Serviços de Nuvem. A Tabela 9 apresenta os casos de teste mapeados para as principais necessidades do sistema.

Estratégia de Teste: Será utilizada uma estratégia Incremental Ascendente (*Bottom-Up*) focada em serviços:

- Isolamento: Testes automatizados utilizarão *Test Containers* para subir instâncias reais e descartáveis do banco de dados Neo4j, garantindo que as *queries* Cypher funcionem em um ambiente controlado.
- *Mocks*: Para serviços pagos ou lentos (Google Cloud Vision e Storage), serão utilizados *Mocks* (simuladores) nos ambientes de desenvolvimento e CI/CD para validar a lógica de tratamento de respostas sem gerar custos.
- Integração Cliente-Servidor: Testes de API (contrato) validarão se os *endpoints* REST retornam os dados no formato JSON esperado pelo React Native.

Interface	Componentes Envolvidos	ID Teste	Cenário de Teste	Critério de Sucesso
REST API	Cliente (*Mobile*) e Servidor (*Backend*)	TI-01	Sincronização de Anotações: Cliente envia pacote de novas anotações *offline* para o servidor.	O servidor deve receber o JSON, retornar status 200 OK e os dados devem aparecer em uma consulta subsequente ao *backend*.
Storage API	Servidor e Google Cloud Storage	TI-02	*Upload* de Arquivo: O serviço de biblioteca envia um *stream* de *bytes* do PDF.	O serviço deve receber um URL público/assinado do Google Storage e o arquivo deve estar acessível via esse *link*.
Bolt Protocol	Servidor e Neo4j AuraDB	TI-03	Persistência de Grafo: O processador de conteúdo tenta salvar 50 nós e 100 relações de uma vez.	A transação deve ser commitada no Neo4j e uma *query* de contagem deve retornar os números exatos inseridos.
Vision API	Servidor e Google Cloud Vision	TI-04	Fluxo de OCR: O servidor envia uma imagem para a API de Visão (simulada via Mock).	O sistema deve tratar corretamente a resposta JSON da API, extraindo o bloco de texto ("fullTextAnnotation") e ignorando metadados irrelevantes.

Tabela 9. Casos de Teste de Integração

6.3 Avaliação dos Modelos de IA e Mitigação de Alucinações

Dada a natureza probabilística dos modelos de Processamento de Linguagem Natural (PLN) utilizados para a construção do grafo de conhecimento (NER e Extração de Relações), é necessário estabelecer métricas específicas para avaliar a qualidade das informações geradas e mitigar o risco de "alucinações" (geração de informações falsas ou não presentes no texto fonte).

Para minimizar a ocorrência de relações inventadas pelo modelo, será adotada a seguinte estratégias técnica:

- *Prompt Engineering* Restritivo: Os *prompts* enviados ao LLM para a extração de relações incluirão instruções explícitas de "*Grounding*" (ancoragem), exigindo que o modelo cite o trecho exato do texto original que justifica a conexão, reduzindo a "criatividade" do modelo.

A qualidade do resultado da IA não será medida apenas pela execução bem-sucedida do código, mas pela precisão semântica do grafo gerado. Para isso, será realizado um teste de validação manual comparativo:

- Conjunto de Validação (*Gold Standard*): Será selecionada uma amostra de 5 documentos acadêmicos de domínios variados. Estes documentos serão anotados manualmente por humanos, definindo quais entidades e relações "ideais" deveriam ser extraídas.

7. Cronograma e Processo de Implementação

Esta seção detalha o plano de gerenciamento do projeto, incluindo o cronograma de desenvolvimento e os processos que serão adotados para a implantação e entrega do sistema Cognita.

7.1 Cronograma do Projeto

O desenvolvimento do sistema será gerenciado utilizando uma abordagem ágil, baseada em ciclos de *sprint* com duração de duas semanas (quinzenais). O cronograma total de desenvolvimento está planejado para ocorrer ao longo de aproximadamente quatro meses, com início em fevereiro de 2026 e conclusão na primeira semana de junho de 2026, totalizando 9 *sprints* de desenvolvimento.

O foco de cada *sprint* está alinhado com a entrega de valor incremental, priorizando a construção das funcionalidades centrais da arquitetura (*offline-first*, *pipeline* de processamento) antes das funcionalidades de IA mais complexas. A Tabela 10 detalha o plano de entregas para cada *sprint*.

Sprint	Período (2026)	Foco Principal	Entregáveis Chave
1	01/Fev – 15/Fev	Configuração e Arquitetura Base	- Configuração dos ambientes (Local, Servidor). - Setup do projeto React Native. - Setup da API do Servidor e do banco de dados Neo4j.
2	16/Fev – 28/Mar	Modelo de Dados e Autenticação	- Implementação do modelo no Neo4j. - Implementação do backend de autenticação (Cadastro, Login). - Telas de Login e Cadastro no cliente.
3	01/Mar – 15/Mar	Cliente *Offline-First* e Gestão de Projetos	- Implementação do banco de dados local. - Telas de gerenciamento de Projetos. - Sincronização de usuários e projetos.

4	16/Mar – 31/Mar	*Pipeline* de *Upload* e Processamento	- Implementação do *upload* de arquivos. - Integração com o Google Cloud Storage. - *Backend*: Gatilho para o *pipeline* de processamento.
5	01/Abr – 15/Abr	Anotação Manuscrita	- Implementação da tela de anotação com React Native Skia. - Captura de traço da stylus e salvamento local. - Sincronização das anotações.
6	16/Abr – 30/Abr	*Pipeline* de IA (Parte 1): OCR/HCR	- Integração do *backend* com a API Google Cloud Vision. - *Backend*: Processamento de PDFs e imagens de notas. - Armazenamento do texto extraído.
7	1/Mai – 15/Mai	Pipeline de IA (Parte 2): NER e RE	- Integração dos modelos Hugging Face (NER e LLM para RE). - *Backend*: Extração de entidades e relações do texto processado.
8	16/Mai – 31/Mai	Persistência e Visualização do Grafo	- Armazenamento dos nós e arestas no Neo4j. - Implementação da tela de visualização do grafo. - Validação visual do *pipeline* de IA de ponta a ponta.
9	1/Jun – 15/Jun	Busca Global e Teste de Aceite	- Implementação da funcionalidade de Busca Global. - Execução dos casos de testes. - Correção de *bugs* e refatoração final.

Tabela 10. Cronograma

7.2 Processo de Implantação

O processo de implantação seguirá as práticas de integração e entrega contínua (CI/CD) para o ambiente de servidor, enquanto a aplicação cliente será distribuída para fins de teste e demonstração.

Metodologia de Desenvolvimento: O projeto adotará a metodologia ágil Scrum, alinhada com os *sprints* quinzenais definidos no cronograma. Cada *sprint* incluirá as cerimônias de planejamento (início) e uma revisão/retrospectiva (final), permitindo a adaptação contínua do plano de trabalho.

Processo de Implantação (Servidor): O *backend* será configurado com um *pipeline* de CI/CD. Cada *commit* no repositório principal acionará um *script* (GitHub Actions) que automaticamente testará, construirá a imagem de contêiner do API Servidor e a implantará no ambiente de Desenvolvimento/Testes na nuvem.

Testes Automatizados: A garantia de qualidade segue a pirâmide de testes, utilizando ferramentas específicas para cada camada da arquitetura:

- Testes Unitários (*Backend*): Utilização do *framework* Pytest para validar a lógica de negócios, endpoints da API e funções de processamento de texto em Python.
- Testes Unitários (*Frontend*): Utilização do *framework* Jest combinado com a React Native Testing Library para validar a renderização de componentes e a lógica de interface do aplicativo móvel.
- Testes de Integração: *Scripts* automatizados (via Pytest) que validam a comunicação entre a API, o banco de dados Neo4j e os serviços externos (Google Vision), garantindo a integridade dos contratos de dados.

Artefatos a serem produzidos:

- Código-fonte: Inclui *front-end* mobile e back-end com testes automatizados.
- Documentação: README detalhado com instruções de instalação e uso.