

EDUARDO LOPES FONSECA GONZALES

GABRIEL MACHADO DOS SANTOS

GUILHERME ANGELO SILVA

JOÃO PEREIRA NETO

SGED

Sistema Gerenciador de Documentos

Projeto Integrador

EDUARDO LOPES FONSECA GONZALES

GABRIEL MACHADO DOS SANTOS

GUILHERME ANGELO SILVA

JOÃO PEREIRA NETO

SGED

Sistema Gerenciador de Documentos

Projeto Integrador apresentado à Faculdade de Tecnologia Professor José Camargo – Fatec Jales, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Jales

2024

EDUARDO LOPES FONSECA GONZALES

GABRIEL MACHADO DOS SANTOS

GUILHERME ANGELO SILVA

JOÃO PEREIRA NETO

SGED

Sistema Gerenciador de Documentos

Projeto Integrador apresentado à Faculdade de Tecnologia Professor José Camargo – Fatec Jales, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Banca Examinadora:

Prof. (Orientador)
Fatec Jales

Prof. (Orientador)
Fatec Jales

Prof. (Orientador)
Fatec Jales

Jales, ____ de _____ de 2024.

RESUMO

Este artigo apresenta o desenvolvimento do Sistema de Gerenciamento de Documentos (SGED), uma solução de *Enterprise Content Management* (ECM) implementada para atender às necessidades da Secretaria de Obras de uma prefeitura. O SGED visa modernizar a administração dos documentos gerados nos processos de construção e reforma, como concessão de alvarás e demais serviços relacionados a edificações. A proposta surgiu da necessidade de superar limitações do sistema legado, que não atendia mais às demandas atuais de eficiência e segurança. O SGED foi desenvolvido com uma arquitetura orientada a objetos, organizada para oferecer flexibilidade e integração entre diferentes conjuntos de dados, garantindo a consistência e segurança das informações. A aplicação utiliza uma interface intuitiva que facilita a navegação e otimiza o tempo de operação, minimizando erros e perda de dados. O sistema incorpora uma arquitetura em camadas com operações CRUD (*Create, Read, Update, Delete*) e garante segurança adicional através do uso de tokens e criptografia. A metodologia de desenvolvimento incluiu análise de requisitos e modelagem em UML, com o uso de software livre. A solução proposta oferece uma ferramenta robusta para o gerenciamento de documentos, contribuindo para a eficiência administrativa e para a transparência nos processos de gestão pública.

Palavras-chave: Gestão documental; Secretaria de Obras; Administração Pública; ECM; Desenvolvimento de Software.

ABSTRACT

This article presents the development of the Document Management System (SGED), an Enterprise Content Management (ECM) solution implemented to meet the needs of a municipality's Department of Public Works. SGED aims to modernize the administration of documents generated in construction and renovation processes, such as permit issuance and other services related to buildings. The proposal emerged from the need to overcome limitations of the legacy system, which no longer met current demands for efficiency and security. SGED was developed with an object-oriented architecture, designed to provide flexibility and integration across different data sets, ensuring data consistency and security. The application utilizes an intuitive interface that facilitates navigation and optimizes operational time, minimizing errors and data loss. The system incorporates a layered architecture with CRUD operations (Create, Read, Update, Delete) and provides additional security through the use of tokens and encryption. The development methodology included requirements analysis and modeling in UML, using open-source software. The proposed solution offers a robust tool for document management, contributing to administrative efficiency and transparency in public management processes.

Keywords: *Document Management; Department of Public Works; Public Administration; ECM; Software Development.*

LISTA DE FIGURAS

Figura 1 – Tela de Listagem de Documentos do Sistema GetQual	16
Figura 2 – Sistema Legado: Tela Principal.	16
Figura 3 – Sistema Legado: (a) Menu principal do software, (b) Cadastro das aprovações nos processos e (c) Barra com dados na tela de pesquisa.	17
Figura 4 – Diagrama de Classes	24
Figura 5 – Diagrama de Atores	39
Figura 6 – Diagrama de Caso de Uso Geral: Visão do Administrador	50
Figura 7 – Diagrama de Caso de Uso Específico: Administrador – Cadastrar Processo	51
Figura 8 – Diagrama de Caso de Uso Específico: Administrador – Alterar Documento Processo	53
Figura 9 – Diagrama de Sequência - Ator Administrador: Fluxo do cadastro de Estado	56
Figura 10 – Diagrama de Sequência - Ator Administrador: Fluxo do excluir de Estado.....	57
Figura 11 – Diagrama de Máquina de Estado - Status do Processo	58
Figura 12 – Cenário Engenheira de Obras da Prefeitura	60
Figura 13 – Cenário Engenheira de Obras da Prefeitura	60
Figura 14 – Persona Engenheira Civil.....	61
Figura 15 – Persona Engenheiro de Campo	62
Figura 16 – <i>Wireframe</i> da Tela de Login	63
Figura 17 – <i>Wireframe</i> da Tela Inicial do Sistema.....	64
Figura 18 – <i>Wireframe</i> Tela de Cadastro de Estado.....	65
Figura 19 – <i>Wireframe</i> Tela de Cadastro de Cidade	66
Figura 20 – Protótipo de Tela de Login.....	68
Figura 21 – Protótipo de Tela Inicial.....	69
Figura 22 – Protótipo da Tela de Cadastro de Estado	70
Figura 23 – Protótipo da Tela de Cadastro de Cidade.....	71
Figura 24 – Mapeamento do Objeto Relacional	73
Figura 25 – Tela Inicial do sistema	87
Figura 26 – Diagrama de Implantação	89

LISTA DE QUADROS

Quadro 1 – Requisitos Funcionais do Sistema	18
Quadro 2 – Requisitos Não Funcionais do Sistema	21
Quadro 3 – Descrição da Interface Pessoa	27
Quadro 4 – Descrição da Classe Município.....	27
Quadro 5 – Descrição da Classe Engenheiro.....	27
Quadro 6 – Descrição da Classe Fiscal	28
Quadro 7 – Descrição da Classe TipoUsuario.....	28
Quadro 8 – Descrição da Classe Usuário	28
Quadro 9 - Descrição da Classe Sessão.....	29
Quadro 10 - Descrição da Classe Auditoria	29
Quadro 11 – Descrição da Classe Estado	30
Quadro 12 – Descrição da Classe Cidade.....	30
Quadro 13 – Descrição da Classe Bairro.....	30
Quadro 14 – Descrição da Classe TipoLogradouro.....	31
Quadro 15 – Descrição da Classe Logradouro	31
Quadro 16 – Descrição da Classe Imóvel.....	31
Quadro 17 – Descrição da Classe Topografia	32
Quadro 18 – Descrição da Classe TipoUso	32
Quadro 19 – Descrição da Classe OcupaçãoAtual	33
Quadro 20 – Descrição da Classe TipoInfraestrutura.....	33
Quadro 21 – Descrição da Classe Infraestrutura	33
Quadro 22 – Descrição da Classe Associativa Instalação	34
Quadro 23 – Descrição do Enum StatusProcesso.....	34
Quadro 24 – Descrição da Classe Processo.....	34
Quadro 25 – Descrição da Classe TipoProcesso	35
Quadro 26 – Descrição da Interface Posição.....	35
Quadro 27 – Descrição da Classe Etapa.....	36
Quadro 28 – Descrição da Classe TipoDocumento.....	36
Quadro 29 – Descrição da Classe Associativa TipoDocumentoEtapa.....	36
Quadro 30 – Descrição do Enum StatusDocumentoProcesso	36
Quadro 31 – Descrição da Classe DocumentoProcesso	37
Quadro 32 – Descrição dos Atores	39
Quadro 33 – Mensagens de saída	40

Quadro 34 – Lista de Casos de Uso: Ações do Administrador	41
Quadro 35 – Lista de Casos de Uso: Ações do Secratário	45
Quadro 36 – Lista de Casos de Uso: Ações do Estagiário	48
Quadro 37 – Lista de Casos de Uso: Ações do Sistema.....	48
Quadro 38 – Fluxo do Caso de Uso: Administrador – Cadastrar Processo.....	51
Quadro 39 – Fluxo do Caso de Uso: Adminsitrador – Alterar Documento Processo	53
Quadro 40 – Tabela Auditoria	74
Quadro 41 – Tabela Bairro	74
Quadro 42 – Tabela Cidade	74
Quadro 43 – Tabela DocumentoProcesso	75
Quadro 44 – Tabela Engenheiro	75
Quadro 45 – Tabela Estado	76
Quadro 46 – Tabela Etapa	76
Quadro 47 – Tabela Fiscal.....	76
Quadro 48 – Tabela Imovel.....	77
Quadro 49 – Tabela Infraestrutura.....	78
Quadro 50 – Tabela Instalacao	78
Quadro 51 – Tabela Logradouro	78
Quadro 52 – Tabela Municipe	79
Quadro 53 – Tabela OcupacaoAtual	79
Quadro 54 – Tabela Processo	79
Quadro 55 – Tabela Sessao	80
Quadro 56 – Tabela TipoDocumento	80
Quadro 57 – Tabela TipoDocumentoEtapa	81
Quadro 58 – Tabela TipoInfraestrutura	81
Quadro 59 – Tabela TipoLogradouro.....	81
Quadro 60 – Tabela TipoProcesso	81
Quadro 61 – Tabela TipoUso	82
Quadro 62 – Tabela TipoUsuario	82
Quadro 63 – Tabela Topografia.....	82
Quadro 64 – Tabela Usuario.....	82

SUMÁRIO

1 INTRODUÇÃO.....	11
2 LEVANTAMENTO DE REQUISITOS DE SOFTWARE.....	14
2.1 DESCRIÇÃO DOS OBJETIVOS DO SISTEMA.....	14
2.2 DESCRIÇÃO DO SISTEMA ATUAL.....	14
2.3 ANÁLISE DE SISTEMAS EXISTENTES.....	15
2.4 DESCRIÇÃO DOS PRINCIPAIS PROBLEMAS.....	17
2.5 DESCRIÇÃO DOS REQUISITOS FUNCIONAIS.....	18
2.6 DESCRIÇÃO DOS REQUISITOS NÃO FUNCIONAIS.....	20
3 VISÃO DE CASO DE USO – UML.....	22
3.1 DIAGRAMA DE CLASSES.....	22
3.2 DICIONÁRIO DE CLASSES.....	24
3.3 DEFINIÇÃO DOS ATORES.....	38
3.4 LISTA DE CASOS DE USO.....	40
3.4. DIAGRAMA DE CASOS DE USO.....	49
3.5. DIAGRAMA DE CASOS DE USO INDIVIDUAIS.....	50
3.6. DIAGRAMA DE SEQUÊNCIA.....	55
3.6.1 DIAGRAMA DE SEQUÊNCIA PARA CADASTRO DE ESTADO.....	55
3.6.2 DIAGRAMA DE SEQUÊNCIA PARA EXCLUSÃO DE ESTADO.....	56
3.7. DIAGRAMA DE MÁQUINA DE ESTADO.....	58
4 DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX).....	59
4.1 DESCRIÇÃO DE CENÁRIO.....	59
4.2 DESCRIÇÃO DE PERSONAS.....	61
4.3 ESBOÇOS DE TELA (WIREFRAMES).....	62
4.4 PROTÓTIPOS DE TELA.....	66
5 BANCO DE DADOS.....	72
5.1 MODELO ENTIDADE RELACIONAMENTO.....	72
5.2 SCRIPT DAS TABELAS.....	73
5.3 MAPEAMENTO OBJETO RELACIONAL – ORM.....	83
6 ARQUITETURA DE SOFTWARE.....	84
6.1 ARQUITETURA DE DESENVOLVIMENTO.....	84
6.1.1 BACK-END.....	85
6.1.2 FRONT-END - WEB.....	85
6.2 SEGURANÇA DA INFORMAÇÃO.....	87
6.3 IMPLANTAÇÃO.....	89
7 CONCLUSÃO.....	92
8 REFERÊNCIAS.....	93

1 INTRODUÇÃO

A geração de documentos relacionados a obras é essencial em qualquer processo construtivo, configurando-se como registros oficiais das etapas, decisões, normas e regulamentações aplicáveis a cada fase da construção. Esses documentos não apenas atestam a conformidade da obra com as exigências legais e técnicas, como também asseguram a transparência e a rastreabilidade de todos os procedimentos realizados ao longo do projeto.

“O licenciamento para obras é imprescindível ao construir um imóvel. Sem ele, o dono do imóvel está sujeito a notificação, ao embargo da obra, aplicação de multa e, até mesmo, à demolição da construção. Para licenciá-la, é preciso obedecer ao Código de Obras. Quando um projeto para construção de um imóvel é aprovado pela prefeitura, significa que o mesmo atendeu à legislação e a construção pode ser iniciada após a liberação do alvará, documento autorizando o início dos serviços”, informa a titular da Seplu, Lenise Ferreira (apud Costa, 2020).

A documentação é indispensável para assegurar a preservação dos padrões de qualidade em todos os aspectos da obra, sendo fundamental para a segurança pública e a durabilidade das infraestruturas construídas. Além disso, sua correta gestão é crucial para evitar multas, atrasos no cronograma do projeto e possíveis litígios legais, promovendo, assim, a conformidade com as normas e o sucesso do empreendimento.

O processo de gestão de autorizações de obras em uma secretaria de prefeitura envolve várias etapas essenciais, desde a solicitação e análise do projeto até a emissão de documentos que garantem a conformidade da obra com a legislação vigente. Cada município possui seu próprio Código de Obras, que define os requisitos técnicos e procedimentos para a aprovação de projetos e execução de obras. Esses códigos estabelecem diretrizes sobre zoneamento, uso do solo, padrões construtivos e segurança das edificações.

A solicitação de licenciamento que é realizado pelo responsável pela obra (proprietário ou construtor) apresenta o projeto arquitetônico junto à secretaria de urbanismo ou planejamento. Nessa etapa, são verificados documentos como matrícula do imóvel e comprovação de propriedade, além de informações sobre a área e a finalidade da obra.

A análise do projeto e o processo no qual os técnicos e engenheiros da prefeitura analisam o projeto para garantir que ele está de acordo com o Plano Diretor, o Código de Obras e as legislações ambientais e urbanísticas locais. Nessa fase, verificam-se os parâmetros urbanísticos, recuos, altura da edificação e impactos ambientais.

A seguir ocorre a emissão de alvará de construção caso o projeto esteja em conformidade com as normas, a secretaria emite o alvará de construção, que autoriza o início das obras. Esse documento especifica o prazo de validade e as condições sob as quais a obra

pode ser realizada, podendo incluir requisitos adicionais, como a adoção de medidas de segurança.

Durante a execução da obra, ocorre a fase de acompanhamento e fiscalização, onde os fiscais da prefeitura realizam inspeções para verificar se a construção segue o projeto aprovado e cumpre com as exigências legais. Qualquer desvio significativo pode resultar em notificações, multas ou até mesmo embargos à obra.

Após a conclusão da obra, o responsável deve solicitar a emissão do "Habite-se", documento que comprova que a edificação está pronta e apta para ser habitada ou utilizada. A solicitação envolve a apresentação de laudos e documentos de vistoria que confirmam o cumprimento das exigências. Técnicos realizam uma vistoria final para verificar que a obra respeita o projeto aprovado e as normas de segurança e habitabilidade. São avaliados aspectos como segurança elétrica, acessibilidade, sistemas de segurança contra incêndio e regularidade estrutural.

Com a aprovação final, a secretaria emite o documento de "habite-se", que permite o uso efetivo do imóvel. Este é um requisito essencial para que o imóvel seja registrado em cartório, possibilitando venda, aluguel ou ocupação formal. Magalhães, Melo e Bandeira (2018) ressaltam que o controle e o planejamento no processo de construção são fundamentais para o sucesso do projeto a ser realizado.

Esse processo visa assegurar que as construções atendam aos requisitos técnicos e legais, promovendo a segurança, a organização urbana e o uso responsável do espaço. Cada uma das etapas do projeto pode demandar a geração de diferentes documentos, que necessitam ser analisados, aprovados e devidamente arquivados. Dessa forma, a gestão eficiente desses documentos, assim como a atualização contínua dos dados, torna-se essencial para a conformidade e a qualidade do processo construtivo.

Segundo Mobuss (2018), "A gestão de documentos que conta com ferramentas inovadoras assegura a disseminação de dados rápida, eficiente e uniforme." Dessa forma, uma gestão adequada possibilita que as informações sejam prontamente localizadas quando necessário, garantindo o controle de documentos sensíveis e a proteção de informações confidenciais contra acessos não autorizados. Muitas organizações estão sujeitas a rigorosos requisitos quanto à gestão documental, e uma gestão eficiente assegura conformidade com as regulamentações, reduzindo riscos legais e possíveis penalidades.

A incorporação de tecnologias modernas torna a gestão documental mais eficiente e segura. A digitalização de documentos físicos e sua conversão em formatos eletrônicos reduzem a dependência de arquivos em papel, facilitando tanto o armazenamento quanto a recuperação

das informações. Além disso, essas tecnologias permitem a implementação de medidas de segurança eficazes, como a criptografia de dados e o controle de acesso baseado em funções específicas no sistema, garantindo que apenas pessoas autorizadas possam acessar informações confidenciais.

Este projeto surge da necessidade da secretaria de obras de gerenciar de forma eficiente os documentos relacionados aos processos de construção dos municípios. Atualmente, a prefeitura desta cidade utiliza um sistema de gestão pública, mas ele não dispõe de um módulo específico para o gerenciamento de processos e da documentação gerada. Assim, quando é necessário localizar algum processo, a busca deve ser realizada manualmente nos arquivos físicos da secretaria. A secretaria de obras conta com um software legado, desenvolvido em parceria com alunos para agilizar a gestão e a busca desses processos; no entanto, devido ao tempo de uso e à obsolescência tecnológica, o sistema não atende mais às demandas atuais da secretaria.

Nesse contexto, este projeto propõe o desenvolvimento de um software voltado para o gerenciamento de obras e dos documentos gerados ao longo desse processo. O sistema proporcionará uma funcionalidade de pesquisa avançada, visando otimizar o tempo de resposta e reduzir o esforço necessário para localizar documentos. Além disso, garantirá o controle de acesso e a segurança dos arquivos, caracterizando-se como um sistema de informação robusto e confiável.

2 LEVANTAMENTO DE REQUISITOS DE SOFTWARE

Os requisitos é a descrição de como o *software* irá se comportar de acordo com informações de *hardware* e limites operacionais. De acordo com (Sommerville, 2011), “o termo requisito não é usado pela indústria de software de maneira consistente. [...] um requisito é simplesmente uma declaração abstrata de alto nível de um serviço que o sistema deve fornecer”.

O levantamento de requisitos é uma etapa crucial no processo de desenvolvimento de sistemas e projetos, convergindo as necessidades do usuário na solução que será desenvolvida. O processo de coleta de informações para garantir a exatidão do sistema é feito por meio de entrevistas com o futuro usuário do sistema (Guedes, 2011).

Comunicação é um grande problema encontrado na fase de levantamento de requisitos, se tornando um desafio a compreensão dos conceitos, abstrações e complexos que caracterizam as necessidades do usuário. Na fase de levantamento de requisitos que são as condições necessárias para que o sistema responda adequadamente às ações do cliente, os não funcionais se tornam contenções, validações e consistências sobre os requisitos funcionais (Guedes, 2011).

2.1 DESCRIÇÃO DOS OBJETIVOS DO SISTEMA

O sistema a ser desenvolvido tem como objetivo principal simplificar e aprimorar o acesso e busca aos arquivos e documentos relacionados aos projetos de obras da Prefeitura de Jales, centralizando e organizando de forma mais estruturada.

2.2 DESCRIÇÃO DO SISTEMA ATUAL

O sistema atual utilizado pela Secretaria de Obras revela-se inadequado para suprir as crescentes necessidades e demandas cotidianas do órgão, principalmente devido à sua limitação quanto à capacidade de armazenamento de informações. Esta escassez de espaço para registros se traduz em obstáculos consideráveis na tarefa de localizar e gerenciar os dados essenciais.

Essa deficiência prejudica significativamente a eficiência administrativa e a capacidade de busca e resposta da Secretaria de Obras aos arquivos e documentos, sendo assim urgente a busca por uma solução mais robusta e eficaz para gerenciamento de informações.

Com o sistema que está sendo desenvolvido será possível suprir todas as necessidades que está presente na Secretaria de Obras em questão a tudo, como por exemplo visualizar dados de um documento, finalizar os processos desses documentos e armazená-los.

2.3 ANÁLISE DE SISTEMAS EXISTENTES

Um sistema de gerenciamento de documentos é uma ferramenta digital projetada para simplificar o armazenamento, a organização, a supervisão e o acesso centralizado aos documentos de uma organização. Esse sistema assegura que todos os documentos relevantes estejam organizados de maneira estruturada, proporcionando acesso rápido e eficiente às informações. Além disso, a adoção desse sistema reduz o uso excessivo de papel e permite a integração com diversas plataformas, promovendo uma gestão documental mais sustentável e tecnológica (TOTVS, 2024).

O principal objetivo de um sistema de gerenciamento de documentos é aprimorar o fluxo de trabalho, aumentando a eficiência organizacional. Esse sistema automatiza tarefas manuais relacionadas à criação, edição, compartilhamento e armazenamento de documentos. Entre suas funcionalidades destacam-se a pesquisa rápida, que permite localizar documentos facilmente por meio de palavras-chave; o armazenamento centralizado, que reúne todos os arquivos em um único local acessível; as permissões de acesso, que definem quem pode visualizar ou modificar documentos específicos; a automação de processos, que facilita a gestão de aprovações e revisões; e o controle de versões, que permite o rastreamento e o gerenciamento do histórico de alterações nos documentos (TOTVS, 2024).

Para fundamentar a gestão do processo de obras e analisar os recursos disponíveis em soluções de software do mercado, foram pesquisados sistemas que atendam a essas necessidades. Dentre as soluções, o GestQual (2024) se destaca por oferecer uma ferramenta completa para o gerenciamento documental, permitindo o cadastro de documentos e o registro de suas respectivas etapas. O sistema também conta com controle de acesso personalizado, ajustando permissões de acordo com as responsabilidades de cada cargo, além de incluir uma auditoria que monitora as ações realizadas. A Figura 1 apresenta a tela de listagem de documentos no sistema, onde é possível inserir novos documentos, filtrar as informações exibidas e exportar a lista completa como relatório.

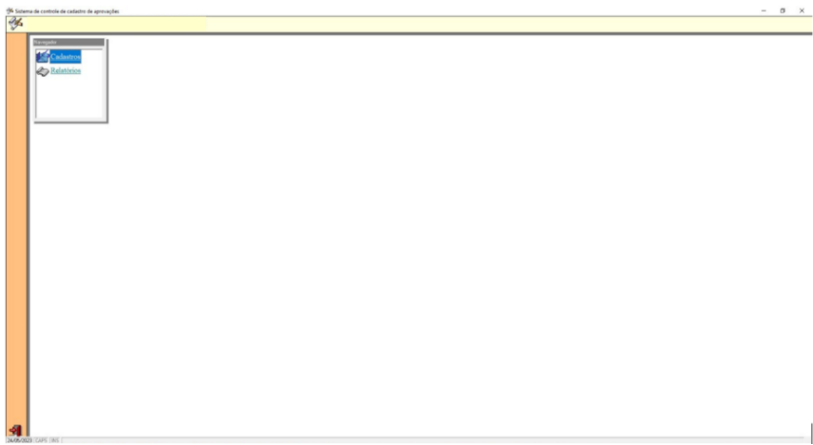
Figura 1 – Tela de Listagem de Documentos do Sistema GetQual

CODIFICAÇÃO	Ícone	NOME	VERSÃO	RESPONSÁVEL	VALIDADE	STATUS	Contagem	Ícone	Ícone	Ícone	Ícone
CTB-007-001		TESTE ADD	001	COORDENADOR DA GARANTIA DA QUALIDADE	22/12/2019	ATIVO	6				
Ctr-003-001		teste	001	COORDENADOR COLETAS	29/09/2020	APROVAÇÃO	0				
Ctr-004-000		pop teste	000	COORDENADOR COLETAS	29/09/2020	EM ELABORAÇÃO	0				
FT-002-001		teste ficha	001	COORDENADOR DA GARANTIA DA QUALIDADE		ATIVO	25				
FT-003-001		Documento Teste	001	COORDENADOR DA GARANTIA DA QUALIDADE	12/07/2017	ATIVO	31				
IT-001		COLETA DE AMOSTRAS	001	COORDENADOR DA GARANTIA DA QUALIDADE	15/09/2017	ATIVO	3				
ITA-001-001		teste	001	COORDENADOR COLETAS	31/08/2019	ATIVO	2				
POP-001		Procedimento de Coleta	001	COORDENADOR DA GARANTIA DA QUALIDADE	03/10/2020	ATIVO	18				
POP-001		PROCEDIMENTO DE COLETA	001	COORDENADOR DA GARANTIA DA QUALIDADE	13/09/2020	ATIVO	11				
POP-001		PROCEDIMENTO DE COLETA	002	COORDENADOR DA GARANTIA DA QUALIDADE	15/08/2020	ATIVO	1				

Fonte: GestQual (2024).

O software legado atualmente utilizado pela Secretaria de Obras foi analisado com o objetivo de identificar o fluxo e controle das informações de obras. A Figura 2 representa a interface principal dessa solução herdada, que possui restrições em relação às demandas atuais da secretaria. Este sistema já não responde adequadamente às necessidades operacionais e administrativas, destacando a necessidade de uma atualização ou substituição que esteja em sintonia com os processos contemporâneos e proporcione um apoio mais eficiente na administração de informações de obras.

Figura 2 – Sistema Legado: Tela Principal.



Fonte: Elaborado pelos Autores.

Na Figura 3(a), são exibidos os cadastros que integram a solução legada; na Figura 3(b), observa-se a tela de cadastro de aprovações, onde cada processo de obra civil do município

é registrado. Este software, no entanto, não realiza o gerenciamento de documentos, apenas armazena os dados do processo para futuras consultas. Por fim, a Figura 3(c) apresenta a tela de pesquisa de aprovações, permitindo consultas por atributos como bairro, quadra, lote e proprietário.

Figura 3 – Sistema Legado: (a) Menu principal do software, (b) Cadastro das aprovações nos processos e (c) Barra com dados na tela de pesquisa.



Fonte: Elaborado pelos Autores.

2.4 DESCRIÇÃO DOS PRINCIPAIS PROBLEMAS

A equipe do projeto identificou um problema crítico relacionado ao desenvolvimento do banco de dados, essencial para o sistema de gestão de documentos de aprovação de obras. O principal desafio é a necessidade de armazenar um grande volume de informações, incluindo dados sensíveis, que exigem uma abordagem cuidadosa e bem planejada. Garantir a integridade e a acessibilidade dessas informações é fundamental para o funcionamento eficaz do sistema, atendendo às expectativas dos usuários finais.

Além disso, a segurança do banco de dados foi destacada como uma prioridade indispensável. Por lidar com dados confidenciais, como documentos técnicos e informações de projetos de obras, é imprescindível adotar medidas robustas de proteção contra acessos não autorizados, perda ou exposição de informações. Assim, o planejamento e a implementação de mecanismos de segurança serão tratados como requisitos fundamentais durante todas as fases do desenvolvimento do sistema.

Para assegurar a eficiência e segurança do banco de dados, serão adotadas estratégias como a normalização das tabelas para evitar redundâncias, a aplicação de políticas de acesso restrito com base em níveis de permissão e o uso de criptografia para proteger os dados mais sensíveis. Além disso, será implementado um sistema de auditoria para monitorar alterações e acessos, garantindo rastreabilidade e conformidade com os padrões de segurança exigidos. Essas medidas visam proporcionar uma base sólida para o armazenamento e gerenciamento das informações essenciais ao sistema.

2.5 DESCRIÇÃO DOS REQUISITOS FUNCIONAIS

Requisitos funcionais são as declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também estabelecer explicitamente o que o sistema não deve fazer. (Sommerville, 2018).

Dessa forma, ao identificar e documentar de forma clara as funções principais do sistema, os requisitos funcionais fornecem uma base mensurável e testável para assegurar que o sistema atenda às expectativas dos usuários. No Quadro 1, estão descritas as principais funções identificadas, que devem ser revisadas continuamente ao longo do ciclo de vida do projeto para garantir a qualidade e o alinhamento do sistema com as necessidades reais.

Quadro 1 – Requisitos Funcionais do Sistema

	Requisitos Funcionais	Descrição
1	Cadastro de usuários	O sistema deve ser capaz de cadastrar usuários para utilização do sistema, desde atendente até administrador.
2	Autenticação de Usuários	O sistema deve verificar as credenciais (E-mail e senha) para realizar o login.
3	Controle de Acessos	O sistema deve verificar as credenciais apresentadas e apenas realizar o login se o usuário estiver cadastrado.
4	Alteração de usuários	O sistema deve ser capaz de realizar a alteração de dados de usuários se for necessário.
5	Listagem de usuários	O sistema deve listar os usuários para que o Administrador - possa manter o controle dos usuários.
6	Exclusão de usuários	O sistema deve ser capaz de excluir usuários via Administrador.
7	Desativação de usuários	O sistema deve ser capaz de desativar usuários via Administrador - dessa forma evitando que possam fazer login, preservando arquivos que dependem do mesmo.
8	Cadastro de Imóveis	O sistema deve permitir que os usuários autorizados cadastrem imóveis.
9	Alteração de Imóveis	O sistema deve permitir que os usuários autorizados alterem imóveis cadastrados.

10	Exclusão de imóveis	O sistema deve permitir que os usuários autorizados possam excluir imóveis do sistema
11	Desativação de imóveis	O sistema deve permitir que os usuários autorizados desativem imóveis, dessa forma não irá interagir no banco de dados apenas existindo para que arquivos dependentes do mesmo não sejam perdidos.
12	Cadastro de processos	O sistema deve permitir que os usuários autorizados façam o cadastro de processos.
13	Alteração de processos	O sistema deve permitir que os usuários autorizados façam alterações nos processos cadastrados.
14	Listagem de processos	O sistema deve listar os processos presentes nele.
15	Exclusão de processos	O sistema deve permitir que usuários autorizados façam a exclusão de processos
16	Desativação de processos	O sistema deve permitir a desativação de processos cadastrados por usuários autorizados, dessa forma arquivos dependentes não serão perdidos.
17	Cadastro de tipos de processo	O sistema deve permitir o cadastro do tipo do processo.
18	Alteração de tipos de processo	O sistema deve permitir alteração por usuários autorizados dos tipos de processos cadastrados.
19	Listagem de tipos de processo	O sistema deve listar todos os tipos de processos cadastrados.
20	Exclusão de tipos de processo	O sistema deve permitir usuários capazes excluir tipos de processos cadastrados.
21	Desativação de tipos de processo	O sistema deve permitir usuários autorizados desativar tipos de processos dessa forma arquivos dependentes não serão perdidos.
22	Cadastro de etapa	O sistema deve permitir o cadastro de etapas por usuários autorizados.
23	Alteração de etapa	O sistema deve permitir a alteração de etapas cadastradas por usuários autorizados.
24	Listagem de etapa	O sistema deve listar todas as etapas cadastradas.
25	Exclusão de etapa	O sistema deve permitir a exclusão de etapas cadastradas por usuários autorizados.
26	Desativação de etapa	O sistema deve permitir a desativação de etapas cadastradas por usuários autorizados, dessa forma arquivos dependentes não serão afetados.
27	Cadastro de tipos de etapa	O sistema deve permitir o cadastro de tipos de etapas por usuários autorizados
28	Alteração de tipos de etapa	O sistema deve permitir alterações para tipos de etapas por usuários autorizados.
29	Exclusão de tipos de etapa	O sistema deve permitir a exclusão de tipos de etapa cadastrados por usuários autorizados
30	Desativação de tipos de etapa	O sistema deve permitir a desativação de tipos de etapa cadastrados por usuários autorizados.
31	Listagem de tipos de etapa	O sistema deve listar todos os tipos de etapas cadastradas.
32	Cadastro de documentos	O sistema deve cadastrar documentos por meio de usuários autorizados.

33	Alteração de documentos	O sistema deve permitir alteração de documentos cadastrados no sistema por meio de usuários autorizados.
34	Exclusão de documentos	O sistema deve permitir a exclusão de documentos cadastrados no sistema por meio de usuários autorizados.
35	Desativação de documentos	O sistema deve permitir a desativação de documentos cadastrados no sistema por usuários autorizados, dessa forma arquivos dependentes não serão afetados.
36	Listagem de documentos	O sistema deve listar todos os documentos cadastrados.
37	Cadastro de tipos de documento	O sistema deve permitir o cadastro de tipos de documentos por usuários autorizados.
38	Alteração de tipos de documento	O sistema deve permitir a alteração de tipos de documentos por usuários autorizados
39	Exclusão de tipos de documento	O sistema deve permitir a exclusão de tipos de documentos por usuários autorizados.
40	Listagem de tipos de documento	O sistema deve listar todos os tipos de documentos cadastrados.
41	Desativação de tipos de documento	O sistema deve permitir a desativação de tipos de documento por usuários autorizados, assim arquivos dependentes não serão afetados.

Fonte: Elaborado pelos autores.

2.6 DESCRIÇÃO DOS REQUISITOS NÃO FUNCIONAIS

Segundo Guedes (2011), os requisitos não funcionais frequentemente afetam todo o sistema, afetando o design e a implementação. Esses requisitos incluem tempo de resposta aceitável, capacidade do sistema para suportar cargas de usuários simultâneos e componentes de segurança robustos contra ameaças externas. Para garantir que o sistema cumpra os requisitos de qualidade e desempenho exigidos pelos stakeholders, sua identificação e gestão são cruciais.

Para construir sistemas mais eficientes e adaptados às necessidades dos usuários e das organizações. Guedes (2011) afirma que é essencial considerar os requisitos não funcionais desde as fases iniciais do desenvolvimento. Embora a UML não forneça diagramas específicos para representar diretamente esses requisitos, eles podem ser documentados e conectados a elementos arquiteturais, como classes, componentes ou casos de uso, por meio de diagramas. Isso permite uma modelagem do sistema mais completa e robusta.

O Quadro 2 é referente à os principais requisitos não funcionais identificados para o projeto em questão. Esses requisitos abrangem desde a agilidade na atualização do banco de dados até a compatibilidade das máquinas com navegador.

Quadro 2 – Requisitos Não Funcionais do Sistema

	Requisitos não funcionais	Descrição
1	Atualização do banco de dados	Aprimorar a agilidade na atualização do banco de dados do cliente é essencial para evitar a presença de informações desatualizadas no sistema.
2	Máquina compatível com navegador	É fundamental que o dispositivo suporte um navegador para acessar e utilizar o sistema.
3	Auxiliar na atualização do sistema	Vamos fornecer suporte aos usuários, orientando-os a utilizar o software de maneira mais produtiva.

Fonte: Elaborado pelos autores.

3 VISÃO DE CASO DE USO – UML

A UML (*Unified Modeling Language*), conforme descrito por Guedes (2011), é uma linguagem padrão projetada para especificar, visualizar, construir e documentar artefatos de sistemas orientados a objetos, auxiliando no desenvolvimento de software de maneira organizada e clara. Ela, segundo o autor, permite criar diagramas que representam tanto a estrutura estática quanto o comportamento dinâmico dos sistemas, oferecendo uma visão abrangente e detalhada dos componentes e suas interações.

Dentro da UML, o Diagrama de Casos de Uso ocupa, segundo Guedes (2011), um papel de destaque ao descrever as funcionalidades do sistema sob a perspectiva do usuário final. Ele menciona que esse diagrama é utilizado para documentar de forma clara e objetiva as principais funções que o sistema deve executar, enfatizando as interações entre os atores (usuários ou sistemas externos) e os casos de uso (funcionalidades ou serviços oferecidos pelo sistema). Para Guedes (2011), a modelagem de casos de uso é essencial para a definição dos requisitos do sistema, funcionando como o ponto de partida para a modelagem, pois assegura que as expectativas dos stakeholders sejam atendidas e facilita a comunicação eficaz entre desenvolvedores, analistas e usuários.

O autor explica que o diagrama vai além de simplesmente identificar as funcionalidades, pois também revela as dependências e relações entre elas, oferecendo uma visão abrangente do sistema e seu contexto de operação. O autor também destaca a importância da ferramenta no desenvolvimento de software, especialmente nas fases iniciais de levantamento e análise dos requisitos (Guedes, 2011).

3.1 DIAGRAMA DE CLASSES

O Diagrama de Classes é uma das ferramentas mais importantes na modelagem orientada a objetos, pois oferece uma visão clara e organizada da estrutura interna de um sistema. Ele permite a identificação das classes do software, bem como suas características, métodos e conexões entre elas, como associações, heranças e composições. O uso de um diagrama de classes para representar classes, atributos, operações e relacionamentos permite a modelagem da estrutura estática de um sistema. Isso ajuda na construção de uma base sólida para a implementação do sistema (Guedes, 2011).

O processo de refatoração é facilitado pelo uso do modelo de diagrama de classe, que identifica erros de design durante as etapas iniciais. Esse diagrama ilustra como a complexidade do sistema tende a aumentar com o tempo. No entanto, a clareza visual do sistema facilita a

refatoração e reduz a probabilidade de novos problemas de software surgirem (Guedes, 2011). Assim, o Diagrama de Classes é uma ferramenta dinâmica útil para o desenvolvimento e manutenção de sistemas.

Segundo Sommerville (2018), O diagrama de classes é essencial para representar os componentes essenciais de um sistema, como as classes e seus relacionamentos, permitindo que os engenheiros de software visualizem a estrutura do sistema de forma coesa. Ele enfatiza o uso desse diagrama para aprender sobre a "arquitetura estática" do sistema, pois facilita a tradução do design em código durante a fase de implementação. Sommerville enfatiza que os diagramas de classes são especialmente úteis para a comunicação entre os membros da equipe de desenvolvimento porque fornecem uma visão compartilhada das estruturas fundamentais do sistema.

O diagrama apresentado na Figura 4 ilustra o Diagrama de Classes, que detalha a configuração da camada de negócios do sistema em questão. Este diagrama pode ser dividido em três esferas principais que representam os conceitos centrais do modelo: entidade, imóvel e processo.

- Entidade: A primeira esfera abrange as classes Pessoa, TipoUsuario, Usuario, Municipio, Engenheiro, Fiscal, Sessao e Auditoria. Ela lida com as informações relacionadas aos indivíduos que interagem ou participam do contexto do projeto, incluindo detalhes sobre seus papéis, status e histórico de ações dentro do sistema.
- Imóvel: A segunda esfera foca nas classes Imovel, Logradouro, TipoLogradouro, Bairro, Cidade, Estado, Topografia, TipoUso, OcupacaoAtual, Instalacao, Infraestrutura e TipoInfraestrutura. Ela modela os aspectos relacionados aos imóveis, cobrindo desde seus dados cadastrais e características físicas até informações sobre ocupação, endereço e infraestrutura associada.
- Processo: A última esfera é dedicada à modelagem dos fluxos e estados dos processos, englobando as classes Processo, StatusProcesso, DocumentoProcesso, StatusDocumentoProcesso, TipoProcesso, Etapa, TipoDocumentoEtapa e TipoDocumento. Ela descreve as etapas do ciclo de vida dos processos, incluindo a documentação associada, e define as interações entre as entidades e os imóveis, garantindo a execução eficiente dos fluxos de trabalho no sistema, sendo o núcleo.

O Diagrama de Classes oferece uma visão abrangente e estruturada da arquitetura do sistema, evidenciando a interdependência entre as entidades, imóveis e processos, fundamentais para o entendimento e desenvolvimento da camada de negócios.

3.2 DICIONÁRIO DE CLASSES

Na UML, a classe é um dos principais elementos para a modelagem de sistemas. Ela define um tipo específico de objeto e serve como um molde ou *template* para a criação de instâncias. Uma classe descreve as propriedades (atributos) e os comportamentos (métodos)

que são compartilhados por todos os objetos dessa classe. Ou seja, ela especifica o que os objetos terão em termos de dados (atributos) e ações que poderão executar (métodos). Por exemplo, a classe TipoUsuario, conforme mencionado por Guedes (2011), é responsável por modelar os diferentes tipos de usuários dentro do sistema, incluindo atributos como nivelDeAcesso, descricao e nomeTipoUsuario, que descrevem o nível de acesso e as permissões de cada tipo.

A interface, por sua vez, é uma estrutura que define um conjunto de operações (métodos) ou dados (atributos) que devem ser implementadas por uma classe, mas sem especificar como essas operações devem ser realizadas. Em essência, ela estabelece um contrato, garantindo que as classes que a implementam forneçam determinada funcionalidade, mas de forma flexível e sem uma implementação rígida. Isso torna as interfaces ideais para promover a reutilização do código, permitindo que diferentes classes compartilhem o mesmo conjunto de operações, mas com implementações independentes (Guedes, 2011). Na UML, as interfaces são representadas com o nome precedido pela estampa <<interface>>.

Por fim, o enum (ou enumeração), nas palavras de Guedes (2011), é um tipo de dado que consiste em um conjunto fixo de valores constantes. Na UML, ele é utilizado para representar atributos que podem assumir apenas um número limitado de valores predefinidos. As enums são especialmente úteis quando se deseja restringir o valor de um atributo a um conjunto específico e bem definido de opções. Por exemplo, o enum StatusProcesso define os seguintes valores pré-determinados: 'Em Espera' (0), 'Em Progresso' (1), 'Em Análise' (2), 'Aprovado' (3) e 'Reprovado' (4). Esse uso de enumerações garante que o sistema só atribua um desses valores ao atributo status, evitando entradas inválidas ou inconsistentes.

Em relação aos tipos de relacionamentos entre classes na UML, conforme explorado por Guedes (2011), podemos destacar os seguintes:

- Associação Binária: Guedes (2011) explica que a associação binária é um tipo de relacionamento entre duas classes, no qual os objetos de uma classe estão conectados aos objetos de outra classe. Esse tipo de associação é o mais comum nos diagramas UML e é representado por uma linha simples que liga as duas classes. Essa linha pode ser complementada para mostrar aspectos adicionais, como a multiplicidade (quantidade de objetos envolvidos) e a navegabilidade (direção da associação).
- Classe Associativa: A classe associativa surge quando o relacionamento entre duas classes precisa ser descrito de forma mais detalhada do que uma simples associação. Guedes (2011) descreve uma classe associativa como uma classe intermediária que

modela o relacionamento entre as duas classes principais, contendo atributos e comportamentos específicos para esse relacionamento.

- **Realização:** O relacionamento de realização é descrito por Guedes (2011) como o vínculo entre uma classe concreta e uma interface. Ele é representado por uma linha pontilhada com um triângulo na ponta, apontando para a interface. Isso significa que a classe concreta é responsável por implementar os métodos ou operações definidas pela interface, garantindo que a interface forneça um "contrato" e a classe concreta se encarregue de cumprir os detalhes dessa implementação.
- **Dependência:** A dependência, por outro lado, descreve uma relação mais frágil entre duas classes, onde uma classe depende de outra para funcionar, mas sem que haja uma associação forte ou permanente. Guedes (2011) ilustra esse relacionamento com uma linha pontilhada e uma seta apontando para a classe da qual a dependência se origina. Exemplificando, a classe Pedido pode ter um atributo do tipo StatusPedido, representado por um enum com valores como "Em Espera", "Concluído" e "Cancelado". Embora o Pedido dependa do StatusPedido para definir seu estado, essa dependência não é forte, pois o StatusPedido já possui valores pré-definidos e não é uma instância que o Pedido modifique diretamente.

Quanto aos tipos de dados, de acordo com Sommerville (2018), tipos primitivos como string, bool, int e Guid, desempenham um papel essencial na definição de atributos e métodos em classes, interfaces e enums. Esses tipos fundamentam como informações e comportamentos devem ser representados no sistema de forma consistente e padronizada. O tipo string, por exemplo, é amplamente usado para armazenar texto, como nomes ou descrições; o bool é utilizado para expressar valores lógicos (verdadeiro ou falso), usados em condições ou atributos binários, como "ativo" ou "inativo". Já o tipo int é essencial para armazenar valores numéricos inteiros, como identificadores, quantidades ou valores predefinidos em enums, enquanto o Guid é uma representação única utilizada para identificar entidades ou objetos de maneira exclusiva.

No contexto da UML e da modelagem orientada a objetos, esses tipos primitivos são frequentemente utilizados para descrever atributos de classes no dicionário de classes, conforme sugerido por Pressman e Maxim (2021). Eles fornecem uma base concreta para a implementação das propriedades definidas em modelos abstratos.

No Quadro 3 apresenta-se a interface *Pessoa*, projetada para armazenar atributos comuns às classes Usuário, Município, Engenheiro e Fiscal. Essa interface serve como um molde, facilitando a reutilização de atributos compartilhados e simplificando a implementação.

Quadro 3 – Descrição da Interface Pessoa

Atributo	Tipo	Descrição
imagemPessoa	String	Imagem de identificação da pessoa no formato Base64.
nomePessoa	String	Nome de identificação da pessoa.
emailPessoa	String	E-mail de identificação utilizado para acessar o sistema.
numeroTelefone	String	Número de telefone da pessoa.
cpfCNPJ	String	CPF para pessoas físicas e CNPJ para pessoas jurídicas.
rgIE	String	RG (Registro Geral) para identificação pessoal e IE (Inscrição Estadual) para identificação empresarial.

Fonte: Elaborado pelos autores.

O Quadro 4 apresenta a classe *Munícipe*, que desempenha o papel de representar todos os cidadãos no sistema.

Quadro 4 – Descrição da Classe *Munícipe*

Atributo	Tipo	Descrição
idMunícipe	Int	Código para identificação do munícipe.

Fonte: Elaborado pelos autores.

No Quadro 5 é apresentada a descrição dos atributos da classe *Engenheiro*, que define as características e identificadores necessários.

Quadro 5 – Descrição da Classe *Engenheiro*

Atributo	Tipo	Descrição
idEngenheiro	Int	Código para identificação do engenheiro.
creaEngenheiro	String	CREA de identificação do engenheiro.

Fonte: Elaborado pelos autores.

No Quadro 6, é descrita a classe *Fiscal*, relacionada a fiscalização da obra associada ao documento e processo.

Quadro 6 – Descrição da Classe Fiscal

Atributo	Tipo	Descrição
idFiscal	Int	Código para identificação do fiscal.

Fonte: Elaborado pelos autores.

No quadro 7, apresenta a classe *TipoUsuario* é fundamental para a organização e controle de acesso no sistema, sendo composta por atributos que definem a identificação e os níveis de permissão de cada usuário. No Quadro 3, são apresentados os atributos dessa classe.

Quadro 7 – Descrição da Classe TipoUsuario

Atributo	Tipo	Descrição
idTipoUsuario	Int	Código para identificar o tipo de usuário.
nivelAcesso	String	Letra para identificar o tipo de nível de acesso.
nomeTipoUsuario	String	Nome de identificação do tipo de usuário.
descricaoTipoUsuario	String	Descrição sobre o tipo de usuário.

Fonte: Elaborado pelos autores.

O Quadro 8 apresenta a classe *Usuário*, que representa as entidades responsáveis por acessar o sistema e executar ações dentro dele.

Quadro 8 – Descrição da Classe Usuário

Atributo	Tipo	Descrição
idUsuario	Int	Código para identificação do usuário.
senhaUsuario	String	Senha de acesso ao sistema, criptografada com SHA256.
statusUsuario	Bool	Indica se o usuário está ativo ou inativo.

Fonte: Elaborado pelos autores.

A classe *Sessão* é responsável por fornecer uma chave de acesso ao sistema para o usuário, administrando por quanto tempo ela estará válida e se permanece integra. No Quadro 9 são apresentados seus atributos.

Quadro 9 - Descrição da Classe Sessão

Atributo	Tipo	Descrição
idSessao	Guid	Código para identificação da sessão.
dataHoraInicio	String	Data e hora em que a sessão foi aberta.
dataHoraFechamento	String	Data e hora em que a sessão foi fechada.
tokenSessao	String	Token atribuído a sessão, servindo como chave de acesso para o sistema e identificação do usuário.
statusSessao	Bool	Status booleano para identificar se a sessão está aberta (true) ou fechada (false).
ipv4	String	Endereço IPv4 do dispositivo que fez a requisição.
ipv6	String	Endereço IPv6 do dispositivo que fez a requisição.

Fonte: Elaborado pelos autores.

A classe *Auditoria* é responsável por guardar o histórico de logs das requisições do usuário, sobre o que o mesmo acessou, a data e se foi bem-sucedida ou não. No Quadro 10 são apresentados seus atributos.

Quadro 10 - Descrição da Classe Auditoria

Atributo	Tipo	Descrição
idAuditoria	Guid	Código para identificação da auditoria.
dataAuditoria	String	Data em que a ação foi feita.
horaAuditoria	String	Hora em que a ação foi feita.
endpointAuditoria	String	Endpoint da API requisitado.
acaoAuditoria	String	Qual o tipo de ação: <ul style="list-style-type: none"> • GET: Solicitação, • POST: Inserção, • PUT: Alteração, • DELETE: Exclusão.

tabelaAuditoria	String	Qual tabela/classe foi acessada ou afetada.
registroAuditoria	String	Se for um Post, Put ou Delete, aqui virá o id do registro.
statusAuditoria	String	Status se a requisição foi bem-sucedida, se ocorreu algum erro ou bloqueio.
descricaoAuditoria	String	Descrição do que foi feito ou ocorreu.

Fonte: Elaborado pelos autores.

No Quadro 11, a classe *Estado* tem objetivo armazenar informações relacionadas aos estados do país.

Quadro 11 – Descrição da Classe Estado

Atributo	Tipo	Descrição
idEstado	Int	Código para identificação do Estado.
nomeEstado	String	Nome de identificação do Estado.
ufEstado	String	Sigla do estado.

Fonte: Elaborado pelos autores.

No Quadro 12, descreve-se a classe *Cidade* com o objetivo armazenar dados relacionados à cidade dentro do estado.

Quadro 12 – Descrição da Classe Cidade

Atributo	Tipo	Descrição
idCidade	Int	Código para identificação da cidade.
nomeCidade	String	Nome de identificação da cidade.

Fonte: Elaborado pelos autores.

No Quadro 13, apresenta a classe *Bairro*, com o objetivo armazenar informações sobre os bairros de uma determinada cidade.

Quadro 13 – Descrição da Classe Bairro

Atributo	Tipo	Descrição
idBairro	Int	Código para identificação do bairro.

nomeBairro	String	Nome de identificação do bairro.
------------	--------	----------------------------------

Fonte: Elaborado pelos autores.

No Quadro 14, aborda-se a classe *TipoLogradouro* que tem como objetivo armazenar o tipo de logradouro de um endereço, como por exemplo avenida, alameda, rua, entre outros.

Quadro 14 – Descrição da Classe TipoLogradouro

Atributo	Tipo	Descrição
idTipoLogradouro	Int	Código para identificação do tipo de endereço.
codigoInformativo	String	Identificador para classificar e padronizar os diferentes tipos de vias públicas.
descricao	String	Descrição do tipo logradouro.

Fonte: Elaborado pelos autores.

No Quadro 15, é descrita a classe *Logradouro*, com o objetivo de armazenar informações sobre o endereço do imóvel.

Quadro 15 – Descrição da Classe Logradouro

Atributo	Tipo	Descrição
idLogradouro	Int	Código para identificação do logradouro.
cep	String	CEP do logradouro.
nomeLogradouro	String	Nome do logradouro.
numeroInicial	String	Número que indica o início da rua.
numeroFinal	String	Número que indica o final da rua.

Fonte: Elaborado pelos autores.

No Quadro 16, é apresentada a classe *Imóvel*, cujo objetivo é armazenar as informações referentes ao imóvel.

Quadro 16 – Descrição da Classe Imóvel

Atributo	Tipo	Descrição
idImovel	Int	Código para identificação do imóvel.

inscricaoCadastral	String	Código cadastral do imóvel.
numeroCasa	String	Número de identificação da casa.
areaTerreno	String	Área total do terreno do imóvel.
areaConstruida	String	Área ocupada do terreno do imóvel.
condicoesSolo	String	Condições do terreno do imóvel.
valorVenal	String	Valor venal do imóvel.
valorMercado	String	Valor do mercado do imóvel.

Fonte: Elaborado pelos autores.

No Quadro 17, apresenta-se a classe *Topografia*, que tem como objetivo armazenar as informações referentes à topografia do imóvel.

Quadro 17 – Descrição da Classe Topografia

Atributo	Tipo	Descrição
idTopografia	Int	Código para identificação da topografia.
nomeTopografia	String	Nome de identificação da topografia.

Fonte: Elaborado pelos autores.

No Quadro 18, aborda-se a classe *TipoUso*, responsável por armazenar as informações referentes aos usos do imóvel.

Quadro 18 – Descrição da Classe TipoUso

Atributo	Tipo	Descrição
idTipoUso	Int	Código para identificação do tipo de uso.
nomeTipoUso	String	Nome de identificação do tipo de uso.
descricaoTipoUso	String	Descrição do tipo de uso.

Fonte: Elaborado pelos autores.

No Quadro 19, é apresentada a classe *OcupaçãoAtual*, cujo objetivo é armazenar as informações referentes às diferentes ocupações do imóvel.

Quadro 19 – Descrição da Classe OcupaçãoAtual

Atributo	Tipo	Descrição
idOcupacaoAtual	Int	Código para identificação da ocupação atual.
nomeOcupacaoAtual	String	Nome de identificação da ocupação atual.
descricaoOcupacaoAtual	String	Descrição da ocupação atual.

Fonte: Elaborado pelos autores.

No Quadro 20, é apresentada a classe *TipoInfraestrutura*, cujo objetivo é armazenar as informações referentes aos tipos de infraestrutura existentes, como saneamento, acessibilidade e segurança.

Quadro 20 – Descrição da Classe TipoInfraestrutura

Atributo	Tipo	Descrição
idTipoInfraestrutura	Int	Código para identificação do tipo de infraestrutura.
nomeTipoInfraestrutura	String	Nome de identificação do tipo de infraestrutura.
descricaoTipoInfraestrutura	String	Descrição do tipo de infraestrutura.

Fonte: Elaborado pelos autores.

No Quadro 21, apresenta a classe *Infraestrutura*, que tem como objetivo armazenar as informações referentes as infraestruturas existentes.

Quadro 21 – Descrição da Classe Infraestrutura

Atributo	Tipo	Descrição
idInfraestrutura	Int	Código para identificação da infraestrutura.
nomeInfraestrutura	String	Nome de identificação da infraestrutura.

Fonte: Elaborado pelos autores.

No Quadro 22, é apresentada a classe associativa *Instalação*, cujo objetivo é armazenar as informações referentes às infraestruturas instaladas no imóvel.

Quadro 22 – Descrição da Classe Associativa Instalação

Atributo	Tipo	Descrição
idInstalacao	Int	Código para identificação da instalação.
dataInstalacao	String	Data em que a instalação foi feita.
situacaoInstalacao	String	Situação da instalação.

Fonte: Elaborado pelos autores.

No Quadro 23, é apresentado o enum *StatusProcesso*, cujo objetivo é definir os diferentes estados que o processo pode assumir.

Quadro 23 – Descrição do Enum StatusProcesso

Atributo	Tipo	Descrição
EmEspera	Int	Estado de quando o processo está em espera, representado pelo valor 0.
EmProgresso	Int	Estado de quando o processo está em progresso, representado pelo valor 1.
EmAnalise	Int	Estado de quando o processo está em análise, representado pelo valor 2.
Aprovado	Int	Estado de quando o processo foi aprovado, representado pelo valor 3.
Reprovado	Int	Estado de quando o processo foi reprovado, representado pelo valor 4.

Fonte: Elaborado pelos autores.

No Quadro 24, é apresentada a classe *Processo*, cujo objetivo é conter as informações relacionadas ao processo de obras.

Quadro 24 – Descrição da Classe Processo

Atributo	Tipo	Descrição
idProcesso	Guid	Código para identificação do processo.
identificacaoProcesso	String	Código de identificação do processo, gerado a partir do tipo de processo.
situacaoProcesso	String	Situação atual do processo.

descricaoProcesso	String	Descrição sobre o processo.
dataInicio	String	Data de quando o processo foi deliberado para início.
dataFinalizacao	String	Data em que o processo foi registrado como terminado.
dataAprovacao	String	Data e hora em que o administrador aprovou o processo.

Fonte: Elaborado pelos autores.

No Quadro 25, é apresentada a classe *TipoProcesso*, que tem como objetivo definir o tipo de processo a ser armazenado.

Quadro 25 – Descrição da Classe TipoProcesso

Atributo	Tipo	Descrição
idTipoProcesso	Int	Código para identificação do tipo de processo.
nomeTipoProcesso	String	Nome de identificação do tipo de processo.
descricaoTipoProcesso	String	Informações sobre o tipo de processo.

Fonte: Elaborado pelos autores.

No Quadro 26, é apresentada a interface *Posição*, que fornece um contrato para ordenar as etapas relacionadas ao processo e os tipos de documentos associados a essas etapas.

Quadro 26 – Descrição da Interface Posição

Atributo	Tipo	Descrição
posicao	Int	Atributo que armazena em qual posição os elementos se encontram. Exemplo: 1 (primeiro), 2 (segundo) e etc.

Fonte: Elaborado pelos autores.

No Quadro 27, é apresentada a classe *Etapas*, cuja finalidade é armazenar o estágio em que o processo se encontra.

Quadro 27 – Descrição da Classe Etapa

Atributo	Tipo	Descrição
idEtapa	Int	Código para identificação da etapa.
nomeEtapa	String	Nome de identificação da etapa.
descricaoEtapa	String	Informações sobre a etapa do processo.

Fonte: Elaborado pelos autores.

No Quadro 28, é apresentada a classe *TipoDocumento*, cuja finalidade é armazenar os tipos de documentos existentes no processo.

Quadro 28 – Descrição da Classe TipoDocumento

Atributo	Tipo	Descrição
idTipoDocumento	Int	Código para identificação do tipo de documento.
nomeTipoDocumento	String	Nome de identificação do tipo de documento.
descricaoTipoDocumento	String	Informações sobre a descrição do tipo de documento.

Fonte: Elaborado pelos autores.

No Quadro 29, é apresentada a classe *TipoDocumentoEtapa*, que detalha quais tipos de documentos são necessários para cada etapa dos processos.

Quadro 29 – Descrição da Classe Associativa TipoDocumentoEtapa

Atributo	Tipo	Descrição
idTipoDocumentoEtapa	Int	Código para identificação do tipo de documento relacionado a etapa.

Fonte: Elaborado pelos autores.

No Quadro 30, é apresentada a definição do enum *StatusDocumentoProcesso*, responsável por indicar o estado do documento do processo, determinando se as informações, como o arquivo, estão corretas e íntegras.

Quadro 30 – Descrição do Enum StatusDocumentoProcesso

Atributo	Tipo	Descrição
----------	------	-----------

EmEspera	Int	Estado de quando o documento do processo está em espera, representado pelo valor 0.
Pendente	Int	Estado de quando o documento processo está pendente, representado pelo valor 1.
NaoAnexado	Int	Estado de quando o arquivo do documento do processo não foi anexado junto com as informações passadas, representado pelo valor 2.
NaoAnexado	Int	Estado de quando o arquivo do documento do processo não está inteiro (diferença entre o hash recebido com o hash gerado do arquivo), representado pelo valor 3.
Anexado	Int	Estado de quando o documento do processo possui as informações necessárias e o arquivo, representado pelo valor 4.
EmAnalise	Int	Estado de quando o documento do processo está em análise, representado pelo valor 5.
Aprovado	Int	Estado de quando o documento do processo foi aprovado, representado pelo valor 6.
Reprovado	Int	Estado de quando o documento do processo foi reprovado, representado pelo valor 7.

Fonte: Elaborado pelos autores.

No Quadro 31, é apresentada a classe *DocumentoProcesso*, responsável pela representação do documento físico associado a todo o processo realizado, incluindo suas etapas específicas.

Quadro 31 – Descrição da Classe DocumentoProcesso

Atributo	Tipo	Descrição
idDocumentoProcesso	Guid	Código para identificação do documento do processo.
identificacaoDocumento	String	Código de identificação do documento do processo, gerado a partir do tipo de documento que está relacionado.

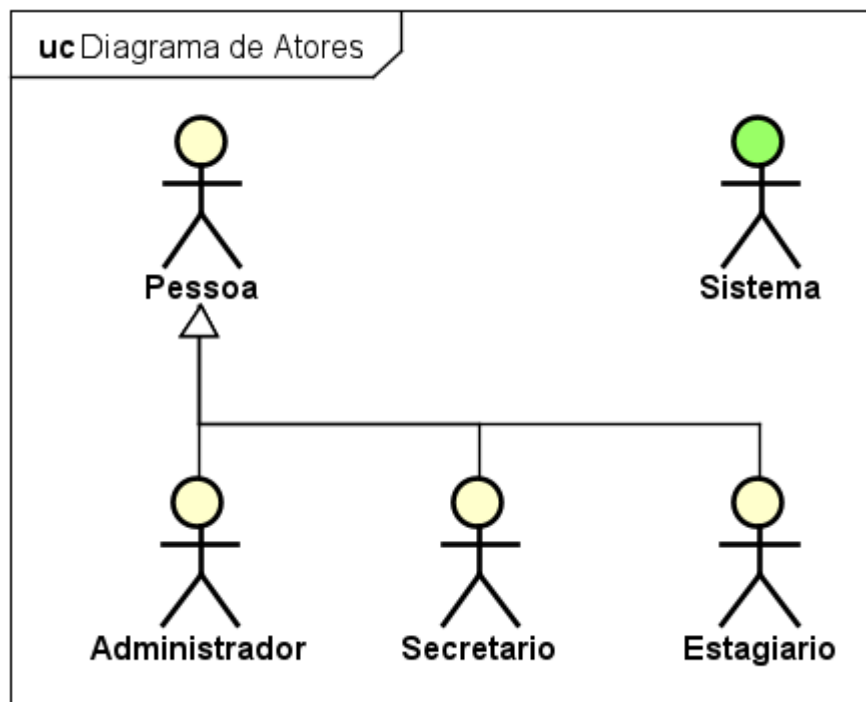
descricaoDocumento	String	Descrição do documento do processo.
observacaoDocumento	String	Observação sobre o documento do processo.
arquivo	String	Atributo que armazena o objeto Archive, contendo hash, bytes, fileName e mimeType, serializado em formato JSON.
dataExpedicao	String	Data oficial de expedição do documento pelo órgão responsável.
dataAprovacao	String	Data e hora em que o administrador aprovou o documento.

Fonte: Elaborado pelos autores.

3.3 DEFINIÇÃO DOS ATORES

O diagrama de atores é essencial para a modelagem de sistemas, especialmente quando se trata de diagramas de casos de uso. Guedes (2011) afirma que os atores são entidades externas que têm uma interação direta com o sistema. Eles podem ser humanos ou outros sistemas. A identificação adequada dos atores permite uma melhor compreensão dos requisitos e interações do sistema, ajudando no mapeamento dos requisitos funcionais e na criação de uma interface eficiente. A definição de como o sistema deve reagir a vários tipos de usuários ou sistemas externos depende desses atores.

Sommerville (2018) reforça essa ideia ao destacar que o diagrama de atores permite visualizar as interações entre os papéis externos e o sistema. Ele enfatiza que os atores não são apenas indivíduos; eles podem ser qualquer entidade externa que interage com o sistema. Isso facilita a comunicação entre a equipe de desenvolvimento e os interessados, garantindo que todas as interações relevantes sejam abordadas no processo de design e implementação do sistema. Considerando a implementação da classe Tipo Usuário, podemos definir alguns do possível tipo de usuários (ou atores) que poderão ter acesso ao sistema como demonstrado na Figura 6.

Figura 5 – Diagrama de Atores

Fonte: Elaborado pelos autores.

Segundo Guedes (2011), descrição de atores é uma ferramenta usada para fornecer detalhes sobre os atores que interagem com o sistema em análise, especificamente no contexto de modelagem UML. O Quadro 32 descreve as características e funções dos atores dentro de um sistema, como sua finalidade, papel, e as interações que terão com o sistema.

Quadro 32 – Descrição dos Atores

Tipo	Descrição
Sistema	O ator sistema é a interface visual hospedada na web onde ocorre a comunicação direta com usuário.
Pessoa	O ator pessoa tem por objetivo generalizar (representar) todos os usuários que acessam o sistema.
Administrador	O ator secretário geral tem acesso à todas as funcionalidades implementadas no sistema que são voltadas ao usuário.
Secretário	O ator secretário tem acesso à todas as funcionalidades implementadas no sistema que são voltadas ao usuário, sendo o nível médio dentro da aplicação, porém necessitando de validação do administrador para a aprovação da ação, como anexar documento.

Estagiário	O ator estagiário é o nível inferior, com acesso a pontos de baixo impacto, até completa validação e deliberação a acesso aos módulos superiores.
------------	---

Fonte: Elaborado pelos autores.

3.4 LISTA DE CASOS DE USO

Neste subtópico, será detalhado as ações que cada ator pode executar dentro do sistema. Para padronizar e organizar as possíveis respostas do sistema, foram predefinidos status específicos representados por um enum na aplicação, voltado ao gerenciamento e padronização de respostas que serão exibidas ao ator em diferentes cenários, junto de um objeto que contém a descrição de como ocorreu o processamento da requisição. Essas mensagens fornecem uma forma de feedback sobre o resultado das ações executadas como representado no Quadro 33.

Quadro 33 – Mensagens de saída

Identificação	Mensagem	Descrição
Msg1	Sucesso.	Está mensagem ocorre a requisição é bem-sucedida, seja para ações de solicitação de dados, cadastro, alteração ou exclusão.
Msg2	Inválido.	Está mensagem ocorre quando alguma informação está incorreta.
Msg3	Não encontrado.	Está mensagem ocorre quando a informação passada não é encontrada no banco de dados.
Msg4	Conflito.	Está mensagem ocorre quando o ator informado um objeto em que parte dos dados (que devem ser únicos) já foram informados e estão armazenados.
Msg5	Não autorizado.	Está mensagem ocorre quando a sessão do usuário se torna inválida, ou o log de auditoria aponte que os IP identificados não correspondem ao armazenados na sessão.
Msg6	Proibido.	Está mensagem ocorre quando o ator requisita por um modulo em que não tem permissão de acesso.
Msg7	Erro.	Está mensagem ocorre quando ocorre um erro de execução, levando ao bloco <i>catch</i> de tratativas.

Fonte: Elaborado pelos autores.

Após a predefinição das mensagens que podem ocorrer durante as ações efetuadas pelo ator, podemos visualizar quais ações cada ator pode efetuar dentro do sistema, bem como seus dados de entrada e saída. Nas palavras de Guedes (2011), lista de casos de uso é uma ferramenta de modelagem usada para descrever as interações entre os atores e o sistema, com foco nas

ações que cada ator pode realizar dentro de um processo específico, contendo informações essenciais sobre o comportamento do sistema do ponto de vista dos usuários ou sistemas externos (atores), detalhando os passos envolvidos em cada caso de uso.

É apresentado no Quadro 34 a lista de casos de uso na visão do ator Administrador, com acesso a maior parte dos módulos do sistema.

Quadro 34 – Lista de Casos de Uso: Ações do Administrador

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Administrador cadastra estado	Dados Estado	Cadastrar Estado	Msg1
02	Administrador cadastra cidade	Dados Cidade	Cadastrar Cidade	Msg1
03	Administrador cadastra bairro	Dados Bairro	Cadastrar Bairro	Msg1
04	Administrador cadastra tipo logradouro	Dados Tipo Logradouro	Cadastrar Tipo Logradouro	Msg1
05	Administrador cadastra logradouro	Dados Logradouro	Cadastrar Logradouro	Msg1
06	Administrador cadastra município	Dados Município	Cadastrar Município	Msg1
07	Administrador cadastra engenheiro	Dados Engenheiro	Cadastrar Engenheiro	Msg1
08	Administrador cadastra fiscal	Dados Fiscal	Cadastrar Fiscal	Msg1
09	Administrador cadastra tipo usuário	Dados Tipo Usuario	Cadastrar Tipo Usuario	Msg1
10	Administrador cadastra usuário	Dados Usuario	Cadastrar Usuario	Msg1
11	Administrador cadastra imóvel	Dados Imovel	Cadastrar Imovel	Msg1
12	Administrador cadastra topografia	Dados Topografia	Cadastrar Topografia	Msg1
13	Administrador cadastra tipo de uso	Dados Tipo Uso	Cadastrar Tipo Uso	Msg1
14	Administrador cadastra ocupação atual	Dados Ocupacao Atual	Cadastrar Ocupacao Atual	Msg1
15	Administrador cadastra tipo de infraestrutura	Dados Tipo Infraestrutura	Cadastrar Tipo Infraestrutura	Msg1
16	Administrador cadastra infraestrutura	Dados Infraestrutura	Cadastrar Infraestrutura	Msg1
17	Administrador cadastra instalação	Dados Instalacao	Cadastrar Instalacao	Msg1
18	Administrador cadastra tipo processo	Dados Tipo Processo	Cadastrar Tipo Processo	Msg1
19	Administrador cadastra processo	Dados Processo	Cadastrar Processo	Msg1
20	Administrador cadastra etapa	Dados Etapa	Cadastrar Etapa	Msg1
21	Administrador cadastra tipo documento	Dados Tipo Documento	Cadastrar Tipo Documento	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
22	Administrador cadastra documento processo	Dados Documento Processo	Cadastrar Documento Processo	Msg1
23	Administrador relaciona tipo documento a etapa	Dados Tipo Documento Etapa	Cadastrar Tipo Documento Etapa	Msg1
24	Administrador altera estado	Dados Estado	Alterar Estado	Msg1
25	Administrador altera cidade	Dados Cidade	Alterar Cidade	Msg1
26	Administrador altera bairro	Dados Bairro	Alterar Bairro	Msg1
27	Administrador altera tipo logradouro	Dados Tipo Logradouro	Alterar Tipo Logradouro	Msg1
28	Administrador altera logradouro	Dados Logradouro	Alterar Logradouro	Msg1
29	Administrador altera município	Dados Municípe	Alterar Municípe	Msg1
30	Administrador altera engenheiro	Dados Engenheiro	Alterar Engenheiro	Msg1
31	Administrador altera fiscal	Dados Fiscal	Alterar Fiscal	Msg1
32	Administrador altera tipo usuário	Dados Tipo Usuario	Alterar Tipo Usuario	Msg1
33	Administrador altera usuário	Dados Usuario	Alterar Usuario	Msg1
34	Administrador altera imóvel	Dados Imovel	Alterar Imovel	Msg1
35	Administrador altera topografia	Dados Topografia	Alterar Topografia	Msg1
36	Administrador altera tipo de uso	Dados Tipo Uso	Alterar Tipo Uso	Msg1
37	Administrador altera ocupação atual	Dados Ocupacao Atual	Alterar Ocupacao Atual	Msg1
38	Administrador altera tipo de infraestrutura	Dados Tipo Infraestrutura	Alterar Tipo Infraestrutura	Msg1
39	Administrador altera infraestrutura	Dados Infraestrutura	Alterar Infraestrutura	Msg1
40	Administrador altera instalação	Dados Instalacao	Alterar Instalacao	Msg1
41	Administrador altera tipo processo	Dados Tipo Processo	Alterar Tipo Processo	Msg1
42	Administrador altera processo	Dados Processo	Alterar Processo	Msg1
43	Administrador altera etapa	Dados Etapa	Alterar Etapa	Msg1
44	Administrador altera tipo documento	Dados Tipo Documento	Alterar Tipo Documento	Msg1
45	Administrador altera documento processo	Dados Documento Processo	Alterar Documento Processo	Msg1
46	Administrador move o processo para em progresso	Id do Processo	Alterar Status Processo	Msg1
47	Administrador move o processo para em análise	Id do Processo	Alterar Status Processo	Msg1
48	Administrador aprova o processo	Id do Processo	Alterar Status Processo	Msg1
49	Administrador reprova o processo	Id do Processo	Alterar Status Processo	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
50	Administrador move o documento do processo para pendente	Id do Documento Processo	Alterar Status Documento Processo	Msg1
51	Administrador move o documento do processo para em análise	Id do Documento Processo	Alterar Status Documento Processo	Msg1
52	Administrador aprova o documento do processo	Id do Documento Processo	Alterar Status Documento Processo	Msg1
53	Administrador reprova o documento do processo	Id do Documento Processo	Alterar Status Documento Processo	Msg1
54	Administrador exclui estado	Id do Estado	Excluir Estado	Msg1
55	Administrador exclui cidade	Id da Cidade	Excluir Cidade	Msg1
56	Administrador exclui bairro	Id do Bairro	Excluir Bairro	Msg1
57	Administrador exclui tipo logradouro	Id do Tipo Logradouro	Excluir Tipo Logradouro	Msg1
58	Administrador exclui logradouro	Id do Logradouro	Excluir Logradouro	Msg1
59	Administrador exclui município	Id do Municípe	Excluir Municípe	Msg1
60	Administrador exclui engenheiro	Id do Engenheiro	Excluir Engenheiro	Msg1
61	Administrador exclui fiscal	Id do Fiscal	Excluir Fiscal	Msg1
62	Administrador exclui tipo usuário	Id do Tipo Usuario	Excluir Tipo Usuario	Msg1
63	Administrador exclui usuário	Id do Usuario	Excluir Usuario	Msg1
64	Administrador exclui imóvel	Id do Imovel	Excluir Imovel	Msg1
65	Administrador exclui topografia	Id da Topografia	Excluir Topografia	Msg1
66	Administrador exclui tipo de uso	Id do Tipo Uso	Excluir Tipo Uso	Msg1
67	Administrador exclui ocupação atual	Id ao Ocupacao Atual	Excluir Ocupacao Atual	Msg1
68	Administrador exclui tipo de infraestrutura	Id do Tipo Infraestrutura	Excluir Tipo Infraestrutura	Msg1
69	Administrador exclui infraestrutura	Id da Infraestrutura	Excluir Infraestrutura	Msg1
70	Administrador exclui instalação	Id da Instalacao	Excluir Instalacao	Msg1
71	Administrador exclui tipo processo	Id do Tipo Processo	Excluir Tipo Processo	Msg1
72	Administrador exclui processo	Id do Processo	Excluir Processo	Msg1
73	Administrador exclui etapa	Id da Etapa	Excluir Etapa	Msg1
74	Administrador exclui tipo documento	Id do Tipo Documento	Excluir Tipo Documento	Msg1
75	Administrador exclui documento processo	Id do Documento Processo	Excluir Documento Processo	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
76	Administrador exclui relacionamento de tipo documento com etapa	Id do TipoDocumento Etapa	Excluir TipoDocumento Etapa	Msg1
77	Administrador solicita lista de estado		Listar Estado	Msg1
78	Administrador solicita lista de cidade		Listar Cidade	Msg1
79	Administrador solicita lista de bairro		Listar Bairro	Msg1
80	Administrador solicita lista de tipo logradouro		Listar Tipo Logradouro	Msg1
81	Administrador solicita lista de logradouro		Listar Logradouro	Msg1
82	Administrador solicita lista de município		Listar Municípe	Msg1
83	Administrador solicita lista de engenheiro		Listar Engenheiro	Msg1
84	Administrador solicita lista de fiscal		Listar Fiscal	Msg1
85	Administrador solicita lista de tipo usuário		Listar Tipo Usuario	Msg1
86	Administrador solicita lista de usuário		Listar Usuario	Msg1
87	Administrador solicita lista de imóvel		Listar Imovel	Msg1
88	Administrador solicita lista de topografia		Listar Topografia	Msg1
89	Administrador solicita lista de tipo de uso		Listar Tipo Uso	Msg1
90	Administrador solicita lista de ocupação atual		Listar Ocupacao Atual	Msg1
91	Administrador solicita lista de tipo de infraestrutura		Listar Tipo Infraestrutura	Msg1
92	Administrador solicita lista de infraestrutura		Listar Infraestrutura	Msg1
93	Administrador solicita lista de instalação		Listar Instalacao	Msg1
94	Administrador solicita lista de tipo processo		Listar Tipo Processo	Msg1
95	Administrador solicita lista de processo		Listar Processo	Msg1
96	Administrador solicita lista de etapa		Listar Etapa	Msg1
97	Administrador solicita lista de tipo documento		Listar Tipo Documento	Msg1
98	Administrador solicita lista de documento processo		Listar DocumentoProcesso	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
99	Administrador solicita lista de relacionamento de tipo documento com a etapa		Listar Tipo Documento Etapa	Msg1

Fonte: Elaborado pelos autores.

No Quadro 35 apresenta a lista de casos de uso na visão do ator Secretário, com acesso semelhante ao Administrador, com exceção no processo de validação de processos e documentos e controle de usuários.

Quadro 35 – Lista de Casos de Uso: Ações do Secretário

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Secretário cadastra estado	Dados Estado	Cadastrar Estado	Msg1
02	Secretário cadastra cidade	Dados Cidade	Cadastrar Cidade	Msg1
03	Secretário cadastra bairro	Dados Bairro	Cadastrar Bairro	Msg1
04	Secretário cadastra tipo logradouro	Dados Tipo Logradouro	Cadastrar Tipo Logradouro	Msg1
05	Secretário cadastra logradouro	Dados Logradouro	Cadastrar Logradouro	Msg1
06	Secretário cadastra município	Dados Municipio	Cadastrar Municipio	Msg1
07	Secretário cadastra engenheiro	Dados Engenheiro	Cadastrar Engenheiro	Msg1
08	Secretário cadastra fiscal	Dados Fiscal	Cadastrar Fiscal	Msg1
09	Secretário cadastra imóvel	Dados Imovel	Cadastrar Imovel	Msg1
10	Secretário cadastra topografia	Dados Topografia	Cadastrar Topografia	Msg1
11	Secretário cadastra tipo de uso	Dados Tipo Uso	Cadastrar Tipo Uso	Msg1
12	Secretário cadastra ocupação atual	Dados Ocupacao Atual	Cadastrar Ocupacao Atual	Msg1
13	Secretário cadastra tipo de infraestrutura	Dados Tipo Infraestrutura	Cadastrar Tipo Infraestrutura	Msg1
14	Secretário cadastra infraestrutura	Dados Infraestrutura	Cadastrar Infraestrutura	Msg1
15	Secretário cadastra instalação	Dados Instalacao	Cadastrar Instalacao	Msg1
16	Secretário cadastra tipo processo	Dados Tipo Processo	Cadastrar Tipo Processo	Msg1
17	Secretário cadastra processo	Dados Processo	Cadastrar Processo	Msg1
18	Secretário cadastra etapa	Dados Etapa	Cadastrar Etapa	Msg1
19	Secretário cadastra tipo documento	Dados Tipo Documento	Cadastrar Tipo Documento	Msg1
20	Secretário cadastra documento processo	Dados Documento Processo	Cadastrar DocumentoProcesso	Msg1
21	Secretário relaciona tipo documento a etapa	Dados Tipo Documento Etapa	Cadastrar Tipo Documento Etapa	Msg1
22	Secretário altera estado	Dados Estado	Alterar Estado	Msg1
23	Secretário altera cidade	Dados Cidade	Alterar Cidade	Msg1
24	Secretário altera bairro	Dados Bairro	Alterar Bairro	Msg1
25	Secretário altera tipo logradouro	Dados Tipo Logradouro	Alterar Tipo Logradouro	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
26	Secretário altera logradouro	Dados Logradouro	Alterar Logradouro	Msg1
27	Secretário altera município	Dados Município	Alterar Município	Msg1
28	Secretário altera engenheiro	Dados Engenheiro	Alterar Engenheiro	Msg1
29	Secretário altera fiscal	Dados Fiscal	Alterar Fiscal	Msg1
30	Secretário altera imóvel	Dados Imovel	Alterar Imovel	Msg1
31	Secretário altera topografia	Dados Topografia	Alterar Topografia	Msg1
32	Secretário altera tipo de uso	Dados Tipo Uso	Alterar Tipo Uso	Msg1
33	Secretário altera ocupação atual	Dados Ocupacao Atual	Alterar Ocupacao Atual	Msg1
34	Secretário altera tipo de infraestrutura	Dados Tipo Infraestrutura	Alterar Tipo Infraestrutura	Msg1
35	Secretário altera infraestrutura	Dados Infraestrutura	Alterar Infraestrutura	Msg1
36	Secretário altera instalação	Dados Instalacao	Alterar Instalacao	Msg1
37	Secretário altera tipo processo	Dados Tipo Processo	Alterar Tipo Processo	Msg1
38	Secretário altera processo	Dados Processo	Alterar Processo	Msg1
39	Secretário altera etapa	Dados Etapa	Alterar Etapa	Msg1
40	Secretário altera tipo documento	Dados Tipo Documento	Alterar Tipo Documento	Msg1
41	Secretário altera documento processo	Dados Documento Processo	Alterar Documento Processo	Msg1
42	Secretário move o processo para em progresso	Id do Processo	Alterar Status Processo	Msg1
43	Secretário move o processo para em análise	Id do Processo	Alterar Status Processo	Msg1
44	Secretário move o documento do processo para pendente	Id do Documento Processo	Alterar Status Documento Processo	Msg1
45	Secretário move o documento do processo para em análise	Id do Documento Processo	Alterar Status Documento Processo	Msg1
46	Secretário exclui estado	Id do Estado	Excluir Estado	Msg1
47	Secretário exclui cidade	Id da Cidade	Excluir Cidade	Msg1
48	Secretário exclui bairro	Id do Bairro	Excluir Bairro	Msg1
49	Secretário exclui tipo logradouro	Id do Tipo Logradouro	Excluir Tipo Logradouro	Msg1
50	Secretário exclui logradouro	Id do Logradouro	Excluir Logradouro	Msg1
51	Secretário exclui município	Id do Município	Excluir Município	Msg1
52	Secretário exclui engenheiro	Id do Engenheiro	Excluir Engenheiro	Msg1
53	Secretário exclui fiscal	Id do Fiscal	Excluir Fiscal	Msg1
54	Secretário exclui imóvel	Id do Imovel	Excluir Imovel	Msg1
55	Secretário exclui topografia	Id da Topografia	Excluir Topografia	Msg1
56	Secretário exclui tipo de uso	Id do Tipo Uso	Excluir Tipo Uso	Msg1
57	Secretário exclui ocupação atual	Id ao Ocupacao Atual	Excluir Ocupacao Atual	Msg1
58	Secretário exclui tipo de infraestrutura	Id do Tipo Infraestrutura	Excluir Tipo Infraestrutura	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
59	Secretário exclui infraestrutura	Id da Infraestrutura	Excluir Infraestrutura	Msg1
60	Secretário exclui instalação	Id da Instalacao	Excluir Instalacao	Msg1
61	Secretário exclui tipo processo	Id do Tipo Processo	Excluir Tipo Processo	Msg1
62	Secretário exclui processo	Id do Processo	Excluir Processo	Msg1
63	Secretário exclui etapa	Id da Etapa	Excluir Etapa	Msg1
64	Secretário exclui tipo documento	Id do Tipo Documento	Excluir Tipo Documento	Msg1
65	Secretário exclui documento processo	Id do Documento Processo	Excluir Documento Processo	Msg1
66	Secretário exclui relacionamento de tipo documento com etapa	Id do Tipo Documento Etapa	Excluir Tipo Documento Etapa	Msg1
67	Secretário solicita lista de estado		Listar Estado	Msg1
68	Secretário solicita lista de cidade		Listar Cidade	Msg1
69	Secretário solicita lista de bairro		Listar Bairro	Msg1
70	Secretário solicita lista de tipo logradouro		Listar Tipo Logradouro	Msg1
71	Secretário solicita lista de logradouro		Listar Logradouro	Msg1
72	Secretário solicita lista de município		Listar Municípe	Msg1
73	Secretário solicita lista de engenheiro		Listar Engenheiro	Msg1
74	Secretário solicita lista de fiscal		Listar Fiscal	Msg1
75	Secretário solicita lista de imóvel		Listar Imovel	Msg1
76	Secretário solicita lista de topografia		Listar Topografia	Msg1
77	Secretário solicita lista de tipo de uso		Listar Tipo Uso	Msg1
78	Secretário solicita lista de ocupação atual		Listar Ocupacao Atual	Msg1
79	Secretário solicita lista de tipo de infraestrutura		Listar Tipo Infraestrutura	Msg1
80	Secretário solicita lista de infraestrutura		Listar Infraestrutura	Msg1
81	Secretário solicita lista de instalação		Listar Instalacao	Msg1
82	Secretário solicita lista de tipo processo		Listar Tipo Processo	Msg1
83	Secretário solicita lista de processo		Listar Processo	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
84	Secretário solicita lista de etapa		Listar Etapa	Msg1
85	Secretário solicita lista de tipo documento		Listar Tipo Documento	Msg1
86	Secretário solicita lista de documento processo		Listar Documento Processo	Msg1
87	Secretário solicita lista de relacionamento de tipo documento com a etapa		Listar Tipo Documento Etapa	Msg1

Fonte: Elaborado pelos autores.

E por último no Quadro 36 detalha-se a lista de casos de uso na visão do ator Estagiário, com acesso somente a esfera de controle de endereços.

Quadro 36 – Lista de Casos de Uso: Ações do Estagiário

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Estagiário cadastra estado	Dados Estado	Cadastrar Estado	Msg1
02	Estagiário cadastra cidade	Dados Cidade	Cadastrar Cidade	Msg1
03	Estagiário cadastra bairro	Dados Bairro	Cadastrar Bairro	Msg1
04	Estagiário cadastra tipo logradouro	Dados Tipo Logradouro	Cadastrar Tipo Logradouro	Msg1
05	Estagiário cadastra logradouro	Dados Logradouro	Cadastrar Logradouro	Msg1
06	Estagiário altera estado	Dados Estado	Alterar Estado	Msg1
07	Estagiário altera cidade	Dados Cidade	Alterar Cidade	Msg1
08	Estagiário altera bairro	Dados Bairro	Alterar Bairro	Msg1
09	Estagiário altera tipo logradouro	Dados Tipo Logradouro	Alterar Tipo Logradouro	Msg1
10	Estagiário altera logradouro	Dados Logradouro	Alterar Logradouro	Msg1
11	Estagiário solicita lista de estado		Listar Estado	Msg1
12	Estagiário solicita lista de cidade		Listar Cidade	Msg1
13	Estagiário solicita lista de bairro		Listar Bairro	Msg1
14	Estagiário solicita lista de tipo logradouro		Listar Tipo Logradouro	Msg1
15	Estagiário solicita lista de logradouro		Listar Logradouro	Msg1

Fonte: Elaborado pelos autores.

Quadro 37 – Lista de Casos de Uso: Ações do Sistema

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Sistema carrega estado		Carregar Estado	Msg1
02	Sistema carrega cidade		Carregar Cidade	Msg1
03	Sistema carrega bairro		Carregar Bairro	Msg1

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
04	Sistema carrega tipo logradouro		Carregar Tipo Logradouro	Msg1
05	Sistema carrega logradouro		Carregar Logradouro	Msg1
06	Sistema carrega município		Carregar Municipio	Msg1
07	Sistema carrega engenheiro		Carregar Engenheiro	Msg1
08	Sistema carrega fiscal		Carregar Fiscal	Msg1
09	Sistema carrega tipo usuário		Carregar Tipo Usuario	Msg1
10	Sistema carrega usuário		Carregar Usuario	Msg1
11	Sistema carrega imóvel		Carregar Imovel	Msg1
12	Sistema carrega topografia		Carregar Topografia	Msg1
13	Sistema carrega tipo de uso		Carregar Tipo Uso	Msg1
14	Sistema carrega ocupação atual		Carregar Ocupacao Atual	Msg1
15	Sistema carrega tipo de infraestrutura		Carregar Tipo Infraestrutura	Msg1
16	Sistema carrega infraestrutura		Carregar Infraestrutura	Msg1
17	Sistema carrega instalação		Carregar Instalacao	Msg1
18	Sistema carrega tipo processo		Carregar Tipo Processo	Msg1
19	Sistema carrega processo		Carregar Processo	Msg1
20	Sistema carrega etapa		Carregar Etapa	Msg1
21	Sistema carrega tipo documento		Carregar Tipo Documento	Msg1
22	Sistema carrega documento processo		Carregar Documento Processo	Msg1
23	Sistema carrega os relacionamentos de tipo documento com etapa		Carregar Tipo Documento Etapa	Msg1

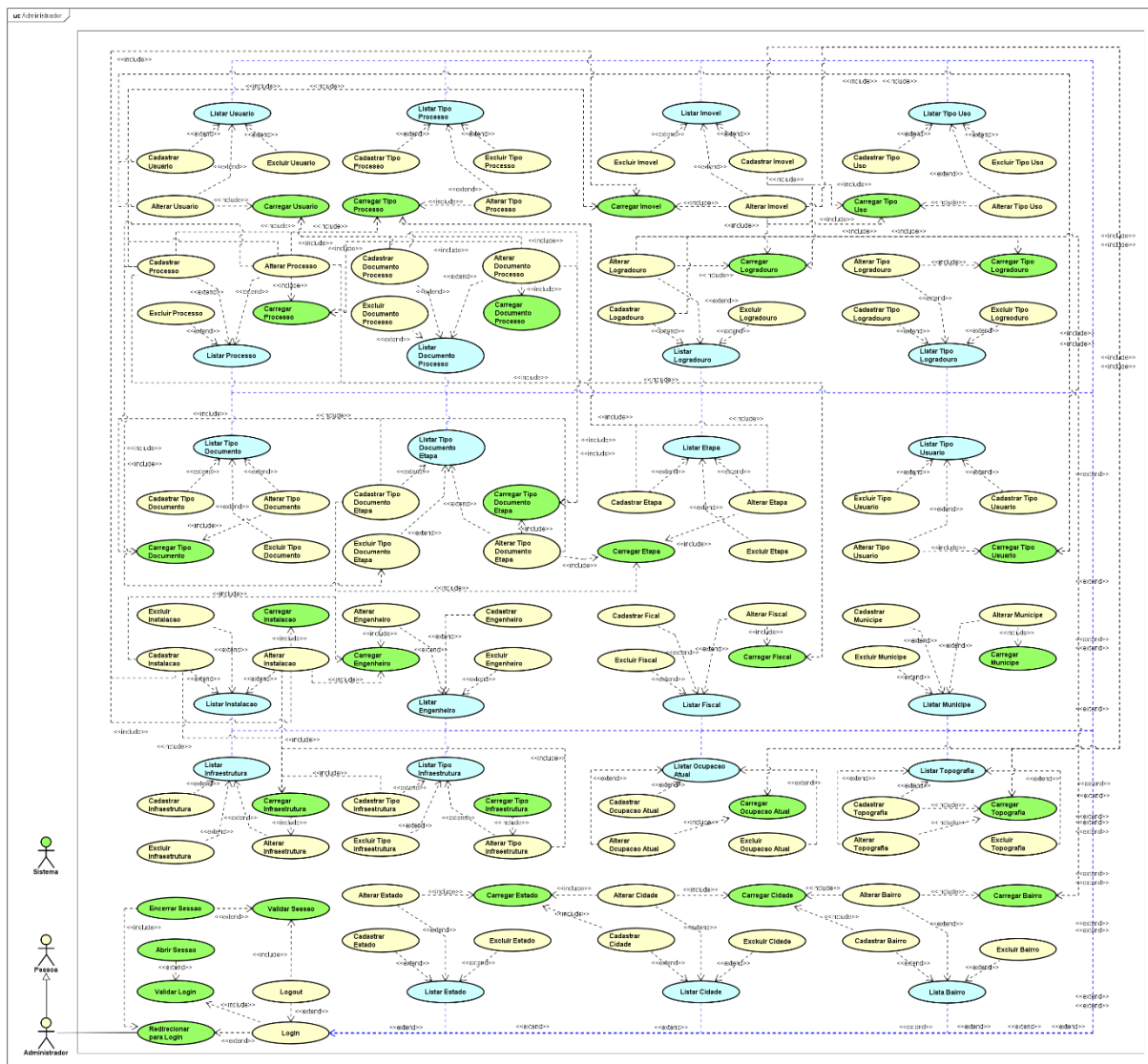
Fonte: Elaborado pelos autores.

3.4. DIAGRAMA DE CASOS DE USO

O Diagrama de Casos de Uso é uma ferramenta essencial dentro da Engenharia de Software, usada para representar visualmente as interações entre os atores externos e o sistema. Ele fornece uma visão clara das funcionalidades que o sistema deve oferecer, facilitando o entendimento tanto por desenvolvedores quanto por usuários. De acordo com Pressman e Maxim (2021), o diagrama de casos de uso representa o comportamento de um sistema sob a visão de um usuário externo, sendo uma ferramenta essencial para análise de requisitos.

A Figura 7 apresenta os diagramas de casos de uso individuais, destacando as interações e funcionalidades específicas disponíveis para o administrador do sistema. Esses diagramas demonstram de forma clara as ações que podem ser realizadas por esse perfil, evidenciando as responsabilidades e os processos que envolvem a administração no contexto da aplicação.

Figura 6 – Diagrama de Caso de Uso Geral: Visão do Administrador



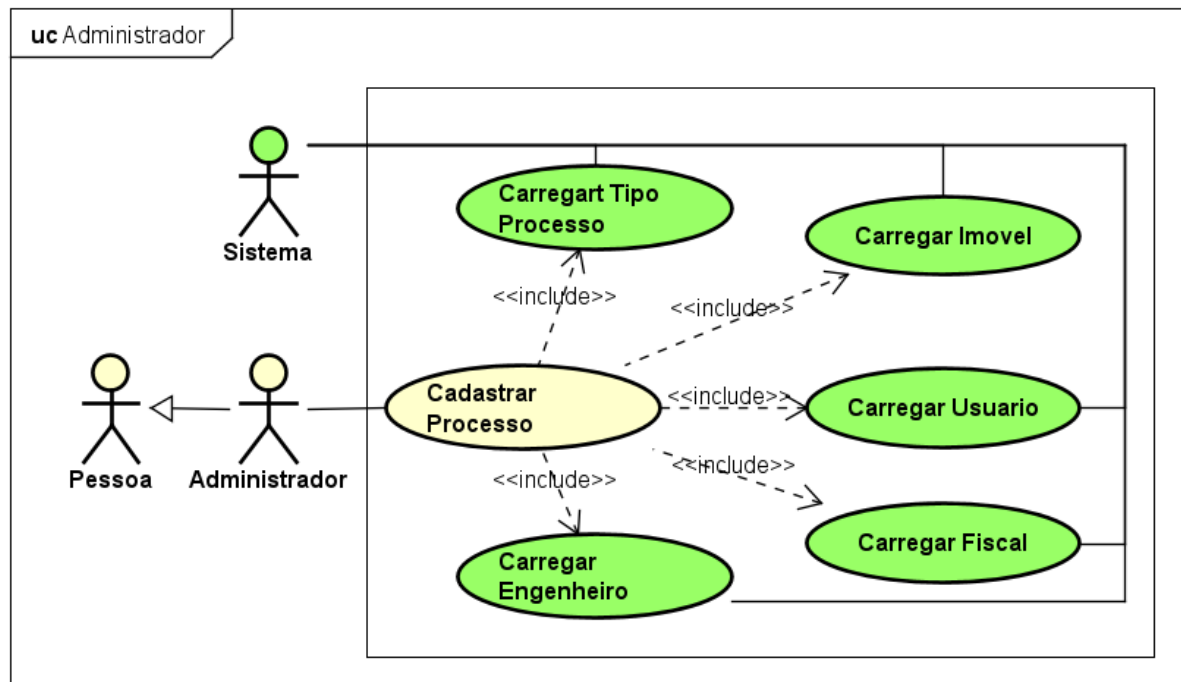
Fonte: Elaborado pelos autores.

3.5. DIAGRAMA DE CASOS DE USO INDIVIDUAIS

De acordo com Pressman e Maxim (2021), um caso de uso específico descreve uma interação específica entre um ator e o sistema para realizar uma tarefa ou atingir um objetivo concreto. Neste contexto, serão abordados dois dos casos de uso, extraídos do diagrama de casos de uso do administrador, que exemplificam as responsabilidades desse ator no sistema.

3.5.1 – Caso de uso: Administrador - Cadastrar Processo

Na Figura 8 apresenta-se o diagrama de caso de uso que descreve a funcionalidade de Cadastrar Processo executada pelo administrador. Neste caso, o administrador interage com o sistema para cadastrar informações do processo.

Figura 7 – Diagrama de Caso de Uso Específico: Administrador – Cadastrar Processo

Fonte: Elaborado pelos autores.

No Quadro 37 descreve-se o procedimento para o caso de uso Cadastrar Processo, especificando as fases que um administrador realiza ao interagir com o sistema para inserir ou modificar informações de processo. Este procedimento assegura o registro adequado das informações, assegurando desta forma a integridade dos dados administrativos do sistema.

Quadro 38 – Fluxo do Caso de Uso: Administrador – Cadastrar Processo

Fluxo do Caso de Uso	
Caso de Uso:	Cadastrar Processo
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrador deve passar e/ou pode passar para concluir a função de cadastrar o processo.
Pré-condições:	Administrador logado e autenticado.
Pós-condições:	O sistema fecha o formulário e lista os processos.
Fluxo Normal	
Ações do Administrador	Ações do Sistema
1. Administrador requisita formulário de cadastro de processo.	
	2. Sistema exibe modal de cadastro de processo com dados de Tipo Processo,

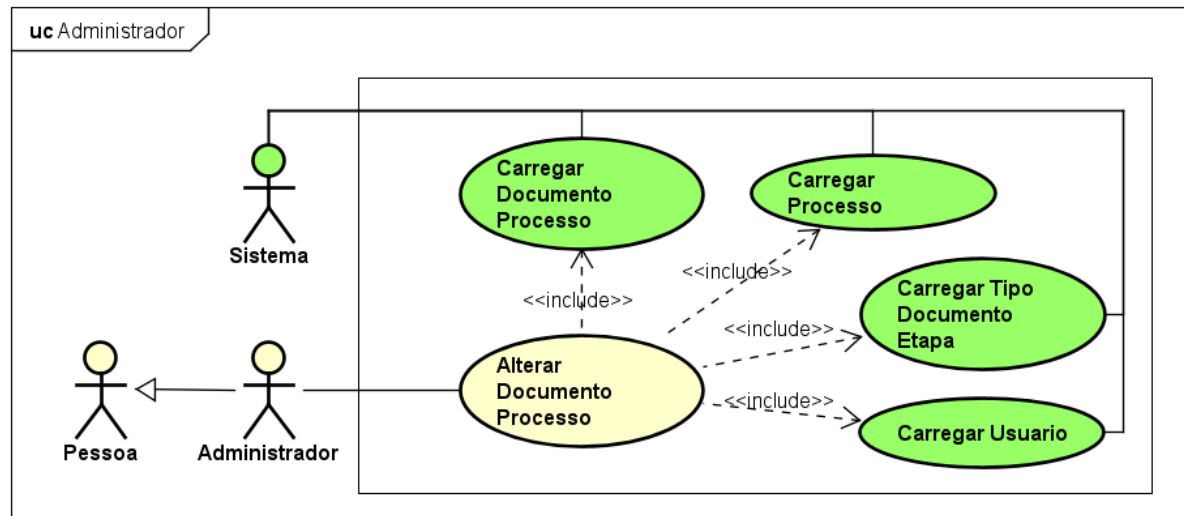
	Imóvel, Usuário, Fiscal e Engenheiro carregados.
3. Administrador requisita cadastro de processo.	
	4. Sistema valida os dados de entrada do processo.
	5. Sistema salva dados do processo no banco de dados.
	6. Sistema fecha modal de cadastro de processo.
	7. Sistema recarrega a lista de processos.
	8. Sistema exibe a lista de processos. Sistema exibe popup com Msg1.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazios. Sistema exibe Msg2
	4.2 Dados de entrada inválidos. Sistema exibe Msg2
	5.1 Não é possível estabelecer conexão com a API. Sistema exibe Msg7.
	5.2 Não é possível estabelecer conexão com o Banco de Dados. Sistema exibe Msg7.

Fonte: Elaborado pelos autores.

3.5.2 – Caso de uso: Administrador - Cadastrar Documento Processo

Na Figura 9 é apresentado o diagrama de caso de uso da funcionalidade Alterar Documento Processo, onde o administrador se comunica com o sistema informando o id do Documento Processo para obter seus dados. Este caso de uso é crucial para definir como o administrador pode alterar os dados do Documento Processo sem afetar sua integridade e relacionamento.

Figura 8 – Diagrama de Caso de Uso Específico: Administrador – Alterar Documento Processo



Fonte: Elaborado pelos autores.

No Quadro 38 apresenta-se o caso de uso Cadastrar Processo, especificando as fases que um administrador realiza ao interagir com o sistema para inserir ou modificar informações dos usuários. Este procedimento assegura o registro adequado das informações, assegurando desta forma a integridade dos dados administrativos do sistema.

Quadro 39 – Fluxo do Caso de Uso: Adminsitrador – Alterar Documento Processo

Fluxo do Caso de Uso	
Caso de Uso:	Alterar Documento Processo
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrador deve passar e/ou pode passar para concluir a função de alterar o documento do processo.
Pré-condições:	Administrador logado e autenticado.
Pós-condições:	O sistema fecha o formulário e lista os estados.
Fluxo Normal	
Ações do Administrador	Ações do Sistema

9. Administrador requisita formulário de edição do documento do processo, passado o id por url.	
	10. Sistema exibe modal de edição do documento do processo, trazendo os dados relacionados ao mesmo, bem como do Processo, do Tipo Documento Etapa e Usuário carregados.
11. Administrador requisita alteração do documento do processo.	
	12. Sistema valida os dados de entrada do documento do processo.
	13. Sistema salva dados do documento do processo no banco de dados.
	14. Sistema fecha modal de edição do documento do processo.
	15. Sistema recarrega a lista de documentos dos processos.
	16. Sistema exibe a lista de documentos dos processos. Sistema exibe popup com Msg1.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg2.
	4.2 Dados de entrada inválidos. Sistema exibe Msg2.
	5.1 Não é possível estabelecer conexão com a API. Sistema exibe Msg7.
	5.2 Não é possível estabelecer conexão com o Banco de Dados. Sistema exibe Msg7.

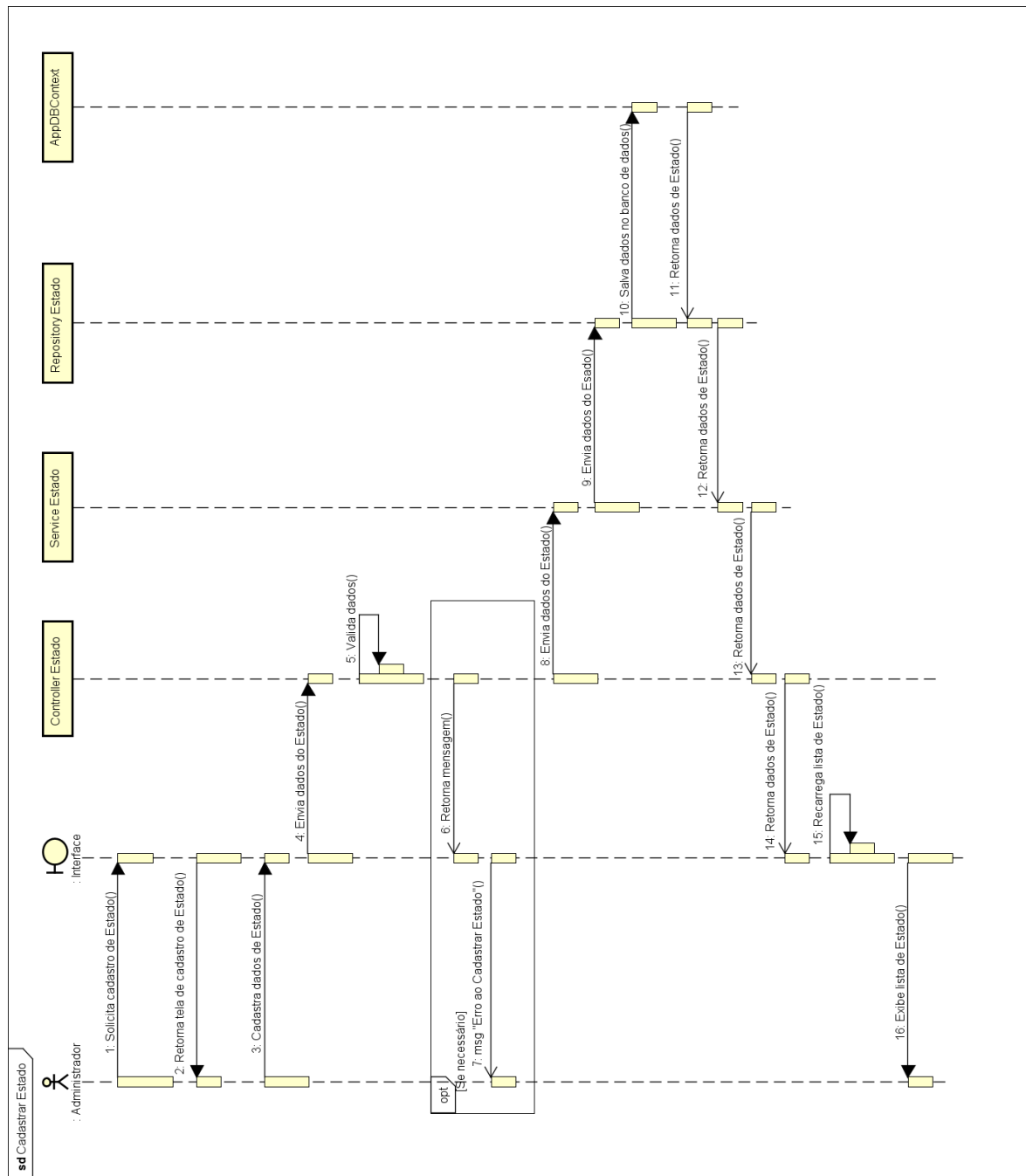
Fonte: Elaborado pelos autores.

3.6. DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência é um dos principais diagramas UML (Unified Modeling Language), utilizado para descrever as interações entre objetos em um sistema, com ênfase na sequência temporal dessas interações. De acordo com Sommerville (2018), esse tipo de diagrama organiza as interações entre os atores e o sistema, bem como entre os componentes internos, em ordem cronológica. Trata-se de uma ferramenta essencial para compreender como as funções são executadas por meio da troca de mensagens entre os componentes envolvidos.

3.6.1 DIAGRAMA DE SEQUÊNCIA PARA CADASTRO DE ESTADO

Na Figura 12 são detalhados os processos para o cadastro de um Estado. O administrador inicia o cadastro através da interface, que solicita a operação e retorna a lista de estados. A interface envia os dados ao *Service Estado*, que os valida e encaminha ao *Repository Estado*. O *Repository* interage com o *AppDBContext* para salvar os dados no banco de dados e retorna uma mensagem de confirmação. O *Service Estado* pode realizar operações adicionais e retorna os dados processados ao *Repository Estado*, que recupera a lista atualizada de estados. Finalmente, o *Service Estado* envia a lista de estados atualizada à interface, confirmando o sucesso da operação ao administrador.

Figura 9 – Diagrama de Sequência - Ator Administrador: Fluxo do cadastro de Estado

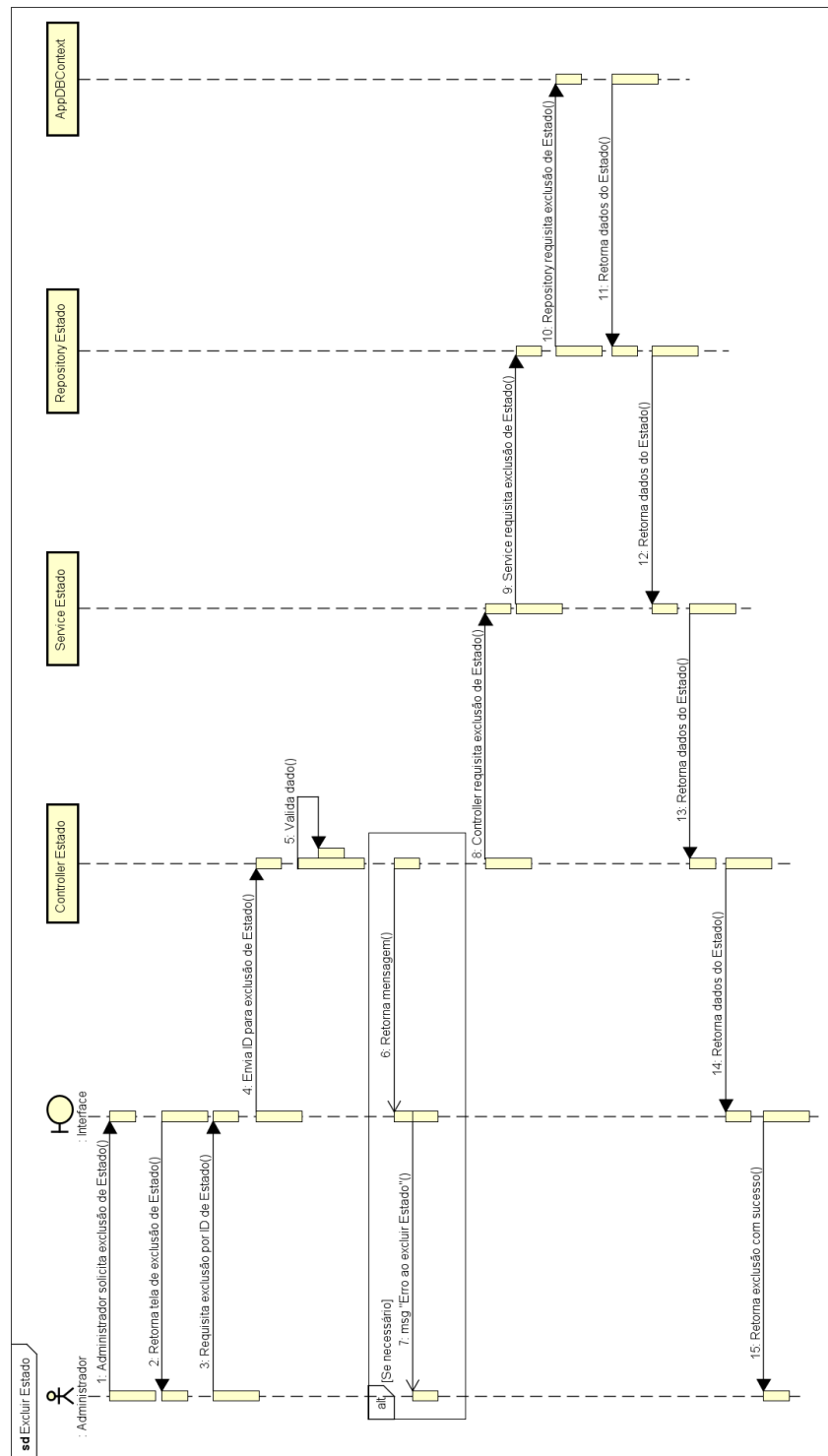
Fonte: Elaborado pelos Autores.

3.6.2 DIAGRAMA DE SEQUÊNCIA PARA EXCLUSÃO DE ESTADO

Na Figura 13 apresenta os processos e etapas necessários para a exclusão de um Estado. O processo começa com o administrador solicitando a exclusão através da interface, que retorna a tela correspondente. A solicitação é enviada ao *Controller* Estado, que valida os dados. Se houver erros, uma mensagem é retornada. Caso contrário, o *Controller* encaminha a solicitação ao *Service* Estado, que a repassa ao *Repository* Estado. Este interage com o

AppDbContext para realizar a exclusão no banco de dados. Após a exclusão, os dados são retornados através dos componentes, confirmando ao administrador que a operação foi realizada com sucesso. Este diagrama ilustra claramente as interações e a ordem das operações envolvidas.

Figura 10 – Diagrama de Sequência - Ator Administrador: Fluxo do excluir de Estado



Fonte: Elaborado pelos autores.

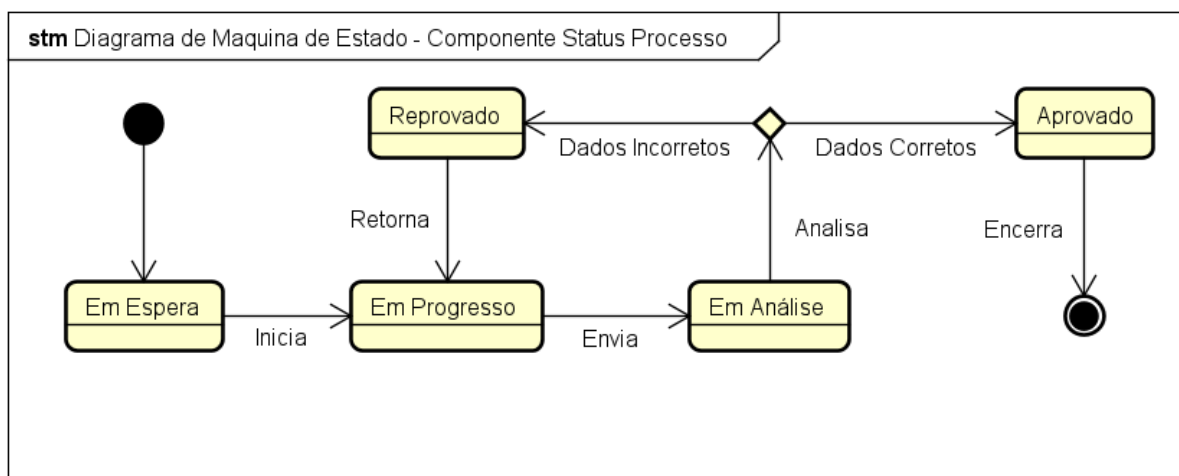
3.7. DIAGRAMA DE MÁQUINA DE ESTADO

Segundo Guedes (2011), o diagrama de máquina de estados é uma representação gráfica que modela o comportamento de um sistema, ilustrando os estados pelos quais ele pode passar e as transições entre esses estados em resposta a eventos específicos. Ele é amplamente utilizado na UML para descrever a lógica de sistemas dinâmicos, sendo particularmente eficaz para sistemas que apresentam comportamentos sequenciais ou interdependentes. Nesse diagrama, os estados representam condições em que o sistema ou um objeto pode estar, enquanto as transições indicam como ele muda de um estado para outro devido a estímulos, como eventos externos, temporizadores ou ações internas (Guedes, 2011).

Sob a perspectiva de Sommerville (2018), o diagrama de máquina de estados é especialmente útil na especificação de sistemas interativos ou embarcados, pois detalha como os componentes respondem a eventos e mantém a consistência ao longo de suas operações. Ele destaca que esse modelo ajuda a identificar potenciais gargalos, erros de lógica e situações de exceção, fornecendo uma base sólida para o desenvolvimento de software de alta qualidade. Além disso, é uma ferramenta importante para comunicação entre analistas e desenvolvedores, permitindo que as decisões sobre comportamentos esperados sejam visualizadas e ajustadas de maneira colaborativa.

Na Figura 11 é mostrado os diferentes estados que o processo pode assumir, de acordo com seu status atual.

Figura 11 – Diagrama de Máquina de Estado - Status do Processo



Fonte: Elaborado pelos autores.

4 DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX)

Segundo Teixeira (2014), o conceito de UX, apesar de sua origem estrangeira, é mais simples do que parece: refere-se à experiência do usuário. No cotidiano, nos tornamos usuários de diversos objetos e produtos – digitais ou físicos – que são projetados para cumprir determinadas funções, como o alarme do celular, o caixa eletrônico e a cadeira, cada um com seu propósito específico de uso.

4.1 DESCRIÇÃO DE CENÁRIO

A técnica de descrição do cenário envolve a criação de uma narrativa detalhada que descreve a interação potencial do usuário com o seu produto, sistema ou serviço em um contexto específico. A construção desse cenário visa proporcionar uma compreensão mais clara de como será a experiência do usuário ao utilizar a interface, bem como quando interagir em diversas situações da vida real. Frequentemente, aplicamos essa técnica sem perceber que tem um nome específico. Quando pensamos “suponha que o usuário faça isso, então...”, estamos, na verdade, criando uma descrição de cenário que representa situações que os usuários do sistema podem encontrar. Conforme Preece, Rogers e Sharp (2015), os cenários são amplamente usados no design de interação para explorar e avaliar as possíveis experiências dos usuários, permitindo um entendimento mais profundo de suas necessidades e contextos de uso.

Segundo AMSTEL (2007), as técnicas de descrição de cenário possuem vantagens, como o engajamento e a conscientização da equipe que está desenvolvendo o projeto, permite o foco no usuário durante todo o projeto, facilita a tomada de decisões. A seguir, serão apresentados dois exemplos de descrição de cenários:

A Figura 14 apresenta uma engenheira de obras da Prefeitura de Jales. Com prazos rigorosos e a necessidade de garantir a conformidade documental, sistemas modernos otimizam processos, permitindo acesso rápido a arquivos, revisões eficientes e organização centralizada. O uso desses sistemas reduz discrepâncias e melhora a qualidade das operações no setor de engenharia pública.

Figura 12 – Cenário Engenheira de Obras da Prefeitura

Maria, Engenheira de Obras da Prefeitura de Jales



Maria, engenheira na Prefeitura de Jales, recebe uma notificação urgente para revisar documentos cruciais de uma obra. Utiliza o sistema de gerenciamento de documentos, acessa facilmente os arquivos, corrige discrepâncias e deixa comentários. Conclui a revisão, registra suas ações e segue sua rotina, confiante na eficiência do sistema para agilizar processos de aprovação e garantir a conformidade documental.

Fonte: Elaborado pelos autores.

Na Figura 15 apresenta um fiscal de obras da Prefeitura de Jales, a tecnologia desempenha um papel essencial. Sistemas de gerenciamento de documentos possibilitam agilidade no acesso a registros, identificação de alterações em projetos e sincronização de dados em tempo real. Esses recursos aumentam a eficiência das inspeções e contribuem para a segurança e qualidade das construções públicas.

Figura 13 – Cenário Engenheira de Obras da Prefeitura

José, Fiscal de Obras da Prefeitura de Jales



Cenário 2

José, um fiscal de obras, utiliza o sistema de gerenciamento de documentos para realizar uma inspeção não programada em uma obra. Acessa rapidamente os registros, identifica mudanças no projeto e usa o aplicativo móvel durante a inspeção para documentar não conformidades em tempo real. Ao retornar ao escritório, sincroniza os dados, garantindo uma gestão eficiente e atualizada das obras municipais. O sistema agiliza suas responsabilidades e contribui para a qualidade e segurança das construções.

Fonte: Elaborado pelos autores.

4.2 DESCRIÇÃO DE PERSONAS

A descrição de personas consiste em representações fictícias baseadas em usuários reais, desenvolvidas para auxiliar designers e equipes de UX (User Experience) na compreensão assertiva das necessidades, comportamentos e características dos usuários. Segundo Lisboa (2017), o uso de personas permite que os negócios sejam mais estratégicos ao alcançar seu público, pois elas ajudam a comunicar de forma clara as características do público-alvo para todos os stakeholders, incluindo a equipe de design. Como parte deste estudo, foram desenvolvidos dois exemplos de personas para ilustrar o sistema abordado nesta pesquisa.

Na Figura 16 é apresentado uma engenheira civil experiente, atuando como Coordenadora de Projetos na Prefeitura de Jales há cinco anos. Com 35 anos, ela demonstra um forte compromisso com o desenvolvimento sustentável e busca otimizar projetos de infraestrutura e construção na cidade. Sua paixão pela inovação e pelo uso estratégico da tecnologia reflete seu foco em processos eficientes e sustentáveis, enfrentando desafios relacionados à gestão simultânea de projetos e à conformidade com regulamentações municipais.

Figura 14 – Persona Engenheira Civil



Ana Silva

Engenheira Civil

- Idade: 35 anos
- Gênero: Feminino
- Localidade(cidade): Jales
- Profissão: Coordenadora de Projetos de Obra
- Familiaridade com tecnologia (0 a 10): 8

Bio

Ana, é uma engenheira civil experiente e atua como Coordenadora de Projetos na Prefeitura de Jales há cinco anos. Ela é apaixonada por facilitar o desenvolvimento sustentável da cidade por meio de projetos de infraestrutura e construção.

Desafios e Objetivos

Gerenciar vários projetos de construção simultaneamente, garantir eficiência na aprovação de documentos para evitar atrasos e manter a conformidade com os regulamentos municipais são os pilares fundamentais para uma administração bem-sucedida de projetos de construção. Esses desafios destacam a importância de uma abordagem precisa e diligente em cada etapa do processo.

Fonte: Elaborado pelos autores.

A Figura 17 apresenta engenheiro civil dedicado, com 30 anos e uma trajetória de três anos como Engenheiro de Campo na Prefeitura de Jales. Ele se destaca pelo seu trabalho prático em obras de infraestrutura urbana, unindo paixão pela construção a uma abordagem técnica e

eficiente. Carlos se concentra em coordenar projetos com foco na segurança, qualidade e na implementação de ferramentas que melhorem os processos de campo.

Figura 15 – Persona Engenheiro de Campo



Fonte: Elaborado pelos autores.

4.3 ESBOÇOS DE TELA (WIREFRAMES)

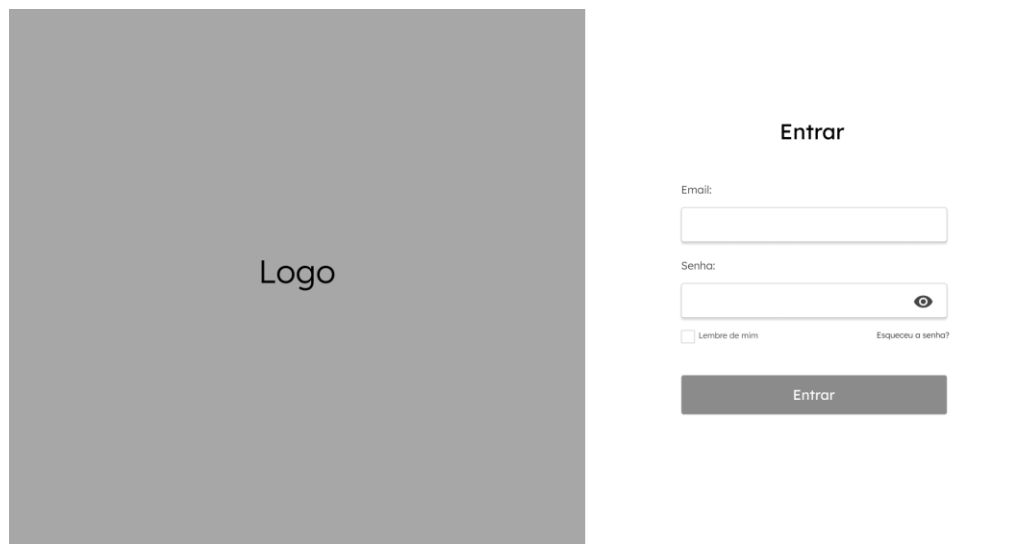
De acordo com Krug (2014), os *wireframes* desempenham um papel importante no *design* de interfaces ao estruturar informações e priorizar a funcionalidade e a clareza. Eles ajudam a evitar distrações visuais precoces, focando no que realmente importa para a experiência do usuário. Serve como guia para o layout e a organização dos elementos na tela, sem incluir gráficos, cores ou quaisquer estilos visuais. O *wireframe* destaca a arquitetura e a disposição dos principais componentes sem se preocupar com o design final.

Krug (2014) destaca que *wireframes* são ferramentas valiosas para alinhar ideias entre os envolvidos em um projeto. Antes de partir para detalhes visuais, eles permitem validar conceitos iniciais e garantir que a estrutura da interface esteja clara para todos. Sendo útil para a comunicação de conceitos, obter feedback inicial e garantir que todos os stakeholders tenham uma compreensão clara da arquitetura da interface antes que o trabalho de design visual mais detalhado comece.

Segundo AWARI (2022), a utilidade do *wireframe* não se resume somente no começo do projeto, pois quando houver a necessidade de realizar teste e validação de uma nova funcionalidade, o *wireframe* se torna uma técnica valiosa para definir a melhor solução ao cliente ou usuário.

Na Figura 18 é apresentado a tela de *login* é a porta de entrada para muitas interações online, e seu design desempenha um papel crucial na experiência do usuário. Neste contexto, o *wireframe* criado para a tela de login foi concebido com uma abordagem centrada no usuário, visando simplificar o processo de autenticação e oferecer recursos que promovem eficiência e usabilidade.

Figura 16 – Wireframe da Tela de Login



O wireframe da tela de login é dividido em duas seções principais. À esquerda, há um retângulo cinza escuro contendo o texto "Logo" no centro. À direita, o formulário de login é apresentado. No topo da seção direita, o título "Entrar" está centralizado. Abaixo dele, o rótulo "Email:" precede um campo de entrada retangular. Logo abaixo, o rótulo "Senha:" precede um campo de entrada retangular com um ícone de olho para alternar a visibilidade da senha. Abaixo do campo de senha, há duas opções: "Lembre de mim" com uma caixa de seleção vazia e "Esqueceu a senha?" com um link. No final da seção, um botão cinza escuro com o texto "Entrar" em branco está centralizado.

Fonte: Elaborado pelos autores.

Os campos “Email” e “Senha” tem o objetivo de facilitar o acesso do usuário ao sistema. O design decisivo inclui posicionamento claro e espaçamento adequado para uma entrada de dados intuitiva, com a escolha de fontes legíveis contribuindo para uma experiência sem atritos.

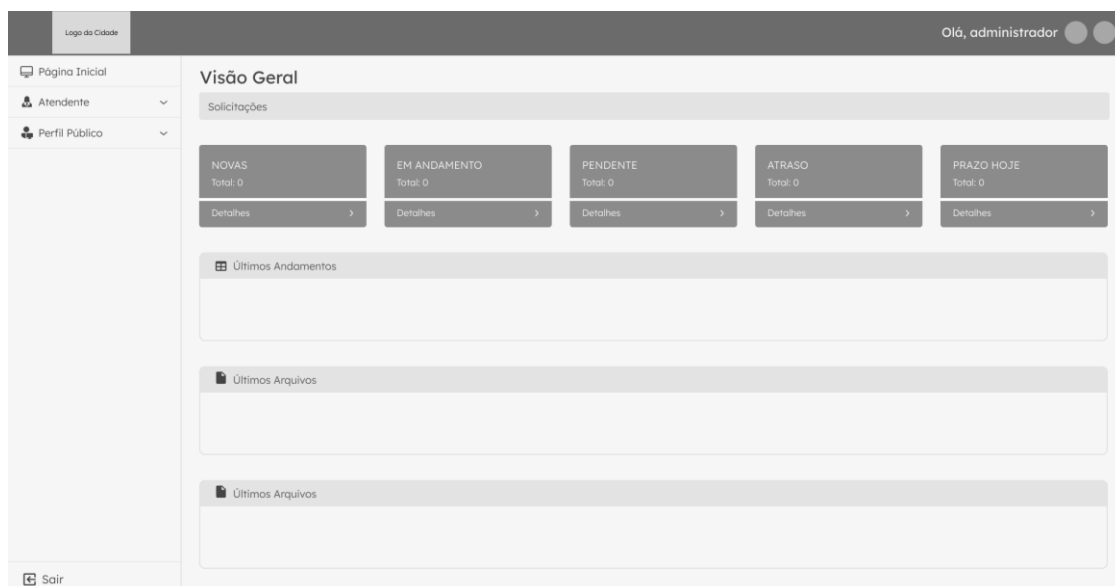
A opção “Lembre de Mim” visa oferecer conveniência para usuários frequentes, a opção é estrategicamente posicionada para visibilidade, com uma seleção clara. O design amigável incentiva os usuários a manterem-se autenticados para futuras sessões.

A opção “Esqueceu a senha?” possui o objetivo de facilitar a recuperação de conta em caso de esquecimento de senha, a opção recebe destaque visual. Isso incentiva ações proativas dos usuários em situações de senha esquecida, com um fluxo de recuperação de senha claro e acessível.

O botão “Entrar” tem o objetivo é iniciar o processo de autenticação. O design destaca visualmente o botão, com cores e texto que indicam claramente sua função. Além disso, há feedback visual após a ação para indicar progresso.

A Figura 19 apresenta a tela inicial a busca pela excelência na experiência do usuário é evidente através do *wireframe* cuidadosamente elaborado. Cada elemento, desde o *Navbar* até o *Dashboard*, foi projetado para proporcionar uma interação intuitiva e informativa.

Figura 17 – Wireframe da Tela Inicial do Sistema



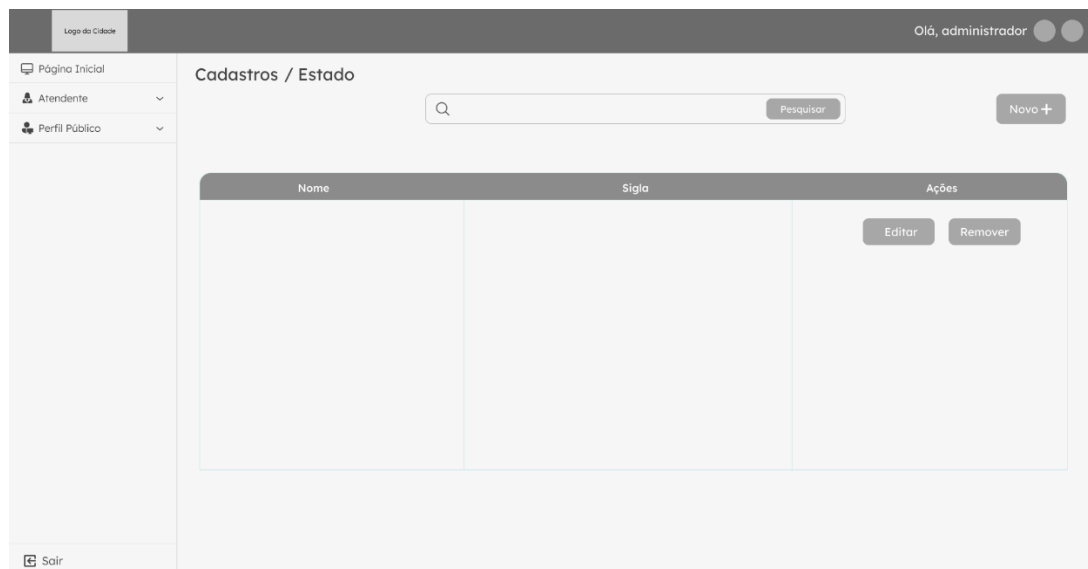
Fonte: Elaborado pelos autores.

No topo da tela, o *Navbar* apresenta elementos essenciais, proporcionando uma navegação simplificada e personalizada. A presença da logo do sistema, opções de perfil, notificações, promove uma experiência centrada no usuário desde o início.

À esquerda, o *Sidebar* oferece opções de navegação, constituindo uma ferramenta contextual e eficiente para explorar diferentes áreas do sistema. Essa abordagem facilita a descoberta de funcionalidades, proporcionando aos usuários uma compreensão clara da estrutura e organização do sistema.

No centro da tela, o *Dashboard* surge como um centro de informações, apresentando de forma visualmente apelativa os últimos arquivos cadastrados no sistema. A inclusão de categorias como "Novas Atualizações", "Em Andamento", "Pendentes", "Em Atraso", e "Prazo Hoje" oferece uma visão instantânea do status e progresso das atividades, fornecendo insights valiosos ao usuário.

A Figura 20 apresenta-se o *wireframe* tela de cadastro de estado oferece diversas funcionalidades. A barra de pesquisa se destaca como uma ferramenta vital para navegação eficiente. Permitindo a busca de estados específicos, os usuários têm acesso rápido às informações desejadas, simplificando a interação com o sistema.

Figura 18 – Wireframe Tela de Cadastro de Estado

Fonte: Elaborado pelos autores.

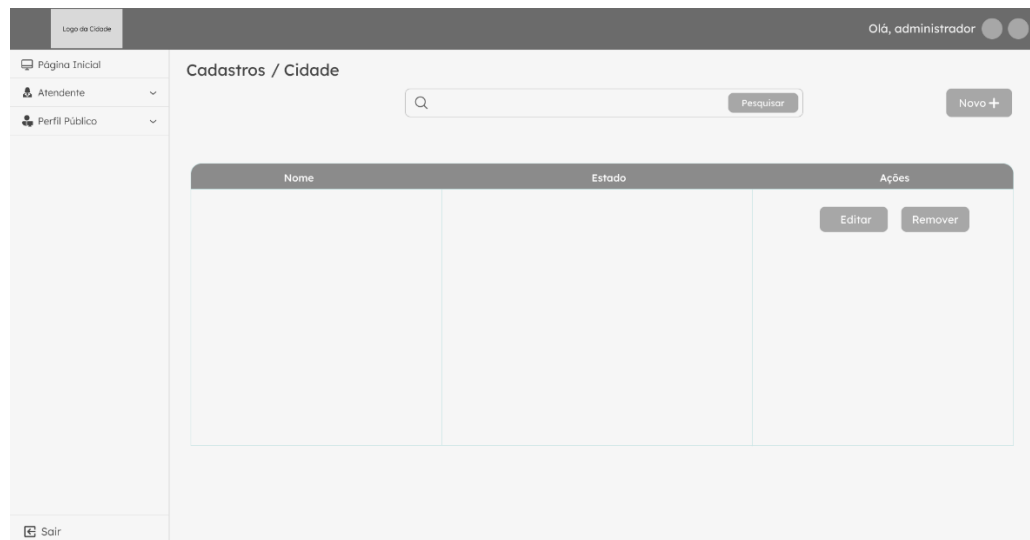
O botão "Novo" representa uma adição intuitiva à tela, possibilitando a inserção direta de novos estados de maneira descomplicada. Essa funcionalidade agiliza o processo de cadastro, oferecendo aos usuários uma maneira rápida de expandir e atualizar a base de dados.

A tabela exibindo todos os estados cadastrados proporciona uma visão consolidada, permitindo uma análise eficiente. Cada entrada na tabela é equipada com opções de edição e exclusão, proporcionando controle total sobre os dados. Essas ações são estrategicamente integradas, garantindo que os usuários possam ajustar as informações conforme necessário.

A opção de voltar, localizada no canto superior esquerdo, é uma adição crucial para a usabilidade. Essa funcionalidade permite uma transição suave entre diferentes áreas do sistema, mantendo a consistência na experiência do usuário.

Na Figura 21 é apresentado o *wireframe* da tela de gerenciamento de cidades assume um papel essencial na organização eficiente do sistema. Com componentes cuidadosamente projetados, essa interface visa simplificar a administração das informações, proporcionando uma experiência fluida e produtiva.

Figura 19 – Wireframe Tela de Cadastro de Cidade



Fonte: Elaborado pelos autores.

A barra de pesquisa oferece uma maneira rápida e eficaz de localizar informações específicas sobre cidades cadastradas, otimizando a navegação no sistema.

O botão "Novo" facilita a inclusão direta de novas cidades, simplificando o processo de cadastro e garantindo uma experiência eficiente para usuários que desejam expandir a base de dados.

A tabela que lista todas as cidades cadastradas proporciona uma visão clara e organizada do conteúdo. Equipada com opções de edição e exclusão, essa tabela oferece controle total sobre as informações, permitindo aos usuários ajustarem e gerenciar os dados com facilidade.

Assim como no *wireframe* da tela de cadastro de estado, a opção de voltar se encontra localizada no canto superior esquerdo, assegura uma transição suave entre diferentes áreas do sistema, promovendo consistência na experiência do usuário e facilitando o retorno aos cadastros anteriores de maneira intuitiva.

4.4 PROTÓTIPOS DE TELA

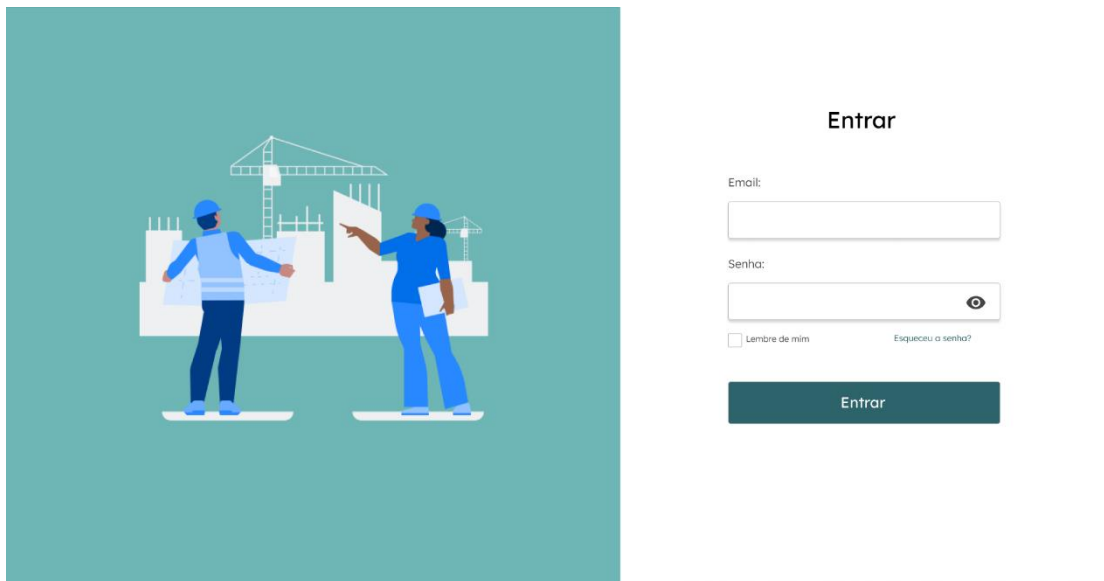
Os protótipos de tela são representações visuais interativas de interfaces do usuário (UI) usadas no Design de Experiência do Usuário (UX). Essas representações simulam a aparência e o comportamento de um produto, aplicativo ou site antes da implementação real. Segundo Teixeira (2014), os protótipos, geralmente compostos por *wireframes* clicáveis ou layouts, são uma maneira eficiente de validar e testar a navegação e as funcionalidades de um

sistema antes de seu desenvolvimento completo. Eles servem como ferramentas cruciais para designers e desenvolvedores testarem a usabilidade, a navegação e a interação, além de obterem *feedback* valioso.

A utilização de protótipos de tela no desenvolvimento do sistema de gerenciamento de documentos de obra para a prefeitura de Jales traz diversos benefícios. Esses modelos visuais interativos possibilitam validar requisitos de apresentação, *layout* e usabilidade. Ao simular a interação do usuário com o sistema, os protótipos facilitam a identificação de potenciais problemas na navegação e permitem obter *feedback* antecipado de partes interessadas, incluindo membros da prefeitura e futuros usuários. Nielsen (1993) destaca que a prototipagem é uma ferramenta essencial para validar essas interações, permitindo detectar problemas de usabilidade e navegar de forma mais eficiente durante o processo de desenvolvimento.

Além disso, a criação de protótipos de tela promove a capacidade de identificar requisitos omitidos nas fases iniciais do projeto, ajuda a evitar retrabalho e garante que o sistema atenda plenamente às necessidades da prefeitura. Garrett (2011) enfatiza que a prototipagem permite envolver os *stakeholders* desde as primeiras etapas, proporcionando um espaço para *feedback* e ajustes antes da implementação final. Dessa forma, é possível economizar recursos ao evitar a implementação de funcionalidades inadequadas e, ao mesmo tempo, facilitar o treinamento dos usuários finais. Os protótipos de tela se tornam, assim, uma ferramenta valiosa para garantir que o sistema seja eficiente, amigável e atenda completamente aos requisitos estabelecidos pela prefeitura de Jales.

O protótipo da tela de login do sistema de gerenciamento de obras da Prefeitura de Jales foi cuidadosamente estilizada para refletir as cores distintivas da identidade visual da prefeitura. A paleta de cores escolhida não apenas contribui para uma estética atraente, mas também reforça a marca e cria uma experiência coesa para os usuários, (Figura 22).

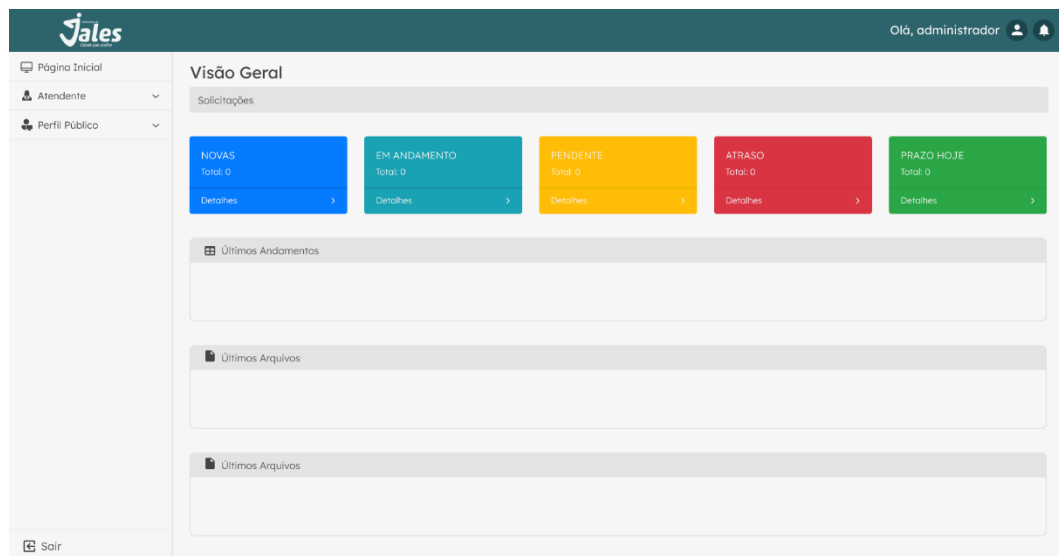
Figura 20 – Protótipo de Tela de Login

Fonte: Elaborado pelos autores.

Os campos de entrada para e-mail e senha, assim como os botões interativos, incorporam as tonalidades específicas da prefeitura de Jales. Isso não apenas garante consistência com a identidade visual, mas também ajuda os usuários a associarem imediatamente a tela de login ao contexto municipal.

Além disso, a imagem representativa na lateral esquerda também é elaborada de modo a complementar as cores predominantes da prefeitura, contribuindo para a harmonia visual da tela. A escolha cuidadosa dessas cores não só enfatiza a identidade da prefeitura, mas também cria uma interface atraente e reconhecível para os usuários ao iniciar a sessão no sistema de gerenciamento de obras.

A tela inicial do sistema de gerenciamento de obras da Prefeitura de Jales foi estrategicamente projetada para oferecer aos usuários uma visão abrangente e acessível das informações mais relevantes. A barra de navegação superior destaca-se pela presença da logo da prefeitura, proporcionando uma identidade visual consistente. Nela, encontram-se elementos essenciais, como o nome do usuário, notificações e acesso ao perfil, garantindo uma experiência personalizada, (Figura 23).

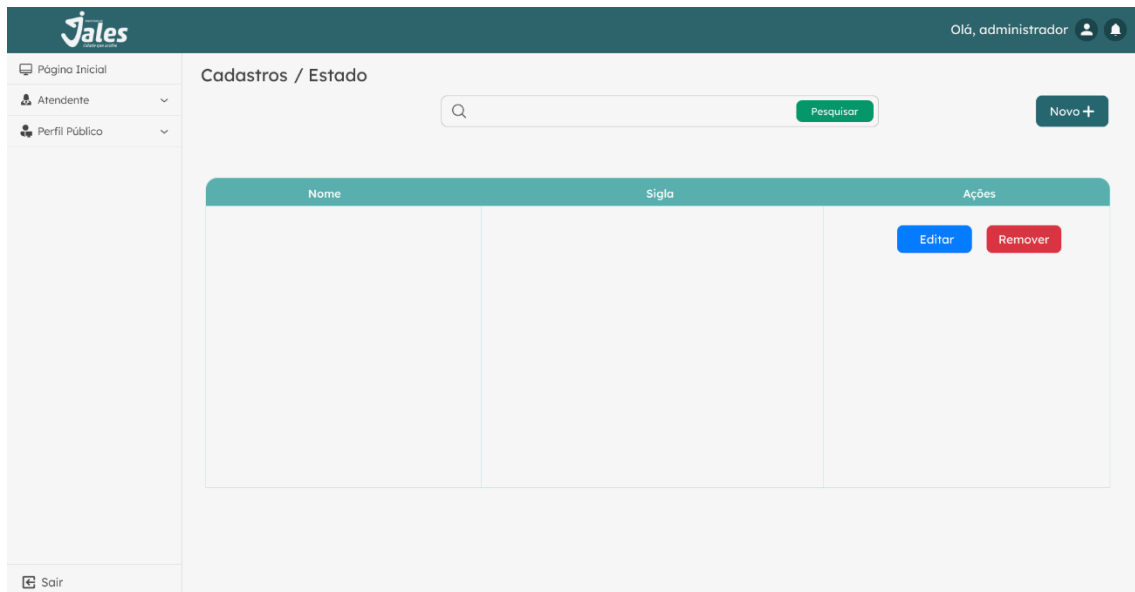
Figura 21 – Protótipo de Tela Inicial

Fonte: Elaborado pelos autores.

Na lateral esquerda, um *sidebar* foi incorporado, apresentando de maneira clara e organizada as diversas opções disponíveis no sistema. Isso não apenas simplifica a navegação, mas também oferece uma visão rápida das funcionalidades acessíveis, contribuindo para a eficiência na utilização.

O centro da tela é ocupado pelo *dashboard*, um espaço dinâmico e informativo que apresenta cartões indicativos de diferentes categorias de documentos. Esses *cards* destacam documentos novos, em andamento, pendentes, em atraso e aqueles com prazo para o dia, proporcionando uma visão consolidada do estado geral dos projetos.

O protótipo da tela de cadastro de estado no sistema de gerenciamento de obras demonstra um design funcional e prático, priorizando a eficiência na gestão dos estados. A tabela no centro da tela apresenta uma lista completa de estados cadastrados, fornecendo uma visão geral rápida e organizada, (Figura 24).

Figura 22 – Protótipo da Tela de Cadastro de Estado

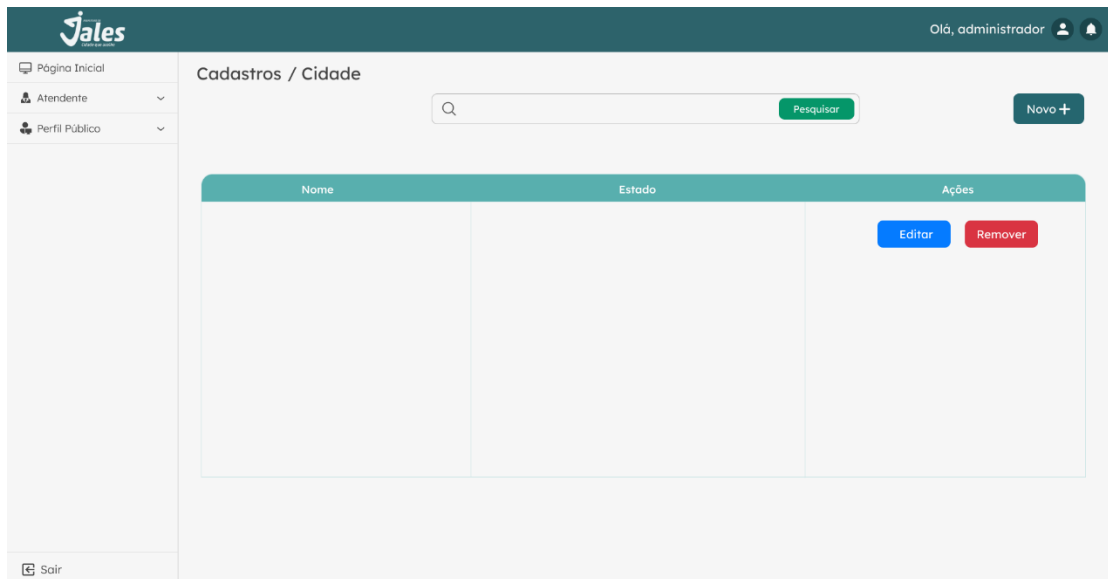
Fonte: Elaborado pelos autores.

Os botões de ação associados a cada entrada da tabela oferecem funcionalidades cruciais, como editar e excluir estados, permitindo aos usuários realizarem operações específicas de forma direta e intuitiva.

A barra de pesquisa localizada na parte superior da tela proporciona uma ferramenta valiosa para a localização rápida de estados específicos na lista. Essa funcionalidade é reforçada pelo botão de adição de novo estado, situado ao lado direito da barra de pesquisa. Esse botão permite a inclusão eficiente de novos estados no sistema, contribuindo para a atualização e expansão contínua da base de dados.

Ao integrar esses elementos, a tela de cadastro de estado oferece uma solução abrangente para a administração e manutenção dos estados no sistema de gerenciamento de obras. A combinação de uma interface intuitiva, funcionalidades de pesquisa e operações diretas melhora significativamente a experiência do usuário ao lidar com informações estaduais.

A tela de cadastro de cidade no sistema de gerenciamento de obras apresenta um design intuitivo e eficiente, priorizando a facilidade de uso para os usuários. A tabela no centro da tela exibe uma lista completa das cidades cadastradas, proporcionando uma visão geral fácil de entender, (Figura 25).

Figura 23 – Protótipo da Tela de Cadastro de Cidade

Fonte: Elaborado pelos autores.

Os botões de ação associados a cada entrada na tabela oferecem funcionalidades essenciais, permitindo aos usuários editarem ou excluïrem cidades de maneira direta e conveniente.

Na parte superior da tela, uma barra de pesquisa oferece uma maneira rápida e eficaz de localizar cidades específicas na lista. Complementando essa funcionalidade, o botão de adicionar nova cidade, situado ao lado direito da barra de pesquisa, simplifica o processo de inclusão de novas informações no sistema.

A integração desses elementos cria uma experiência de usuário coesa e eficaz na administração das informações relacionadas às cidades. A combinação de uma interface clara, recursos de pesquisa e operações diretas contribui para a agilidade e eficiência na gestão de dados relacionados às cidades no contexto do gerenciamento de obras.

5 BANCO DE DADOS

Um banco de dados é uma coleção organizada de dados armazenados eletronicamente em um sistema de computador, geralmente controlado por um Sistema de Gerenciamento de Banco de Dados, a maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL) como padrão para criação, consulta, atualização e exclusão de dados, garantindo flexibilidade e precisão no tratamento das informações Silberschatz (2012).

Segundo o Oracle (2023) “SQL é uma linguagem de programação usada por quase todos os bancos de dados relacionais para consultar, manipular e definir dados e fornecer controle de acesso. O SQL foi desenvolvido pela primeira vez na IBM nos anos 1970, com a Oracle como principal contribuinte, o que levou à implementação do padrão SQL ANSI; o SQL estimulou muitas extensões de empresas como IBM, Oracle e Microsoft. Embora o SQL ainda seja amplamente usado hoje em dia, novas linguagens de programação estão começando a aparecer.”

Silberschatz (2012), os Sistemas de Gerenciamento de Banco de Dados (SGBDs) são essenciais para a organização e gestão dos dados, assegurando a integridade e a segurança desses dados. Ele afirma que os bancos de dados possibilitam a manipulação mais eficaz de grandes quantidades de dados, simplificando a busca e atualização.

5.1 MODELO ENTIDADE RELACIONAMENTO

Silberschatz (2012) afirma, o Modelo Entidade-Relacionamento (MER) é um método muito importante para a modelagem de dados, possibilitando a representação gráfica da estrutura de um banco de dados. O MER apresenta conceitos como entidades, atributos e relações para detalhar as informações e interações existentes no sistema. É crucial para converter as necessidades do mundo real em um modelo lógico que possa fundamentar a implementação do banco de dados. Esta estratégia simplifica a compreensão e a comunicação entre programadores e partes interessadas, além de auxiliar na identificação das principais entidades e suas relações, assegurando que o sistema corresponda às expectativas e demandas dos usuários.

Teorey (2011) reforça essa perspectiva ao enfatizar que o MER é essencial na etapa de projeto conceitual. Ele oferece aos arquitetos uma perspectiva nítida das estruturas de dados que serão postas em prática. O MER não apenas auxilia na identificação de entidades e seus atributos, mas também evidencia a interação entre essas entidades, o que é crucial para a integridade do banco de dados.

O mapeamento objeto-relacional tem como objetivo ligar noções de programação orientada a objetos, como classes e atributos, até componentes de um banco de dados relacional, tabelas e colunas. Este processo promove a interação entre as duas metodologias, garantindo

funcionamento correto do sistema, facilitando a integração entre a camada de aplicação e o banco de dados.

De acordo com a Microsoft (2024), o EF (*Entity Framework*) “é um mapeador relacional de objeto que permite aos desenvolvedores do .NET trabalhar com os dados relacionais usando objetos específicos do domínio. Com ele, não há a necessidade da maioria dos códigos de acesso a dados que os desenvolvedores geralmente precisam para escrever.”

Quadro 40 – Tabela Auditoria

```
CREATE TABLE Auditoria (  
    idauditoria uuid NOT NULL,  
    endpointauditoria character varying(300) NOT NULL,  
    tabelaauditoria character varying(100) NOT NULL,  
    registroauditoria text,  
    acaoauditoria character varying(50) NOT NULL,  
    statusrequisicao character varying(50) NOT NULL,  
    dataauditoria character varying(10) NOT NULL,  
    horaauditoria character varying(12) NOT NULL,  
    descricaoauditoria character varying(500),  
    idsessao integer NOT NULL,  
  
    CONSTRAINT pk_auditoria PRIMARY KEY (idauditoria),  
    CONSTRAINT fk_auditoria_sessao FOREIGN KEY (idsessao) REFERENCES Sessao  
    (idsessao)  
);
```

Fonte: Elaborado pelos autores.

Quadro 41 – Tabela Bairro

```
CREATE TABLE Bairro (  
    idbairro integer NOT NULL,  
    nomebairro character varying(50) NOT NULL,  
    idcidade integer NOT NULL,  
  
    CONSTRAINT pk_bairro PRIMARY KEY (idbairro),  
    CONSTRAINT fk_bairro_cidade FOREIGN KEY (idcidade) REFERENCES Cidade (idcidade)  
);
```

Fonte: Elaborado pelos autores.

Quadro 42 – Tabela Cidade

```
CREATE TABLE Cidade (  
    idcidade integer NOT NULL,
```

```

nomecidade character varying(100) NOT NULL,
idestado integer NOT NULL,

CONSTRAINT pk_cidade PRIMARY KEY (idcidade),
CONSTRAINT fk_cidade_estado FOREIGN KEY (idestado) REFERENCES Estado (idestado)
);

```

Fonte: Elaborado pelos autores.

Quadro 43 – Tabela DocumentoProcesso

```

CREATE TABLE DocumentoProcesso (
    iddocumentoprocesso uuid NOT NULL,
    identificacaodocumento character varying(50) NOT NULL,
    descricao_documento character varying(500),
    observacaodocumento character varying(300),
    dataexpedicao character varying(10),
    dataaprovacao character varying(10),
    arquivo string NOT NULL,
    status integer NOT NULL,
    idprocesso uuid NOT NULL,
    idtipodocumentoetapa integer NOT NULL,
    idresponsavel integer,
    idaprovador integer,

    CONSTRAINT pk_documento PRIMARY KEY (iddocumento),
    CONSTRAINT fk_documento_processo FOREIGN KEY (idprocesso) REFERENCES Processo
(idprocesso),
    CONSTRAINT fk_documento_tipo_documento_etapa FOREIGN KEY (idtipodocumentoetapa)
REFERENCES TipoDocumentoEtapa (idtipodocumentoetapa),
    CONSTRAINT fk_documento_responsavel FOREIGN KEY (idresponsavel) REFERENCES
Usuario (idusuario),
    CONSTRAINT fk_documento_aprovador FOREIGN KEY (idaprovador) REFERENCES Usuario
(idusuario)
);

```

Fonte: Elaborado pelos autores.

Quadro 44 – Tabela Engenheiro

```

CREATE TABLE Engenheiro (
    idengenheiro integer NOT NULL,
    imagempessoa text NOT NULL,
    nomepessoa character varying(70) NOT NULL,
    emailpessoa text NOT NULL,

```

```

telefonepessoa character varying(15) NOT NULL,
cpfcnpjpessoa character varying(18) NOT NULL,
rgiepessoa character varying(15) NOT NULL,
creaengenheiro character varying(8) NOT NULL,

CONSTRAINT pk_engenheiro PRIMARY KEY (idengenheiro)
);

```

Fonte: Elaborado pelos autores.

Quadro 45 – Tabela Estado

```

CREATE TABLE Estado (
    ideestado integer NOT NULL,
    nomeestado character varying(50) NOT NULL,
    ufestado character varying(2) NOT NULL,

    CONSTRAINT pk_estado PRIMARY KEY (ideestado)
);

```

Fonte: Elaborado pelos autores.

Quadro 46 – Tabela Etapa

```

CREATE TABLE Etapa (
    idetapa integer NOT NULL,
    nomeetapa character varying(50) NOT NULL,
    descricaoetapa character varying(500) NOT NULL,
    posicaoetapa integer NOT NULL,
    idtipoprocesso integer NOT NULL,

    CONSTRAINT pk_etapa PRIMARY KEY (idetapa),
    CONSTRAINT fk_etapa_tipoprocesso FOREIGN KEY (idtipoprocesso) REFERENCES
TipoProcesso (idtipoprocesso)
);

```

Fonte: Elaborado pelos autores.

Quadro 47 – Tabela Fiscal

```

CREATE TABLE Fiscal (
    idfiscal integer NOT NULL,
    imagem pessoa text NOT NULL,
    nomepessoa character varying(70) NOT NULL,
    emailpessoa text NOT NULL,
    telefonepessoa character varying(15) NOT NULL,
    cpfcnpjpessoa character varying(18) NOT NULL,

```

```

rgieessoa character varying(15) NOT NULL,

CONSTRAINT pk_fiscal PRIMARY KEY (idfiscal)
);

```

Fonte: Elaborado pelos autores.

Quadro 48 – Tabela Imovel

```

CREATE TABLE Imovel (
    idimovel integer NOT NULL,
    imagemimovel text,
    inscricao cadastral text NOT NULL,
    numeroimovel character varying(6) NOT NULL,
    areaterreno text NOT NULL,
    areacomstruida text NOT NULL,
    condicoessolo text NOT NULL,
    valorvenal text NOT NULL,
    valormercado text NOT NULL,
    localizacaogeografica text,
    idlogradouro integer NOT NULL,
    idproprietario integer NOT NULL,
    idcontribuinte integer NOT NULL,
    idtopografia integer NOT NULL,
    iduso integer NOT NULL,
    idocupacaoatual integer NOT NULL,

    CONSTRAINT pk_imovel PRIMARY KEY (idimovel),
    CONSTRAINT fk_imovel_proprietario FOREIGN KEY (idproprietario) REFERENCES Municipio
(idmunicipio),
    CONSTRAINT fk_imovel_contribuinte FOREIGN KEY (idcontribuinte) REFERENCES Municipio
(idmunicipio),
    CONSTRAINT fk_imovel_logradouro FOREIGN KEY (idlogradouro) REFERENCES Logradouro
(idlogradouro),
    CONSTRAINT fk_imovel_topografia FOREIGN KEY (idtopografia) REFERENCES Topografia
(idtopografia),
    CONSTRAINT fk_imovel_tipouso FOREIGN KEY (idtipouso) REFERENCES TipoUso
(idtipouso),
    CONSTRAINT fk_imovel_ocupacaoatual FOREIGN KEY (idocupacaoatual) REFERENCES
OcupacaoAtual (idocupacaoatual)
);

```

Fonte: Elaborado pelos autores.

Quadro 49 – Tabela Infraestrutura

```

CREATE TABLE Infraestrutura (
    idinfraestrutura integer NOT NULL,
    nomeinfraestrutura character varying(50) NOT NULL,
    idtipoinfraestrutura integer NOT NULL,

    CONSTRAINT pk_infraestrutura PRIMARY KEY (idinfraestrutura),
    CONSTRAINT fk_infraestrutura_tipoinfraestrutura FOREIGN KEY (idtipoinfraestrutura)
    REFERENCES TipoInfraestrutura (idtipoinfraestrutura)
);

```

Fonte: Elaborado pelos autores.

Quadro 50 – Tabela Instalacao

```

CREATE TABLE Instalacao (
    idinstalacao integer NOT NULL,
    datainstalacao character varying(10) NOT NULL,
    situacaoinstalacao text NOT NULL,
    idinfraestrutura integer NOT NULL,
    idimovel integer NOT NULL,
    idengenheiro integer,

    CONSTRAINT pk_instalacao PRIMARY KEY (idinstalacao),
    CONSTRAINT fk_instalacao_imovel FOREIGN KEY (idimovel) REFERENCES Imovel (idimovel),
    CONSTRAINT fk_instalacao_infraestrutura FOREIGN KEY (idinfraestrutura) REFERENCES
    Infraestrutura (idinfraestrutura),
    CONSTRAINT fk_instalacao_engenheiro FOREIGN KEY (idengenheiro) REFERENCES
    Engenheiro (idengenheiro)
);

```

Fonte: Elaborado pelos autores.

Quadro 51 – Tabela Logradouro

```

CREATE TABLE Logradouro (
    idlogradouro integer NOT NULL,
    ceplogradouro character varying(9) NOT NULL,
    nomelogradouro character varying(100) NOT NULL,
    numeroInicial character varying(10) NOT NULL,
    numeroFinal character varying(10) NOT NULL,
    idbairro integer NOT NULL,
    idtipologradouro integer NOT NULL,

    CONSTRAINT pk_logradouro PRIMARY KEY (idlogradouro),

```

```

CONSTRAINT fk_logradouro_tipologradouro FOREIGN KEY (idtipologradouro) REFERENCES
TipoLogradouro (idtipologradouro)
);

```

Fonte: Elaborado pelos autores.

Quadro 52 – Tabela Municipe

```

CREATE TABLE Municipe (
    idmunicipe integer NOT NULL,
    imagempessoa text NOT NULL,
    nomepessoa character varying(70) NOT NULL,
    emailpessoa text NOT NULL,
    telefonepessoa character varying(18) NOT NULL,
    cpfcpnpjpessoa character varying(18) NOT NULL,
    rgiepessoa character varying(15) NOT NULL,

    CONSTRAINT pk_municipe PRIMARY KEY (idmunicipe)
);

```

Fonte: Elaborado pelos autores.

Quadro 53 – Tabela OcupacaoAtual

```

CREATE TABLE OcupacaoAtual (
    idocupacaoatual integer NOT NULL,
    nomeocupacaoatual character varying(50) NOT NULL,
    descricaoocupacaoatual text NOT NULL,

    CONSTRAINT pk_ocupacaoatual PRIMARY KEY (idocupacaoatual)
);

```

Fonte: Elaborado pelos autores.

Quadro 54 – Tabela Processo

```

CREATE TABLE Processo (
    idprocesso uuid NOT NULL,
    identificacaoprocesso character varying(50) NOT NULL,
    descricaooprocesso character varying(500) NOT NULL,
    situacaoprocesso character varying(300) NOT NULL,
    dataaprovacao character varying(10) NOT NULL,
    statusprocesso integer NOT NULL,
    idimovel integer NOT NULL,
    idtipoprocesso integer NOT NULL,
    idengenheiro integer,
    idfiscal integer,

```

```

idresponsavel integer,
idaprovador integer,

CONSTRAINT pk_processo PRIMARY KEY (idprocesso),
CONSTRAINT fk_processo_imovel FOREIGN KEY (idimovel) REFERENCES Imovel (idimovel),
CONSTRAINT fk_processo_responsavel FOREIGN KEY (idresponsavel) REFERENCES Usuario
(idusuario),
CONSTRAINT fk_processo_aprovador FOREIGN KEY (idaprovador) REFERENCES Usuario
(idusuario),
CONSTRAINT fk_processo_fiscal FOREIGN KEY (idfiscal) REFERENCES Fiscal (idfiscal),
CONSTRAINT fk_processo_engenheiro FOREIGN KEY (idengenheiro) REFERENCES
Engenheiro (idengenheiro)
);

```

Fonte: Elaborado pelos autores.

Quadro 55 – Tabela Sessao

```

CREATE TABLE Sessao (
idsessao integer NOT NULL,
datahoraabertura text NOT NULL,
datahorafechamento text,
tokensessao text NOT NULL,
statussessao boolean NOT NULL,
emailpessoa text NOT NULL,
nivelacesso text NOT NULL,
idusuario integer NOT NULL,

CONSTRAINT pk_sessao PRIMARY KEY (idsessao),
CONSTRAINT fk_sessao_usuario FOREIGN KEY (idusuario) REFERENCES Usuario (idusuario)
);

```

Fonte: Elaborado pelos autores.

Quadro 56 – Tabela TipoDocumento

```

CREATE TABLE TipoDocumento (
idtipodocumento integer NOT NULL,
nometipodocumento character varying(50) NOT NULL,
descricao tipodocumento character varying(500) NOT NULL,

CONSTRAINT pk_tipodocumento PRIMARY KEY (idtipodocumento)
);

```

Fonte: Elaborado pelos autores.

Quadro 57 – Tabela TipoDocumentoEtapa

```

CREATE TABLE TipoDocumentoEtapa (
    idtipodocumentoetapa integer NOT NULL,
    posicaotipodocumentoetapa integer NOT NULL,
    idtipodocumento integer NOT NULL,
    idetapa integer NOT NULL,

    CONSTRAINT pk_tipodocumentoetapa PRIMARY KEY (idtipodocumentoetapa),
    CONSTRAINT fk_tipodocumentoetapa_tipodocumento FOREIGN KEY (idtipodocumento)
REFERENCES TipoDocumento (idtipodocumento),
ONSTRAINT fk_tipodocumentoetapa_etapa FOREIGN KEY (idetapa) REFERENCES Etapa
(idetapa)
);

```

Fonte: Elaborado pelos autores.

Quadro 58 – Tabela TipoInfraestrutura

```

CREATE TABLE TipoInfraestrutura (
    idtipoinfraestrutura integer NOT NULL,
    nometipoinfraestrutura character varying(50) NOT NULL,
    descricaotipoinfraestrutura character varying(500) NOT NULL,

    CONSTRAINT pk_tipoinfraestrutura PRIMARY KEY (idtipoinfraestrutura)
);

```

Fonte: Elaborado pelos autores.

Quadro 59 – Tabela TipoLogradouro

```

CREATE TABLE TipoLogradouro (
    idtipologradouro integer NOT NULL,
    codigoinformativo character varying(3) NOT NULL,
    descricaotipologradouro character varying(35) NOT NULL,

    CONSTRAINT pk_tipologradouro PRIMARY KEY (idtipologradouro)
);

```

Fonte: Elaborado pelos autores.

Quadro 60 – Tabela TipoProcesso

```

CREATE TABLE TipoProcesso (
    idtipoprocesso integer NOT NULL,
    tipoprocesso character varying(50) NOT NULL,
    descricaotipoprocesso character varying(500) NOT NULL,

```

```
CONSTRAINT pk_tipoprocesso PRIMARY KEY (idtipoprocesso)
);
```

Fonte: Elaborado pelos autores.

Quadro 61 – Tabela TipoUso

```
CREATE TABLE TipoUso (
    iduso integer NOT NULL,
    nomeuso character varying(50) NOT NULL,
    descricaouso text NOT NULL,

    CONSTRAINT pk_tipouso PRIMARY KEY (idtipouso)
);
```

Fonte: Elaborado pelos autores.

Quadro 62 – Tabela TipoUsuario

```
CREATE TABLE TipoUsuario (
    idtipousuario integer NOT NULL,
    nivelacesso character varying(1) NOT NULL,
    nometipousuario character varying(20) NOT NULL,
    descricaootipousuario character varying(300) NOT NULL,

    CONSTRAINT pk_tipousuario PRIMARY KEY (idtipousuario)
);
```

Fonte: Elaborado pelos autores.

Quadro 63 – Tabela Topografia

```
CREATE TABLE Topografia (
    idtopografia integer NOT NULL,
    nometopografia character varying(50) NOT NULL,

    CONSTRAINT pk_topografia PRIMARY KEY (idtopografia)
);
```

Fonte: Elaborado pelos autores.

Quadro 64 – Tabela Usuario

```
CREATE TABLE Usuario (
    idusuario integer NOT NULL,
    imagempessoa text NOT NULL,
    nomepessoa character varying(70) NOT NULL,
    emailpessoa text NOT NULL,
```

```

telefonepessoa character varying(15) NOT NULL,
cpfcnpjpessoa character varying(18) NOT NULL,
rgiepessoa character varying(15) NOT NULL,
senhausuario character varying(50) NOT NULL,
cargousuario character varying(50) NOT NULL,
statususuario boolean NOT NULL,
idtipousuario integer NOT NULL,

CONSTRAINT pk_usuario PRIMARY KEY (idusuario)
CONSTRAINT fk_usuario_tipousuario FOREIGN KEY (idtipousuario) REFERENCES
Tipousuario (idtipousuario)
);

```

Fonte: Elaborado pelos autores.

5.3 MAPEAMENTO OBJETO RELACIONAL – ORM

O Mapeamento Objeto-Relacional (ORM) é um método que simplifica a comunicação entre aplicações baseadas em objetos e bases de dados relacionais. Ele possibilita o mapeamento automático de classes e objetos no código para tabelas e registros no banco de dados, eliminando a necessidade de redigir instruções SQL manuais. Isso facilita o desenvolvimento, mantendo a manipulação de dados dentro da lógica orientada a objetos, o que facilita a manutenção e torna o código mais compreensível (Macoratti, 2019).

No ambiente do C#, o Entity Framework é uma das ferramentas fundamentais para ORM. Ele serve como um elo entre o código C# e o banco de dados, administrando a geração, leitura, atualização e eliminação de dados sem que o programador tenha que lidar diretamente com consultas SQL. Com o Entity Framework, o mapeamento entre classes e tabelas é realizado automaticamente através de convenções ou configurações personalizadas, possibilitando ao programador manipular os dados através de classes C# e utilizar funcionalidades como validação, relação entre entidades e gerenciamento de transações, reduzindo a complexidade da persistência de dados (Macoratti, 2019).

6 ARQUITETURA DE SOFTWARE

A arquitetura de software refere-se à estrutura fundamental de um sistema de software, incluindo a organização de seus componentes ou módulos e as relações entre esses elementos. Essa estrutura fornece uma visão de alto nível do sistema, guiando o design e a implementação para atender aos requisitos funcionais e não funcionais do software. Além disso, a arquitetura de software visa reduzir o esforço humano necessário para construir e manter um sistema (Martin, 2019).

Segundo Gonçalves (2021), existem diversos tipos de padrões arquiteturais que funcionam como soluções abrangentes e reutilizáveis de componentes de aplicação, criadas para resolver problemas comuns em contextos específicos. Assim, os padrões representam uma abordagem consistente e reaplicável para desafios recorrentes no desenvolvimento de software.

6.1 ARQUITETURA DE DESENVOLVIMENTO

O desenvolvimento da aplicação foi realizado prioritariamente com o uso de softwares gratuitos. A aplicação, que oferece serviços via API RESTful, segue uma arquitetura de web services baseada nos princípios de *Representational State Transfer* – REST (AWS, 2024). Essa arquitetura emprega o protocolo HTTP para executar operações CRUD (*Create, Read, Update, Delete*) sobre os recursos, representados por identificadores uniformes de recursos (URIs). Cada recurso é tratado como uma entidade única e pode ser manipulado por meio dos métodos HTTP padrão, como GET, POST, PUT e DELETE (Sommerville, 2018).

Conforme demonstrado por Sommerville (2018), essa arquitetura de design de sistemas utiliza o formato JSON (JavaScript *Object Notation*) para a troca de dados entre cliente e servidor. JSON é um padrão leve e de fácil leitura, ideal para a serialização eficiente dos dados, permitindo a comunicação entre diferentes sistemas e linguagens de programação de maneira independente. Esse formato favorece a interoperabilidade e simplifica o intercâmbio de informações entre plataformas distintas.

A arquitetura RESTful adotada também facilita a manutenção e evolução da aplicação, pois cada recurso é projetado para ser desacoplado dos demais, o que permite que atualizações ou modificações em um serviço específico sejam implementadas sem impactar diretamente outros componentes do sistema. Esse desacoplamento estrutural aumenta a modularidade da aplicação e permite que novos recursos sejam integrados facilmente. Além disso, o uso de APIs RESTful, junto com o formato JSON, contribui para a escalabilidade horizontal, possibilitando

que a aplicação distribua a carga de trabalho entre múltiplos servidores, atendendo a um grande número de requisições de forma eficiente e confiável (Aws, 2024).

6.1.1 BACK-END

A aplicação servidora *backend*, é responsável por gerenciar as funcionalidades principais da aplicação. Quando o usuário interage com o sistema por meio da interface, o servidor processa a solicitação e retorna uma resposta à aplicação, utilizando JSON. Esse componente foi desenvolvido em C# (Microsoft, 2024), enquanto o gerenciamento de dados é realizado com PostgreSQL, um banco de dados relacional de código aberto.

Durante o desenvolvimento do projeto, identificou-se a necessidade de uma arquitetura em camadas na aplicação servidora *backend* para organizar e gerenciar as funcionalidades de forma mais eficiente. A estrutura implementada conta com várias camadas, cada uma com responsabilidades específicas.

A camada DTO (*Data Transfer Object*), conforme Lima (2023), é um padrão de projeto utilizado para transferir dados entre diferentes camadas da aplicação, como o *backend* e o *frontend*, e realiza a validação inicial das informações. A camada *Controller*, segundo Silvestre (2022), é responsável por receber as solicitações enviadas pela interface do usuário e encaminhá-las para as próximas etapas, acionando métodos de outras camadas conforme necessário.

A camada *Service* gerencia a lógica de negócios, incluindo a validação e controle de acesso aos dados, enquanto se comunica diretamente com a camada *Model*, que define as abstrações das classes do projeto. Por fim, a camada *Repository*, descrita por Barbosa (2021), lida com o acesso aos dados e valida as informações necessárias para a camada *Service*, estabelecendo uma comunicação direta e eficiente para atender às necessidades de processamento e consulta de dados.

6.1.2 FRONT-END - WEB

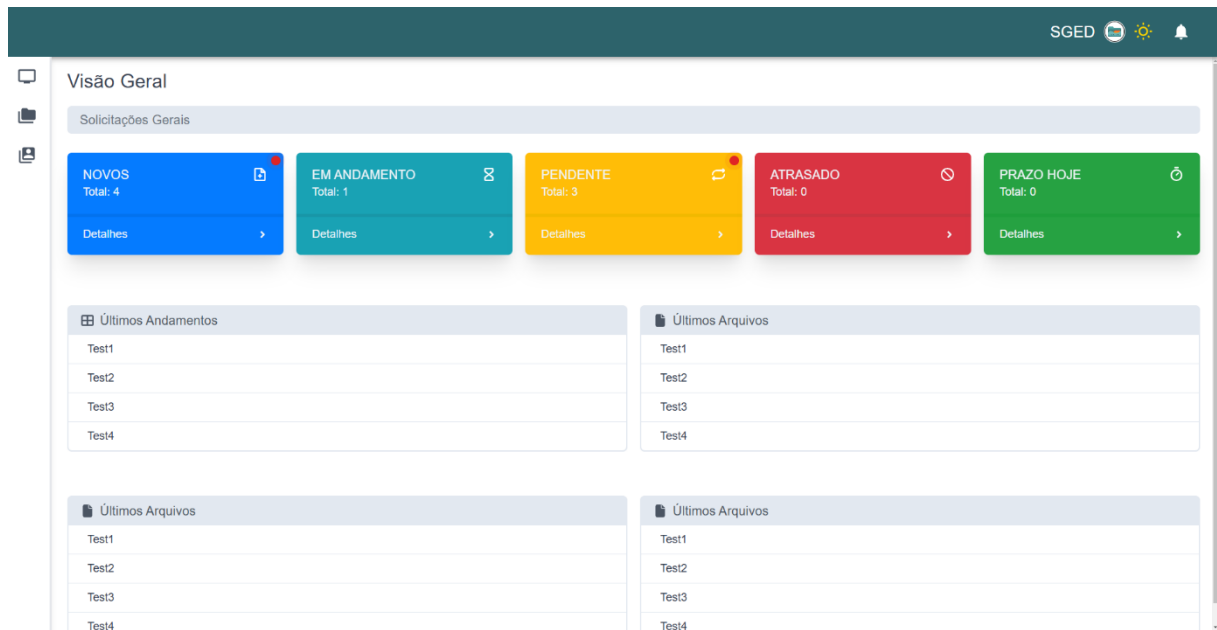
A arquitetura do desenvolvimento do *frontend* desta aplicação foi organizada em uma estrutura modular e escalável, utilizando React com JavaScript. O uso dessa biblioteca permite uma abordagem de desenvolvimento baseadas em componente, facilitando a criação de interfaces do usuário (UI) de forma aninhada e reutilizável (React, 2024). Os protótipos foram elaborados no Figma, uma ferramenta de design digital colaborativa que permite a criação e interação de interfaces em tempo real e facilita a prototipagem interativa, proporcionando a validação de ideias e a obtenção de feedback instantâneo (Figma, 2024).

O projeto foi desenvolvido com o diretório (*src*) como raiz, agrupando arquivos e pastas conforme suas responsabilidades, visando proporcionar clareza ao código, mantendo uma hierarquia lógica e segmentada para atender às diferentes necessidades da aplicação. No núcleo da estrutura do software, existe o diretório (*assets*) armazena os recursos estáticos, como imagens e arquivos de estilo, que são utilizados por diferentes componentes. O diretório (*components*) abriga os elementos reutilizáveis da interface, como botões, formulários e ícones, permite uma abordagem centralizada para criar e gerenciar componentes visuais.

O diretório (*pages*) concentra as páginas completas da aplicação que fornece funcionalidades aos usuários. O diretório (*routes*) é responsável pela definição e gerenciamento das rotas, permitindo a navegação entre diferentes páginas, utilizando a biblioteca React Router. O uso dessa biblioteca permite realizar associação de segmentos de URL a componentes, carregamento de dados e a execução de mutações nos dados (React Router, 2024).

Um dos principais diretórios de organização lógica do projeto é o (*object*), que estrutura o código em subdiretórios para implementar funcionalidades específicas, como a comunicação com o *backend* por meio do axios, sendo um cliente HTTP baseado em promessas, desenvolvido tanto para o ambiente Node.js quanto para navegadores (Axios, 2024).

A componentização é um dos princípios fundamentais desta arquitetura. Na Figura 27, apresenta a visão do usuário da interface, dividida em três componentes principais. O componente *Navbar*, localizado na parte superior, permite que o usuário acesse informações do perfil, visualize notificações e altere a cor do sistema. O *Sidebar* fornece a navegação pelos módulos que compõem o sistema. No centro da tela, encontra-se o componente gerado pelas rotas da aplicação, responsável por exibir as informações específicas de cada página.

Figura 25 – Tela Inicial do sistema

Fonte: Elaborado pelos autores.

6.2 SEGURANÇA DA INFORMAÇÃO

A segurança da informação é essencial para proteger os dados contra acessos não autorizados, vazamentos, alterações ou perdas. Ela é baseada em três princípios fundamentais conhecidos como a tríade da segurança: confidencialidade, integridade e disponibilidade (CIA). A confidencialidade garante que apenas indivíduos autorizados possam acessar informações sensíveis, enquanto a integridade assegura que os dados não sejam modificados indevidamente. Conforme Anderson (2021), esses princípios são a base de qualquer estratégia eficaz de segurança da informação em sistemas computacionais.

Um dos métodos amplamente utilizados para garantir a segurança no gerenciamento de autenticação e autorização de usuários é o JWT (*JSON Web Token*). O JWT é um padrão aberto que permite a troca segura de informações entre o cliente e o servidor de maneira compacta e autossuficiente. Ele utiliza criptografia para garantir a integridade e autenticidade dos dados. De acordo com Bradley (2020), o uso de *tokens* como o JWT em sistemas distribuídos é particularmente eficaz, pois elimina a necessidade de armazenar sessões no servidor e permite a autenticação sem a necessidade de reapplicar credenciais a cada requisição. Isso reduz o risco de roubo de senhas e facilita a escalabilidade de sistemas.

A integridade dos dados também pode ser garantida através do uso de funções de hash, como o SHA-256, que são fundamentais para verificar se as informações foram alteradas de forma não autorizada. O SHA-256, um algoritmo de hash seguro que gera um resumo único de

256 bits, é amplamente utilizado para proteger dados em trânsito e garantir que o conteúdo de tokens ou mensagens não tenha sido modificado. Ele é comumente aplicado em combinação com o JWT, para garantir que o token não tenha sido alterado durante sua transmissão. De acordo com a IBM (2021), o algoritmo SHA-256 é frequentemente utilizado em sistemas de autenticação para garantir a integridade e a autenticidade das mensagens, assegurando que os dados não sejam alterados durante a transmissão.

A auditoria das operações do software desempenha um papel crucial na segurança da informação, pois permite o rastreamento de todas as ações realizadas no sistema, garantindo que qualquer atividade maliciosa ou anômala possa ser detectada. Como Schneier (2015) ressalta, as auditorias devem registrar detalhadamente as ações dos usuários, como acessos a dados, modificações e tentativas de invasão. Esses registros, ou logs de auditoria, são essenciais não apenas para detectar falhas de segurança, mas também para atender a requisitos legais de conformidade, como os exigidos por regulamentos como o GDPR. A análise desses logs pode fornecer uma visão precisa do que ocorreu em um sistema, facilitando a identificação de vulnerabilidades e melhorando a resposta a incidentes de segurança.

No desenvolvimento de sistemas modernos, a segurança na comunicação entre o *frontend* e o *backend* é fundamental para proteger os dados em trânsito. O uso de HTTPS, em conjunto com protocolos de criptografia como o TLS (*Transport Layer Security*), é imprescindível para garantir que informações sensíveis, como credenciais de usuários e *tokens* de autenticação, não sejam interceptadas durante a troca de dados entre o cliente e o servidor. A utilização de JWT nesse contexto também é uma prática comum, pois ele circula entre o cliente e o servidor por meio de pacotes HTTP, sendo transportado de forma segura. Williams (2018) afirma que o JWT, quando transmitido sobre uma conexão segura (HTTPS), garante que o *token* não será interceptado por agentes maliciosos, mantendo a confidencialidade da comunicação.

Em síntese, a segurança da informação é uma preocupação constante no desenvolvimento de sistemas, que envolve a adoção de tecnologias e práticas que garantam a proteção dos dados. O uso de JWT para autenticação e autorização, aliado a auditorias eficazes e à proteção das comunicações entre *frontend* e *backend*, é essencial para criar sistemas seguros e confiáveis. As melhores práticas de segurança devem ser aplicadas de forma contínua e abrangente, garantindo a proteção dos dados e a conformidade com os requisitos legais e normativos.

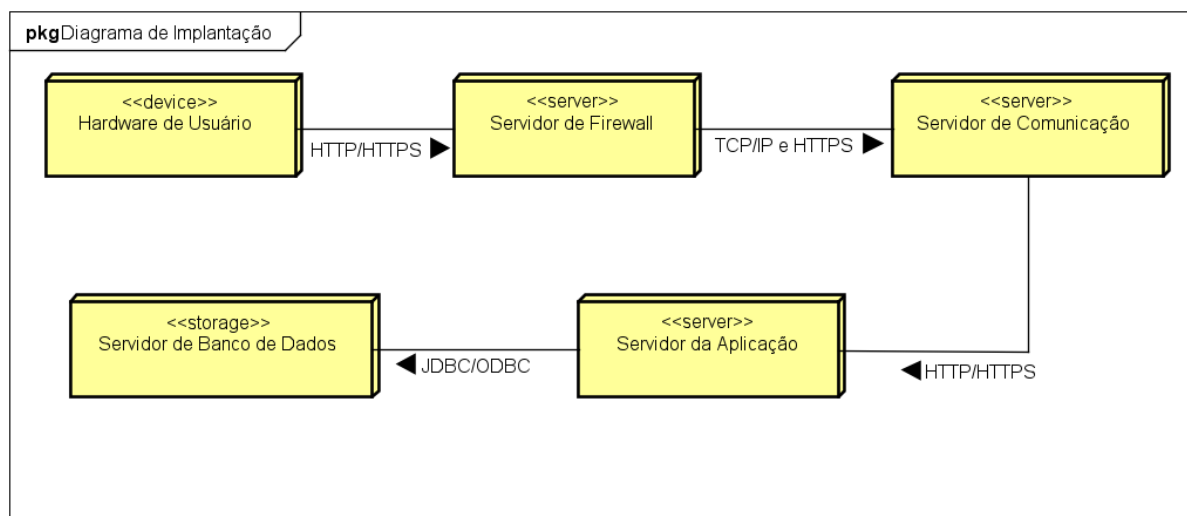
6.3 IMPLANTAÇÃO

A implantação de sistemas em uma organização é um processo essencial, porém, extremamente complexo, que envolve a integração de novas soluções tecnológicas aos processos já existentes. Segundo Prada (2021), uma implantação mal planejada pode resultar em impactos negativos significativos, como desorganização, perda de produtividade e até mesmo falhas operacionais. Diante disso, o planejamento detalhado e o acompanhamento rigoroso de cada etapa do processo são fundamentais para garantir o sucesso da implementação. A atenção aos detalhes é imprescindível, pois qualquer erro na fase de implantação pode comprometer a eficácia do sistema, prejudicando o desempenho organizacional e gerando caos nos fluxos de trabalho. Portanto, é necessário um cuidado especial para que todas as fases da implantação sejam cuidadosamente executadas, desde o planejamento até o suporte pós-implantação.

O diagrama de implantação na UML descreve a arquitetura física do sistema, representando como os componentes de software são distribuídos em diferentes máquinas, como servidores ou computadores pessoais. Sendo possível visualizar a conexão entre as máquinas e os protocolos usados para a comunicação entre elas (Guedes, 2011).

Conforme apresentado na Figura 28, o diagrama de implantação apresenta a estrutura de comunicação entre os principais componentes do sistema. Ele detalha a interação entre o hardware de usuário e os diversos servidores, como o servidor de firewall, servidor de comunicação, servidor de aplicação e servidor de banco de dados.

Figura 26 – Diagrama de Implantação



Fonte: Elaborado pelos autores.

O Hardware de Usuário representa os dispositivos utilizados pelos usuários finais, como computadores, tablets ou smartphones. Esses dispositivos são responsáveis por iniciar a comunicação com o sistema, utilizando protocolos HTTP ou HTTPS. Através deles, os usuários podem enviar e receber dados de maneira segura, iniciando a conexão com o sistema.

O Servidor de Firewall atua como uma camada de proteção entre o hardware de usuário e os demais servidores da aplicação. Ele filtra e monitora o tráfego de dados entre os dispositivos dos usuários e os componentes internos do sistema, permitindo apenas conexões seguras e autorizadas. No diagrama, o *firewall* se conecta ao hardware de usuário via HTTP/HTTPS e ao servidor de comunicação usando os protocolos TCP/IP e HTTPS, adicionando uma camada de segurança ao fluxo de dados.

O Servidor de Comunicação é responsável pela troca de informações entre os diferentes servidores do sistema. Ele facilita a comunicação entre os diversos componentes, assegurando que os dados sejam transmitidos de forma eficiente e segura. Ele se conecta tanto ao servidor de firewall quanto ao servidor de aplicação, utilizando o protocolo HTTPS para garantir a segurança na transferência de dados.

O Servidor de Aplicação é o núcleo onde reside a lógica do sistema. Ele processa as solicitações dos usuários, executa as regras de negócio e gerencia o fluxo de informações entre o *frontend* e o banco de dados. No diagrama, ele se comunica com o servidor de comunicação via HTTP/HTTPS e com o servidor de banco de dados através do protocolo JDBC/ODBC, possibilitando o acesso seguro aos dados necessários para a aplicação.

Por fim, o Servidor de Banco de Dados é onde são armazenados os dados essenciais para o funcionamento do sistema, como informações de usuários e transações. Esse servidor se conecta ao servidor de aplicação via JDBC/ODBC, fornecendo um meio seguro e estruturado para o armazenamento e recuperação de dados. A presença deste servidor é crucial para assegurar que as informações estejam disponíveis e sejam facilmente acessíveis pelo sistema.

A implantação da aplicação foi realizada utilizando Docker, uma plataforma de código aberto que permite o desenvolvimento, envio e execução de aplicativos em contêineres. Facilita a separação entre aplicativos e infraestrutura, proporcionando maior agilidade na entrega de software, sendo possível gerenciar a infraestrutura de maneira similar aos aplicativos (Docker, 2024).

Para a implantação do *backend*, foi utilizado um Dockerfile que define todas as etapas necessárias para criar um contêiner Docker para a aplicação web API em C#. O processo começa com a criação de uma imagem base do SDK do .NET, onde o código-fonte é copiado, as dependências são restauradas e a aplicação é compilada. Em seguida, uma segunda imagem

base do *runtime* do .NET é utilizada para executar a aplicação. Esse Dockerfile configura o diretório de trabalho, copia os arquivos compilados e expõe a porta da aplicação, permitindo que o contêiner seja acessado externamente. Além disso, na configuração do ambiente, a conexão com o banco de dados é definida através de uma variável de ambiente, facilitando a integração com diferentes ambientes de execução.

Para o *frontend*, que é desenvolvido em JavaScript, um Dockerfile também foi configurado para criar o contêiner adequado. A primeira etapa envolve a utilização de uma imagem base do Node.js para instalar as dependências e compilar a aplicação para o ambiente de produção. Posteriormente, uma imagem do Nginx é utilizada para servir a aplicação compilada. O Dockerfile configura a cópia dos arquivos necessários e expõe a porta padrão do Nginx para permitir o acesso ao *frontend*. Além disso, um arquivo de configuração do Nginx foi incluído para direcionar corretamente as requisições e facilitar a comunicação entre o *frontend* e o *backend*. Variáveis de ambiente foram configuradas para adaptar o sistema ao ambiente de execução, definindo a URL da API no *frontend*.

O banco de dados também foi incluído no ambiente Docker, proporcionando uma solução isolada e consistente para armazenamento de dados. Para facilitar o backup e a restauração do banco de dados, foi criado um procedimento onde a estrutura e os dados do banco são exportados para um arquivo .sql por meio do comando `pg_dump`. Esse arquivo é armazenado em uma pasta específica no projeto, permitindo que o banco de dados possa ser facilmente restaurado em outros ambientes, caso necessário.

Na implantação do sistema com Docker Compose, foi organizada uma estrutura de contêineres para o banco de dados, *backend* e *frontend*, permitindo que todos os serviços interajam de maneira isolada e consistente. Primeiramente, foi criada uma pasta chamada Docker onde o arquivo `docker-compose.yml` foi configurado. Esse arquivo descreve cada serviço e suas dependências, facilitando a orquestração dos contêineres.

7 CONCLUSÃO

Em conclusão, o desenvolvimento do SGED (Sistema de Gestão de Documentos) tem apresentado resultados promissores, confirmando que sua arquitetura e lógica atendem amplamente aos requisitos definidos pela Secretaria de Obras. A estrutura de níveis de acesso e controle de ações assegura que cada usuário realize atividades dentro de seu escopo autorizado, com validação e deliberação tanto no *frontend* quanto no *backend* do sistema, garantindo segurança contra acessos não autorizados. Essa organização hierárquica facilita a delegação de tarefas, preservando a confidencialidade de informações restritas a determinados níveis, como o de estagiário.

Embora nem todas as funcionalidades previstas tenham sido implementadas para a fase de implantação, as principais já estão desenvolvidas compõem o núcleo funcional do sistema, onde todos os dados externos são centralizados. Esse núcleo permite a gestão segura de processos e documentos, além do acompanhamento do progresso e do status atual, validando a integridade dos arquivos por meio de um hash SHA-256, que acompanha cada transação com a API.

Adicionalmente, o SGED foi projetado para atender às necessidades de longo prazo do cliente, armazenando dados de forma segura e acessível para auditorias ou verificações. Isso garante que o sistema possa ser utilizado de maneira confiável, preservando a integridade e a disponibilidade das informações para consultas por autoridades, se necessário.

Atualmente, o sistema encontra-se em fase de homologação junto à Secretaria de Obras, passando por uma etapa de testes com o cliente para verificar a correta aderência do processo automatizado às necessidades reais dos usuários. Para versões futuras, prevê-se a implementação de um módulo de análise de dados, que permitirá ao usuário obter uma visão abrangente da gestão da secretaria no que diz respeito à aprovação dos processos de obras.

8 REFERÊNCIAS

- AMSTEL, Frederick van. *Personas e cenários para antecipar o futuro*. 2007. Disponível em: <https://www.usabilidoido.com.br/personas_e_cenarios_para_antecipar_o_futuro_.html>. Acesso em: 26 nov. 2023.
- ANDERSON, R. *Security Engineering: A Guide to Building Dependable Distributed Systems*. 2. ed. Wiley, 2021.
- AWARI. *Entenda a importância de wireframes para UI/UX Design*. 2022. Disponível em: <https://awari.com.br/entenda-a-importancia-de-wireframes-para-ui-ux-design/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=Entenda%20a%20importancia%20e%20wireframes%20para%20UI/UX%20Design>. Acesso em: 27 nov. 2023.
- AWS. *O que é uma API RESTful?*. Amazon Web Service, 2024. Disponível em: <<https://aws.amazon.com/pt/what-is/restful-api/>>. Acesso em: 20 out. 2024.
- AXIOS. *Introdução*. 2024. Disponível em: <<https://axios-http.com/ptbr/docs/intro>>. Acesso em: 07 nov. 2024.
- BRADLEY, M. *JWT Handbook*. 1. ed. O'Reilly Media, 2020.
- BARBOSA, Tadeu. *Movendo a lógica de sua aplicação para Services e Repositories*. 2021. Disponível em: <<https://dev.to/tadeubdev/movendo-a-logica-de-sua-aplicacao-para-services-e-repositories-4lee>>. Acesso em: 22 nov. 2024.
- COSTA, L. *Prefeitura orienta sobre a importância de ter uma construção regularizada*. 2020. Disponível em: <<https://imperatriz.ma.gov.br/noticias/planejamento/importancia-de-se-ter-uma-construcao-regularizada.html>>. Acesso em: 13 jun. 2024.
- DOCKER. *Docker overview*. Disponível em: <<https://docs.docker.com/get-started/docker-overview/>>. Acesso em: 11 nov. 2024.
- FIGMA. *Figma Design*. 2024. Disponível em: <<https://www.figma.com/pt-br/design/>>. Acesso em: 05 nov. 2024.
- GARRETT, Jesse James. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. 2. ed. Berkeley: New Riders, 2011.
- GestQual. *GestQual DOCUMENTOS*. 2024. Disponível em: <<https://gestqual.com.br/gestqual-documentos-4/>>. Acesso em: 16 jun. 2024.
- GONÇALVES, Marcelo M. *Arquitetura de Software: Estilos e Padrões de Design*. 2021. Disponível em: <<https://medium.com/@marcelomg21/arquitetura-de-software-estilos-e-padr%C3%B5es-de-design-50d62d684ef2>>. Acesso em: 24 nov. 2023.
- GUEDES, Gilleanes T. A. *UML 2: Uma Abordagem prática*. São Paulo: Novatec Editora, 2011.

IBM. (2021). *Authentication with JSON Web Tokens (JWT)*. Disponível em: <<https://www.ibm.com/docs/pt-br/db2/12.1?topic=authentication-json-web-tokens-jwt>>. Acesso em: 20 dez. 2024.

LIMA, João. *DTO - “Data Transfer Object”*. Dio, 2023. Disponível em: <<https://www.dio.me/articles/dto-data-transfer-object>>. Acesso em: 22 nov. 2024.

LISBOA, Ândlei. *Por que criar Personas?*. Disponível em: <<https://brasil.uxdesign.cc/por-que-criar-personas-bc796a1ffc7e>>. Acesso em: 26 nov. 2023.

LUCIDCHART. *O que é um diagrama de sequência UML?*. Disponível em: <<https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-sequencia-uml>>. Acesso em: 07 jun. 2024.

MACORATTI, J. *Entity Framework Core - Mapeando um Banco de Dados Relacional*. Novatec Editora, 2019.

MAGALHÃES, R. M.; MELLO, L. C. B.; BANDEIRA, R. A. de M. *Planejamento e controle de obras civis: estudo de caso múltiplo em construtoras no Rio de Janeiro*. Gestão & Produção, 25(1), 44-55, 2018. Disponível em: <<http://dx.doi.org/10.1590/0104-530X2079-15>>.

MARTIN, Robert C. *Arquitetura limpa: o guia do artesão para estrutura e design de software*. Tradução de Daniel A. Martins. São Paulo: Alta Books, 2019.

MICROSOFT. *Documentação do Entity Framework*. Disponível em: <<https://learn.microsoft.com/pt-br/aspnet/entity-framework>>. Acesso em: 03 out. 2024.

MICROSOFT. *Criar DTOs (objetos de transferência de dados)*. 17 julho 2023. Disponível em: <<https://learn.microsoft.com/pt-br/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>>. Acesso em: 03 dez. 2023.

MICROSOFT. *O que é o Visual Studio?*. Disponível em: <<https://learn.microsoft.com/pt-br/visualstudio/get-started/visual-studio-ide?view=vs-2022>>. Acesso em: 03 dez. 2023.

MICROSOFT. *Um tour pela linguagem C#*. 15 fevereiro 2023. Disponível em: <<https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>>. Acesso em: 03 dez. 2023.

MICROSOFT. *Visual Studio Code*. 2023. Disponível em: <<https://visualstudio.microsoft.com/pt-br/#vscode-section>>. Acesso em: 03 dez. 2023.

MOBUSS. *A importância da gestão de documentos no canteiro de obras*. 2018. Disponível em: <<https://www.mobussconstrucao.com.br/blog/gestao-de-documentos/#:~:text=A%20gest%C3%A3o%20de%20documentos%20que,empecilhos%20entre%20equipes%20e%20clientes>>. Acesso em: 12 jun. 2024.

NEVES, V. *React: o que é, como funciona e um Guia dessa popular ferramenta JS*. 17 janeiro 2023. Disponível em: <<https://www.alura.com.br/artigos/react-js>>. Acesso em: 03 dez. 2023.

NIELSEN, Jakob. *Usability Engineering*. San Francisco: Morgan Kaufmann, 1993.

ORACLE. *O que é um Banco de Dados?*. 2023. Disponível em: <<https://www.oracle.com/br/database/what-is-database/>>. Acesso em: 21 nov. 2023.

PRADA, Charles. *Como preparar a sua empresa para a implantação de sistema: confira os 3 aspectos principais*. Euax, 2021. Disponível em: <<https://www.euax.com.br/2021/04/implantacao-de-sistema/>>. Acesso em: 11 nov. 2024.

PRESSMAN, Roger S.; MAXIM, Bruce R. *Engenharia de Software: uma abordagem profissional*. 9. ed. Porto Alegre: AMGH, 2021.

REACT ROUTER. *Route*. 2024. Disponível em: <<https://reactrouter.com/en/main/route/route>>. Acesso em: 07 nov. 2024.

REACT. *Descrevendo a IU*. 2024. Disponível em: <<https://react.dev/learn/describing-the-ui>>. Acesso em: 07 nov. 2024.

SCHNEIER, B. *Secrets and Lies: Digital Security in a Networked World*. Wiley, 2015.

SILBERSCHATZ, Abraham. *Sistemas de Banco de Dados*. 6. ed. São Paulo: McGraw-Hill, 2012.

SILVESTRE, G. *Controller e Service - Uma breve introdução*. Dev.to, 2022. Disponível em: <<https://dev.to/gabrielhsilvestre/controller-e-service-uma-breve-introducao-49j9>>. Acesso em: 20 nov. 2024.

STEIN, M. *O que é um DTO (Data Transfer Object)?*. 2024. Disponível em: <<https://www.marcusmuller.com.br/o-que-e-um-dto-data-transfer-object/>>. Acesso em: 03 dez. 2023.

SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. São Paulo: Pearson Education do Brasil, 2018.