Faculdade de Tecnologia Professor José Camargo

EDUARDO LOPES FONSECA GONZALES

GABRIEL MACHADO DOS SANTOS

GUILHERME ANGELO SILVA

JOÃO PEREIRA NETO

SISTEMA GERENCIADOR DE DOCUMENTOS

Projeto Integrador

Disciplinas Envolvidas: Engenharia de Software, Técnicas Avançadas de Banco de Dados Relacional e não Relacional, Técnicas Avançadas de Programação, Técnicas Avançadas de Programação Web e Mobile, Gestão Ágil de Projetos de Software.

Jales

2024

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

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.

> Segundo Costa (2020), "O licenciamento para obras é imprescindível ao construir um imóvel. 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".

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 munícipes. 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 (Sommervile, 2007), “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.

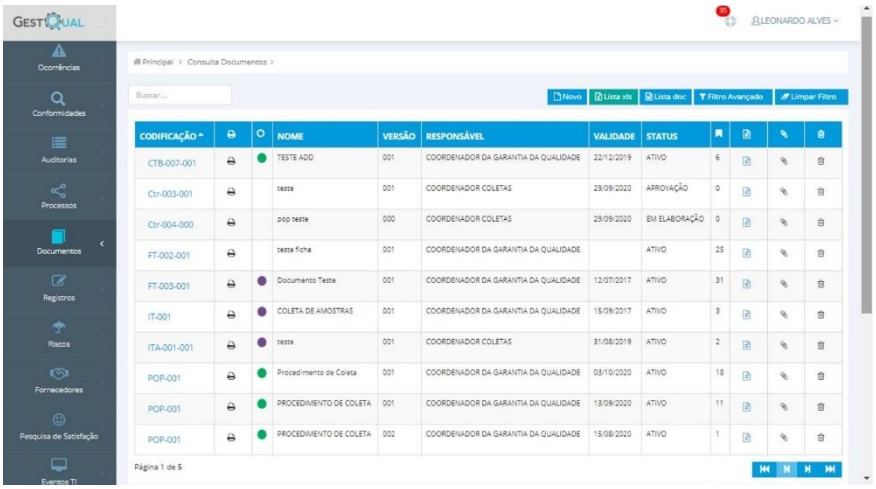
2.3 Análise De Sistemas Existentes

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 GestQual.



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 relacionado ao desenvolvimento do banco de dados. Ele envolve a necessidade de armazenar uma vasta quantidade de dados, bem como informações sensíveis. Portanto, não será permitido a negligência da segurança do banco de dados.

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, 2007, p.80).

Dessa forma ao identificar e documentar de forma clara as funções principais do sistema, os requisitos funcionais fornecem uma base mensurável e testável para garantir que o sistema atenda às expectativas dos usuários. Sua revisão contínua ao longo do ciclo de vida do

projeto é crucial 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 (2009), 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 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 maquinas 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*) é uma linguagem padrão utilizada para modelar e documentar sistemas orientados a objetos. O Diagrama de Casos de Uso é essencial entre os diagramas para descrever as funcionalidades do sistema do ponto de vista do usuário, documentando as principais funções (Guedes, 2011).

3.1 DIAGRAMA DE CLASSES

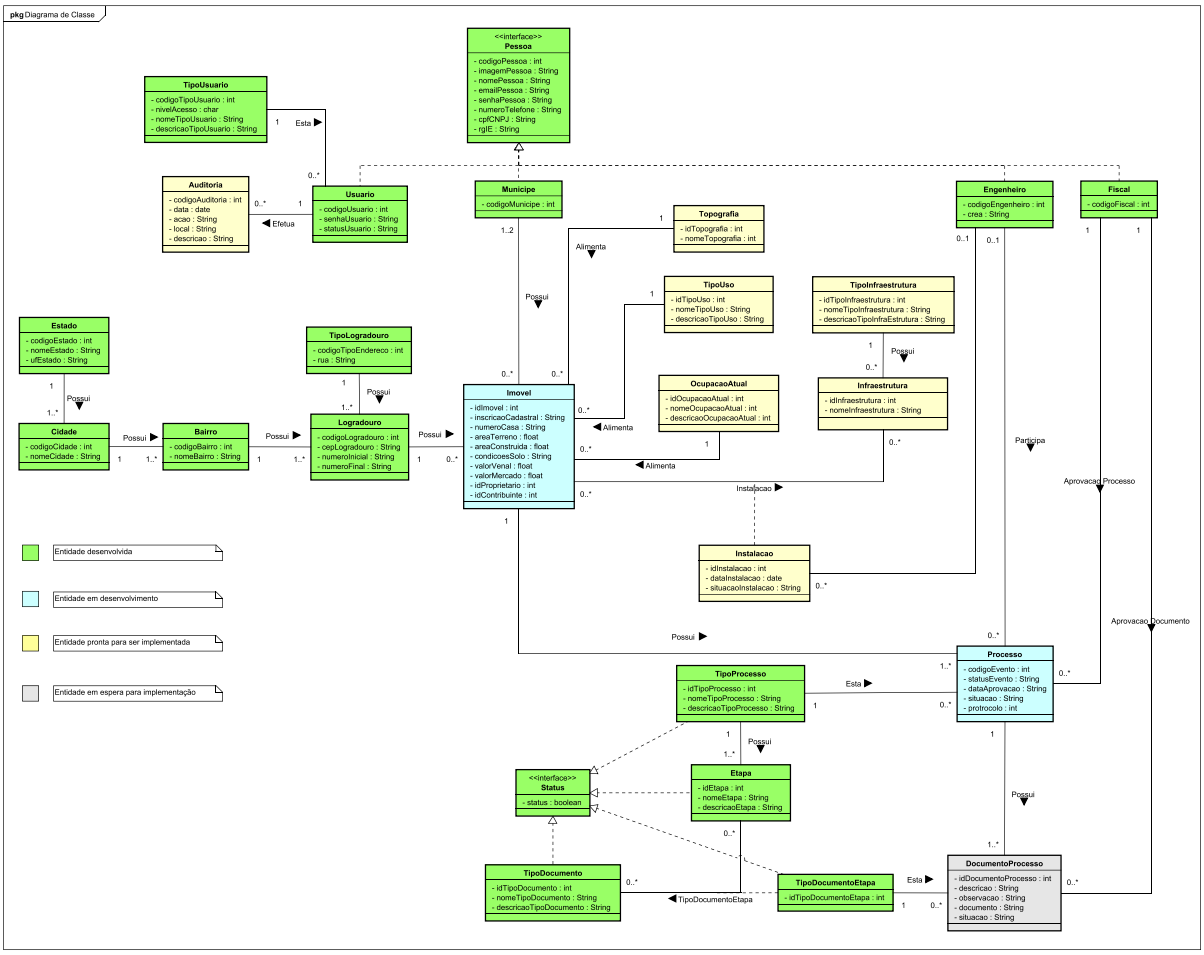
O Diagrama de Classes é uma das ferramentas mais importantes na modelagem orientada a objetos, pois oferece uma visão clara e organizada da estrutura interna de um sistema. Ele permite a identificação das classes do software, bem como suas características, métodos e conexões entre elas, como associações, 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 (2007), 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. Além disso, 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.

Apresentasse na Figura 1 o Diagrama de Classe, abrangendo a configuração da camada de negócios correspondente.

Figura 4 – Diagrama de Classes



Fonte: Elaborado pelos autores.

3.2 DICIONÁRIO DE CLASSES

No Quadro 3 a classe Tipo Usuário que é responsável por guardar todo tipo de dado, sobre o nível de acesso do usuário, dentro do sistema.

Quadro 3 – Descrição da Classe TipoUsuário

Atributo	Tipo	Descrição
codigoTipoUsuario	Integer	Código para identificar o tipo de usuário.
nivelAcesso	Char	Letra para identificar o tipo de nível de acesso.
nomeTipoUsuario	String	Nome de identificação do tipo de usuário.

descricaoTipoUsuario	String	Descrição do local onde as informações acessíveis ao tipo de usuário serão disponibilizadas.

Fonte: Elaborado pelos autores.

No Quadro 4 a classe Auditoria visa registrar todas as operações efetuadas pelo usuário, montando um histórico das ações realizadas.

Quadro 4 - Descrição da Classe Auditoria

Atributo	Tipo	Descrição
codigoAuditoria	Integer	Código para identificação da auditoria.
acaoEfetuada	String	Ação efetuada no sistema.
dataAcao	Date	Data em que a alteração foi realizada.
classeAfetada	String	Local no sistema que foi impactado pela modificação.
codigoRegistro	Integer	Código de registro das informações.

Fonte: Elaborado pelos autores.

No Quadro 5 a classe Usuário destinada a acessar o sistema e executar operações internas.

Quadro 5 – Descrição da Classe Usuário

Atributo	Tipo	Descrição
codigoUsuario	Integer	Código para identificação do usuário.
senhaUsuario	String	Senha de acesso ao sistema.
statusUsuario	Boolean	Indica se o usuário está ativo ou inativo.

Fonte: Elaborado pelos autores.

No Quadro 6 a classe Pessoa tem como objetivo armazenar atributos comuns de outras classes, simplificando a implementação.

Quadro 6 – Descrição da Classe Pessoa

Atributo	Tipo	Descrição

codigoPessoa	Integer	Código para identificação da pessoa.
nomePessoa	String	Nome de identificação da pessoa.
emailPessoa	String	Email de identificação utilizado para acessar o sistema.
senhaPessoa	String	Senha de identificação utilizada para acessar o sistema.
numeroTelefone	String	Número de telefone da pessoa.
cpfCNPJ	String	CPF para pessoas físicas e CPNJ para pessoas jurídicas.
rgIE	String	RG (Registro Geral) para identificação pessoal e IE (Inscrição Estadual) para identificação empresarial.

Fonte: Elaborado pelos autores.

No Quadro 7 a classe Munícipe cujos dados estão registrados nos documentos de obra na prefeitura, permite visualizar documentos, mas sem autorização para efetuar alterações.

Quadro 7 – Descrição da Classe Munícipe

Atributo	Tipo	Descrição
codigoMunicipe	Integer	Código para identificação do munícipe.

Fonte: Elaborado pelos autores.

No Quadro 8 a classe Engenheiro cujos dados estão registrados nos documentos de obra na prefeitura, permite visualizar documentos, mas sem autorização para efetuar alterações.

Quadro 8 – Descrição da Classe Engenheiro

Atributo	Tipo	Descrição
codigoEngenheiro	Integer	Código para identificação do engenheiro.

Fonte: Elaborado pelos autores.

No Quadro 9 a classe Fiscal é responsável por fiscalizar a obra associada ao documento, limitando-se à visualização dos dados pertinentes.

Quadro 9 – Descrição da Classe Fiscal

Atributo	Tipo	Descrição

codigoFiscal	Integer	Código para identificação do fiscal.

Fonte: Elaborado pelos autores.

No Quadro 10 a classe Estado tem objetivo armazenar informações relacionadas aos estados do país.

Quadro 10 – Descrição da Classe Estado

Atributo	Tipo	Descrição
codigoEstado	Integer	Código para identificação do estado.
nomeEstado	String	Nome de identificação do estado.
ufEstado	String	Sigla do estado.

Fonte: Elaborado pelos autores.

No Quadro 11 a classe Cidade tem como objetivo armazenar dados relacionados à cidade dentro do estado.

Quadro 11 – Descrição da Classe Cidade

Atributo	Tipo	Descrição
codigoCidade	Integer	Código para identificação da cidade.
nomeCidade	String	Nome de identificação da cidade.

Fonte: Elaborado pelos autores.

No Quadro 12 a classe Bairro tem como objetivo armazenar informações sobre os bairros de uma determinada cidade.

Quadro 12 – Descrição da Classe Bairro

Atributo	Tipo	Descrição
codigoBairro	Integer	Código para identificação do bairro.
nomeBairro	String	Nome de identificação do bairro.

Fonte: Elaborado pelos autores.

No Quadro 13 a classe Tipo Logradouro tem como objetivo armazenar o tipo de logradouro de um endereço, como por exemplo avenida, alameda, rua, entre outros.

Quadro 13 – Descrição da Classe TipoLogradouro

Atributo	Tipo	Descrição
codigoTipoEndereco	Integer	Código para identificação do tipo de endereço.
rua	String	Rua para Identificação do endereço.

Fonte: Elaborado pelos autores.

No Quadro 14 a classe Logradouro tem como objetivo armazenar informações sobre o terreno ou espaço associado ao documento onde a obra foi aprovada.

Quadro 14 – Descrição da Classe Logradouro

Atributo	Tipo	Descrição
codigoLogradouro	Integer	Código para identificação do logradouro.
cepLogradouro	String	CEP 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 15 a classe Imóvel tem como objetivo armazenar as informações referentes ao imóvel, relacionadas à obra aprovada no logradouro.

Quadro 15 – Descrição da Classe Imóvel

Atributo	Tipo	Descrição
inscricaoImovel	Integer	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 16 a classe Topografia tem como objetivo armazenar as informações referentes à topografia do imóvel.

Quadro 16 – Descrição da Classe Topografia

Atributo	Tipo	Descrição
codigoTopografia	Integer	Código para identificação da topografia.
nomeTopografia	String	Nome de identificação da topografia.

Fonte: Elaborado pelos autores.

No Quadro 17 a classe uso tem como objetivo armazenar as informações referentes aos usos do imóvel.

Quadro 17 – Descrição da Classe Uso

Atributo	Tipo	Descrição
codigoUso	Integer	Código para identificação do uso.
nomeUso	String	Nome de identificação do uso.
descricaoUso	String	Descrição do uso.

Fonte: Elaborado pelos autores.

No Quadro 18 a classe Ocupação Atual tem como objetivo armazenar as informações referentes as diferentes ocupações do imóvel.

Quadro 18 – Descrição da Classe Ocupacao Atual

Atributo	Tipo	Descrição
codigoOcupacaoAtual	Integer	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 19 a classe Tipo Infraestrutura tem como objetivo armazenar as informações referentes aos usos do imóvel.

Quadro 19 – Descrição da Classe Tipo Infraestrutura

Atributo	Tipo	Descrição
codigoTipoInfraestrutura	Integer	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 20 a classe Infraestrutura tem como objetivo armazenar as informações referentes as Infraestrutura do imóvel.

Quadro 20 – Descrição da Classe Infraestrutura

Atributo	Tipo	Descrição
codigoInfraestrutura	Integer	Código para identificação da ocupação atual.
nomeInfraestrutura	String	Nome de identificação da ocupação atual.
descricaoInfraestrutura	String	Descrição da ocupação atual.

Fonte: Elaborado pelos autores.

No Quadro 21 a classe Multirelacional Instalação tem como objetivo armazenar as informações referentes as Infraestrutura do imóvel.

Quadro 21 – Descrição da Classe MultirelacionalInstalacao

Atributo	Tipo	Descrição
codigoInstalacao	Integer	Código para identificação da instalação.
dataInstalacao	Date	Data em que a instalação foi feita.
situacaoInstalacao	String	Situação da instalação.

Fonte: Elaborado pelos autores.

No Quadro 22 a classe Enum tem como objetivo definir o status de um registro.

Quadro 22 – Descrição da Classe EnumStatus

Atributo	Tipo	Descrição

status	Integer	Status ou estado atual de um registro.

Fonte: Elaborado pelos autores.

No Quadro 23 a classe Processo tem como objetivo conter as informações relacionadas ao processo de aprovação do documento de obras.

Quadro 23 – Descrição da Classe Processo

Atributo	Tipo	Descrição
codigoEvento	Integer	Código para identificação do processo
statusEvento	Boolean	Identificação do status do processo, indicando se está ativo ou desativado.
dataAprovacao	Date	Data em que a aprovação foi realizada.
situacao	String	Estado atual do andamento do processo.
protocolo	String	Protocolo de identificação do processo.

Fonte: Elaborado pelos autores.

No Quadro 24 a classe Tipo Processo visa definir qual tipo de processo será armazenado.

Quadro 24 – Descrição da Classe TipoProcesso

Atributo	Tipo	Descrição
idTipoProcesso	Integer	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 25 a classe Etapa tem como objetivo armazenar em qual estágio o processo se encontra.

Quadro 25 – Descrição da Classe Etapa

Atributo	Tipo	Descrição
idEtapa	Integer	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 26 a classe Tipo Usuário tem como objetivo armazenar os tipos de documentos existentes no processo.

Quadro 26 – Descrição da Classe TipoDocumento

Atributo	Tipo	Descrição
idTipoDocumento	Integer	Código para identificação do tipo de documento
nomeTipoProcesso	String	Nome de identificação do tipo de processo
descricaoTipoProcesso	String	Informações sobre a descrição do tipo de processo

Fonte: Elaborado pelos autores.

No Quadro 27 a classe Documento Processo é responsável pela representação do documento associado a todo o processo realizado.

Quadro 27 – Descrição da Classe DocumentoProcesso

Atributo	Tipo	Descrição
idDocumentoProcesso	Integer	Código para identificação do documento do processo
descricao	String	Informação sobre como foi o andamento do processo
observacao	String	Observação sobre se houve alguma objeção durante o desenvolvimento do processo.
documento	String	Documento do processo realizado
situacao	String	Como está o andamento do documento do processo

Fonte: Elaborado pelos autores.

No Quadro 28 a classe Multirelacional Tipo Documento Etapa tem como objetivo representar os relacionamentos entre Etapa e TipoDocumento.

Quadro 28 – Descrição da Classe Multirelacional TipoDocumentoEtapa

Atributo	Tipo	Descrição
idTipoDocumento	Integer	Código para identificação do tipo de documento
nomeTipoProcesso	String	Nome de identificação do tipo de processo
descricaoTipoProcesso	String	Informações sobre a descrição do tipo de processo

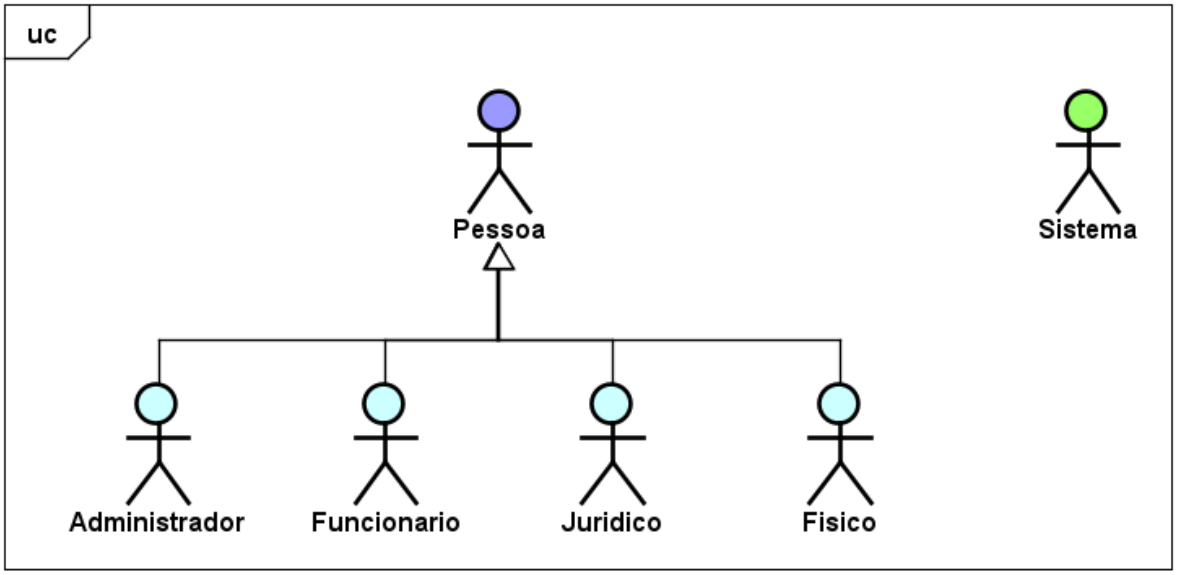
Fonte: Elaborado pelos autores.

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 (2007) 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 2.

Figura 5 – Diagrama de Atores



Fonte: Elaborado pelos autores.

Quadro 29 – 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 ter por objetivo generalizar (representar) todos os usuários que acessam o sistema.
Administrador	O ator secretário geral tem acesso à todas as funcionalidades implementas no sistema que são voltadas ao usuário.
Funcionário	O ator funcionário tem acesso à todas as funcionalidades implementas no sistema que são voltadas ao usuário, porém quando o mesmo efetua uma ação de impacto (cadastrar, alterar ou excluir), está ação é salva na auditoria de ações.
Jurídico	O ator jurídico tem acesso à visualização de todos dados dentro do sistema, exceto senhas de outros usuários (dado sensível).
Físico	O ator físico tem acesso à visualização de todos dados dentro do sistema, exceto dados sensíveis de outros usuários (como senha, cpf, cnpj, rg, ie etc).

Fonte: Elaborado pelos autores.

3.4 LISTA DE CASOS DE USO

Neste subtópico, iremos abordar o que cada ator pode realizar dentro sistema. A fim de generalizar os dados de saída de cada ação, predefinimos as possíveis mensagens que serão mostradas ao ator:

Quadro 29 – Mensagens de saída

Identificação	Mensagem	Descrição
Msg01	Cadastrado com sucesso.	Está mensagem ocorre quando o ator cadastra um novo registro no banco de dados.
Msg02	Alterado com sucesso.	Está mensagem ocorre quando o ator altera um registro no banco de dados.
Msg03	Excluído com sucesso.	Está mensagem ocorre quando o ator exclui um registro no banco de dados.
Msg04	Dados objetos.	Está mensagem ocorre quando o ator requisita a lista de uma classe.
Msg05	Dados objeto.	Está mensagem ocorre quando o ator requisita dados de um registro.
Msg11	Deseja realmente excluir?	Está mensagem ocorre quando o ator requisita a exclusão de um registro para o sistema.
Msg21	Dados vazios!	Está mensagem ocorre quando os dados para cadastro ou alteração estão vazios.
Msg22	Dados inválidos!	Está mensagem ocorre quando os dados para cadastro ou alteração são inválidos.

Fonte: Elaborado pelos autores.

O primeiro dígito da mensagem representa seu tipo:

- 0 para ações bem-sucedidas;
- 1 para ações de confirmação (espera um resultado externo);
- 2 para ações malsucedidas.

O segundo dígito é a identificação da mensagem dentro do seu tipo de mensagem.

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:

Quadro 30 – 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	Msg01
02	Administrador cadastra cidade	Dados Cidade	Cadastrar Cidade	Msg01

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
03	Administrador cadastra bairro	Dados Bairro	Cadastrar Bairro	Msg01
04	Administrador cadastra tipo logradouro	Dados TipoLogradouro	Cadastrar TipoLogradouro	Msg01
05	Administrador cadastra logradouro	Dados Logradouro	Cadastrar Logradouro	Msg01
06	Administrador cadastra munícipe	Dados Municipe	Cadastrar Municipe	Msg01
07	Administrador cadastra engenheiro	Dados Engenheiro	Cadastrar Engenheiro	Msg01
08	Administrador cadastra fiscal	Dados Fiscal	Cadastrar Fiscal	Msg01
09	Administrador cadastra tipo usuário	Dados TipoUsuario	Cadastrar TipoUsuario	Msg01
10	Administrador cadastra usuário	Dados Usuario	Cadastrar Usuario	Msg01
11	Administrador cadastra auditoria	Dados Auditoria	Cadastrar Auditoria	Msg01
12	Administrador cadastra imóvel	Dados Imovel	Cadastrar Imovel	Msg01
13	Administrador cadastra tipo processo	Dados TipoProcesso	Cadastrar TipoProcesso	Msg01
14	Administrador cadastra processo	Dados Processo	Cadastrar Processo	Msg01
15	Administrador cadastra etapa	Dados Etapa	Cadastrar Etapa	Msg01
16	Administrador cadastra tipo documento	Dados TipoDocumento	Cadastrar TipoDocumento	Msg01
17	Administrador cadastra documento processo	Dados DocumentoProcesso	Cadastrar DocumentoProcesso	Msg01
18	Administrador altera estado	Dados Estado	Alterar Estado	Msg02
19	Administrador altera cidade	Dados Cidade	Alterar Cidade	Msg02
20	Administrador altera bairro	Dados Bairro	Alterar Bairro	Msg02
21	Administrador altera tipo logradouro	Dados TipoLogradouro	Alterar TipoLogradouro	Msg02
22	Administrador altera logradouro	Dados Logradouro	Alterar Logradouro	Msg02
23	Administrador altera munícipe	Dados Municipe	Alterar Municipe	Msg02
24	Administrador altera engenheiro	Dados Engenheiro	Alterar Engenheiro	Msg02
25	Administrador altera fiscal	Dados Fiscal	Alterar Fiscal	Msg02
26	Administrador altera tipo usuário	Dados TipoUsuario	Alterar TipoUsuario	Msg02
27	Administrador altera usuário	Dados Usuario	Alterar Usuario	Msg02
28	Administrador altera auditoria	Dados Auditoria	Alterar Auditoria	Msg02
29	Administrador altera imóvel	Dados Imovel	Alterar Imovel	Msg02
30	Administrador altera tipo processo	Dados TipoProcesso	Alterar TipoProcesso	Msg02

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
31	Administrador altera processo	Dados Processo	Alterar Processo	Msg02
32	Administrador altera etapa	Dados Etapa	Alterar Etapa	Msg02
33	Administrador altera tipo documento	Dados TipoDocumento	Alterar TipoDocumento	Msg02
34	Administrador altera documento processo	Dados DocumentoProcesso	Alterar DocumentoProcesso	Msg02
35	Administrador exclui estado	Id do Estado	Excluir Estado	Msg03
36	Administrador exclui cidade	Id da Cidade	Excluir Cidade	Msg03
37	Administrador exclui bairro	Id do Bairro	Excluir Bairro	Msg03
38	Administrador exclui tipo logradouro	Id do TipoLogradouro	Excluir TipoLogradouro	Msg03
39	Administrador exclui logradouro	Id do Logradouro	Excluir Logradouro	Msg03
40	Administrador exclui munícipe	Id do Municipe	Excluir Municipe	Msg03
41	Administrador exclui engenheiro	Id do Engenheiro	Excluir Engenheiro	Msg03
42	Administrador exclui fiscal	Id do Fiscal	Excluir Fiscal	Msg03
43	Administrador exclui tipo usuário	Id do TipoUsuario	Excluir TipoUsuario	Msg03
44	Administrador exclui usuário	Id do Usuario	Excluir Usuario	Msg03
45	Administrador exclui auditoria	Id do Auditoria	Excluir Auditoria	Msg03
46	Administrador exclui imóvel	Id do Imovel	Excluir Imovel	Msg03
47	Administrador exclui tipo processo	Id do TipoProcesso	Excluir TipoProcesso	Msg03
48	Administrador exclui processo	Id do Processo	Excluir Processo	Msg03
49	Administrador exclui etapa	Id da Etapa	Excluir Etapa	Msg03
50	Administrador exclui tipo documento	Id do TipoDocumento	Excluir TipoDocumento	Msg03
51	Administrador exclui documento processo	Id do DocumentoProcesso	Excluir DocumentoProcesso	Msg03
52	Administrador consulta estado		Consultar Estado	Msg05
53	Administrador consulta cidade		Consultar Cidade	Msg05
54	Administrador consulta bairro		Consultar Bairro	Msg05
55	Administrador consulta tipo logradouro		Consultar TipoLogradouro	Msg05
56	Administrador consulta logradouro		Consultar Logradouro	Msg05
57	Administrador consulta munícipe		Consultar Municipe	Msg05
58	Administrador consulta engenheiro		Consultar Engenheiro	Msg05
59	Administrador consulta fiscal		Consultar Fiscal	Msg05

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
60	Administrador consulta tipo usuário		Consultar TipoUsuario	Msg05
61	Administrador consulta usuário		Consultar Usuario	Msg05
62	Administrador consulta auditoria		Consultar Auditoria	Msg05
63	Administrador consulta imóvel		Consultar Imovel	Msg05
64	Administrador consulta tipo processo		Consultar TipoProcesso	Msg05
65	Administrador consulta processo		Consultar Processo	Msg05
66	Administrador consulta etapa		Consultar Etapa	Msg05
67	Administrador consulta tipo documento		Consultar TipoDocumento	Msg05
68	Administrador consulta documento processo		Consultar DocumentoProcesso	Msg05

Fonte: Elaborado pelos autores.

Quadro 31 – Lista de Casos de Uso: Ações do Funcionário

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Funcionário cadastra estado	Dados Estado	Cadastrar Estado	Msg01
02	Funcionário cadastra cidade	Dados Cidade	Cadastrar Cidade	Msg01
03	Funcionário cadastra bairro	Dados Bairro	Cadastrar Bairro	Msg01
04	Funcionário cadastra tipo logradouro	Dados TipoLogradouro	Cadastrar TipoLogradouro	Msg01
05	Funcionário cadastra logradouro	Dados Logradouro	Cadastrar Logradouro	Msg01
06	Funcionário cadastra munícipe	Dados Municipe	Cadastrar Municipe	Msg01
07	Funcionário cadastra engenheiro	Dados Engenheiro	Cadastrar Engenheiro	Msg01
08	Funcionário cadastra fiscal	Dados Fiscal	Cadastrar Fiscal	Msg01
09	Funcionário cadastra tipo usuário	Dados TipoUsuario	Cadastrar TipoUsuario	Msg01
10	Funcionário cadastra usuário	Dados Usuario	Cadastrar Usuario	Msg01
11	Funcionário cadastra auditoria	Dados Auditoria	Cadastrar Auditoria	Msg01
12	Funcionário cadastra imóvel	Dados Imovel	Cadastrar Imovel	Msg01
13	Funcionário cadastra tipo processo	Dados TipoProcesso	Cadastrar TipoProcesso	Msg01
14	Funcionário cadastra processo	Dados Processo	Cadastrar Processo	Msg01
15	Funcionário cadastra etapa	Dados Etapa	Cadastrar Etapa	Msg01
16	Funcionário cadastra tipo documento	Dados TipoDocumento	Cadastrar TipoDocumento	Msg01
17	Funcionário cadastra documento processo	Dados DocumentoProcesso	Cadastrar DocumentoProcesso	Msg01

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
18	Funcionário altera estado	Dados Estado	Alterar Estado	Msg02
19	Funcionário altera cidade	Dados Cidade	Alterar Cidade	Msg02
20	Funcionário altera bairro	Dados Bairro	Alterar Bairro	Msg02
21	Funcionário altera tipo logradouro	Dados TipoLogradouro	Alterar TipoLogradouro	Msg02
22	Funcionário altera logradouro	Dados Logradouro	Alterar Logradouro	Msg02
23	Funcionário altera munícipe	Dados Municipe	Alterar Municipe	Msg02
24	Funcionário altera engenheiro	Dados Engenheiro	Alterar Engenheiro	Msg02
25	Funcionário altera fiscal	Dados Fiscal	Alterar Fiscal	Msg02
26	Funcionário altera tipo usuário	Dados TipoUsuario	Alterar TipoUsuario	Msg02
27	Funcionário altera usuário	Dados Usuario	Alterar Usuario	Msg02
28	Funcionário altera auditoria	Dados Auditoria	Alterar Auditoria	Msg02
29	Funcionário altera imóvel	Dados Imovel	Alterar Imovel	Msg02
30	Funcionário altera tipo processo	Dados TipoProcesso	Alterar TipoProcesso	Msg02
31	Funcionário altera processo	Dados Processo	Alterar Processo	Msg02
32	Funcionário altera etapa	Dados Etapa	Alterar Etapa	Msg02
33	Funcionário altera tipo documento	Dados TipoDocumento	Alterar TipoDocumento	Msg02
34	Funcionário altera documento processo	Dados DocumentoProcesso	Alterar DocumentoProcesso	Msg02
35	Funcionário exclui estado	Id do Estado	Excluir Estado	Msg03
36	Funcionário exclui cidade	Id da Cidade	Excluir Cidade	Msg03
37	Funcionário exclui bairro	Id do Bairro	Excluir Bairro	Msg03
38	Funcionário exclui tipo logradouro	Id do TipoLogradouro	Excluir TipoLogradouro	Msg03
39	Funcionário exclui logradouro	Id do Logradouro	Excluir Logradouro	Msg03
40	Funcionário exclui munícipe	Id do Municipe	Excluir Municipe	Msg03
41	Funcionário exclui engenheiro	Id do Engenheiro	Excluir Engenheiro	Msg03
42	Funcionário exclui fiscal	Id do Fiscal	Excluir Fiscal	Msg03
43	Funcionário exclui tipo usuário	Id do TipoUsuario	Excluir TipoUsuario	Msg03
44	Funcionário exclui usuário	Id do Usuario	Excluir Usuario	Msg03
45	Funcionário exclui auditoria	Id do Auditoria	Excluir Auditoria	Msg03
46	Funcionário exclui imóvel	Id do Imovel	Excluir Imovel	Msg03
47	Funcionário exclui tipo processo	Id do TipoProcesso	Excluir TipoProcesso	Msg03
48	Funcionário exclui processo	Id do Processo	Excluir Processo	Msg03
49	Funcionário exclui etapa	Id da Etapa	Excluir Etapa	Msg03
50	Funcionário exclui tipo documento	Id do TipoDocumento	Excluir TipoDocumento	Msg03
51	Funcionário exclui documento processo	Id do DocumentoProcesso	Excluir DocumentoProcesso	Msg03
52	Funcionário consulta estado		Consultar Estado	Msg05
53	Funcionário consulta cidade		Consultar Cidade	Msg05

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
54	Funcionário consulta bairro		Consultar Bairro	Msg05
55	Funcionário consulta tipo logradouro		Consultar TipoLogradouro	Msg05
56	Funcionário consulta logradouro		Consultar Logradouro	Msg05
57	Funcionário consulta munícipe		Consultar Municipe	Msg05
58	Funcionário consulta engenheiro		Consultar Engenheiro	Msg05
59	Funcionário consulta fiscal		Consultar Fiscal	Msg05
60	Funcionário consulta tipo usuário		Consultar TipoUsuario	Msg05
61	Funcionário consulta usuário		Consultar Usuario	Msg05
62	Funcionário consulta auditoria		Consultar Auditoria	Msg05
63	Funcionário consulta imóvel		Consultar Imovel	Msg05
64	Funcionário consulta tipo processo		Consultar TipoProcesso	Msg05
65	Funcionário consulta processo		Consultar Processo	Msg05
66	Funcionário consulta etapa		Consultar Etapa	Msg05
67	Funcionário consulta tipo documento		Consultar TipoDocumento	Msg05
68	Funcionário consulta documento processo		Consultar DocumentoProcesso	Msg05

Fonte: Elaborado pelos autores.

Quadro 32 – Lista de Casos de Uso: Ações do Jurídico

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Jurídico consulta estado		Consultar Estado	Msg05
02	Jurídico consulta cidade		Consultar Cidade	Msg05
03	Jurídico consulta bairro		Consultar Bairro	Msg05
04	Jurídico consulta tipo logradouro		Consultar TipoLogradouro	Msg05
05	Jurídico consulta logradouro		Consultar Logradouro	Msg05
06	Jurídico consulta munícipe		Consultar Municipe	Msg05
07	Jurídico consulta engenheiro		Consultar Engenheiro	Msg05
08	Jurídico consulta fiscal		Consultar Fiscal	Msg05
09	Jurídico consulta tipo usuário		Consultar TipoUsuario	Msg05
10	Jurídico consulta usuário		Consultar Usuario	Msg05
11	Jurídico consulta auditoria		Consultar Auditoria	Msg05
12	Jurídico consulta imóvel		Consultar Imovel	Msg05
13	Jurídico consulta tipo processo		Consultar TipoProcesso	Msg05
14	Jurídico consulta processo		Consultar Processo	Msg05
15	Jurídico consulta etapa		Consultar Etapa	Msg05

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
16	Jurídico consulta tipo documento		Consultar TipoDocumento	Msg05
17	Jurídico consulta documento processo		Consultar DocumentoProcesso	Msg05

Fonte: Elaborado pelos autores.

Quadro 33 – Lista de Casos de Uso: Ações do Físco

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Físico consulta estado		Consultar Estado	Msg05
02	Físico consulta cidade		Consultar Cidade	Msg05
03	Físico consulta bairro		Consultar Bairro	Msg05
04	Físico consulta tipo logradouro		Consultar TipoLogradouro	Msg05
05	Físico consulta logradouro		Consultar Logradouro	Msg05
06	Físico consulta munícipe		Consultar Municipe	Msg05
07	Físico consulta engenheiro		Consultar Engenheiro	Msg05
08	Físico consulta fiscal		Consultar Fiscal	Msg05
09	Físico consulta tipo usuário		Consultar TipoUsuario	Msg05
10	Físico consulta usuário		Consultar Usuario	Msg05
11	Físico consulta auditoria		Consultar Auditoria	Msg05
12	Físico consulta imóvel		Consultar Imovel	Msg05
13	Físico consulta tipo processo		Consultar TipoProcesso	Msg05
14	Físico consulta processo		Consultar Processo	Msg05
15	Físico consulta etapa		Consultar Etapa	Msg05
16	Físico consulta tipo documento		Consultar TipoDocumento	Msg05
17	Físico consulta documento processo		Consultar DocumentoProcesso	Msg05

Fonte: Elaborado pelos autores.

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

Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
01	Sistema lista estado		Listar Estado	Msg04
02	Sistema lista cidade		Listar Cidade	Msg04
03	Sistema lista bairro		Listar Bairro	Msg04
04	Sistema lista tipo logradouro		Listar TipoLogradouro	Msg04
05	Sistema lista logradouro		Listar Logradouro	Msg04
06	Sistema lista munícipe		Listar Municipe	Msg04
07	Sistema lista engenheiro		Listar Engenheiro	Msg04
08	Sistema lista fiscal		Listar Fiscal	Msg04
09	Sistema lista tipo usuário		Listar TipoUsuario	Msg04
10	Sistema lista usuário		Listar Usuario	Msg04
11	Sistema lista auditoria		Listar Auditoria	Msg04
12	Sistema lista imóvel		Listar Imovel	Msg04

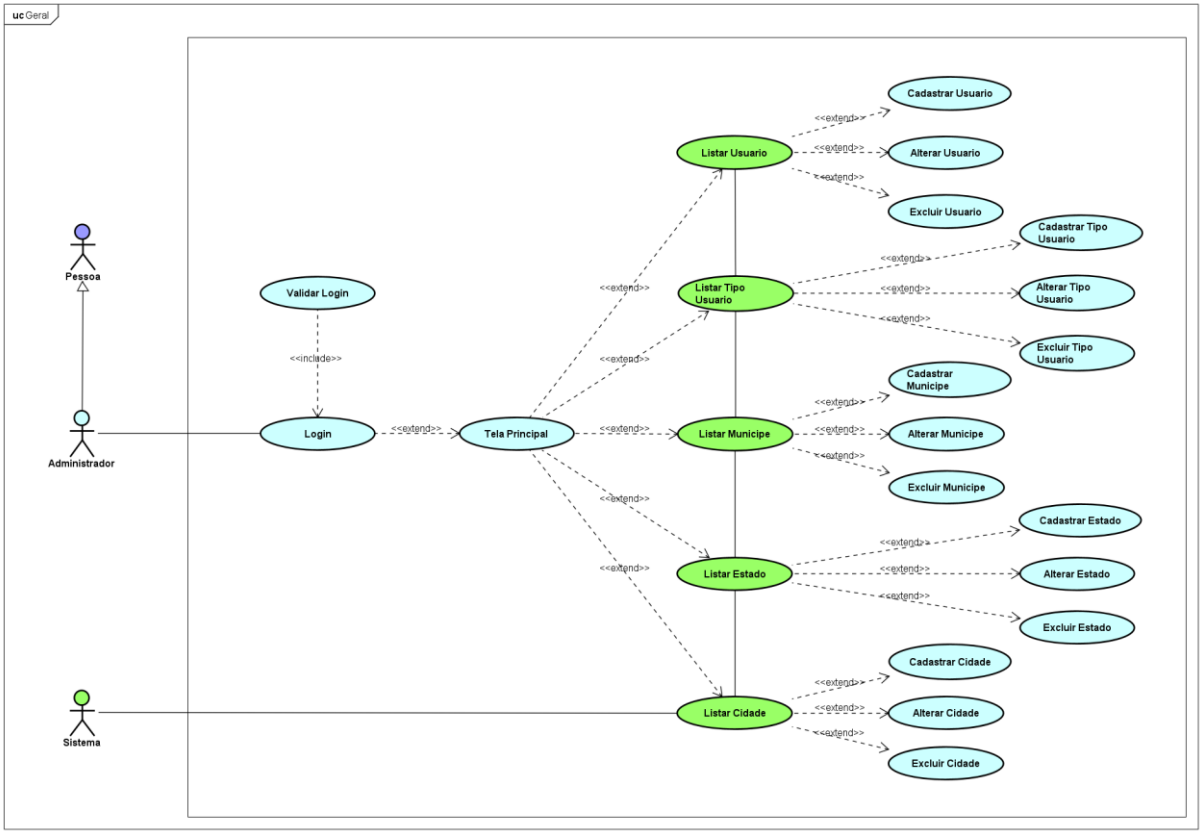
Nº	Descrição do Caso de Uso	Entrada	Caso de Uso	Resposta
13	Sistema lista tipo processo		Listar TipoProcesso	Msg04
14	Sistema lista processo		Listar Processo	Msg04
15	Sistema lista etapa		Listar Etapa	Msg04
16	Sistema lista tipo documento		Listar TipoDocumento	Msg04
17	Sistema lista documento processo		Listar DocumentoProcesso	Msg04

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. Segundo Pressman (2016), "o diagrama de casos de uso descreve o comportamento de um sistema do ponto de vista de um usuário externo", sendo uma ferramenta fundamental para a análise de requisitos e comunicação entre equipes.

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



Fonte: Elaborado pelos autores.

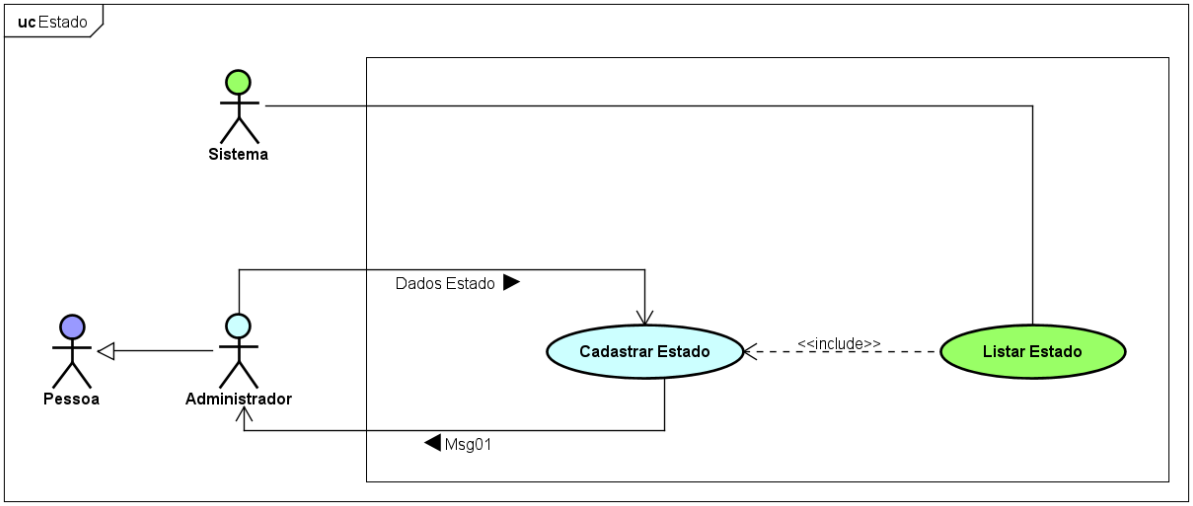
3.5. DIAGRAMA DE CASOS DE USO INDIVIDUAIS

Abordaremos dois casos de uso específicos, baseados no diagrama de caso de uso do administrador, que ilustram as principais responsabilidades deste ator no sistema. O papel do administrador é fundamental, pois ele é encarregado de tarefas como a manutenção de dados e a gestão de permissões, assegurando o funcionamento correto e seguro do sistema.

3.5.1 – Caso de uso: Administrador - Cadastrar Estado

A **Figura 4** mostra um diagrama de caso de uso que descreve a funcionalidade de Cadastrar Estado executada pelo administrador. Neste caso, o administrador interage com o sistema para cadastrar informações de estado.

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



Fonte: Elaborado pelos autores.

O Quadro 26 ilustra o procedimento para o caso de uso Cadastrar Estado, especificando as fases que um administrador realiza ao interagir com o sistema para inserir ou modificar informações de estado. Este procedimento assegura o registro adequado das informações, assegurando desta forma a integridade dos dados administrativos do sistema.

Quadro 26 – Fluxo do Caso de Uso: Adminsitrador – Cadastrar Estado

Fluxo do Caso de Uso	
Caso de Uso:	**Cadastrar Estado**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de cadastrar o estado.
Pré-condições:	Administrado logado e autenticado.

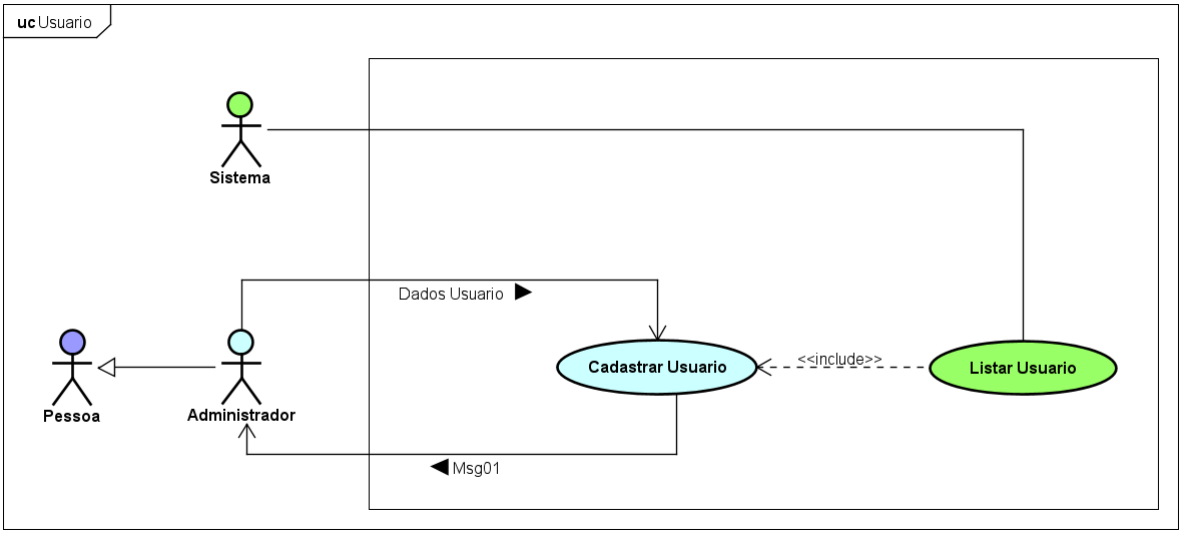
Pós-condições:	O sistema fecha o formulário e lista os estados.
Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
1. Administrador requisita formulário de cadastro de estado.	
	2. Sistema exibe modal de cadastro de estado.
3. Administrador requisita cadastro de estado.	
	4. Sistema valida os dados de entrada do estado.
	5. Sistema salva dados do estado no banco de dados.
	6. Sistema fecha modal de cadastro de estado.
	7. Sistema recarrega a lista de estados.
	8. Sistema exibe modal com a lista de estados.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

3.5.2 – Caso de uso: Administrador - Cadastrar Usuário

A Figura 6 ilustra o diagrama de caso de uso da funcionalidade "Cadastrar Usuário", onde o administrador se comunica com o sistema para estabelecer e administrar contas de usuários. Este caso de uso é crucial para gerir o acesso ao sistema, assegurando que novos utilizadores sejam cadastrados com as permissões apropriadas.

Figura 8 – Diagrama de Caso de Uso Específico: Administrador – Cadastrar Usuário



Fonte: Elaborado pelos autores.

O Quadro 27 ilustra o caso de uso Cadastrar Usuário, 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 16 – Fluxo do Caso de Uso: Adminsitrador – Cadastrar Usuário

Fluxo do Caso de Uso	
Caso de Uso:	**Cadastrar Usuario**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de cadastrar o usuário.
Pré-condições:	Administrado 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 cadastro de usuário.	

	10. Sistema exibe modal de cadastro de usuário.
11. Administrador requisita cadastro de usuário.	
	12. Sistema valida os dados de entrada do usuário.
	13. Sistema salva dados do estado no banco de dados.
	14. Sistema fecha modal de cadastro de usuário.
	15. Sistema recarrega a lista de usuários.
	16. Sistema exibe modal com a lista de usuários.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

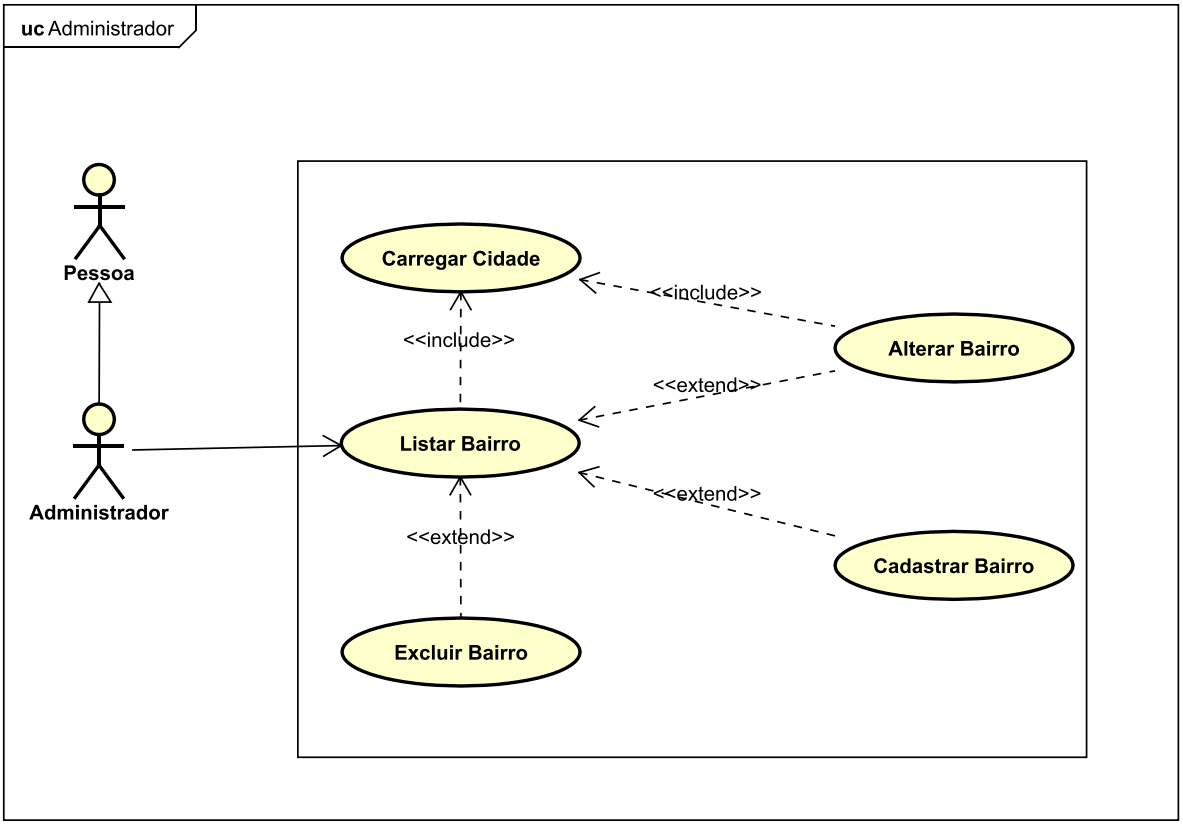
Quadro 17

Fonte: Elaborado pelos autores.

3.5.3 – Caso de uso: Administrador - Cadastrar Usuário

A Figura 8 ilustra o diagrama de caso de uso geral de bairro, onde o administrador se comunica com o sistema para estabelecer e administrar os bairros da cidade.

Figura 9 – Fluxo do Caso de Uso Geral de Bairro.



Fonte: Elaborado pelos autores.

O Quadro 28 ilustra o processo do caso de uso "Cadastrar Bairro", especificando as fases que o administrador percorre ao interagir com o sistema para inserir ou modificar dados de bairros. Este processo é crucial para assegurar que as informações sejam devidamente registradas, auxiliando na organização e administração eficaz das informações no sistema.

Quadro 18 – Fluxo do Caso de Uso: Administrador – Cadastrar Bairro

Fluxo do Caso de Uso	
Caso de Uso:	**Cadastrar Bairro**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de cadastrar o bairro.
Pré-condições:	Administrado logado e autenticado.
Pós-condições:	O sistema fecha o formulário e lista os bairros.
Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
17. Administrador requisita formulário de cadastro de bairro.	

	18. Sistema exibe modal de cadastro de bairro.
19. Administrador requisita cadastro de bairro.	
	20. Sistema valida os dados de entrada do bairro.
	21. Sistema salva dados do bairro no banco de dados.
	22. Sistema fecha modal de cadastro de bairro.
	23. Sistema recarrega a lista de bairro.
	24. Sistema exibe modal com a lista de bairros.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

O Quadro 29 ilustra o processo do caso de uso "Alterar Bairro", especificando as fases que o gestor deve percorrer para alterar as informações de bairros no software.

Quadro 19 – Fluxo do Caso de Uso: Administrador – Alterar Bairro

Fluxo do Caso de Uso	
Caso de Uso:	**Alterar Bairro**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de alterar o bairro.
Pré-condições:	Administrado logado e autenticado.
Pós-condições:	O sistema fecha o formulário e lista os bairros.
Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
1. Administrador requisita formulário de alteração de bairro.	
	2. Sistema exibe modal de alteração de bairro.
3. Administrador requisita alteração de bairro.	

	4. Sistema valida os dados de entrada do bairro.
	5. Sistema salva dados do bairro no banco de dados.
	6. Sistema fecha modal de alteração de bairro.
	7. Sistema recarrega a lista de bairro.
	8. Sistema exibe modal com a lista de bairros.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

O Quadro 30 ilustra o processo do caso de uso "Excluir Bairro", especificando as fases que o gestor precisa cumprir para apagar dados de bairros do sistema.

Quadro 20 – Fluxo do Caso de Uso: Administrador – Excluir Bairro

Fluxo do Caso de Uso	
Caso de Uso:	**Excluir Bairro**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de excluir o bairro.
Pré-condições:	Administrado logado e autenticado.
Pós-condições:	O sistema fecha o formulário e lista os bairros.
Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
1. Administrador requisita formulário de exclusão de bairro.	
	2. Sistema exibe modal de exclusão de bairro.
3. Administrador requisita exclusão de bairro.	
	4. Sistema valida os dados de entrada do bairro.
	5. Sistema exclui os dados do bairro no banco de dados.

	6. Sistema fecha modal de exclusão de bairro.
	7. Sistema recarrega a lista de bairro.
	8. Sistema exibe modal com a lista de bairros.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

O Quadro 31 ilustra o processo do caso de uso "Listar Bairro", especificando as fases que o gestor deve percorrer para obter informações sobre os bairros registrados no software.

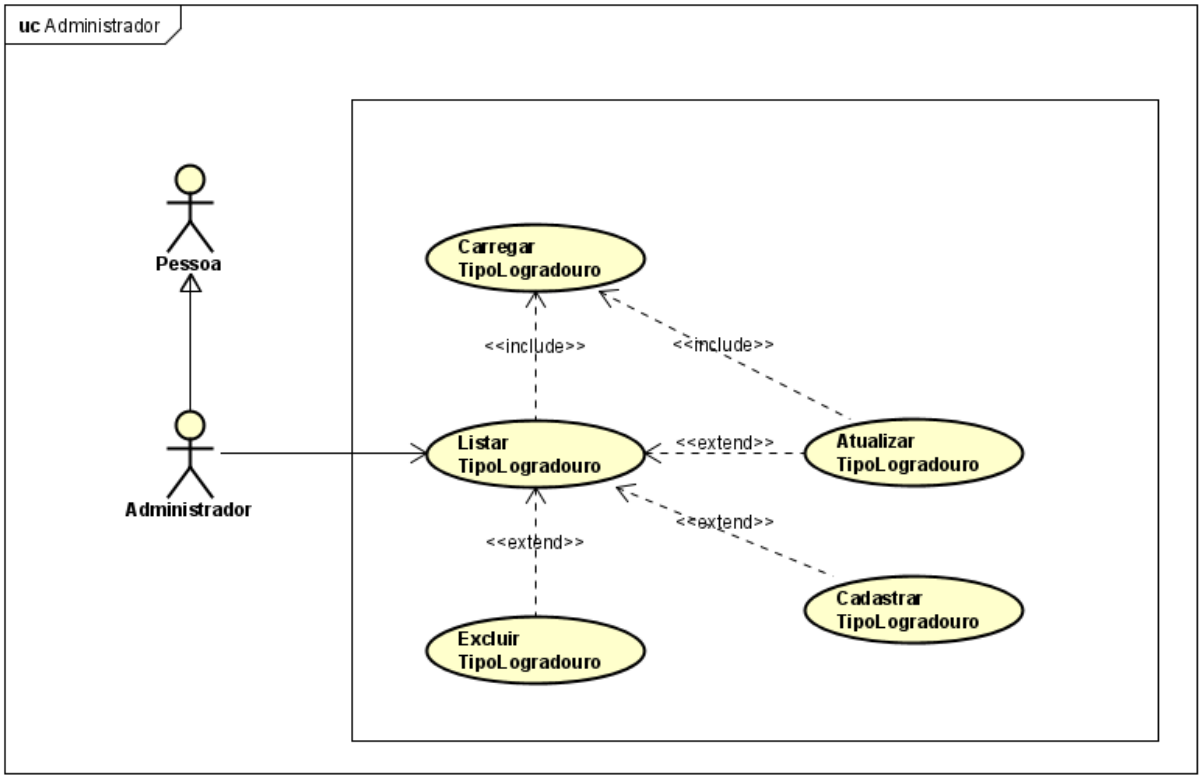
Quadro 21 – Fluxo do Caso de Uso: Administrador – Listar Bairro

Fluxo do Caso de Uso	
Caso de Uso:	**Excluir Bairro**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de listar o bairro.
Pré-condições:	Administrado logado e autenticado.
Pós-condições:	lista os bairros.
Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
1. Administrador requisita lista de bairro.	
	2. Sistema exibe tabela de listagem de bairro.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

A Figura 9 representa o processo do caso de uso geral para a gestão de TipoLogradouro no sistema. Este processo descreve as interações entre os usuários e o sistema para registrar, alterar ou apagar dados sobre tipos de logradouro, garantindo uma administração eficaz e estruturada dessas informações.

Figura 10 – Fluxo do Caso de Uso Geral de TipoLogradouro



Fonte: Elaborado pelos autores.

O Quadro 32 ilustra o processo do caso de uso geral de TipoLogradouro, especificando as fases que os usuários percorrem para administrar informações sobre os tipos de logradouro no sistema. Este processo é crucial para assegurar que os dados sejam registrados, atualizados e eliminados de maneira eficiente, auxiliando na organização das informações.

Quadro 22 – Fluxo do Caso de Uso Geral de TipoLogradouro

Fluxo do Caso de Uso	
Caso de Uso:	**Cadastrar TipoLogradouro**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de cadastrar o tipologradouro.
Pré-condições:	Administrado logado e autenticado.
Pós-condições:	O sistema fecha o formulário e lista os tipologradouros.

Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
25. Administrador requisita formulário de cadastro de tipologradouro.	
	26. Sistema exibe modal de cadastro de tipologradouro.
27. Administrador requisita cadastro de tipologradouro.	
	28. Sistema valida os dados de entrada do tipologradouro.
	29. Sistema salva dados do tipologradouro no banco de dados.
	30. Sistema fecha modal de cadastro de tipologradouro.
	31. Sistema recarrega a lista de tipologradouro.
	32. Sistema exibe modal com a lista de tipologradouros.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

O Quadro 33 ilustra o processo do caso de uso "Alterar TipoLogradouro", especificando as fases que o gestor deve cumprir para alterar dados sobre os tipos de logradouro no sistema.

Quadro 23 – Fluxo do Caso de Uso: Administrador – Alterar TipoLogradouro

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

9. Administrador requisita formulário de alteração de tipologradouro.	
	10. Sistema exibe modal de alteração de tipologradouro.
11. Administrador requisita alteração de tipologradouro.	
	12. Sistema valida os dados de entrada do tipologradouro.
	13. Sistema salva dados do tipologradouro no banco de dados.
	14. Sistema fecha modal de alteração de tipologradouro.
	15. Sistema recarrega a lista de tipologradouro.
	16. Sistema exibe modal com a lista de tipologradouros.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

O Quadro 34 ilustra o processo do caso de uso "Excluir TipoLogradouro", especificando as fases que o administrador realiza para apagar dados sobre os tipos de logradouro do sistema.

Quadro 24 – Fluxo do Caso de Uso: Administrador – Excluir Bairro

Fluxo do Caso de Uso	
Caso de Uso:	**Excluir Bairro**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de excluir o tipologradouro.
Pré-condições:	Administrado logado e autenticado.
Pós-condições:	O sistema fecha o formulário e lista os tipologradouros.
Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
9. Administrador requisita formulário de exclusão de tipologradouro.	

	10. Sistema exibe modal de exclusão de tipologradouro.
11. Administrador requisita exclusão de tipologradouro.	
	12. Sistema valida os dados de entrada do tipologradouro.
	13. Sistema exclui os dados do tipologradouro no banco de dados.
	14. Sistema fecha modal de exclusão de tipologradouro.
	15. Sistema recarrega a lista de tipologradouro.
	16. Sistema exibe modal com a lista de tipologradouros.
Fluxo Alternativo	
	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

O Quadro 35 ilustra o processo do caso de uso "Listar TipoLogradouro", especificando as fases que o administrador realiza para apagar dados sobre os tipos de logradouro do sistema.

Quadro 25 – Fluxo do Caso de Uso: Administrador – Listar TipoLogradouro

Fluxo do Caso de Uso	
Caso de Uso:	**Excluir Bairro**
Ator Principal:	Administrador
Ator Secundário:	Sistema
Descrição:	Este caso de uso apresenta as ações em que o administrado deve passar e/ou pode passar para concluir a função de listar o tipologradouro.
Pré-condições:	Administrado logado e autenticado.
Pós-condições:	lista os tipologradouros.
Fluxo Normal	
Ações do Administrador	**Ações do Sistema**
3. Administrador requisita lista de tipologradouro.	
	4. Sistema exibe tabela de listagem de tipologradouro.
Fluxo Alternativo	

	1.1 Não é possível estabelecer conexão com o Sistema.
	4.1 Dados de entrada vazio. Sistema exibe Msg21
	4.2 Dados de entrada inválidos. Sistema exibe Msg22
	5.1 Não é possível estabelecer conexão com a API.
	5.2 Não é possível estabelecer conexão com o Banco de Dados.

Fonte: Elaborado pelos autores.

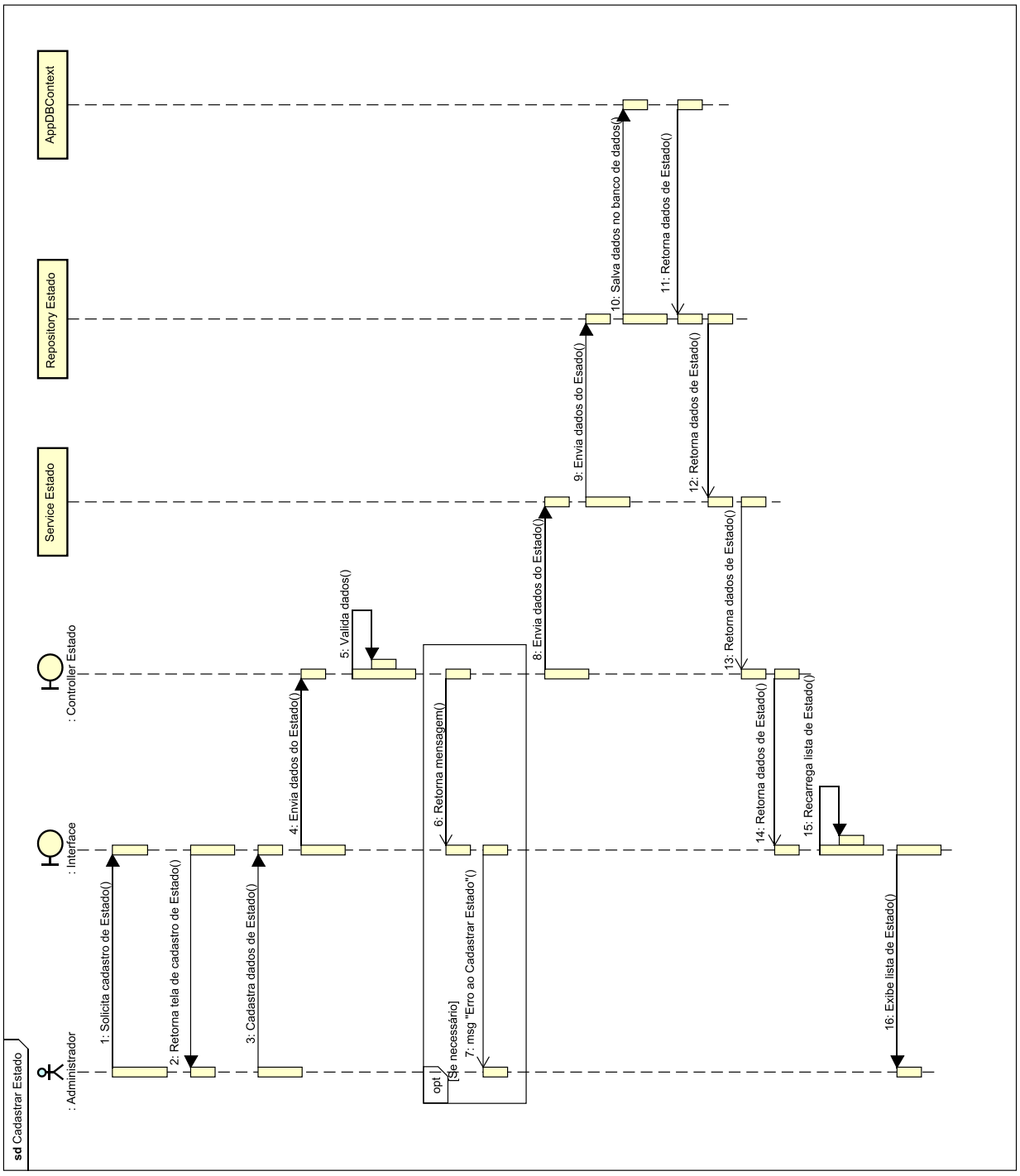
3.6. DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência é um dos principais diagramas UML (Unified Modeling Language) usados para descrever as interações entre objetos em um sistema, com foco na sequência temporal dessas interações. Segundo Sommerville (2007), “Um diagrama de sequência mostra as interações entre os atores e o sistema e entre os componentes do sistema, organizados em ordem cronológica”. É uma ferramenta importante para entender 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 10 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 11 – Diagrama de Sequência - Ator Administrador: Fluxo do cadastro de Estado



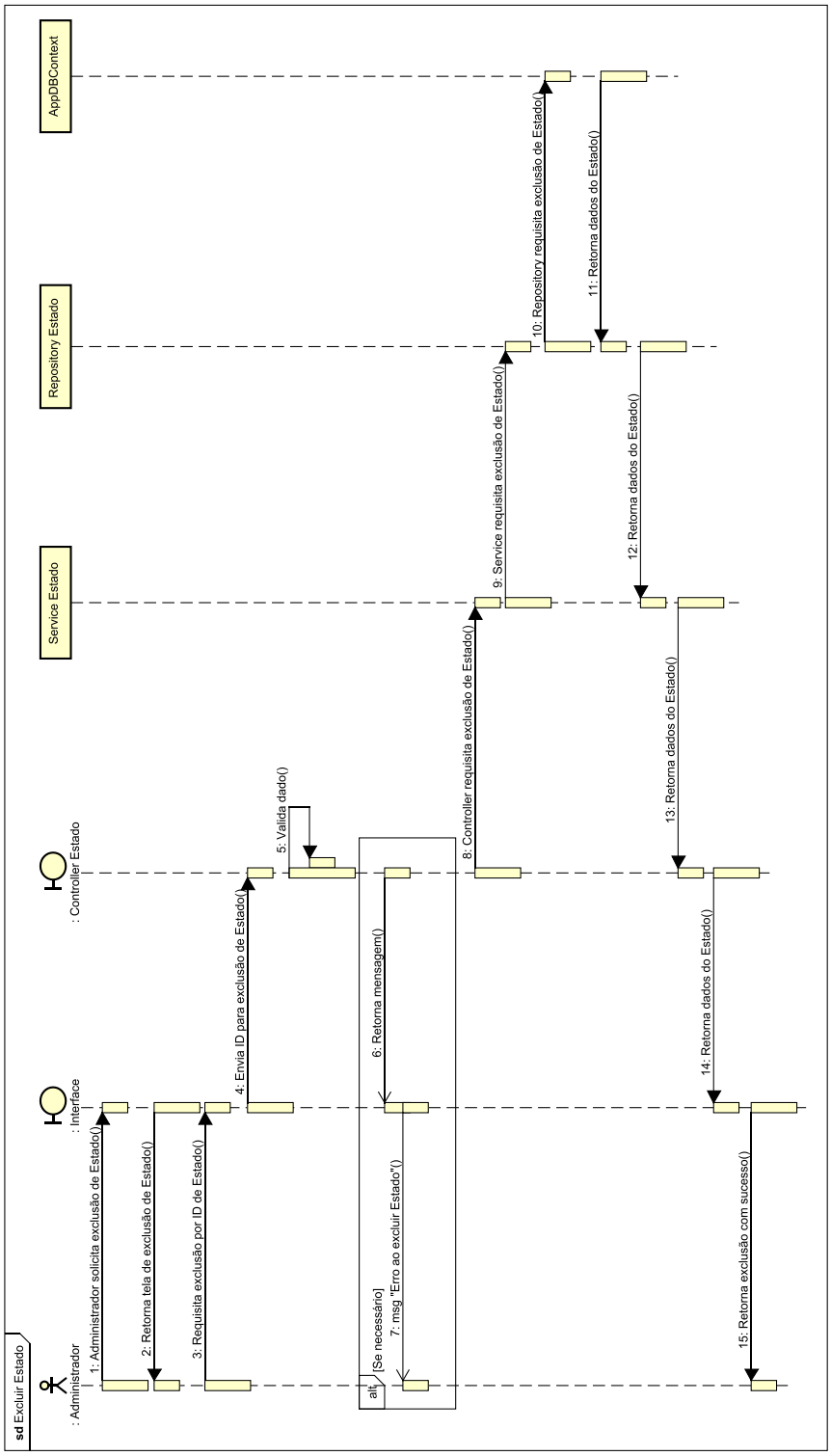
Fonte: Elaborado pelos Autores.

3.6.2 Diagrama de Sequência para Exclusão de Estado

Na Figura 11 são detalhados 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 12 – Diagrama de Sequência - Ator Administrador: Fluxo do excluir de Estado



Fonte: Elaborado pelos autores.

4 DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX) (3º SEMESTRE)

Segundo Fabricio Teixeira em Introdução e boas práticas em UX Design (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.

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:

Figura 13 – Cenário Engenheira de Obras da Prefeitura

Fonte: Elaborado pelos autores.

Figura 14 – Cenário Engenheira de Obras da Prefeitura

Fonte: Elaborado pelos autores.

4.2 DESCRIÇÃO DE PERSONAS

A descrição de personas são representações feitas de forma fictícias de usuário reais, são criadas para ajudar o designer e equipes de UX (*User Experience*), de forma que, a compreensão das necessidades, comportamentos e características dos usuários seja feita de forma assertiva.

De acordo com LISBOA (2017), “Utilizando personas, negócios podem ser mais estratégicos em alcançar seu público, pois elas podem claramente ilustrar para todos os stakeholders, incluindo a equipe de design.”

A seguir, serão apresentados dois exemplos de personas conforme o sistema abordado nesta pesquisa:

Figura 15 – Persona Engenheira Civil

Fonte: Elaborado pelos autores.

Figura 16 – Persona Engenheiro de Campo

Fonte: Elaborado pelos autores.

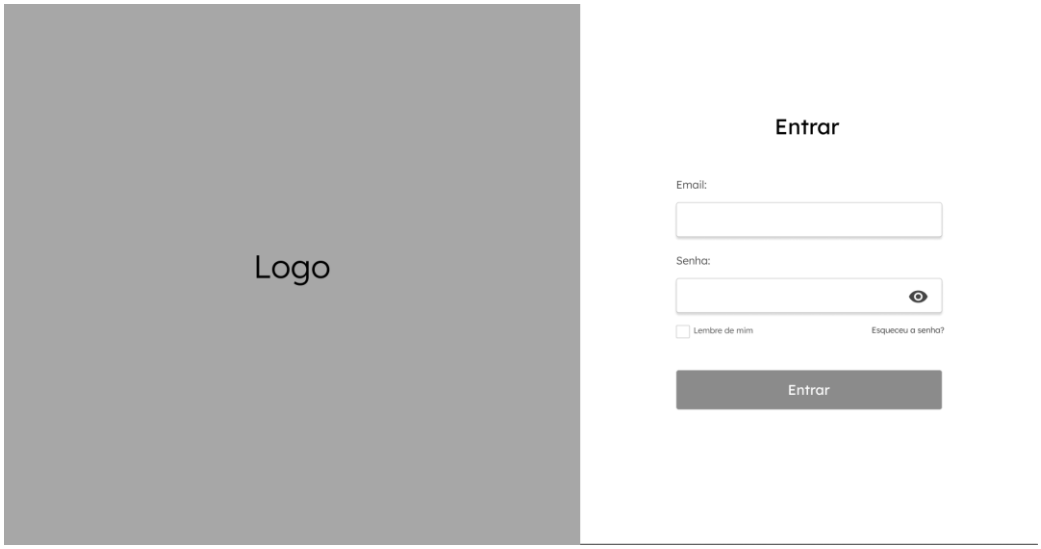
4.3 ESBOÇOS DE TELA (WIREFRAMES)

Um *wireframe* é uma representação visual esquemática e simplificada da estrutura de uma página da web, aplicativo ou interface de 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.

Durante o processo de design o *wireframe* se torna uma ferramenta valiosa, permitindo que a equipe coloque suas ideias no papel de maneira rápida e eficiente. 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.

Figura 17 – *Wireframe* da Tela de Login



Fonte: Elaborado pelos autores.

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.

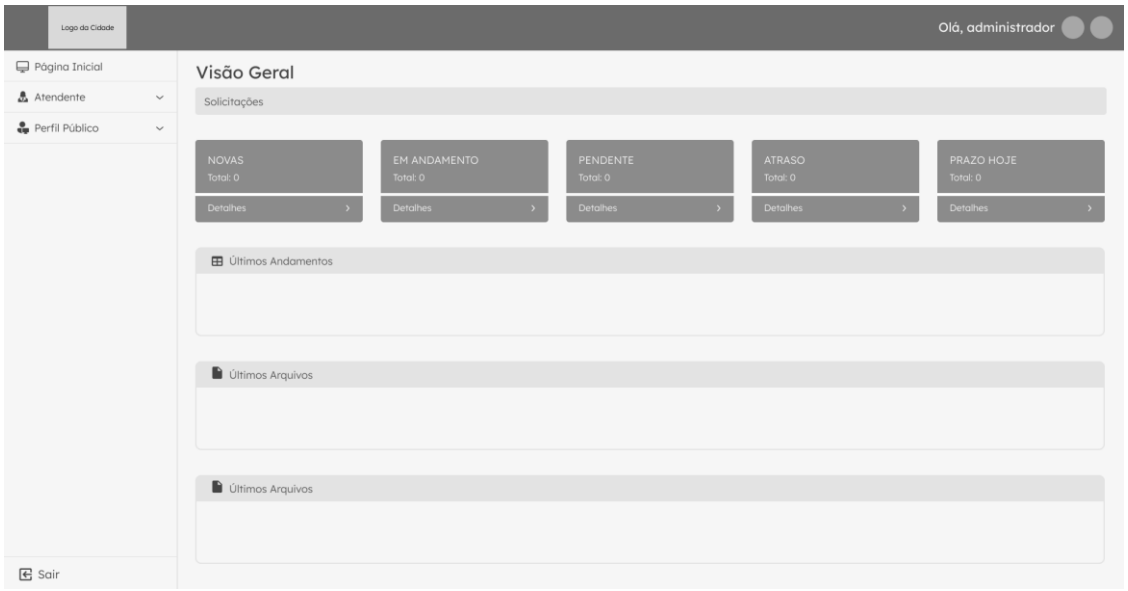
Os campos “Email” e “Senha” tem o objetivo de facilitar o acesso do usuário ao sistema. O design decisivo inclui posicionamento claro e espaçamento adequado para uma entrada de dados intuitiva, com a escolha de fontes legíveis contribuindo para uma experiência sem atritos.

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.

Figura 18 – Wireframe da Tela Inicial do Sistema



Fonte: Elaborado pelos autores.

Na 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.

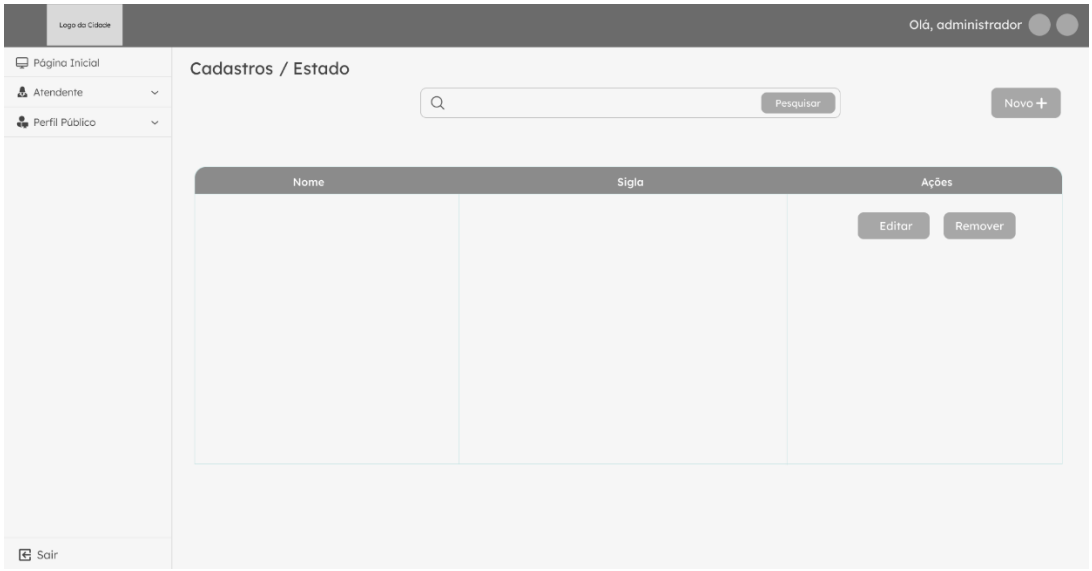
No topo da tela, o *Navbar* apresenta elementos essenciais, proporcionando uma navegação simplificada e personalizada. A presença da logo do sistema, opções de perfil, notificações, promove uma experiência centrada no usuário desde o início.

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.

Figura 19 – Wireframe Tela de Cadastro de Estado



Fonte: Elaborado pelos autores.

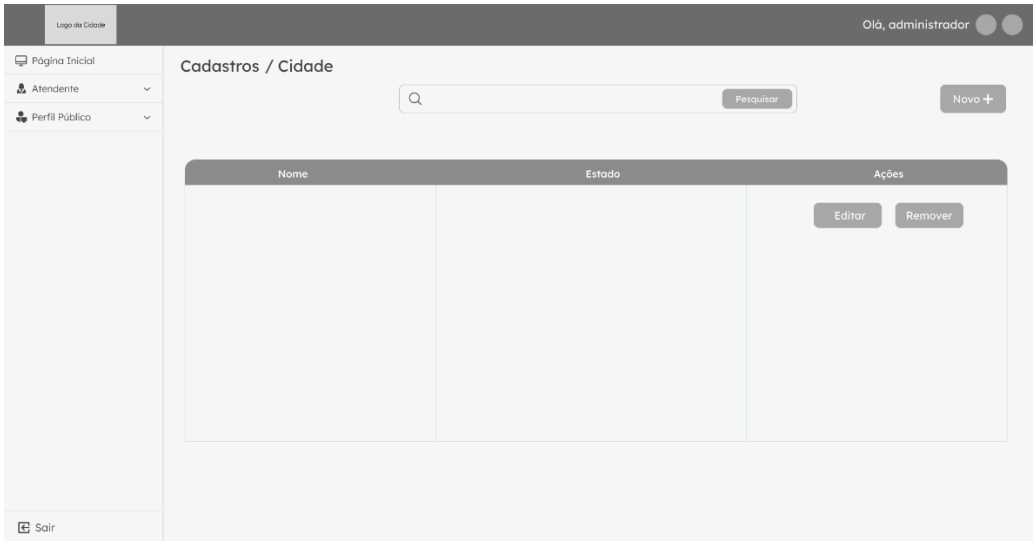
O *wireframe* tela de cadastro de estado oferece diversas funcionalidades. A barra de pesquisa se destaca como uma ferramenta vital para navegação eficiente. Permitindo a busca de estados específicos, os usuários têm acesso rápido às informações desejadas, simplificando a interação com o sistema.

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.

Figura 20 – Wireframe Tela de Cadastro de Cidade



Fonte: Elaborado pelos autores.

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.

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

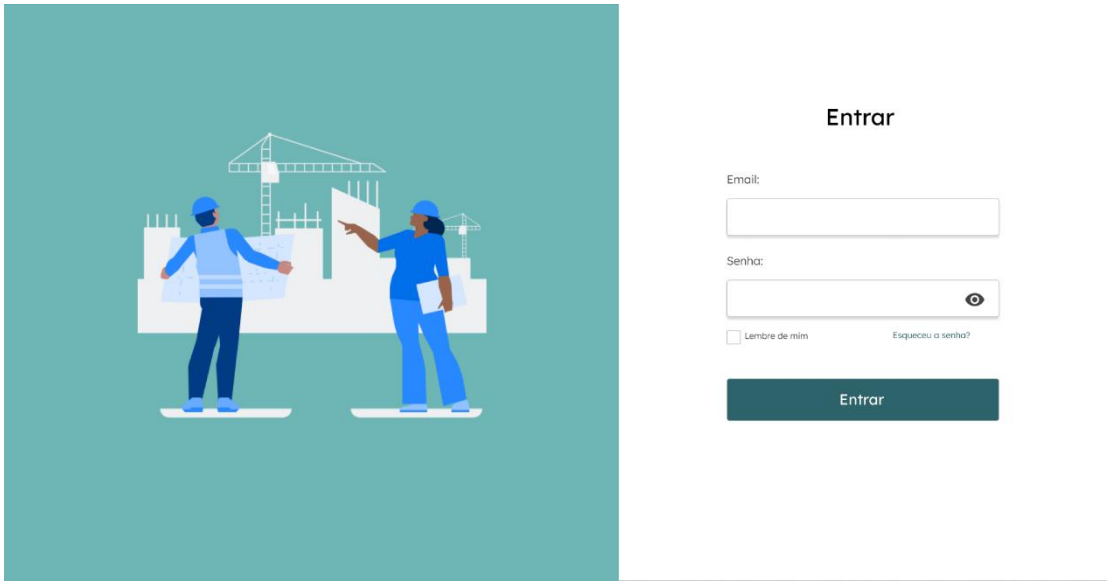
Os protótipos de tela são representações visuais interativas de interfaces do usuário (UI) usadas no Design de Experiência do Usuário (UX). Essas representações simulam a aparência e o comportamento de um produto, aplicativo ou site antes da implementação real.

Os protótipos 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.

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. Contudo, 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.

Figura 21 – Protótipo de Tela de Login



Fonte: Elaborado pelos autores.

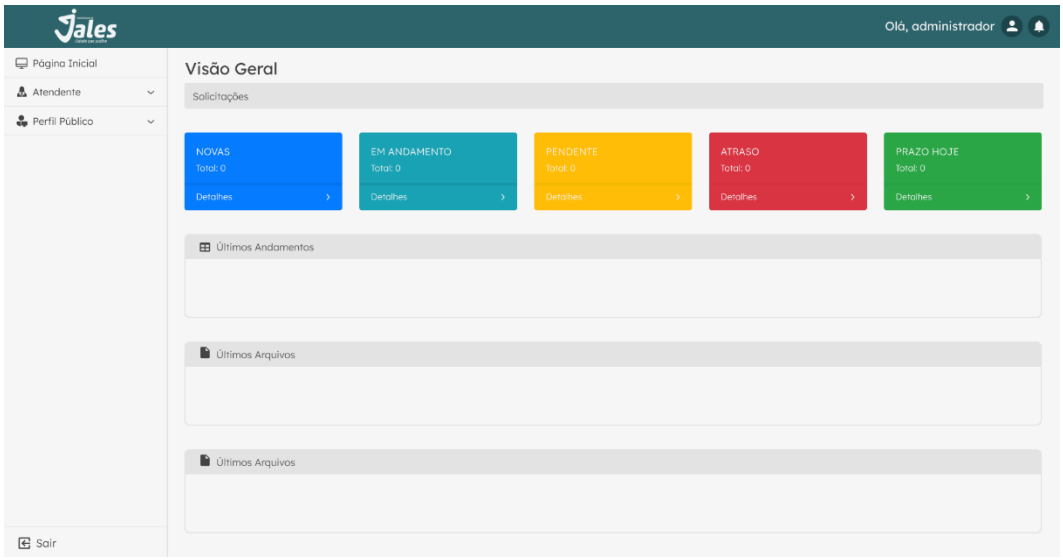
O protótipo da tela de login do sistema de gerenciamento de obras da Prefeitura de Jales foi cuidadosamente estilizada para refletir as cores distintivas da identidade visual da prefeitura. A paleta de cores escolhida não apenas contribui para uma estética atraente, mas também reforça a marca e cria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

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.

Figura 22 – Protótipo de Tela Inicial



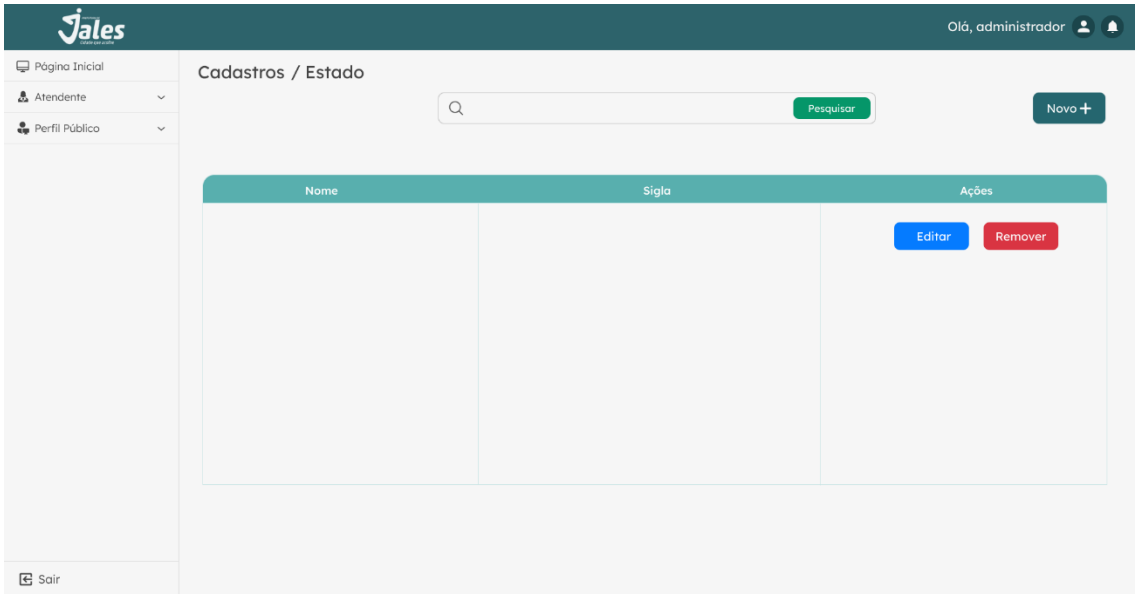
Fonte: Elaborado pelos autores.

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.

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.

Figura 23 – Protótipo da Tela de Cadastro de Estado



Fonte: Elaborado pelos autores.

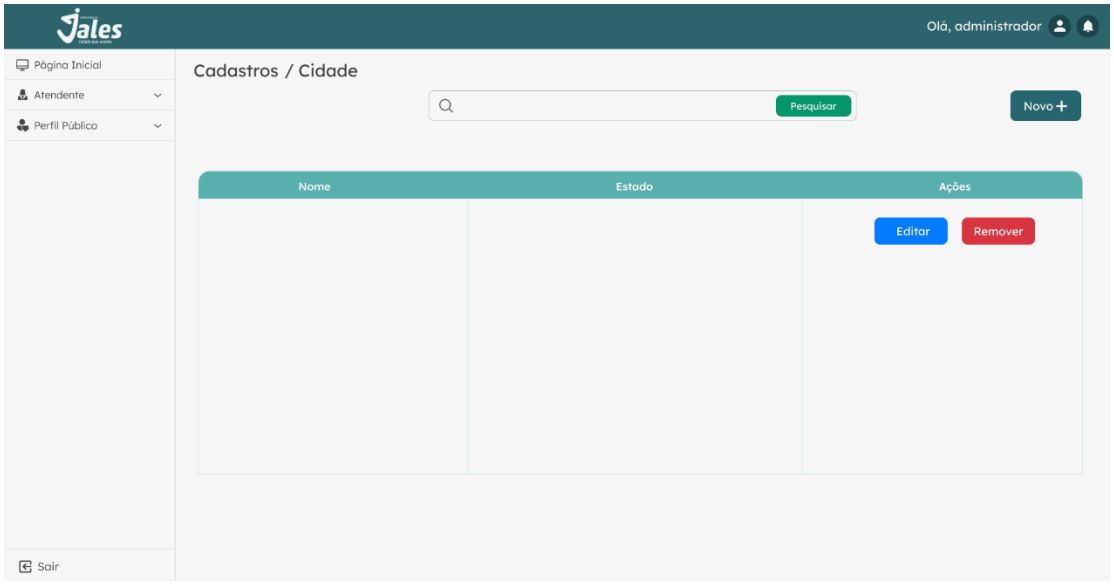
O protótipo da tela de cadastro de estado no sistema de gerenciamento de obras demonstra um design funcional e prático, priorizando a eficiência na gestão dos estados. A tabela no centro da tela apresenta uma lista completa de estados cadastrados, fornecendo uma visão geral rápida e organizada.

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.

Figura 24 – Protótipo da Tela de Cadastro de Cidade



Fonte: Elaborado pelos autores.

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.

Os botões de ação associados a cada entrada na tabela oferecem funcionalidades essenciais, permitindo aos usuários editarem ou excluírem 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) para escrever e consultar dados.

> Segundo o site Oracle “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), no livro Sistemas de Banco de Dados, 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

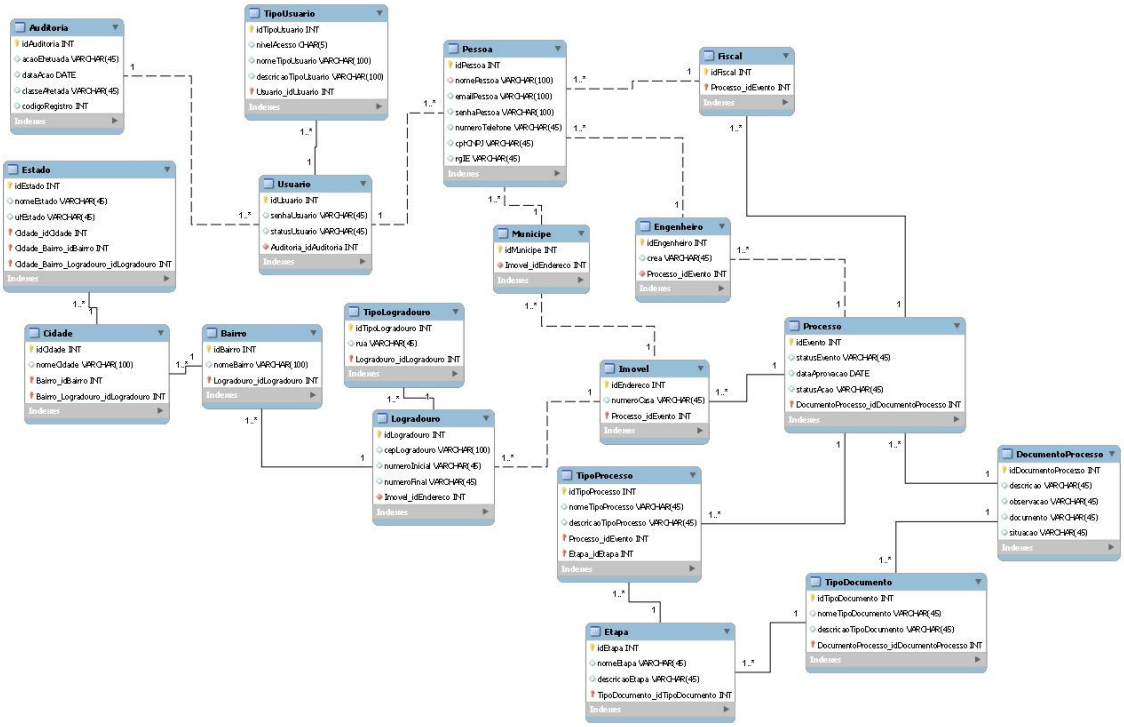
Silberschatz (2012) diz que, o Modelo Entidade-Relacionamento (MER) é um método muito importante para a modelagem de dados, possibilitando a representação gráfica da estrutura de um banco de dados. O MER apresenta conceitos como entidades, atributos e relações para detalhar as informações e interações 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 que as normas de negócio estabelecidas no código sejam espelhadas no banco de dados. Segundo Teorey (2011), esse mapeamento é crucial para preservar a consistência e integridade dos dados, facilitando a manutenção e progresso de sistemas complexos que empregam ambas as estratégias.

Na Figura 26 mostra um exemplo no contexto do projeto atual.

Figura 25 – Mapeamento do Objeto Relacional



Fonte: Elaborado pelos autores.

5.2 SCRIPT DAS TABELAS

Segundo o livro “Banco de dados: Projeto e Implementação”, os scripts são as instruções ou comandos escritos em uma linguagem de consulta estruturada (SQL). Elas são utilizadas para definir a estrutura de um banco de dados relacional.

A seguir, Quadro (25 - 48), serão apresentados os scripts do banco de dados, gerados automaticamente pelo Entity Framework, com base no contexto atual da aplicação. Esses scripts representam a estrutura das tabelas, relacionamentos e demais componentes necessários para o funcionamento correto do sistema, facilitando a integração entre a camada de aplicação e o banco de dados.

> "Segundo a Microsoft 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 26 – Tabela Bairro

```
CREATE TABLE Bairro (
    idbairro integer NOT NULL,
    nomebairro character varying(50) NOT NULL,
    idcidade integer NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 27 – Tabela Cidade

```
CREATE TABLE Cidade (
    idcidade integer NOT NULL,
    nomecidade character varying(100) NOT NULL,
    idestado integer NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 28 – Tabela Configuração

```
CREATE TABLE Configuracao (
    idconfiguracao integer NOT NULL,
    valorconfiguracao boolean NOT NULL,
    descricaoconfiguracao text NOT NULL,
    tipoconfiguracao text NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 29 – Tabela Documentoprocesso

```
CREATE TABLE Documentoprocesso (
    iddocumentoprocesso uuid NOT NULL,
    identificacaodocumento character varying(50) NOT NULL,
    descricaodocumento character varying(500) NOT NULL,
    observacaodocumento character varying(300) NOT NULL,
    arquivodocumento bytea NOT NULL,
    statusdocumentoprocesso integer NOT NULL,
    idprocesso uuid NOT NULL,
    idtipodocumentoetapa integer NOT NULL,
    idresponsavel integer,
    idaprovador integer
);
```

Fonte: Elaborado pelos autores.

Quadro 30 – 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
);
```

Fonte: Elaborado pelos autores.

Quadro 31 – 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,
    statusetapa integer NOT NULL,
    idtipoprocesso integer NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 32 – Tabela Fiscal

```
CREATE TABLE Fiscal (
    idfiscal 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
);
```

Fonte: Elaborado pelos autores.

Quadro 33 – Tabela Imovel

```
CREATE TABLE Imovel (
    idimovel integer NOT NULL,
    imagemimovel text,
    inscricaocadastral 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
);
```

Fonte: Elaborado pelos autores.

Quadro 34 – Tabela Infraestrutura

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

```
);
```

Fonte: Elaborado pelos autores.

Quadro 35 – Tabela Instalação

```
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
);
```

Fonte: Elaborado pelos autores.

Quadro 36 – Tabela Logradouro

```
CREATE TABLE Logradouro (
    idlogradouro integer NOT NULL,
    ceplogradouro character varying(9) NOT NULL,
    rualogradouro 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
);
```

Fonte: Elaborado pelos autores.

Quadro 37 – 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,
    cpfcnpjpessoa character varying(18) NOT NULL,
    rgiepessoa character varying(15) NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 38 – Tabela Ocupacaoatual

```
CREATE TABLE Ocupacaoatual (
    idocupacaoatual integer NOT NULL,
    nomeocupacaoatual character varying(50) NOT NULL,
    descricaoocupacaoatual text NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 39 – Tabela Processo

```
CREATE TABLE Processo (
    idprocesso uuid NOT NULL,
    identificacaoprocesso character varying(50) NOT NULL,
    descricaoprocesso 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
);
```

Fonte: Elaborado pelos autores.

Quadro 40 – 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
);
```

Fonte: Elaborado pelos autores.

Quadro 41 – Tabela Tipodocumento

```
CREATE TABLE Tipodocumento (
    idTipoDocumento integer NOT NULL,
    nomeTipoDocumento character varying(50) NOT NULL,
    descricaoTipoDocumento character varying(500) NOT NULL,
    statustipoprocesso integer NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 42 – Tabela Tipodocumentoetapa

```
CREATE TABLE Tipodocumentoetapa (
    idtipodocumentoetapa integer NOT NULL,
    posicaotipodocumentoetapa integer NOT NULL,
    statustipodocumentoetapa integer NOT NULL,
    idtipodocumento integer NOT NULL,
    idetapa integer NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 43 - Tipoinfraestrutura

```
CREATE TABLE Tipoinfraestrutura (
    idtipoinfraestrutura integer NOT NULL,
    nometipoinfraestrutura character varying(50) NOT NULL,
    descricaotipoinfraestrutura character varying(500) NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 44 - Tipologradouro

```
CREATE TABLE Tipologradouro (
    idtipologradouro integer NOT NULL,
    codigoinformativo character varying(3) NOT NULL,
    descricaotipologradouro character varying(35) NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 45 – Tabela Tipoprocesso

```
CREATE TABLE Tipoprocesso (
    idtipoprocesso integer NOT NULL,
    tipoprocesso character varying(50) NOT NULL,
    descricaotipoprocesso character varying(500) NOT NULL,
    statustipoprocesso integer NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 46 – Tabela Tipouso

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

Fonte: Elaborado pelos autores.

Quadro 47 – Tabela Tipousuario

```
CREATE TABLE Tipousuario (
    idtipousuario integer NOT NULL,
    nivelacesso character varying(1) NOT NULL,
    nometipousuario character varying(20) NOT NULL,
    descricaotipousuario character varying(300) NOT NULL
);
```

Fonte: Elaborado pelos autores.

Quadro 48 – Tabela Topografia

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

Fonte: Elaborado pelos autores.

Quadro 49 – 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
);
```

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, 2011).

Conforme demonstrado por Sommerville (2011), 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 RESful, 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).

A estrutura do código no *backend* é composta por vários elementos cruciais que colaboram para uma arquitetura robusta e eficiente. Os DTOs, ou *Data Transfer Objects*, são peças essenciais nesse contexto, desempenhando um papel vital na transferência eficiente de dados entre diferentes partes da aplicação. Essas estruturas encapsulam informações específicas, frequentemente representando entidades de negócios, otimizando a comunicação entre as camadas da aplicação.

Além dos DTOs, outros componentes fundamentais incluem *services*, *controllers* e *repositories*. Os *services* representam a camada de lógica de negócios, encapsulando operações e regras específicas do domínio, facilitando a modularização e organização do código. Já os *controllers* atuam como intermediários entre as requisições do usuário e os *services*, gerenciando o fluxo de dados e interações. Esses componentes recebem as requisições, acionam operações nos *services* correspondentes e respondem ao cliente com os resultados apropriados.

Quanto aos *repositories*, são responsáveis pela interação com o armazenamento de dados, isolando as operações de persistência e permitindo que o restante do sistema interaja com os dados sem se preocupar com os detalhes específicos do armazenamento. Também foram utilizadas APIs, como o Swagger, desempenhando um papel crucial na comunicação eficiente entre diferentes partes do sistema ou sistemas distintos. Essas APIs funcionam como pontes padronizadas para a exposição de funcionalidades e dados, permitindo integrações eficientes de serviços e facilitando a construção de aplicações modulares.

Essa combinação de DTOs, *services*, *controllers*, *repositories* e APIs forma uma arquitetura coesa que promove modularidade, clareza e escalabilidade no desenvolvimento de software.

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, 2024a), 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).

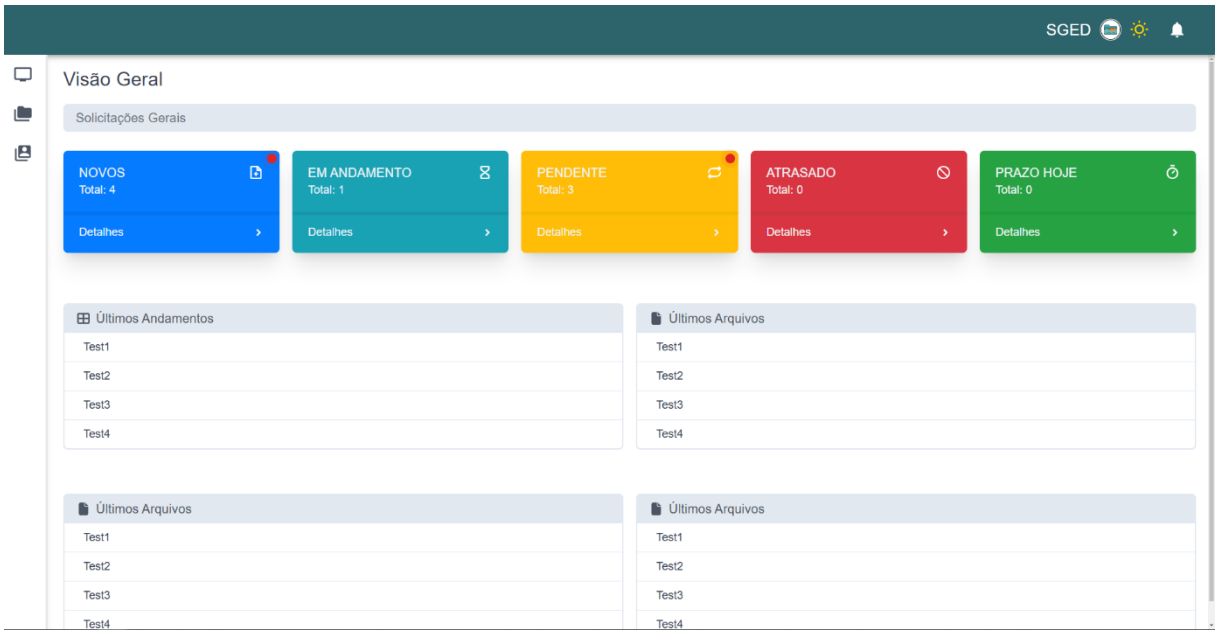
O projeto foi desenvolvido com o diretório **src** como raiz, agrupando arquivos e pastas conforme suas responsabilidades, visando proporcionar clareza ao código, mantendo uma hierarquia lógica e segmentada para atender às diferentes necessidades da aplicação. No núcleo da estrutura do software, existe 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**, observa-se 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 26 – 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 reaplicar credenciais a cada requisição. Isso reduz o risco de roubo de senhas e facilita a escalabilidade de sistemas.

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 front-end e o back-end 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 front-end e back-end, 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.

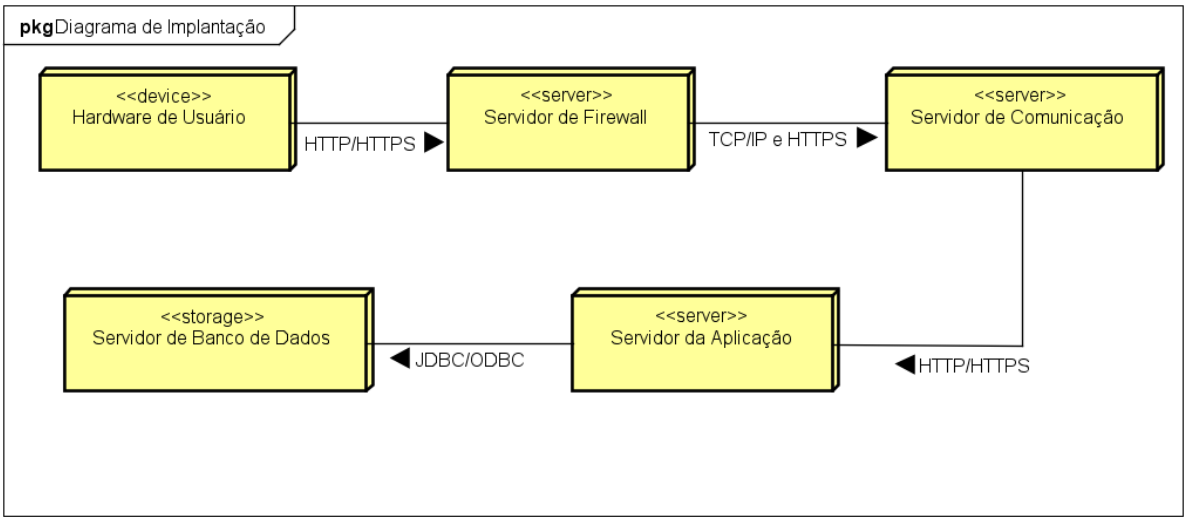
6.3 IMPLANTAÇÃO

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 ilustra a **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 27 – 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 *front-end* quanto no *back-end* 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

AMAZON Web Services. **O que é uma API RESTful? 2024**. Disponível em: https://aws.amazon.com/pt/what-is/restful-api/. Acesso em: 07 nov. 2024.

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%20importância%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 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.

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.

DAVIS, A. M. (1993). ***Software requirements: objects, functions, & states***. Englewood Cliffs, NJ: Prentice Hall.

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.

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, 2011.

GUEDES, Gilleanes T. A. **UML Uma Abordagem prática**, 3 ed. São Paulo: Novatec, 2008.

IBM. **O que é uma API?.** 2023. Disponível: <https://www.ibm.com/br-pt/topics/api>. Acesso: 03 dez. 2023.

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: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-sequencia-uml>. Acesso em: 07 jun. 2024.

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

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**.

MICROSOFT. "**Documentação do Entity Framework."** Disponível em: <https://learn.microsoft.com/pt-br/aspnet/entity-framework>. Acesso em: 3 out. 2024.

MICROSOFT**. Criar DTOs (objetos de transferência de dados).** 17 julho 2023. Disponível: <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: <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: <https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>. Acesso em: 03 dez. 2023.

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

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

ORACLE. **O que é um Banco de Dados?.** 2023. Disponível: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.

Prentice Hall, 2011.
PRESSMAN, R. S. (2016). **Engenharia de Software: Uma Abordagem Profissional** (8ª ed.). AMGH.

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.

SOMMERVILLE, Ian. **Engenharia de Software / Ian Sommerville** ; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama. — 9. ed. — São Paulo : Pearson
SOMMERVILLE, Ian. **Engenharia de software**. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

TEIXEIRA, F. Introdução e boas práticas em UX Design, Casa do Código. 2014.

TEOREY, Toby J. **Projeto e Modelagem de Banco de Dados**. 2. ed. São Paulo: Pearson, 2011.

TOTVS. **Sistema de gestão de documentos: benefícios para empresas. TOTVS. 2024**. Disponível em: <https://www.totvs.com/blog/gestao-para-assinatura-de-documentos/sistema-gestao-de-documentos/>. Acesso em: 29 out. 2024.

WATRALL, E; SIARTO, J. **Use A Cabeça! Web Design**. Alta Books, 2009.

OLIVEIRA, D. **MER e DER: Definições, Banco de Dados e Exemplos**. 18 setembro 2023. Disponível:<https://www.alura.com.br/artigos/mer-e-der-funcoes>. Acesso em: 21 nov. 2023.

WILLIAMS, J. ***Web Security for Developers***. 2. ed. O'Reilly Media, 2018.