

FUNDAÇÃO EDUCACIONAL DE FERNANDÓPOLIS – FEF

FACULDADES INTEGRADAS DE FERNANDÓPOLIS – FIFE

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

**Luis Mateus dos Reis Faria**

**DIGITALIZAÇÃO DE ATIVIDADES MANUAIS  
Sistematização de atividades manuais**

**Estágio Supervisionado**

Fernandópolis - SP

2024

**Luis Mateus dos Reis Faria**

**Matheus Henrique Garcia Queiroz**

**DIGITALIZAÇÃO DE TRABALHOS MANUAIS  
Sistematização de atividades manuais**

Projeto apresentado às Faculdade Integradas de Fernandópolis - FIFE, como requisito parcial para aprovação do estágio supervisionado do curso de bacharelado em Sistemas de Informação.

Fernandópolis - SP

2024

**Luis Mateus dos Reis Faria**

**DIGITALIZAÇÃO DE TRABALHOS MANUAIS  
Sistematização de atividades manuais**

Projeto apresentado às Faculdade Integradas de Fernandópolis - FIFE, como requisito parcial para aprovação do estágio supervisionado do curso de bacharelado em Sistemas de Informação.

**Banca Examinadora:**

Prof. (Orientador)  
Fundação Educacional de Fernandópolis - FEF

Prof.   
Fundação Educacional de Fernandópolis - FEF

Prof.   
Fundação Educacional de Fernandópolis - FEF

Fernandópolis - SP, \_\_\_ de \_\_\_\_\_\_\_\_\_\_ de 2024.

**AGRADECIMENTOS**

Agradecemos, primeiramente, .....

A todos, nossos agradecimentos profundos.

**RESUMO**

Este trabalho apresenta o desenvolvimento de um sistema web, voltado à digitalização e gerenciamento de processos manuais em pequenas e médias empresas. O sistema proposto visa substituir documentos em papel por formulários digitais inteligentes, permitindo o preenchimento estruturado, rastreável e centralizado de dados operacionais. Com foco em eficiência e padronização, a solução permite criar fluxos personalizados de documentos, facilitando a colaboração entre diferentes tipos de usuários (administradores, gestores e colaboradores), garantindo a integridade e versionamento das informações.

A arquitetura do sistema é composta por um front-end desenvolvido em React.js, com autenticação via Firebase e comunicação com um back-end em Java 17 por meio de APIs RESTful. O armazenamento dos dados é realizado em um banco de dados PostgreSQL, e os documentos são organizados em fluxos que podem conter múltiplas versões e status (como não preenchido, em andamento, finalizado ou alterado), otimizando a rastreabilidade e o controle de qualidade dos registros.

Cada documento digital pode conter múltiplos campos posicionáveis, possibilitando a visualização e preenchimento de informações de forma intuitiva, com potencial integração futura a sistemas de OCR e IoT para automação de entrada de dados. Como proposta de evolução, o sistema poderá se integrar a ferramentas de análise de dados (como BigQuery ou Power BI), viabilizando iniciativas de Business Intelligence e Big Data empresarial. O projeto propõe-se, assim, como uma base para a transformação digital em ambientes corporativos, promovendo produtividade, segurança da informação e decisões orientadas por dados.

**Palavras-chave:** Digitalização; Atividades Manuais; Otimização; Redução de Custos;

Sistema Web; Desenvolvimento de Software.

**ABSTRACT**

This paper presents the development of a web system aimed at digitizing and managing manual processes in small and medium-sized businesses. The proposed system replaces paper documents with intelligent digital forms, enabling structured, traceable, and centralized data entry. With a focus on efficiency and standardization, the solution allows the creation of customized document flows, facilitating collaboration among different user roles (administrators, managers, and collaborators) while ensuring data integrity and version control.

The system architecture consists of a React.js-based frontend with Firebase authentication and a backend developed in Java 17, communicating via RESTful APIs. Data storage is handled by a PostgreSQL database, and documents are organized into flows that support multiple versions and statuses (such as not filled, in progress, completed, or modified), improving traceability and quality control.

Each digital document can contain multiple positionable fields, allowing for intuitive data entry with potential future integration of OCR and IoT systems to automate data capture. As a proposed enhancement, the system may integrate with data analysis tools (such as BigQuery or Power BI), enabling Business Intelligence and Big Data initiatives. Thus, the project serves as a foundation for digital transformation in corporate environments, fostering productivity, information security, and data-driven decision-making.

**Keywords:** Public Expenses; Municipality; Financial Management; Web System; Software Development.

Lista de Figuras

[Figura 1 – Controle de Despesas – Prefeitura de Jales 15](#_Toc183761528)

[Figura 2 – Software Controlle 16](#_Toc183761529)

[Figura 3 – Sistema Soma 17](#_Toc183761530)

[Figura 4 – Sistema Meu Dinheiro 18](#_Toc183761531)

[Figura 5 – Software Desktop Expenses Controller 18](#_Toc183761532)

[Figura 6 – Sistema SAGEF 19](#_Toc183761533)

[Figura 7 – Aplicativo EcoCalc 20](#_Toc183761534)

[Figura 8 – Diagrama de Classe 27](#_Toc183761535)

[Figura 9 – Atores do Sistema 36](#_Toc183761536)

[Figura 10 – Diagrama de Caso de Uso Geral - Visão Administrador 46](#_Toc183761537)

[Figura 11 – Diagrama de Caso de Uso Geral – Visão Funcionário 47](#_Toc183761538)

[Figura 12 – Diagrama de Caso de Uso Geral - Visão Visitante 48](#_Toc183761539)

[Figura 13 – Diagrama de Caso de Uso Individual – Administrador Lista Despesa 49](#_Toc183761540)

[Figura 14 – Diagrama de Caso de Uso Individual – Administrador Lista Instituição 54](#_Toc183761541)

[Figura 15 – Diagrama de Caso de Uso Individual – Administrador Lista Secretaria 60](#_Toc183761542)

[Figura 16 – Diagrama de Sequência – Administrador Cadastra Despesa 67](#_Toc183761543)

[Figura 17 – Diagrama de Sequência – Administrador Altera Despesa 68](#_Toc183761544)

[Figura 17 – Diagrama de Sequência – Administrador Altera Despesa (continuação) 69](#_Toc183761545)

[Figura 18 – Diagrama de Sequência - Administrador Exclui Despesa 70](#_Toc183761546)

[Figura 18 – Diagrama de Sequência - Administrador Exclui Despesa (continuação) 71](#_Toc183761547)

[Figura 19 – Diagrama de Atividade – Cadastro de Despesa 72](#_Toc183761548)

[Figura 20 – Diagrama de Máquina de Estado – Componente Situação 73](#_Toc183761549)

[Figura 21 – Persona 1 76](#_Toc183761550)

[Figura 22 – Persona 2 77](#_Toc183761551)

[Figura 23 – Wireframe - Tela Home 78](#_Toc183761552)

[Figura 24 – Wireframe – Tela Cadastro de Despesa 79](#_Toc183761553)

[Figura 25 – Wireframe - Tela Cadastro de Secretaria 80](#_Toc183761554)

[Figura 26 – Wireframe - Tela Cadastro de Instituição 80](#_Toc183761555)

[Figura 27 – Protótipo Tela – Home 82](#_Toc183761556)

[Figura 28 – Protótipo Tela – Cadastro de Despesa 82](#_Toc183761557)

[Figura 29 – Protótipo Tela – Cadastro de Secretaria 83](#_Toc183761558)

[Figura 30 – Protótipo Tela – Cadastro de Instituição 84](#_Toc183761559)

[Figura 31 – Mapeamento do Objeto Relacional 86](#_Toc183761560)

[Figura 32 – Diagrama de Implantação 100](#_Toc183761561)

Lista de Quadros

[Quadro 1 - Requisitos Funcionais 21](#_Toc183425064)

[Quadro 2 - Requisitos Não Funcionais 23](#_Toc183425065)

[Quadro 3 - Descrição Classe Despesa 27](#_Toc183425066)

[Quadro 4 - Descrição Classe Secretaria 28](#_Toc183425067)

[Quadro 5 - Descrição Classe Instituição 29](#_Toc183425068)

[Quadro 6 - Descrição Classe Fornecedor 30](#_Toc183425069)

[Quadro 7 - Descrição Classe Orçamento 30](#_Toc183425070)

[Quadro 8 - Descrição Classe Unidade Medida 31](#_Toc183425071)

[Quadro 9 - Descrição Classe Unidade Consumidora 31](#_Toc183425072)

[Quadro 10 - Descrição Classe Tipo Instituição 32](#_Toc183425073)

[Quadro 11 - Descrição Classe TipoDespesa 32](#_Toc183425074)

[Quadro 12 - Descrição Classe Usuário 32](#_Toc183425075)

[Quadro 13 - Descrição Classe Auditoria 33](#_Toc183425076)

[Quadro 14 - Descrição Classe Tipo Usuário - Enumeração dos tipos de usuário no sistema 34](#_Toc183425077)

[Quadro 15 - Descrição Classe Situação - Enumeração de Situação no sistema 34](#_Toc183425078)

[Quadro 16 - Descrição Classe Solicita UC - Enumeração de Solicita UC no sistema 34](#_Toc183425079)

[Quadro 17 - Descrição Classe Tipo Usuário - Enumeração dos status de uma Despesa 35](#_Toc183425080)

[Quadro 18 - Lista de Mensagens 38](#_Toc183425081)

[Quadro 19 - Lista de Casos de Uso 38](#_Toc183425082)

[Quadro 20 - Documentação - Administrador Lista Despesa 50](#_Toc183425083)

[Quadro 21 - Documentação - Administrador Cadastra Despesa 51](#_Toc183425084)

[Quadro 22 - Documentação - Administrador Altera Tipo Despesa 52](#_Toc183425085)

[Quadro 23 - Documentação - Administrador Exclui Tipo Despesa 53](#_Toc183425086)

[Quadro 24 - Documentação - Administrador Lista Instituição 55](#_Toc183425087)

[Quadro 25 - Documentação - Administrador Cadastra Instituição 56](#_Toc183425088)

[Quadro 26 - Documentação - Administrador Altera Instituição 57](#_Toc183425089)

[Quadro 27 - Documentação - Administrador Exclui Instituição 58](#_Toc183425090)

[Quadro 28 - Documentação - Funcionário Lista Secretaria 60](#_Toc183425091)

[Quadro 29 - Documentação - Funcionário Cadastra Secretaria 62](#_Toc183425092)

[Quadro 30 - Documentação - Funcionário Altera Secretaria 63](#_Toc183425093)

[Quadro 31 - Documentação - Funcionário Exclui Secretaria 64](#_Toc183425094)

[Quadro 32 – Cenário - Registro de Despesas por um Funcionário da Prefeitura 75](#_Toc183425095)

[Quadro 33 – Cenário - Análise de Dashboards pelo Funcionário da Secretaria da Fazenda 75](#_Toc183425096)

[Quadro 34 - Script SQL – Tabela Despesa 87](#_Toc183425097)

[Quadro 35 - Script SQL – Tabela Secretaria 88](#_Toc183425098)

[Quadro 36 - Script SQL – Tabela Instituição 89](#_Toc183425099)

[Quadro 37 - Script SQL – Tabela Fornecedor 89](#_Toc183425100)

[Quadro 38 - Script SQL – Tabela Orçamento 90](#_Toc183425101)

[Quadro 39 - Script SQL – Tabela Usuário 90](#_Toc183425102)

[Quadro 40 - Script SQL – Tabela Auditoria 91](#_Toc183425103)

[Quadro 41 - Script SQL – Tabela Tipo Despesa 91](#_Toc183425104)

[Quadro 42 - Script SQL – Tabela Tipo Instituição 91](#_Toc183425105)

[Quadro 43 - Script SQL – Tabela Unidade Medida 92](#_Toc183425106)

[Quadro 44 - Script SQL – Tabela Unidade Consumidora 92](#_Toc183425107)

[Quadro 45 – Softwares Utilizados 94](#_Toc183425108)

SUMÁRIO

[1 INTRODUÇÃO 12](#_Toc182237761)

[2 LEVANTAMENTO DE REQUISITOS DE SOFTWARE 14](#_Toc182237762)

[2.1 Descrição dos objetivos do sistema 14](#_Toc182237763)

[2.2 Descrição do sistema atual 14](#_Toc182237764)

[2.3 Análise de Sistemas Existentes 16](#_Toc182237765)

[2.4 Descrição dos Principais Problemas 20](#_Toc182237766)

[2.5 Descrição dos Requisitos Funcionais 21](#_Toc182237767)

[2.6 Descrição dos Requisitos Não Funcionais 23](#_Toc182237768)

[3 VISÃO DE CASO DE USO – UML 25](#_Toc182237769)

[3.1 Diagrama de Classes 25](#_Toc182237770)

[3.2 Dicionário de Classes 27](#_Toc182237771)

[3.3 Definição dos Atores 35](#_Toc182237772)

[3.4 Lista de Casos de Uso 37](#_Toc182237773)

[3.5 Diagrama de Casos de Uso 44](#_Toc182237774)

[3.6 Diagrama de Casos de Uso Individuais 48](#_Toc182237775)

[3.6.1 Diagrama de Casos de Uso Individuais – Administrador Lista Despesa 49](#_Toc182237776)

[3.6.2 Diagrama de Casos de Uso Individuais – Administrador Lista Orçamento 54](#_Toc182237777)

[3.6.3 Diagrama de Casos de Uso Individuais – Funcionário Lista Secretaria 59](#_Toc182237778)

[3.7 Diagrama de Sequência 65](#_Toc182237779)

[3.8 Diagrama de Atividade 72](#_Toc182237780)

[3.9 Diagrama de Máquina de Estados 73](#_Toc182237781)

[4 DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX) 74](#_Toc182237782)

[4.1 Descrição de Cenário 74](#_Toc182237783)

[4.2 Descrição de Personas 75](#_Toc182237784)

[4.3 Esboços de tela (*wireframes*) 77](#_Toc182237785)

[4.4 Protótipos de Tela 81](#_Toc182237786)

[5 BANCO DE DADOS 85](#_Toc182237787)

[5.1 Modelo Entidade Relacionamento 85](#_Toc182237788)

[5.2 Script das tabelas 87](#_Toc182237789)

[5.3 Mapeamento Objeto Relacional – ORM 92](#_Toc182237790)

[6 ARQUITETURA DE SOFTWARE 93](#_Toc182237791)

[6.1 Arquitetura de desenvolvimento 93](#_Toc182237792)

[6.1.1 Back-end 95](#_Toc182237793)

[6.1.2 Front-end 96](#_Toc182237794)

[6.2 Segurança da Informação 97](#_Toc182237795)

[6.3 Implantação do Sistema 99](#_Toc182237796)

[7 CONCLUSÃO 103](#_Toc182237797)

[8 REFERÊNCIAS 104](#_Toc182237798)

# INTRODUÇÃO

No cenário atual, pequenas e médias empresas (PMEs) enfrentam desafios significativos decorrentes da execução de processos manuais apoiados em documentos físicos e fluxos de trabalho pouco estruturados. Procedimentos como ordens de produção, check-lists de inspeção de qualidade, requisições de compra e relatórios de turno são frequentemente realizados em papel, o que ocasiona atrasos operacionais, retrabalho e dificuldades de auditoria, comprometendo a eficiência e a confiabilidade das informações.

A transformação digital desses processos revela-se imprescindível para as organizações que almejam manter-se competitivas. Ao promover a substituição de formulários impressos por fluxos de documentos eletrônicos inteligentes, é possível centralizar dados, assegurar a integridade e o versionamento das informações, reduzir a ocorrência de erros humanos e agilizar a circulação de registros entre administradores, gestores e colaboradores. Ademais, a consolidação de um repositório estruturado de documentos favorece, a médio e longo prazo, a implementação de iniciativas de Business Intelligence e Big Data, oferecendo subsídios para a análise preditiva de indicadores-chave, como produtividade, falhas operacionais e custos.

Estudos recentes reforçam a importância da digitalização para o aumento da eficiência operacional. Segundo a Confederação Nacional da Indústria (CNI), sete em cada dez empresas brasileiras já utilizam ferramentas digitais em seus processos produtivos. Um estudo da McKinsey estima que, até 2025, os processos relacionados à Indústria 4.0 poderão aumentar a eficiência do trabalho entre 10% e 25%. Além disso, a automação documental pode reduzir os custos operacionais em até 40% e aumentar a produtividade em 50%, tornando-se, portanto, uma solução essencial para modernizar a gestão empresarial.

Diante desse panorama, o presente trabalho propõe o desenvolvimento de um sistema web voltado à digitalização e ao gerenciamento de processos manuais em pequenas e médias empresas. A solução será implementada com tecnologias modernas, utilizando Java 17 para o back-end, React.js para a interface front-end, PostgreSQL como sistema de gerenciamento de banco de dados e Firebase para autenticação de usuários. O sistema permitirá a criação de formulários digitais configuráveis e rastreáveis, organizados em fluxos de documentos com controle de status e versões, acessíveis conforme os níveis de permissão de usuários categorizados como administradores, gestores ou colaboradores.

A comunicação entre cliente e servidor será estruturada por meio de APIs RESTful, com possibilidade de integração futura a sistemas de reconhecimento óptico de caracteres (OCR), dispositivos de Internet das Coisas (IoT) e plataformas de análise de dados como Power BI. Espera-se, com este trabalho, demonstrar que a digitalização de processos manuais em empresas não apenas automatiza atividades rotineiras, mas constitui um pilar estratégico para a segurança da informação, a produtividade e a tomada de decisão baseada em dados.

# LEVANTAMENTO DE REQUISITOS DE SOFTWARE

## Descrição dos objetivos do sistema

O principal objetivo dos Sistemas Manuais Digitais (SMD) é fornecer às pequenas e médias empresas (PMEs) uma plataforma robusta e acessível que permita substituir formulários manuais, como checklists operacionais, relatórios de inspeção, ordens de serviço e demais documentos impressos, por fluxos digitais estruturados e rastreáveis.

A proposta central do sistema é aumentar a eficiência dos processos internos, garantir a integridade e o versionamento de dados operacionais, e proporcionar maior controle e visibilidade sobre as atividades realizadas em diferentes setores das empresas. Por meio da digitalização, pretende-se reduzir significativamente o tempo gasto com preenchimento, armazenamento e consulta de documentos, além de minimizar erros humanos e falhas de comunicação.

O SMD oferecerá uma solução integrada que conecta setores como produção, manutenção, qualidade e administração, promovendo a automatização dos registros e facilitando o acesso às informações em tempo real. O sistema também contará com trilhas de auditoria, autenticação de usuários, geração de relatórios customizáveis e notificações automáticas, permitindo que gestores acompanhem as operações com precisão e em conformidade com normas internas e exigências legais.

Assim, o sistema não apenas moderniza os fluxos documentais, mas também contribui para a transformação digital das PMEs, promovendo maior competitividade, redução de custos operacionais e adoção de boas práticas organizacionais.

## Descrição do sistema atual

A Prefeitura Municipal de Jales utiliza o Microsoft Excel como ferramenta principal para o registro e gestão de contas a pagar e pendentes. Este sistema baseia-se em uma planilha organiza as despesas em três categorias principais: energia, telefone e água, proporcionando uma visão detalhada dos custos em cada área.

Para cada uma dessas categorias, são apresentados diversos dados relevantes. Primeiro, a unidade consumidora é identificada, permitindo a associação dos gastos com um ponto específico de consumo. Em seguida, são listados os endereços correspondentes a cada unidade, facilitando a localização geográfica dos custos. Além disso, a tabela inclui informações sobre a instituição e a secretaria responsáveis, estabelecendo uma conexão entre as despesas e os departamentos envolvidos. Outro dado essencial apresentado é o período de data, que define o intervalo de tempo ao qual os valores se referem, permitindo uma análise temporal das despesas.

A planilha (Figura 1) está organizada de forma a separar as informações por secretarias, cada uma contendo suas respectivas instituições. Para cada instituição, é listado o código UC (Unidade Consumidora), que identifica de forma única a unidade em questão. Além disso, há uma coluna para cada mês, onde os valores referentes a despesa de cada instituição são inseridos, permitindo um acompanhamento detalhado e mensal dos gastos.

Atualmente, essa planilha (Figura 1) passa por frequentes atualizações diárias, o que torna a tarefa de controle ainda mais desafiadora. Isso cria dificuldades para os responsáveis pelo acompanhamento dessas informações, uma vez que não é fácil determinar o status de pagamento de cada despesa.

Figura – Controle de Despesas – Prefeitura de Jales

Tabela

Descrição gerada automaticamente

Fonte: Elaborada pelos autores.

## Análise de Sistemas Existentes

A digitalização de processos manuais é uma necessidade crescente entre pequenas e médias empresas (PMEs) que buscam eficiência operacional, redução de custos e conformidade

com normas regulatórias. Diversas soluções de software estão disponíveis no mercado para atender a essa demanda.

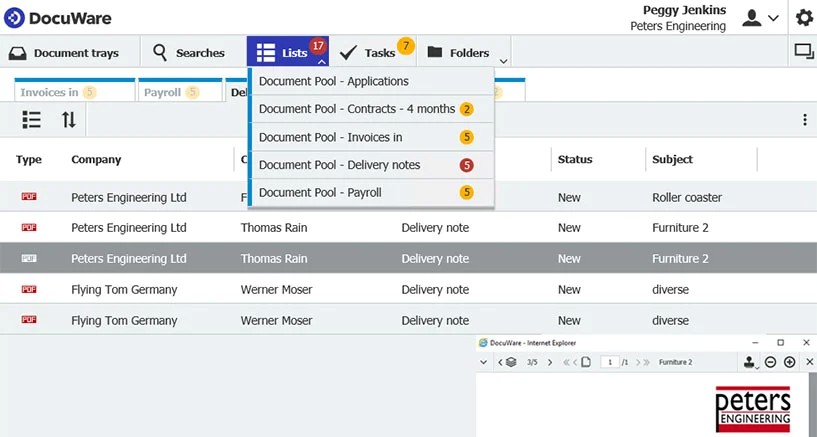
Atualmente, diversas soluções tecnológicas disponíveis no mercado têm como objetivo otimizar a gestão de documentos e a digitalização de processos operacionais em pequenas e médias empresas. Esses sistemas foram desenvolvidos com o propósito de substituir fluxos baseados em papel, promover a automação de tarefas repetitivas e assegurar o armazenamento seguro e acessível de informações.

Entre os principais benefícios promovidos por essas soluções estão o aumento da eficiência, a padronização de procedimentos e a possibilidade de rastreamento detalhado das atividades, aspectos essenciais para setores como produção, manutenção, qualidade e administração. Contudo, é importante observar que muitas dessas plataformas apresentam um nível de complexidade ou um custo que nem sempre se adapta à realidade das PMEs.

A seguir, são apresentados alguns dos sistemas mais relevantes utilizados para gerenciamento de documentos e automação de fluxos em empresas, com foco em funcionalidades que se aproximam da proposta dos Sistemas Manuais Digitais (SMD).

O DocuWare (Figura 2) é uma plataforma de gestão de documentos que oferece recursos de digitalização, automação de fluxos de trabalho e armazenamento seguro na nuvem. É voltado para empresas que desejam eliminar processos baseados em papel e melhorar a colaboração entre equipes.

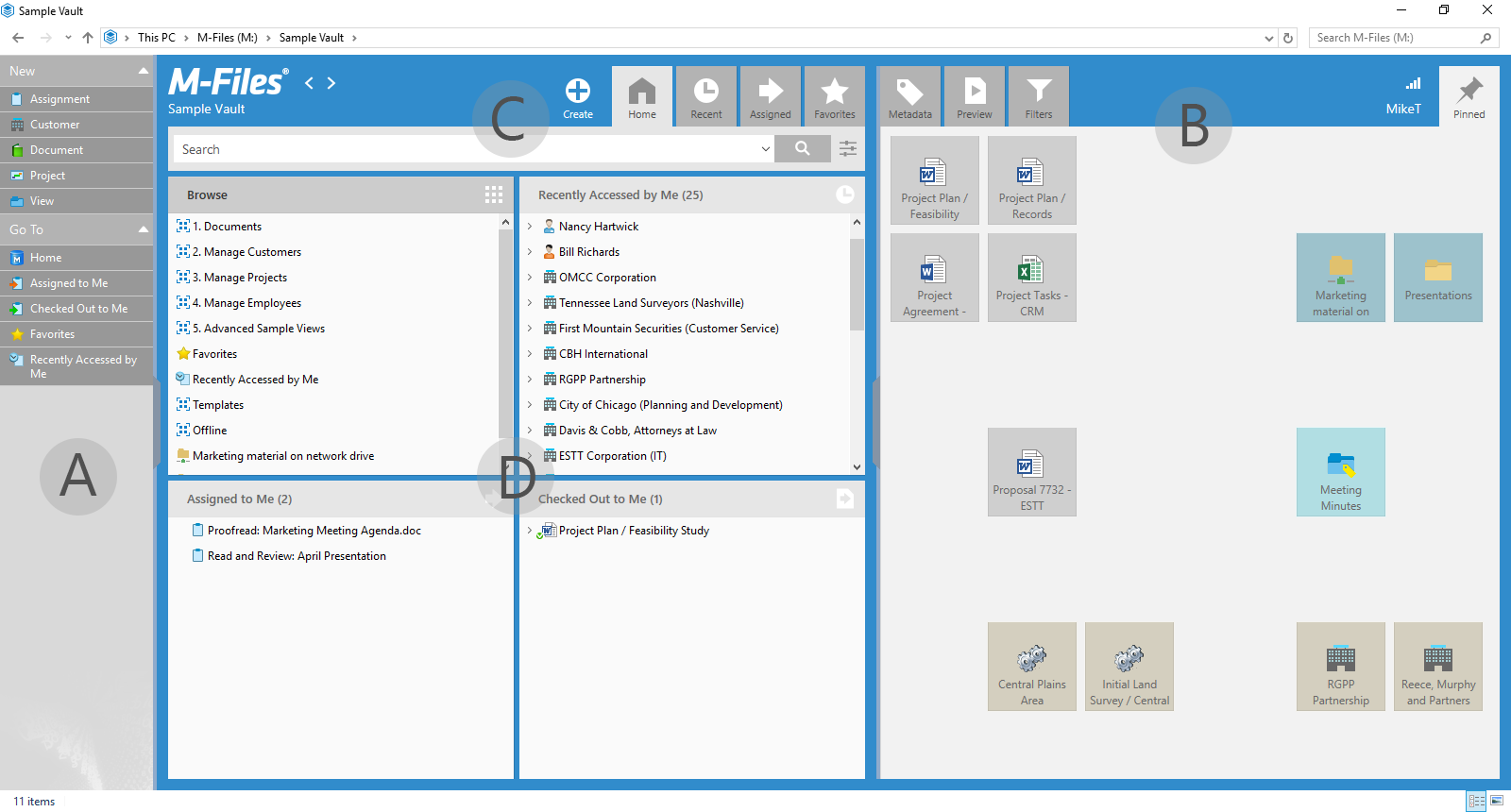
Figura – DocuWare



Fonte: DocuWare, 2025.

O M-Files (Figura 3) utiliza inteligência artificial para organizar e gerenciar documentos com base em seu conteúdo, em vez de depender de estruturas de pastas tradicionais. Isso facilita a localização e o controle de versões de documentos, sendo útil para PMEs que lidam com grandes volumes de informações.

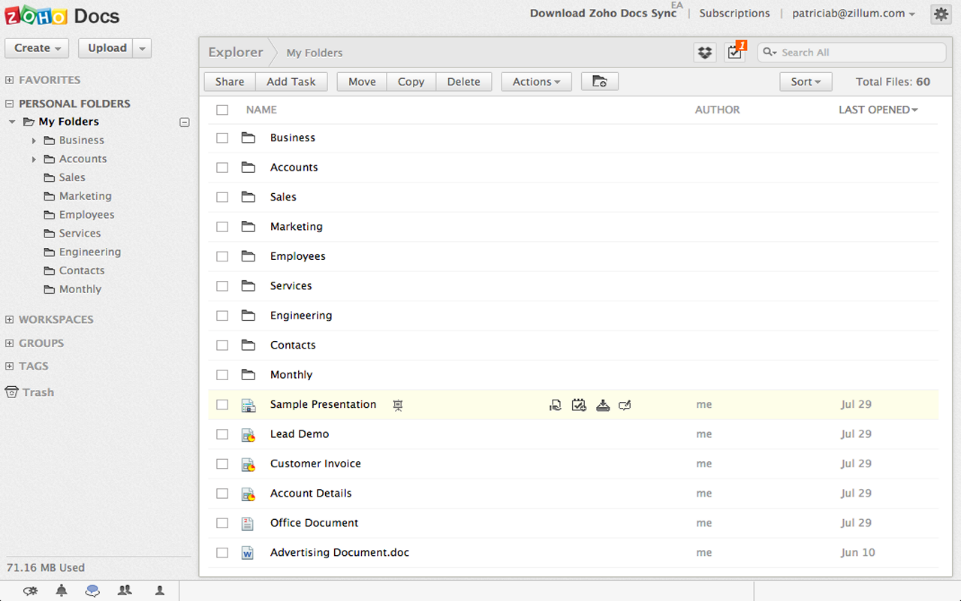
Figura – M-Flies



Fonte: M-Files, 2025.

o Zoho Docs (Figura 4) oferece armazenamento em nuvem, colaboração em tempo real e gerenciamento de documentos. É uma opção acessível para PMEs que buscam uma solução integrada para documentos e comunicação interna.

**Figura 4 –** Zoho Docs

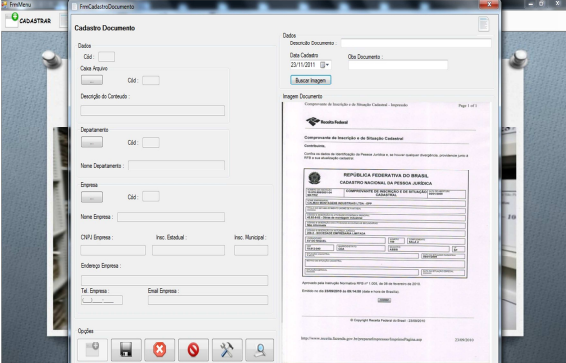


Fonte: Zoho Docs, 2025.

Além dos softwares pesquisados, também foram analisados dois Trabalhos de Conclusão de Curso (TCCs) desenvolvidos por estudantes de instituições brasileiras, cujos temas se relacionam diretamente com o conceito de controle de despesas, digitalização de processos e gestão financeira em pequenas e médias empresas. A análise desses estudos complementa a compreensão das necessidades práticas enfrentadas por empresas de menor porte e reforça a relevância da proposta dos Sistemas Manuais Digitais (SMD).

O primeiro artigo analisado possui como tema “Sistema de Gerenciamento Eletrônico de Documentos” (Figura 5), que, de acordo com Oliveira (2011), apresenta uma solução simples para digitalizar e organizar documentos em empresas, substituindo arquivos físicos por versões digitais.

**Figura 5 –** Sistema de Gerenciamento Eletrônico de Documentos



Fonte: Bruno Miranda de Oliveira, 2011, p.55.

O segundo artigo analisado possui como tema “GED – Gerenciamento Eletrônico de Documentos: Dificuldades na Implantação do GED nas Empresas Médias e Pequenas” (Figura 6), que, segundo Anjos (2020), identifica barreiras técnicas e culturais enfrentadas por pequenas empresas ao tentar adotar soluções digitais.

**Figura 6 –** GED – Gerenciamento Eletrônico de Documentos: Dificuldades na Implantação do GED nas Empresas Médias e Pequenas



Fonte: Sistema Interno -Fade-UFPE

## Descrição dos Principais Problemas

Foram identificados diversos problemas no modelo atual de gerenciamento de documentos operacionais em pequenas e médias empresas. Entre eles, destaca-se a dificuldade de registrar, consultar e monitorar informações de forma eficiente, devido ao uso predominante de formulários físicos e planilhas manuais. Esse cenário resulta em falhas de controle, perda de informações e baixa rastreabilidade, impactando diretamente a tomada de decisões e o desempenho das rotinas internas.

Uma das causas desses problemas está relacionada à participação de diferentes setores no preenchimento e processamento dos documentos. Essa fragmentação gera uma divisão desordenada de responsabilidades, em que cada área tem acesso apenas a partes isoladas dos dados. Como consequência, as informações se tornam incompletas, dificultando o entendimento do fluxo completo dos processos, o que compromete a padronização, a comunicação e a eficiência geral das atividades.

Diante disso, é evidente a necessidade de revisão dos métodos atualmente utilizados para controle de dados operacionais. O projeto SMD propõe-se a solucionar esses entraves por meio da digitalização e integração dos formulários manuais, oferecendo uma plataforma unificada, segura e acessível, capaz de melhorar significativamente a gestão das informações internas das PMEs.

## Descrição dos Requisitos Funcionais

Sommerville (2018) explica que requisitos funcionais são as especificações gerais sobre o que um sistema deve realizar. Esses requisitos “devem descrever em detalhes as funções do sistema, suas entradas, saídas e exceções” (Sommerville, 2018, p.89).

Os requisitos podem ser divididos em duas categorias principais. Requisitos para usuários, que são direcionados aos usuários finais e gerentes do sistema, descrevendo as funcionalidades do sistema sob a ótica do usuário. Requisitos do sistema, destinados aos desenvolvedores, detalham as funcionalidades do sistema do ponto de vista técnico (Sommerville, 2018Embora os requisitos funcionais geralmente descrevam o que o sistema Embora os requisitos funcionais geralmente descrevam o que o sistema deve fazer, “em alguns casos, [...] também podem declarar explicitamente o que o sistema não deve fazer” (Sommerville, 2018, p.89).

No contexto do Sistema de Manuais Digitais (SMD), que visa a digitalização e automação de documentos manuais como checklists, formulários e ordens de serviço em pequenas e médias empresas, foram identificados os seguintes requisitos funcionais, conforme demonstrado no Quadro 1.

Quadro 1 - Requisitos Funcionais

|  |  |  |
| --- | --- | --- |
|  | **Requisitos Funcionais** | **Descrição** |
| **1** | Autenticação de Usuário | O sistema deve permitir que usuários se autentiquem utilizando e-mail e senha, garantindo acesso seguro às funcionalidades. |
| **2** | Cadastro de Usuários | O sistema deve possibilitar o registro de novos usuários, armazenando informações como nome, e-mail, senha e tipo de usuário (Administrador, Gestor ou Colaborador). |
| **3** | Gerenciamento de Perfis de Acesso | O sistema deve permitir a definição e gerenciamento de diferentes níveis de acesso, controlando as permissões de acordo com o tipo de usuário. |
| **4** | Criação de Documentos Digitais | O sistema deve permitir a criação de documentos digitais, como checklists e formulários, associando-os a fluxos de trabalho específicos. |
| **5** | Edição de Documentos | O sistema deve possibilitar a edição de documentos existentes, permitindo atualizações em seu conteúdo e estrutura. |
| **6** | Exclusão de Documentos | O sistema deve permitir a exclusão de documentos, removendo-os permanentemente do sistema quando necessário. |
| **7** | Criação de Campos Interativos | O sistema deve permitir a adição de campos interativos nos documentos, especificando propriedades como nome, conteúdo, posição (coordenadas x e y) e página. |
| **8** | Edição de Campos | O sistema deve possibilitar a edição de campos existentes nos documentos, permitindo alterações em suas propriedades. |
| **9** | Exclusão de Campos | O sistema deve permitir a remoção de campos dos documentos, excluindo-os permanentemente quando necessário. |
| **10** | Criação de Fluxos de Documentos | O sistema deve permitir a criação de fluxos de documentos, agrupando múltiplos documentos sob uma ordem de serviço específica. |
| **11** | Edição de Fluxos | O sistema deve possibilitar a edição de fluxos existentes, permitindo atualizações em seus documentos associados e propriedades. |
| **12** | Exclusão de Fluxos | O sistema deve permitir a exclusão de fluxos de documentos, removendo-os permanentemente do sistema quando necessário. |
| **113** | Atribuição de Documentos a Fluxos | O sistema deve permitir a associação de documentos específicos a fluxos de trabalho, organizando-os de forma estruturada. |
| **114** | Preenchimento de Campos por Colaboradores | O sistema deve permitir que colaboradores preencham os campos designados nos documentos atribuídos a eles. |
| **115** | Acompanhamentos de Fluxos por Gestores | O sistema deve permitir que gestores acompanhem o andamento dos fluxos de documentos, visualizando o status de cada documento em tempo real. |
| **116** | Controle de Versões de Documentos | O sistema deve manter um histórico de versões dos documentos, permitindo o rastreamento de alterações ao longo do tempo. |
| **117** | Registro de Atividades | O sistema deve registrar as atividades dos usuários, como logins, edições e preenchimentos de campos, para fins de auditoria. |
| **118** | Notificações Internas | O sistema deve enviar notificações internas aos usuários sobre eventos relevantes, como a atribuição de novos documentos ou atualizações em fluxos. |
| **119** | Exportação de Documentos | O sistema deve permitir a exportação de documentos preenchidos em formatos como PDF, facilitando o compartilhamento e arquivamento. |
| **220** | Pesquisa de Documentos | O sistema deve oferecer funcionalidades de busca, permitindo que usuários localizem documentos específicos com base em critérios como nome, status ou data. |

Fonte: Elaborada pelos autores.

## Descrição dos Requisitos Não Funcionais

Além dos requisitos funcionais, Sommerville (2018) também aborda os requisitos não funcionais do sistema. Esses requisitos abrangem aspectos gerais do sistema, em vez de se concentrarem em serviços e características individuais como os requisitos funcionais.

Alguns exemplos de requisitos não funcionais são “restrições de tempo, restrições sobre o processo de desenvolvimento e restrições impostas por padrões” (Sommerviller, 2018, p. 89).

Os requisitos não funcionais surgem das necessidades dos usuários, que se devem a restrições orçamentárias, políticas organizacionais, necessidade de interoperabilidade com outros sistemas de software ou hardware, ou fatores externos, como normas de segurança (safety) ou legislação relativa à privacidade (Sommerville, 2018, p. 91).

No contexto do Sistema de Manuais Digitais (SMD), os requisitos não funcionais foram identificados com base nas necessidades das pequenas e médias empresas atendidas pelo sistema. A seguir, são apresentados os principais requisitos não funcionais, conforme o Quadro 2.

Quadro - Requisitos Não Funcionais

|  |  |
| --- | --- |
| **Requisitos Não Funcionais** | **Descrição** |
| Desempenho do Sistema | O sistema deve ser capaz de processar requisições de leitura e escrita em até 2 segundos. |
| Segurança de Acesso | O acesso ao sistema deve ser autenticado por meio de Firebase Authentication, garantindo proteção dos dados. |
| Escalabilidade | A arquitetura do sistema deve permitir a inclusão de novos fluxos, documentos e usuários sem prejuízo ao desempenho. |
| Disponibilidade | O sistema deve estar disponível 24 horas por dia, 7 dias por semana, com tempo de inatividade inferior a 1%. |
| Usabilidade | A interface do sistema deve ser intuitiva, com fluxos claros e instruções acessíveis ao usuário. |
| Backup e Recuperação | O sistema deve realizar backups periódicos do banco de dados e permitir recuperação em caso de falha. |
| Portabilidade | O sistema deve ser responsivo e acessível tanto em desktops quanto em dispositivos móveis. |
| Compatibilidade de Navegador | O sistema deve ser compatível com os navegadores modernos (Chrome, Firefox, Edge). |
| Registro de Erros | Todas as falhas e exceções devem ser registradas em logs de erro para posterior análise pela equipe técnica. |

Fonte: Elaborada pelos autores.

# VISÃO DE CASO DE USO – UML

O entendimento profundo de um sistema é crucial para o sucesso de qualquer projeto de desenvolvimento de software. Nesse contexto, a Linguagem de Modelagem Unificada (*UML*) emerge como uma ferramenta essencial para representar visualmente a estrutura e as interações entre os componentes do sistema. De acordo com Miro (2024), “desenvolvedores criam diagramas UML para entender projetos, arquitetura de código e propostas de implementação de sistemas de software complexos.”

A análise a seguir se dedica à exploração dos principais diagramas da *UML* aplicados no desenvolvimento do sistema. Estes diagramas desempenham um papel crucial na compreensão da estrutura e na narrativa do processo de desenvolvimento do sistema, oferecendo uma visão aprofundada de sua arquitetura. Ao examinar cada diagrama, buscaremos desvendar não apenas a estrutura técnica, mas também a lógica e a dinâmica subjacentes ao desenvolvimento, proporcionando uma base sólida para a compreensão abrangente do projeto.

## 3.1 Diagrama de Classes

O diagrama de classe é fundamental na *UML*, sendo essencial para criar os demais diagramas do conjunto.

Seu principal enfoque está em permitir a visualização das classes que comporão o sistema com seus respectivos atributos e métodos, bem como em demonstrar como as classes do diagrama se relacionam, complementam e transmitem informações entre si. Esse diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em como definir a estrutura lógica delas (Guedes, 2018, p.133).

Sommerville (2018) esclarece que os diagramas de classe são estruturados em torno de um retângulo simples que representa uma classe. Dentro desse retângulo, o nome da classe é exibido na parte superior, enquanto os atributos e seus tipos são listados logo abaixo. As relações entre as classes são indicadas por linhas que conectam os retângulos, evidenciando a existência de associações entre elas.

O diagrama de classe apresentado na Figura 8 foi modelado para o projeto SIGEDESP, tendo a classe “Despesa” como seu componente central. Esta classe é responsável pelo cadastro e gestão das despesas, que constitui a funcionalidade principal do sistema, sendo essencial para a operação eficiente da estrutura financeira.

Além da classe “Despesa”, o diagrama inclui diversas outras classes que desempenham papéis específicos na organização e gerenciamento das informações. A classe “Instituição” registra as instituições vinculadas às despesas, enquanto a classe “Secretaria” define a secretaria da prefeitura à qual a instituição pertence, estabelecendo a hierarquia administrativa envolvida. A classe “Fornecedor” gerencia o cadastro dos fornecedores de bens ou serviços, crucial para o rastreamento e controle das despesas. A classe “Usuário” permite o cadastro e a gestão de perfis de usuários, atribuindo permissões e responsabilidades específicas no gerenciamento das despesas.

Complementando essa estrutura, o sistema inclui classes que proporcionam categorização adicional das informações. A classe “Tipo Instituição” classifica as instituições por tipo, e a classe “Unidade Consumidora” identifica cada instituição em relação às despesas geradas. A classe “Tipo Despesa” categoriza as despesas conforme sua natureza (como água, energia, internet), e a classe “Unidade de Medida” possibilita a quantificação dos gastos com base em diferentes unidades de medida.

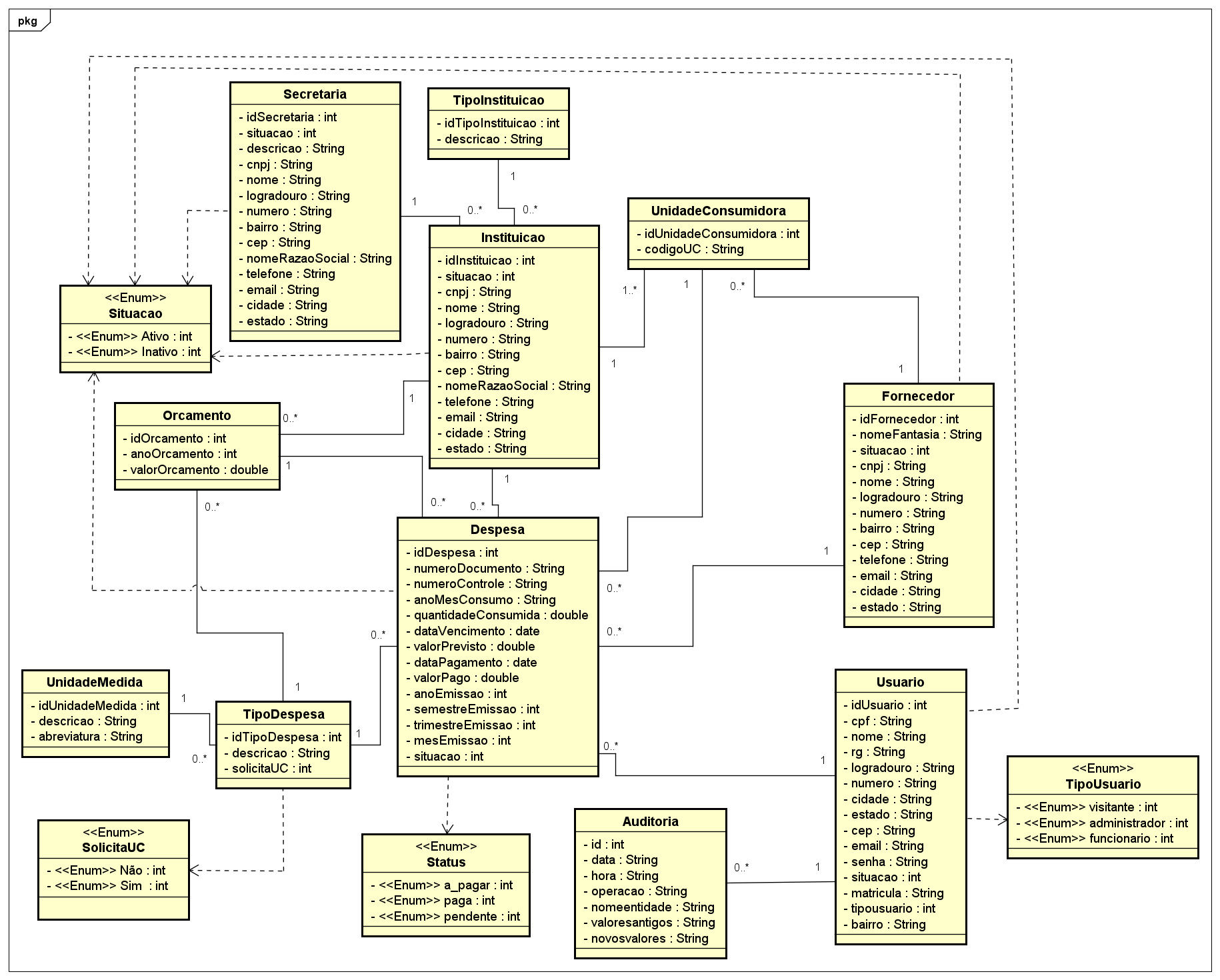
A classe “Orçamento” tem a função de avaliar os gastos realizados e projetar o orçamento futuro com base nas despesas registradas e nas tendências identificadas.

O controle de acesso é realizado por meio de autenticação de usuários, que necessitam realizar login com um nome de usuário e senha. A classe “Auditoria” registra informações sobre o acesso e as ações realizadas, garantindo a segurança e integridade do sistema.

O diagrama também inclui quatro classes de enumeração, conforme descrito por Guedes (2018), que definem os valores possíveis para determinadas variáveis. A classe “Tipo Usuário” categoriza os usuários do sistema, permitindo sua organização hierárquica com permissões específicas. A classe “Situação” classifica os elementos das demais classes como ativos ou inativos. A classe “SolicitaUC” (Unidade Consumidora) permite à classe “Tipo Despesa” definir a necessidade de solicitar um código de “Unidade Consumidora”. A classe “Status” categoriza os status da despesa no sistema, definindo-a como a pagar, paga ou pendente, isso permite a organização das despesas para identificar se elas precisam ser pagas ou se já estão pagas.

Em suma, o diagrama de classe do SIGEDESP está estruturado para proporcionar uma gestão financeira detalhada e eficiente, com categorização precisa das despesas, controle rigoroso de acesso e auditoria, e suporte para análise orçamentária.

Figura 8 – Diagrama de Classe



Fonte: Elaborada pelos autores.

## 3.2 Dicionário de Classes

Nesta subseção, serão apresentados os Dicionários de Classes, que têm como objetivo proporcionar uma compreensão detalhada do funcionamento do sistema. Cada quadro a seguir (Quadros 3 a 17) descreve uma classe, seus atributos e suas funcionalidades no contexto da gestão de despesas públicas pela Prefeitura Municipal.

A classe “Despesa” (Quadro 3) é responsável pelo registro das despesas realizadas pelas instituições, de acordo com a gestão das secretarias. Esse registro permite a organização e o controle das transações financeiras realizadas no âmbito da administração pública municipal, sendo crucial para o monitoramento e o controle dos gastos de cada instituição.

Quadro - Descrição Classe Despesa

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idDespesa | Integer | Código que irá identificar a despesa. |
| numeroDocumento | String | Número do Documento (Despesa) emitido. |
| numeroControle | String | Número de Controle da Despesa. |
| anoMesConsumo | String | Ano e Mês do Consumo da Despesa. |
| quantidadeConsumida | Double | Quantidade de recursos consumidos. |
| dataVencimento | Date | Data de Vencimento da Despesa. |
| valorPrevisto | Double | Valor total previsto na Despesa. |
| dataPagamento | Date | Data de Pagamento da Despesa. |
| valorPago | Double | Valor que foi pago. |
| anoEmissao | Integer | Ano de Emissão da Despesa. |
| semestreEmissao | Integer | Semestre de Emissão da Despesa. |
| trimestreEmissao | Integer | Trimestre de Emissão da Despesa. |
| mesEmissao | Integer | Mês de Emissão da Despesa. |
| Situação | Integer | Situação que a Despesa se encontra. |

Fonte: Elaborada pelos autores.

A classe “Secretaria” (Quadro 4) tem como objetivo registrar as secretarias da Prefeitura Municipal. Ela permite associar cada instituição e suas respectivas despesas à secretaria responsável, proporcionando uma visão clara sobre os gastos de cada secretaria. Isso é fundamental para garantir o planejamento e controle orçamentário adequado em relação aos recursos disponíveis para cada secretaria.

Quadro - Descrição Classe Secretaria

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idSecretaria | Integer | Código que irá identificar a secretária. |
| Situação | Integer | Classificação da situação em que a secretária se encontra, como por exemplo, ativo ou inativo. |
| descricao | String | Identificar para qual secretaria a despesa está sendo cadastrada. |
| cnpj | String | CNPJ da Secretaria. |
| nome | String | Nome da Secretaria. |
| logradouro | String | Logradouro que a Secretaria fica. |
| numero | String | Número do local que a Secretaria fica. |
| bairro | String | Bairro da Secretaria. |
| cep | Integer | CEP da Secretaria. |
| nomeRazaoSocial | String | Nome Razão Social da Secretaria. |
| telefone | String | Telefone da Secretaria. |
| email | String | E-mail da Secretaria. |
| cidade | String | Cidade da Secretaria. |
| estado | String | Estado da Secretaria. |

Fonte: Elaborada pelos autores.

A classe “Instituição” (Quadro 5) é destinada ao cadastro das instituições que compõem a administração pública municipal. Cada instituição pode ter despesas associadas a ela, o que permite a organização e categorização dessas despesas dentro do sistema. Essa classe é essencial para facilitar a filtragem de dados e a geração de relatórios financeiros, tanto para controle interno quanto para auditorias.

Quadro - Descrição Classe Instituição

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idInstituicao | Integer | Código que irá identificar a instituição. |
| situação | Integer | Classificação da situação em que a instituição se encontra, como por exemplo, ativo ou inativo. |
| cnpj | String | CNPJ da Instituição. |
| nome | String | Nome da Instituição. |
| logradouro | String | Logradouro que a Instituição fica. |
| numero | Integer | Número do local que a Instituição fica. |
| bairro | String | Bairro da Instituição. |
| cep | Integer | CEP da Instituição. |
| nomeRazaoSocial | String | Nome Razão Social da Instituição. |
| telefone | String | Telefone da Instituição. |
| email | String | E-mail da Instituição. |
| cidade | String | Cidade da Instituição. |
| estado | String | Estado da Instituição. |

Fonte: Elaborada pelos autores.

A classe “Fornecedor” (Quadro 6) é responsável pelo cadastro das empresas ou entidades que fornecem bens ou serviços ao município, resultando em despesas. Associar fornecedores às respectivas despesas permite um acompanhamento eficiente dos processos de aquisição e contratação, além de facilitar a gestão de pagamentos e o controle orçamentário.

Quadro - Descrição Classe Fornecedor

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idFornecedor | Integer | Código que irá identificar o fornecedor. |
| nomeFantasia | String | Nome Comercial da empresa que é fornecedora. |
| situação | Integer | Classificação ativo ou inativo no sistema. |
| cnpj | String | CNPJ do Fornecedor. |
| nome | String | Nome do Fornecedor. |
| logradouro | String | Logradouro que o Fornecedor fica. |
| numero | String | Número do local que o Fornecedor fica. |
| bairro | String | Bairro do Fornecedor. |
| cep | Integer | CEP do Fornecedor. |
| telefone | String | Telefone do Fornecedor. |
| email | String | E-mail do Fornecedor. |
| cidade | String | Cidade da Fornecedor. |
| estado | String | Estado da Fornecedor. |

Fonte: Elaborada pelos autores.

A classe “Orçamento” (Quadro 7) tem como objetivo registrar os orçamentos planejados para cobrir os pagamentos das despesas ao longo de um determinado período. Esses dados são fundamentais para a elaboração de relatórios financeiros e dashboards, permitindo um acompanhamento detalhado das previsões de gastos e facilitando o planejamento orçamentário para os anos seguintes.

Quadro - Descrição Classe Orçamento

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idOrcamento | Integer | Código que irá identificar o orçamento. |
| anoOrcamento | Integer | Ano em que o orçamento foi calculado. |
| valorOrcamento | Double | Valor que foi calculado para o orçamento. |

Fonte: Elaborada pelos autores.

A classe “Unidade de Medida” (Quadro 8) é responsável pelo cadastro e gerenciamento das unidades de medida associadas às diferentes despesas. Seu principal objetivo é garantir a contabilização precisa dos gastos, levando em consideração as unidades específicas em que cada despesa se manifesta. Esse processo facilita a análise financeira detalhada e o controle orçamentário por unidade de medida.

Quadro - Descrição Classe Unidade Medida

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idUnidadeMedida | Integer | Código que irá identificar a unidade de medida de cada instituição. |
| descricao | String | Nome da unidade de medida. |
| abreviatura | String | Abreviatura do nome da unidade de medida. |

Fonte: Elaborada pelos autores.

A classe “Unidade Consumidora” (Quadro 9) é responsável pelo cadastro dos códigos identificadores das unidades consumadoras de despesas, que correspondem às diferentes instituições. Isso possibilita a organização e o controle das despesas de maneira individualizada por unidade consumidora, promovendo maior transparência e precisão nas informações financeiras.

Quadro - Descrição Classe Unidade Consumidora

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idUnidadeConsumidora | Integer | Código que irá identificar a unidade consumidora de cada instituição. |
| codigoUC | Integer | Código da unidade consumidora. |

Fonte: Elaborada pelos autores.

A classe “Tipo Instituição” (Quadro 10) tem como finalidade cadastrar e classificar os tipos de instituições que fazem parte do sistema. Essa categorização permite que as despesas e orçamentos sejam segregados e apresentados nos relatórios e dashboards conforme o tipo de instituição, facilitando a análise e o acompanhamento financeiro de cada categoria institucional.

Quadro - Descrição Classe Tipo Instituição

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idTipoInstituicao | Integer | Código que irá identificar o tipo de instituição. |
| descricao | String | Classificação do Tipo de instituição. |

Fonte: Elaborada pelos autores.

A classe “Tipo Despesa” (Quadro 11) visa registrar e classificar as diversas categorias de despesas. A categorização das despesas facilita a identificação do tipo específico de gasto, possibilitando a geração de relatórios e dashboards mais detalhados. Esses relatórios ajudam a monitorar os tipos de despesas e os montantes associados a cada categoria, melhorando a gestão e o controle financeiro.

Quadro - Descrição Classe TipoDespesa

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idTipoDespesa | Integer | Código que irá identificar o tipo de despesa. |
| descricao | String | Classificação do tipo de despesa. |
| solicitaUC | Integer | Solicita ou Não o Código UC. |

Fonte: Elaborada pelos autores.

A classe “Usuário” (Quadro 12) armazena informações essenciais sobre as pessoas que acessam o sistema. Ela permite o gerenciamento do acesso ao sistema, garantindo que as permissões de cada usuário sejam adequadas ao seu papel, além de assegurar a segurança e a integridade dos dados no sistema.

Quadro - Descrição Classe Usuário

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| idUsuario | Integer | Código que irá identificar o usuário. |
| cpf | String | CPF do Usuário. |
| nome | String | Nome do Usuário. |
| rg | String | RG do Usuário. |
| logradouro | String | Logradouro do Usuário. |
| numero | Integer | Número da habitação. |
| cidade | String | Cidade do usuário. |
| estado | String | Estado que o usuário vive. |
| cep | Integer | CEP do Usuário. |
| bairro | String | Bairro do Usuário. |
| email | String | E-mail do Usuário. |
| senha | String | Senha do Usuário. |
| situação | Integer | Situação que o Usuário se encontra em relação ao Sistema, como ativo ou inativo. |
| matricula | String | Número da Matrícula do Usuário no sistema. |
| tipousuario | Integer | Nível de hierarquia do usuário no sistema. |

Fonte: Elaborada pelos autores.

A classe “Auditoria” (Quadro 13) é responsável pelo armazenamento de informações relacionadas a todas as interações dos usuários com o sistema, como alterações de dados, inclusões, exclusões e qualquer outra operação relevante. Seu principal objetivo é garantir a conformidade, permitindo o monitoramento detalhado de cada ação realizada, a fim de assegurar a integridade das informações e facilitar a realização de auditorias.

Quadro - Descrição Classe Auditoria

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| id | Integer | Código que irá identificar a auditoria. |
| data | String | Data da auditoria. |
| hora | String | Horário da auditoria. |
| operacao | String | Operação que foi realizada. |
| nomeEntidade | String | Entidade que foi sujeita a inserção, remoção ou edição. |
| valorAntigo | String | Informação substituída. |
| novosValores | String | Novo valor atualizado. |
| idUsuario | Interger | Código do Usuário que fez as alterações. |

Fonte: Elaborada pelos autores.

A enum “Tipo Usuário” (Quadro 14) é uma enumeração que define os diferentes papéis que os usuários podem exercer dentro do sistema. Esta funcionalidade é essencial para o controle de acesso, pois permite a personalização das permissões e restrições de cada usuário de acordo com sua função, garantindo a segurança e a eficiência no uso do sistema.

Quadro - Descrição Enum Tipo Usuário - Enumeração dos tipos de usuário no sistema

|  |  |  |
| --- | --- | --- |
| **Enumerações** | **Tipo** | **Descrição** |
| Visitante | Integer | Representa um usuário comum do sistema, com acesso básico a algumas funcionalidades do sistema. |
| Funcionario | Integer | Representa um usuário com funcionalidades limitadas no sistema, em comparação com o administrador. |
| Administrador | Integer | Representa um usuário com privilégios administrativos no sistema |

Fonte: Elaborada pelos autores.

A enum “Situação” (Quadro 15) é uma enumeração que define os diferentes estados de uma entidade dentro do sistema. Ela é essencial para acompanhar o status de cada item ou processo, fornecendo uma visão atualizada e precisa sobre a condição ou o andamento de qualquer elemento no sistema.

Quadro - Descrição Enum Situação - Enumeração de Situação no sistema

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| Ativo | Integer | Representa a situação de ativo de uma entidade no sistema |
| Inativo | Integer | Representa o status de inatividade de uma entidade no sistema. |

Fonte: Elaborada pelos autores.

A enum “Solicita UC” é uma enumeração que possibilita à classe “Tipo Despesa” escolher se será necessário ou não solicitar um código de “Unidade Consumidora” para registrar a despesa. Isso permite flexibilidade na gestão das despesas, dependendo da necessidade de se associar ou não a um código de unidade consumidora.

Quadro - Descrição Enum Solicita UC - Enumeração de Solicita UC no sistema

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Tipo** | **Descrição** |
| Não | Integer | Representa que a classe Tipo Despesa não solicita um Código de Unidade Consumidora. |
| Sim | Integer | Representa que a classe Tipo Despesa solicita um Código de Unidade Consumidora. |

Fonte: Elaborada pelos autores.

A enum “Status” (Quadro 17) é uma enumeração que define os diferentes estados que uma despesa pode ter dentro do sistema. Ela é essencial para acompanhar o status de cada despesa, fornecendo uma visão atualizada e precisa sobre o andamento do processo da despesa no sistema.

Quadro - Descrição Enum Status - Enumeração dos status de uma Despesa

|  |  |  |
| --- | --- | --- |
| **Enumerações** | **Tipo** | **Descrição** |
| A Pagar | Integer | Representa uma despesa com o estado de a pagar no sistema. |
| Paga | Integer | Representa uma despesa com o estado de paga no sistema. |
| Pendente | Integer | Representa uma despesa com o estado de pendente no sistema. |

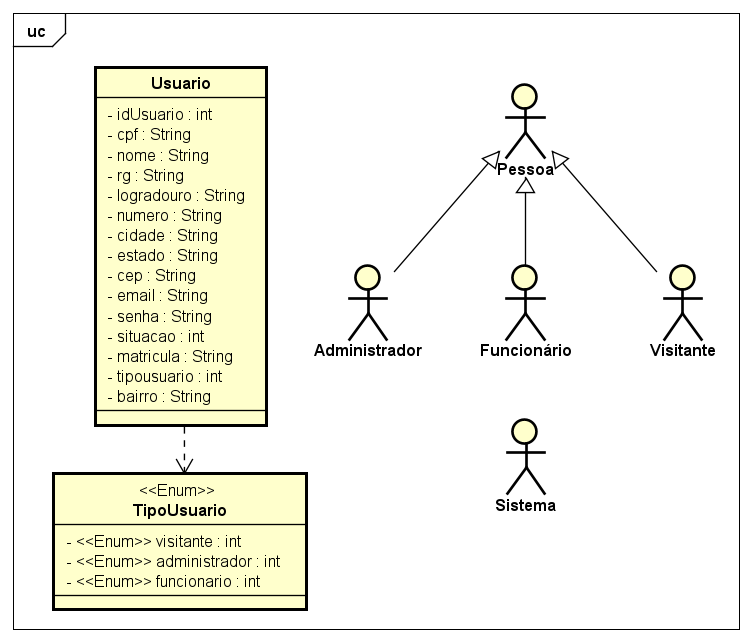
Fonte: Elaborada pelos autores.

## 3.3 Definição dos Atores

Conforme apresentado na Figura 9, a definição estrutural no diagrama de classes para o diagrama de atores evita o uso de herança. Essa escolha é fundamentada nos princípios do *SOLID*[[1]](#footnote-1), em especial no Princípio da Substituição de Liskov (LSP) e no Princípio de Segregação de Interfaces (ISP). De acordo com esses princípios, deve-se priorizar o uso da composição em vez da herança, pois a composição oferece maior flexibilidade e modularidade. Além disso, essa abordagem reduz o acoplamento entre as classes, mantendo a coesão e a consistência do sistema (Aniche, 2011).

Na Figura 9 também é apresentado a classe “Tipo Usuário”, que é classificada como um estereótipo do tipo <<*enumeration*>>. De acordo com Guedes (2018, p. 171), “essa classe lista todos os valores válidos que um tipo de dados pode assumir”. Ou seja, no contexto do sistema, o dado em questão é o “Usuário” e os valores que ele pode assumir são “Visitante”, “Funcionário” e “Administrador”. Isso possibilita a definição de diferentes níveis de acesso ao sistema, garantindo que cada tipo de usuário tenha permissões específicas, de acordo com suas responsabilidades e funções dentro da plataforma.

Figura – Atores do Sistema



Fonte: Elaborada pelos autores.

O ator "Pessoa" (Figura 9) representa coletivamente todos os tipos de usuários envolvidos, incluindo Administradores, Funcionários e o Público. Esse ator abrange todas as interações possíveis no sistema, desde administração e manutenção por parte dos Administradores, até as operações diárias dos Funcionários responsáveis pelo registro e acompanhamento das despesas. Além disso, o Público, como Pessoa, pode acessar informações específicas de forma mais restrita, promovendo transparência nas atividades financeiras da prefeitura. O ator "Pessoa" encapsula as diversas necessidades e permissões associadas aos diferentes papéis dentro do contexto do sistema de gerenciamento de despesas.

O ator "Administrador" (Figura 9) desempenha o papel central na administração e supervisão do sistema. Suas responsabilidades incluem a configuração e manutenção do sistema, a gestão de usuários e permissões, a definição de parâmetros do sistema, e a geração de relatórios estratégicos. O Administrador possui o controle global sobre as funcionalidades do sistema, garantindo que este atenda aos requisitos da prefeitura e que as operações sejam conduzidas de maneira eficiente e segura.

O ator "Funcionário" (Figura 9) representa os usuários operacionais que utilizam o sistema para realizar tarefas específicas relacionadas ao gerenciamento e registro de despesas. Suas responsabilidades podem incluir o cadastro de despesas, a associação de despesas a instituições, a consulta de relatórios e a interação direta com as funcionalidades cotidianas do sistema. O Funcionário desempenha um papel crucial na entrada de dados e na execução de operações diárias para garantir a precisão e integridade das informações registradas no sistema de gerenciamento de despesas.

O ator "Visitante" (Figura 9) refere-se aos usuários externos, como cidadãos, contribuintes ou qualquer outra parte interessada que tenha acesso limitado ao sistema. O acesso do Visitante geralmente é direcionado para consultar informações públicas sobre despesas, orçamentos ou relatórios que a prefeitura opta por tornar disponíveis. Essa categoria de usuários pode visualizar dados específicos, geralmente de forma transparente, contribuindo para a prestação de contas e transparência na gestão financeira da prefeitura. Suas interações são frequentemente limitadas a consultas e visualização de informações, sem permissão para realizar ações de entrada de dados ou configuração no sistema.

O ator "Sistema" (Figura 9) refere-se à própria aplicação ou plataforma. Este ator não representa um usuário humano, mas sim o conjunto de funcionalidades, processos e lógicas incorporadas no software. O Sistema desempenha um papel central na automação de tarefas, processamento de dados, armazenamento de informações e execução de operações específicas relacionadas ao gerenciamento de despesas. Ele é projetado para suportar os diferentes papéis de usuários (Administrador, Funcionário, Visitante) e facilitar eficientemente as operações e análises relacionadas às atividades financeiras da prefeitura.

## 3.4 Lista de Casos de Uso

Guedes (2018, p. 63) destaca que os casos de uso visam identificar tanto os usuários do sistema quanto os requisitos necessários, definindo "serviços, tarefas ou funcionalidades identificados como necessários ao software e que podem ser utilizados de alguma maneira pelos atores que interagem com o sistema". Para assegurar uma compreensão completa do funcionamento do sistema, foram elaboradas duas listas. A primeira apresenta todas as mensagens que o sistema retornará (Quadro 18) e a outra, os requisitos de casos de uso (Quadro 19).

A lista de mensagens inclui as principais respostas que o sistema fornecerá durante a execução dos casos de uso, ou seja, durante a interação do usuário com as funcionalidades do sistema. O Quadro 18 apresenta a lista completa das mensagens do sistema SIGEDESP.

Quadro - Lista de Mensagens

|  |  |
| --- | --- |
| **N° da Mensagem** | **Conteúdo da Mensagem** |
| Msg 1 | Cadastrado com Sucesso |
| Msg 2 | Alterado com Sucesso |
| Msg 3 | Excluído com Sucesso |
| Msg 4 | Desativado com Sucesso |
| Msg 5 | Login realizado com Sucesso |
| Msg 6 | Campos em branco |
| Msg 7 | Erro ao conectar com o banco de dados |
| Msg 8 | Dado já existe |
| Msg 9 | Retorna Dados |
| Msg 10 | Relatório Gerado com Sucesso |

Fonte: Elaborada pelos autores.

A lista de casos de uso pode ser considerada um dicionário de dados que armazena informações detalhadas sobre como o sistema funcionará e como os atores interagirão com ele. O Quadro 19 apresenta a lista completa dos casos de uso do sistema.

Quadro - Lista de Casos de Uso

| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| --- | --- | --- | --- | --- |
| 1 | Administrador realiza Login | Informa dados de acesso | Efetuar login | Msg 5 |
| 2 | Administrador altera administrador | Informa dados administrador | Alterar administrador | Msg 2 |
| 3 | Administrador altera funcionário | Informa dados funcionário | Alterar funcionário | Msg 2 |
| 4 | Administrador altera visitante | Informa dados visitante | Alterar visitante | Msg 2 |
| 5 | Administrador gera relatório | Id relatório | Gerar Relatório | Msg 2 |
| 6 | Administrador altera usuário | Informa dados de usuário | Alterar usuário | Msg 2 |
| 7 | Administrador altera tipo despesa | Informa dados de tipo despesa | Alterar tipo despesa | Msg 2 |
| 8 | Administrador altera secretaria | Informa dados de secretaria | Alterar secretaria | Msg 2 |
| 9 | Administrador altera instituição | Informa dados de instituição | Alterar instituição | Msg 2 |
| 10 | Administrador altera orçamento | Informa dados de orçamento | Alterar orçamento | Msg 2 |
| 11 | Administrador altera fornecedor | Informa dados de fornecedor | Alterar fornecedor | Msg 2 |
| 12 | Administrador altera despesa | Informa dados de despesa | Alterar despesa | Msg 2 |
| 13 | Administrador altera unidade consumidora | Informa dados de unidade consumidora | Alterar unidade consumidora | Msg 2 |
| 14 | Administrador altera unidade medida | Informa dados de unidade medida | Alterar unidade medida | Msg 2 |
| 15 | Administrador altera tipo instituição | Informa dados de tipo instituição | Alterar tipo instituição | Msg 2 |
| 16 | Administrador cadastra administrador | Informa dados de administrador | Cadastrar administrador | Msg 1 |
| 17 | Administrador cadastra funcionário | Informa dados usuário | Cadastrar funcionário | Msg 1 |
| 18 | Administrador cadastra visitante | Informa dados de funcionário | Cadastrar visitante | Msg 1 |
| 19 | Administrador cadastra usuário | Informa dados de usuário | Cadastrar usuário | Msg 1 |
| 20 | Administrador cadastra tipo despesa | Informa dados de tipo despesa | Cadastrar tipo despesa | Msg 1 |
| 21 | Administrador cadastra secretaria | Informa dados de secretaria | Cadastrar secretaria | Msg 1 |
| 22 | Administrador cadastra instituição | Informa dados de instituição | Cadastrar instituição | Msg 1 |
| 23 | Administrador cadastra orçamento | Informa dados de orçamento | Cadastrar orçamento | Msg 1 |
| 24 | Administrador cadastra fornecedor | Informa dados de fornecedor | Cadastrar fornecedor | Msg 1 |
| 25 | Administrador cadastra despesa | Informa dados de despesa | Cadastrar despesa | Msg 1 |
| 26 | Administrador cadastra unidade consumidora | Informa dados de unidade consumidora | Cadastrar unidade consumidora | Msg 1 |
| 27 | Administrador cadastra unidade medida | Informa dados de unidade medida | Cadastrar unidade medida | Msg 1 |
| 28 | Administrador cadastra tipo instituição | Informa dados de tipo instituição | Cadastrar tipo instituição | Msg 1 |
| 29 | Administrador exclui administrador | Id administrador | Excluir administrador | Msg 3 |
| 30 | Administrador exclui funcionário | Id funcionário | Excluir funcionário | Msg 3 |
| 31 | Administrador exclui visitante | Id visitante | Excluir visitante | Msg 3 |
| 32 | Administrador exclui usuário | Id usuário | Excluir usuário | Msg 3 |
| 33 | Administrador exclui tipo despesa | Id tipo despesa | Excluir tipo despesa | Msg 3 |
| 34 | Administrador exclui secretaria | Id secretaria | Excluir secretaria | Msg 3 |
| 35 | Administrador exclui instituição | Id instituição | Excluir instituição | Msg 3 |
| 36 | Administrador exclui orçamento | Id orçamento | Excluir orçamento | Msg 3 |
| 37 | Administrador exclui fornecedor | Id fornecedor | Excluir fornecedor | Msg 3 |
| 38 | Administrador exclui despesa | Id despesa | Excluir despesa | Msg 3 |
| 39 | Administrador exclui unidade consumidora | Id unidade consumidora | Excluir unidade consumidora | Msg 3 |
| 40 | Administrador exclui unidade medida | Id unidade medida | Excluir unidade medida | Msg 3 |
| 41 | Administrador exclui tipo instituição | Id tipo instituição | Excluir tipo instituição | Msg 3 |
| 42 | Administrador consulta administrador | Id administrador | Consultar administrador | Msg 9 |
| 43 | Administrador consulta funcionário | Id funcionário | Consultar funcionário | Msg 9 |
| 44 | Administrador consulta visitante | Id visitante | Consultar visitante | Msg 9 |
| 45 | Administrador consulta usuário | Id usuário | Consultar usuário | Msg 9 |
| 46 | Administrador consulta tipo despesa | Id tipo despesa | Consultar tipo despesa | Msg 9 |
| 47 | Administrador consulta secretaria | Id secretaria | Consultar secretaria | Msg 9 |
| 48 | Administrador consulta instituição | Id instituição | Consultar instituição | Msg 9 |
| 49 | Administrador consulta orçamento | Id orçamento | Consultar orçamento | Msg 9 |
| 50 | Administrador consulta fornecedor | Id fornecedor | Consultar fornecedor | Msg 9 |
| 51 | Administrador consulta despesa | Id despesa | Consultar despesa | Msg 9 |
| 52 | Administrador consulta unidade consumidora | Id unidade consumidora | Consultar unidade consumidora | Msg 9 |
| 53 | Administrador consulta unidade medida | Id unidade medida | Consultar unidade medida | Msg 9 |
| 54 | Administrador consulta tipo instituição | Id tipo instituição | Consultar tipo instituição | Msg 9 |
| 55 | Administrador consulta relatório | Id relatório | Consultar relatório | Msg 9 |
| 56 | Administrador solicita lista de administrador | Id administrador | Listar administrador | Msg 9 |
| 57 | Administrador solicita lista de funcionário | Id funcionário | Listar funcionário | Msg 9 |
| 58 | Administrador solicita lista de visitante | Id visitante | Listar visitante | Msg 9 |
| 59 | Administrador solicita lista de usuário | Id usuário | Listar usuário | Msg 9 |
| 60 | Administrador solicita lista de tipo despesa | Id tipo despesa | Listar tipo despesa | Msg 9 |
| 61 | Administrador solicita lista de secretaria | Id secretaria | Listar secretaria | Msg 9 |
| 62 | Administrador solicita lista de instituição | Id instituição | Listar instituição | Msg 9 |
| 63 | Administrador solicita lista de orçamento | Id orçamento | Listar orçamento | Msg 9 |
| 64 | Administrador solicita lista de fornecedor | Id fornecedor | Listar fornecedor | Msg 9 |
| 65 | Administrador solicita lista de despesa | Id despesa | Listar despesa | Msg 9 |
| 66 | Administrador solicita lista de unidade consumidora | Id unidade consumidora | Listar unidade consumidora | Msg 9 |
| 67 | Administrador solicita lista de unidade medida | Id unidade medida | Listar unidade medida | Msg 9 |
| 68 | Administrador solicita lista de tipo instituição | Id tipo instituição | Listar tipo instituição | Msg 9 |
| 69 | Administrador solicita lista de relatório | Id relatório | Listar relatório | Msg 9 |
| 70 | Funcionário realiza login | Informa dados de acesso | Efetuar login | Msg 5 |
| 71 | Funcionário gera relatório | Id relatório | Gerar Relatório | Msg 10 |
| 72 | Funcionário altera tipo despesa | Informa dados despesa | Alterar despesa | Msg 2 |
| 73 | Funcionário altera secretaria | Informa dados secretaria | Alterar secretaria | Msg 2 |
| 74 | Funcionário altera instituição | Informa dados de instituição | Alterar instituição | Msg 2 |
| 75 | Funcionário altera orçamento | Informa dados de orçamento | Alterar orçamento | Msg 2 |
| 76 | Funcionário altera fornecedor | Informa dados de fornecedor | Alterar fornecedor | Msg 2 |
| 77 | Funcionário altera despesa | Informa dados de despesa | Alterar despesa | Msg 2 |
| 78 | Funcionário altera unidade consumidora | Informa dados unidade consumidora | Alterar unidade consumidora | Msg 2 |
| 79 | Funcionário altera unidade medida | Informa dados unidade medida | Alterar unidade medida | Msg 2 |
| 80 | Funcionário altera tipo instituição | Informa dados tipo instituição | Alterar tipo instituição | Msg 2 |
| 81 | Funcionário cadastra tipo despesa | Informa dados de tipo despesa | Cadastrar tipo despesa | Msg 1 |
| 82 | Funcionário cadastra secretaria | Informa dados de secretaria | Cadastrar secretaria | Msg 1 |
| 83 | Funcionário cadastra instituição | Informa dados de instituição | Cadastrar instituição | Msg 1 |
| 84 | Funcionário cadastra orçamento | Informa dados de orçamento | Cadastrar orçamento | Msg 1 |
| 85 | Funcionário cadastra fornecedor | Informa dados fornecedor | Cadastrar fornecedor | Msg 1 |
| 86 | Funcionário cadastra despesa | Informa dados de despesa | Cadastrar despesa | Msg 1 |
| 87 | Funcionário cadastra unidade consumidora | Informa dados de unidade consumidora | Cadastrar unidade consumidora | Msg 1 |
| 88 | Funcionário cadastra unidade medida | Informa dados de unidade medida | Cadastrar unidade medida | Msg 1 |
| 89 | Funcionário cadastra tipo instituição | Informa dados de tipo instituição | Cadastrar tipo instituição | Msg 1 |
| 90 | Funcionário exclui tipo despesa | Id tipo despesa | Excluir tipo despesa | Msg 3 |
| 91 | Funcionário exclui secretaria | Id secretaria | Excluir secretaria | Msg 3 |
| 92 | Funcionário exclui instituição | Id instituição | Excluir instituição | Msg 3 |
| 93 | Funcionário exclui orçamento | Id orçamento | Excluir orçamento | Msg 3 |
| 94 | Funcionário exclui fornecedor | Id fornecedor | Excluir fornecedor | Msg 3 |
| 95 | Funcionário exclui despesa | Id despesa | Excluir despesa | Msg 3 |
| 96 | Funcionário exclui unidade consumidora | Id unidade consumidora | Excluir unidade consumidora | Msg 3 |
| 97 | Funcionário exclui unidade medida | Id unidade medida | Excluir unidade medida | Msg 3 |
| 98 | Funcionário exclui tipo instituição | Id tipo instituição | Excluir tipo instituição | Msg 3 |
| 99 | Funcionário consulta tipo despesa | Id tipo despesa | Consultar tipo despesa | Msg 9 |
| 100 | Funcionário consulta secretaria | Id secretaria | Consultar secretaria | Msg 9 |
| 101 | Funcionário consulta instituição | Id instituição | Consultar instituição | Msg 9 |
| 102 | Funcionário consulta orçamento | Id orçamento | Consultar orçamento | Msg 9 |
| 103 | Funcionário consulta fornecedor | Id fornecedor | Consultar fornecedor | Msg 9 |
| 104 | Funcionário consulta despesa | Id despesa | Consultar despesa | Msg 9 |
| 105 | Funcionário consulta unidade consumidora | Id unidade consumidora | Consultar unidade consumidora | Msg 9 |
| 106 | Funcionário consulta unidade medida | Id unidade medida | Consultar unidade medida | Msg 9 |
| 107 | Funcionário consulta tipo instituição | Id tipo instituição | Consultar tipo instituição | Msg 9 |
| 108 | Funcionário consulta relatório | Id relatório | Consultar relatório | Msg 9 |
| 109 | Funcionário solicita lista de tipo despesa | Id tipo despesa | Listar tipo despesa | Msg 9 |
| 110 | Funcionário solicita lista de secretaria | Id secretaria | Listar secretaria | Msg 9 |
| 111 | Funcionário solicita lista de instituição | Id instituição | Listar instituição | Msg 9 |
| 112 | Funcionário solicita lista de orçamento | Id orçamento | Listar orçamento | Msg 9 |
| 113 | Funcionário solicita lista de fornecedor | Id fornecedor | Listar fornecedor | Msg 9 |
| 114 | Funcionário solicita lista de despesa | Id despesa | Listar despesa | Msg 9 |
| 115 | Funcionário solicita lista de unidade consumidora | Id unidade consumidora | Listar unidade consumidora | Msg 9 |
| 116 | Funcionário solicita lista de unidade medida | Id unidade medida | Listar unidade medida | Msg 9 |
| 117 | Funcionário solicita lista de tipo instituição | Id tipo instituição | Listar tipo instituição | Msg 9 |
| 118 | Funcionário solicita lista de relatório | Id relatório | Listar relatório | Msg 9 |
| 119 | Visitante realiza login | Informa dados de acesso | Efetuar login | Msg 5 |
| 120 | Visitante gera relatório | Id relatório | Gerar Relatório | Msg 10 |
| 121 | Visitante consulta secretaria | Id secretaria | Consultar secretaria | Msg 9 |
| 122 | Visitante consulta instituição | Id instituição | Consultar instituição | Msg 9 |
| 123 | Visitante consulta orçamento | Id orçamento | Consultar orçamento | Msg 9 |
| 124 | Visitante consulta fornecedor | Id fornecedor | Consultar fornecedor | Msg 9 |
| 125 | Visitante consulta despesa | Id despesa | Consultar despesa | Msg 9 |
| 126 | Visitante consulta relatório | Id relatório | Consultar relatório | Msg 9 |
| 127 | Visitante solicita lista de secretaria | Id secretaria | Listar secretaria | Msg 9 |
| 128 | Visitante solicita lista de instituição | Id instituição | Listar instituição | Msg 9 |
| 129 | Visitante solicita lista de orçamento | Id orçamento | Listar orçamento | Msg 9 |
| 130 | Visitante solicita lista de fornecedor | Id fornecedor | Listar fornecedor | Msg 9 |
| 131 | Visitante solicita lista de despesa | Id despesa | Listar despesa | Msg 9 |
| 132 | Visitante solicita lista de relatório | Id relatório | Listar relatório | Msg 9 |

Fonte: Elaborada pelos autores.

## 3.5 Diagrama de Casos de Uso

Guedes (2018) descreve o diagrama de casos de uso como uma representação do sistema a partir da perspectiva do usuário, cujo objetivo é apresentar de forma geral as funcionalidades que o sistema deve oferecer ao usuário. Além disso, Guedes (2018) destaca que esse diagrama é uma ferramenta essencial para identificar e compreender os requisitos do sistema, auxiliando na especificação, visualização e documentação das características, funções e serviços que o sistema deve proporcionar.

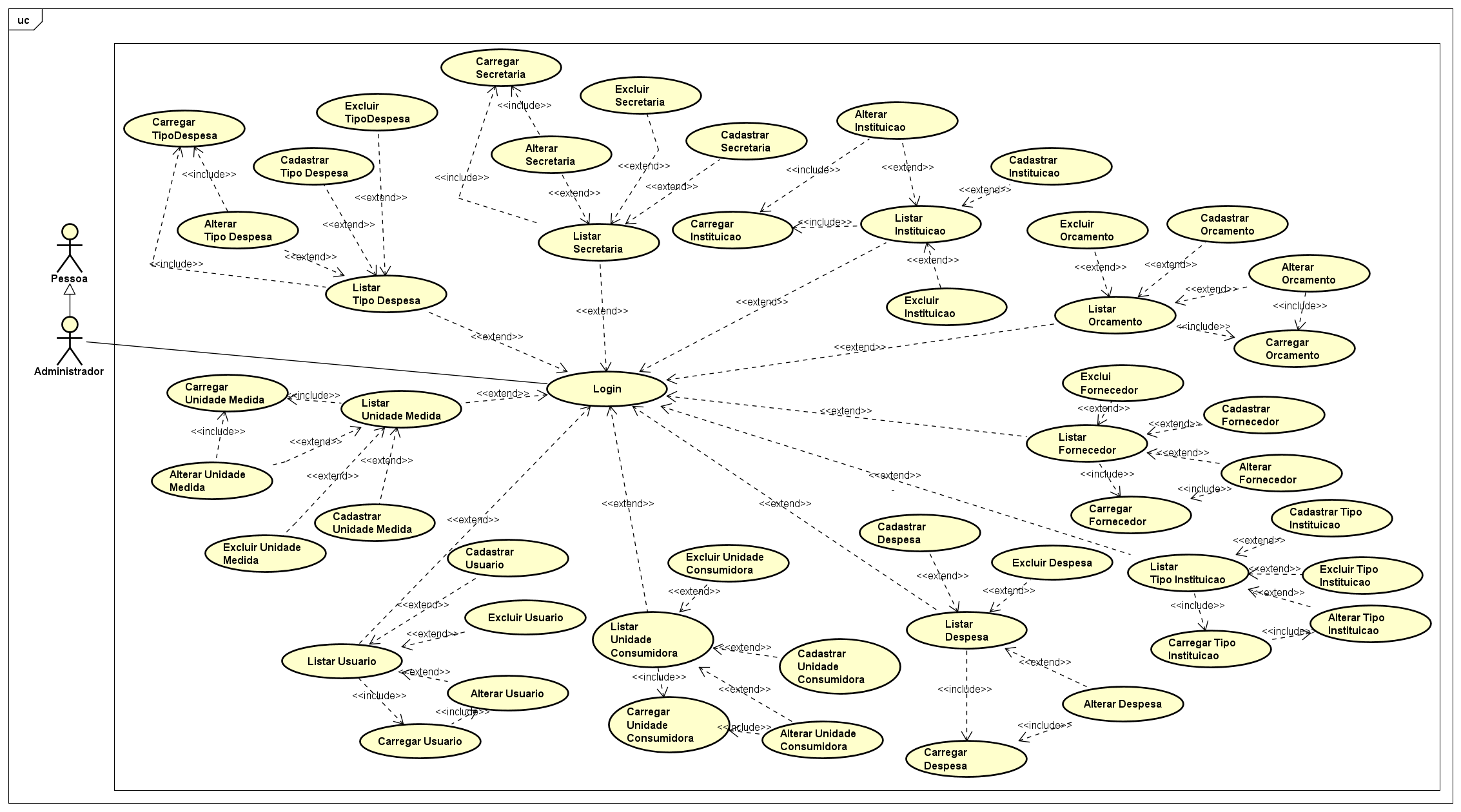
O diagrama de caso de uso geral fornece uma visão abrangente das diversas interações que cada usuário pode realizar no sistema. Essa representação é especialmente valiosa para obter uma compreensão global das funcionalidades disponíveis para cada ator envolvido. Nas Figuras 10, 11 e 12, serão apresentados os casos de uso gerais, destacando as ações que cada ator pode executar no sistema. Essa abordagem oferece uma visão consolidada das capacidades do sistema, facilitando a compreensão das funcionalidades disponíveis para cada usuário.

Apresenta-se na Figura 10, o diagrama de casos de uso da visão do “Administrador” em relação ao sistema. O “Administrador” é um usuário com o nível hierárquico mais elevado e, portanto, possui acesso irrestrito a todas as funcionalidades do sistema. Esse usuário tem a capacidade de carregar, alterar, cadastrar, excluir e listar informações do sistema, além de realizar o acompanhamento das dashboards geradas. A funcionalidade de administração permite, assim, a total gestão de dados e configurações, incluindo o controle sobre outros usuários e suas permissões. Este nível de acesso reflete a responsabilidade do Administrador em manter e operar o sistema de maneira completa e integrada.

Na Figura 11, é apresentado a visão do “Funcionário” em relação ao sistema. O “Funcionário” possui um nível de acesso inferior ao do “Administrador”. Embora tenha permissão para acessar as funcionalidades gerais do sistema e realizar ações como carregar e consultar informações, ele não possui privilégios para cadastrar, alterar, excluir ou listar outros usuários, nem para modificar configurações do sistema. Além disso, o “Funcionário” tem permissão para modificar apenas suas próprias informações e visualizar as dashboards geradas, possibilitando o acompanhamento do desempenho e das métricas relevantes. Essa estrutura de permissões é definida para garantir a integridade dos dados e a segurança do sistema, limitando a interferência de usuários com privilégios menores.

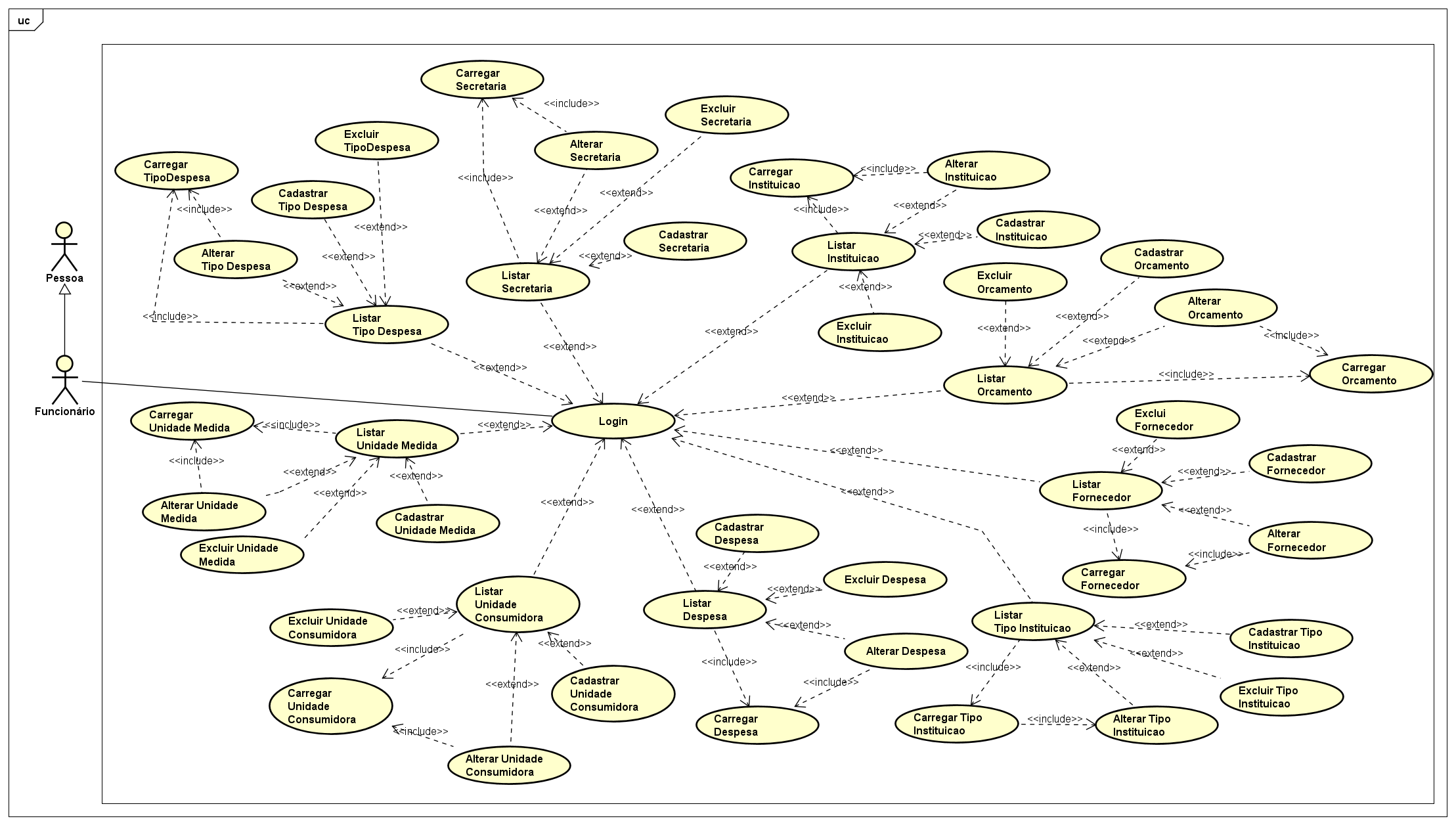
Já na Figura 12, apresenta-se a visão do “Visitante” em relação ao sistema. O “Visitante” é o usuário com o menor nível de acesso, o que implica em uma série de restrições em suas interações com o sistema. O “Visitante” pode somente acessar o sistema para carregar e listar informações relevantes, sem permissão para modificar ou interagir de maneira significativa com os dados ou usuários do sistema. Além disso, o “Visitante” tem a capacidade de visualizar as dashboards geradas, mas sem a possibilidade de realizar ações de modificação ou consulta mais profundas. Este nível de acesso é tipicamente destinado a usuários que necessitam de informações gerais sobre o sistema, mas não têm um papel ativo na sua operação.

Figura 10 – Diagrama de Caso de Uso Geral - Visão Administrador



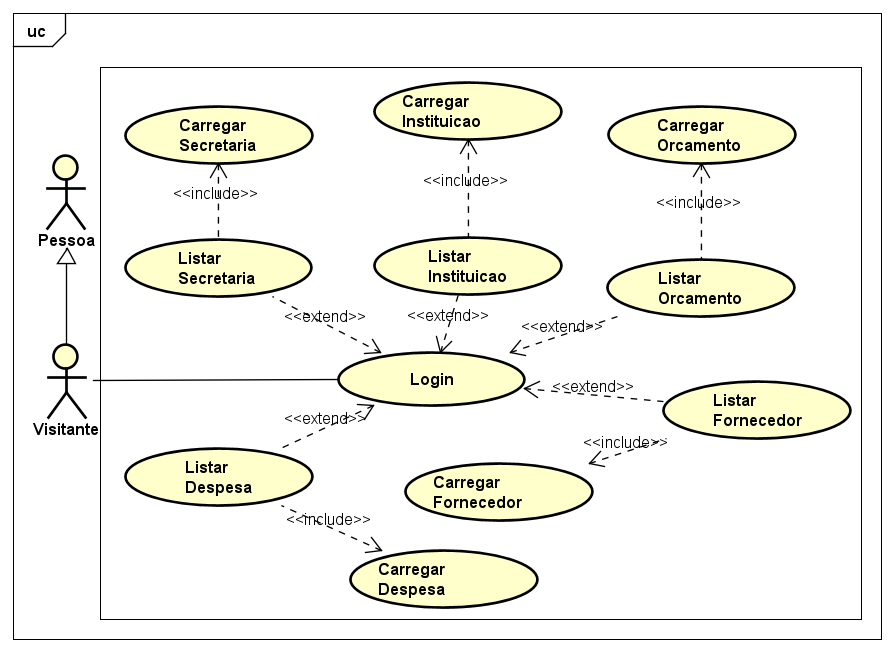
Fonte: Elaborada pelos autores.

Figura 11 – Diagrama de Caso de Uso Geral – Visão Funcionário



Fonte: Elaborada pelos autores.

Figura 12 – Diagrama de Caso de Uso Geral - Visão Visitante



Fonte: Elaborada pelos autores.

## 3.6 Diagrama de Casos de Uso Individuais

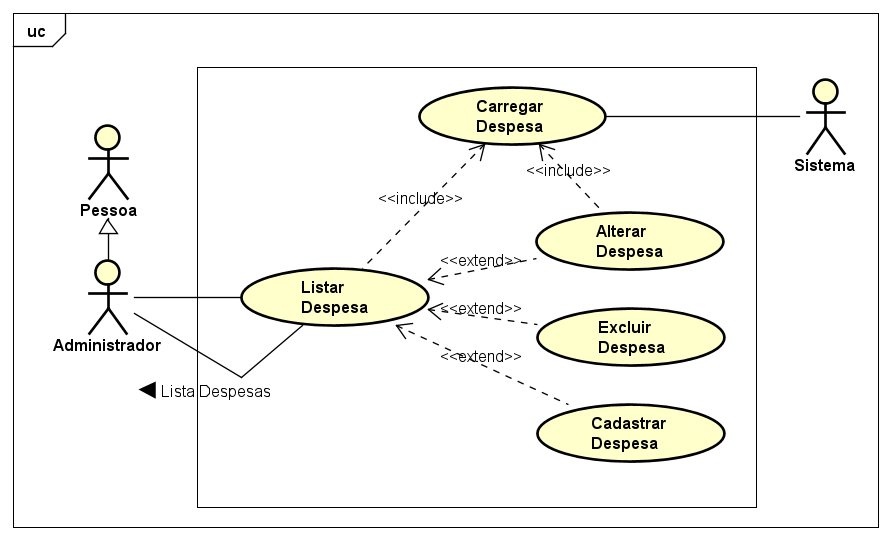
De acordo com Guedes (2018), o diagrama de caso de uso é uma representação gráfica das interações entre um sistema e seus usuários. Seu principal objetivo é descrever as funcionalidades que o sistema oferece de maneira simples e intuitiva, facilitando o entendimento tanto para os desenvolvedores quanto para os usuários. Além disso, o diagrama proporciona uma visão geral de como o sistema funcionará, destacando as interações com os diferentes tipos de usuários que poderão acessá-lo.

Nos tópicos a seguir, serão apresentados alguns dos casos de uso do sistema SIGEDESP, abordando funcionalidades específicas como “Despesa”, “Instituição” e “Secretaria”. Esses casos de uso foram desenvolvidos para fornecer uma compreensão mais detalhada sobre como o sistema opera, ilustrando as interações entre os usuários e o sistema nas diferentes áreas de gerenciamento. A descrição de cada caso de uso visa facilitar a visualização do fluxo de atividades e das tarefas que podem ser executadas dentro do sistema, além de destacar a lógica envolvida em cada operação.

## 3.6.1 Diagrama de Casos de Uso Individuais – Administrador Lista Despesa

Na Figura 13, é apresentado o caso de uso em que o ator “Administrador” interage com o sistema para listar uma “Despesa”. Ao acessar este módulo, o “Administrador” poderá não apenas visualizar a lista de despesas, mas também acessar as funcionalidades de cadastrar, alterar ou excluir uma despesa. O ator “Sistema” será responsável por retornar os dados relativos à “Despesa”, permitindo que o “Administrador” visualize as informações desejadas.

Figura – Diagrama de Caso de Uso Individual – Administrador Lista Despesa



Fonte: Elaborada pelos autores.

No Quadro 20, apresenta-se o fluxo normal e alternativo pelo qual o ator “Administrador” pode passar ao listar uma “Despesa” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, o “Administrador” solicita a lista de despesas, e o ator “Sistema” retorna essa lista para que o “Administrador” possa visualizá-la. No fluxo alternativo, caso ocorra um erro durante a requisição da lista, o sistema retornará uma mensagem de erro, informando o “Administrador” sobre a situação e permitindo que o problema seja corrigido.

Quadro - Documentação - Administrador Lista Despesa

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Listar Despesa |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) liste as despesas. |
| Pré-condições | Administrador logado e autenticado. |
| Pós-condições | Listar as despesas. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Administrador requisita lista de despesa. |  |
|  | 1. Sistema exibe tabela de listagem de despesa. |
| **Fluxo alternativo** | |
|  | 1. Não é possível estabelecer conexão com o Sistema. |
|  | 1. Não é possível estabelecer conexão com a API. |
|  | 1. Não é possível estabelecer conexão com o Banco de Dados. |
|  | 1. Sistema não possui despesa cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 21, é apresentado o fluxo normal e alternativo relacionado ao processo de cadastro de uma “Despesa” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para cadastrar uma nova despesa. O ator “Sistema” retorna um formulário que o “Administrador” deve preencher com as informações necessárias. Após a submissão do formulário, o sistema valida os dados e, caso não haja inconsistências, a despesa é cadastrada com sucesso, sendo confirmada a operação com uma mensagem de sucesso. Caso haja inconsistências nos dados, o fluxo alternativo será acionado, retornando uma mensagem de erro que orienta o “Administrador” a realizar as correções necessárias.

Quadro - Documentação - Administrador Cadastra Despesa

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Cadastrar Despesa |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) adicione novas despesas. |
| Pré-condições | Preencher todos os campos do formulário de inserção. |
| Pós-condições | O sistema fecha a modal e lista as despesas. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita a inserção de uma nova despesa. |  |
|  | 1. Sistema recebe solicitação e mostra modal para inserção dos dados. |
| 1. Usuário informa dados da despesa na modal e clica em “Salvar”. |  |
|  | 1. Sistema valida campos e dados adicionados. |
|  | 1. Sistema retorna mensagem “Cadastrada com Sucesso!”. |
| **Fluxo alternativo** | |
|  | 1. Sistema valida campos e dados |
|  | 1. Sistema identifica inconsistência de dados informados ou campos em branco |
|  | 1. Sistema envia mensagem “Preencha todos os campos” |
|  | 1. Sistema já possui despesa cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 22, é apresentado o fluxo normal e alternativo que o ator “Administrador” pode seguir ao alterar uma “Despesa” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para alterar uma despesa já existente. O sistema retorna um formulário com os dados da despesa selecionada. O “Administrador” então realiza as alterações necessárias e salva as modificações. O sistema valida os dados alterados e, caso sejam válidos, atualiza a lista com a despesa alterada. Caso ocorra algum erro durante o processo de alteração, o fluxo alternativo será acionado, retornando uma mensagem de erro ao “Administrador” e permitindo a correção do problema identificado.

Quadro - Documentação - Administrador Altera Despesa

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Alterar Despesa |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) altere informações referentes as despesas adicionadas anteriormente. |
| Pré-condições | Despesa cadastrado. |
| Pós-condições | O sistema fecha a modal e lista as despesas. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita a alteração de uma despesa. |  |
|  | 1. Sistema recebe solicitação e mostra modal com os dados de despesa. |
| 1. Usuário altera os dados de despesa e seleciona o botão “Alterar” |  |
|  | 1. Sistema valida solicitação e altera dados. |
|  | 1. Sistema envia mensagem “Alterado com sucesso” |
|  | 1. Sistema retorna para a tela de listagem. |
| **Fluxo alternativo** | |
|  | 1. Sistema valida solicitação e altera dados. |
|  | 1. Sistema falha ao conectar-se com o banco de dados |
|  | 1. Sistema envia mensagem “Erro de conexão com o banco de dados” |
|  | 1. Sistema não possui despesa cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 23, é apresentado o fluxo normal e alternativo para o processo de exclusão de uma “Despesa” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para excluir uma despesa já cadastrada. O sistema retorna um formulário de confirmação de exclusão, permitindo que o “Administrador” decida se confirma ou não a exclusão. Caso a exclusão seja confirmada, o sistema válida a operação e remove a despesa da listagem. Se ocorrer algum erro durante o processo de exclusão, o fluxo alternativo será acionado, informando o “Administrador” sobre o erro ocorrido e permitindo a realização das correções necessárias.

Quadro - Documentação - Administrador Exclui Despesa

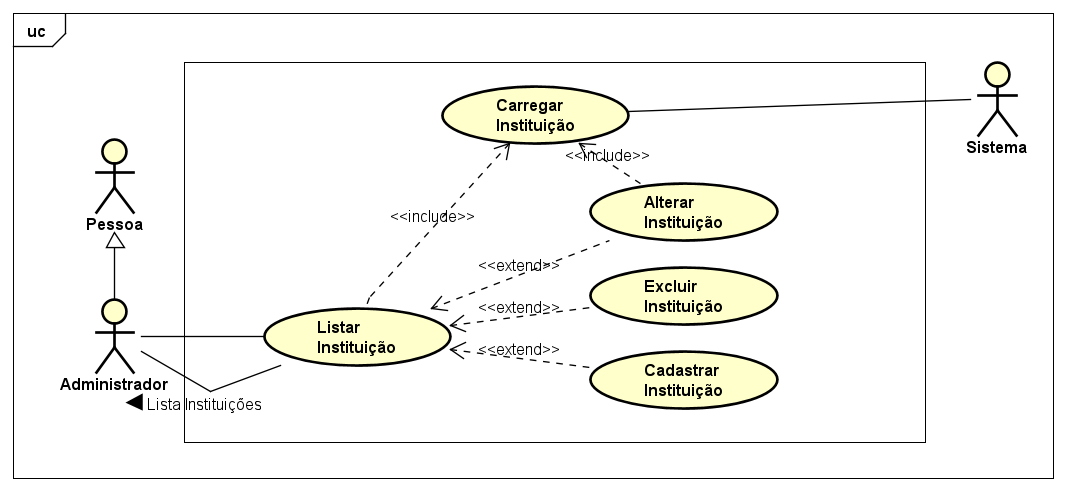
|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Excluir Despesa |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) apague informações referentes as despesas cadastradas. |
| Pré-condições | Despesa cadastrada. |
| Pós-condições | O sistema fecha a modal e lista as despesas. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita exclusão da despesa cadastrada. |  |
|  | 1. Sistema recebe solicitação e mostra modal com os dados da despesa. |
| 1. Usuário seleciona botão “Excluir”. |  |
|  | 1. Sistema valida solicitação. |
|  | 1. Sistema envia mensagem “Excluído com sucesso” |
|  | 1. Sistema retorna para a tela de listagem. |
| **Fluxo alternativo** | |
|  | 1. Sistema lista solicitação. |
|  | 1. Sistema falha ao conectar-se com o banco de dados. |
|  | 1. Sistema envia mensagem “Erro de conexão com o banco de dados”. |
|  | 1. Sistema não possui despesa cadastrada. |

Fonte: Elaborada pelos autores.

## 3.6.2 Diagrama de Casos de Uso Individuais – Administrador Lista Instituição

Na Figura 14, é apresentado o caso de uso em que o ator “Administrador” interage com o sistema para listar uma “Instituição”. Ao acessar este módulo, o “Administrador” poderá não apenas visualizar a lista de instituições, mas também acessar as funcionalidades de cadastrar, alterar ou excluir uma instituição. O ator “Sistema” será responsável por retornar os dados relativos à “Instituição”, permitindo que o “Administrador” visualize as informações desejadas.

Figura – Diagrama de Caso de Uso Individual – Administrador Lista Instituição

Fonte: Elaborada pelos autores.

O Quadro 24 apresenta o fluxo normal e alternativo pelo qual o ator “Administrador” pode passar ao listar uma “Instituição” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, o “Administrador” solicita a lista de instituições, e o ator “Sistema” retorna essa lista para que o “Administrador” possa visualizá-la. No fluxo alternativo, caso ocorra um erro durante a requisição da lista, o sistema retornará uma mensagem de erro, informando o “Administrador” sobre a situação e permitindo que o problema seja corrigido.

Quadro - Documentação - Administrador Lista Instituição

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Listar Instituição |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) liste as instituições. |
| Pré-condições | Administrador logado e autenticado. |
| Pós-condições | Lista as instituições. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Administrador requisita lista de instituições. |  |
|  | 1. Sistema exibe tabela de listagem de instituições. |
| **Fluxo alternativo** | |
|  | 1. Não é possível estabelecer conexão com o Sistema. |
|  | 1. Não é possível estabelecer conexão com a API. |
|  | 1. Não é possível estabelecer conexão com o Banco de Dados. |
|  | 1. Sistema não possui instituição cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 25, é apresentado o fluxo normal e alternativo relacionado ao processo de cadastro de uma “Instituição” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para cadastrar uma nova instituição. O ator “Sistema” retorna um formulário que o “Administrador” deve preencher com as informações necessárias. Após a submissão do formulário, o sistema valida os dados e, caso não haja inconsistências, a instituição é cadastrada com sucesso, sendo confirmada a operação com uma mensagem de sucesso. Caso haja inconsistências nos dados, o fluxo alternativo será acionado, retornando uma mensagem de erro que orienta o “Administrador” a realizar as correções necessárias.

Quadro - Documentação - Administrador Cadastra Instituição

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Cadastrar Instituição |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) adicione novas instituições. |
| Pré-condições | Preencher todos os campos do formulário de inserção. |
| Pós-condições | O sistema fecha a modal e lista as instituições. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita a inserção de uma nova instituição. |  |
|  | 1. Sistema recebe solicitação e mostra modal para inserção dos dados. |
| 1. Usuário informa dados da instituição na modal e clica em “Salvar”. |  |
|  | 1. Sistema valida campos e dados adicionados |
|  | 1. Sistema retorna mensagem “Cadastrado com Sucesso!”. |
| **Fluxo alternativo** | |
|  | 1. Sistema valida campos e dados |
|  | 1. Sistema identifica inconsistência de dados informados ou campos em branco |
|  | 1. Sistema envia mensagem “Preencha todos os campos”. |
|  | 1. Sistema já possui instituição cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 26, é apresentado o fluxo normal e alternativo que o ator “Administrador” pode seguir ao alterar uma “Instituição” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para alterar uma instituição já existente. O sistema retorna um formulário com os dados da instituição selecionada. O “Administrador” então realiza as alterações necessárias e salva as modificações. O sistema valida os dados alterados e, caso sejam válidos, atualiza a lista com a instituição alterada. Caso ocorra algum erro durante o processo de alteração, o fluxo alternativo será acionado, retornando uma mensagem de erro ao “Administrador” e permitindo a correção do problema identificado.

Quadro - Documentação - Administrador Altera Instituição

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Alterar Instituição |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) altere informações referentes as Instituição adicionadas anteriormente. |
| Pré-condições | Instituição cadastrada. |
| Pós-condições | O sistema fecha a modal e lista as instituições. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita a alteração de instituição. |  |
|  | 1. Sistema recebe solicitação e mostra modal com os dados de instituição. |
| 1. Usuário altera os dados de instituição e seleciona o botão “Alterar” |  |
|  | 1. Sistema valida solicitação e altera dados. |
|  | 1. Sistema envia mensagem “Alterado com sucesso” |
|  | 1. Sistema retorna para a tela de listagem. |
| **Fluxo alternativo** | |
|  | 1. Sistema valida solicitação e altera dados. |
|  | 1. Sistema falha ao conectar-se com o banco de dados. |
|  | 1. Sistema envia mensagem “Erro de conexão com o banco de dados”. |
|  | 1. Sistema não possui instituição cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 27, é apresentado o fluxo normal e alternativo para o processo de exclusão de uma “Instituição” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para excluir uma instituição já cadastrada. O sistema retorna um formulário de confirmação de exclusão, permitindo que o “Administrador” decida se confirma ou não a exclusão. Caso a exclusão seja confirmada, o sistema válida a operação e remove a instituição da listagem. Se ocorrer algum erro durante o processo de exclusão, o fluxo alternativo será acionado, informando o “Administrador” sobre o erro ocorrido e permitindo a realização das correções necessárias.

Quadro - Documentação - Administrador Exclui Instituição

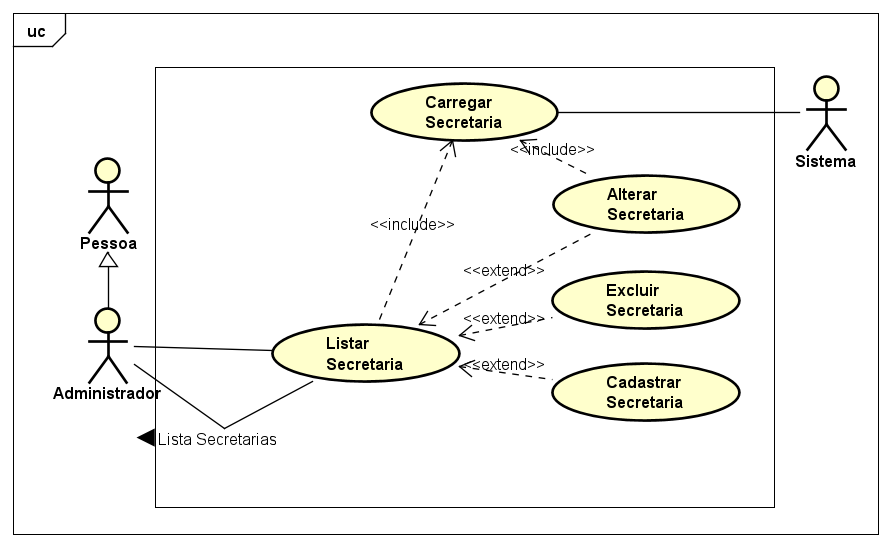
|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Excluir Instituição |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) apague informações referentes as instituições cadastradas. |
| Pré-condições | Instituição cadastrada |
| Pós-condições | O sistema fecha a modal e lista as instituições. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita exclusão da instituição cadastrada. |  |
|  | 1. Sistema recebe solicitação e mostra modal com os dados de instituição. |
| 1. Usuário seleciona botão “Excluir”. |  |
|  | 1. Sistema valida solicitação. |
|  | 1. Sistema envia mensagem “Excluído com sucesso”. |
|  | 1. Sistema retorna para a tela de listagem. |
| **Fluxo alternativo** | |
|  | 1. Sistema lista solicitação. |
|  | 1. Sistema falha ao conectar-se com o banco de dados. |
|  | 1. Sistema envia mensagem “Erro de conexão com o banco de dados”. |
|  | 1. Sistema não possui instituição cadastrada. |

Fonte: Elaborada pelos autores.

## 3.6.3 Diagrama de Casos de Uso Individuais – Administrador Lista Secretaria

É apresentado, na Figura 15, o caso de uso em que o ator “Administrador” interage com o sistema para listar uma “Secretaria”. Ao acessar este módulo, o “Administrador” poderá não apenas visualizar a lista de secretarias, mas também acessar as funcionalidades de cadastrar, alterar ou excluir uma “Secretaria”. O ator “Sistema” será responsável por retornar os dados relativos à secretaria, permitindo que o “Administrador” visualize as informações desejadas.

Figura 15 – Diagrama de Caso de Uso Individual – Administrador Lista Secretaria



Fonte: Elaborada pelos autores.

O Quadro 28 apresenta o fluxo normal e alternativo pelo qual o ator “Administrador” pode passar ao listar uma “Secretaria” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, o “Administrador” solicita a lista de secretarias, e o ator “Sistema” retorna essa lista para que o “Administrador” possa visualizá-la. No fluxo alternativo, caso ocorra um erro durante a requisição da lista, o sistema retornará uma mensagem de erro, informando o “Administrador” sobre a situação e permitindo que o problema seja corrigido.

Quadro - Documentação - Administrador Lista Secretaria

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Listar Secretaria |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) liste as secretarias. |
| Pré-condições | Administrador logado e autenticado. |
| Pós-condições | Lista as secretarias. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Administrador requisita lista de secretarias. |  |
|  | 1. Sistema exibe tabela de listagem de secretarias. |
| **Fluxo alternativo** | |
|  | 1. Não é possível estabelecer conexão com o Sistema. |
|  | 1. Não é possível estabelecer conexão com a API. |
|  | 1. Não é possível estabelecer conexão com o Banco de Dados. |
|  | 1. Sistema não possui secretaria cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 29, é apresentado o fluxo normal e alternativo relacionado ao processo de cadastro de uma “Secretaria” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para cadastrar uma nova secretaria. O ator “Sistema” retorna um formulário que o “Administrador” deve preencher com as informações necessárias. Após a submissão do formulário, o sistema valida os dados e, caso não haja inconsistências, a secretaria é cadastrada com sucesso, sendo confirmada a operação com uma mensagem de sucesso. Caso haja inconsistências nos dados, o fluxo alternativo será acionado, retornando uma mensagem de erro que orienta o “Administrador” a realizar as correções necessárias.

Quadro - Documentação - Administrador Cadastra Secretaria

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Cadastrar Secretaria |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) adicione novas secretarias. |
| Pré-condições | Preencher todos os campos do formulário de inserção. |
| Pós-condições | O sistema fecha a modal e lista as secretarias. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita a inserção de uma nova secretaria. |  |
|  | 1. Sistema recebe solicitação e mostra modal para inserção dos dados. |
| 1. Usuário informa dados da secretaria na modal e clica em “Salvar”. |  |
|  | 1. Sistema valida campos e dados adicionados |
|  | 1. Sistema retorna mensagem “Cadastrado com Sucesso!”. |
| **Fluxo alternativo** | |
|  | 1. Sistema valida campos e dados |
|  | 1. Sistema identifica inconsistência de dados informados ou campos em branco |
|  | 1. Sistema envia mensagem “Preencha todos os campos”. |
|  | 1. Sistema já possui secretaria cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 30, é apresentado o fluxo normal e alternativo que o ator “Administrador” pode seguir ao alterar uma “Secretaria” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para alterar uma secretaria já existente. O sistema retorna um formulário com os dados da secretaria selecionada. O “Administrador” então realiza as alterações necessárias e salva as modificações. O sistema valida os dados alterados e, caso sejam válidos, atualiza a lista com a secretaria alterada. Caso ocorra algum erro durante o processo de alteração, o fluxo alternativo será acionado, retornando uma mensagem de erro ao “Administrador” e permitindo a correção do problema identificado.

Quadro - Documentação - Administrador Altera Secretaria

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Alterar Secretaria |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) altere informações referentes as Secretarias adicionadas anteriormente. |
| Pré-condições | Secretaria cadastrada. |
| Pós-condições | O sistema fecha a modal e lista as secretarias. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita a alteração de secretaria. |  |
|  | 1. Sistema recebe solicitação e mostra modal com os dados de secretaria. |
| 1. Usuário altera os dados de secretaria e seleciona o botão “Alterar” |  |
|  | 1. Sistema valida solicitação e altera dados. |
|  | 1. Sistema envia mensagem “Alterado com sucesso” |
|  | 1. Sistema retorna para a tela de listagem. |
| **Fluxo alternativo** | |
|  | 1. Sistema valida solicitação e altera dados. |
|  | 1. Sistema falha ao conectar-se com o banco de dados. |
|  | 1. Sistema envia mensagem “Erro de conexão com o banco de dados”. |
|  | 1. Sistema não possui secretaria cadastrada. |

Fonte: Elaborada pelos autores.

No Quadro 31, é apresentado o fluxo normal e alternativo para o processo de exclusão de uma “Secretaria” no sistema. Para iniciar qualquer um dos fluxos, o “Administrador” deve estar logado no sistema. No fluxo normal, ao acessar a página de listagem, o “Administrador” seleciona a opção para excluir uma secretaria já cadastrada. O sistema retorna um formulário de confirmação de exclusão, permitindo que o “Administrador” decida se confirma ou não a exclusão. Caso a exclusão seja confirmada, o sistema válida a operação e remove a secretaria da listagem. Se ocorrer algum erro durante o processo de exclusão, o fluxo alternativo será acionado, informando o “Administrador” sobre o erro ocorrido e permitindo a realização das correções necessárias.

Quadro - Documentação - Administrador Exclui Secretaria

|  |  |
| --- | --- |
| **Documentação** | |
| Nome do caso de uso | Excluir Secretaria |
| Ator principal | Usuário (Administrador) |
| Resumo | Esse caso de uso descreve as ações necessárias para que o usuário (Administrador) apague informações referentes as secretarias cadastradas. |
| Pré-condições | Secretaria cadastrada. |
| Pós-condições | O sistema fecha a modal e lista as secretarias. |
| **Fluxo normal** | |
| Ações do ator | Ações do sistema |
| 1. Usuário solicita exclusão da secretaria cadastrada. |  |
|  | 1. Sistema recebe solicitação e mostra modal com os dados da secretaria. |
| 1. Usuário seleciona botão “Excluir”. |  |
|  | 1. Sistema valida solicitação. |
|  | 1. Sistema envia mensagem “Excluído com sucesso”. |
|  | 1. Sistema retorna para a tela de listagem. |
| **Fluxo alternativo** | |
|  | 1. Sistema lista solicitação. |
|  | 1. Sistema falha ao conectar-se com o banco de dados. |
|  | 1. Sistema envia mensagem “Erro de conexão com o banco de dados”. |
|  | 1. Sistema não possui secretaria cadastrada. |

Fonte: Elaborada pelos autores.

## 3.7 Diagrama de Sequência

Guedes (2018) afirma que o diagrama de sequência é uma representação gráfica da interação entre objetos em um sistema. Ele mostra como os objetos trocam mensagens entre si, na ordem em que isso acontece. O diagrama de sequência é baseado em um caso de uso, que é uma descrição das funcionalidades que um sistema fornece aos usuários. O caso de uso fornece uma visão geral das interações entre os usuários e o sistema. O diagrama de sequência também se baseia no diagrama de classes, que é uma representação dos objetos e suas relações em um sistema. O diagrama de classes fornece os nomes dos objetos que participam das interações no diagrama de sequência.

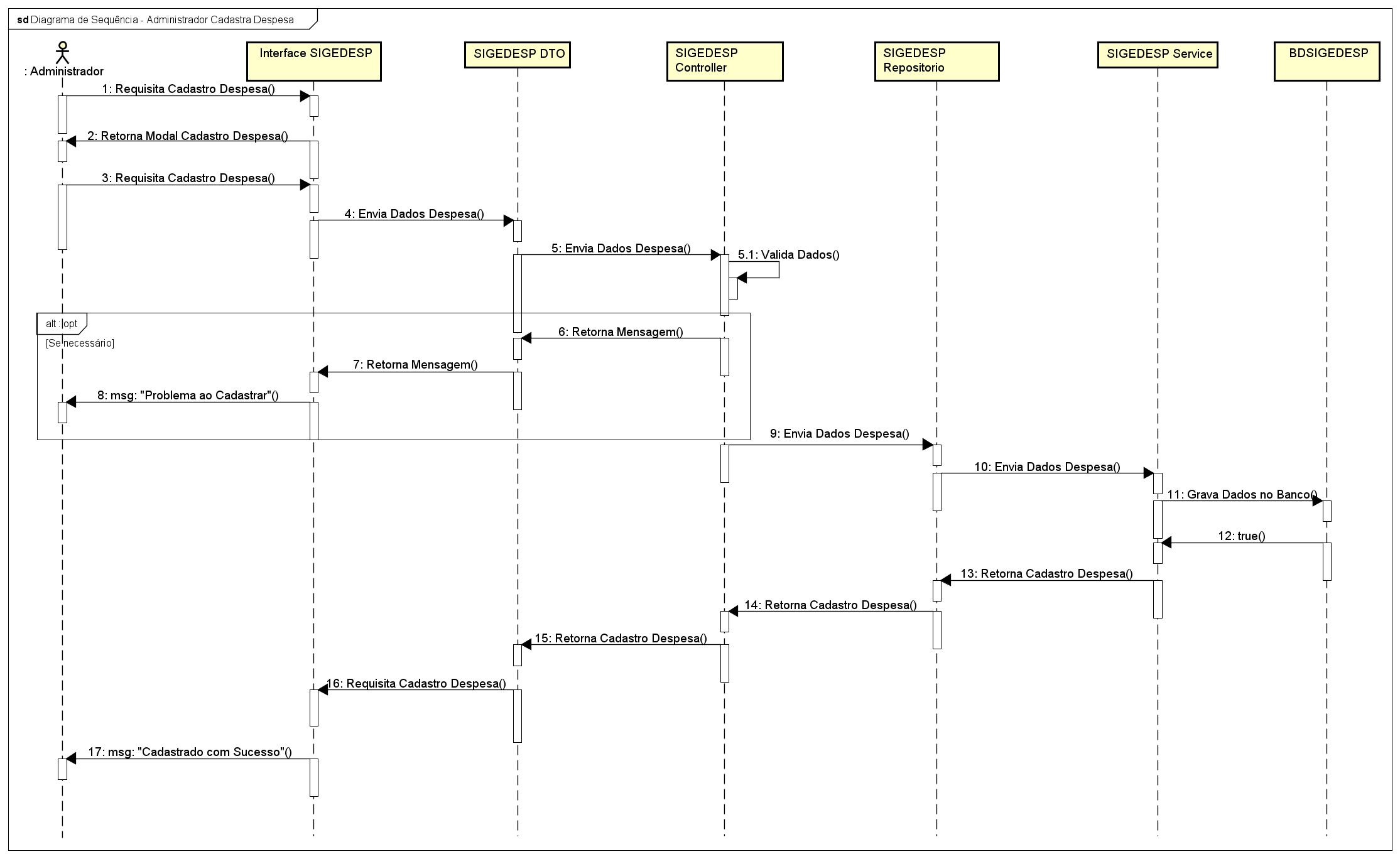
A seguir, serão apresentados alguns diagramas de sequência, que foram desenvolvidos para ilustrar o funcionamento do sistema SIGEDESP. Esses diagramas (Figuras 16 a 18) têm como objetivo fornecer uma visão de alguns dos processos e interações entre os diferentes componentes do sistema, ajudando a entender como as operações são executadas de forma sequencial. Através dessas representações, será possível compreender melhor a dinâmica entre os usuários e o sistema, além de visualizar as trocas de mensagens e as dependências entre os diversos módulos envolvidos no fluxo de execução das funcionalidades.

Apresenta-se na Figura 16, o processo de cadastro de “Despesa”. O fluxo inicia com a solicitação de cadastro pelo usuário “Administrador” e atravessa diversas camadas do sistema. Primeiramente, os dados são encapsulados na *DTO*[[2]](#footnote-2) e submetidos a uma validação na camada Controller. Caso a validação seja bem-sucedida, o fluxo prossegue, passando pelas camadas Repositório e *Service*, antes de serem persistidos no Banco de Dados. Concluído o cadastro, os dados são retornados à interface do usuário, finalizando o processo com o envio da lista de despesas. No entanto, se ocorrer uma falha na validação dos dados, o fluxo é interrompido e o erro é reportado ao usuário responsável pelo cadastro da “Despesa”.

Na Figura 17 é apresentado o processo de alteração de “Despesa”. O fluxo tem início com a requisição de alteração pelo usuário “Administrador” e percorre as camadas *DTO*, *Controller* e Repositório. Na camada Repositório, os dados já cadastrados são carregados e uma modal de alteração é enviada de volta ao usuário. Caso ocorra uma falha nesta etapa, o erro é reportado ao usuário. Se a modal de alteração for apresentada com sucesso, o usuário realiza as alterações nos dados de despesa. Essas alterações, então, atravessam as camadas até chegarem ao Banco de Dados, onde são gravadas. Se o processo for bem-sucedido, uma confirmação é enviada ao usuário informando que a alteração foi realizada. No entanto, se surgir um problema durante a gravação dos dados alterados no banco de dados, o usuário é notificado sobre o ocorrido.

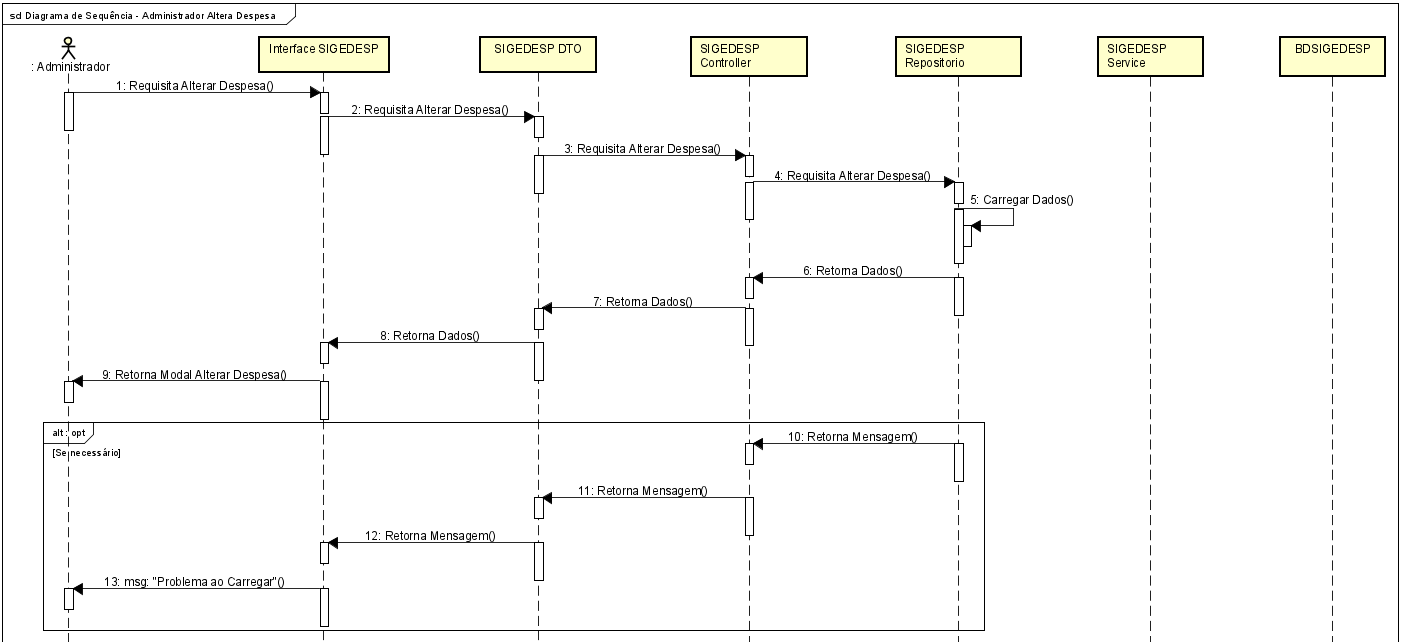
Conforme apresentado na Figura 18, temos o processo de exclusão de “Despesa”. O fluxo tem início com a requisição de exclusão pelo usuário “Administrador” e percorre as camadas *DTO*, *Controller* e Repositório. Na camada Repositório, os dados já cadastrados são carregados e uma modal de exclusão é enviada de volta ao usuário. Caso ocorra uma falha nesta etapa, o erro é reportado ao usuário. Se a modal de exclusão for apresentada com sucesso, o usuário realiza a exclusão da despesa. Essa exclusão atravessa as camadas até chegar ao Banco de Dados, onde é efetivamente excluída. Se o processo for bem-sucedido, uma confirmação é enviada ao usuário informando que a exclusão foi realizada. No entanto, se surgir um problema durante a exclusão dos dados no banco de dados, o usuário é notificado sobre o ocorrido.

Figura – Diagrama de Sequência – Administrador Cadastra Despesa



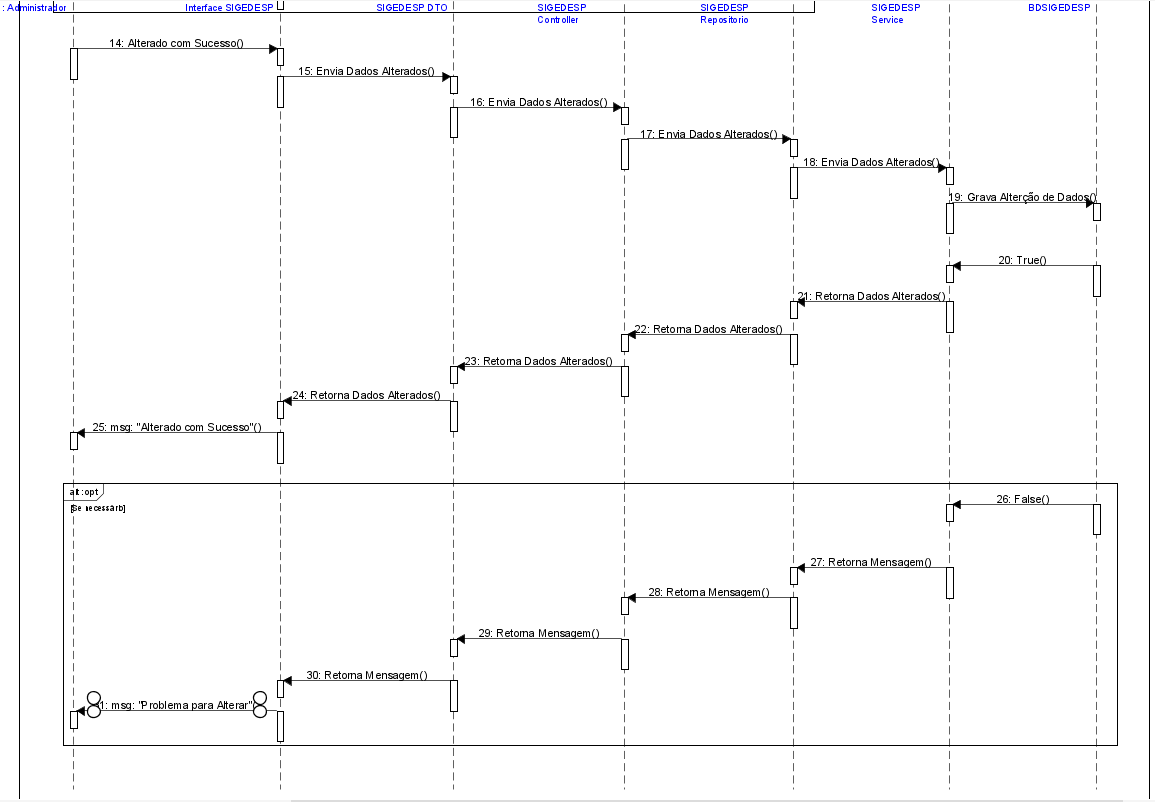
Fonte: Elaborada pelos autores.

Figura – Diagrama de Sequência – Administrador Altera Despesa



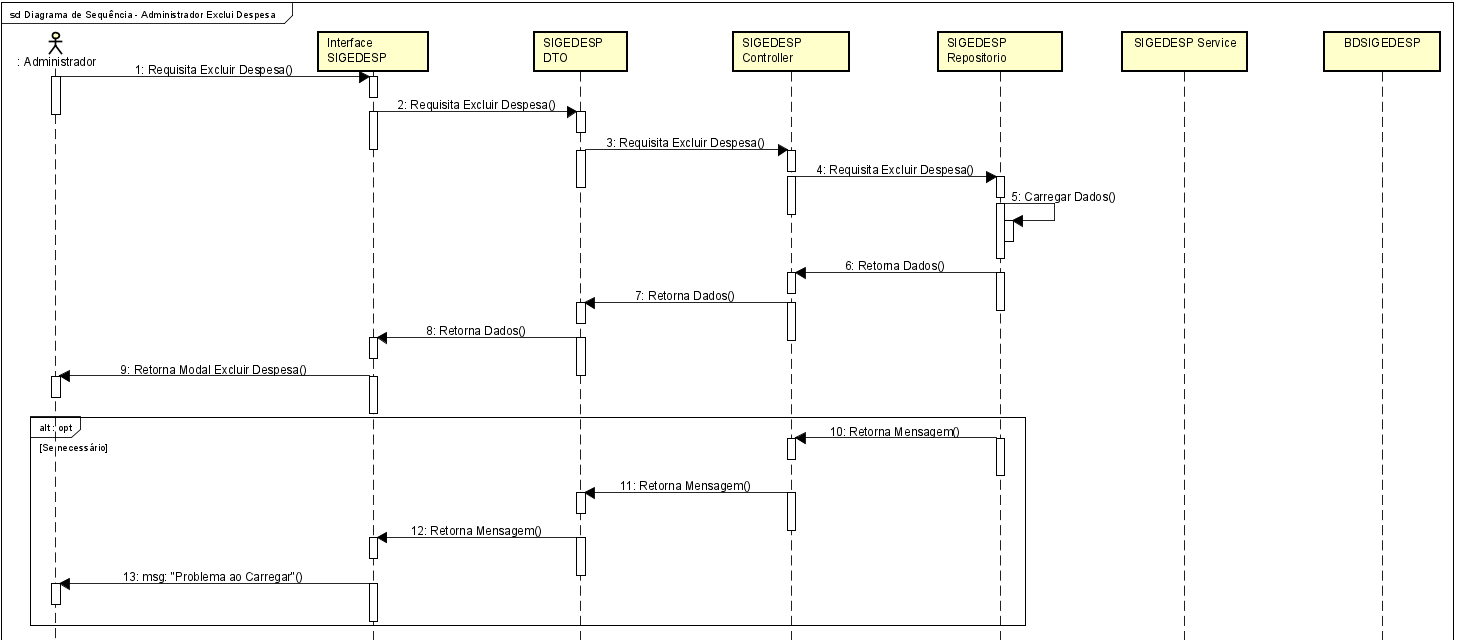
Fonte: Elaborada pelos autores.

Figura – Diagrama de Sequência – Administrador Altera Despesa (continuação)



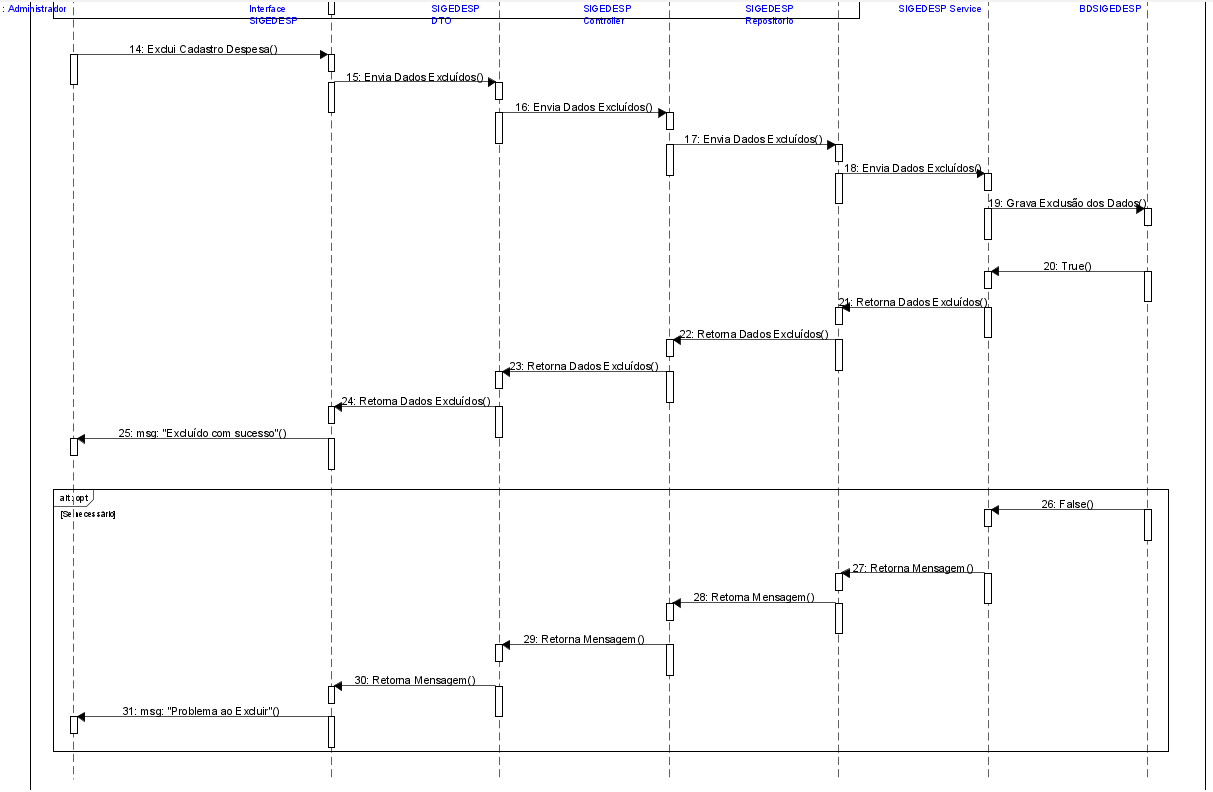
Fonte: Elaborada pelos autores.

Figura – Diagrama de Sequência - Administrador Exclui Despesa



Fonte: Elaborada pelos autores.

**Figura 18 –** Diagrama de Sequência - Administrador Exclui Despesa (continuação)



Fonte: Elaborada pelos autores.

## 3.8 Diagrama de Atividade

Antigamente, o diagrama de atividade era considerado um caso especial do diagrama de máquina de estado, que será abordado no próximo tópico. Atualmente, esse diagrama é reconhecido como uma entidade independente. Trata-se do diagrama com "mais ênfase em nível de algoritmo da UML e provavelmente um dos mais detalhistas" (Guedes, 2018, p. 390).

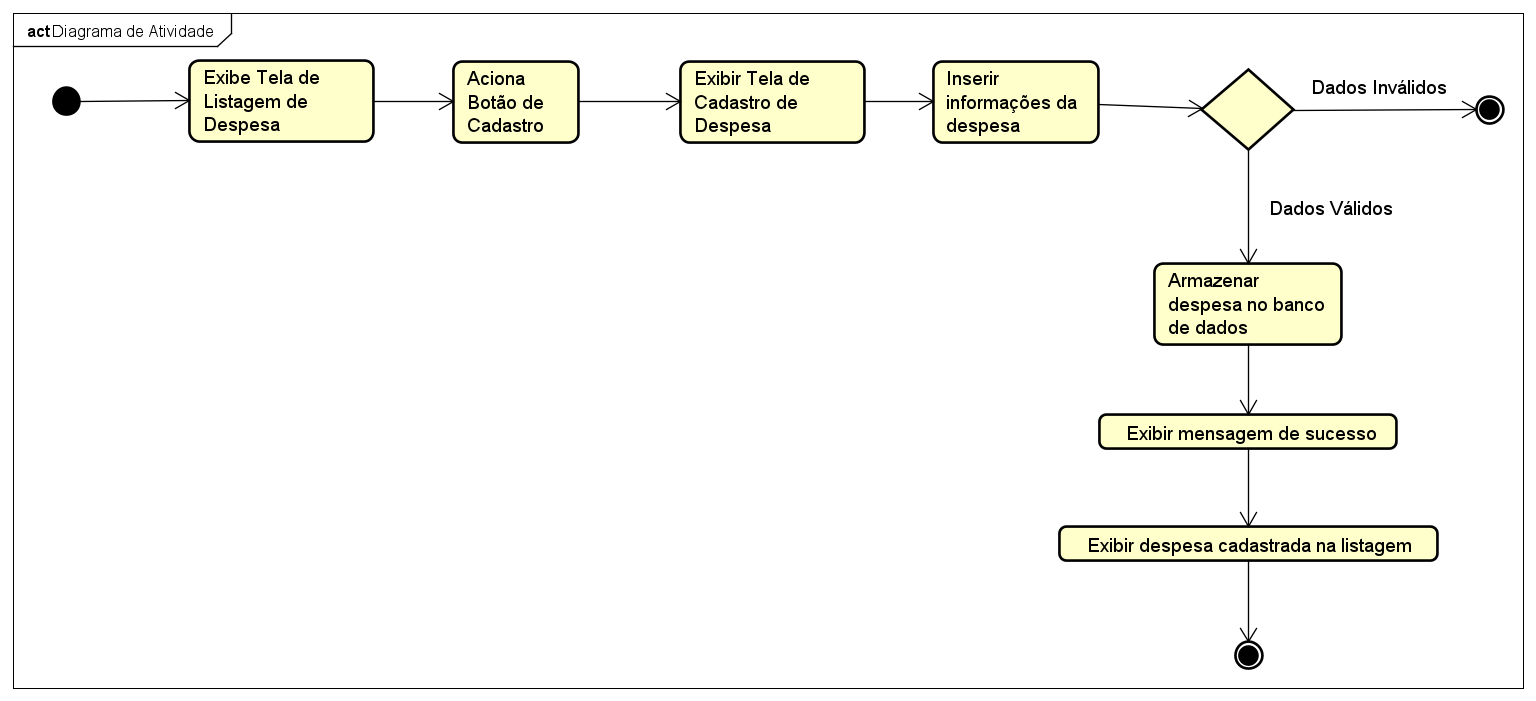
Esse diagrama é empregado na modelagem de atividades, que podem ser entendidas como métodos, algoritmos ou processos completos. Conforme afirmado, "uma atividade é composta por um conjunto de ações, ou seja, os passos necessários para que a atividade seja concluída" (Guedes, 2011, p. 391).

Figura 19 apresenta o fluxo de atividades do processo de cadastro de uma nova “Despesa” no sistema. O processo tem início quando o usuário, “Administrador”, acessa a tela de listagem de despesas e aciona o botão de cadastro de despesa. O sistema, então, exibe a tela de cadastro, onde o usuário deve inserir as informações solicitadas, como valor, categoria e data da despesa.

Após a inserção dos dados, o fluxo segue para um nó de decisão, onde o sistema valida as informações fornecidas. Caso os dados sejam inválidos, o processo é interrompido e uma mensagem de erro é exibida ao usuário, indicando que os dados precisam ser corrigidos. Se, por outro lado, os dados forem válidos, o fluxo prossegue para a etapa de armazenamento da despesa no banco de dados (Figura 19).

Na etapa subsequente, o sistema retorna uma mensagem de sucesso ao usuário, confirmando que a despesa foi cadastrada corretamente. Por fim, a despesa recém-cadastrada é exibida na tela de listagem de despesas, e o processo é encerrado (Figura 19).

Figura – Diagrama de Atividade – Cadastro de Despesa



Fonte: Elaborada pelos autores.

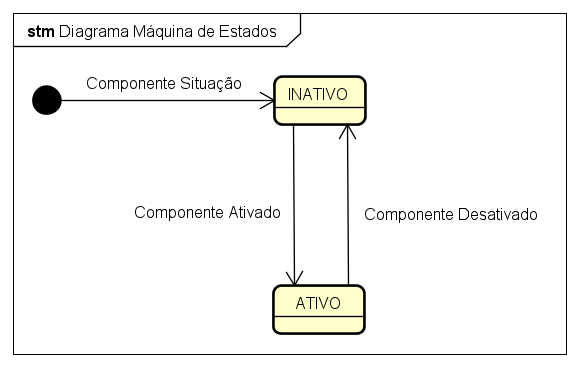
## 3.9 Diagrama de Máquina de Estados

Guedes (2018) descreve que o diagrama de máquina de estados ilustra o comportamento de um elemento por meio de um conjunto finito de transições. Este processo é conhecido como máquina de estado comportamental. Assim como o diagrama de sequência, este diagrama pode ser fundamentado em um caso de uso, mas também pode ser empregado para monitorar o estado de outros elementos.

Entre os componentes principais deste diagrama, destacam-se o estado, que representa a condição em que um elemento se encontra durante o processo, e as transições, que correspondem a eventos que provocam mudanças no estado de um elemento, resultando assim em um novo estado (Guedes, 2018).

Conforme apresentado na Figura 20, o diagrama de máquina de estados do componente “Situação” inicia-se no estado inativo. Para que ocorra a transição do estado inativo para o estado ativo, é necessário que o componente “Situação” sofra a mudança do estado componente desativado para o estado componente ativado. Da mesma forma, a transição do estado ativo para o estado inativo ocorre quando o componente “Situação” passa do estado componente ativado para o estado componente desativado. Dessa forma, um objeto do sistema pode ser classificado como ativo ou inativo, dependendo do seu estado atual.

Figura – Diagrama de Máquina de Estado – Componente Situação



Fonte: Elaborada pelos autores.

# DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX)

Nesta seção, será apresentada a pesquisa realizada para o planejamento da parte visual do projeto SIGEDESP, com foco na Experiência do Usuário (*UX*). O conceito de *UX* engloba o conjunto de percepções e emoções que um usuário experimenta ao interagir com um produto, seja ao utilizá-lo, observá-lo, abri-lo ou fechá-lo. Em outras palavras, a experiência do usuário se refere à qualidade da interação, com ênfase na satisfação, conforto e facilidade durante o uso do produto (Rogers; Sharp; Preece, 2013). Esse conceito não se aplica apenas a produtos físicos, mas também a interfaces digitais, como é o caso do sistema web SIGEDESP (Unger; Chandler, 2009).

## Descrição de Cenário

Um aspecto fundamental no desenvolvimento de *UX* em projetos é a criação de cenários. Segundo Rogers, Sharp e Preece (2013), cenários são representações narrativas que descrevem atividades ou tarefas humanas em forma de história. Esses cenários desempenham um papel crucial na definição dos contextos, necessidades e requisitos dos usuários, servindo como base para a construção de soluções mais adequadas e eficazes. São amplamente utilizados no design conceitual, principalmente para ilustrar situações hipotéticas ou propostas, as quais ajudam a guiar o processo de design e a garantir que o produto final atenda às expectativas e demandas do público-alvo.

Para o desenvolvimento deste projeto, foram elaborados dois cenários com o objetivo de simular situações específicas de gerenciamento de despesas públicas. Esses cenários oferecem narrativas que ilustram como os usuários poderão interagir com o sistema.

Para apresentar um dos cenários do sistema, o Quadro 32, descreve o processo de registro de despesas realizado por um funcionário da prefeitura. Este cenário foi desenvolvido com o objetivo de simular a interação do usuário com a plataforma, destacando as etapas e o fluxo de trabalho necessários para a inserção de informações fiscais. Através do quadro, é possível visualizar o contexto em que o funcionário, autenticado no sistema, preenche os dados de uma nova despesa mensal, assegurando que o processo seja eficiente e livre de erros. A validação do sistema e o retorno do funcionário à sua rotina com a despesa registrada ilustram a experiência de uso e a efetividade da interface proposta.

Quadro – Cenário - Registro de Despesas por um Funcionário da Prefeitura

Em um tranquilo escritório da prefeitura, um funcionário autenticado no sistema de gerenciamento de despesas se encontra diante da tarefa de registrar uma nova despesa mensal. Na atmosfera organizada e focada do ambiente de trabalho, o funcionário se sente determinado a preencher os detalhes precisos da despesa, inserindo cuidadosamente informações como data, categoria, valor e descrição. Ao clicar em "Salvar", uma sensação de realização permeia o escritório, pois o sistema valida os dados, confirmando o registro e redirecionando o funcionário de volta à sua rotina com a despesa devidamente documentada.

Fonte: Elaborada pelos autores.

O Quadro 33 apresenta o cenário em que um funcionário da “Secretaria da Fazenda” utiliza o sistema para analisar as informações financeiras geradas pelas dashboards. O quadro descreve como o sistema facilita o monitoramento das despesas e o acompanhamento do desempenho orçamentário, permitindo que o usuário gere relatórios sobre a gestão das finanças públicas.

Quadro – Cenário - Análise de Dashboards pelo Funcionário da Secretaria da Fazenda

No ambiente da Secretaria da Fazenda, um funcionário autenticado acessa o sistema de gerenciamento de despesas para analisar as dashboards financeiras. O sistema exibe uma visão geral das despesas a pagar, pendentes e vencidas, além de fornecer um gráfico com a porcentagem do orçamento utilizado, destacando áreas críticas. O funcionário pode filtrar os dados conforme necessário para obter informações mais detalhadas e, com base nessas análises, elabora um relatório sobre os gastos da prefeitura, incluindo as despesas das organizações sob sua responsabilidade.

Fonte: Elaborada pelos autores.

## Descrição de Personas

No processo de desenvolvimento de sistemas, o uso de personas é uma prática importante para assegurar que as necessidades e características do público-alvo sejam contempladas. Segundo Unger e Chandler (2009), personas são representações detalhadas de usuários-alvo que ajudam a compreender de maneira prática quem utiliza o sistema e como ele é utilizado. Elas são criadas a partir de dados reais sobre os usuários e são úteis para orientar decisões de design e funcionalidades do sistema. Para a criação de personas efetivas, é necessário identificar o público e reunir informações suficientes que revelem o comportamento, as motivações e as expectativas do usuário.

As personas apresentadas para o projeto SIGEDESP foram criadas com base em informações detalhadas sobre os usuários potenciais. Essas personas incluem dados como foto, nome, idade, localização e ocupação, além de objetivos específicos de uso do sistema (Unger; Chandler, 2009). Tais dados foram utilizados para representar com precisão o perfil dos usuários e para guiar o desenvolvimento do sistema de acordo com as necessidades específicas de cada persona.

A Figura 21 apresenta a persona "Leonor Alves Martins", uma contadora municipal experiente, responsável pelo monitoramento e pela análise detalhada das despesas públicas. Com mais de 15 anos de atuação, “Leonor” desempenha um papel essencial na gestão financeira da prefeitura, assegurando a precisão e a integridade das contas públicas.

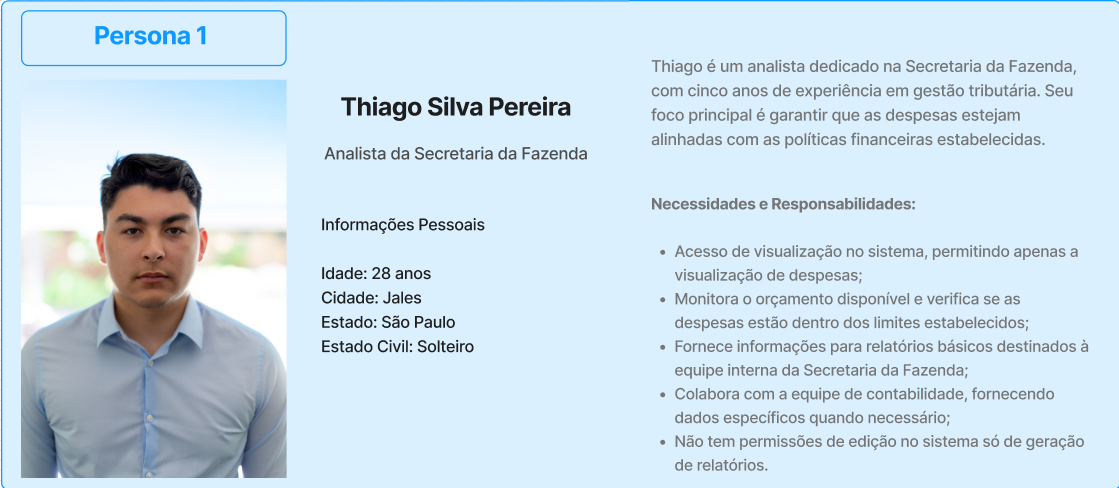
Figura 21 – Persona 1



Fonte: Elaborada pelos autores.

A Figura 22 apresenta a persona "Thiago Silva Pereira", um analista da “Secretaria da Fazenda”, dedicado a garantir que as despesas estejam em conformidade com as políticas financeiras vigentes. Com cinco anos de experiência na área, “Thiago” concentra-se no acompanhamento do orçamento, assegurando que os gastos estejam dentro dos limites estabelecidos e contribuindo para a transparência e o controle financeiro.

Figura 22 – Persona 2



Fonte: Elaborada pelos autores.

## Esboços de tela (wireframes)

*Wireframes* são protótipos de baixa fidelidade de páginas na web ou telas de aplicações. Eles são utilizados para mapear elementos-chave, como navegação, seções de conteúdo, necessidade de imagens e/ou mídias, componentes de formulário e chamadas para ação, que serão exibidos na página ou tela final (Unger; Chandler, 2009).

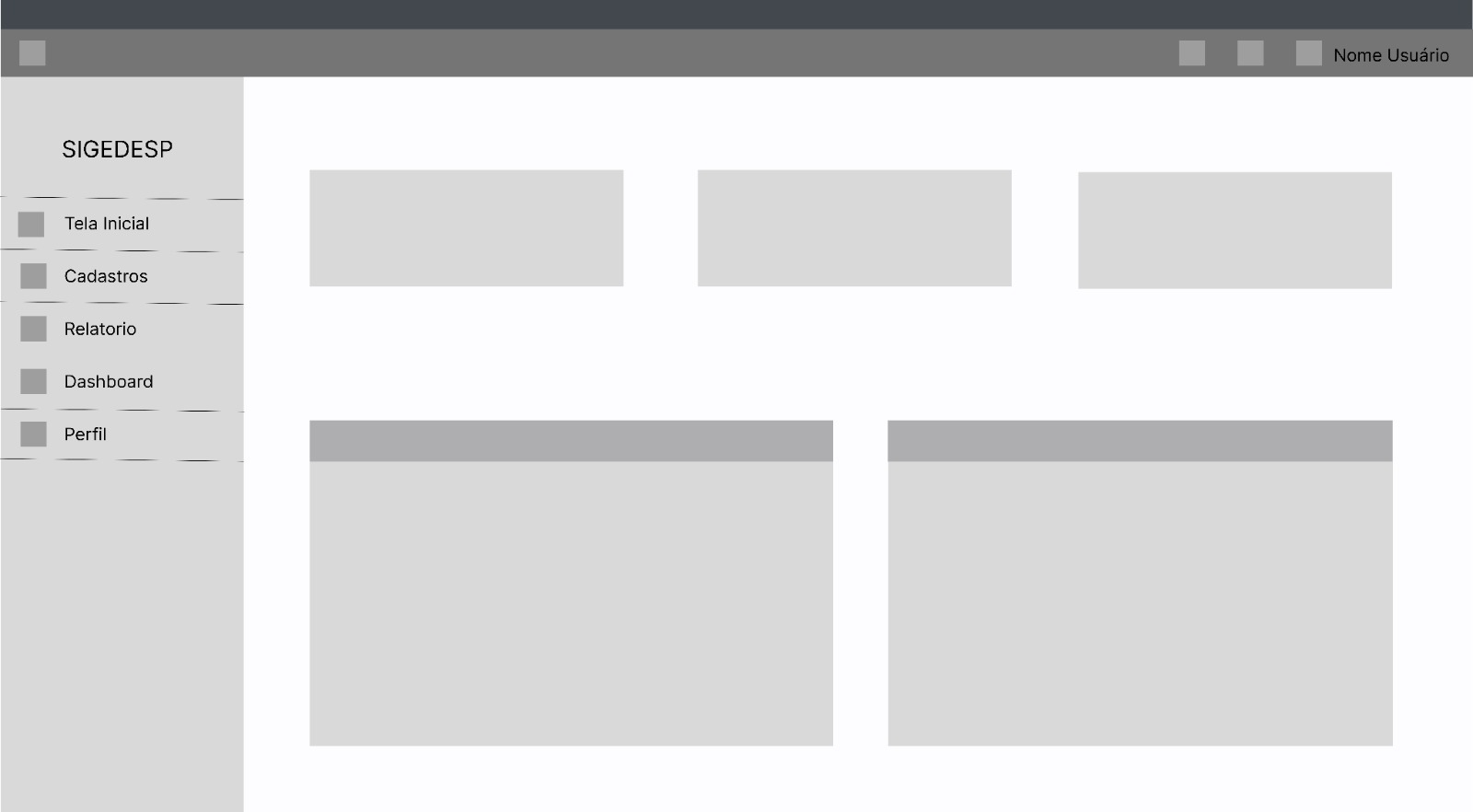
O design de um *wireframe* é, em geral, feito em escalas de preto, branco e cinza, usando substitutos para imagens e tipografias não especificadas, pois o foco é estrutural e funcional, e não visual. Essa simplicidade é intencional, já que o *wireframe* serve como uma ferramenta de comunicação para visualizar a disposição dos elementos e obter uma validação inicial de *stakeholders* antes das etapas de design visual e desenvolvimento (Unger; Chandler, 2009).

A relevância dos *wireframes* reside em sua capacidade de oferecer uma representação inicial e compreensível do layout do sistema, facilitando o alinhamento das expectativas do cliente e da equipe de desenvolvimento. Assim, a validação dos *wireframes* pelo cliente permite ajustar a estrutura da interface e funcionalidades propostas com antecedência, evitando retrabalho nas fases posteriores (Unger; Chandler, 2009).

Nas Figuras 23 a 26, são apresentados os *wireframes* das telas mais relevantes do sistema. Com base nessas representações, será possível desenvolver o protótipo visual das telas e de outras que estruturam o sistema SIGEDESP, avançando com segurança na criação da interface final.

Na Figura 23, apresenta-se o *wireframe* da tela *Home*. A estrutura exibe uma *Navbar* superior, que facilita o acesso rápido às funcionalidades principais e às informações de perfil, como o nome do usuário logado, ícones de logout e configurações. À esquerda, encontra-se uma *Sidebar* que oferece uma navegação complementar, permitindo uma experiência de navegação hierárquica e organizada. Essa estrutura de navegação lateral é especialmente recomendada para sistemas que contêm múltiplos módulos ou funcionalidades, como o sistema SIGEDESP, pois facilita a compreensão do escopo da aplicação e minimiza a sobrecarga de informações para o usuário.

No centro da interface, encontra-se a Área de Conteúdo Principal, onde são exibidas informações e dashboards relevantes para o usuário. Esta área central desempenha um papel essencial na captura da atenção do usuário, fornecendo uma visão geral das informações essenciais da plataforma. A organização limpa e o uso de componentes de alto contraste para elementos clicáveis contribuem para uma navegação intuitiva e amigável (Figura 23).

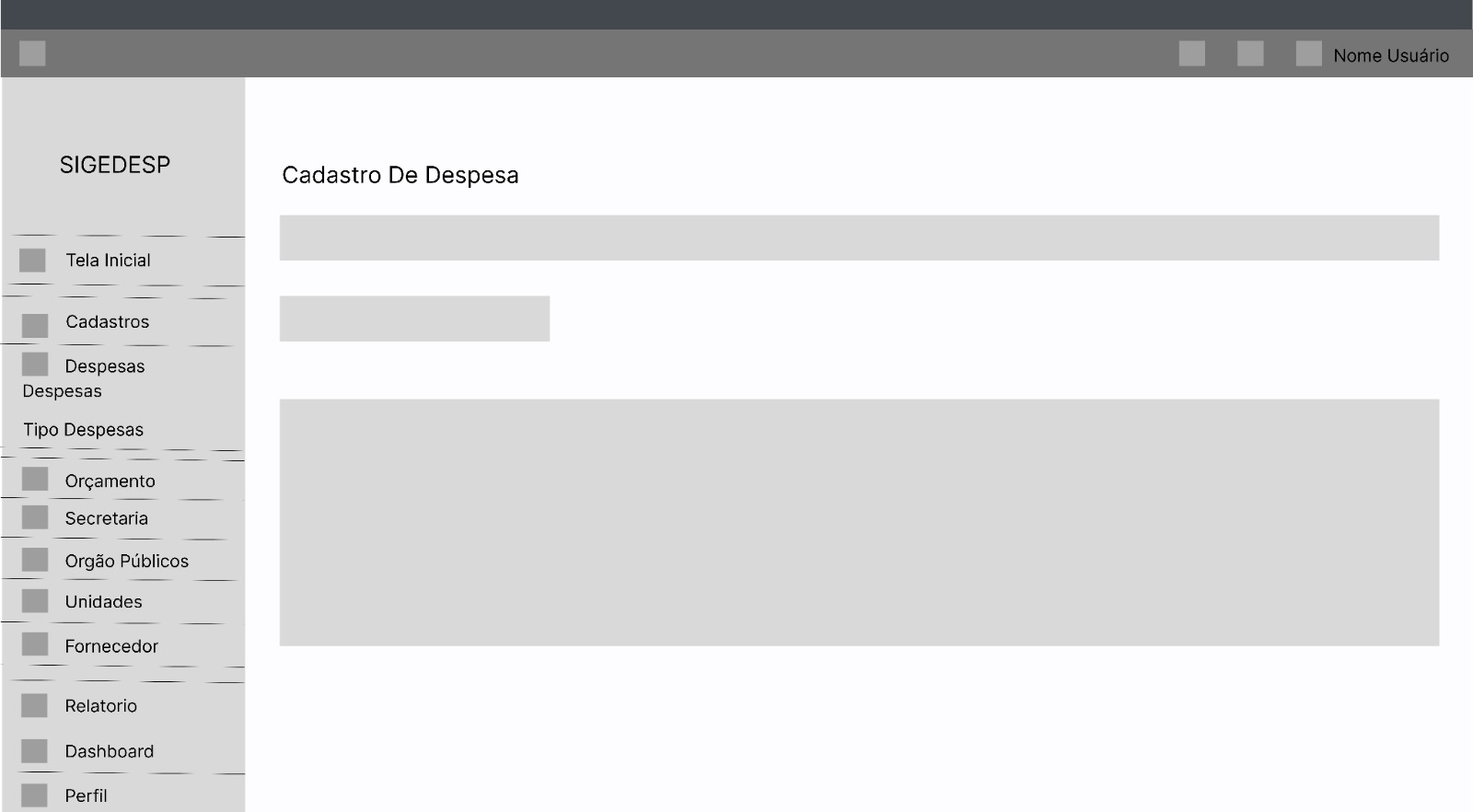
Figura – Wireframe - Tela Home

Fonte: Elaborada pelos autores.

Na Figura 24, é apresentado o *wireframe* da tela de cadastro de “Despesa”. A *Navbar* e a *Sidebar* seguem o mesmo padrão da tela *Home*, assegurando uniformidade na experiência de navegação do usuário. O design consistente da navegação facilita o aprendizado e familiarização com o sistema, promovendo uma navegação mais eficiente.

No centro da interface, é posicionado um botão para cadastro de novas despesas, além de uma tabela que lista as informações mais relevantes sobre cada despesa registrada. A tabela também apresenta botões de alteração e exclusão, possibilitando ao usuário gerenciar facilmente as despesas cadastradas no sistema (Figura 24).

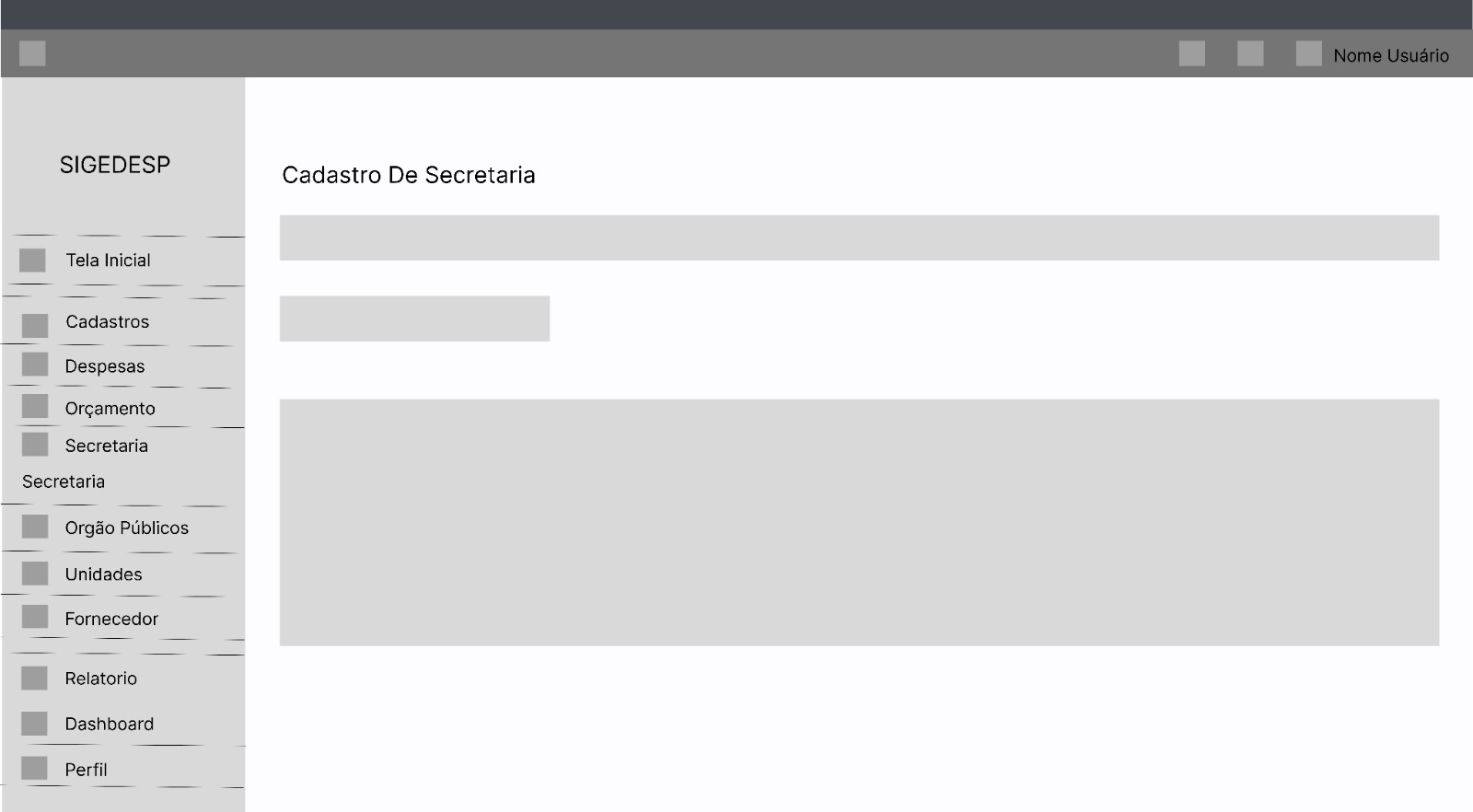
Figura – Wireframe – Tela Cadastro de Despesa



Fonte: Elaborada pelos autores.

Conforme apresentado na Figura 25, o *wireframe* da tela de cadastro de “Secretaria” mantém o padrão de design das telas *Home* e “Despesa”, com a *Navbar* e *Sidebar* garantido a uniformidade da experiência de navegação do usuário. No centro da interface, o *layout* segue o padrão das telas de cadastro, com um botão para adicionar novas secretarias e uma tabela que lista as informações mais relevantes sobre os cadastros existentes. Botões de alteração e exclusão são disponibilizados para possibilitar a gestão das informações já cadastradas no sistema.

Figura – Wireframe - Tela Cadastro de Secretaria



Fonte: Elaborada pelos autores.

O *wireframe* da tela de cadastro de “Instituição” (Figura 26) segue os mesmos padrões previamente estabelecidos. A *Navbar* e *Sidebar* são mantidas para assegurar a uniformidade da experiência de navegação do usuário. No centro da interface, o *layout* também segue o padrão das demais telas de cadastro, contendo um botão para cadastrar novas instituições e uma tabela que exibe as informações mais importantes sobre cada instituição cadastrada. Botões de alteração e exclusão estão disponíveis para facilitar o gerenciamento das informações já existentes no sistema.

Figura – Wireframe - Tela Cadastro de Instituição



Fonte: Elaborada pelos autores.

## Protótipos de Tela

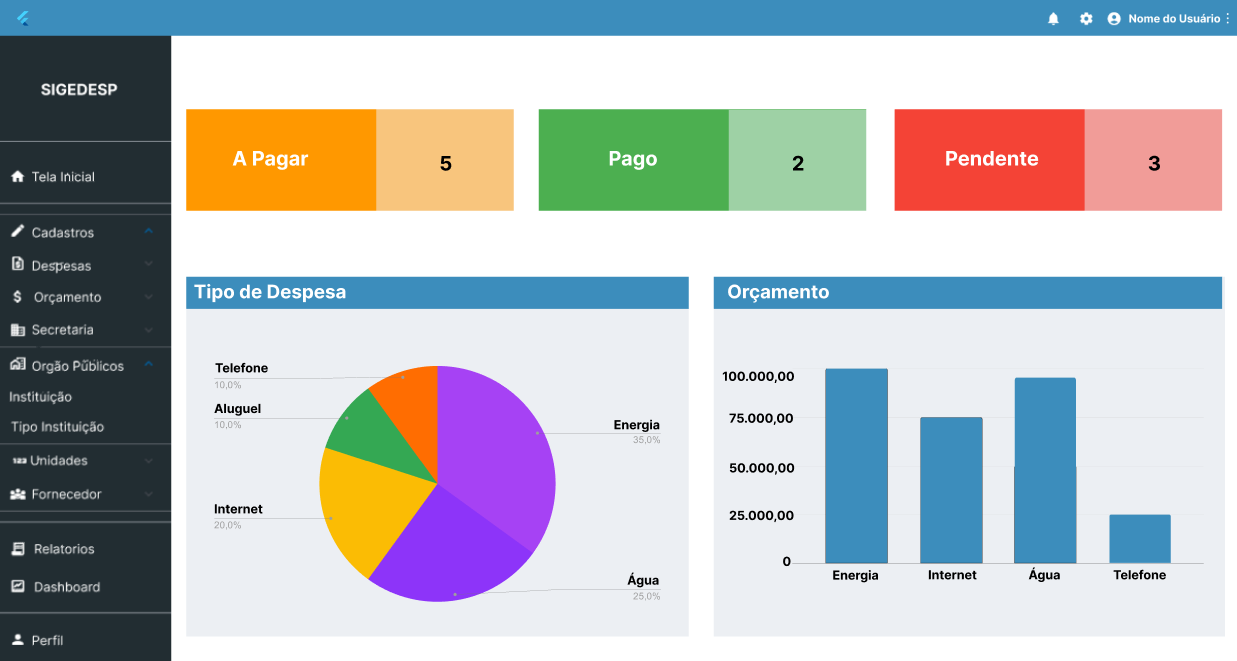
Segundo Rosa (2024), "a finalidade principal de um protótipo é validar hipóteses com uma versão mais simples". Em outras palavras, a criação de protótipos permite testar ideias e verificar, na prática, o que funciona e o que pode ser aprimorado. O *wireframe*, por sua vez, busca estruturar o *layout* e a navegação, enquanto o protótipo tem como objetivo simular de forma mais realista a experiência do usuário, possibilitando uma compreensão mais profunda de suas interações com o produto (Rosa, 2024).

Os protótipos podem ser classificados em três tipos principais. O de baixa fidelidade oferece uma versão simplificada e limitada, com foco nos aspectos gerais e no fluxo básico. O de média fidelidade inclui mais detalhes visuais e interativos, embora permaneça incompleto. Já o de alta fidelidade apresenta uma versão avançada, próxima do produto final, com elementos gráficos detalhados e funcionalidades interativas, permitindo testar de forma mais realista a experiência e as reações dos usuários (PM3, 2023).

No desenvolvimento dos protótipos de tela do sistema SIGEDESP, foi utilizada a ferramenta Figma (2024). Com ela, foram criados protótipos de alta fidelidade, incorporando cores, elementos gráficos e funcionalidades interativas, a fim de proporcionar uma visualização mais precisa do produto final e de validar a experiência do usuário antes da implementação completa.

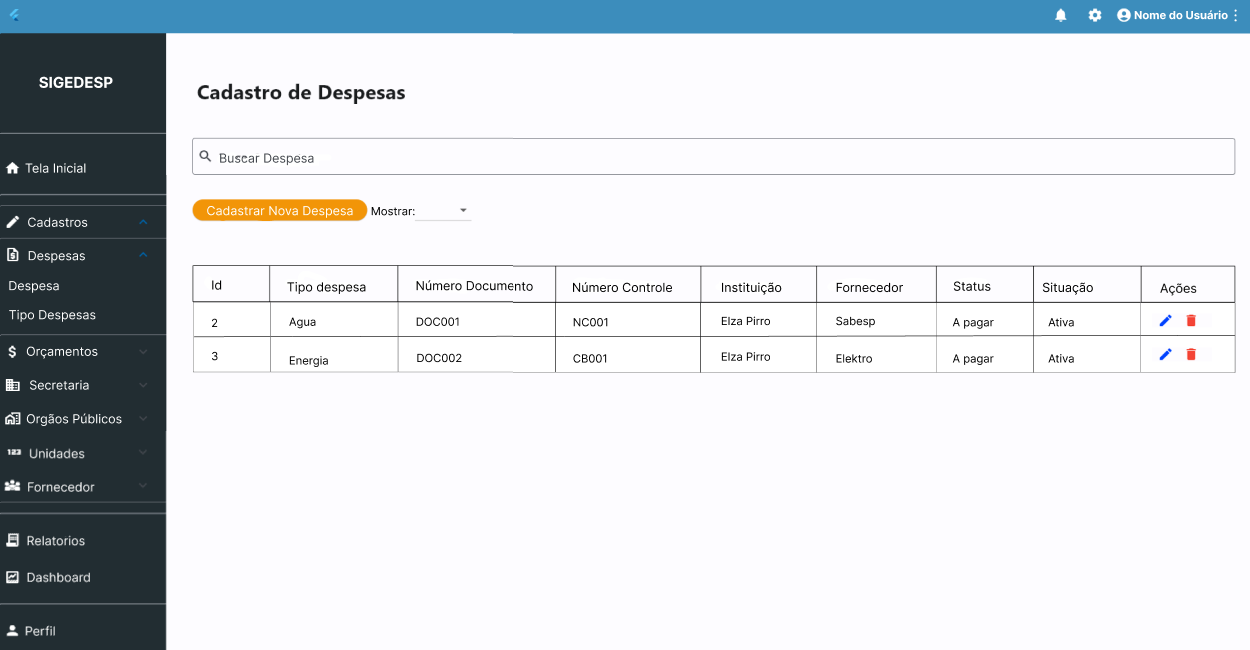
O protótipo da tela inicial do sistema (Figura 27) exibe cards e gráficos que apresentam informações relevantes de forma clara e acessível. Nos *cards*, são detalhadas as despesas registradas no sistema, com o status de cada uma, como "a pagar," "paga," ou "pendente." Já os gráficos fornecem dados adicionais importantes, como os tipos de despesa que estão cadastradas no sistema e o orçamento relacionado a cada tipo de despesa.

Figura – Protótipo Tela – Home



Fonte: Elaborada pelos autores.

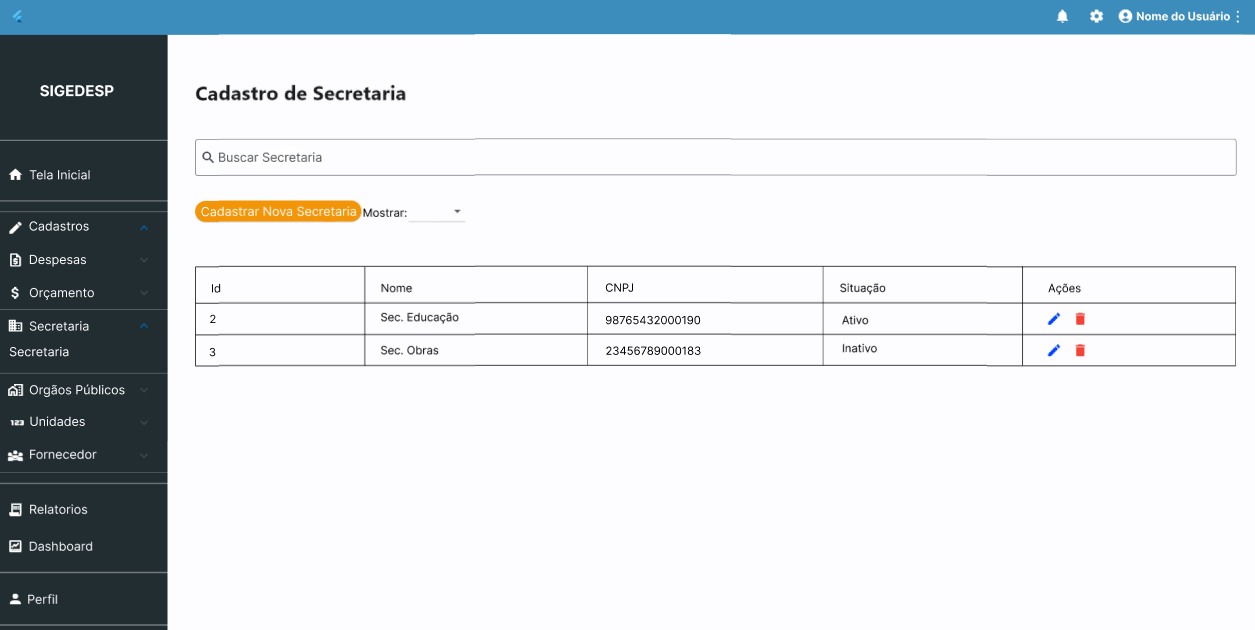
O protótipo da tela de cadastro de despesas, representa a principal regra de negócio do sistema. Nessa interface (conforme apresentado na Figura 28), usuários com perfil de administrador ou funcionário podem cadastrar novas despesas, inserindo as informações necessárias para registro. Além disso, é possível editar dados de despesas já cadastradas ou, caso necessário, removê-las da lista.

Figura – Protótipo Tela – Cadastro de Despesa

Fonte: Elaborada pelos autores.

As secretarias representam as principais divisões administrativas do município, cada uma focada em uma área específica, como saúde, educação, finanças, infraestrutura, entre outras. Cada secretaria possui uma equipe e recursos alocados para desenvolver políticas públicas, coordenar projetos e garantir a execução dos serviços pertinentes à sua área de atuação. Na Figura 29 apresenta-se o protótipo da tela que vai permite a manutenção de uma secretaria no sistema. Nessa interface, usuários com perfil de administrador ou funcionário podem cadastrar novas secretarias, inserindo as informações necessárias para registro. Além disso, é possível editar dados de secretarias já cadastradas ou, caso necessário, removê-las da lista.

Figura – Protótipo Tela – Cadastro de Secretaria

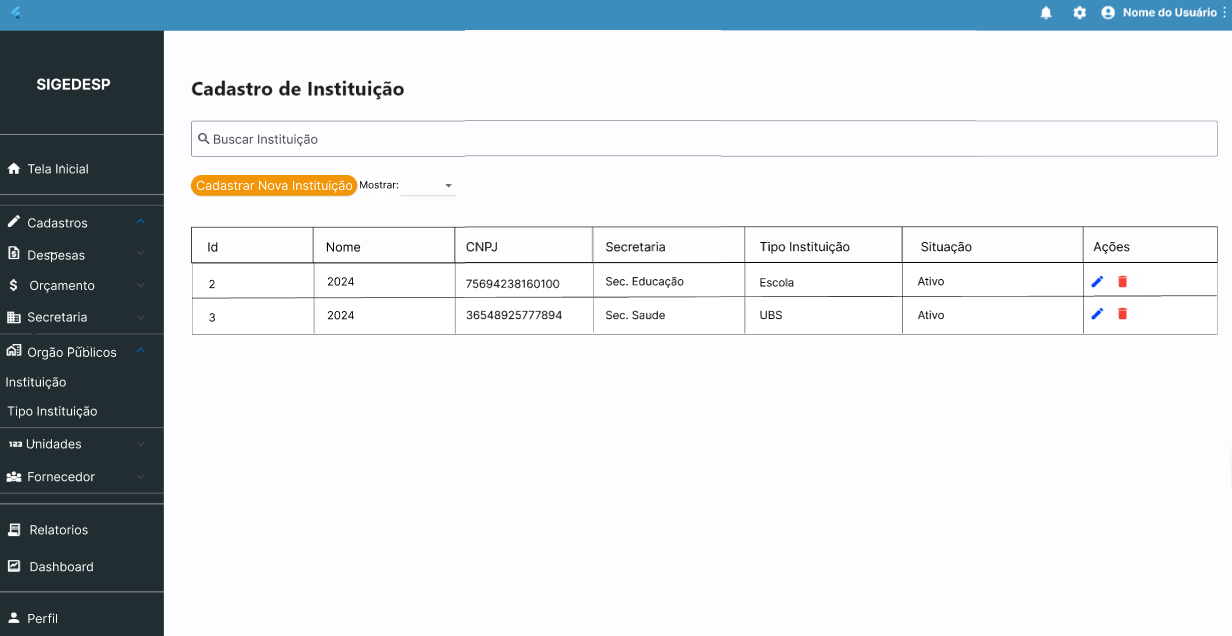


Fonte: Elaborada pelos autores.

As instituições vinculadas a cada secretaria são unidades operacionais que desempenham as atividades diárias e específicas dessas secretarias. Por exemplo, na “Secretaria de Educação”, estão vinculadas as escolas municipais, centros de educação infantil e demais entidades educativas sob a responsabilidade do município. Essas instituições trabalham diretamente com a comunidade, fornecendo os serviços educacionais definidos pela secretaria e garantindo que as políticas públicas educacionais sejam implementadas. Assim é necessário realizar a manutenção (Figura 30) destas instituições no sistema, pois elas são as geradoras de despesas no processo da informação dentro do software.

Nesse protótipo (conforme apresentado na Figura 30), usuários com perfil de administrador ou funcionário podem cadastrar novas instituições, inserindo as informações necessárias para registro. Além disso, é possível editar dados de instituições já cadastradas ou, caso necessário, removê-las da lista.

Figura – Protótipo Tela – Cadastro de Instituição

****

Fonte: Elaborada pelos autores.

# BANCO DE DADOS

Um banco de dados pode ser definido como um conjunto de dados relacionados que representam aspectos do mundo real, armazenados com um propósito específico (Machado, 2020).

O modelo relacional é o principal utilizado para o processamento de dados. Ele organiza as informações em tabelas que representam os dados e suas relações (Silberschatz; Korth; Sudarshan, 2012). Entre os bancos de dados relacionais mais populares, destaca-se o PostgreSQL. Este é um sistema de código aberto, amplamente acessível, do tipo objeto-relacional, que oferece diversos recursos para armazenar e processar dados (Silberschatz; Korth; Sudarshan, 2012). No sistema SIGEDESP, adotou-se o banco de dados PostgreSQL para gerenciar e processar o fluxo de dados de forma eficiente.

Os bancos de dados realizam diversas operações para acessar e atualizar informações, denominadas transações. Para assegurar que essas transações sejam confiáveis e consistentes, aplica-se o conceito *ACID*[[3]](#footnote-3), que compreende quatro propriedades fundamentais (Silberschatz; Korth; Sudarshan, 2012).

A atomicidade garante que todas as operações de uma transação sejam executadas integralmente ou, em caso de falha, nenhuma seja aplicada. A consistência assegura que o banco de dados permaneça em um estado válido antes e após a execução de uma transação. O isolamento impede que transações simultâneas interfiram umas nas outras, mantendo a integridade dos dados. Por fim, a durabilidade garante que as alterações realizadas por uma transação concluída sejam permanentes, mesmo em caso de falhas no sistema (Silberschatz; Korth; Sudarshan, 2012).

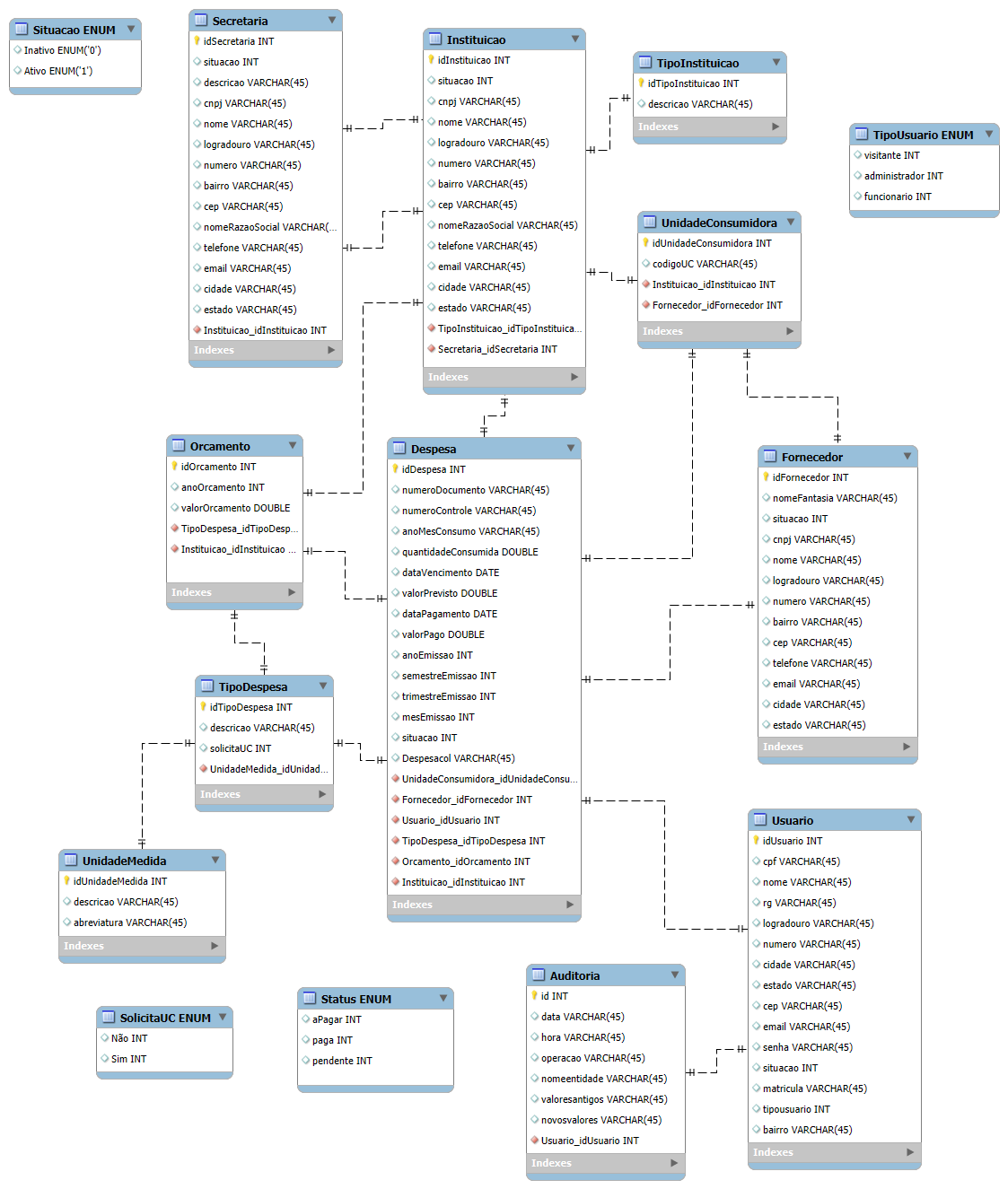
## Modelo Entidade Relacionamento

O Modelo Entidade-Relacionamento (E-R) é uma ferramenta amplamente utilizada para representar a estrutura lógica de um banco de dados de forma clara e simplificada, conforme discutido por Silberschatz, Korth e Sudarshan (2012).

O mapeamento objeto-relacional utiliza elementos como tabelas identificadas por chaves primárias, que servem para individualizar registros, atributos que representam os dados associados a cada registro, e chaves estrangeiras, que estabelecem relacionamentos entre tabelas (Silberschatz; Korth; Sudarshan, 2012).

A Figura 31 apresenta o mapeamento objeto-relacional do sistema SIGEDESP, mostrando uma visão detalhada das tabelas, seus atributos e os relacionamentos entre elas. Essa visualização facilita a compreensão da estrutura do banco de dados e destaca a integração lógica entre as tabelas, importante para a implementação e o funcionamento do sistema.

Figura – Mapeamento do Objeto Relacional



Fonte: Elaborada pelos autores.

## Script das tabelas

Segundo Machado (2020), o script de tabelas corresponde a um conjunto de instruções em *SQL*[[4]](#footnote-4) destinadas a definir a estrutura de uma tabela em um banco de dados. Por meio do comando “*CREATE TABLE”*, são especificados elementos como o nome da tabela, os campos, seus tipos de dados e as chaves estrangeiras.

Nos Quadros 34 a 44, serão apresentados os códigos de script que definem cada uma das classes no banco de dados do sistema SIGEDESP. Esses scripts têm como objetivo estruturar as tabelas e estabelecer as relações entre os diferentes elementos que compõem o sistema, garantindo a organização e o armazenamento eficiente dos dados. Cada script corresponde a uma classe específica, representando entidades fundamentais no SIGEDESP, e segue as melhores práticas para assegurar a integridade e a performance do banco de dados.

Quadro - Script SQL – Tabela Despesa

|  |
| --- |
| CREATE TABLE despesa ( despesaid SERIAL PRIMARY KEY,  numerodocumento VARCHAR(50) NOT NULL,  numerocontrole VARCHAR(50) NOT NULL,  anomesconsumo VARCHAR(50) NOT NULL,  quantidadeconsumida DOUBLE PRECISION NOT NULL,  datavencimento DATE NOT NULL,  valorprevisto DOUBLE PRECISION NOT NULL,  datapagamento DATE NOT NULL,  valorpago DOUBLE PRECISION NOT NULL,  anoemissao INTEGER NOT NULL,  semestreemissao INTEGER NOT NULL,  trimestreemissao INTEGER NOT NULL,  mesemissao INTEGER NOT NULL,  situcao INTEGER NOT NULL,  fornecedorid INTEGER NOT NULL REFERENCES fornecedor(fornecedorid) ON DELETE CASCADE,  unidadeconsumidoraid INTEGER NOT NULL REFERENCES unidadeconsumidora(unidadeconsumidoraid) ON DELETE CASCADE,  instituicaoid INTEGER NOT NULL REFERENCES instituicao(instituicaoid) ON DELETE CASCADE,  orcamentoid INTEGER NOT NULL REFERENCES orcamento(orcamentoid) ON DELETE CASCADE,  tipodespesaid INTEGER NOT NULL REFERENCES tipodespesa(tipodespesaid) ON DELETE CASCADE,  statusdespesa INTEGER NOT NULL,  usuarioid INTEGER NOT NULL REFERENCES usuario(usuarioid) ON DELETE CASCADE); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Secretaria

|  |
| --- |
| CREATE TABLE secretaria (  secretariaid SERIAL PRIMARY KEY,  situcao INTEGER NOT NULL,  descricao VARCHAR(100) NOT NULL,  cnpj VARCHAR(14) NOT NULL,  nome VARCHAR(100) NOT NULL,  logradouro VARCHAR(100) NOT NULL,  numero VARCHAR(15) NOT NULL,  bairro VARCHAR(100) NOT NULL,  cep VARCHAR(8) NOT NULL,  nomerazaosocial VARCHAR(100) NOT NULL,  telefone VARCHAR(11) NOT NULL,  email VARCHAR(100) NOT NULL,  cidade VARCHAR(25) NOT NULL,  estado VARCHAR(25) NOT NULL  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Instituição

|  |
| --- |
| CREATE TABLE instituicao (  instituicaoid SERIAL PRIMARY KEY,  situcao INTEGER NOT NULL,  cnpj VARCHAR(14) NOT NULL,  nome VARCHAR(100) NOT NULL,  logradouro VARCHAR(100) NOT NULL,  numero VARCHAR(15) NOT NULL,  bairro VARCHAR(100) NOT NULL,  cep VARCHAR(8) NOT NULL,  nomerazaosocial VARCHAR(100) NOT NULL,  telefone VARCHAR(11) NOT NULL,  email VARCHAR(100) NOT NULL,  cidade VARCHAR(50) NOT NULL,  estado VARCHAR(50) NOT NULL,  tipoinstituicaoid INTEGER NOT NULL REFERENCES tipoinstituicao(tipoinstituicaoid) ON DELETE CASCADE,  secretariaid INTEGER NOT NULL REFERENCES secretaria(secretariaid) ON DELETE CASCADE  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Fornecedor

|  |
| --- |
| CREATE TABLE fornecedor (  fornecedorid SERIAL PRIMARY KEY,  nomefantasia VARCHAR(100) NOT NULL,  situcao INTEGER NOT NULL,  cnpj VARCHAR(14) NOT NULL,  nome VARCHAR(100) NOT NULL,  logradouro VARCHAR(100) NOT NULL,  numero VARCHAR(15) NOT NULL,  bairro VARCHAR(100) NOT NULL,  cep VARCHAR(8) NOT NULL,  telefone VARCHAR(11) NOT NULL,  email VARCHAR(100) NOT NULL,  cidade VARCHAR(25) NOT NULL,  estado VARCHAR(25) NOT NULL  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Orçamento

|  |
| --- |
| CREATE TABLE orcamento (  orcamentoid SERIAL PRIMARY KEY,  anoorcamento INTEGER NOT NULL,  valororcamento DOUBLE PRECISION NOT NULL,  tipodespesaid INTEGER NOT NULL REFERENCES tipodespesa(tipodespesaid) ON DELETE CASCADE,  instituicaoid INTEGER NOT NULL REFERENCES instituicao(instituicaoid) ON DELETE CASCADE  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Usuário

|  |
| --- |
| CREATE TABLE usuario (  usuarioid SERIAL PRIMARY KEY,  cpf VARCHAR(11) NOT NULL,  rg VARCHAR(12) NOT NULL,  nome VARCHAR(100) NOT NULL,  logradouro VARCHAR(100) NOT NULL,  numero VARCHAR(15) NOT NULL,  cidade VARCHAR(25) NOT NULL,  estado VARCHAR(25) NOT NULL,  cep VARCHAR(8) NOT NULL,  email VARCHAR(100) NOT NULL,  senha VARCHAR(15) NOT NULL,  bairro VARCHAR(100) NOT NULL,  situcao INTEGER NOT NULL,  matricula VARCHAR(100) NOT NULL,  tipousuario INTEGER NOT NULL  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Auditoria

|  |
| --- |
| CREATE TABLE auditoria (  auditoriaid UUID PRIMARY KEY,  data TEXT NOT NULL,  hora TEXT NOT NULL,  operacao TEXT NOT NULL,  nomeentidade TEXT NOT NULL,  valoresantigos TEXT,  novosvalores TEXT,  idusuario INTEGER NOT NULL  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Tipo Despesa

|  |
| --- |
| CREATE TABLE tipodespesa (  tipodespesaid SERIAL PRIMARY KEY,  descricao VARCHAR(50) NOT NULL,  solicitauc INTEGER NOT NULL,  idunidademedida INTEGER NOT NULL REFERENCES unidademedida(unidademedidaid) ON DELETE CASCADE  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Tipo Instituição

|  |
| --- |
| CREATE TABLE tipoinstituicao (  tipoinstituicaoid SERIAL PRIMARY KEY,  descricao VARCHAR(100) NOT NULL  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Unidade Medida

|  |
| --- |
| CREATE TABLE unidademedida (  unidademedidaid SERIAL PRIMARY KEY,  descricao VARCHAR(100) NOT NULL,  abreviatura VARCHAR(10) NOT NULL  ); |

Fonte: Elaborada pelos autores.

Quadro - Script SQL – Tabela Unidade Consumidora

|  |
| --- |
| CREATE TABLE unidadeconsumidora (  unidadeconsumidoraid SERIAL PRIMARY KEY,  unidadeconsumidora VARCHAR(50) NOT NULL,  fornecedorid INTEGER NOT NULL REFERENCES fornecedor(fornecedorid) ON DELETE CASCADE,  instituicaoid INTEGER NOT NULL REFERENCES instituicao(instituicaoid) ON DELETE CASCADE  ); |

Fonte: Elaborada pelos autores.

## 5.3 Mapeamento Objeto Relacional – ORM

Os sistemas de mapeamento objeto-relacional são baseados em bancos de dados relacionais tradicionais e facilitam a integração entre o modelo de objetos utilizado na programação e o banco de dados relacional na criação de aplicações. Esse tipo de mapeamento permite realizar consultas, que são automaticamente convertidas em instruções *SQL* (Silberschatz; Korth; Sudarshan, 2012).

No projeto SIGEDESP, desenvolvido em C#, foi empregado o Entity Framework como ferramenta para o mapeamento objeto-relacional. Esse recurso exige a instalação de pacotes específicos, a criação de classes de entidade para representar as tabelas e a configuração do banco de dados por meio de comandos. Por exemplo, o comando “*add-migration* Banco” gera uma migração, que corresponde à estrutura inicial das tabelas, enquanto o comando “*database-update*” aplica a migração e integra as alterações ao banco de dados (Microsoft, 2023).

# ARQUITETURA DE SOFTWARE

Segundo Sommerville (2018), a arquitetura do software deve ser definida antes da etapa de implementação, pois isso assegura uma conexão clara entre os requisitos do sistema e a estruturação de seus componentes.

Uma arquitetura bem planejada possibilita o desenvolvimento de um sistema eficiente, promovendo uma divisão clara dos componentes, organização e comunicação eficaz entre eles. Além disso, facilita uma implementação direta e objetiva, atendendo às necessidades operacionais e garantindo manutenção simplificada, o que resulta na redução de custos. Esses processos têm como objetivo otimizar o ciclo de vida do sistema e evitar a sobrecarga dos desenvolvedores durante o processo de criação (Martin, 2020).

## 6.1 Arquitetura de desenvolvimento

Para a execução deste projeto, foram realizadas etapas como a coleta de dados, o diagnóstico da situação atual e o levantamento das informações essenciais ao processo de gestão de despesas públicas. Com base na análise dos dados coletados e no estudo das funcionalidades necessárias, procedeu-se ao levantamento dos requisitos para a modelagem e o desenvolvimento do software, seguindo os princípios metodológicos da engenharia de software delineados por Pressman (2011).

Para representar a análise de requisitos, utilizou-se a Linguagem de Modelagem Unificada (*UML*), uma linguagem de modelagem baseada no paradigma de orientação a objetos (Guedes, 2018). Por meio do software Astah UML (Astah, 2024), foram desenvolvidos diversos diagramas *UML* que permitiram uma análise detalhada das funcionalidades e dos requisitos operacionais do sistema, facilitando a identificação precisa dos serviços e recursos que a plataforma pode oferecer aos usuários.

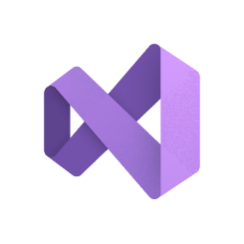
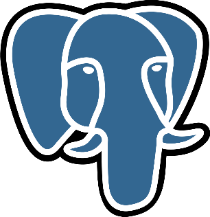
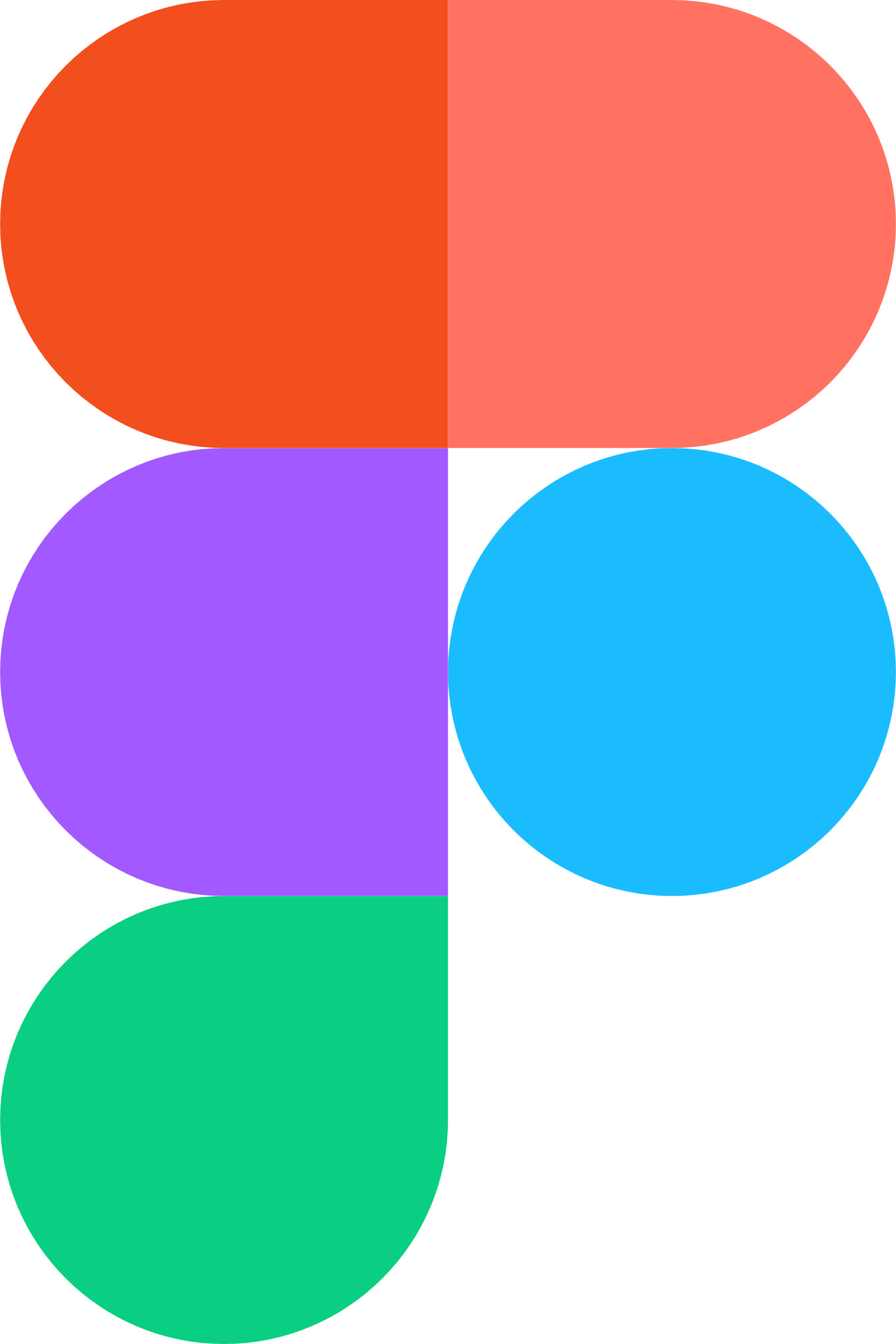
Na fase de desenvolvimento do *back-end* da aplicação, foi utilizado o Ambiente de Desenvolvimento Integrado (*IDE*) Visual Studio, reconhecido por suas funcionalidades avançadas de edição de código e recursos robustos para testes. A linguagem de programação selecionada foi o C#, conhecida por ser uma linguagem de uso geral, multiplataforma e de alto desempenho (Wagner, 2024). Para o armazenamento e a gestão dos dados, foi empregado o banco de dados PostgreSQL, um Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR) projetado para administrar o acesso eficiente e seguro às informações (Milani, 2008).

A prototipação das interfaces do sistema foi realizada com a ferramenta Figma (2024), e o desenvolvimento do *front-end*, foi escolhido o *framework* Flutter, uma estrutura de código aberto que permite a criação de sistemas multiplataforma a partir de uma única base de código (Flutter, 2024). O ambiente de desenvolvimento utilizado foi o Visual Studio Code, com a linguagem de programação Dart (2024), garantindo maior eficiência e uniformidade no desenvolvimento da interface de usuário.

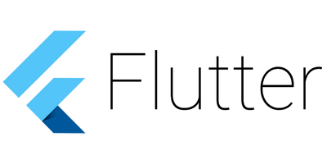
Para a gestão do projeto, foi adotada a metodologia Scrum[[5]](#footnote-5) (Sutherland; Sutherland, 2019), utilizando sprints para um desenvolvimento iterativo e incremental. Após cada entrega parcial, foram realizadas análises detalhadas dos módulos específicos, seguidas por uma avaliação global do sistema. As etapas do projeto foram organizadas e monitoradas rigorosamente na plataforma Azure DevOps (Microsoft, 2024), uma ferramenta de controle de tarefas oferece funcionalidades gratuitas para o gerenciamento de projetos, incluindo planejamento de sprints, controle de tarefas e relatórios de desempenho. Para o armazenamento e versionamento de código do projeto *front-end* e *back-end*, empregou-se a ferramenta GitHub (2024).

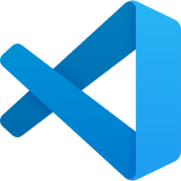
O Quadro 45 apresenta a representação visual dos *softwares* gratuitos utilizados ao longo de todo o processo de desenvolvimento do sistema. Estes programas foram empregados nas diferentes etapas do projeto, abrangendo desde a gestão do sistema até a codificação e o design, contribuindo para a organização e eficiência no desenvolvimento do software.

Quadro – Softwares Utilizados

Logotipo

Descrição gerada automaticamente  

Fonte: Elaborada pelos autores.

## 6.1.1 Back-end

Para o desenvolvimento do *back-end* do projeto SIGEDESP, foi utilizada a linguagem de programação C#, uma linguagem orientada a objetos, multiplataforma e de código aberto (Wagner, 2024). A escolha do C# foi baseada em sua robustez, escalabilidade e flexibilidade, características importantes para o desenvolvimento de sistemas complexos e de fácil manutenção (Wagner, 2024).

A aplicação foi estruturada para fornecer serviços por meio de uma *API[[6]](#footnote-6) RESTful*, que segue os princípios do *Representational State Transfer* (REST). Essa arquitetura, amplamente adotada em sistemas modernos, utiliza o protocolo *HTTP[[7]](#footnote-7)* para realizar operações de *CRUD* (*Create, Read, Update, Delete*) sobre recursos, os quais são identificados por *URIs* (*Uniform Resource Identifiers*). Cada recurso é tratado como uma entidade única e manipulado por meio dos métodos *HTTP* padrão, como *GET, POST, PUT e DELETE*. Essa abordagem permite uma interface padronizada e eficiente, além de garantir a flexibilidade e a escalabilidade necessárias para o gerenciamento de dados do sistema (Sommerville, 2018).

A comunicação entre o cliente e o servidor no sistema é realizada no formato *JSON* (*JavaScript Object Notation*), um formato leve e de fácil leitura para a troca de dados. A escolha do JSON foi estratégica, pois facilita a serialização eficiente das informações e assegura a integração entre diferentes sistemas e linguagens de programação. Esse formato também permite que dados sejam compartilhados de forma padronizada e acessível entre plataformas distintas, o que aprimora a interoperabilidade do sistema e torna a comunicação mais fluida, independentemente das tecnologias utilizadas em cada camada da aplicação (Sommerville, 2018).

A arquitetura do sistema foi organizada em camadas para garantir a modularidade e facilitar a manutenção e escalabilidade do código. A camada de Model é responsável por representar os dados e as regras de negócio do sistema, incluindo a estrutura do banco de dados, o que facilita o controle e a integridade dos dados (Baradel, 2024). A camada *Repository* atua como uma abstração entre a lógica de negócios e o acesso aos dados, separando essas duas responsabilidades e permitindo uma maior flexibilidade na manipulação e persistência das informações (Meine, 2024).

A camada *Service*, por sua vez, é responsável pela implementação das regras de negócio e pela validação das informações recebidas da camada *Controller*, garantindo que todos os dados sejam consistentes e atendam aos requisitos da aplicação (Silvestre, 2022). Já a camada *Controller* tem a função de gerenciar as requisições dos usuários, delegando a execução da lógica de negócio para a camada *Service* e, posteriormente, retornando a resposta ao cliente (Silvestre, 2022).

Além disso, a camada *DTO* foi incorporada ao projeto para otimizar o transporte de dados entre as diferentes camadas do sistema. Os *DTOs* encapsulam os dados de forma simples, sem incluir lógica de negócio, o que facilita a transferência de informações entre as camadas de maneira eficiente e organizada (Walker, 2023).

Com essa abordagem em camadas e a adoção de uma *API RESTful*, o sistema SIGEDESP garante uma estrutura bem definida, escalável e de fácil manutenção, permitindo uma comunicação eficiente entre os diversos componentes da aplicação e promovendo a integração de tecnologias de forma harmoniosa e independente.

## 6.1.2 Front-end

Para o desenvolvimento do *front-end* do projeto SIGEDESP, foi utilizado o Flutter como *framework* e Dart como linguagem de programação. O Flutter, desenvolvido pelo Google, é um *framework* multiplataforma que permite escrever o código uma única vez e executá-lo em diversas plataformas, como *iOS*, *Android* e *Web*. Essa abordagem oferece agilidade e versatilidade, otimizando o processo de desenvolvimento e manutenção do sistema (Alberto, 2023).

O Flutter, embora seja um *framework*, utiliza Dart como linguagem de programação. Dart também foi criado pelo Google e é projetado para o desenvolvimento de aplicações *cliente-side*, com foco em alta performance em plataformas móveis e *web*. Trata-se de uma linguagem orientada a objetos, que visa ser produtiva e flexível, facilitando a criação de interfaces dinâmicas e responsivas (Alberto, 2023).

A estrutura do Flutter é baseada em *widgets*, que são os blocos de construção da interface do usuário. Os *widgets* podem ser classificados em dois tipos: *StatefulWidget* e *StatelessWidget*. O *StatefulWidget* é um componente que possui estado, ou seja, ele pode ser alterado dinamicamente durante a execução da aplicação, sem que seja necessário navegar para uma nova tela. Por outro lado, o *StatelessWidget* não possui estado e, portanto, não sofre alterações após ser renderizado. Essa abordagem permite maior flexibilidade na construção da interface e facilita a criação de componentes reutilizáveis e altamente responsivos (Santana, 2019).

Além de utilizar a estrutura básica do Flutter, foram integradas diversas bibliotecas que aprimoram a funcionalidade e a eficiência no desenvolvimento. A biblioteca *flutter\_modular*, por exemplo, facilita a modularização da aplicação, permitindo dividir o aplicativo em módulos independentes, cada um com suas próprias rotas, dependências e dados. Essa modularização contribui para a organização do código, tornando-o mais fácil de entender, testar e manter (Pub.Dev, 2024a).

Para lidar com operações que podem resultar em falhas, foi utilizada a biblioteca *result\_dart*, que define um tipo *Result*. Esse tipo permite armazenar o valor retornado em caso de sucesso ou um erro em caso de falha, proporcionando uma maneira estruturada e segura de tratar exceções e falhas de operação (Pub.Dev, 2024b).

A conversão entre objetos *JSON* e Dart também foi otimizada por meio da biblioteca *dson\_adapter*, que simplifica a transformação de dados entre esses dois formatos, utilizando o *build\_runner*. Isso resulta em um código mais enxuto e facilita a integração com APIs externas, além de garantir a consistência na manipulação de dados (Pub.Dev, 2023).

Para a internacionalização e localização do sistema, a biblioteca *intl* foi empregada. Ela oferece funcionalidades para a formatação de datas, números e strings de acordo com a localidade do usuário, permitindo que a aplicação forneça uma experiência consistente em diferentes regiões e idiomas (Pub.Dev, 2024c).

Por fim, a biblioteca *http* foi utilizada para realizar solicitações *HTTP*, proporcionando uma *API* de alto nível para interagir com *APIs* da *web*. Com essa biblioteca, é possível realizar operações de comunicação com servidores, como solicitações *GET, POST, PUT e DELETE*, facilitando a troca de dados entre o *front-end* e o *back-end* da aplicação (Pub.Dev, 2024d).

Com a combinação de Flutter, Dart e as bibliotecas mencionadas, foi possível desenvolver um *front-end* modular, eficiente e altamente responsivo para o sistema SIGEDESP, garantindo uma experiência de usuário otimizada e facilitando a manutenção e expansão da aplicação no futuro.

## 6.2 Segurança da Informação

A segurança da informação é um campo importante dentro da área de tecnologia da informação, e seu principal objetivo é garantir a proteção de dados e informações essenciais, de modo que apenas usuários autorizados possam acessá-las (Holdsworth; Kosinski, 2024). De acordo com Holdsworth e Kosinski (2024), a segurança da informação *(InfoSec)* envolve a proteção das informações para que elas permaneçam confidenciais, íntegras e disponíveis de acordo com as necessidades dos usuários autorizados. Em outras palavras, a segurança da informação visa assegurar que os dados não sejam acessados ou alterados por pessoas não autorizadas, e que estejam acessíveis para os usuários legítimos sempre que necessário.

O *InfoSec* é sustentado por três princípios essenciais, conhecidos como a tríade CIA, que se referem a Confidencialidade, Integridade e Disponibilidade. A Confidencialidade garante que apenas indivíduos autorizados tenham acesso a informações sensíveis, evitando que dados sejam divulgados de forma inadequada. A Integridade assegura que as informações sejam mantidas precisas, completas e sem modificações não autorizadas. Já a Disponibilidade garante que as informações estejam acessíveis a usuários autorizados quando necessário, sem interrupções indevidas (Holdsworth; Kosinski, 2024). Além disso, Holdsworth e Kosinski (2024) destacam outros dois princípios importantes: a Garantia da Informação, que envolve a contínua implementação dos elementos da tríade CIA, e a Não Repudiação, que impede que um usuário negue a autoria de uma ação realizada no sistema, pois todas as suas interações são autenticadas e rastreáveis.

Em relação a segurança da informação, uma das tecnologias frequentemente utilizadas para garantir a troca segura de dados entre sistemas e usuários é o *JSON Web Token* (JWT, 2024). O JWT transmiti informações entre diferentes partes de um sistema de maneira segura, utilizando um objeto *JSON*. Este token pode ser utilizado em diversos cenários, como na autenticação e autorização de usuários. No processo de autorização, por exemplo, após a autenticação inicial de um usuário no sistema, o JWT é gerado e retornado ao cliente. Em todas as requisições subsequentes, o cliente inclui esse token no cabeçalho da solicitação, permitindo que o servidor verifique a identidade do usuário e conceda acesso apenas às rotas, serviços e recursos que foram previamente autorizados (JWT, 2024).

No sistema SIGEDESP, quando um usuário realiza o login com suas credenciais, como e-mail e senha, e a autenticação é bem-sucedida, um *JSON Web Token* é gerado e retornado ao usuário. Esse token serve como uma chave que permite ao usuário acessar as informações e funcionalidades autorizadas, garantindo que a comunicação entre o cliente e o servidor seja realizada de forma segura e eficiente, sem a necessidade de reiniciar o processo de autenticação a cada requisição.

Além do uso de tokens, o sistema adota mecanismos de auditoria para reforçar sua segurança. A auditoria consiste no registro detalhado de acessos e alterações realizadas no sistema, permitindo a geração de relatórios confiáveis e seguros sobre sua movimentação. Esse processo é essencial para rastrear ações realizadas pelos usuários, contribuindo para a transparência e a identificação de possíveis irregularidades (Pressman; Maxim, 2021).

No sistema SIGEDESP, a auditoria desempenha um papel importante na segurança e confiabilidade das operações. O processo registra informações como valores anteriores e atuais, datas, horários e o usuário responsável pela ação. Isso permite documentar detalhadamente quem acessou o sistema, o que foi modificado ou adicionado, e quando a ação ocorreu. Dessa forma, é possível monitorar todas as atividades de forma precisa, garantindo não apenas a segurança, mas também a responsabilidade e a rastreabilidade das operações. Além disso, os dados gerados pela auditoria podem ser utilizados para análises mais aprofundadas, como a identificação de padrões de uso ou a detecção de atividades suspeitas.

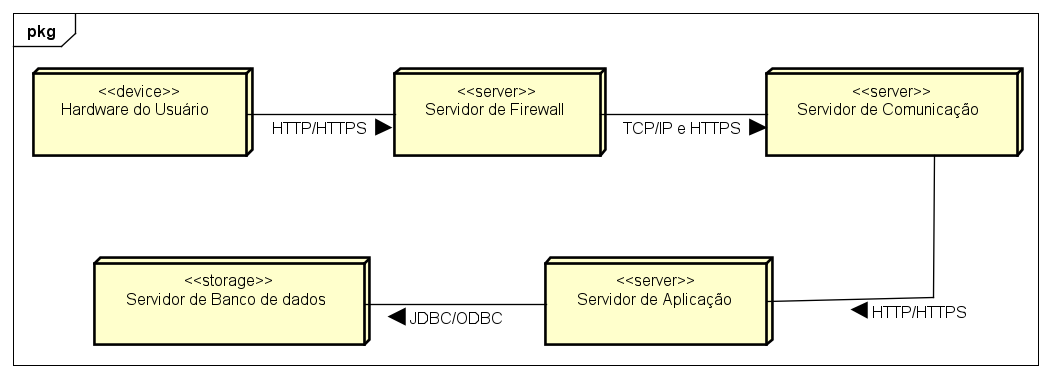
## 6.3 Implantação do Sistema

Para a implementação do sistema, foi necessário seguir o modelo do diagrama de implantação. Segundo Guedes (2018), o diagrama de implantação possui a visão mais física do sistema. Ele descreve a organização da arquitetura física, indicando onde o software será implantado e executado em termos de hardware, como computadores pessoais, servidores, entre outros. Também detalha como esses dispositivos estarão interligados e quais protocolos serão utilizados para a comunicação e troca de informações. Esse tipo de diagrama é relevante apenas quando o sistema em desenvolvimento será executado em múltiplas máquinas.

O componente básico e mais importante do diagrama de implantação são os “Nós”, que são representados como cubos contendo um texto que indica qual é o tipo de Nó. No diagrama, um Nó pode representar várias coisas, como um item de hardware ou um ambiente de execução (Guedes, 2018).

O diagrama de implantação do sistema SIGEDESP (Figura 33) foi desenvolvido com o objetivo de ilustrar a arquitetura física do sistema, destacando os diferentes componentes que o compõem e como eles interagem. A arquitetura do SIGEDESP foi projetada para ser eficiente, segura e escalável, e o diagrama reflete esses requisitos, especificando como os diversos nós — representando máquinas, servidores e componentes de software — estão distribuídos na infraestrutura e se comunicam.

Figura – Diagrama de Implantação



Fonte: Elaborada pelos autores.

A primeira camada do sistema é representada pelo hardware do usuário (Figura 33), que pode ser um computador ou qualquer outro dispositivo que o usuário final utilize para interagir com o sistema. Este nó é classificado com o estereótipo *<<device>>* e simboliza os dispositivos de entrada/saída, que se comunicam com os servidores do sistema por meio de *HTTP* ou *HTTPS[[8]](#footnote-8)* (Guedes, 2018). O uso do protocolo *HTTPS* garante a segurança na comunicação entre o dispositivo do usuário e o servidor, protegendo os dados transmitidos contra interceptação ou manipulação (Akemi, 2024).

Depois que as requisições são feitas a partir do dispositivo do usuário, elas são direcionadas para o servidor de *firewall* (Figura 33), que desempenha um papel crucial na segurança do sistema. Este nó é representado com o estereótipo *<<server>>* e serve para filtrar o tráfego de rede, garantindo que apenas as conexões seguras e autorizadas sejam permitidas para os servidores internos (Guedes, 2018). O servidor de *firewall* utiliza protocolos como *TCP/IP[[9]](#footnote-9)* e *HTTPS* para garantir a integridade da comunicação e proteger contra acessos não autorizados, controlando o fluxo de dados entre os diferentes componentes do sistema (Santana, 2024).

Uma vez que o tráfego é aprovado pelo firewall, ele é encaminhado para o servidor de comunicação. Este servidor, também representado como *<<server>>* (Figura 33), é responsável por intermediar a comunicação entre o usuário e os sistemas internos do SIGEDESP (Guedes, 2018). Ele expõe *APIs RESTful*, que são utilizadas para que o *front-end* possa interagir com o *back-end* de maneira eficiente. As *APIs RESTful* são uma escolha popular devido à sua simplicidade e flexibilidade, permitindo que o sistema se integre facilmente com diferentes tipos de clientes e plataformas (Sommerville, 2018). A comunicação entre o servidor de comunicação e o cliente é realizada por meio dos protocolos *HTTP/HTTPS*, assegurando que todas as requisições e respostas sejam enviadas de forma segura (Akemi, 2024).

O servidor de aplicação, também classificado como *<<server>>* (Figura 33), é o núcleo do sistema, onde a lógica de negócios é processada. Neste servidor, são executadas as operações fundamentais para o gerenciamento de despesas, como o cálculo de valores, categorização de transações e a geração de relatórios. O servidor de aplicação também gerencia as requisições que envolvem a manipulação de dados, interagindo diretamente com o servidor de banco de dados para armazenar e recuperar informações (Guedes, 2018). O servidor de aplicação pode utilizar protocolos como *HTTP/HTTPS* para se comunicar com o servidor de comunicação, enquanto a comunicação com o banco de dados é realizada por meio de protocolos como *JDBC[[10]](#footnote-10)* ou *ODBC[[11]](#footnote-11)*, garantindo a conexão eficiente e segura com o sistema de banco de dados (Haroon, 2024).

O servidor de banco de dados é responsável pelo armazenamento persistente das informações do sistema. No caso do SIGEDESP, é utilizado um banco de dados relacional PostgreSQL, que armazena dados críticos como registros de despesas, usuários, categorias de transações e outros dados financeiros. O estereótipo utilizado para este nó é *<<storage>>*, refletindo sua função de armazenamento persistente de dados (Figura 33). As conexões entre o servidor de aplicação e o banco de dados são feitas por meio dos protocolos *JDBC* ou *ODBC*, que permitem a execução de consultas SQL e a manipulação eficiente das tabelas e registros do banco (Haroon, 2024).

Para dar suporte à estrutura de implantação do sistema SIGEDESP, foi escolhido o Docker, uma plataforma que permite, por meio de contêineres, padronizar e combinar o código-fonte da aplicação com as bibliotecas e dependências do sistema operacional necessárias para executar o código em qualquer ambiente. O uso do Docker facilita a configuração e o gerenciamento de ambientes isolados, garantindo consistência no desenvolvimento, testes e produção. Neste trabalho, abordaremos a implantação do sistema SIGEDESP utilizando Docker e a integração do *back-end*, *front-end* e banco de dados, detalhando as configurações necessárias para que o sistema funcione de maneira integrada e eficiente (Susnjara e Smalley, 2024).

A configuração do ambiente de *back-end* no Docker começa com a criação de um arquivo chamado *Dockerfile*, que deve ficar na mesma pasta do arquivo principal do projeto. Esse arquivo define como o contêiner será montado, incluindo as etapas para preparar a aplicação e garantir que ela funcione corretamente. O código do projeto é copiado para o contêiner, e o ambiente é configurado para rodar a aplicação (Bafilli, 2024).

Para o banco de dados PostgreSQL, é criada uma pasta chamada *Database* na raiz do projeto. Dentro dessa pasta, é necessário incluir um arquivo de *backup* do banco de dados, que será usado para inicializar o contêiner com as tabelas e os dados necessários para a aplicação.

Para configurar o *front-end* com Flutter no Docker, é necessário criar um *Dockerfile* específico que defina o ambiente do Flutter, geralmente baseado em uma imagem oficial, como a google/flutter. Esse arquivo deve configurar todas as dependências e ferramentas necessárias, incluindo o *SDK* do Flutter e ajustes no ambiente de trabalho. Após configurar o *Dockerfile*, a criação da imagem do contêiner garante que o *front-end* funcione de forma isolada e com todos os recursos necessários, semelhante ao que é feito no *back-end* (Awari, 2023).

A integração entre *back-end*, *front-end* e banco de dados é feita usando um arquivo de configuração chamado *docker-compose.yml*, que organiza como esses serviços funcionarão juntos. Esse arquivo descreve como cada parte do sistema será iniciada, como os contêineres irão se comunicar e como as redes serão configuradas. Quando tudo está configurado corretamente, o sistema pode ser iniciado de forma automática, com todos os componentes conectados (Susnjara; Smalley, 2024).

Para manter o ambiente organizado, é importante realizar limpezas periódicas, garantindo que não fiquem arquivos ou configurações desnecessárias. Após a limpeza, o ambiente pode ser recriado com a versão mais recente do sistema, garantindo que todas as partes estejam atualizadas e funcionando corretamente.

Essa abordagem de usar Docker para configurar e integrar *back-end, front-end* e banco de dados permite um gerenciamento simplificado de ambientes de desenvolvimento e produção, além de garantir que todos os componentes da aplicação estejam rodando de maneira consistente e isolada, independentemente da máquina ou sistema operacional utilizado.

# CONCLUSÃO

A proposta deste trabalho consistiu no desenvolvimento de um software para apoiar a gestão municipal na administração de despesas dos órgãos públicos. Durante o processo de desenvolvimento, foi necessário realizar uma análise detalhada para identificar as informações mais relevantes para exibição e cadastro, a fim de criar uma solução que atendesse aos requisitos essenciais do setor de controladoria.

Além de possibilitar o cadastro de despesas, o sistema permite que os usuários acompanhem o status dessas despesas e visualizem, de forma ágil e prática, dashboards com dados relevantes, como orçamentos e informações sobre as instituições que consomem maiores recursos. Esse recurso facilita a análise e proporciona uma visão ampla e organizada das finanças públicas.

Para futuras evoluções, identificaram-se algumas melhorias que poderiam agregar valor ao sistema. Entre elas, destaca-se a criação de uma interface para geração de relatórios com periodicidades variadas — semanal, bimestral, trimestral, semestral e anual —, possibilitando uma visão detalhada dos gastos e orçamentos vigentes. Essa funcionalidade auxiliaria na tomada de decisões sobre alocação de recursos futuros, permitindo a identificação de oportunidades de economia e destacando os órgãos com maiores despesas e as justificativas para esses gastos. Outra melhoria proposta é a integração com ferramentas de análise de negócios (BI), como o Power BI, para aprimorar a visualização dos indicadores-chave e facilitar a gestão pública. Também está prevista a implementação de novos perfis de usuários no sistema, incluindo Funcionário e Visitante, ampliando a flexibilidade e acessibilidade do sistema para diferentes tipos de usuários.

No estágio atual, o software atende às necessidades essenciais da controladoria da prefeitura, configurando-se como uma ferramenta de apoio à gestão. Com as melhorias sugeridas, o projeto poderá evoluir, incorporando funcionalidades adicionais que promovam uma gestão mais eficiente e transparente das despesas públicas, beneficiando tanto a administração quanto a população.

# REFERÊNCIAS

AKEMI, A. **HTTP: um guia completo sobre o que é e como funciona o protocolo da web.** Disponível em: https://www.alura.com.br/artigos/http. 26 mar. 2024.

ALBERTO, M. **Flutter: o que é e tudo sobre o framework.** Disponível em: https://www.alura.com.br/artigos/flutter. 18 set. 2023.

ANICHE, M. **Orientação a Objetos e SOLID para Ninjas: Projetando Classes Flexíveis.** São Paulo: Casa do Código, 2011.

ARAÚJO, J. L. **Análise e Desenvolvimento de um Sistema para auxiliar a Gestão de Despesas Pessoais.** Disponível em:

https://repositorio.ifgoiano.edu.br/bitstream/prefix/3422/1/TCC\_JAQUELINE%20LOPES.pdf. 2018.

ASTAH. **The Best UML Diagramming Tool Available.** Disponível em: https://astah.net/products/astah-uml/. Acesso em: 10 set. 2024.

AWARI. **Docker: Acelere o desenvolvimento de aplicativos Flutter.** Disponível em: https://awari.com.br/docker-acelere-o-desenvolvimento-de-aplicativos-flutter/. 19 nov. 2023.

BAFILLI, B. **Conteinerização da API Web .NET Core 8 usando Docker.** Disponível em: https://medium.com/@brunobafilli\_20480/conteineriza%C3%A7%C3%A3o-da-api-web-net-core-8-usando-docker-3c9c04bde80b. 14 abr. 2024.

BARADEL, L. **MVC – O que significa essa sigla e suas camadas.** Disponível em: https://lucasbaradel.medium.com/mvc-o-que-significa-essa-sigla-e-suas-camadas-142615b78c81. 24 jan. 2024

BRASIL. **Constituição da República Federativa do Brasil de 1988.** *Diário Oficial da União*, Brasília, DF, 5 out. 1988. Disponível em: http://www.planalto.gov.br/ccivil\_03/constituicao/constituicao.htm. Acesso em: 15 set. 2024.

CONTROLLE. **Controle financeiro descomplicado para a sua empresa.** Disponível em: https://controlle.com/. Acesso em: 04 jun. 2024.

DART. **Paint your UI to life.** Disponível em: https://dart.dev/. Acesso em: 05 nov. 2024.

FIGMA. **Pense grande. Crie mais rápido.** Disponível em: <https://www.figma.com/>. Acesso em: 24 out. 2024.

FLUTTER. **Build for any screen.** Disponível em: https://flutter.dev/. 01 nov. 2024.

GITHUB. **Build and ship software on a single, collaborative platform.** Disponível em: https://github.com. Acesso em: 30 out. 2024.

GUEDES, R. **UML 2: Uma abordagem prática.** 3. ed. 2018.

HAROON, R. **The Gateway to Data — Understanding JDBC and ODBC in Programming.** Disponível em: https://medium.com/@remisharoon/the-gateway-to-data-understanding-jdbc-and-odbc-in-programming-4fa70ecf4c01. 15 jan. 2024.

HOLDSWORTH, J; KOSINSKI, M. **O que é segurança da informação?** Disponível em: https://www.ibm.com/br-pt/topics/information-security#:~:text=Seguran%C3%A7a%20da%20informa%C3%A7%C3%A3o%20%28InfoSec%29%20%C3%A9%20a%20prote%C3%A7%C3%A3o%20de,usu%C3%A1rios%20autorizados%2C%20permane%C3%A7am%20confidenciais%20e%20mantenham%20sua%20integridade. 24 jul. 2024.

JUNIOR, C. E. V. **Software para Gerenciamento e Controle de Despesas.** Disponível em: https://ric.cps.sp.gov.br/bitstream/123456789/684/1/20171S\_VELKEJUNIORCarlosEduardo\_OD0117.pdf. 2017.

JWT. **Introduction to JSON Web Tockes.** Disponível em: https://jwt.io/introduction. Acesso em: 8 nov. 2024.

LEITE, D. J. N. G. **EcoCalc: Sistema de gerenciamento de despesas residenciais e pessoais.** Disponível em: http://dspace.sti.ufcg.edu.br:8080/xmlui/bitstream/handle/riufcg/29249/DANIEL%20JOS%C3%89%20DI%20NAVARONNE%20GAUD%C3%8ANCIO%20LEITE%20-%20TCC%20ARTIGO%20CI%C3%8ANCIA%20DA%20COMPUTA%C3%87%C3%83O%20CEEI%202022.pdf?sequence=1&isAllowed=y. 2022.

MACHADO, F. N. R. **Banco de Dados: Projeto e Implementação.** 4. ed. São Paulo: Érica, 2020.

MARIANO, A. **Sistema Integrado de Gestão (SIG): o que é, como funciona e como aplicar na sua indústria.** Disponível em: https://www.nomus.com.br/blog-industrial/sistema-integrado-de-gestao-sig/. 25 jan. 2024.

MARTIN, R. C. **Arquitetura Limpa.** O Guia do Artesão para Estrutura e Design de Software. Tradução: Samantha Batista. Rio de Janeiro: Alta Books, 2020. Tradução de: Clean Architecture: A Craftsman’s Guide to Software Structure and Design.

MEINE, B. **Repository Pattern em aplicações .NET.** Disponível em: https://blog.balta.io/repository-pattern-em-aplicacoes-net/. 24 abr. 2024.

MEU DINHEIRO. **Sistema para controle financeiro online.** Disponível em: https://www.meudinheiroweb.com.br/. Acesso em 04 jun. 2024.

MICROSOFT. **Entity Framework.** Disponível em: https://learn.microsoft.com/pt-br/aspnet/entity-framework. 13 jul. 2023.

MICROSOFT. **Azure DevOps.** Disponível em: <https://azure.microsoft.com/pt-br/services/devops/>. Acesso em: 25 out. 2024.

MILANI, A. **PostgreSQL: guia do programador.** São Paulo: Novatec Editora, 2008.

MIRO. **O que é UML?** Disponível em: https://miro.com/pt/diagrama/o-que-e-uml/. 2024.

ORACLE. **O que é um sistema de gerenciamento financeiro?** Disponível em: https://www.oracle.com/br/erp/financials/what-is-financial-management-system/#:~:text=De%20forma%20simples%2C%20sem%20algum,entra%20e%20sai%20da%20empresa. Acesso em: 02 ago. 2023.

PM3. **Prototipação: o que é, tipos, benefícios e etapas do processo.** Disponível em: https://www.cursospm3.com.br/blog/prototipacao-o-que-e/. 14 out. 2023.

PM3. **Prototipação: o que é, tipos, benefícios e etapas do processo.** Disponível em: https://www.cursospm3.com.br/blog/prototipacao-o-que-e/. 14 out. 2023.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional.** 7. ed. Porto Alegre: AMGH, 2011.

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

PUB.DEV. **flutter\_modular 6.3.4.** Disponível em:

https://pub.dev/packages/flutter\_modular. 23 mai. 2024a.

PUB.DEV. **result\_dart 1.1.1.** Disponível em: https://pub.dev/packages/result\_dart. 28 mar. 2024b.

PUB.DEV. **dson\_adapter 1.2.0+2.** Disponível em:

https://pub.dev/packages/dson\_adapter. 11 abr. 2023.

PUB.DEV. **intl 0.20.1.** Disponível em: https://pub.dev/packages/intl. 28 nov. 2024c.

PUB.DEV. **http 1.2.2.** Disponível em: https://pub.dev/packages/http. 16 jul. 2024d.

ROGERS, Y; SHARP, H; PREECE, J. **Design de Interação – Além da interação humano computador.** 3. ed. Porto Alegre: Bookman, 2013.

ROSA, A. **Protótipo: entenda o que é, tipos, exemplos e como fazer na prática.** Disponível em: https://softdesign.com.br/blog/prototipo-baixa-e-alta-fidelidade/#Qual-a-diferenca-entre-um-wireframe-e-prototipo. 25 set. 2024.

SANTANA, B. **O Que é o Protocolo TCP/IP e Como Ele Funciona?** Disponível em: https://www.hostinger.com.br/tutoriais/tcp-ip. 01 ago. 2024.

SANTANA, F. **Entendendo os tipos de Widgets do Flutter.** Disponível em: https://medium.com/tableless/entendendo-os-tipos-de-widgets-do-flutter-de9bb9296bf6. 14 abr. 2019.

SILBERSCHATZ, A; KORTH, H. F; SUDARSHAN, S. **Sistema de Banco de Dados.** 6. ed. Rio de Janeiro: Elsevier, 2012.

SILVESTRE, G. **Controller e Service - Uma breve introdução.** Disponível em: https://dev.to/gabrielhsilvestre/controller-e-service-uma-breve-introducao-24hk. 25 mar. 2022.

SISTEMA SOMA. **Sistema de contas a pagar e receber.** Disponível em: https://sistemasoma.com.br/. Acesso em: Acesso em: 04 jun. 2024.

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

SUSNJARA, S; SMALLEY, I. **O que é Docker?** Disponível em: https://www.ibm.com/br-pt/topics/docker. 6 jun. 2024.

SUTHERLAND, J.; SUTHERLAND, J. J. **Scrum: A arte de fazer o dobro do trabalho na metade do tempo.** 1. ed. São Paulo: Sextante, 2019.

UNGER, R; CHANDLER, C. **O Guia para projetar UX.** Rio de Janeiro: Alta Books, 2009.

WAGNER, B. **Um tour pela linguagem C#.** Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/overview/. 09 mai. 2024.

WALKER, J. **Desvendando e entendendo as diferenças entre: DTO’s, interfaces e types – Um guia para escolher a abordagem certa.** Disponível em: https://medium.com/@jhonywalkeer/desvendando-e-entendendo-as-diferen%C3%A7as-entre-dtos-interfaces-e-types-um-guia-para-escolher-a-ddb7cd84eb47. 01 ago. 2023.

1. Do inglês: *Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, Dependency Inversion Principle -* Princípio da responsabilidade única, Princípio Aberto-Fechado, Princípio da substituição de Liskov, Princípio da Segregação da Interface, Princípio da inversão da dependência. [↑](#footnote-ref-1)
2. Do inglês: *Data Transfer Objects -* Objeto de Transferência de Dados [↑](#footnote-ref-2)
3. Do inglês: *Atomicity, Consistency, Isolation, Durability -* Atomicidade, Consistência, Isolamento e Durabilidade [↑](#footnote-ref-3)
4. Do inglês: *Structured Query Language -* Linguagem de consulta estruturada [↑](#footnote-ref-4)
5. Do inglês: *Scrimmage -* Escaramuça [↑](#footnote-ref-5)
6. Do inglês: *Application Programming Interface -* Interface de Programação de Aplicações [↑](#footnote-ref-6)
7. Do inglês: *Hypertext Transfer Protocol -* Protocolo de Transferência de Hipertexto [↑](#footnote-ref-7)
8. Do inglês: *Hypertext Transfer Protocol Secure.-* Protocolo de Transferência de Hipertexto Seguro [↑](#footnote-ref-8)
9. Do inglês: *Transmission Control Protocol/Internet Protocol -* Protocolo de Controle de Transmissão/Protocolo de Internet [↑](#footnote-ref-9)
10. Do inglês: *Java Database Connectivity -* Conectividade com Banco de Dados em Java [↑](#footnote-ref-10)
11. Do inglês: *Open Database Connectivity -* Conectividade Aberta com Banco de Dados [↑](#footnote-ref-11)