UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO

FACULDADE DE CIÊNCIAS DE BAURU

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Trabalho de conclusão de Curso

Carlos Alberto de Oliveira Nackamura Junior

CAMPUS ISSUE MANAGEMENT(CIM): SISTEMA PARA RELATAR E GERIR INCIDENTES NO CAMPUS

Bauru, 2025

UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO

FACULDADE DE CIÊNCIAS DE BAURU

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Trabalho de Conclusão de Curso

Carlos Alberto de Oliveira Nackamura Junior

CAMPUS ISSUE MANAGEMENT (CIM): SISTEMA PARA RELATAR E GERIR INCIDENTES NO CAMPUS

Trabalho apresentado como exigência parcial para a conclusão do Curso de Bacharelado em Sistemas de informação da Faculdade de Ciências – UNESP Campus Bauru.

Orientador: Prof. Dr. José Remo Ferreira Brega

Bauru, 2025

N125c

Nackamura Junior, Carlos Alberto de Oliveira
Campus issue managment (CIM) : Sistema para relatar e gerir incidentes no Campus / Carlos Alberto de Oliveira Nackamura Junior. -- Bauru, 2025
52 f.

Trabalho de conclusão de curso (Bacharelado - Sistemas de Informação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientador: José Remo Ferreira Brega

1. Localização. 2. Sistema. 3. Gestão. 4. Incidentes. 5. Manutenção. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Carlos Alberto de Oliveira Nackamura Junior

CAMPUS ISSUE MANAGEMENT (CIM): SISTEMA PARA RELATAR E GERIR INCIDENTES NO CAMPUS

Trabalho apresentado como exigência parcial para a conclusão do Curso de Bacharelado em Sistemas de informação da Faculdade de Ciências – UNESP Campus Bauru.

Banca Examinadora

Prof. Dr. José Remo Ferreira Brega
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Dr. Higor Amario de Souza
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Dr. Leandro Aparecido Passos Junior
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

AGRADECIMENTOS

Agradeço, primeiramente, à minha mãe, Rosangela Soares de Camargo, por todo amor, apoio e incentivo incondicional ao longo da minha jornada acadêmica. Sua força e dedicação sempre foram minha maior inspiração.

Às minhas irmãs, Francielle Nackamura de Almeida e Valéria Camargo Nackamura, por estarem sempre ao meu lado, oferecendo palavras de encorajamento e apoio nos momentos em que mais precisei.

À minha namorada, Maria Alice Gonçalves, pela paciência, compreensão e motivação constante, mesmo nos momentos mais desafiadores. Sua presença foi fundamental para que eu pudesse chegar até aqui.

Ao meu orientador, Prof. Dr. José Remo, pelo valioso apoio, pela orientação dedicada e pelas contribuições que foram essenciais para o desenvolvimento deste trabalho.

Agradeço também aos meus colegas de classe pela convivência, companheirismo e pelas valiosas trocas de conhecimento ao longo do curso. Em especial, gostaria de mencionar Vinicius Prado Cavagna e Eduardo Conde Pires, cuja amizade, apoio e parceria tornaram essa jornada mais leve, significativa e enriquecedora. Levo comigo não apenas o aprendizado acadêmico, mas também as memórias e experiências compartilhadas com vocês.

- Carlos Alberto de Oliveira Nackamura Junior

RESUMO

Este trabalho visa desenvolver um sistema de gestão de incidentes para a Universidade, para melhorar a eficiência no atendimento e resolução de incidentes relacionados à infraestrutura. Atualmente, a comunicação e o registro de problemas são realizados informalmente e descentralizados dentro comunidade do câmpus, resultando em atrasos e falta de transparência no processo de resolução. A solução proposta é um sistema integrado com ferramentas que permitem a abertura, acompanhamento e fechamento de tickets dos incidentes relatados. O sistema inclui funcionalidades como a utilização de um mapa para identificar a localização dos incidentes, categorização e priorização das solicitações, bem como notificações em tempo real para os usuários e as equipes responsáveis.

Palavras-chaves: Gestão de incidentes, Transparência, Mapa de localização

ABSTRACT

The project aims to develop an incident management system for the university to improve the efficiency in handling and resolving incidents related to campus infrastructure and security. Currently, the communication and reporting of issues are done informally and in a decentralized manner by the campus community, resulting in delays and a lack of transparency in the resolution process. The proposed solution is an integrated system that will allow the opening, tracking, and closing of incident tickets. The system will include features such as the use of a map to identify the location of incidents, categorization and prioritization of requests, and real-time notifications for users and responsible teams.

Keywords: Incident management, Transparency, Location map

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface (Interface de Programação de Aplicações)
AWS	Amazon Web Services (Plataforma de serviços em nuvem)
BD	Banco de Dados
CIM	Campus Issue Management (Sistema para relatar e gerir incidentes no campus)
DOM	Document Object Model (Modelo de Objeto de Documento)
Fatec	Faculdade de Tecnologia
HTML	HyperText Markup Language (Linguagem de marcação para páginas web)
JWT	JSON Web Token (Token de autenticação baseado em JSON)
Leaflet	Biblioteca JavaScript para mapas interativos
MySQL	Sistema de gerenciamento de banco de dados relacional
NestJS	Framework para backend em Node.js
Node.js	Ambiente de execução JavaScript no servidor
OTRS	Open-source Ticket Request System (Sistema de gestão de chamados open-source)

React.js	Biblioteca JavaScript para interfaces de usuário (Meta)
REST	Representational State Transfer (Arquitetura para APIs web)
SLA	Service Level Agreement (Acordo de Nível de Serviço)
SQL	Structured Query Language (Linguagem para consulta a bancos de dados)
UML	Unified Modeling Language (Linguagem de Modelagem Unificada)
UNESP	Universidade Estadual Paulista "Júlio de Mesquita Filho"

SUMÁRIO

1. INTRODUÇÃO

Atualmente, a comunicação e o registro de problemas relacionados à manutenção em ambientes universitários ainda ocorrem, em grande parte, de maneira informal e descentralizada. Essa abordagem frequentemente resulta em falhas na comunicação entre usuários, prestadores de serviço e gestores, ocasionando atrasos na resolução, duplicidade de esforços e falta de transparência quanto ao andamento das solicitações.

Segundo Marshall (2016), a descentralização e a ausência de um sistema unificado comprometem a fluidez das informações e dificultam o gerenciamento eficiente das demandas. Por outro lado, a centralização das atividades de manutenção por meio de um sistema informatizado promove maior integração entre os envolvidos, permitindo uma comunicação clara, rastreável e ágil, além de reduzir falhas operacionais e aumentar a eficácia na resposta aos incidentes.

Além disso, recursos como geolocalização e anexação de imagens ao chamado agregam valor significativo ao processo. Conforme Noor et al. (2012), a possibilidade de registrar visualmente o problema e marcar sua localização exata contribui para a precisão no atendimento, evita ambiguidade na descrição, facilita a triagem das ocorrências e permite o direcionamento mais eficiente da equipe de manutenção. Isso é especialmente relevante em ambientes extensos e dinâmicos, como os campi universitários.

Dessa forma, a adoção de um sistema centralizado, que inclua a captura de imagens e a geolocalização dos incidentes, configura-se como uma solução estratégica para garantir eficiência na comunicação, maior rastreabilidade dos processos e melhoria contínua na qualidade do ambiente universitário, impactando positivamente, o conforto e a experiência de toda a comunidade acadêmica.

1.1 Detalhamento do problema

Os usuários do câmpus enfrentam atrasos na resolução de problemas que podem afetar seu ambiente de estudo. Problemas não resolvidos de maneira oportuna podem impactar negativamente a experiência acadêmica dos alunos, afetando tanto o desempenho acadêmico quanto o bem-estar geral.

Funcionários lidam com condições de trabalho subótimas devido à demora na resolução de problemas de manutenção. Professores e equipe administrativa são frequentemente confrontados com problemas que podem variar desde equipamento de escritório defeituoso até condições de trabalho insalubres, o que pode afetar a produtividade e a moral do corpo docente.

A incapacidade de resolver rapidamente esses problemas pode levar a um ambiente de trabalho estressante e menos eficiente.

Equipes de manutenção podem ter dificuldade em gerenciar e priorizar os incidentes de forma eficiente sem um sistema centralizado. Sem uma plataforma única para registrar, acompanhar e resolver problemas, essas equipes são frequentemente sobrecarregadas e desorganizadas. Além disso, a ausência de notificações em tempo real pode resultar em uma resposta tardia a incidentes que exigem atenção imediata, exacerbando ainda mais a situação.

A falta de um sistema centralizado resulta em um uso ineficiente dos recursos de manutenção, aumentando os custos operacionais. Quando os recursos não são alocados de maneira eficiente, os problemas que poderiam ser resolvidos rapidamente acabam consumindo mais tempo e mão de obra. Além disso, a duplicação de esforços e a necessidade de retrabalho são consequências diretas da falta de um sistema integrado, onde os incidentes podem ser gerenciados de forma clara e ordenada. A ineficiência no uso dos recursos também pode resultar na necessidade de contratação de pessoal adicional ou no pagamento de horas extras, elevando ainda mais os custos operacionais da universidade (MORAES, 2022).

Problemas que demoram para ser resolvidos podem levar a danos maiores e custos adicionais. Um pequeno problema de manutenção que não é tratado prontamente pode se transformar em uma questão mais grave e dispendiosa. Por exemplo, uma simples infiltração de água pode, com o tempo, causar danos estruturais significativos se não for reparada rapidamente. O tempo prolongado de resolução também pode causar interrupções no funcionamento normal do campus, afetando as atividades acadêmicas e administrativas, o que, por sua vez, pode gerar custos indiretos substanciais.

A ausência de um sistema para a gestão de incidentes resulta em uma falta de transparência no processo de resolução de problemas. Os membros da comunidade universitária não têm uma visão clara do status dos incidentes relatados, o que pode gerar frustração e desconfiança. Sem transparência, é difícil para os gestores identificar e abordar rapidamente os gargalos no processo, o que impede a melhoria contínua dos serviços de manutenção. Além disso, a falta de visibilidade sobre os problemas e suas soluções pode comprometer a prestação de contas e dificultar a justificativa de investimentos necessários em infraestrutura (ALBUQUERQUE, 2025).

Quando os problemas de manutenção não são tratados de maneira eficaz, isso cria um ambiente de descontentamento entre os membros da comunidade universitária. A percepção pública de ineficiência e desorganização pode desencorajar potenciais alunos e funcionários a escolherem a universidade, impactando negativamente a sua capacidade de crescer e prosperar.

2 SOLUÇÕES EXISTENTES E OBJETIVOS DA PROPOSTA

Ao realizar uma pesquisa sobre soluções existentes para a gestão de incidentes, foram identificadas várias iniciativas que abordam os desafios relacionados à comunicação e ao registro de problemas de manutenção. Essas soluções, desenvolvidas tanto em contextos acadêmicos quanto comerciais, demonstram a eficácia de sistemas centralizados para otimizar esses processos. A seguir, são apresentadas algumas das soluções encontradas.

2.1 Chamados e Notificações FATEC Carapicuíba

A solução proposta consiste na implementação de um sistema informatizado que centraliza de forma eficiente a gestão de chamados e notificações relacionadas a incidentes no ambiente universitário. Esse sistema tem como principal objetivo proporcionar um acompanhamento mais detalhado, organizado e contínuo dos problemas reportados (FIGURA 1), permitindo que as demandas sejam registradas, monitoradas e resolvidas com maior agilidade e precisão. Além disso, ao centralizar as informações em uma única plataforma, promove-se uma significativa melhoria na eficiência operacional, uma vez que os responsáveis passam a ter acesso rápido e estruturado às ocorrências.

Do mesmo modo, a transparência no processo de gerenciamento é ampliada, visto que todos os envolvidos — desde os usuários finais até as equipes técnicas — conseguem visualizar o andamento e o histórico das ações tomadas. Dessa forma, o sistema contribui para otimizar a comunicação entre os diversos setores envolvidos, reduzindo ruídos na troca de informações e fortalecendo o compromisso com a qualidade e a prestação de serviços dentro da instituição.

Figura 1: Sistema de Chamados Fatec Carapicuíba

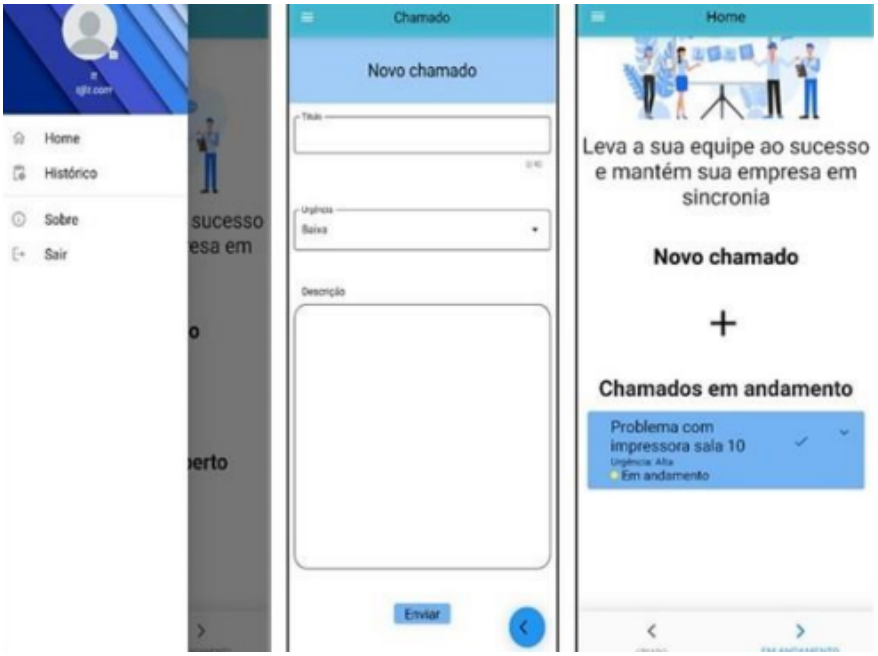
Fonte: (LEITE et al., 2024)

2.2 Aplicativo de Gerenciamento de Chamado usando Framework Flutter

Este projeto consiste no desenvolvimento de um aplicativo móvel voltado para o gerenciamento de chamados, utilizando o framework Flutter, uma tecnologia moderna e multiplataforma. O principal objetivo do aplicativo é possibilitar que os usuários registrem, acompanhem e administrem incidentes e solicitações (FIGURA 2) de forma prática, rápida e intuitiva, promovendo uma comunicação mais ágil entre os usuários e as equipes responsáveis pela resolução dos problemas.

A escolha por uma solução mobile se justifica pela crescente demanda por ferramentas acessíveis em tempo real, permitindo que os chamados sejam abertos e monitorados de qualquer local, a qualquer momento. Isso proporciona maior eficiência no atendimento, redução de tempo de resposta e melhora na qualidade dos serviços prestados.

Figura 2: Aplicativo de Gerenciamento de Chamado Usando Framework Flutter



Fonte: MORETTI et al., 2023

A escolha por uma solução mobile se justifica pela crescente demanda por ferramentas acessíveis em tempo real, permitindo que os chamados sejam abertos e monitorados de qualquer local, a qualquer momento. Isso proporciona maior eficiência no atendimento, redução de tempo de resposta e melhora na qualidade dos serviços prestados.

Além disso, o uso do *Flutter* garante uma única base de código para múltiplas plataformas (*Android* e *iOS*), o que otimiza o tempo de desenvolvimento e manutenção da

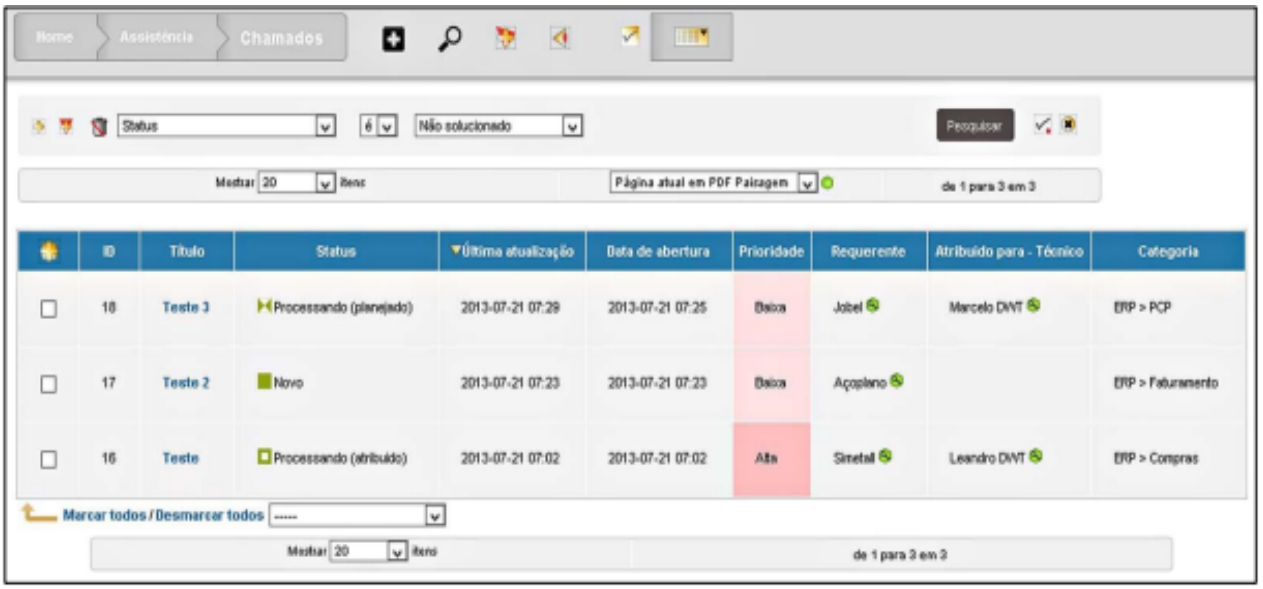
aplicação. A interface amigável e a usabilidade intuitiva do app contribuem para maior engajamento dos usuários e para a adoção efetiva do sistema no ambiente institucional.

2.3 Sistema para Gerenciamento de Chamados Técnicos de Perondi

O aplicativo oferece uma abordagem sistemática para o gerenciamento de chamados técnicos. A solução centraliza o registro de incidentes, permitindo um acompanhamento mais eficiente e uma resolução mais rápida dos problemas reportados (FIGURA 3). Além disso, o sistema melhora a transparência e a responsabilidade, proporcionando uma visão clara do status dos chamados e facilitando a análise de padrões recorrentes. Foco para incidente de T.I.

A “Implantação do Sistema de Chamados" (PEREIRA et al., 2016) na Universidade Federal de Santa Catarina focou na centralização do registro e do acompanhamento de problemas através do *Open-source Ticket Request System* (OTRS), uma ferramenta comercial amplamente utilizada para a gestão de incidentes e serviços

Figura 3: Sistema para Gerenciamento de Chamados Técnicos de Perondi



Fonte: PERONDI,2016

. Ela oferece uma ampla gama de funcionalidades, incluindo a abertura e o acompanhamento de tickets, categorização e priorização de solicitações, e notificações em tempo real. O OTRS permite uma gestão centralizada dos incidentes, ajudando a reduzir os tempos de resolução e a otimizar o uso dos recursos disponíveis.

2.4 Objetivos do Sistema

O *Campus Issue Management* (CIM) foi desenvolvido com o objetivo de oferecer uma solução mais completa e adaptada às necessidades específicas do ambiente universitário, buscando integrar em uma única plataforma funcionalidades que, muitas vezes, estão dispersas ou ausentes em outras ferramentas utilizadas para a gestão de incidentes. A ideia central é proporcionar uma interface acessível e amigável, que facilite o uso tanto por usuários finais quanto por prestadores de serviço e supervisores. Nesse sentido, o sistema se propõe a ir além do básico, como a simples abertura de chamados, incorporando recursos que otimizam a comunicação e o acompanhamento das ocorrências. Um exemplo disso é a interface web responsiva para abertura de incidentes, que está disponível no CIM, mas não em todos os sistemas analisados.

Outro diferencial importante é a implementação de um mapa interativo que permite a indicação precisa da localização dos incidentes, funcionalidade que visa facilitar a atuação das equipes responsáveis — recurso ainda não encontrado nas demais soluções avaliadas. Além disso, o sistema contempla a definição de papéis de usuário e o uso de SLA (Acordo de Nível de Serviço), promovendo uma organização mais clara das responsabilidades e dos prazos de atendimento. Assim, o CIM não pretende apenas reproduzir o que já existe em outros sistemas, mas sim integrar e aprimorar funcionalidades, com foco na usabilidade e na eficiência do processo de gestão de incidentes. A seguir, uma tabela comparativa

Tabela 1 - Tabela comparativa entre as soluções

Funcionalidade	Flutter App	Chamados Técnicos	Fatec Carapicuíba	OTRS	CIM
Interface web para abertura de incidentes	✅	❌	✅	❌	✅
Gestão de Incidentes	✅	✅	✅	✅	✅
Papéis de usuário	✅	✅	✅	❌	✅
Localização via mapa interativo	❌	❌	❌	❌	✅
SLA (Acordo de Nível de Serviço)	❌	✅	✅	✅	✅

Fonte: autoria própria

3 DESCRIÇÃO DO SISTEMA

O sistema de gestão de tickets para o ambiente universitário tem como objetivo principal registrar, acompanhar e resolver incidentes relacionados à manutenção no campus. A proposta é oferecer uma plataforma centralizada que promova uma comunicação eficiente entre os diferentes usuários — alunos, funcionários e administradores — e as equipes responsáveis pela resolução dos problemas reportados.

Entre as funcionalidades implementadas, destaca-se a possibilidade de abertura de chamados por parte dos usuários do campus, permitindo o registro de incidentes com informações detalhadas, como a descrição do problema, categoria, nível de prioridade, imagens ilustrativas da ocorrência e a localização geográfica obtida por meio de um mapa interativo. Após o registro, os usuários podem acompanhar o andamento dos chamados, com atualizações de status em tempo real e o recebimento de notificações automáticas sempre que houver alguma modificação relevante.

A gestão dos chamados inclui mecanismos de classificação e priorização, os quais podem ser realizados por usuários com perfil de funcionário, com base nas informações fornecidas pelo solicitante e em critérios definidos por acordos de nível de serviço (SLA). O sistema também contempla uma estrutura de controle de acesso baseada em papéis, permitindo a administração de diferentes tipos de usuários, com permissões específicas para cada perfil — aluno, funcionário ou administrador do sistema.

Todas as ações realizadas em cada chamado são registradas em um histórico de atividades, o que garante transparência e rastreabilidade no processo de atendimento. Além disso, o sistema oferece recursos para a geração de relatórios e análises gerenciais, possibilitando a identificação de padrões recorrentes e áreas críticas que demandam atenção por parte da administração universitária.

A interface do sistema foi projetada para ser intuitiva e acessível, visando proporcionar uma experiência de uso simplificada, independentemente do perfil do usuário. O software será desenvolvido como uma aplicação web, compatível com navegadores modernos, assegurando portabilidade, acessibilidade e facilidade de uso em diferentes dispositivos conectados à internet.

4 ESPECIFICAÇÃO DO SISTEMA

Nesta seção serão abordados os detalhamentos das especificações técnicas do software desenvolvido. Para uma melhor definição, ilustra-se e exemplifica-se apresentando a modelagem arquitetural e estrutural e a definição das relações base para a implementação.

4.1 Modelagem C4

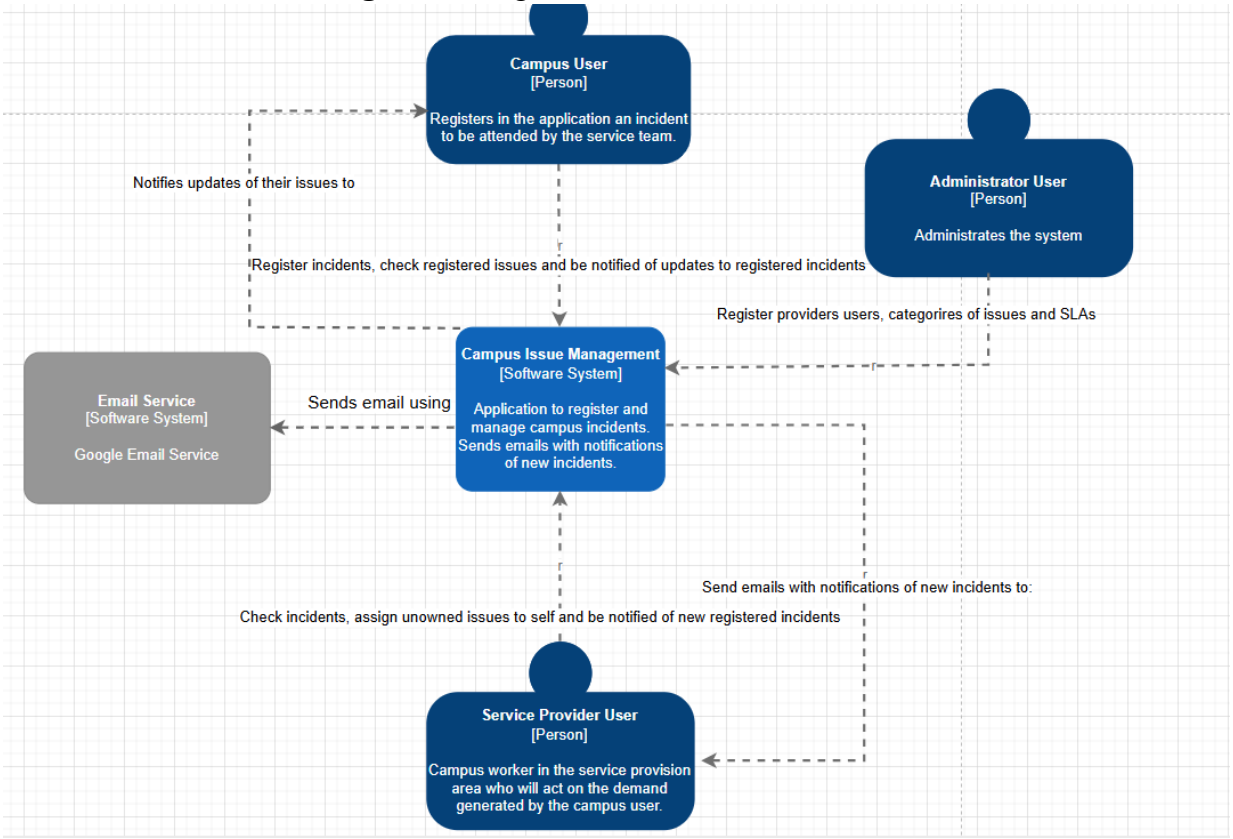
O C4 *Model* é uma técnica moderna de visualização da arquitetura de *software* que busca fornecer diferentes níveis de abstração para representar a estrutura de um sistema, facilitando a comunicação entre stakeholders técnicos e não técnicos. Essa abordagem organiza os diagramas em quatro níveis hierárquicos: contexto, contêineres, componentes e código-fonte, permitindo uma compreensão progressiva do sistema, desde sua interação com o ambiente externo até os detalhes internos de sua implementação.

A proposta do modelo é ser simples, clara e acessível, sendo especialmente útil em ambientes ágeis e colaborativos (BROWN, 2016).

Apresenta-se o Diagrama de Contexto do sistema, no formato C4. Esse nível permite visualizar de forma clara quem são os atores (usuários ou sistemas externos que interagem com o sistema) e em qual contexto ele está inserido (FIGURA 4).

Através da figura, identifica-se dois principais usuários: o *Campus User,* responsável por registrar incidentes ocorridos no campus, e o *Service Provider User*, que representa o trabalhador encarregado de atender essas demandas. O sistema atua como intermediário entre esses usuários, permitindo o cadastro, consulta e atualização de incidentes. Além disso, o diagrama evidencia a integração com sistemas externos como a biblioteca *Leaflet*, utilizada para exibir e associar a localização dos incidentes, e o *Email Service*, responsável por enviar notificações automáticas por *e-mail* aos usuários sobre novos registros ou atualizações. Esse modelo facilita a compreensão do papel do sistema dentro do ambiente organizacional e suas principais interfaces externas.

Figura 4: Diagrama de Contexto C4 *Model*

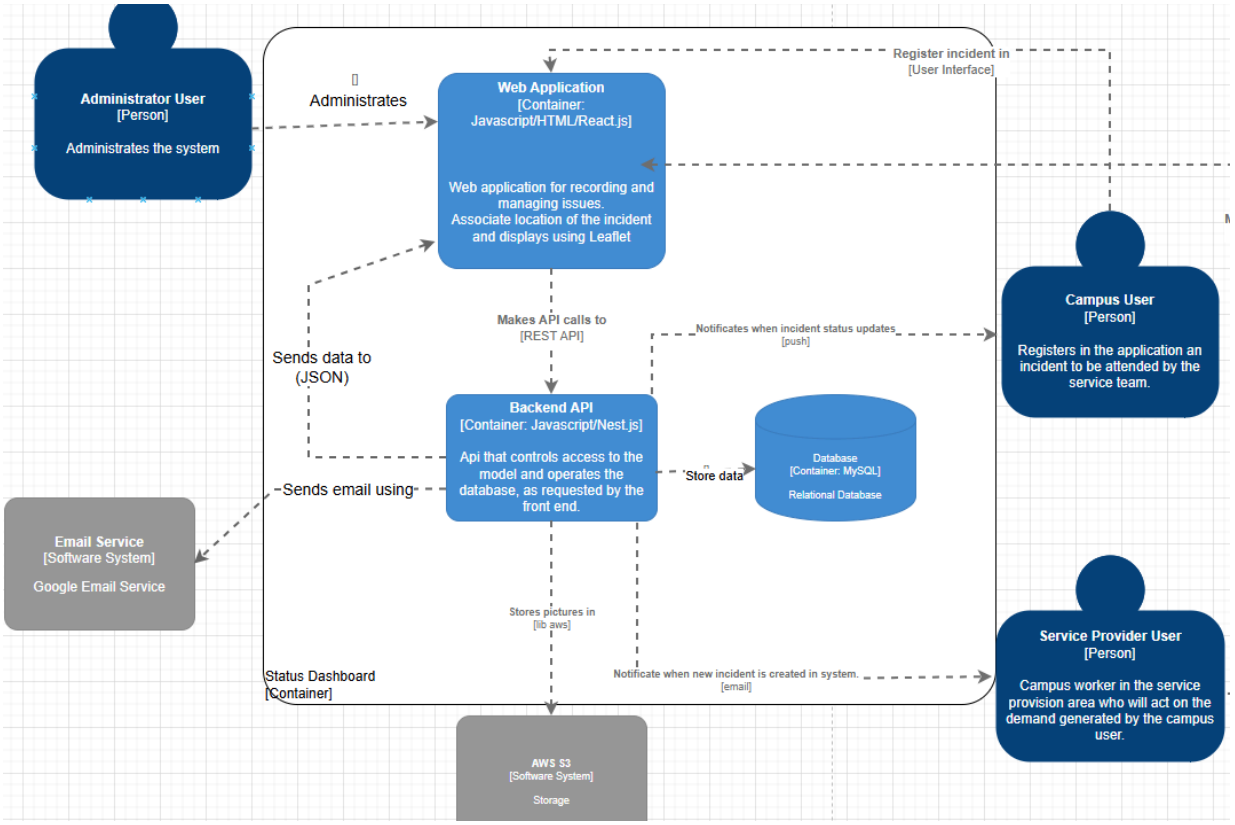


Fonte: autoria própria

O diagrama de contêineres é apresentado, segue o modelo C4 e descreve a arquitetura do sistema. O diagrama mostra a interação entre os principais usuários e os componentes de software da aplicação. (FIGURA 5) O sistema é composto por uma aplicação web desenvolvida com *Javascript/HTML/React.js*, que permite aos usuários do campus registrar e acompanhar incidentes, e aos prestadores de serviço gerenciarem as demandas recebidas. Essa aplicação se comunica com uma *Application Programming Interface* (API) *backend* desenvolvida em Javascript/Nest.js, responsável por processar as requisições, controlar o acesso aos dados e interagir com os demais serviços.

O *backend* utiliza um banco de dados relacional (MySQL) para armazenar as informações dos incidentes e também interage com o serviço de armazenamento *Amazon Web Services* (AWS) S3 para guardar imagens relacionadas às ocorrências. Notificações são enviadas via *e-mail* por meio de uma integração com um serviço de e-mail externo (como o *Google Email Service*), e atualizações de status são entregues aos usuários por push. Além disso, o sistema utiliza a biblioteca *Leaflet* para associar e exibir a localização geográfica dos incidentes registrados.

Figura 5: Diagrama de *Container* C4 *Model*



Fonte: autoria própria

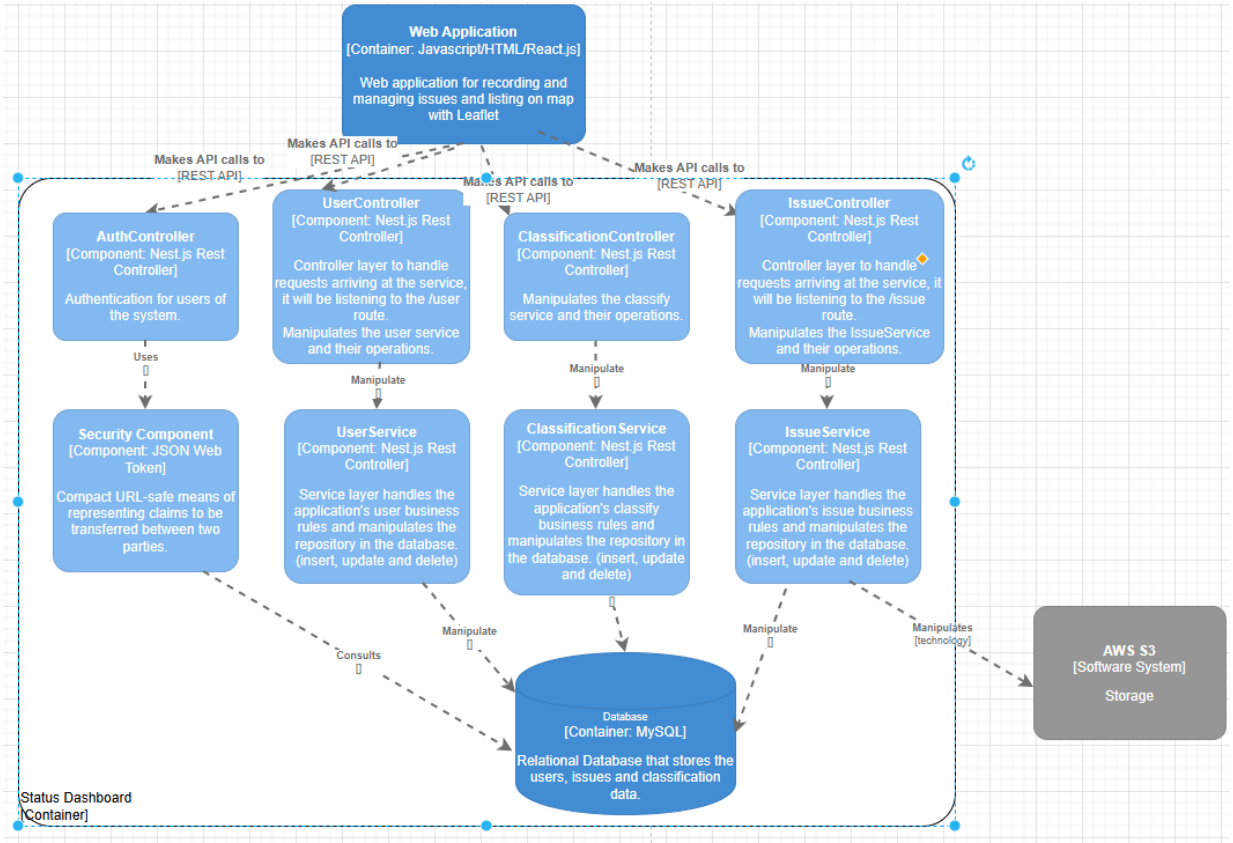
Om diagrama de componentes que representa a estrutura interna da aplicação no lado do servidor (*backend*), evidenciando como os diferentes módulos do sistema estão organizados e interagem entre si. Esse tipo de diagrama é fundamental para compreender a arquitetura lógica da aplicação, pois descreve os principais blocos funcionais responsáveis por processar requisições, executar regras de negócio e manipular dados (FIGURA 6).

O *backend* foi desenvolvido utilizando o *framework* Nest.js, que oferece uma estrutura robusta baseada em módulos e altamente aderente a boas práticas de engenharia de software, como a separação de responsabilidades e a reutilização de código. A arquitetura adotada segue o modelo *Representational State Transfer* (REST). *Controller*, no qual cada controlador representa um ponto de entrada específico para um domínio funcional da aplicação. Entre os principais domínios, destacam-se a autenticação de usuários (*AuthController*), a gestão de usuários (*UserController*), a classificação de incidentes (*ClassificationController*) e o gerenciamento dos chamados (*IssueController*).

Cada controlador é responsável por receber requisições HTTP da aplicação web, realizar validações iniciais e encaminhar essas solicitações para as respectivas camadas

de serviço. Essas camadas, como *UserService*, *ClassificationService* e *IssueService*, encapsulam a lógica de negócio da aplicação, realizando operações mais complexas como regras de validação, transformação de dados e persistência no banco de dados relacional MySQL, que é utilizado como repositório principal da aplicação.

Figura 6: Diagrama de Componentes C4 *Model*



Fonte: autoria própria

O componente de segurança, representado como Security Component, é responsável por garantir que todas as requisições sejam autenticadas e autorizadas corretamente. Para isso, é utilizado o padrão *JSON (Javascript Object Nature) Web Token* (JWT), que permite uma comunicação segura e controlada, assegurando que apenas usuários válidos tenham acesso às rotas protegidas.

Além dos componentes internos, o sistema também realiza integrações com serviços externos, o que demonstra a flexibilidade e a capacidade de expansão da arquitetura. Um exemplo é a integração com a AWS S3, serviço da *Amazon* utilizado para o armazenamento seguro das imagens anexadas aos chamados, evitando sobrecarga do servidor local e proporcionando escalabilidade na gestão de arquivos. Outro serviço essencial é a biblioteca *Leaflet*, utilizada para associar coordenadas geográficas aos

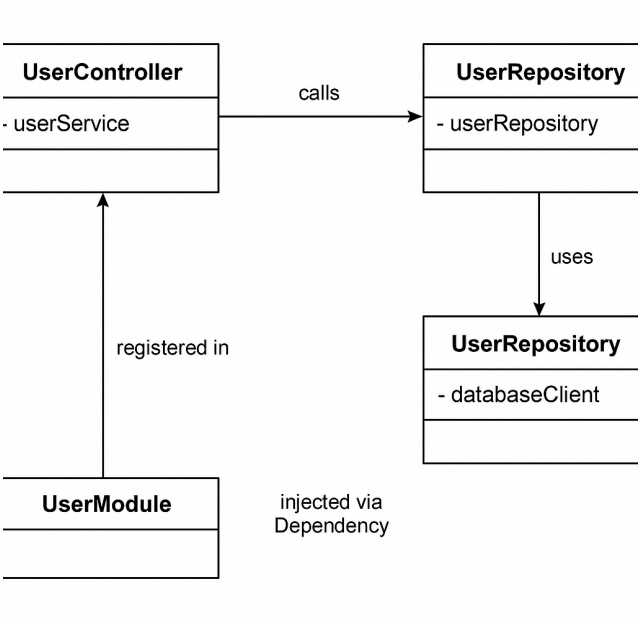
incidentes reportados e exibir de forma interativa a localização dos mesmos na interface do usuário.

Essa combinação de componentes internos bem estruturados com serviços externos consolidados resulta em uma arquitetura moderna, modular e altamente escalável. Além de facilitar a manutenção e o teste de cada parte do sistema de forma isolada, essa abordagem também garante maior clareza no fluxo de dados, melhorando a eficiência geral da aplicação e a experiência dos usuários finais.

O diagrama de Código tem como objetivo descrever a organização interna de um dos componentes do sistema. Esse nível de detalhamento permite compreender como os principais elementos da aplicação interagem entre si no nível do código-fonte.

Adota-se como referência uma funcionalidade comum, o gerenciamento de usuários, ilustrando as classes e suas dependências em uma estrutura do NestJS. A classe UserController é responsável por receber e tratar as requisições HTTP da aplicação cliente, funcionando como ponto de entrada da API. Ela se comunica diretamente com a classe UserService, que centraliza a lógica de negócio, realizando validações e procedimentos necessários antes da persistência ou consulta de dados (FIGURA 7).

Figura 7: Diagrama de Classes



Fonte: autoria própria

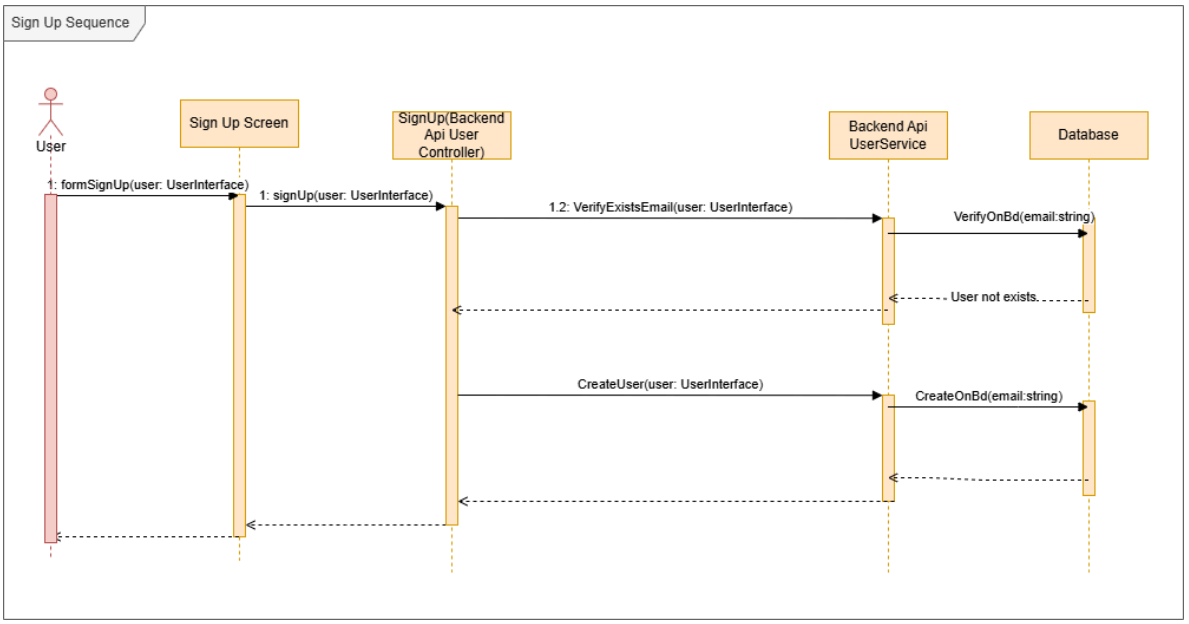
4.2 Diagramas de Sequência

O diagrama de sequência é um dos diagramas da UML (Unified Modeling Language) utilizado para modelar a dinâmica de um sistema, descrevendo a interação entre objetos em uma ordem temporal. Ele é particularmente útil para representar cenários de casos de uso, mostrando como as mensagens são trocadas entre os componentes do sistema ao longo do tempo, desde o início até a finalização de um processo (LARMAN, 2007). Neste trabalho, apresentarei nas Figuras 8, 9, 10 e 11 as sequências correspondentes às funcionalidades *Signup, Create Classification, Create Issue* e *Issue Visualization*, respectivamente. Através desses diagramas, é possível visualizar o fluxo de chamadas de métodos, facilitando a compreensão do comportamento do sistema e contribuindo para uma melhor definição da lógica de implementação (SOMMERVILLE, 2011).

4.2.1 Sequência de Registro

A sequência de registro evidencia a importância de uma interface web funcional e acessível, por meio da qual o usuário inicia sua interação com o sistema. Essa etapa inicial é essencial, pois é a partir dela que são desencadeadas todas as ações subsequentes no backend e no banco de dados. Ou seja, ao submeter o formulário de registro, o sistema realiza validações, armazena as informações e estabelece a base para a personalização da experiência do usuário. Essa mesma lógica de interação, em que uma ação na interface do usuário aciona processos internos no sistema, é aplicada nas demais sequências modeladas, como a abertura de chamados, a visualização do histórico de solicitações e a seleção de categorias (FIGURA 8).

Figura 8: Diagrama de Sequência do SignUp Sequence

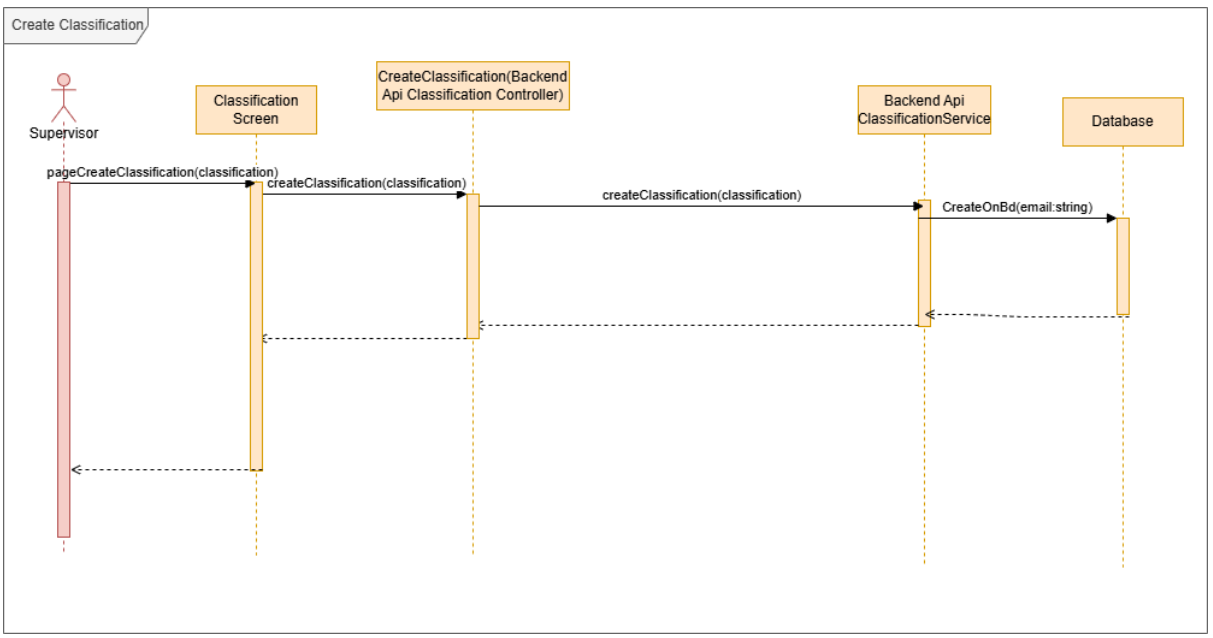


Fonte: autoria própria

4.2.2 Sequência de Criação de Classificação

Já a sequência de criação da classificação diz respeito ao processo de categorização dos chamados dentro do sistema, sendo uma etapa fundamental para garantir a organização e o direcionamento adequado das solicitações registradas. Essa funcionalidade permite que os administradores definam previamente as categorias, de forma que, ao abrir um chamado, o usuário possa selecionar a opção que melhor descreve a natureza do incidente. Essa categorização contribui diretamente para a triagem automatizada, otimizando a alocação das demandas às equipes responsáveis, além de facilitar análises futuras sobre os tipos de ocorrências mais frequentes. É importante destacar que esse fluxo de criação, edição e manutenção das classificações é exclusivo do administrador do sistema, garantindo maior controle, padronização e consistência nas opções apresentadas aos usuários. Dessa forma, busca-se evitar duplicidades, ambiguidades ou classificações imprecisas que possam comprometer a eficiência do atendimento. Em suma, a estruturação criteriosa desse processo fortalece a gestão da informação e reflete diretamente na qualidade dos serviços prestados no ambiente universitário (FIGURA 9).

Figura 9: Diagrama de Sequência da Criação da Classificação



Fonte: autoria própria

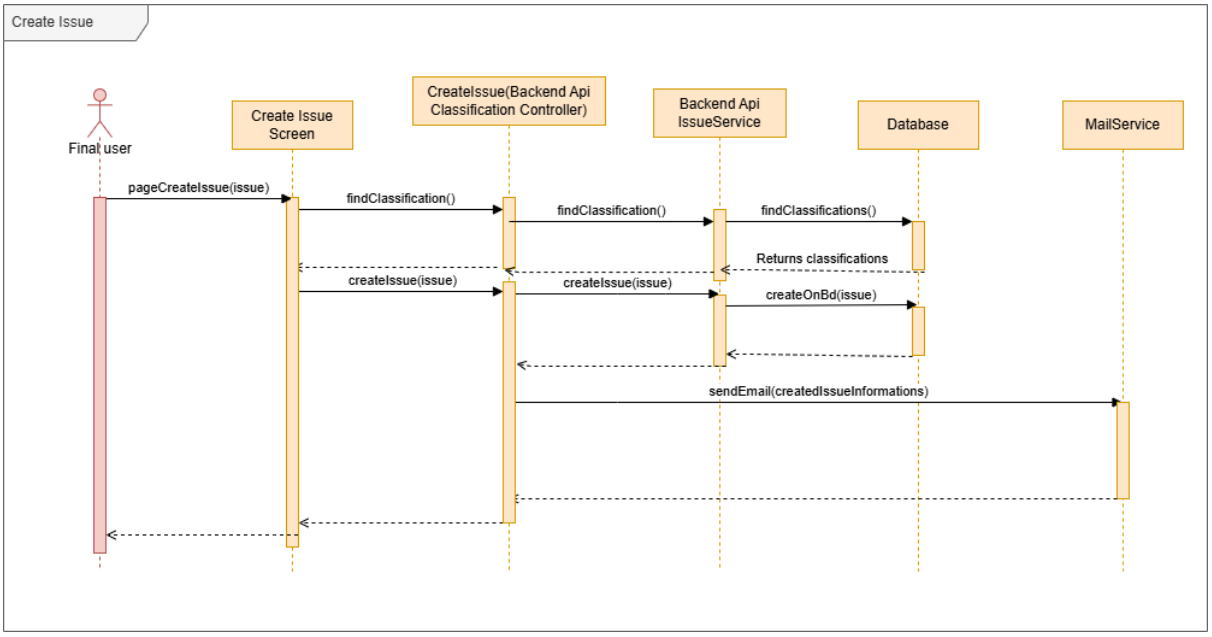
4.2.3 Sequência de Criação de Chamados

Podemos observar, por meio do diagrama apresentado, que o processo de criação de chamados é iniciado com a busca pelas categorias previamente cadastradas no sistema. Esse passo inicial é essencial para garantir que o usuário tenha acesso a um conjunto de opções organizadas e padronizadas, facilitando a correta classificação da ocorrência que deseja registrar.

Essa consulta rápida às categorias não apenas orienta o usuário durante o preenchimento do formulário, como também assegura que os dados inseridos estejam alinhados com a estrutura de atendimento estabelecida pela administração do sistema. Somente após essa etapa é que o sistema permite a efetiva criação do chamado, abrindo espaço para que o usuário forneça informações mais detalhadas sobre a solicitação.

Nesse momento do fluxo, o usuário é incentivado a anexar imagens que possam ilustrar melhor o problema relatado, bem como a indicar sua localização por meio de um recurso interativo de mapa, o que contribui significativamente para a agilidade e precisão no atendimento. Todo esse processo ocorre de forma sequencial e integrada (FIGURA 10), permitindo que os dados sejam enviados de maneira estruturada ao *backend*, onde serão processados e armazenados no banco de dados para posterior análise e resolução pela equipe responsável. Esse fluxo demonstra a preocupação com a experiência do usuário e com a eficiência na coleta de informações relevantes para a gestão dos chamados.

Figura 10: Diagrama de Sequência da Criação do Chamado



Fonte: autoria própria

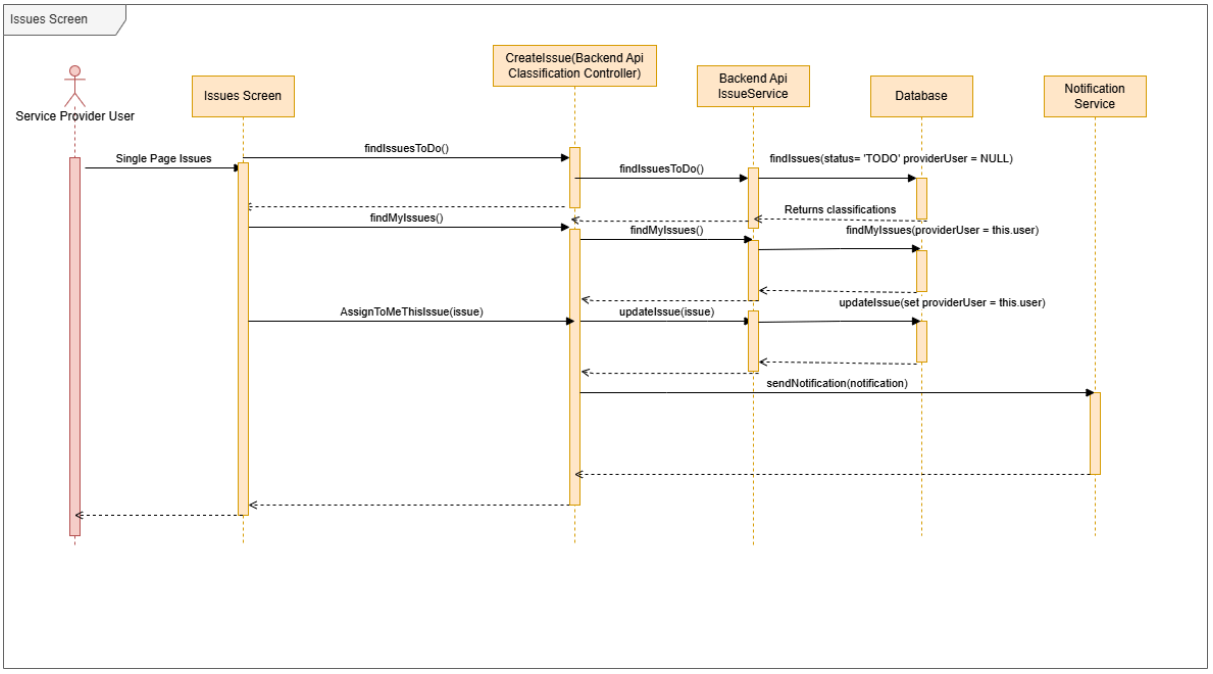
4.2.4 Sequência de Visualização dos Chamados

O diagrama de sequência apresentado representa de forma clara o fluxo de interação entre o usuário prestador de serviço e os diversos componentes do sistema durante a gestão de chamados. Ao acessar a tela de visualização de chamados, o front-end inicia a comunicação com o *back-end* para recuperar dois conjuntos de informações essenciais: os chamados que ainda estão pendentes de atribuição e aqueles que já foram assumidos pelo próprio usuário. Essas requisições são processadas por camadas intermediárias da aplicação, que por sua vez realizam consultas ao banco de dados para retornar os dados mais atualizados. Essa interação entre interface e servidor é fundamental para garantir que o usuário visualize, em tempo real, a situação atual dos chamados e possa tomar decisões com base em informações consistentes.

Uma vez apresentados os dados na tela, o usuário tem a possibilidade de assumir um chamado específico. Essa ação desencadeia um novo fluxo entre o *front-end* e o *back-end*, que atualiza os dados do chamado no banco de dados, vinculando-o ao prestador de serviço em questão. Como consequência dessa atualização, o sistema também aciona o serviço de notificações, garantindo que os responsáveis e interessados sejam informados sobre a nova atribuição.

Todo esse processo (FIGURA 11) demonstra não apenas a integração eficiente entre os diferentes módulos do sistema — interface, lógica de negócio, banco de dados e serviço de notificação —, mas também a preocupação com a fluidez da experiência do usuário e com a integridade dos dados em tempo real.

Figura 11: Diagrama de Sequência da Visualização dos Chamados



Fonte: autoria própria

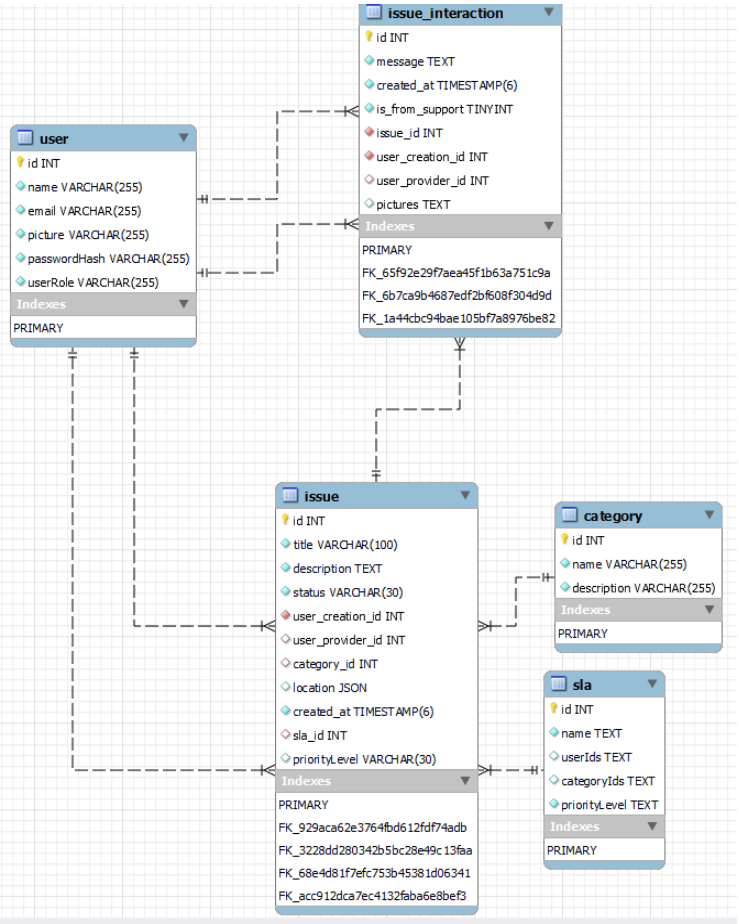
4.3 Modelo Entidade Relacionamento

O Modelo Relacional (MR) é uma abordagem lógica de modelagem de dados que organiza as informações em estruturas tabulares chamadas de relações, onde cada tabela representa uma entidade ou um relacionamento identificado durante o processo de modelagem conceitual. Baseado na teoria dos conjuntos, o modelo relacional visa garantir a integridade e a consistência dos dados por meio do uso de chaves primárias, estrangeiras e restrições, sendo amplamente adotado nos sistemas de gerenciamento de banco de dados relacionais (SGBDR).

O modelo relacional do sistema de gestão de incidentes (FIGURA 12), representando de forma lógica e normalizada a estrutura do banco de dados. O modelo é composto por seis tabelas principais: *user, issue, issue_interaction, category, sla*. A tabela *user* armazena dados dos usuários, como nome, e-mail, foto, função e credenciais de autenticação. A tabela *issue* representa os chamados ou incidentes, com atributos como título, descrição, status, localização (em formato JSON), data de criação, nível de prioridade e referências aos usuários que criaram e atenderam ao chamado. A tabela *issue_interaction* registra as interações relacionadas a cada incidente, contendo informações como data, conteúdo e identificação do emissor.

Esse modelo relacional reflete as decisões tomadas na modelagem conceitual e prepara a estrutura dos dados para sua posterior implementação no modelo físico, garantindo organização, integridade referencial e facilidade de manutenção no sistema.

Figura 12: Modelo Entidade Relacionamento CIM



Fonte: autoria própria

5 TECNOLOGIAS PESQUISADAS E ESCOLHIDAS

Para o desenvolvimento do sistema, foram avaliadas diversas tecnologias que pudessem atender aos requisitos funcionais, de desempenho, manutenção e escalabilidade da aplicação. As escolhas consideraram tanto a adequação técnica quanto a maturidade, a comunidade de suporte e a documentação disponível. A seguir, são detalhadas as principais tecnologias

adotadas em cada camada do sistema, justificando sua seleção com base em suas características e vantagens específicas.

5.1 Plataforma de Desenvolvimento

A biblioteca React.js foi adotada como base da plataforma de desenvolvimento no frontend. Desenvolvida pelo Facebook, a biblioteca permite a criação de interfaces de usuário reativas, com foco em aplicações de página única (SPA – *Single Page Applications*). Sua estrutura baseada em componentes reutilizáveis e o uso do DOM virtual proporcionam alto desempenho e facilidade de manutenção. Além disso, conta com uma vasta comunidade e ampla documentação. Segundo a documentação oficial, o React "torna simples criar interfaces de usuário interativas" (META, 2024).

5.2 Backend

No backend, optou-se pela utilização do Node.js, juntamente com o *framework* NestJS. O Node.js é um ambiente de execução JavaScript que opera de forma assíncrona e orientada a eventos, o que permite lidar eficientemente com aplicações de alta concorrência, como sistemas de gestão de incidentes.

Conforme destaca a documentação oficial, o Node.js foi projetado "para construir aplicações de rede escaláveis" (OPENJS FOUNDATION, 2024). O NestJS, por sua vez, oferece uma arquitetura modular baseada em controladores e serviços, promovendo organização e reutilização de código. Ele é descrito como um framework que permite "construir aplicações eficientes e escaláveis no lado do servidor com Node.js" (KAMIL, 2024).

5.3 Banco de Dados

O sistema utiliza o banco de dados *MySQL*, uma solução relacional amplamente consolidada. A escolha se baseia na necessidade de relações complexas entre entidades (usuários, incidentes, classificações), que são melhor representadas em um modelo relacional do que em um banco orientado a documentos como o *MongoDB*. Além disso, o *MySQL* oferece suporte a transações com propriedades Atomicidade, Consistência, Isolamento e Durabilidade

(ACID), integridade referencial e bom desempenho para leitura e escrita em larga escala. De acordo com a documentação oficial, o *MySQL* é considerado o banco de dados open source mais popular do mundo, destacando-se por “desempenho comprovado, confiabilidade e facilidade de uso” (ORACLE, 2024).

5.4 Docker

Para garantir consistência entre os ambientes de desenvolvimento, teste e produção, foi utilizada a plataforma Docker. A tecnologia de contêineres permite empacotar a aplicação e suas dependências de forma isolada, evitando conflitos e tornando o processo de implantação mais ágil e seguro. Além disso, contribui para a escalabilidade do sistema e facilita a automação de processos de entrega contínua (CI/CD). A própria documentação oficial afirma que o Docker possibilita “separar aplicações da infraestrutura, permitindo entregas rápidas” (DOCKER INC., 2024).

6 TESTES

A cobertura de testes é fundamental para avaliar a qualidade dos testes realizados em um sistema. Representa a proporção do código fonte que foi executada durante a execução dos testes, podendo incluir cobertura de linhas, de ramos condicionais, de funções, entre outros critérios (BECK, 2003). Um alto nível de cobertura de testes está associado a uma maior confiança na robustez do software, pois reduz a probabilidade de erros não detectados (MYERS; SANDLER; BADGETT, 2011). No contexto deste projeto, a cobertura de testes foi monitorada para assegurar que todas as funcionalidades implementadas fossem efetivamente testadas, contribuindo para a confiabilidade e a manutenção do sistema.

Para os testes unitários, que visam validar o funcionamento isolado de componentes ou funções específicas, foi utilizado predominantemente o *framework* Jest. Jest é uma ferramenta moderna e amplamente adotada para testes em ambientes JavaScript, conhecida por sua facilidade de configuração, velocidade na execução dos testes e recursos integrados para análise de cobertura (KENT C. DODDS, 2020).

O relatório de testes automatizados, gerado por meio da ferramenta, resume os percentuais de cobertura de código alcançados durante a execução dos testes da aplicação. As

métricas exibidas referem-se à cobertura de instruções executáveis (*Statements*), ramificações lógicas (*Branches*), funções ou métodos (*Functions*) e linhas de código (*Lines*). Os dados estão organizados por diretório e arquivo, permitindo uma análise detalhada de cada módulo testado.

Figura 13: Resultado dos testes automatizados

```
Test Suites: 12 passed, 12 total
Tests:       24 passed, 24 total
Snapshots:   0 total
Time:        1.745 s
Ran all test suites.

--------------------------|---------|----------|---------|---------|
File                      | % Stmts | % Branch | % Funcs | % Lines |
--------------------------|---------|----------|---------|---------|
All files                 |   92.85 |    86.66 |   91.66 |   92.85 |
 src/auth                 |     100 |      100 |     100 |     100 |
  auth.controller.ts      |     100 |      100 |     100 |     100 |
  auth.service.ts         |     100 |      100 |     100 |     100 |
 src/category             |     100 |    83.33 |     100 |     100 |
  category.controller.ts  |     100 |    83.33 |     100 |     100 |
  category.service.ts     |     100 |    83.33 |     100 |     100 |
 src/issue                |   83.33 |    75.00 |   83.33 |   83.33 |
  issue.controller.ts     |   85.71 |       75 |   83.33 |   85.71 |
  issue.service.ts        |   81.81 |       75 |   83.33 |   81.81 |
 src/issue-interaction    |   87.50 |    75.00 |     100 |   87.50 |
  issue-interaction.controller.ts |   83.33 |      100 |     100 |   83.33 |
  issue-interaction.service.ts    |     100 |       50 |     100 |     100 |
 src/sla                  |   90.00 |       80 |     100 |   90.00 |
  sla.controller.ts       |   80.00 |       75 |     100 |   80.00 |
  sla.service.ts          |     100 |       85 |     100 |     100 |
 src/user                 |     100 |      100 |     100 |     100 |
  user.controller.ts      |     100 |      100 |     100 |     100 |
  user.service.ts         |     100 |      100 |     100 |     100 |
```

Fonte: autoria própria

Os módulos principais — como autenticação, usuários, chamados, interações, categorias e acordos de nível de serviço — apresentaram cobertura consistente em seus respectivos controladores e serviços. Essa abrangência indica que as funcionalidades foram suficientemente testadas, o que contribui para a redução de falhas em tempo de execução e para o aumento da confiabilidade geral do sistema (FIGURA 13).

7 IMPLEMENTAÇÃO E USO

Neste capítulo é apresentado detalhadamente as tecnologias e algoritmos utilizados para o desenvolvimento, bem como as funcionalidades do aplicativo *Campus Issue Management* (CIM) - e como pode ser utilizado. A seguir, expõe-se a lógica do sistema e sua identidade visual (FIGURA 14).

Figura 14: Logotipo do CIM

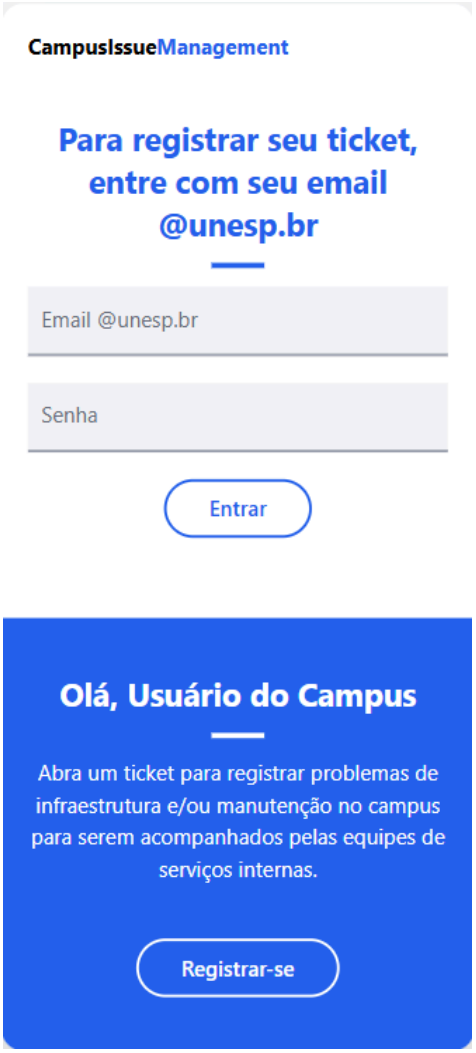
Fonte: autoria própria

7.1 Autenticação

Foram definidas três rotas distintas de login, cada uma correspondente a uma tela específica do sistema, de acordo com os diferentes papéis dos usuários. A primeira rota, */user*, direciona para a tela de login do usuário final, responsável pelo registro e acompanhamento de solicitações ou incidentes no campus (FIGURA 15). A segunda rota, */provider*, conduz à tela de login do prestador de serviço, que acessa o sistema para visualizar, atender e atualizar os incidentes atribuídos. Por fim, a terceira rota, */master*, refere-se à tela de login do administrador, cuja função é gerenciar os prestadores de serviço, supervisionar os chamados registrados e administrar o sistema como um todo.

No código dessas telas, os dados de login inseridos (usuário e senha) são enviados ao backend, onde são validados. As senhas são armazenadas de forma segura utilizando hash criptográfico (*passwordHash*), garantindo a integridade e a confidencialidade das credenciais dos usuários. Em caso de autenticação bem-sucedida, o backend retorna um token de autenticação no formato JWT (*JSON Web Token*), que é armazenado no frontend para manter a sessão ativa e permitir o acesso às funcionalidades autorizadas conforme o perfil do usuário.

Figura 15: Autenticação

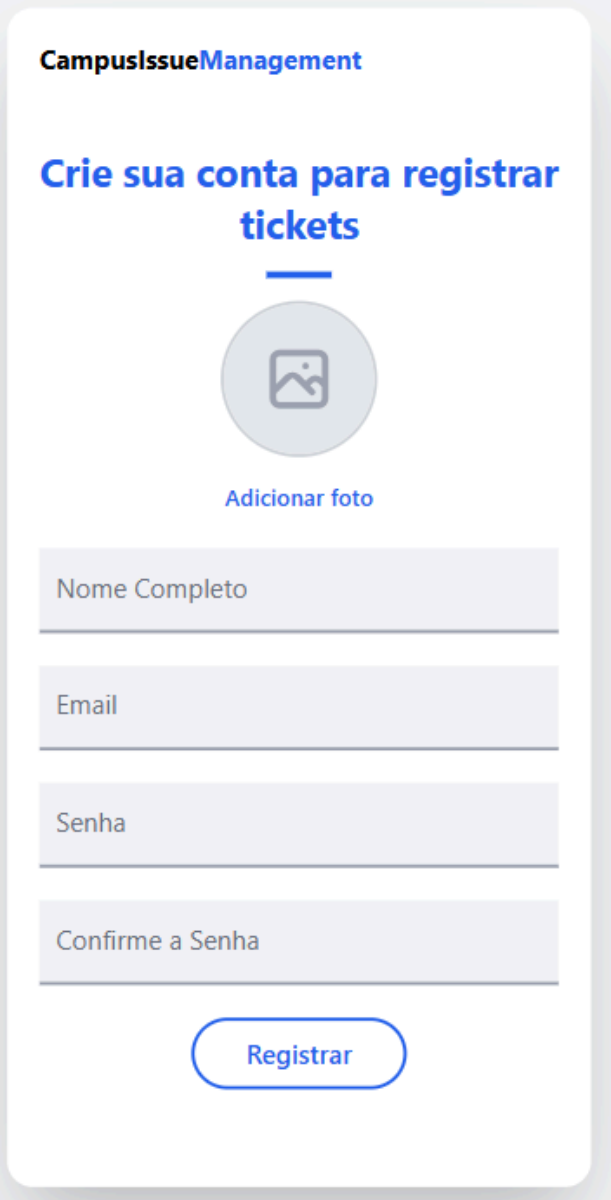


Fonte: autoria própria

7.2 Cadastro

O usuário final pode realizar seu cadastro no sistema por meio do botão "Registrar-se" (FIGURA 16). Ao acessar essa tela, é apresentado um formulário destinado à inserção dos dados necessários para o registro do usuário. Como critério de validação, o sistema exige que o endereço de e-mail informado pertença ao domínio @unesp.br, restringindo o acesso exclusivamente a membros da comunidade universitária. No código da tela também está preparado para receber nesse formulário foto para associar ao perfil do usuário.

Figura 16: Cadastro de usuário



Fonte: autoria própria

7.3 Registrar Chamado

Na tela de registro chamado é onde o usuário final pode enfim registrar sua solicitação ou incidente encontrado no campus (FIGURA 17). Nesta tela é disponibilizado um formulário com o título do chamado, a descrição da ocorrência. São carregados os chamados cadastrados no backend e apresentados para seleção do usuário.

Via código, são solicitadas permissões no navegador para coletar informações do usuário (localização) e também para uso da câmera do dispositivo para captura de imagem relacionada a solicitação ou incidente.

Figura 17: Registrar chamado

Fonte: autoria própria

7.4 Chamados do Usuário

A tela "Meus Chamados" (FIGURA 18) tem como objetivo permitir que o usuário acompanhe todos os chamados que ele próprio registrou no sistema, funcionando como um painel de controle pessoal. Nela, são listadas informações essenciais de cada chamado, como o título (exibido em destaque), o status atual (indicado por um selo, como "*TO DO*"), a categoria (por exemplo, "Infraestrutura predial" ou "Mecânica"), uma descrição resumida do problema e a data e hora em que o chamado foi aberto.

A tela oferece um botão de atalho chamado "Novo Chamado", permitindo que o usuário registre novas solicitações de forma rápida. A interface foi pensada para ser clara e funcional, facilitando o acompanhamento e a organização dos chamados por parte do usuário.

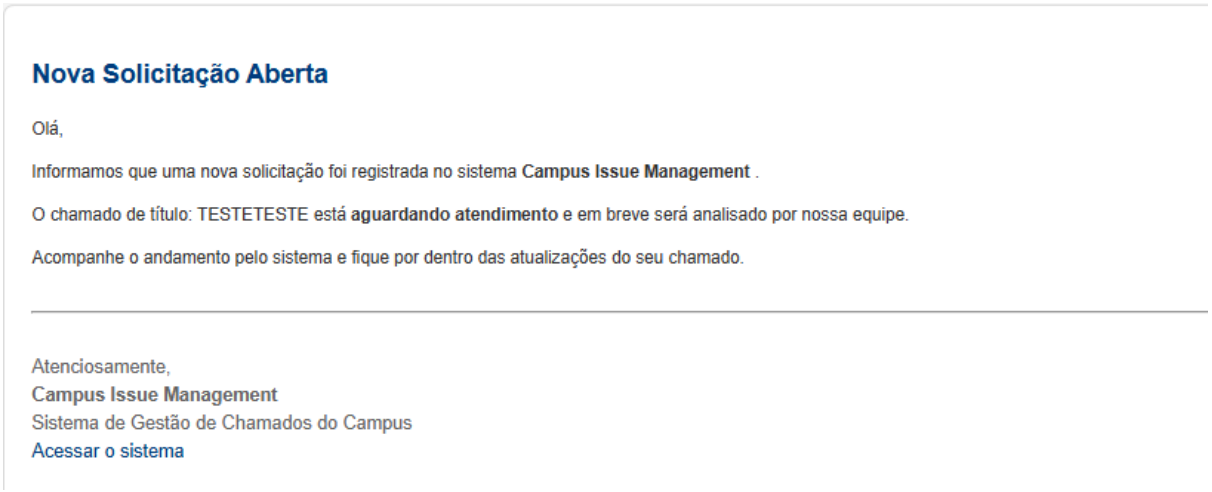
Figura 18: Meus chamados (abertos pelo usuário)

Fonte: autoria própria

7.5 Envio de E-mail

Quando um novo chamado é registrado no sistema, um mecanismo automático é acionado para garantir que todas as partes envolvidas sejam devidamente notificadas. Nesse processo, é enviado um e-mail de confirmação para o endereço eletrônico previamente cadastrado pelo usuário responsável pela abertura do chamado. Essa mensagem contém informações essenciais sobre o incidente reportado, como exemplificado na Figura 19.

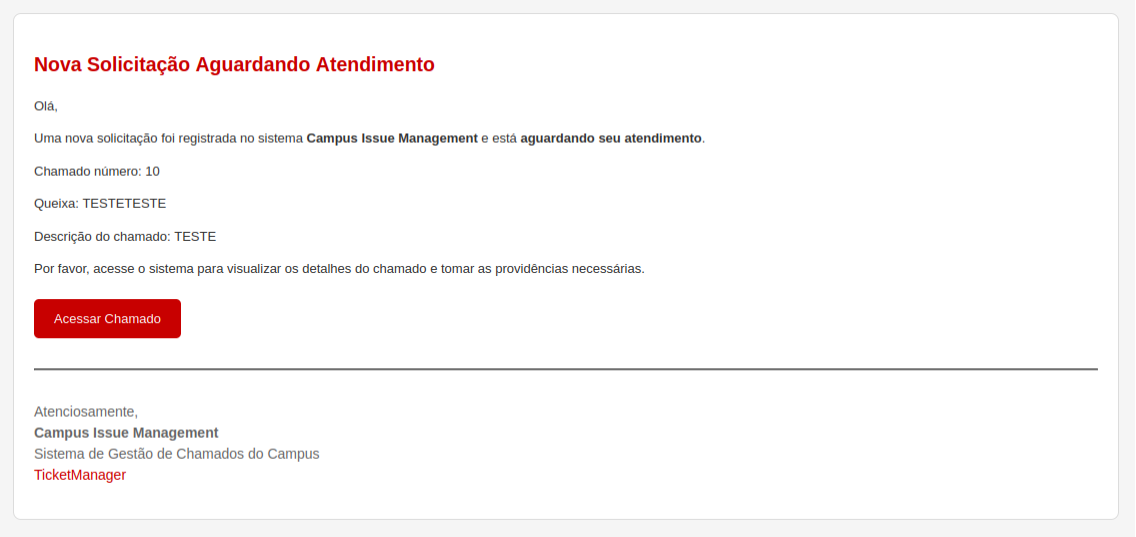
Figura 19: E-mail enviado para o usuário de abertura



Fonte: autoria própria

Além disso, de forma simultânea, o sistema realiza o envio de uma notificação via e-mail para a equipe prestadora de serviços correspondente à categoria do problema informado. O objetivo dessa comunicação é informar de maneira clara e imediata a existência de um novo chamado que demanda atenção, fornecendo todos os dados necessários para que os responsáveis possam realizar uma análise inicial do ocorrido, como ilustrado na Figura 20.

Figura 20: E-mail enviado para o prestador de serviço durante abertura de chamado



Fonte: autoria própria

7.6 Chamados Abertos (Visão Prestador)

Exibe uma lista de chamados georreferenciados, sendo utilizada principalmente por responsáveis pelo atendimento ou pela manutenção. Ela é composta por um mapa interativo (utilizando a biblioteca *Leaflet* com base no *OpenStreetMap*) que apresenta a localização exata de cada chamado por meio de marcadores, além de uma listagem lateral com os chamados disponíveis para ação. Cada item da lista exibe o título do chamado, uma descrição curta, a data de criação, o status (como "*TO-DO*") e dois botões: "Atribuir a mim", para que o usuário assuma a responsabilidade pelo atendimento do chamado, e "Detalhes", para visualizar mais informações.

Um diferencial importante desta tela é que, ao clicar em qualquer chamado da lista, o mapa automaticamente centraliza e destaca a localização correspondente (FIGURA 21), facilitando a identificação visual do ponto exato onde a demanda foi registrada. Esta funcionalidade melhora o direcionamento da equipe técnica e torna a visualização mais eficiente, especialmente em ambientes com múltiplas ocorrências geograficamente distribuídas.

Figura 21: Tela Chamados Abertos

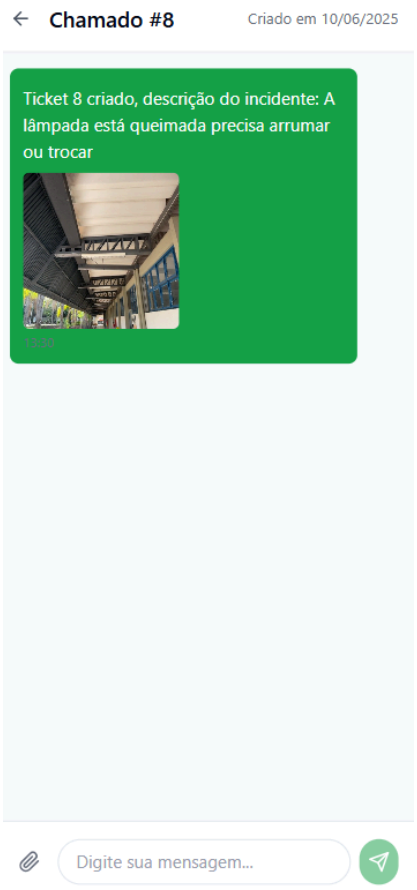
Fonte: autoria própria

7.7 Detalhes do Chamado

A tela permite ao usuário visualizar o histórico completo e a comunicação relacionada a um chamado específico. No topo da tela são exibidos o número identificador do chamado (por exemplo, "Chamado #8") e a data de criação (FIGURA 22). O conteúdo principal segue o estilo de um chat, onde são exibidas mensagens que registram as ações e interações ligadas ao chamado, como a descrição do incidente e imagens anexadas (por exemplo, uma foto da lâmpada queimada).

Cada mensagem contém um timestamp com o horário do registro. Ao final da tela, há um campo de texto, permitindo que os usuários responsáveis (como técnicos ou solicitantes) adicionem novas mensagens ou atualizações ao chamado em tempo real, promovendo uma comunicação direta e organizada entre os envolvidos na resolução do problema. Essa abordagem torna a gestão de chamados mais transparente e colaborativa, além de documentar o histórico completo da ocorrência.

Figura 22: Detalhes do chamado

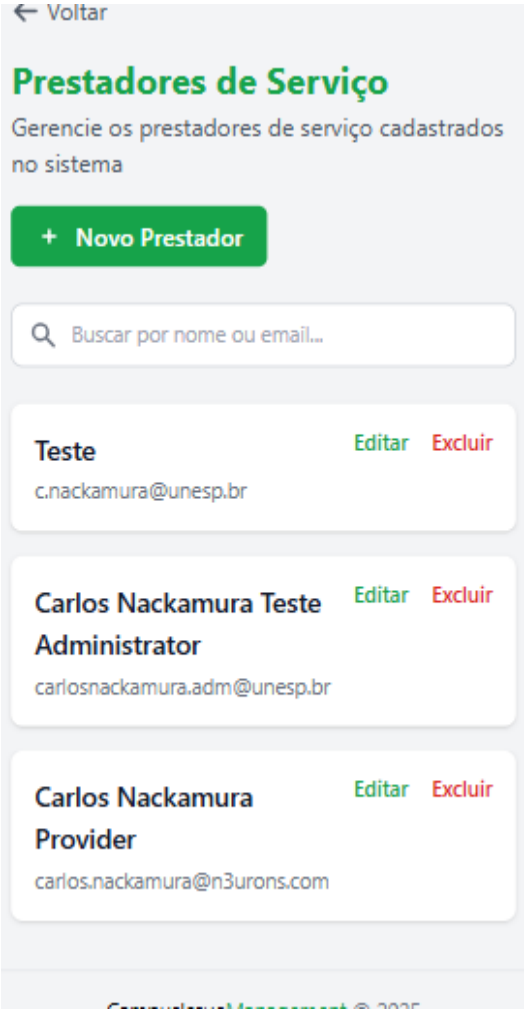


Fonte: autoria própria

7.8 Prestadores

Esta é a primeira tela exibida no fluxo de navegação do administrador do sistema (FIGURA 23). Após o login, o administrador pode acessar o menu "Prestadores" para visualizar todos os prestadores cadastrados no sistema. Esses prestadores têm permissão para atribuir chamados a si próprios, modificar o status dos atendimentos e interagir com o usuário responsável pela abertura do chamado.

Figura 23: Prestadores de serviços



Fonte: autoria própria

Ainda dentro da seção destinada à gestão de prestadores, o sistema oferece a funcionalidade de gerenciamento completo desses usuários por meio do botão "Novo

Prestador". Ao acionar esse recurso, o administrador é direcionado a uma interface que permite realizar operações fundamentais, como o cadastro de novos prestadores, a edição de dados de prestadores já existentes e, quando necessário, a exclusão definitiva de prestadores que não farão mais parte do sistema. Essa funcionalidade é essencial para garantir que apenas usuários autorizados e devidamente registrados tenham acesso às ferramentas e funcionalidades específicas destinadas aos prestadores de serviço, conforme ilustrado na Figura 24.

Figura 24: Cadastro de Prestadores

Fonte: autoria própria

7.9 Categorias

No escopo das funcionalidades disponibilizadas ao perfil de administrador do sistema, encontra-se a possibilidade de cadastrar e gerenciar as categorias disponíveis para a classificação dos chamados. Essa funcionalidade tem como principal objetivo organizar e padronizar as solicitações realizadas pelos usuários, oferecendo um conjunto pré-definido de

opções que refletem com precisão o escopo de atuação da equipe responsável pela prestação de serviços técnicos (FIGURA 25).

Ao restringir as categorias a termos previamente estabelecidos, evita-se o registro de chamados fora da alçada de atendimento do sistema, o que contribui para uma gestão mais eficiente, ágil e direcionada dos incidentes reportados. Essa configuração pode ser realizada diretamente pela interface administrativa, garantindo flexibilidade e controle sobre as demandas que chegam à equipe técnica.

Figura 25: Categorias de serviços

Fonte: autoria própria

Conforme ilustrado na figura, cada categoria deve conter, obrigatoriamente, um título representativo e uma descrição explicativa, a fim de orientar o usuário no momento da abertura do chamado (FIGURA 26).

Essa descrição tem o papel de esclarecer o escopo da categoria, incluindo, sempre que possível, exemplos de solicitações comuns que se enquadram naquele tipo de atendimento. Por exemplo, uma categoria intitulada Infraestrutura Predial pode conter descrições como: “Destinada a problemas relacionados à estrutura física do campus, como lâmpadas queimadas, vazamentos, problemas com portas, janelas ou forros.”

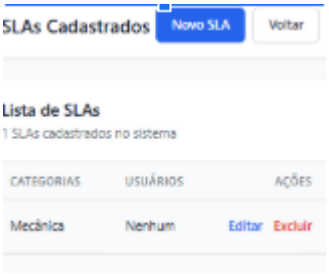
Figura 26: Cadastro de Categorias

Fonte: autoria própria

7.10 SLA

Ainda no âmbito das funcionalidades administrativas do sistema, está disponível a opção de configurar e gerenciar os Acordos de Nível de Serviço (SLAs), os quais podem ser definidos tanto com base nas categorias dos chamados quanto de forma individualizada por usuário (FIGURA 27). Essa funcionalidade permite estabelecer critérios mais precisos para o tratamento e a resolução das solicitações, assegurando que os prazos de atendimento estejam alinhados às diferentes naturezas das demandas ou à criticidade associada a determinados perfis de usuários. Os tempos estipulados para cada nível de prioridade — Baixo, Médio, Alto e Crítico — devem ser previamente definidos por meio de acordos internos da equipe gestora. Esses níveis representam apenas uma forma de classificação que auxilia na priorização das solicitações, considerando sua urgência e impacto, seja no contexto da categoria atribuída ao chamado, seja em relação à relevância do solicitante no processo. Dessa forma, o controle de SLAs contribui diretamente para uma gestão mais estratégica e eficaz dos atendimentos realizados pela equipe técnica.

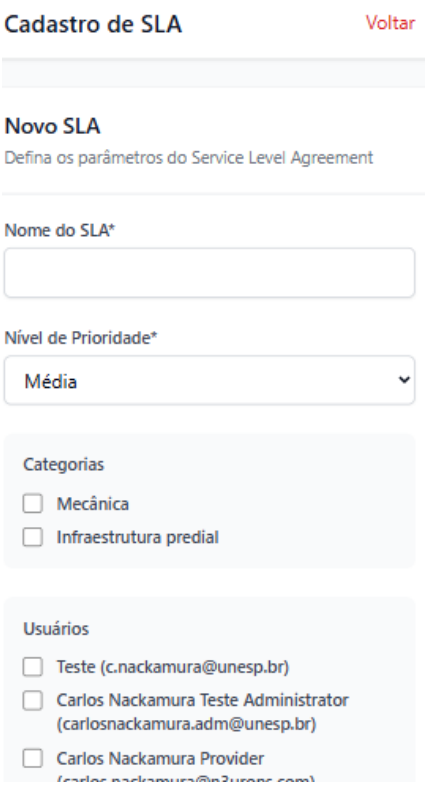
Figura 27: SLAs



Fonte: autoria própria

Nessa interface, é possível atribuir um nome ao SLA, selecionar o nível de prioridade (Baixo, Médio, Alto ou Crítico) e associá-lo a uma ou mais categorias de incidentes, como "Mecânica" e "Infraestrutura predial". Além disso, é possível vincular o SLA a usuários específicos (FIGURA 28), garantindo que chamados criados por esses usuários recebam o tratamento adequado conforme o acordo definido.

Figura 28: Cadastro de SLAs



Fonte: autoria própria

8 CONCLUSÃO

Dessa forma, conclui-se que o sistema de gestão de tickets desenvolvido para o ambiente universitário se apresenta como uma solução eficaz e estratégica para a organização, registro e acompanhamento de demandas relacionadas à manutenção no campus.

O sistema foi apresentado à Diretoria de Serviços, que identificou seu potencial para atender às necessidades reais da instituição, especialmente diante da crescente demanda por organização, rastreabilidade e eficiência na resolução de ocorrências cotidianas.

A plataforma centraliza a comunicação entre usuários — como alunos, funcionários e administradores — e as equipes responsáveis pelo atendimento, proporcionando maior agilidade nas respostas, redução de falhas de comunicação e transparência em todas as etapas do processo. A possibilidade de registrar chamados com informações detalhadas, incluindo localização geográfica, imagens e descrições, permite um diagnóstico mais preciso e um direcionamento mais assertivo das ações corretivas. Com mecanismos de atribuição, controle de acesso por papéis, priorização por critérios institucionais e histórico completo das interações, o sistema se adapta às rotinas operacionais do campus, garantindo que nenhum chamado seja perdido ou esquecido.

Desenvolvido como uma aplicação web responsiva e acessível, o sistema atende usuários com diferentes níveis de familiaridade com tecnologia e pode ser facilmente utilizado em diversos dispositivos. Sua arquitetura modular permite futuras expansões e integrações com outros sistemas institucionais. Assim, a solução vai além do simples registro de ocorrências: ela se consolida como uma ferramenta de gestão moderna, capaz de melhorar significativamente a eficiência dos serviços anteriormente prestados e contribuir com a qualidade do ambiente acadêmico.

Os resultados obtidos dos testes em que o sistema foi submetido indicam elevados índices de cobertura de código, com destaque para os módulos *auth, user, category* e *sla*, que atingiram 100% nas principais métricas: declarações executadas, ramificações lógicas, funções e linhas de código. Tais resultados reforçam a estabilidade da aplicação e a conformidade com boas práticas de desenvolvimento, especialmente no que se refere à garantia da qualidade e à mitigação de falhas em tempo de execução.

8.1 Trabalhos futuros

Como desdobramento deste projeto, identificam-se diversas possibilidades de continuidade que podem contribuir para a evolução e consolidação do sistema. A avaliação do

uso em ambiente real, por meio da aplicação de testes de usabilidade e coleta de feedback dos usuários finais, permitiria ajustes direcionados à interface e às funcionalidades da aplicação. A implantação de um painel de métricas administrativas possibilitaria o acompanhamento de estatísticas relevantes, como tempo médio de resolução, setores com maior volume de chamados e áreas mais impactadas pelas ocorrências.

A integração com sistemas acadêmicos existentes, viabiliza a automação da atribuição de responsáveis e o aumento da rastreabilidade das ações realizadas. A aplicação de algoritmos de priorização baseados em técnicas de aprendizado de máquina (machine learning) permitiria a ordenação automática dos chamados conforme critérios como impacto, urgência e histórico de ocorrências. A criação de um aplicativo mobile nativo também se apresenta como uma alternativa para facilitar o registro de incidentes diretamente em campo pelos usuários.

Além disso, a expansão do sistema para abranger outros tipos de chamados, como demandas relacionadas à tecnologia da informação, segurança, patrimônio ou acessibilidade, possibilitaria a consolidação da solução como uma central unificada de solicitações institucionais. Tais perspectivas evidenciam que o sistema desenvolvido constitui uma base sólida, modular e escalável, apta a acompanhar as transformações organizacionais e tecnológicas futuras.

9 REFERÊNCIAS

SANTOS, M. C. *Desenvolvimento e gestão de sistemas de chamados para manutenção predial*. 2020. Tese (Doutorado) – Faculdade de Tecnologia de Itapetininga, Itapetininga, 2020. Disponível em: https://sif.fatecitapetininga.edu.br/perspectiva/pdf/14/e14artigo%20(10).pdf. Acesso em: 2 set. 2024.

OLIVEIRA, R. T. *Sistema de gestão de tickets: estudo de caso*. 2023. Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) – Faculdade de Tecnologia de Ribeirão Preto, Ribeirão Preto, 2023. Disponível em: http://www.fatecrp.edu.br/WorkTec/edicoes/2023-2/trabalhos/ADS/artigo2.pdf. Acesso em: 2 set. 2024.

LEITE, André Victor do Nascimento et al. *Desenvolvimento de sistema para gerenciamento de chamados e notificações na FATEC Carapicuíba*. Perspectiva: Revista Interdisciplinar da Faculdade de Tecnologia de Tatuí, v. 14, n. 1, 2024. Disponível em: https://sif.fatecitapetininga.edu.br/perspectiva/pdf/14/e14artigo%20(10).pdf. Acesso em: 2 set. 2024.

MORETTI, Diego Carmino; NUNES, Karlos Eduardo Cristóvão Silva; PLOTZE, Rodrigo Oliveira; DINIZ, Debora Pelicano. Desenvolvimento de um aplicativo de gerenciamento de chamado usando framework Flutter. Ribeirão Preto: Faculdade de Tecnologia de FATEC Ribeirão Preto, 2023. Disponível em: http://www.fatecrp.edu.br/WorkTec/edicoes/2023-2/trabalhos/ADS/artigo2.pdf. Acesso em: 2 set. 2024.

PERONDI, L. T. *Sistema de gestão de chamados em ambientes universitários*. 2016. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Universidade de Caxias do Sul, Caxias do Sul, 2016. Disponível em: https://repositorio.ucs.br/xmlui/bitstream/handle/11338/1246/TCC%20Leandro%20Teixeira%20Perondi.pdf?sequence=1&isAllowed=y. Acesso em: 2 set. 2024

NOOR, F. M. et al. Infrastructure Damage Reporting System with Location Awareness. *Applied Mechanics and Materials*, v. 284–287, p. 2393–2396, 2013. Disponível em: https://www.scientific.net/AMM.284-287.2393. Acesso em: 11 dez.. 2024.

SILVA, F. A.; ALMEIDA, R. Implantação de sistemas de gestão de chamados. *Portal de Periódicos da UFSC*, 2020. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/84554883/1d659cca99a904c43a420c6f5ee049423412-libre.pdf. Acesso em: 11 dez.. 2024.

MARSHALL, Jim. *The case for centralizing maintenance activities*. *Plant Services*, 2016. Disponível em: https://www.facilitiesnet.com/facilitiesmanagement/article/The-Case-for-Centralizing-Maintenance-Activities--19880. Acesso em: 11 dez.. 2024.

NOOR, F. M. et al**.** Infrastructure Damage Reporting System with Location Awareness. *Applied Mechanics and Materials*, v. 284–287, p. 2393–2396, 2013. Disponível em: https://www.academia.edu/22315857/Infrastructure_Damage_Reporting_System_with_Location_Awareness. Acesso em: 11 dez. 2024.

ALBUQUERQUE, J. C. M. Sistemas de informação e comunicação no setor público. Florianópolis: Departamento de Ciências da Administração, Universidade Federal de Santa Catarina; Brasília: CAPES/UAB, 2015. Disponível em: https://educapes.capes.gov.br/bitstream/capes/719540/2/Sistemas%20de%20Informa%C3%A7%C3%A3o%20e%20comunica%C3%A7%C3%A3o%20no%20setor%20publico.pdf. Acesso em: 11 dez. 2024.

MORAES, Viviane Junqueira de. Gestão da manutenção em prédios públicos: estudo de caso na Universidade Federal de Ouro Preto. 2022. 88 f. Monografia (Graduação em Engenharia Civil) – Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2022. Disponível em: https://monografias.ufop.br/bitstream/35400000/5234/6/MONOGRAFIA_Gest%C3%A3oMantuten%C3%A7%C3%A3oPredios.pdf. Acesso em: 11 dez. 2024.

LARMAN, Craig. *Utilizando UML e Padrões: uma introdução à análise e ao projeto orientados a objetos e ao processo unificado*. 3. ed. Porto Alegre: Bookman, 2007. Disponível em: https://www.amazon.com.br/Utilizando-UML-Padr%C3%B5es-Craig-Larman/dp/8577800236. Acesso em: 11 dez. 2024.

SOMMERVILLE, Ian. *Engenharia de software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011. Disponível em: https://www.amazon.com.br/Engenharia-Software-Ian-Sommerville/dp/8576056865. Acesso em: 11 dez. 2024.

BROWN, Simon. *Software Architecture for Developers: Volume 2 - Visualise, document and explore your software architecture*. 2. ed. Bracknell: Leanpub, 2016. Disponível em: https://dokumen.pub/visualise-document-and-explore-your-software-architecture-software-architecture-for-developers-volume-2-2.html. Acesso em: 11 dez. 2024.

DOCKER INC. *Get started with Docker*. Disponível em: https://docs.docker.com/get-started/overview/. Acesso em: 11 dez. 2024.

KAMIL, Myśliwiec. *NestJS - A progressive Node.js framework*. Disponível em: https://docs.nestjs.com. Acesso em: 11 dez. 2024.

META. *React – A JavaScript library for building user interfaces*. Disponível em: https://react.dev. Acesso em: 11 dez. 2024.

OPENJS FOUNDATION. *About Node.js*. Disponível em: https://nodejs.org/en/about/. Acesso em: 11 dez. 2024.

ORACLE. *Why MySQL?*. Disponível em: https://www.mysql.com/why-mysql/. Acesso em: 11 dez. 2024.

BECK, Kent. *Test Driven Development: By Example*. Boston: Addison-Wesley, 2003. Disponível em: https://www.amazon.com/Test-Driven-Development-Kent-Beck/dp/0321146530. Acesso em: 11 dez. 2024.

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. *The Art of Software Testing*. 3. ed. Hoboken: Wiley, 2011. Disponível em: https://www.wiley.com/en-us/The+Art+of+Software+Testing%2C+3rd+Edition-p-9781118031964. Acesso em: 11 dez. 2024.

DODDS, Kent C. *Testing JavaScript Applications with Jest*. 2020. Disponível em: https://jestjs.io/. Acesso em: 11 dez. 2024.

DATE, C. J. Introdução a sistemas de banco de dados. 8. ed. Rio de Janeiro: Campus, 2004. Disponível em: https://www.kufunda.net/publicdocs/Introdução%20a%20Sistemas%20de%20Banco%20de%20Dados%20(C.%20J.%20Date).pdf. Acesso em: 11 dez. 2024.

CHEN, Peter. The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems, v. 1, n. 1, p. 9–36, 1976. Disponível em: https://doi.org/10.1145/320434.320440. Acesso em: 11 dez. 2024.