

INSTITUTO FEDERAL DO TOCANTINS - CAMPUS COLINAS DO TOCANTINS
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
JOÃO PAULO DA SILVA MOREIRA

ZETRIDEK: DESENVOLVIMENTO DE UM SISTEMA WEB PARA
GERENCIAMENTO DE CHAMADOS DE SUPORTE TÉCNICO

COLINAS DO TOCANTINS - TO
2026

JOÃO PAULO DA SILVA MOREIRA

**ZETRIDEK: DESENVOLVIMENTO DE UM SISTEMA WEB PARA
GERENCIAMENTO DE CHAMADOS DE SUPORTE TÉCNICO**

Trabalho de curso apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal do Tocantins – Campus Colinas do Tocantins, como requisito parcial para a obtenção do título de Tecnólogo em Tecnologia em Análise e Desenvolvimento de Sistemas, sob orientação do Prof. Prof. Dr. Ronneesley Moura Teles.

COLINAS DO TOCANTINS - TO

2026

Dados Internacionais de Catalogação na Publicação (CIP)
Bibliotecas do Instituto Federal do Tocantins

S586z Silva Moreira, João Paulo da
ZETRIDEK : DESENVOLVIMENTO DE UM SISTEMA WEB
PARA GERENCIAMENTO DE CHAMADOS DE SUPORTE
TÉCNICO / João Paulo da Silva Moreira. – Colinas do Tocantins, TO,
2026.

49 p. : il. color.

Trabalho de Conclusão de Curso (Curso Superior de Tecnologia
em Análise e Desenvolvimento de Sistemas) – Instituto Federal de
Educação, Ciência e Tecnologia do Tocantins, Campus Colinas do
Tocantins, Colinas do Tocantins, TO, 2026.

Orientador: Dr. Ronneesley Moura Teles

1. Sistema web. 2. Gerenciamento de chamados. 3. Django. I.
Teles, Ronneesley Moura. II. Título.

CDD 004.2

A reprodução total ou parcial, de qualquer forma ou por qualquer meio, deste documento é autorizada para fins
de estudo e pesquisa, desde que citada a fonte.

**Elaborado pelo sistema de geração automática de ficha catalográfica do IFTO com os dados fornecidos
pelo(a) autor(a).**

JOÃO PAULO DA SILVA MOREIRA

**ZETRIDEK: DESENVOLVIMENTO DE UM SISTEMA WEB PARA
GERENCIAMENTO DE CHAMADOS DE SUPORTE TÉCNICO**

Trabalho de curso apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal do Tocantins – Campus Colinas do Tocantins, como requisito parcial para a obtenção do título de Tecnólogo em Tecnologia em Análise e Desenvolvimento de Sistemas, sob orientação do Prof. Prof. Dr. Ronneesley Moura Teles.

Banca Examinadora:



Documento assinado digitalmente

RONNEESLEY MOURA TELES

Data: 23/06/2026 21:03:02-0300

Verifique em <https://validar.iti.gov.br>

Prof. Dr. Ronneesley Moura Teles (Orientador)
Instituto Federal do Tocantins Campus Colinas do Tocantins

Prof. Me. Adriano Honorato Braga
Instituto Federal Goiano Campus Ceres



Documento assinado digitalmente

ADRIANO HONORATO BRAGA

Data: 23/06/2026 22:28:51-0300

Verifique em <https://validar.iti.gov.br>

Esp. Luciano Ribeiro da Silva (Analista de TI)
Instituto Federal do Tocantins Campus Colinas do Tocantins



Documento assinado digitalmente

LUCIANO RIBEIRO DA SILVA

Data: 24/06/2026 15:36:49-0300

Verifique em <https://validar.iti.gov.br>

Aprovada em 08 de junho de 2026

Dedico este trabalho à minha família, pelo apoio incondicional e incentivo constante, que foram pilares essenciais durante toda a jornada acadêmica.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela força e sabedoria concedidas. Ao meu orientador, Prof. Dr. Ronneesley Moura Teles, pela valiosa orientação, paciência e incentivo, que foram fundamentais para a realização deste trabalho. Aos professores do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, pelos ensinamentos e pela formação acadêmica. Aos colegas de curso, pela parceria e troca de conhecimentos. E a todos que, direta ou indiretamente, contribuíram para a conclusão desta jornada.

“A tecnologia não é nada. O que é importante é que você tenha fé nas pessoas, que elas sejam basicamente boas e inteligentes, e se você lhes der ferramentas, elas farão coisas maravilhosas com elas.”

(Steve Jobs)

Resumo

Este trabalho apresenta o desenvolvimento do Zetridesk, um sistema *web* para gerenciamento de chamados de suporte técnico, projetado para otimizar a organização, o controle e a eficiência dos processos de atendimento. A pesquisa caracteriza-se como aplicada de desenvolvimento tecnológico, fundamentando-se na concepção, modelagem e implementação de uma solução prática voltada para sanar lacunas de comunicação e controle comuns em processos de suporte técnico. Adotou-se a metodologia ágil Scrum, estruturada em Sprints de duas semanas, abrangendo a definição de requisitos, modelagem de dados, definição arquitetural e desenvolvimento incremental. A arquitetura adotada foi baseada no padrão Model-Template-View (MTV) do framework Django, autenticação baseada em sessão e o Sistema de Gerenciamento de Banco de Dados (SGBD) relacional MySQL. O sistema desenvolvido estabelece a padronização do fluxo de atendimento, rastreabilidade das demandas para o usuário solicitante e indicadores estratégicos para a gestão. A demonstração de viabilidade funcional do sistema em ambiente de desenvolvimento local (*localhost*) evidenciou que a solução atende de forma consistente aos requisitos de projeto estabelecidos, apresentando-se como uma ferramenta centralizada e viável para a governança de incidentes e requisições de TI.

Palavras-chave: Sistema *Web*. Gerenciamento de Chamados. Engenharia de Software. Django. MySQL.

Abstract

This work presents the development of Zetridesk, a *web* system for managing technical support calls designed to improve the organization, control, and efficiency of service processes. The research is characterized as applied technological development, grounded in the design, modeling, and implementation of a practical solution aimed at addressing common communication and control gaps in technical support processes. The Agile Scrum methodology was adopted, structured in two-week Sprints, covering requirements definition, data modeling, architectural definition, and incremental development. The adopted architecture was based on the Model-Template-View (MTV) pattern of the Django framework, session-based authentication, and the MySQL relational Database Management System (DBMS). The developed system establishes service flow standardization, traceability of demands for the requesting user, and strategic indicators for management. The results obtained through functional testing conducted in a development environment local (*localhost*) demonstrate that the system meets the established requirements, offering a centralized and viable tool for IT incident and request governance.

Keywords: *Web* System. Call Management. Software Engineering. Django. MySQL.

LISTA DE ILUSTRAÇÕES

Figura 1 – Diagrama de Casos de Uso do Zetridesk	30
Figura 2 – Diagrama de Sequência — Fluxo de Abertura de Chamado	32
Figura 3 – Diagrama de Classes do Zetridesk	34
Figura 4 – Diagrama de Entidade-Relacionamento do Zetridesk	36
Figura 5 – Interface de Autenticação do Zetridesk	42
Figura 6 – Dashboard do Usuário Solicitante	42
Figura 7 – Formulário de Criação de Novo Chamado	43
Figura 8 – Dashboard Operacional do Técnico de Suporte	44
Figura 9 – Dashboard Gerencial do Administrador	44
Figura 10 – Painel de Administração Integrada do Django	45

LISTA DE TABELAS

Tabela 1 – Cronograma efetivo de Sprints executado no projeto	24
Tabela 2 – Requisitos funcionais do sistema Zetridesk	28
Tabela 3 – Requisitos não funcionais do sistema Zetridesk	29
Tabela 4 – Estrutura física da tabela usuario	37
Tabela 5 – Estrutura física da tabela chamado	37
Tabela 6 – Estrutura física da tabela interacao	37
Tabela 7 – Estrutura física da tabela avaliacao	38

LISTA DE ABREVIATURAS E SIGLAS

CSRF	<i>Cross-Site Request Forgery</i> (Falsificação de Solicitação Entre Sites)
DER	Diagrama de Entidade-Relacionamento
GLPI	<i>Gestionnaire Libre de Parc Informatique</i> (Gerenciador Livre de Parque de Informática)
HTML	<i>HyperText Markup Language</i> (Linguagem de Marcação de Hipertexto)
HTTPS	<i>HyperText Transfer Protocol Secure</i> (Protocolo de Transferência de Hipertexto Seguro)
IHC	Interação Humano-Computador
ITIL	<i>Information Technology Infrastructure Library</i> (Biblioteca de Infraestrutura de Tecnologia da Informação)
KPI	<i>Key Performance Indicator</i> (Indicador-Chave de Desempenho)
MER	Modelo de Entidade-Relacionamento
MVC	<i>Model-View-Controller</i> (Modelo-Visão-Controle)
MTV	<i>Model-Template-View</i> (Modelo-Template-Visão)
ORM	<i>Object-Relational Mapping</i> (Mapeamento Objeto-Relacional)
RAM	<i>Random Access Memory</i> (Memória de Acesso Aleatório)
RBAC	<i>Role-Based Access Control</i> (Controle de Acesso Baseado em Papéis)
SGBD	Sistema de Gerenciamento de Banco de Dados
SLA	<i>Service Level Agreement</i> (Acordo de Nível de Serviço)
TI	Tecnologia da Informação
UML	<i>Unified Modeling Language</i> (Linguagem de Modelagem Unificada)
XSS	<i>Cross-Site Scripting</i> (Scripting Entre Sites)

SUMÁRIO

1	Introdução	14
1.1	Justificativa	15
1.2	Objetivo geral	16
1.3	Objetivos específicos	16
1.4	Organização do trabalho	16
2	Referencial Teórico	18
2.1	Engenharia de software e processo	18
2.2	Levantamento e análise de requisitos do sistema	18
2.2.1	Requisitos funcionais	19
2.2.2	Requisitos não funcionais	19
2.3	Usabilidade e interação humano-computador	19
2.4	Modelagem de software e UML	20
2.5	Modelagem de banco de dados relacional	20
2.6	Arquitetura <i>model-view-controller</i> (MVC)	21
3	Materiais e Métodos	23
3.1	Materiais	23
3.1.1	Tecnologias de software	23
3.1.2	Infraestrutura de hardware	23
3.2	Métodos	24
3.2.1	Processo de desenvolvimento (<i>Scrum</i>)	24
3.2.2	Levantamento de requisitos e análise de similares	25
3.2.3	Notações de modelagem e validação	25
4	Projeto	27
4.1	Arquitetura do projeto	27
4.2	Requisitos do sistema	27
4.2.1	Requisitos funcionais	27
4.2.2	Requisitos não funcionais	28
4.3	Protótipos de interface	29
4.4	Diagrama de casos de uso	29
4.5	Diagrama de sequência	31
4.6	Diagrama de classes	32

4.7	Diagrama entidade-relacionamento (DER)	34
4.7.1	Dicionário de dados e tabelas do banco de dados	36
4.8	Algoritmos propostos	38
4.8.1	Geração de identificador único de chamado	38
4.8.2	Cálculo e verificação de SLA	39
4.8.3	Controle de acesso baseado em papéis (RBAC)	39
5	Resultados e Discussão	41
5.1	Resultados do desenvolvimento	41
5.2	Apresentação das telas e funcionalidades desenvolvidas	41
5.2.1	Interface de autenticação	41
5.2.2	Painel do usuário solicitante	42
5.2.3	Interface de abertura de chamados	43
5.2.4	Painel de atendimento do técnico	43
5.2.5	Painel gerencial e indicadores do administrador	44
5.2.6	Módulo administrativo do <i>Django</i>	45
5.3	Discussão sobre a demonstração e viabilidade do fluxo proposto	45
6	Considerações Finais	47
	Referências	48

1 INTRODUÇÃO

As transformações digitais têm impactado significativamente a forma como as organizações gerenciam seus processos internos e se relacionam com seus usuários. Entre esses processos, o suporte técnico ocupa papel estratégico, pois influencia diretamente a satisfação do usuário, a continuidade das operações e a imagem institucional. Nesse contexto, sistemas informatizados tornaram-se ferramentas essenciais para organizar, registrar e acompanhar solicitações de atendimento. De acordo com Sommerville (2011), a eficiência dos serviços de tecnologia da informação depende diretamente da existência de processos bem definidos e de canais formais de comunicação para o registro de demandas.

O gerenciamento manual ou descentralizado de chamados de suporte técnico, realizado por meio de e-mails, mensagens instantâneas ou anotações informais, tende a gerar retrabalho, perda de informações e dificuldades no acompanhamento das demandas. Embora essa realidade seja observada em diversas organizações, em ambientes educacionais e corporativos de médio porte, a ausência de um sistema centralizado de atendimento acarreta gargalos operacionais que afetam diretamente o fluxo de trabalho de colaboradores, técnicos e usuários finais, que dependem de infraestrutura tecnológica estável para suas atividades cotidianas. De acordo com Pressman e Maxim (2021), a ausência de controle estruturado compromete a análise de indicadores de desempenho operacionais, inviabilizando melhorias preventivas na infraestrutura tecnológica.

Com o avanço das tecnologias *web*, tornou-se viável a proposição de sistemas acessíveis via navegador, sem necessidade de instalação local, permitindo maior flexibilidade, escalabilidade e integração com outros sistemas organizacionais. Segundo Pressman e Maxim (2021), a adoção de plataformas baseadas na *web* potencializa a comunicação interna e a eficiência na gestão de dados operacionais. Um sistema *web* de gerenciamento de chamados possibilita o registro formal das solicitações, classificação por prioridade, atribuição de responsáveis, acompanhamento do status e geração de relatórios gerenciais. De acordo com Costa (2018), a implementação de ferramentas digitais estruturadas é fundamental para garantir a rastreabilidade das informações e o controle rigoroso dos fluxos de trabalho.

Além de otimizar processos internos, um sistema dessa natureza contribui para a transparência na comunicação entre usuários e equipe técnica. O usuário pode acompanhar o andamento de sua solicitação em tempo real, enquanto a equipe de suporte organiza melhor sua carga de trabalho, estabelecendo prazos e prioridades de forma estruturada. Outro aspecto

relevante é a possibilidade de utilização de bancos de dados para armazenamento histórico das ocorrências. Isso permite identificar padrões recorrentes, antecipar falhas, propor melhorias preventivas e apoiar a tomada de decisão baseada em dados. Assim, sistemas dessa natureza deixam de ser apenas operacionais e passam a desempenhar função estratégica dentro da organização.

Diante desse cenário, o desenvolvimento de um sistema *web* para gerenciamento de chamados de suporte técnico apresenta-se como solução tecnológica capaz de aumentar a eficiência operacional, melhorar a experiência do usuário e fortalecer a governança dos processos de atendimento em diferentes contextos organizacionais.

1.1 Justificativa

A justificativa para o desenvolvimento do Zetridesk fundamenta-se na crescente dependência tecnológica das organizações contemporâneas, onde a eficiência do suporte técnico torna-se um pilar estratégico para a continuidade das operações. A ausência de uma ferramenta centralizada resulta em fluxos de trabalho fragmentados, perda de informações críticas e dificuldade na mensuração do desempenho da equipe de Tecnologia da Informação (TI) devido ao uso de canais informais, sendo essa necessidade de um sistema *web* simplificado comum a diversas organizações e instituições de médio porte.

Embora existam no mercado soluções consolidadas para o gerenciamento de serviços de TI — como o Zendesk e o Gerenciador Livre de Parque de Informática (GLPI) —, a adoção direta dessas ferramentas apresenta limitações práticas significativas para organizações com recursos limitados. Plataformas comerciais como o Zendesk demandam custos elevados de licenciamento por agente técnico, inviabilizando sua aplicação em cenários com restrições orçamentárias. O GLPI, por sua vez, embora seja uma excelente ferramenta de código aberto, apresenta uma curva de aprendizado íngreme e uma interface complexa para o usuário final sem treinamento específico, além de exigir esforço contínuo de manutenção e infraestrutura local dedicada. O desenvolvimento do Zetridesk preenche essa lacuna, oferecendo uma solução gratuita, de fácil manutenção, com interface intuitiva e focada na agilidade operacional de centrais de suporte técnico.

O desenvolvimento deste sistema visa, portanto, oferecer uma solução que não apenas registra incidentes, mas gerencia o ciclo de vida completo do atendimento, otimizando o fluxo local e garantindo autonomia tecnológica para a instituição.

Sob a ótica da gestão de serviços de TI, a implementação de uma central de serviços estruturada permite a aplicação de melhores práticas, como as preconizadas pela *Information Technology Infrastructure Library* (ITIL). A automação do cálculo do *Service Level Agreement* (SLA) e a priorização inteligente de chamados garantem que os recursos técnicos sejam alocados de forma otimizada, atendendo primeiro às demandas de maior impacto institucional. Além disso, a transparência proporcionada pelo acompanhamento em tempo real eleva o nível de confiança dos usuários finais no setor de tecnologia.

1.2 Objetivo geral

Propor e desenvolver uma plataforma *web* funcional e segura para centralizar e gerenciar todo o ciclo de vida dos chamados de suporte técnico de TI.

1.3 Objetivos específicos

Para a consecução do objetivo geral, foram definidos os seguintes objetivos específicos:

- Definir os requisitos funcionais e não funcionais do sistema com base na análise de soluções similares e na identificação de fluxos de suporte de TI;
- Modelar a estrutura do sistema, incluindo banco de dados relacional, arquitetura de software e fluxos de atendimento por perfil de usuário;
- Desenvolver as funcionalidades principais, incluindo cadastro de usuários, abertura de chamados, atribuição de técnicos e alteração de status;
- Implementar mecanismos de controle de acesso baseado em papéis (RBAC) e segurança da informação para garantir a integridade dos dados;
- Executar testes funcionais e de usabilidade realizados pelo próprio desenvolvedor para validar a conformidade das regras de negócio e estabilidade da aplicação;

1.4 Organização do trabalho

Este trabalho está estruturado em seis capítulos. Este capítulo inicial expõe a introdução, a justificativa, os objetivos e a organização do estudo. O Capítulo 2 apresenta o referencial teórico que embasa o desenvolvimento do sistema, abordando conceitos de engenharia de

requisitos, usabilidade e Interação Humano-Computador (IHC), modelagem de banco de dados e arquitetura *Model-View-Controller* (MVC). O Capítulo 3 detalha os materiais e métodos adotados, descrevendo as tecnologias utilizadas e o processo de desenvolvimento baseado na metodologia ágil Scrum. O Capítulo 4 apresenta o projeto de software desenvolvido, incluindo diagramas de casos de uso, classes, entidade-relacionamento e protótipos de telas. O Capítulo 5 descreve os resultados obtidos por meio do sistema desenvolvido e discute os testes de validação funcional realizados. Por fim, o Capítulo 6 apresenta as considerações finais e as propostas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a fundamentação teórica que embasa o desenvolvimento do sistema *web* para gerenciamento de chamados de suporte técnico. Serão abordados os conceitos fundamentais de engenharia de software, levantamento de requisitos, usabilidade, modelagem de banco de dados e padrões arquiteturais adotados no projeto.

2.1 Engenharia de software e processo

A Engenharia de Software compreende um conjunto de conceitos, métodos, ferramentas e procedimentos que visam à produção de sistemas de software de alta qualidade de maneira sistemática, controlada e eficiente. Segundo Pressman e Maxim (2021), o desenvolvimento de um sistema deve ser guiado por um processo de software estruturado, que define as etapas, atividades, entregas e controles necessários para garantir a previsibilidade e a robustez do produto final. No mesmo sentido, Sommerville (2011) ressalta que a escolha de um modelo de processo adequado deve alinhar-se à complexidade do domínio do problema e às restrições do projeto.

No desenvolvimento de sistemas modernos, as metodologias ágeis ganharam destaque ao propor um processo iterativo e incremental, focado em entregas rápidas e na colaboração contínua com os envolvidos. De acordo com Soares (2004), o Scrum estabelece um arcabouço de trabalho baseado em papéis definidos, reuniões periódicas e ciclos de desenvolvimento curtos chamados de *Sprints*. Essa abordagem permite que o sistema seja refinado progressivamente com base no *feedback* dos usuários. Pesquisas de caso, como a de Carvalho e Mello (2012), demonstram que a aplicação do método ágil Scrum em pequenas equipes de desenvolvimento resulta em maior produtividade, flexibilidade diante de mudanças de requisitos e melhor alinhamento das expectativas do cliente com o produto entregue.

2.2 Levantamento e análise de requisitos do sistema

A identificação das necessidades dos usuários constitui etapa central no desenvolvimento de sistemas *web*, pois estabelece as bases para a definição adequada dos requisitos. Costa (2018) defende que a qualidade na coleta de requisitos depende de processos estruturados e métricas claras, evitando ambiguidades e retrabalhos que possam comprometer as etapas subsequentes do ciclo de desenvolvimento. Sob essa perspectiva, compreender as demandas

reais dos usuários não é apenas uma etapa preliminar, mas um fator determinante para o sucesso do projeto.

A engenharia de requisitos divide-se classicamente em duas categorias principais de especificações: requisitos funcionais e requisitos não funcionais. Segundo Sommerville (2011, p. 75), enquanto os requisitos funcionais descrevem as ações específicas e os comportamentos que o sistema deve realizar em resposta a entradas específicas, os requisitos não funcionais estabelecem os critérios de qualidade e as restrições sob as quais o sistema deve operar.

2.2.1 Requisitos funcionais

Os requisitos funcionais descrevem as facilidades e os serviços que o sistema deve fornecer. Em sistemas de gerenciamento de chamados de suporte técnico, a especificação funcional deve cobrir de forma clara o fluxo de vida do chamado. De acordo com Perondi (2013), um sistema de suporte típico deve contemplar funcionalidades essenciais como a autenticação de usuários com diferentes níveis de permissão, o registro detalhado de incidentes por parte do usuário final, a triagem e atribuição de técnicos pelo administrador, o registro de interações e a alteração de status do chamado até o seu encerramento definitivo. A clareza na modelagem dessas interações garante a rastreabilidade e o controle operacional do suporte.

2.2.2 Requisitos não funcionais

Os requisitos não funcionais estabelecem restrições sobre os serviços ou funções oferecidos pelo sistema, abrangendo aspectos como desempenho, confiabilidade, portabilidade, segurança e usabilidade. De acordo com Pressman e Maxim (2021), os requisitos não funcionais são fundamentais para garantir a aceitação do sistema no ambiente operacional real, pois determinam a experiência de uso sob condições normais e de estresse. No contexto de sistemas *web*, atributos como a segurança na transmissão de dados por meio do protocolo *Hypertext Transfer Protocol Secure* (HTTPS) e o controle de acesso baseado em papéis (RBAC) são cruciais para proteger informações institucionais contra acessos não autorizados.

2.3 Usabilidade e interação humano-computador

A facilidade de uso e a experiência do usuário são fatores determinantes para o sucesso e a adoção de qualquer sistema de informação. Segundo Barbosa e Silva (2010, p. 94), a Interação Humano-Computador (IHC) estuda como as pessoas interagem com os sistemas

computacionais e busca projetar interfaces que facilitem essa interação de maneira eficiente, segura e agradável. No *design* de sistemas de suporte, onde os usuários podem apresentar diferentes níveis de familiaridade com tecnologia, a interface deve ser intuitiva, minimizando a carga cognitiva e o tempo necessário para a realização de tarefas como a abertura de chamados. Perondi (2013, p. 43) reforça que interfaces complexas ou confusas geram registros incompletos ou incorretos, sobrecarregando a equipe de suporte com chamados mal formulados.

2.4 Modelagem de software e UML

A modelagem de software desempenha papel fundamental no desenvolvimento de sistemas complexos, permitindo que a equipe visualize, especifique, construa e documente os artefatos de um sistema sob diferentes perspectivas. A linguagem padrão utilizada para essa finalidade é a UML (*Unified Modeling Language*). Segundo Booch, Rumbaugh e Jacobson (2005), a UML fornece um conjunto rico de notações gráficas divididas em diagramas estruturais, que representam a organização estática do sistema, e diagramas comportamentais, que descrevem a dinâmica e as interações entre os elementos ao longo do tempo.

No contexto da análise orientada a objetos, a transição entre os requisitos de negócio e o projeto técnico é facilitada pelo uso de padrões de modelagem bem estabelecidos. Larman (2007) defende que o mapeamento sistemático de casos de uso para diagramas de classes e de sequência garante a robustez arquitetural e reduz o risco de falhas na implementação física do código. Para a modelagem comportamental detalhada, o diagrama de sequência destaca-se como uma ferramenta indispensável para representar a troca de mensagens cronológica entre os objetos do sistema. Conforme aponta Fowler (2005, p. 67), esse diagrama permite validar a distribuição de responsabilidades entre as classes e antecipar o comportamento dinâmico da aplicação sob diferentes cenários de execução.

2.5 Modelagem de banco de dados relacional

A modelagem de dados é o processo de definição da estrutura lógica e física das informações que serão armazenadas no sistema. Segundo Elmasri e Navathe (2011, p. 131), a modelagem conceitual baseada no Modelo Entidade-Relacionamento (MER) permite representar as entidades do mundo real, seus atributos e os relacionamentos existentes de forma abstrata e independente de implementação. Essa representação é posteriormente mapeada para o modelo

relacional lógico, que organiza os dados em tabelas compostas por linhas e colunas, garantindo a consistência por meio de chaves primárias e estrangeiras.

Para a persistência física dos dados, a escolha do Sistema de Gerenciamento de Banco de Dados (SGBD) deve considerar critérios como confiabilidade, desempenho, facilidade de integração e custo. O MySQL destaca-se como um SGBD relacional de código aberto amplamente utilizado na indústria. Conforme aponta Elmasri e Navathe (2011, p. 131), o MySQL destaca-se por sua robustez na garantia de transações de Atomicidade, Consistência, Isolamento e Durabilidade (ACID), excelente integração com linguagens de programação populares como Python e o *framework* Django, tornando-o uma solução viável para sistemas *web* corporativos de médio porte.

2.6 Arquitetura *model-view-controller* (MVC)

A arquitetura de software define a organização estrutural do sistema e as diretrizes para a interação entre seus componentes. O padrão arquitetural *Model-View-Controller* (MVC) é amplamente adotado no desenvolvimento *web* para promover a separação de conceitos e reduzir o acoplamento do código. Segundo Pressman e Maxim (2021, p. 349), o MVC divide a aplicação em três camadas distintas:

- **Model** (Modelo): Gerencia os dados, as regras de negócio e a persistência;
- **View** (Visão): Responsável pela renderização da interface gráfica e apresentação dos dados ao usuário;
- **Controller** (Controle): Intermedeia a comunicação entre a visão e o modelo, processando as requisições do usuário e coordenando as ações do sistema.

O *framework* Django implementa uma variação desse padrão conhecida como *Model-Template-View* (MTV), onde a camada *Template* corresponde à visão clássica (HTML/CSS) e a camada *View* atua como o controlador. Embora essa variação possa gerar dúvidas iniciais na transição conceitual do padrão MVC clássico — especialmente pelo fato de que o Django chama de "View" o que classicamente seria o "Controller", e de "Template" o que seria a "View—, a separação de responsabilidades permanece estritamente preservada.

De fato, a implementação do padrão MTV no Django simplifica o fluxo de desenvolvimento de sistemas *web* ao desacoplar a renderização visual da lógica de tratamento de

requisições. O desenvolvedor foca na definição de modelos de dados robustos (*Models*), no mapeamento de rotas e processamento de regras de negócio em funções ou classes controladoras (*Views*), e na exibição limpa dos dados em arquivos de marcação dinâmicos (*Templates*). De acordo com Pressman e Maxim (2021, p. 349), a adoção de *frameworks* baseados nesse padrão acelera o desenvolvimento ao fornecer recursos integrados como mapeamento objeto-relacional (ORM), sistema de rotas, controle de sessões e mecanismos de segurança contra vulnerabilidades comuns na *web*, garantindo a manutenibilidade e a escalabilidade do sistema desenvolvido.

3 MATERIAIS E MÉTODOS

Este capítulo detalha a metodologia adotada no desenvolvimento do sistema Zetridesk, descrevendo os materiais utilizados e os métodos aplicados para a condução do trabalho, desde a fase de levantamento de requisitos até a validação funcional.

3.1 Materiais

Os materiais empregados no projeto compreenderam ferramentas de *software* e a infraestrutura de *hardware* utilizada para o desenvolvimento e testes locais da aplicação.

3.1.1 Tecnologias de software

O desenvolvimento do sistema foi realizado utilizando as seguintes tecnologias:

- **Linguagem de Programação:** Python (versão 3.14.3), adotada pela sua legibilidade, robustez e ecossistema de bibliotecas voltadas ao desenvolvimento *web*;
- **Framework Web:** Django (versão 5.2.11), selecionado pela sua arquitetura baseada no padrão MTV (*Model-Template-View*), facilidade de mapeamento objeto-relacional (ORM) e recursos integrados de segurança e autenticação;
- **Sistema de Gerenciamento de Banco de Dados (SGBD):** MySQL (versão 8.0), utilizado para o armazenamento persistente e relacional dos dados do sistema;
- **Ambiente de Desenvolvimento Integrado (IDE):** *Visual Studio Code*, utilizado para a codificação, depuração e organização dos arquivos do projeto;
- **Interface e Estilização:** HTML5, CSS3 e Bootstrap 5, utilizados para a construção de uma interface *web* responsiva e amigável ao usuário.

3.1.2 Infraestrutura de hardware

O ambiente de desenvolvimento e testes locais consistiu em um computador pessoal com o seguinte perfil de *hardware*:

- **Processador:** Intel Core i5-1135G7 (4 núcleos, 8 *threads*, frequência base de 2.40 GHz);
- **Memória RAM:** 8 GB;

- **Armazenamento:** Disco de estado sólido de 512 GB;
- **Sistema Operacional:** Windows 11 Home de 64 bits.

3.2 Métodos

O presente trabalho caracteriza-se como uma pesquisa aplicada de desenvolvimento tecnológico, focada na concepção, modelagem e implementação de uma solução prática de software. O desenvolvimento do sistema foi conduzido de forma iterativa e incremental, utilizando as melhores práticas de engenharia de software.

3.2.1 Processo de desenvolvimento (*Scrum*)

O desenvolvimento do sistema foi estruturado com base na metodologia ágil Scrum, adaptada para um ciclo de desenvolvimento individual sob orientação. O projeto foi executado em um período total de 16 semanas, entre 05 de janeiro de 2026 e 26 de abril de 2026, dividido em oito ciclos (*Sprints*) de duas semanas cada.

O cronograma efetivo de atividades seguiu a distribuição apresentada na Tabela 1.

Tabela 1 – Cronograma efetivo de Sprints executado no projeto

Sprint	Atividades Realizadas	Duração
1	Levantamento de requisitos e especificação funcional	Semanas 1–2
2	Modelagem de banco de dados e prototipagem de telas	Semanas 3–4
3	Configuração do ambiente Django e criação do modelo de dados	Semanas 5–6
4	Desenvolvimento do módulo de autenticação e perfis (RBAC)	Semanas 7–8
5	Implementação das funcionalidades de abertura e atribuição de chamados	Semanas 9–10
6	Desenvolvimento do módulo de interações, histórico e alteração de status	Semanas 11–12
7	Implementação do módulo de relatórios e indicadores de desempenho	Semanas 13–14
8	Realização de testes funcionais, correção de bugs e escrita do relatório	Semanas 15–16

Ao final de cada ciclo de duas semanas, as entregas incrementais foram apresentadas ao orientador em reuniões periódicas de acompanhamento, que funcionaram como as cerimônias de *Sprint Review* e planejamento do ciclo seguinte.

3.2.2 Levantamento de requisitos e análise de similares

A definição de requisitos funcionais e não funcionais do sistema foi baseada na análise comparativa de sistemas consolidados de gerenciamento de chamados de suporte técnico (como o Zendesk e o GLPI) e na modelagem de um fluxo idealizado para suprir falhas comuns em processos de atendimento informais (conduzidos de forma descentralizada via e-mails e mensagens instantâneas).

A análise focou em identificar:

- Os fluxos básicos de abertura, triagem, atribuição de técnicos e encerramento de chamados;
- Os principais dados e atributos necessários para garantir a rastreabilidade de cada demanda;
- As limitações de usabilidade e complexidade de interface que dificultam o uso de sistemas comerciais por usuários finais sem treinamento técnico.

A partir dessa análise e da identificação dos problemas de perda de informações em fluxos informais de suporte, foram definidos os requisitos funcionais e não funcionais apresentados no Capítulo 4, visando criar uma solução simplificada, responsiva e sob medida para a comunidade acadêmica.

3.2.3 Notações de modelagem e validação

A modelagem do sistema foi realizada utilizando as seguintes notações da UML (*Unified Modeling Language*):

- **Diagrama de Casos de Uso:** Para representar as interações dos diferentes perfis de usuários com as funcionalidades do sistema;
- **Diagrama de Classes:** Para definir a estrutura lógica das classes e seus relacionamentos no framework Django;
- **Diagrama de Entidade-Relacionamento (DER):** Para mapear a estrutura física do banco de dados relacional MySQL.

A validação funcional do sistema foi conduzida por meio de testes de caixa-preta executados diretamente pelo próprio desenvolvedor em ambiente de desenvolvimento local

(*localhost*). Esses testes visaram verificar a integridade física das transações do banco de dados, o controle de acesso por perfil de usuário e o comportamento das telas e regras de negócio frente aos requisitos funcionais previamente definidos.

4 PROJETO

Este capítulo apresenta o projeto do sistema Zetridesk, detalhando a arquitetura de software adotada, os protótipos de interface, a modelagem lógica e física do banco de dados e os diagramas estruturais e comportamentais da aplicação.

4.1 Arquitetura do projeto

O sistema Zetridesk foi projetado e desenvolvido seguindo a arquitetura Model-View-Controller (MVC), implementada por meio do padrão Model-Template-View (MTV) nativo do framework Django. A organização em camadas garantiu a separação entre a interface com o usuário, as regras de negócio e a persistência de dados.

A estrutura do projeto Django foi organizada nos seguintes módulos principais:

- **Módulo de Autenticação (Core):** Responsável pelo controle de acesso, autenticação baseada em sessões seguras e criptografia de senhas usando algoritmo de hash seguro;
- **Módulo de Chamados (Tickets):** Responsável pelo fluxo completo de abertura, triagem, interação, histórico de status e encerramento de chamados;
- **Módulo de Administração (Admin):** Responsável pelo gerenciamento de usuários, categorias de chamados, prazos de SLA e geração de relatórios consolidados.

4.2 Requisitos do sistema

A partir da análise de sistemas similares e da observação direta do fluxo informal de atendimento, foram consolidados os requisitos funcionais (RF) e não funcionais (RNF) que orientaram o desenvolvimento da aplicação.

4.2.1 Requisitos funcionais

Os requisitos funcionais foram mapeados de forma exaustiva para garantir a rastreabilidade completa do projeto e atender a todas as demandas de controle operacional identificadas, conforme apresentado na Tabela 2.

Tabela 2 – Requisitos funcionais do sistema Zetridesk

ID	Atributo	Descrição do Requisito
RF-01	Autenticação	Permitir o login de usuários com base em credenciais de e-mail e senha criptografada.
RF-02	Perfis de Acesso	Controlar o acesso às telas com base em três perfis: Solicitante, Técnico (atendente) e Administrador (gestor).
RF-03	Abertura de Chamado	Permitir que o usuário autenticado abra um chamado informando título, descrição, categoria e prioridade.
RF-04	Atribuição de Chamado	Permitir que o Administrador atribua um chamado pendente a um Técnico específico para atendimento.
RF-05	Reclassificação	Permitir que o Técnico reclassifique a categoria de um chamado ou escalone para outro setor/técnico.
RF-06	Interações	Permitir a troca de mensagens de texto entre técnico e solicitante dentro do chamado.
RF-07	Histórico de Status	Registrar automaticamente cada mudança de status do chamado, associando data, hora e usuário responsável.
RF-08	Cancelamento	Permitir que o solicitante cancele um chamado de sua autoria caso a demanda não seja mais necessária.
RF-09	Pesquisa de Satisfação	Disponibilizar pesquisa de satisfação opcional após o encerramento do chamado (nota de 1 a 5 e comentário).
RF-10	Previsão de SLA	Calcular automaticamente a data limite de atendimento e exibir alertas visuais de proximidade de estouro do prazo.
RF-11	Notificação por E-mail	Enviar e-mails automáticos no momento de abertura, atribuição, novas interações e encerramento do chamado.
RF-12	Relatórios Gerenciais	Exibir gráficos consolidados com tempo médio de atendimento, volume de chamados e índice de satisfação.

Fonte: Elaborado pelo autor (2026).

4.2.2 Requisitos não funcionais

Os requisitos não funcionais definem as restrições e qualidades técnicas que o sistema Zetridesk deve apresentar para garantir robustez, portabilidade e segurança. Conforme destacado por Watanabe (2014), o projeto de interfaces ricas para a internet deve primar pela usabilidade e pela conformidade técnica de seus atributos não funcionais, garantindo uma experiência fluida e acessível ao usuário final. Nesse sentido, os requisitos não funcionais do sistema foram estruturados conforme apresentado na Tabela 3.

Tabela 3 – Requisitos não funcionais do sistema Zetridesk

ID	Atributo	Descrição do Requisito
RNF-01	Segurança	Criptografia de senhas utilizando algoritmo de hash seguro.
RNF-02	Usabilidade	Interface responsiva desenvolvida com Bootstrap 5, compatível com dispositivos móveis e desktops.
RNF-03	Portabilidade	Acesso por meio de navegadores <i>web</i> modernos (Chrome, Firefox, Edge, Safari) sem necessidade de instalação local.
RNF-04	Desempenho	Tempo de resposta inferior a 2 segundos para carregamento de listagens e telas operacionais normais.
RNF-05	Disponibilidade	O sistema deve estar disponível para acesso contínuo (24 horas por dia, 7 dias por semana).
RNF-06	Persistência	Garantia de integridade referencial e transações ACID no banco de dados relacional MySQL.

4.3 Protótipos de interface

A prototipagem de interface foi realizada na Sprint 2 para validar o fluxo visual do sistema antes do início da codificação. Os protótipos de alta fidelidade representaram fielmente as telas desenvolvidas na aplicação real.

- **Tela de Autenticação:** Apresenta um formulário limpo para inserção de credenciais de acesso de forma segura, sem exposição de dados de teste na interface;
- **Dashboard Administrativo:** Apresenta indicadores de desempenho consolidados (KPIs) voltados exclusivamente para o perfil de Administrador, exibindo o tempo médio de resposta e o volume de chamados ativos;
- **Lista de Chamados:** Exibe a listagem de chamados com filtros avançados por status, prioridade e categoria, permitindo a busca rápida por parte dos técnicos e usuários;
- **Interface de Criação de Novo Chamado:** Apresenta um formulário estruturado para que o solicitante registre sua demanda com clareza.

4.4 Diagrama de casos de uso

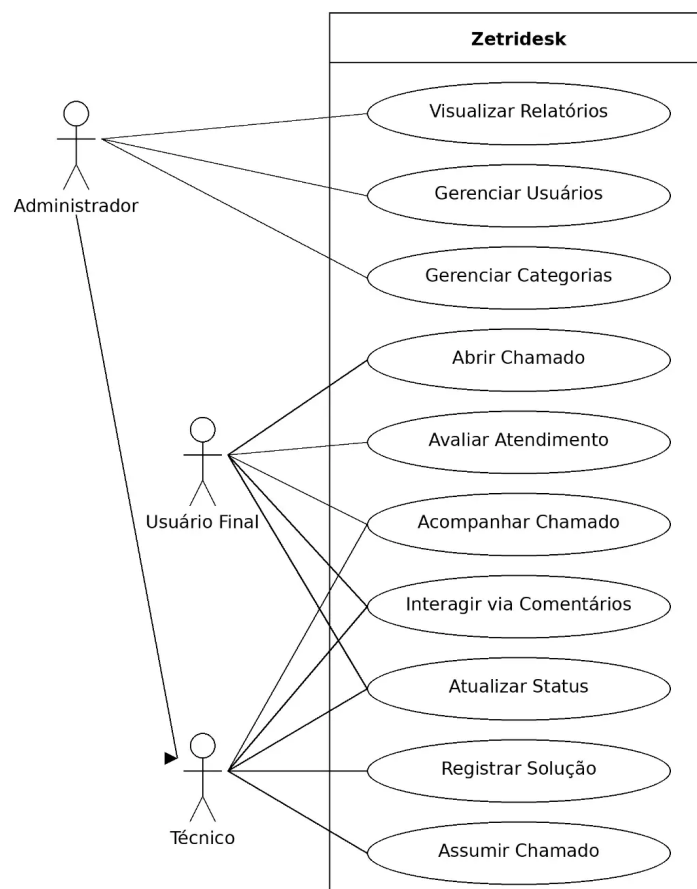
O Diagrama de Casos de Uso representa as interações dos atores com as funcionalidades do sistema. Os atores identificados foram: Solicitante, Técnico (atendente) e Administrador

(gestor).

Como pode ser observado na Figura 1, as permissões foram estritamente delimitadas por perfil de acesso (RBAC):

- O Solicitante possui permissão apenas para abrir chamados próprios, cancelá-los se necessário, avaliar o atendimento (pesquisa de satisfação) e interagir nas mensagens do chamado aberto;
- O Técnico possui permissão para visualizar a fila de chamados atribuídos a ele, reclassificar/escalonar o chamado, alterar o status do chamado para "Em Atendimento" ou "Concluído" e trocar mensagens com o solicitante;
- O Administrador possui controle total, incluindo o gerenciamento de usuários, a atribuição de técnicos aos chamados e o acesso a relatórios de desempenho da equipe de TI.

Figura 1 – Diagrama de Casos de Uso do Zetridesk



Fonte: Elaborado pelo autor (2026).

4.5 Diagrama de sequência

Para complementar a modelagem comportamental do sistema, foi desenvolvido o Diagrama de Sequência com o objetivo de representar, de forma cronológica e precisa, a troca de mensagens entre os participantes do sistema durante o fluxo de abertura de um chamado. O diagrama envolve quatro participantes: o **Cliente** (ator que inicia a solicitação), a **Interface** (componente `FormularioTicket`, responsável pela renderização e coleta dos dados), o **Controlador** (função `criar_ticket`, que concentra as regras de negócio da operação) e o **Banco de Dados** (camada de Modelos do Django, responsável pela persistência).

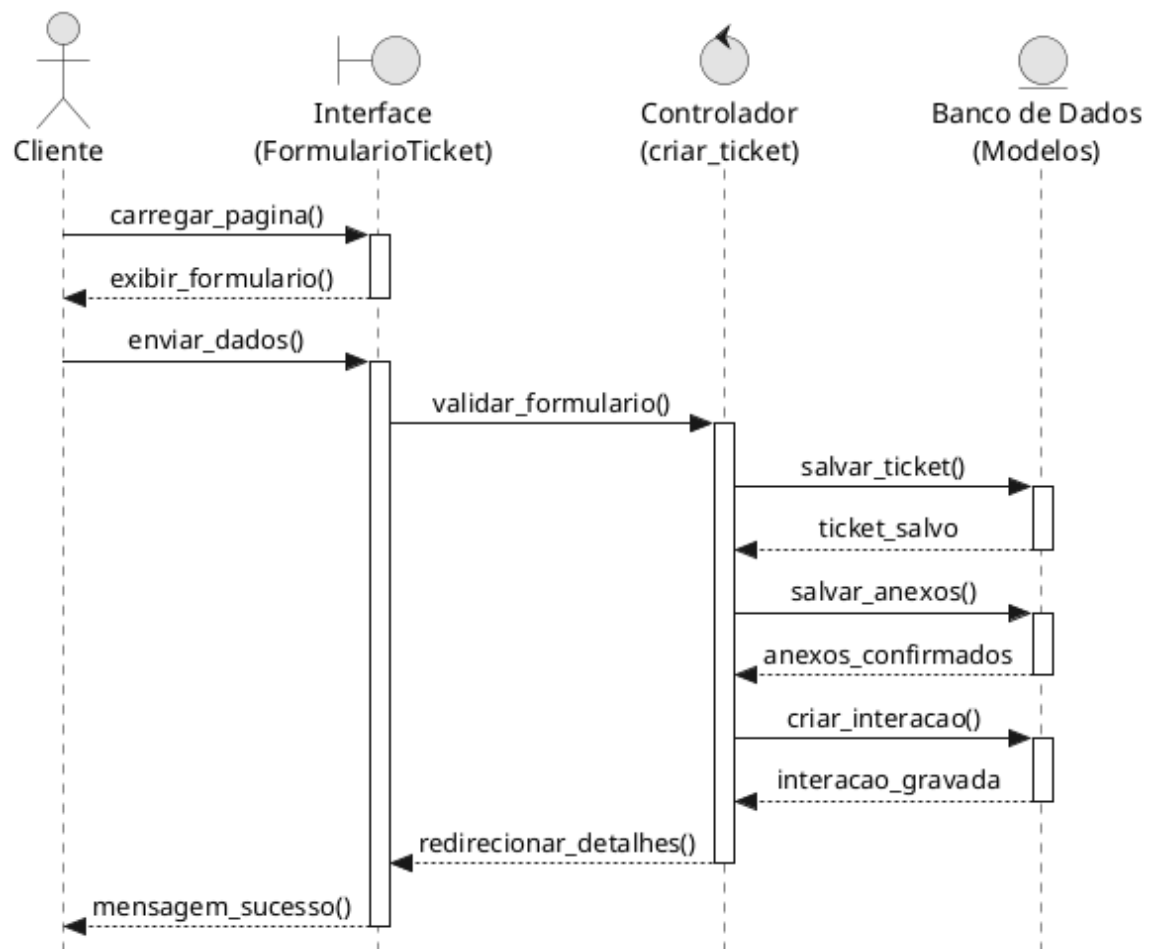
O fluxo de mensagens segue a seguinte sequência lógica de execução:

1. O **Cliente** solicita o carregamento da página de abertura de chamado por meio de `carregar_pagina()`; a **Interface** processa a requisição e retorna o formulário renderizado via `exibir_formulario()`;
2. O **Cliente** preenche os campos obrigatórios (título, descrição, categoria e prioridade) e submete o formulário por meio de `enviar_dados()`;
3. A **Interface** encaminha os dados ao **Controlador**, que executa `validar_formulario()` para verificar a integridade e completude dos campos recebidos antes de prosseguir com qualquer operação de escrita;
4. Aprovada a validação, o **Controlador** invoca `salvar_ticket()` junto ao **Banco de Dados**, que persiste o registro do chamado e confirma a operação retornando `ticket_salvo`;
5. Na sequência, os arquivos eventualmente anexados pelo **Cliente** são persistidos de forma independente por meio de `salvar_anexos()`, com confirmação `anexos_confirmados`;
6. O **Controlador** então invoca `criar_interacao()`, que registra no **Banco de Dados** a primeira entrada no histórico de interações do chamado. Essa interação inicial documenta automaticamente o momento exato da abertura, o identificador do chamado recém-criado e o usuário solicitante, estabelecendo o ponto de partida do rastreamento de todas as comunicações e alterações de status subsequentes. O **Banco de Dados** confirma a gravação retornando `interacao_gravada`;

7. Por fim, o **Controlador** instrui a **Interface** a executar `redirecionar_detalhes()`, encaminhando o **Cliente** para a página de detalhes do chamado e exibindo a mensagem de confirmação de sucesso via `mensagem_sucesso()`.

A Figura 2 ilustra o fluxo cronológico completo de mensagens entre os participantes do sistema durante o processo de criação de um chamado.

Figura 2 – Diagrama de Sequência — Fluxo de Abertura de Chamado



Fonte: Elaborado pelo autor (2026).

4.6 Diagrama de classes

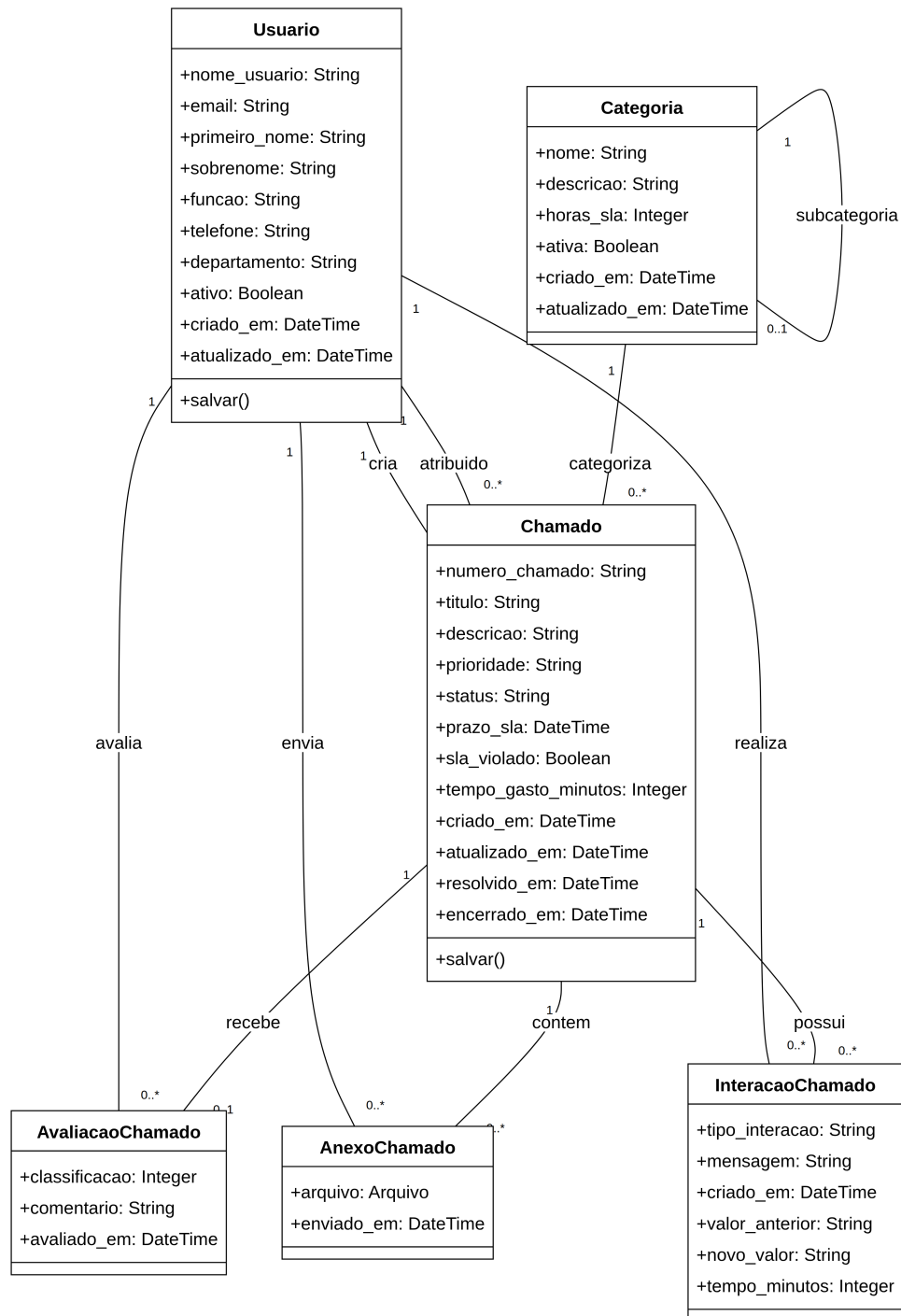
O Diagrama de Classes define a estrutura lógica das classes de domínio mapeadas no framework Django por meio de seu ORM. As classes principais desenvolvidas foram:

- **Usuário:** Representa o cadastro de usuários do sistema, contendo atributos como nome, e-mail, senha criptografada e o perfil de acesso (Solicitante, Técnico, Administrador);

- **Chamado:** Representa o registro do incidente ou requisição, contendo título, descrição, data de abertura, prioridade, status e as associações com o solicitante (Usuario) e o técnico atribuído (Usuario);
- **Interacao:** Representa as mensagens trocadas dentro de um chamado, contendo o texto da mensagem, data e hora do registro, associando diretamente o chamado correspondente e o usuário autor da mensagem;
- **HistoricoStatus:** Registra cada alteração de status do chamado, armazenando o status anterior, o novo status, a data e hora da mudança e o usuário responsável pela ação, garantindo total rastreabilidade.

A Figura 3 apresenta as relações estruturais e cardinalidades entre as classes que compõem o núcleo do sistema Zetridesk.

Figura 3 – Diagrama de Classes do Zetridesk



Fonte: Elaborado pelo autor (2026).

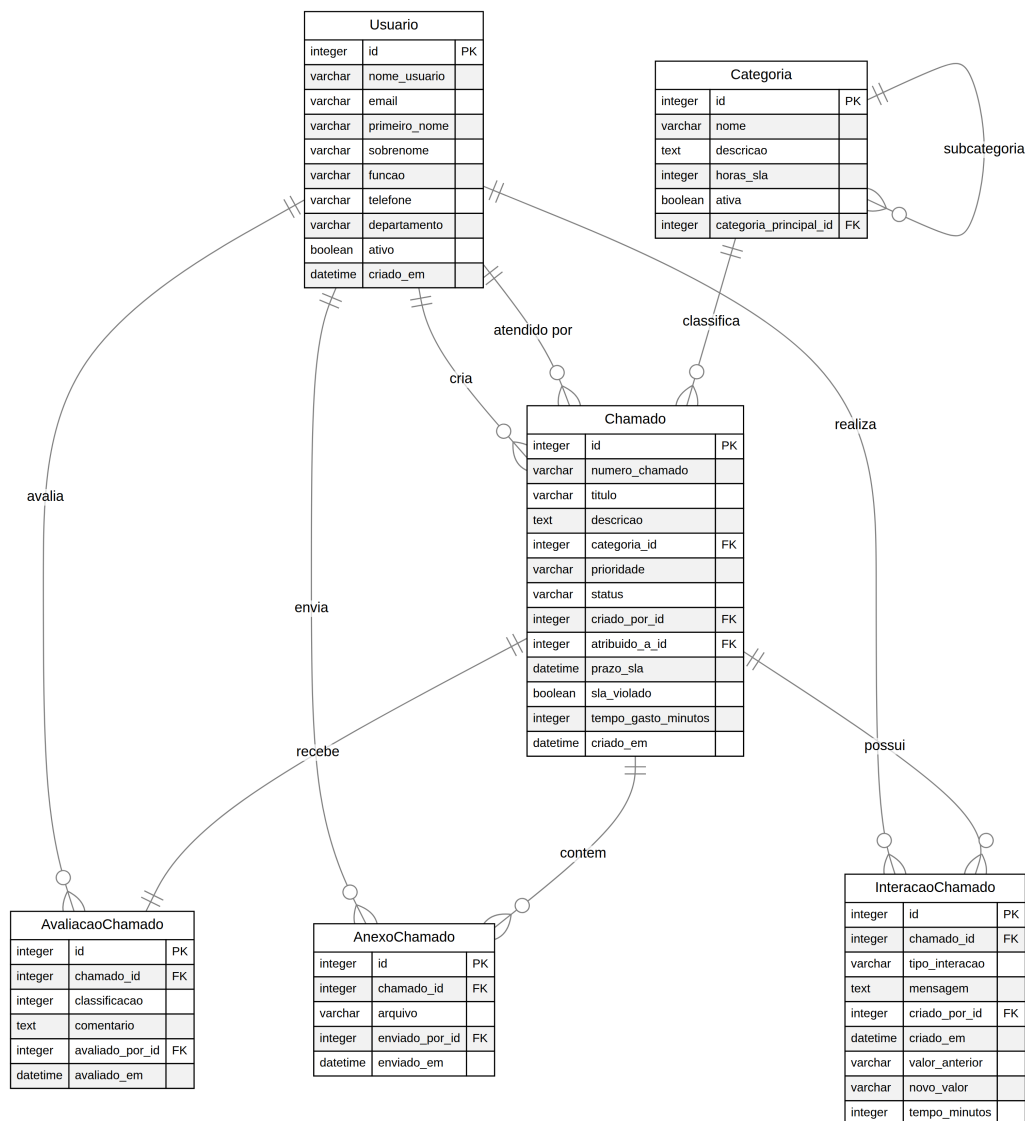
4.7 Diagrama entidade-relacionamento (DER)

O Diagrama de Entidade-Relacionamento (DER) mapeia a estrutura física das tabelas implementadas no SGBD relacional MySQL. A estrutura física reflete com precisão as relações lógicas definidas no diagrama de classes:

- A tabela **usuario** possui chave primária *id_usuario* e armazena os dados cadastrais e o perfil;
- A tabela **chamado** possui chave primária *id_chamado* e chaves estrangeiras *fk_solicitante* e *fk_tecnico* apontando para a tabela **usuario**, estabelecendo relacionamentos de integridade referencial;
- A tabela **interacao** possui chave primária *id_interacao* e chaves estrangeiras *fk_chamado* e *fk_autor* apontando respectivamente para as tabelas **chamado** e **usuario**, garantindo que cada mensagem esteja vinculada a um chamado e a um autor real;
- A tabela **historico_status** possui chave primária *id_historico*, chave estrangeira *fk_chamado* e chave estrangeira *fk_usuario* para registrar de forma auditável quem alterou o status do chamado e quando.

A Figura 4 apresenta o DER completo do sistema Zetridesk.

Figura 4 – Diagrama de Entidade-Relacionamento do Zetridesk



Fonte: Elaborado pelo autor (2026).

4.7.1 Dicionário de dados e tabelas do banco de dados

Para detalhar a implementação física do banco de dados relacional no MySQL, conforme solicitado pela banca examinadora, são apresentadas a seguir as tabelas de definição do esquema físico do Zetridesk, contendo os tipos de dados, restrições e descrições de cada campo.

Tabela 4 – Estrutura física da tabela usuario

Campo	Tipo	Chave	Descrição
id_usuario	INT AUTO_INCREMENT	PK	Identificador único do usuário
nome	VARCHAR(150)	-	Nome completo do usuário
email	VARCHAR(150) UNIQUE	-	E-mail institucional usado como login
senha	VARCHAR(128)	-	Senha criptografada utilizando algoritmo de hash seguro
perfil	VARCHAR(20)	-	Perfil de acesso (SOLICITANTE, TECNICO, ADMIN)

Fonte: Elaborado pelo autor (2026).

Tabela 5 – Estrutura física da tabela chamado

Campo	Tipo	Chave	Descrição
id_chamado	INT AUTO_INCREMENT	PK	Identificador único do chamado
titulo	VARCHAR(100)	-	Título ou resumo curto da demanda
descricao	TEXT	-	Descrição detalhada do incidente/solicitação
prioridade	VARCHAR(15)	-	Nível de urgência (ALTA, MEDIA, BAIXA)
status	VARCHAR(20)	-	Status atual (PENDENTE, ATENDIMENTO, CONCLUIDO, CANCELADO)
data_abertura	DATETIME	-	Data e hora de criação do chamado
prazo_sla	DATETIME	-	Data e hora limite calculada para atendimento
fk_solicitante	INT	FK	Referência ao usuário que abriu o chamado
fk_tecnico	INT NULL	FK	Referência ao técnico atribuído (pode ser nulo)

Fonte: Elaborado pelo autor (2026).

Tabela 6 – Estrutura física da tabela interacao

Campo	Tipo	Chave	Descrição
id_interacao	INT AUTO_INCREMENT	PK	Identificador único da mensagem
mensagem	TEXT	-	Conteúdo textual da interação
data_registro	DATETIME	-	Data e hora do envio da mensagem
fk_chamado	INT	FK	Referência ao chamado correspondente
fk_autor	INT	FK	Referência ao usuário autor da mensagem

Fonte: Elaborado pelo autor (2026).

Tabela 7 – Estrutura física da tabela avaliacao

Campo	Tipo	Chave	Descrição
id_avaliacao	INT AUTO_INCREMENT	PK	Identificador único da avaliação
nota	INT	-	Nota de satisfação atribuída (1 a 5)
comentario	TEXT NULL	-	Comentário qualitativo do usuário (opcional)
data_avaliacao	DATETIME	-	Data e hora do preenchimento
fk_chamado	INT UNIQUE	FK	Referência ao chamado avaliado (relação 1:1)

Fonte: Elaborado pelo autor (2026).

4.8 Algoritmos propostos

Para detalhar o funcionamento lógico dos mecanismos mais críticos do sistema Zetridesk, são apresentados nesta seção os algoritmos propostos para a geração do identificador de chamados, o cálculo e a verificação do prazo limite de atendimento (SLA), e o controle de acesso baseado em papéis (RBAC). Os algoritmos foram modelados em pseudocódigo utilizando as estruturas de controle do pacote *algorithmic*.

4.8.1 Geração de identificador único de chamado

A geração do identificador único do chamado deve garantir a integridade e a ordenação cronológica das demandas. No banco de dados relacional MySQL, isso é resolvido de forma eficiente por meio de uma chave primária numérica auto-incremental combinada a um prefixo alfanumérico que facilita a identificação visual rápida pelos técnicos de suporte, conforme apresentado no Algoritmo 1.

Algorithm 1 Geração de Identificador Único de Chamado

Require: Ano corrente (*ano*), identificador sequencial do banco de dados (*id*)

Ensure: Identificador formatado do chamado (*id_formatado*)

```

1: prefixo ← ZTD
2: ano_str ← ConverterParaTexto(ano)
3: id_str ← ConverterParaTexto(id)
4: enquanto Comprimento(id_str) < 6 faça      ▷ Adiciona zeros à esquerda para 6 dígitos
5:   id_str ← "0" + id_str
6: fim enquanto
7: id_formatado ← prefixo + "-" + ano_str + "-" + id_str
8: retorne id_formatado

```

4.8.2 Cálculo e verificação de SLA

O cálculo do SLA determina o prazo limite para a resolução do chamado com base na prioridade informada no momento da abertura. O Algoritmo 2 apresenta a lógica para a definição desse prazo e para a verificação do status temporal da demanda (se está dentro do prazo ou com SLA estourado).

Algorithm 2 Cálculo e Verificação de SLA

Require: Data de abertura (*data_abertura*), Prioridade (*prioridade*), Data atual (*data_atual*)

Ensure: Prazo limite (*prazo_sla*), Situação do SLA (*status_sla*)

```

1: se prioridade = ALTA então                                ▷ Define o prazo em horas conforme a prioridade
2:   horas_limite  $\leftarrow$  4
3: senão se prioridade = MEDIA então
4:   horas_limite  $\leftarrow$  24
5: senão se prioridade = BAIXA então
6:   horas_limite  $\leftarrow$  72
7: senão
8:   horas_limite  $\leftarrow$  48                                ▷ Prazo padrão para demais categorias
9: fim se
10: prazo_sla  $\leftarrow$  data_abertura + AdicionarHoras(horas_limite)
11: se data_atual > prazo_sla então                            ▷ Verifica a situação do SLA
12:   status_sla  $\leftarrow$  SLA QUEBRADO
13: senão se (prazo_sla - data_atual)  $\leq$  ConverterHoras(2) então
14:   status_sla  $\leftarrow$  PROXIMO DO LIMITE
15: senão
16:   status_sla  $\leftarrow$  DENTRO DO PRAZO
17: fim se
18: retorne prazo_sla, status_sla

```

4.8.3 Controle de acesso baseado em papéis (RBAC)

O controle de acesso baseado em papéis (RBAC) garante que cada usuário acesse exclusivamente as funcionalidades e dados compatíveis com o seu perfil cadastrado. O Algoritmo 3 detalha a lógica de autorização aplicada às requisições do sistema para proteger as rotas administrativas e operacionais.

Algorithm 3 Controle de Acesso Baseado em Papéis (RBAC)

Require: Usuário solicitante (*usuario*), Rota requisitada (*rota*)

Ensure: Booleano indicando se o acesso é permitido (*autorizado*)

```

1: se  $\neg$  usuario.autenticado então                                ▷ Usuário não autenticado: acesso negado
2:   retorne Falso
3: fim se
4: perfil  $\leftarrow$  usuario.perfil
5: autorizado  $\leftarrow$  Falso
6: se perfil = ADMIN então                                          ▷ Administrador possui acesso total
7:   autorizado  $\leftarrow$  Verdadeiro
8: senão se perfil = TECNICO então
9:   se rota  $\in$  RotasOperacionais  $\vee$  rota  $\in$  RotasComuns então
10:    autorizado  $\leftarrow$  Verdadeiro
11:   fim se
12: senão
13:   se rota  $\in$  RotasComuns então
14:    autorizado  $\leftarrow$  Verdadeiro
15:   fim se
16: fim se
17: retorne autorizado

```

5 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados obtidos com o desenvolvimento e a validação funcional do sistema Zetridesk em ambiente de desenvolvimento local (*localhost*), descrevendo os testes de aceitação executados pelo próprio desenvolvedor e discutindo a viabilidade prática da solução proposta para a gestão de chamados em organizações de médio porte.

5.1 Resultados do desenvolvimento

O desenvolvimento do sistema Zetridesk resultou em uma aplicação *web* portátil, responsiva e funcional, plenamente integrada ao banco de dados relacional MySQL. A interface responsiva desenvolvida com Bootstrap 5 adaptou-se adequadamente a diferentes dispositivos, permitindo que os usuários registrassem chamados tanto por computadores de mesa quanto por smartphones.

A segurança da aplicação foi garantida na prática por meio da implementação de mecanismos nativos do Django, incluindo a criptografia de senhas usando algoritmo de hash seguro e a proteção automática contra vulnerabilidades *web* comuns, como ataques de falsificação de solicitação entre sites (CSRF) e injeção de scripts (XSS).

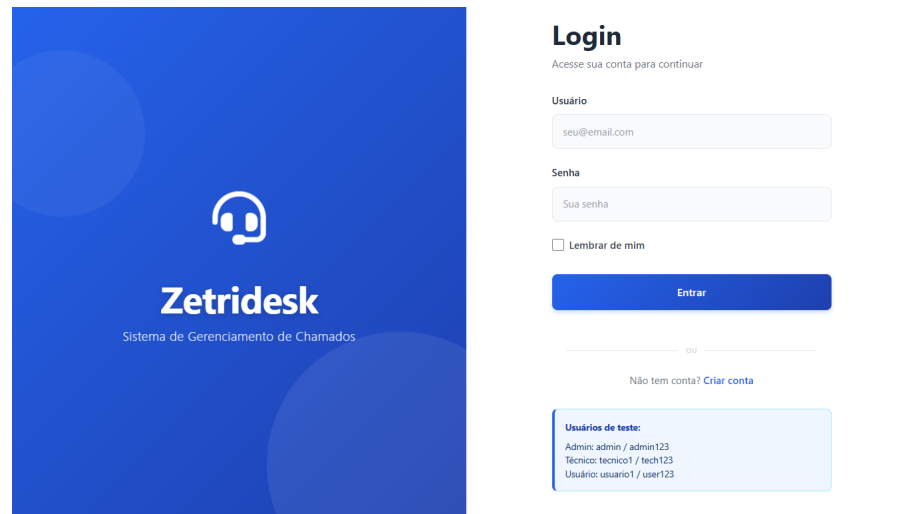
5.2 Apresentação das telas e funcionalidades desenvolvidas

Esta seção apresenta a interface gráfica e o funcionamento prático do sistema Zetridesk a partir das capturas de tela obtidas durante a fase de testes funcionais. O design da aplicação foi desenvolvido utilizando HTML5, CSS3 e o framework Bootstrap 5, garantindo uma interface moderna, intuitiva e responsiva (portável para dispositivos móveis).

5.2.1 Interface de autenticação

A tela de login, apresentada na Figura 5, é a porta de entrada para a plataforma. Ela implementa o mecanismo de autenticação segura do Django. Para fins de testes de aceitação e facilidade de homologação pelos voluntários, foi incluído temporariamente um quadro informativo contendo as credenciais de teste de cada perfil de acesso (Administrador, Técnico e Usuário Final).

Figura 5 – Interface de Autenticação do Zetridesk

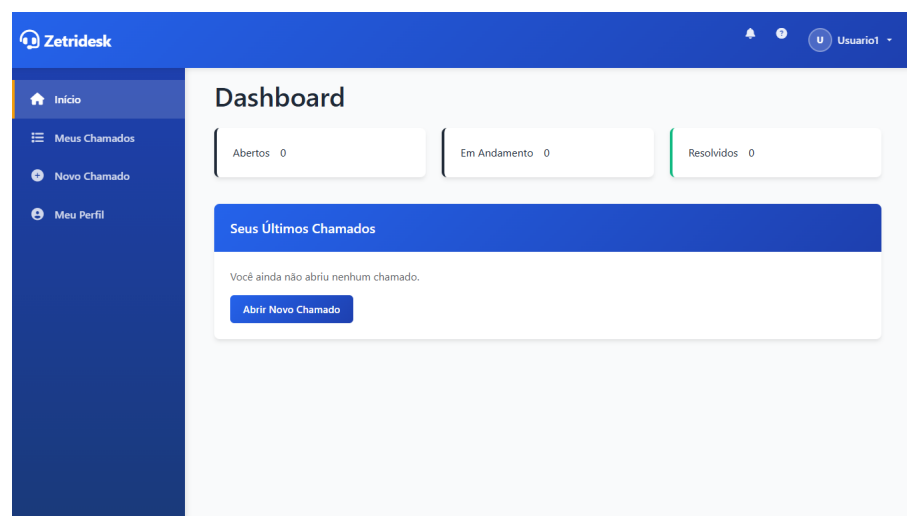


Fonte: Captura de tela obtida pelo autor (2026).

5.2.2 Painel do usuário solicitante

Após efetuar a autenticação com o perfil de solicitante, o usuário é direcionado para o seu painel de controle (dashboard), exibido na Figura 6. O menu lateral fornece acesso rápido às opções "Meus Chamados", "Novo Chamado" e "Meu Perfil". O painel principal exibe de forma sintética os cards quantitativos de chamados abertos, em andamento e resolvidos, além de uma tabela contendo as demandas recentes abertas pelo próprio usuário.

Figura 6 – Dashboard do Usuário Solicitante



Fonte: Captura de tela obtida pelo autor (2026).

5.2.3 Interface de abertura de chamados

A Figura 7 ilustra o formulário de abertura de novos chamados (RF-01). O usuário preenche o título de forma concisa, fornece uma descrição detalhada do incidente ou solicitação, seleciona a prioridade (Baixa, Média ou Alta) e a categoria correspondente à sua demanda (como Redes, Sistemas, Manutenção, entre outras). O formulário também oferece suporte à anexação de capturas de tela ou documentos auxiliares para detalhar o problema técnico, além de exibir dicas contextuais na lateral direita para orientar o preenchimento correto.

Figura 7 – Formulário de Criação de Novo Chamado

Zetridesk Usuário1

Criar Chamado

Título*
Título do chamado

Descrição*
Descrição detalhada
Preencha este campo.

Prioridade*
Média

Categoria*
-- Selecione uma categoria --

Anexos
Escolher arquivos Nenhum arquivo escolhido

Dicas

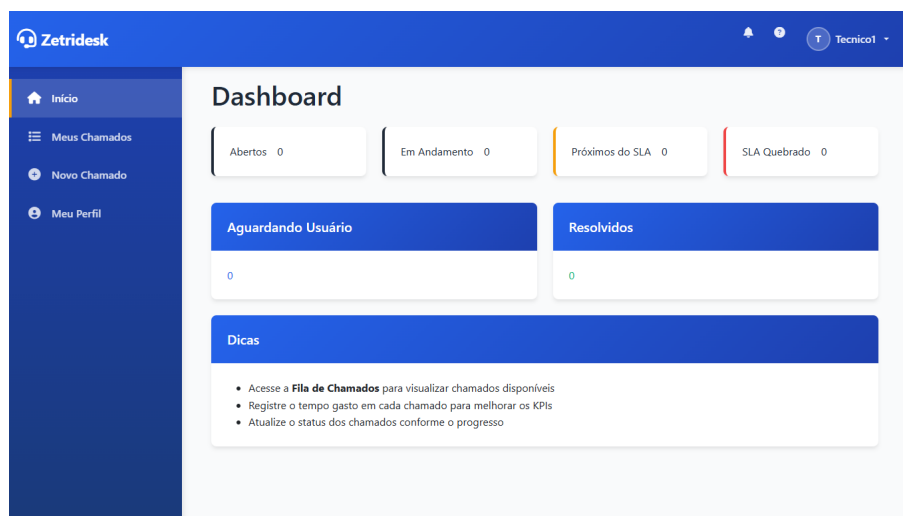
- Descreva o problema de forma clara e detalhada
- Selecione a categoria correta para melhor direcionamento
- Indique a prioridade conforme a urgência
- Anexe screenshots ou arquivos relevantes se necessário

Fonte: Captura de tela obtida pelo autor (2026).

5.2.4 Painel de atendimento do técnico

O dashboard do técnico de suporte, apresentado na Figura 8, exibe uma interface operacional focada na produtividade. Além dos cards tradicionais de chamados abertos e em andamento, o painel do técnico destaca de forma proeminente indicadores de monitoramento temporal de SLA, como "Próximos do SLA" e "SLA Quebrado". Isso permite que o técnico priorize de forma automática as demandas mais críticas que estão prestes a estourar o prazo limite acordado.

Figura 8 – Dashboard Operacional do Técnico de Suporte

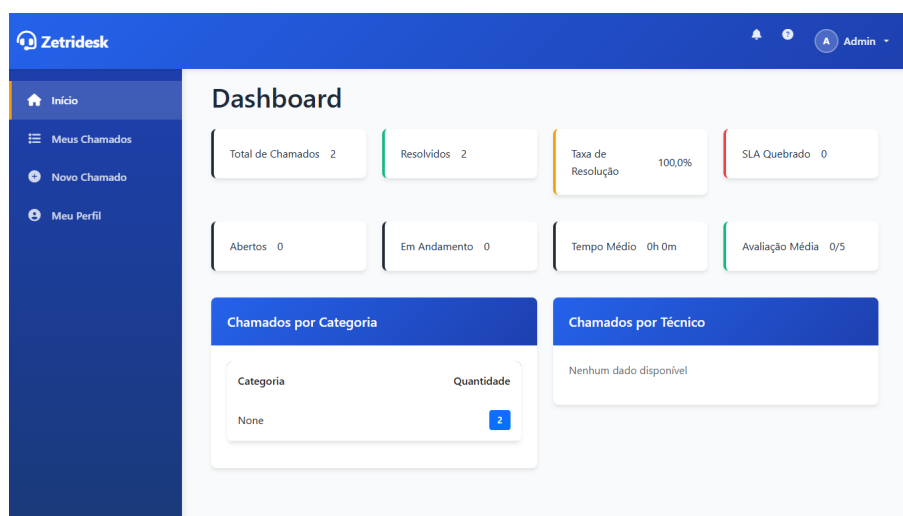


Fonte: Captura de tela obtida pelo autor (2026).

5.2.5 Pannel gerencial e indicadores do administrador

A Figura 9 apresenta o dashboard administrativo, voltado para a coordenação de TI da instituição. Este painel consolida métricas gerenciais cruciais para a governança de TI, como o total de chamados acumulados, a taxa de resolução (em porcentagem), o tempo médio de atendimento e a avaliação média de satisfação obtida através das pesquisas. O painel também exibe gráficos de distribuição de chamados por categoria e por técnico, permitindo ao gestor identificar gargalos de atendimento e planejar melhorias preventivas na infraestrutura da instituição.

Figura 9 – Dashboard Gerencial do Administrador

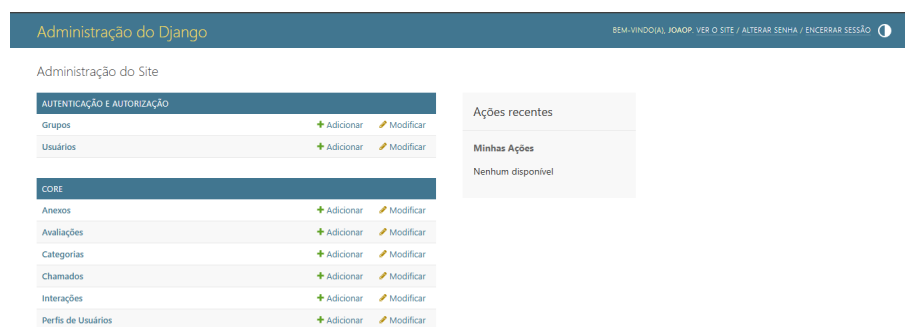


Fonte: Captura de tela obtida pelo autor (2026).

5.2.6 Módulo administrativo do *Django*

Para a administração avançada de dados do sistema, o Zetridesk utiliza a interface integrada do Django Admin, conforme ilustrado na Figura 10. Este módulo permite que o administrador do sistema gerencie de forma direta e segura as tabelas do banco de dados, como o cadastro de novos perfis de usuários, a criação de novas categorias de chamados, a auditoria de logs e o controle de grupos de permissão (autenticação e autorização), fornecendo flexibilidade e agilidade para a equipe de suporte avançado de TI.

Figura 10 – Painel de Administração Integrada do Django



Fonte: Captura de tela obtida pelo autor (2026).

5.3 Discussão sobre a demonstração e viabilidade do fluxo proposto

A demonstração prática do Zetridesk em ambiente local (*localhost*) comprovou a viabilidade técnica e a consistência operacional da solução desenvolvida. A arquitetura implementada em Python e Django mostrou-se altamente eficaz para centralizar os processos de suporte, fornecendo uma base sólida para a futura implantação em produção. A partir da análise das funcionalidades desenvolvidas e demonstradas, avaliou-se o cumprimento de cada um dos objetivos específicos estabelecidos na introdução deste trabalho:

- **Definição de Requisitos (Objetivo 1):** Plenamente atingido através da análise comparativa de soluções de mercado similares, consolidando os requisitos funcionais e não funcionais que guiaram o desenvolvimento sem desvios de escopo;
- **Modelagem do Sistema (Objetivo 2):** Concluído com sucesso através dos diagramas estruturais (classes, DER) e comportamentais (casos de uso, sequência) apresentados no Capítulo 4, garantindo a consistência física das tabelas do banco de dados MySQL;

- **Desenvolvimento das Funcionalidades (Objetivo 3):** Totalmente realizado com a codificação do sistema em Python/Django, incluindo as telas operacionais, o dashboard administrativo e as regras de negócio de cálculo de SLA;
- **Mecanismos de Segurança (Objetivo 4):** Implementado com sucesso através do controle de acesso baseado em papéis (RBAC) e criptografia de senhas, garantindo a proteção das credenciais e telas gerenciais;
- **Análise da Viabilidade (Objetivo 5):** Concluído através da avaliação qualitativa do fluxo de trabalho e da análise de desempenho operacional viabilizada pelo sistema desenvolvido.

A centralização das demandas em uma única plataforma elimina a perda de informações e o retrabalho decorrentes de solicitações duplicadas ou esquecidas. O fluxo de cancelamento de chamados (RF-08) e a reclassificação de chamados (RF-05) demonstram como o sistema centralizado otimiza o fluxo de trabalho técnico. A pesquisa de satisfação (RF-09) também se mostrou factível, permitindo que os solicitantes atribuam notas de 1 a 5 após o encerramento do chamado, integrando as notas diretamente ao banco de dados MySQL para futura análise gerencial.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento do Zetridesk cumpriu com êxito o propósito de projetar e codificar um sistema *web* para o gerenciamento de chamados de suporte técnico. O trabalho demonstrou que a utilização de tecnologias modernas e consagradas, como o framework Django e o SGBD MySQL, viabiliza a criação de uma ferramenta robusta, segura e altamente funcional, mesmo sob restrições de infraestrutura e orçamento.

A demonstração do sistema em ambiente local (*localhost*) evidenciou que a centralização de chamados em uma plataforma *web* resolve os problemas crônicos de fluxos informais de atendimento, fornecendo rastreabilidade completa das demandas, cálculo automatizado de SLA e indicadores gerenciais em tempo real. A segurança foi garantida através de controle de acesso baseado em papéis (RBAC) e criptografia para proteção das credenciais de acesso.

Como limitação do trabalho, destaca-se que o sistema não foi submetido a uma etapa de implantação definitiva em ambiente de produção institucional e nem a uma validação formal com usuários finais em larga escala, limitando-se à demonstração de viabilidade funcional em ambiente de desenvolvimento.

Como propostas de trabalhos futuros, sugerem-se as seguintes frentes de desenvolvimento:

- **Implantação em Ambiente de Produção:** Realizar a implantação definitiva do sistema em um servidor *web* institucional de uma organização para avaliar seu desempenho sob carga real de acessos;
- **Validação com Usuários Finais:** Conduzir uma pesquisa de usabilidade e satisfação formal junto a usuários reais (solicitantes e técnicos) após um período de uso contínuo em produção;
- **Aplicativo Móvel:** Desenvolver uma versão móvel do Zetridesk (para Android e iOS) utilizando frameworks como React Native, focando na facilidade de abertura de chamados e no recebimento de notificações push em tempo real para os técnicos de suporte;
- **Integração com Chatbots:** Implementar um módulo de triagem automatizada via chatbot integrado a canais de comunicação (como Telegram ou WhatsApp) para abertura automática de chamados de baixa complexidade.

REFERÊNCIAS

- BARBOSA, S. D. J.; SILVA, B. S. d. **Interação Humano-Computador**. Rio de Janeiro: Elsevier, 2010.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: Guia do Usuário**. 2. ed. Rio de Janeiro: Elsevier, 2005.
- CARVALHO, B. V. d.; MELLO, C. H. P. Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. **Gestão & Produção**, SciELO Brasil, v. 19, n. 3, p. 521–535, 2012.
- COSTA, E. C. d. A importância da engenharia de requisitos no processo de desenvolvimento de sistemas de informação. **Revista Interface Tecnológica**, v. 15, n. 1, p. 313–324, 2018.
- ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson, 2011.
- FOWLER, M. **UML Essencial: Um breve guia para a linguagem padrão de modelagem de objetos**. 3. ed. Porto Alegre: Bookman, 2005.
- LARMAN, C. **Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo**. 3. ed. Porto Alegre: Bookman, 2007.
- PERONDI, L. T. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação), **Sistema para gerenciamento de chamados técnicos**. Caxias do Sul: Universidade de Caxias do Sul, 2013. Disponível em: <https://repositorio.ucs.br/handle/11338/1246>.
- PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: uma abordagem profissional**. 9. ed. Porto Alegre: McGraw Hill, 2021.
- SOARES, M. d. S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **RESI - Revista Eletrônica de Sistemas de Informação**, v. 3, n. 1, 2004.
- SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Addison-Wesley, 2011.
- WATANABE, W. M. **Avaliação automática de acessibilidade em RIA**. Tese (Tese (Doutorado em Ciências de Computação e Matemática Computacional)) — Universidade de São Paulo, São Carlos, 2014.