**UNIVERSIDADE PAULISTA**

**ICET - INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA**

**CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**PROJETO INTEGRADO MULTIDISCIPLINAR**

**PIM IV**

# Desenvolvimento de um Sistema Integrado para Gestão de Chamados e Suporte Técnico com Apoio de IA



| Nome | R.A |
|---|---|
| ALUNO 1 - Andrei Henrique Mancijo | G922CG4 |
| ALUNO 2 - Filipe Vitor dos Santos | R084353 |
| ALUNO 3 - Jônatas dos Santos Souza | G9038F8 |
| ALUNO 4 - Kaique Batista da Silva | G03IGG0 |
| ALUNO 5 - Mariozan Damasceno Lacerda Júnior | G9884G2 |
| ALUNO 6 - Mateus Teodoro da Silva | G9265G4 |



**SÃO JOSÉ DOS CAMPOS – SP**

**NOVEMBRO / 2025**

| Integrantes | RA |
|---|---|
| Aluno 1 - Andrei Henrique Mancijo | G922CG4 |
| Aluno 2 - Filipe Vitor dos Santos | R084353 |
| Aluno 3 - Jônatas dos Santos Souza | G9038F8 |
| Aluno 4 - Kaique Batista da Silva | G03IGG0 |
| Aluno 5 - Mariozan Damasceno Lacerda Júnior | G9884G2 |
| Aluno 6 - Mateus Teodoro da Silva | G9265G4 |


# Desenvolvimento de um Sistema Integrado para Gestão de Chamados e Suporte Técnico com Apoio de IA

Projeto Integrado Multidisciplinar (PIM) desenvolvido como exigência parcial dos requisitos obrigatórios à aprovação semestral no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da UNIP (Universidade Paulista), orientado pelo corpo docente do curso.

**São José dos Campos – SP**

**NOVEMBRO / 2025**

## RESUMO

Este trabalho apresentou o desenvolvimento de um sistema integrado para gestão de chamados e suporte técnico, com base em Inteligência Artificial (IA), visando atender à crescente demanda por soluções tecnológicas que otimizem o atendimento técnico nas organizações. O objetivo principal foi criar uma ferramenta capaz de realizar a triagem inicial, categorização automática de chamados e sugestão de soluções, com o intuito de reduzir a sobrecarga da equipe de TI e agilizar o tempo de resposta aos usuários. A metodologia Scrum foi adotada para o gerenciamento das etapas do projeto, garantindo organização ágil e entrega contínua de resultados. O levantamento de requisitos foi conduzido em parceria com uma empresa de grande porte chamada SONDA, permitindo identificar com precisão as necessidades reais do sistema, com base nesses dados coletados foi criado uma empresa fictícia de médio porte. Como parte do desenvolvimento, elaborou-se um protótipo funcional com acesso administrativo e estruturação de banco de dados em MS SQL Server, abrangendo informações de usuários, equipe de TI, supervisores e triagem. A IA foi integrada ao sistema para atuar como chatbot na triagem inicial e como ferramenta de apoio na análise e encaminhamento de chamados. Foram aplicadas diretrizes da Lei Geral de Proteção de Dados (LGPD) para garantir segurança e conformidade no tratamento dos dados pessoais. A expectativa é que a solução implementada contribua para a redução na quantidade de chamados pendentes, diminua o tempo de espera e potencialmente aumente a satisfação dos usuários, indicando um possível impacto positivo.



**Palavras-chave:** Suporte técnico, Inteligência Artificial, gestão de chamados, Scrum, LGPD, banco de dados, chatbot.

# SUMÁRIO

# 1 INTRODUÇÃO

Atualmente, o setor de TI enfrenta desafios significativos na gestão de chamados, especialmente no que diz respeito ao tempo de espera para resolução das solicitações. Estudos indicam que a demora na resposta impacta diretamente a produtividade das equipes e a satisfação dos usuários (TOPdesk, 2023). A falta de um sistema eficiente pode resultar em sobrecarga dos profissionais de suporte e dificuldades na priorização de demandas (Desk Manager, 2023). A empresa analisada realiza a triagem inicial por meio do atendimento pelo Service Desk, utilizando telefone e e-mail como principais canais de comunicação.

Esse modelo, embora funcional, pode gerar um alto volume de solicitações simultâneas, dificultando a triagem e o encaminhamento adequado dos chamados. Diante desse cenário, o objetivo deste projeto é desenvolver um sistema de gerenciamento de chamados com IA integrada, permitindo que os usuários internos da empresa realizem a abertura de chamados diretamente na plataforma, reduzindo a dependência de e-mails e telefonemas. A inteligência artificial atuará na triagem das solicitações, validando a categoria do problema e, caso seja uma questão simples, fornecerá orientações ao usuário para a resolução imediata. Se o problema for mais complexo ou não puder ser solucionado automaticamente, o sistema encaminhará a solicitação para as equipes responsáveis, garantindo maior eficiência e agilidade no atendimento.

# 2 OBJETIVO GERAL

Desenvolver e implementar um sistema de suporte técnico inteligente baseado nos requisitos levantados na documentação do semestre anterior, integrando recursos de Inteligência Artificial para realizar a triagem inicial e a categorização automática de chamados, além de sugerir soluções para problemas recorrentes. O sistema incluirá um chatbot interativo e um chat de acompanhamento pós-abertura de chamados, visando otimizar o fluxo de atendimento e reduzir a sobrecarga da equipe de TI. As classificações geradas pela IA passarão pela validação da equipe de Triagem antes de serem encaminhadas às equipes técnicas, garantindo precisão no tratamento das solicitações. Todas as funcionalidades que envolvem dados pessoais serão desenvolvidas em conformidade com a LGPD.

## 2.1 Objetivos Específicos

- Com o propósito de atingir o objetivo geral proposto, serão considerados os seguintes objetivos específicos:
- Aplicar metodologia ágil Scrum para gerenciamento de projetos, dividindo tarefas para o desenvolvimento do sistema.
- Desenvolver artefatos UML, incluindo diagramas de Classe de Implementação, sequência e colaboração, para representar a estrutura e funcionamento do sistema.
- Criar protótipos de interface gráfica para desktop, web e mobile, garantindo acessibilidade e usabilidade para os usuários internos.
- Estruturar o banco de dados utilizando MS SQL Server, definindo os principais relacionamentos entre tabelas para suportar a operação do sistema.
- Implementar IA na classificação automática de chamados, integrando um chatbot para triagem inicial e um chat de acompanhamento para melhorar a interação entre usuário e suporte técnico.

- Validar as triagens da IA inicialmente com a equipe de Triagem, assegurando a correção da categorização e da coleta de informações antes da transferência.
- Desenvolver estratégias para garantir conformidade com a LGPD, aplicando boas práticas na manipulação e proteção dos dados dos usuários.

# 3 PROGRAMAÇÃO ORIENTADA A OBJETOS II

Como a maioria dos grandes aplicativos de grande porte tem o seu desenvolvimento de modo procedural, ocorre dificuldades na integração de novos módulos criados por diferentes setores e posteriormente é causado um grande custo para manutenções e atualizações (AGOSTINI; DECKER; SILVA, 2002).

Para um sistema de grande porte, é recomendado utilizar o paradigma de programação orientado a objetos para sua estrutura, por possibilitar modular, reutilizar e facilitar manutenções futuras (RAUT, 2020).

Na Modelagem Orientada a Objetos, é priorizado a clareza e organização, usando abstrações do mundo real para entendimento, facilitando ao máximo a construção, manutenções e atualizações futuras. Este conceito de **Abstração** possibilita a utilização da criação de um sistema feito com conceitos de objetos, que devem ser devidamente encapsulados (AGOSTINI; DECKER; SILVA, 2002).

Na abstração, os objetos são representações de entidades, conceitos ou processos do mundo real, focando em suas características essenciais e desconsiderando detalhes irrelevantes (AGOSTINI; DECKER; SILVA, 2002).

Este conceito está diretamente ligado ao **Encapsulamento**, que consiste em agrupar os dados (atributos) e os métodos que os manipulam dentro de uma única unidade, a classe. A classe encapsula sua lógica interna, protegendo seus dados de acessos indevidos e garantindo a integridade do objeto (AGOSTINI; DECKER; SILVA, 2002). Por exemplo, a alteração de um atributo não é feita diretamente, mas através de um método público que pode conter regras de negócio e validações.

Para organizar e promover o reuso de código, utiliza-se o conceito de **Herança**. Este pilar permite a criação de uma classe base, que contém atributos e métodos comuns. A partir dela, classes mais específicas (classes-filhas) são derivadas, herdando todas as características da classe base e adicionando as suas próprias.

A herança abre caminho para o **Polimorfismo**, que é a capacidade de um objeto de uma classe derivada ser tratado como um objeto de sua classe base (RAUT,

2020). Isso permite que um método definido na classe base seja sobrescrito (*override*) pelas classes filhas com suas próprias implementações.

Dessa forma, o sistema pode invocar o método em um objeto do tipo da classe base, e o comportamento correto será executado em tempo de execução, dependendo de qual classe-filha a instância realmente pertence. Isso simplifica o código, tornando-o mais flexível e extensível, pois novos tipos de classes derivadas podem ser adicionados no futuro sem a necessidade de alterar o código que os manipula.

# 4 DESENVOLVIMENTO DE SOFTWARE PARA INTERNET

O desenvolvimento de software para a Internet exige a adoção de métodos, padrões e tecnologias que garantam segurança, responsividade, modularidade e experiência adequada ao usuário. De acordo com Sommerville (2011) e Pressman (2016), a aplicação de boas práticas de Engenharia de Software é essencial para sistemas de médio porte, assegurando organização, escalabilidade e manutenção contínua. No âmbito da usabilidade, Nielsen (2012) destaca a importância de interfaces claras, eficientes e orientadas ao usuário, princípios incorporados no presente trabalho. Ressalta-se que este capítulo aborda exclusivamente a versão Web do sistema, considerando que o projeto também contempla versões Desktop e Mobile, porém tratadas em outras etapas do desenvolvimento.

Para a camada de back-end, adotou-se a pilha Python + Flask, que possibilita uma arquitetura leve, modular e compatível com os requisitos do projeto. A persistência de dados foi estruturada com Flask-SQLAlchemy, utilizando mapeamento objeto-relacional. O controle de versões do banco foi executado com Flask-Migrate/Alembic, garantindo rastreabilidade nas migrações. A comunicação com o Microsoft SQL Server foi realizada por meio do pyodbc, atendendo às necessidades do ambiente institucional. O gerenciamento seguro de variáveis sensíveis, como credenciais de banco e chaves de API, foi implementado com python-dotenv.

Quanto à segurança, utilizaram-se Flask-WTF/WTForms para validação e proteção de formulários e funções do Werkzeug para hashing de senhas, alinhando-se às recomendações da OWASP (2021). A integração com inteligência artificial generativa foi conduzida com a biblioteca google-generativeai, cuja chave de acesso (GEMINI_API_KEY) é lida exclusivamente via variáveis de ambiente, seguindo boas práticas de proteção de credenciais.

A garantia de qualidade foi reforçada por meio de pytest, aplicado ao back-end da versão Web, cobrindo rotas, fluxos de processamento e validações internas. No front-end, empregaram-se HTML5, CSS3 e JavaScript, adotando também o framework Bootstrap, que possibilitou a criação de interfaces responsivas, acessíveis e padronizadas. A versão Web do sistema foi construída de forma responsiva, semântica e modularizada, utilizando variáveis root para centralização de estilos, favorecendo manutenibilidade e padronização visual. A prototipação de telas ocorreu no Figma, permitindo a realização de Testes de Usabilidade baseados nos princípios de Nielsen, com usuários representativos avaliando clareza, navegabilidade e eficiência da interação.

Os requisitos da disciplina foram plenamente atendidos, contemplando prototipagem validada por Testes de Usabilidade, desenvolvimento de um sistema Web responsivo e implementação de mecanismos de autenticação e autorização. Toda a documentação e estrutura da versão Web encontram-se organizadas nos arquivos requirements.txt, config.py, README.md e nos diretórios migrations/, static/ e tests/ do repositório oficial do projeto.

Dessa forma, a versão Web do sistema demonstra aderência às melhores práticas de Engenharia de Software, incorporando tecnologias consolidadas, princípios de usabilidade e uma abordagem arquitetural responsável, eficiente e segura.

# 5 TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

A disciplina de Tópicos Especiais de Programação Orientada a Objetos (POO) tem como objetivo aplicar conceitos avançados de orientação a objetos aliados à lógica e boas práticas de programação. No projeto UpDesk — Sistema Integrado para Gestão de Chamados e Suporte Técnico com Inteligência Artificial (IA), esses conceitos foram implementados de forma prática, contemplando tanto fundamentos estruturais quanto técnicas modernas de desenvolvimento.

Segundo Wazlawick (2014), a orientação a objetos permite representar entidades do mundo real por meio de classes e objetos, facilitando a modelagem e manutenção de sistemas. No UpDesk, essa abordagem foi aplicada na definição de classes como Usuário, Chamado e Técnico, utilizando herança, polimorfismo e abstração para promover o reuso de código e reduzir o acoplamento entre módulos (Bezerra, 2014).

A disciplina também destacou o uso de funções e modularização do código, conforme apresentado nas aulas sobre argumentos posicionais, parâmetros nomeados e valores de retorno. No projeto, esses conceitos foram utilizados para dividir o sistema em componentes independentes, facilitando a manutenção e os testes de unidade.

As estruturas condicionais (if, elif, else) e os laços de repetição (for e while) foram amplamente utilizadas para controle de fluxo nas operações de abertura, acompanhamento e encerramento de chamados. O uso de listas e tuplas também foi essencial para o armazenamento e manipulação de dados em memória temporária, contribuindo para a organização dos registros de suporte.

Além disso, foram aplicados testes automatizados de unidade (com o framework pytest) para validar classes e métodos críticos, assegurando a confiabilidade e estabilidade do sistema. Conforme Sommerville (2019), a automação de testes é uma prática indispensável para a engenharia de software moderna.

Para garantir flexibilidade e reuso, o projeto utilizou padrões de projeto como Singleton (para controle centralizado de conexão ao banco de dados) e Factory

Method (para criação de objetos relacionados a chamados e notificações). Gamma et al. (1995) ressaltam que esses padrões padronizam soluções e aumentam a escalabilidade do sistema.

Por fim, princípios de lógica e algoritmos orientaram a estrutura sequencial e condicional do programa, garantindo clareza e coerência na execução das funcionalidades.

A aplicação integrada desses conceitos — lógica, funções, listas, estruturas de decisão, testes automatizados e POO — foi decisiva para que o UpDesk atingisse uma arquitetura sólida, modular e expansível, capaz de evoluir para versões Web, Desktop e Mobile.

# 6 PROJETO DE SISTEMAS ORIENTADO A OBJETOS

A disciplina Projeto de Sistemas Orientada a Objetos propõe a aplicação de recursos técnicos e metodológicos que são fundamentais para o desenvolvimento de soluções eficientes em ambientes computacionais. Entre esses recursos, destaca-se o uso de diagramas UML como elementos essenciais para o projeto e implementação de sistemas voltados à administração de pequenos empreendimentos.

O diagrama de classe de implementação é utilizado para representar a estrutura estática do sistema, evidenciando as classes, seus atributos, métodos e os relacionamentos entre elas. Esse modelo é crucial para organizar o código de forma modular, facilitar a manutenção e promover a reutilização de componentes. Além disso, permite uma visão clara das responsabilidades de cada classe, contribuindo para a robustez da arquitetura do sistema (Silveira, 2024).

O diagrama de sequência, por sua vez, descreve a interação temporal entre os objetos envolvidos em um determinado processo. Ele detalha a troca de mensagens entre os componentes, permitindo compreender o fluxo de execução das funcionalidades. Essa representação é essencial para validar os requisitos funcionais, identificar dependências e garantir que o comportamento do sistema esteja alinhado com os objetivos do projeto (Lucidchart, 2024).

Complementando essa abordagem, o diagrama de colaboração foca na organização estrutural das interações entre os objetos. Ele evidência como os elementos do sistema cooperam entre si para realizar tarefas específicas, destacando os vínculos e responsabilidades compartilhadas. Essa perspectiva é útil para refinar a arquitetura do sistema e promover uma melhor distribuição das funções entre os componentes (Miro, 2024).

A metodologia adotada para o desenvolvimento do sistema foi iterativa e incremental, com base na linguagem UML. Essa abordagem permite ajustes contínuos ao longo do ciclo de vida do projeto, favorecendo a adaptação às mudanças de requisitos. Foram aplicados princípios como encapsulamento, para proteger dados sensíveis; polimorfismo, para permitir variações de comportamento conforme o tipo

de usuário; e herança, para facilitar a expansão de funcionalidades. Tais estratégias oferecem vantagens como redução de retrabalho, clareza na documentação e facilidade de integração com tecnologias emergentes, como inteligência artificial e serviços em nuvem. As perspectivas de evolução incluem a ampliação da interoperabilidade com outras plataformas, o uso de aprendizado de máquina para análise preditiva e a adaptação do sistema a dispositivos móveis e assistentes virtuais.

# 7 GERENCIAMENTO DE PROJETO DE SOFTWARE

O gerenciamento de projetos de software consiste em aplicar princípios, técnicas e ferramentas de planejamento, organização e controle às atividades envolvidas no desenvolvimento de sistemas. De acordo com Pressman (2016) e Sommerville (2019), o objetivo fundamental dessa prática é garantir que o projeto seja concluído dentro do prazo, do orçamento e com a qualidade esperada, atendendo às necessidades do cliente e aos requisitos previamente definidos.

No contexto do projeto UpDesk, o gerenciamento foi conduzido com base nas boas práticas descritas pelo Project Management Institute (PMI®), descritas no Guia PMBOK, e complementadas por metodologias ágeis, especialmente o Scrum. Essa abordagem híbrida permitiu à equipe maior flexibilidade, priorização de entregas incrementais e controle contínuo de riscos, assegurando a evolução constante do sistema integrado de gestão de chamados com apoio de Inteligência Artificial.

# 8 EMPREENDEDORISMO

O empreendedorismo é o motor que transforma ideias em realidade e impulsiona o sucesso de um negócio. Mais do que apenas abrir uma empresa, o empreendedorismo representa um processo contínuo de inovação e adaptação que exige iniciativa, visão e capacidade de lidar com riscos. Segundo Dornelas (2018), o empreendedor é aquele que identifica oportunidades e transforma ideias em negócios rentáveis, agregando valor econômico e social. Nesse sentido, empreender significa fazer as coisas acontecerem, antecipando-se às situações e mantendo uma visão clara do futuro do negócio (CHIAVENATO, 2020).

Um empreendedor é, portanto, um indivíduo proativo e inovador, capaz de visualizar oportunidades e transformá-las em empreendimentos sustentáveis. Dolabela (2008) destaca que o sucesso empreendedor está diretamente ligado à execução disciplinada e à busca constante por aprendizado. Assim, a implementação bem-sucedida de uma ideia em uma oportunidade de negócio representa o ponto de partida para a criação de um empreendimento sólido e competitivo.

# 9 GESTÃO DA QUALIDADE

A gestão da qualidade de software tem como objetivo garantir que o produto final atenda aos requisitos do cliente, mantenha padrões de confiabilidade, desempenho e usabilidade, além de permitir sua manutenção e evolução com segurança. No projeto UpDesk – Sistema Integrado de Gestão de Chamados e Suporte Técnico com IA, foram aplicados conceitos e ferramentas da Gestão da Qualidade Total (TQM – Total Quality Management), buscando assegurar a eficiência dos processos de desenvolvimento e a satisfação dos usuários finais.

Segundo Pressman e Maxim (2016), a qualidade de software está diretamente relacionada à adequação ao uso, ou seja, à capacidade do sistema de executar suas funções de forma precisa, sem falhas e de acordo com as expectativas do usuário. Com base nisso, o projeto adotou práticas de controle e melhoria contínua, utilizando a ferramenta Ciclo PDCA (Plan, Do, Check, Act) como referência metodológica para o acompanhamento da qualidade durante as etapas de desenvolvimento.

Impactos Tecnológicos e Operacionais

A aplicação eficiente da ferramenta PDCA e dos indicadores de qualidade trouxe impactos positivos tanto tecnológicos quanto operacionais:

- Tecnológicos: aumento da confiabilidade do software, redução de bugs em produção e maior estabilidade da integração entre módulos.
- Operacionais: melhoria da produtividade da equipe, identificação antecipada de falhas e comunicação mais eficiente entre os membros do time.

Esses resultados evidenciam que o uso de ferramentas de gestão da qualidade — especialmente o PDCA — foi determinante para o controle de processos e para a composição de métricas que asseguraram o desenvolvimento de um software de alto padrão.

# 10 DESENVOLVIMENTO DO PROJETO

### 10.1.1 Python - run.py

Aplicação, onde está toda a lógica e processamento. O arquivo run.py é o ponto de partida que inicia o sistema.

### 10.1.2 Flask (microframework) - create_app (app/init.py) e run.py

O Flask é um microframework Python, ou seja, um conjunto de ferramentas que facilita a construção de aplicações web.

Papel Principal: Ele é responsável pelo Roteamento (direcionar URLs, tipo /login ou /produtos, para a função Python correta), pelo Contexto de Requisição (lidar com os dados que chegam do usuário) e por ser o ponto central da aplicação.

Figura 1 - Entry Point

```
"""
Ponto de Entrada da Aplicação (Entry Point)
"""
from app import create_app

app = create_app()

if __name__ == '__main__':
    app.run(debug=app.config.get('DEBUG', False)
```

Fonte: Gerada pelo autor

### 10.1.3 Blueprints / Modularização

É uma maneira de organizar a aplicação Flask em componentes independentes e reutilizáveis.

- Onde: registros em create_app (app.register_blueprint(...)).
- Papel: separar módulos (auth, chamados, admin) em pacotes testáveis e independentes.

Exemplo:

Figura 2 - Blueprint de chamados (rotas CRUD simples)

```python
from flask import Blueprint, render_template, request, redirect, url_for, flash
from ..extensions import db
from ..models import Chamado

chamados_bp = Blueprint('chamados', __name__, template_folder='templates/chamados')

@chamados_bp.route('/', methods=['GET'])
def index():
    lista = Chamado.query.all()
    return render_template('chamados/index.html', chamados=lista)

@chamados_bp.route('/create', methods=['GET', 'POST'])
def create():
    if request.method == 'POST':
        titulo = request.form.get('titulo')
        descricao = request.form.get('descricao')
        c = Chamado(titulo=titulo, descricao=descricao)
        db.session.add(c)
        db.session.commit()
        flash('Chamado criado', 'success')
        return redirect(url_for('chamados.index'))
    return render_template('chamados/create.html')
```

Fonte: Gerada pelo Autor

### 10.1.4 Flask-SQLAlchemy

- Onde: extensão inicializada em app/extensions.py ou app/**init**.py; modelos em app/models.py.
- Papel: ORM para mapear classes Python para tabelas; configuração via SQLALCHEMY_DATABASE_URI em config.py.

Figura 3 - Mapear tabelas

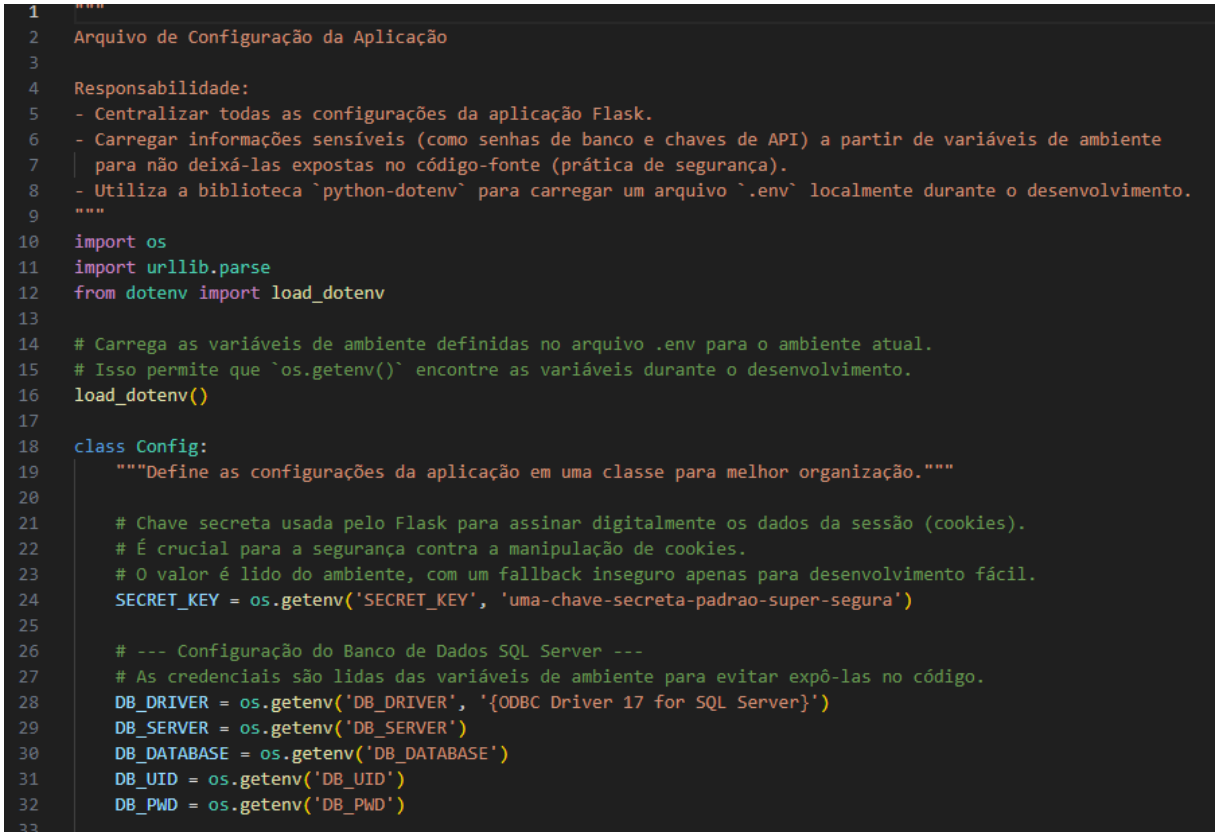
```python
from .extensions import db
from datetime import datetime

class Usuario(db.Model):
    __tablename__ = 'usuarios'
    id = db.Column(db.Integer, primary_key=True)
    nome = db.Column(db.String(120), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    senha_hash = db.Column(db.String(255), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

class Chamado(db.Model):
    __tablename__ = 'chamados'
    id = db.Column(db.Integer, primary_key=True)
    titulo = db.Column(db.String(200), nullable=False)
    descricao = db.Column(db.Text, nullable=True)
    usuario_id = db.Column(db.Integer, db.ForeignKey('usuarios.id'))
    usuario = db.relationship('Usuario', backref='chamados')
```

Fonte: Gerada pelo autor

Figura 4 - Comunicação com o Banco de Dados em Nuvem



```python
"""
Arquivo de Configuração da Aplicação

Responsabilidade:
- Centralizar todas as configurações da aplicação Flask.
- Carregar informações sensíveis (como senhas de banco e chaves de API) a partir de variáveis de ambiente
  para não deixá-las expostas no código-fonte (prática de segurança).
- Utiliza a biblioteca `python-dotenv` para carregar um arquivo `.env` localmente durante o desenvolvimento.
"""
import os
import urllib.parse
from dotenv import load_dotenv

# Carrega as variáveis de ambiente definidas no arquivo .env para o ambiente atual.
# Isso permite que `os.getenv()` encontre as variáveis durante o desenvolvimento.
load_dotenv()

class Config:
    """Define as configurações da aplicação em uma classe para melhor organização."""

    # Chave secreta usada pelo Flask para assinar digitalmente os dados da sessão (cookies).
    # É crucial para a segurança contra a manipulação de cookies.
    # O valor é lido do ambiente, com um fallback inseguro apenas para desenvolvimento fácil.
    SECRET_KEY = os.getenv('SECRET_KEY', 'uma-chave-secreta-padrao-super-segura')

    # --- Configuração do Banco de Dados SQL Server ---
    # As credenciais são lidas das variáveis de ambiente para evitar expô-las no código.
    DB_DRIVER = os.getenv('DB_DRIVER', '{ODBC Driver 17 for SQL Server}')
    DB_SERVER = os.getenv('DB_SERVER')
    DB_DATABASE = os.getenv('DB_DATABASE')
    DB_UID = os.getenv('DB_UID')
    DB_PWD = os.getenv('DB_PWD')
```

Fonte: Gerada pelo autor

### 10.1.5 python-dotenv

- Onde: config.py carrega variáveis do .env (load_dotenv()).
- Papel: separar credenciais do código (SECRET_KEY, DATABASE_URL, GEMINI_API_KEY).

Figura 5 - Exemplo config.py que usa python-dotenv

```python
import os
from dotenv import load_dotenv

# Carrega variáveis do arquivo .env (se existir)
load_dotenv()

class DefaultConfig:
    SECRET_KEY = os.getenv('SECRET_KEY', 'dev-secret')
    DEBUG = os.getenv('DEBUG', 'False').lower() in ('1', 'true', 'yes')
    SQLALCHEMY_DATABASE_URI = os.getenv('DATABASE_URL', 'sqlite:///dev.db')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    GEMINI_API_KEY = os.getenv('GEMINI_API_KEY')
```

Fonte: Gerada pelo autor

### 10.1.6 Werkzeug

O Werkzeug é um conjunto de utilitários para aplicações WSGI (Web Server Gateway Interface) em Python, mas, neste contexto, ele é usado primariamente para hashing seguro de senhas.

Para que serve: O propósito principal dessas funções é converter uma senha de texto simples (como 'minhasenha123') em uma string ilegível e de tamanho fixo, chamada hash (por exemplo, pbkdf2:sha256:260000$g4...$3s...).

A função do Hashing: O hash serve para que você possa armazenar e verificar senhas sem nunca precisar salvar a senha real no banco de dados.

Figura 6 - Modelo Usuário com helpers de hash

```python
from datetime import datetime
from .extensions import db
from werkzeug.security import generate_password_hash, check_password_hash

class Usuario(db.Model):
    __tablename__ = 'usuarios'
    id = db.Column(db.Integer, primary_key=True)
    nome = db.Column(db.String(120), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    senha_hash = db.Column(db.String(255), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    def set_senha(self, senha_plain: str):
        # Especificar iterações se quiser controlar o custo: method='pbkdf2:sha256:260000'
        self.senha_hash = generate_password_hash(senha_plain)

    def verificar_senha(self, senha_plain: str) -> bool:
        return check_password_hash(self.senha_hash, senha_plain)
```

Fonte: Gerada pelo autor

No projeto Web, a camada de apresentação foi implementada com HTML5, CSS3 e JavaScript, e faz uso do framework Bootstrap para garantir responsividade, acessibilidade e consistência visual.

A estrutura de templates Jinja2 (templates/) contém o template base (templates/base.html) que define a meta viewport e inclui o CSS/JS do Bootstrap — isso assegura que todas as páginas herdem comportamento responsivo. As views (templates/*) utilizam a grade e utilitários do Bootstrap (container, row, col-*, navbar, card, btn) para layout adaptativo.

O CSS customizado em static/css/styles.css centraliza variáveis via :root e contém média queries para ajustes finos.

O JavaScript em static/js/app.js implementa comportamentos interativos (ex.: toggles, fetch/AJAX) e inicializações do Bootstrap. Com essa combinação, a versão Web adapta-se a diferentes larguras de tela, oferece navegação com navbar

responsiva (navbar-toggler) e componentes que reordenam/ajustam tamanho conforme breakpoint, suportando acessibilidade básica com atributos ARIA e marcação semântica (header, main, footer).

Figura 7 – Template Bootstrap

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1"> <!-- responsividade -->
  <title>{% block title %}UpDesk{% endblock %}</title>
  <!-- Bootstrap CSS (CDN ou arquivo local em static/vendor/bootstrap) -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
  <header><!-- navbar semântico com classes Bootstrap --></header>
  <main class="container">{% block content %}{% endblock %}</main>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
  <script src="{{ url_for('static', filename='js/app.js') }}"></script>
</body>
</html>
```

Fonte: Gerada pelo autor

Figura 8 - Responsividade

```
:root{
  --primary: #0d6efd;
  --bg: #f8f9fa;
}

/* regras base */
body { background: var(--bg); }

/* ajustes responsivos */
@media (min-width: 768px){
  .card-responsive { display: flex; gap: 1rem; }
}
```

Fonte: Gerada pelo autor

Figura 9 - CSS Chamado

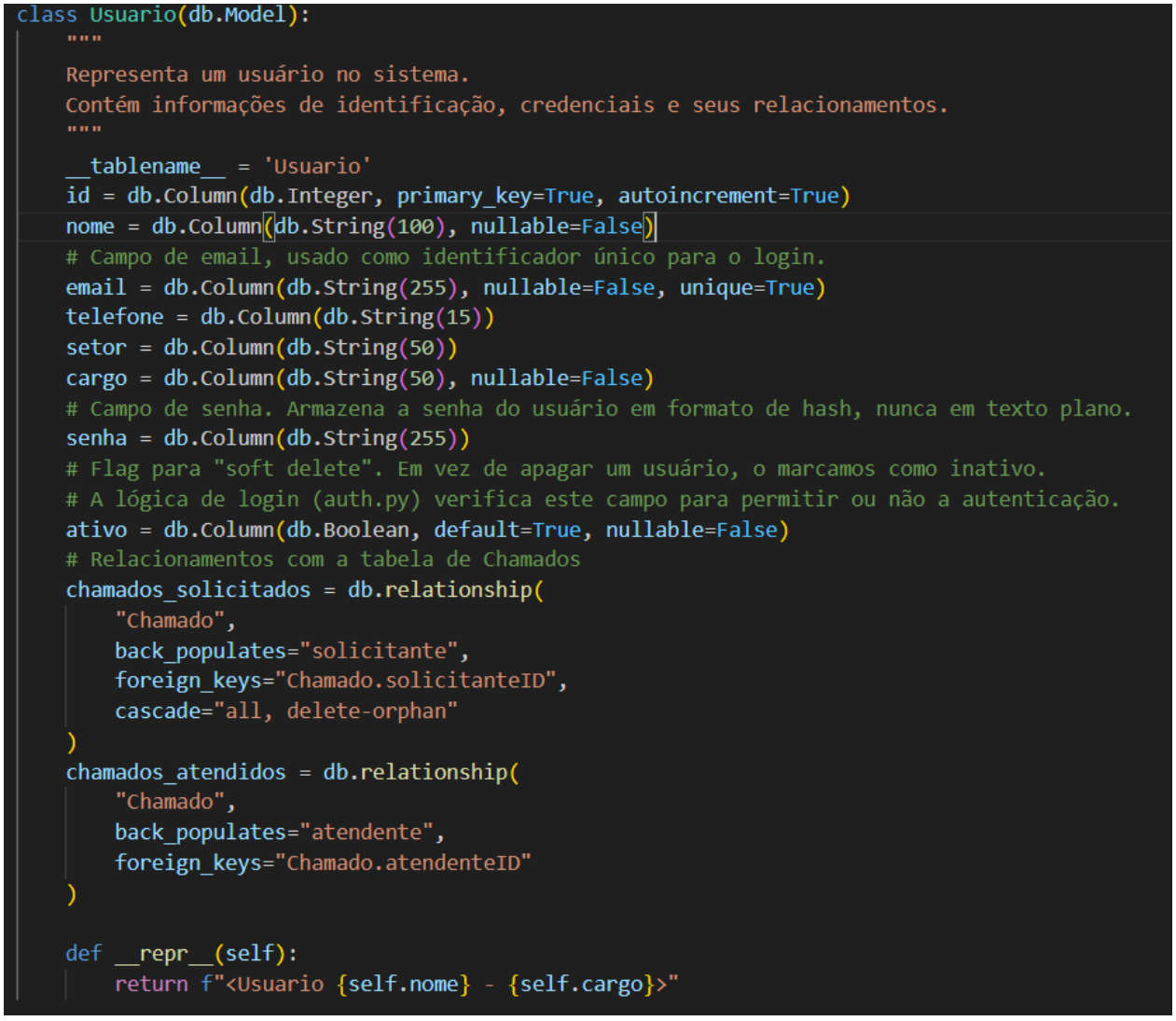
```
{% extends "base.html" %}
{% block content %}
  <div class="row gy-4">
    {% for chamado in chamados %}
      <div class="col-12 col-md-6 col-lg-4">
        <article class="card h-100">
          <div class="card-body">
            <h5 class="card-title">{{ chamado.titulo }}</h5>
            <p class="card-text">{{ chamado.descricao }}</p>
          </div>
        </article>
      </div>
    {% endfor %}
  </div>
{% endblock %}
```

Fonte: Gerada pelo autor

A disciplina de Tópicos Especiais de Programação Orientada a Objetos (TEPOO) contribuiu diretamente para a estruturação e qualidade do sistema UpDesk – Sistema Integrado de Gestão de Chamados e Suporte Técnico com IA. Durante o desenvolvimento, foram aplicados os conceitos estudados em aula, como herança, polimorfismo, abstração, encapsulamento, uso de funções e testes.

No UpDesk, as classes foram utilizadas para representar entidades do sistema, como Usuário, Chamado, Técnico, Cliente e Supervisor. A herança foi aplicada para promover o reuso de atributos e métodos, enquanto o polimorfismo garantiu comportamentos distintos para cada tipo de usuário no tratamento dos chamados.

Figura 10 - Exemplo de herança entre classes de usuários no sistema UpDesk



```python
class Usuario(db.Model):
    """
    Representa um usuário no sistema.
    Contém informações de identificação, credenciais e seus relacionamentos.
    """
    __tablename__ = 'Usuario'
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    nome = db.Column(db.String(100), nullable=False)
    # Campo de email, usado como identificador único para o login.
    email = db.Column(db.String(255), nullable=False, unique=True)
    telefone = db.Column(db.String(15))
    setor = db.Column(db.String(50))
    cargo = db.Column(db.String(50), nullable=False)
    # Campo de senha. Armazena a senha do usuário em formato de hash, nunca em texto plano.
    senha = db.Column(db.String(255))
    # Flag para "soft delete". Em vez de apagar um usuário, o marcamos como inativo.
    # A lógica de login (auth.py) verifica este campo para permitir ou não a autenticação.
    ativo = db.Column(db.Boolean, default=True, nullable=False)
    # Relacionamentos com a tabela de Chamados
    chamados_solicitados = db.relationship(
        "Chamado",
        back_populates="solicitante",
        foreign_keys="Chamado.solicitanteID",
        cascade="all, delete-orphan"
    )
    chamados_atendidos = db.relationship(
        "Chamado",
        back_populates="atendente",
        foreign_keys="Chamado.atendenteID"
    )

    def __repr__(self):
        return f"<Usuario {self.nome} - {self.cargo}>"
```

Fonte: Tirada pelo autor

A abstração e o encapsulamento foram utilizados na classe Usuário, ocultando detalhes internos e expondo apenas métodos essenciais. Essa abordagem facilitou a manutenção e reduziu o acoplamento entre os módulos do sistema.

Figura 11- Aplicação de encapsulamento na classe Usuário

```
public class ApplicationDbContext : DbContext
{
    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

    0 references
    public DbSet<Usuario> Usuarios { get; set; }
    0 references
    public DbSet<Chamado> Chamados { get; set; }
    0 references
    public DbSet<Interacao> Interacoes { get; set; }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        // Configuração para o duplo relacionamento entre Usuario e Chamado
        modelBuilder.Entity<Chamado>()
            .HasOne(c => c.Solicitante)
            .WithMany(u => u.ChamadosSolicitados)
            .HasForeignKey(c => c.SolicitanteId)
            .OnDelete(DeleteBehavior.Restrict); // Evita deleção em cascata

        modelBuilder.Entity<Chamado>()
            .HasOne(c => c.Atendente)
            .WithMany(u => u.ChamadosAtendidos)
            .HasForeignKey(c => c.AtendenteId)
            .OnDelete(DeleteBehavior.Restrict); // Evita deleção em cascata

        // Garante que o email do usuário seja único
        modelBuilder.Entity<Usuario>()
            .HasIndex(u => u.Email)
            .IsUnique();
    }
}
```

Fonte: Tirada pelo autor

O código foi dividido em métodos que executam tarefas específicas, como criar usuário, editar usuário, deletar usuário e atualizar usuário. Isso aumentou a clareza e facilitou a aplicação de testes unitários automatizados com “Pytest”.

Figura 12 - Exemplo de teste unitário automatizado no UpDesk

```python
import pytest
from app import create_app
from app.extensions import db
from app.models import Usuario
from werkzeug.security import generate_password_hash

@pytest.fixture
def client():
    # Cria uma instância do app para teste com configurações específicas
    app = create_app({
        "TESTING": True,
        "SQLALCHEMY_DATABASE_URI": "sqlite:///:memory:",  # banco em memória para teste
        "WTF_CSRF_ENABLED": False # Desabilita CSRF para testes de formulário
    })

    with app.test_client() as client:
        with app.app_context():
            db.create_all()
            # cria um usuário de teste
            usuario = Usuario(
                nome="Mateus",
                email="mateus@teste.com",
                telefone="123456789",
                setor="TI",
                cargo="Dev",
                senha="1234"
            )
            db.session.add(usuario)
            db.session.commit()
        yield client
```

Fonte: Tirada pelo autor

### 10.1.7 Scripts de Dados Iniciais de Teste

Este capítulo apresenta os dados de teste utilizados no sistema UpDesk, simulando um ambiente real para validar funcionalidades como login, abertura de chamados e classificação da IA.

Os scripts de dados iniciais são empregados para popular o banco de dados com informações fictícias. O exemplo inclui a criação de tabelas de usuários, chamados e sugestões da IA, além da inserção de dados fictícios para testes práticos.

10.1.8 Exemplo de Script SQL (SQL Server):

```
CREATE TABLE Usuarios (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Nome NVARCHAR(100),
    Email NVARCHAR(100),
    SenhaHash NVARCHAR(100),
    TipoUsuario NVARCHAR(50)
);

-- Criar tabela de chamados
CREATE TABLE Chamados (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Titulo NVARCHAR(200),
    Descricao NVARCHAR(MAX),
    Status NVARCHAR(50),
    UsuarioId INT,
    DataCriacao DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (UsuarioId) REFERENCES Usuarios(Id)
);

-- Criar tabela de sugestões da IA
CREATE TABLE SolucoesIA (
```

```
    Id INT IDENTITY(1,1) PRIMARY KEY,
    ChamadoId INT,
    Sugestao NVARCHAR(MAX),
    DataSugestao DATETIME DEFAULT GETDATE(),
    IAResponsavel NVARCHAR(100),
    FOREIGN KEY (ChamadoId) REFERENCES Chamados(Id)
);

-- Inserir usuários fictícios
INSERT INTO Usuarios (Nome, Email, SenhaHash, TipoUsuario) VALUES
('João Silva', 'joao@email.com', '123456', 'UsuarioComum'),
('Maria Suporte', 'maria@email.com', 'abc123', 'SuporteN1'),
('Carlos Técnico', 'carlos@email.com', 'senha456', 'SuporteN2'),
('Ana Gestora', 'ana@email.com', 'supervisor1', 'Supervisor');

-- Inserir chamados
INSERT INTO Chamados (Titulo, Descricao, Status, UsuarioId) VALUES
('Erro no login', 'Não consigo acessar o sistema com meu usuário.', 'Aberto', 1),
('Sistema travando', 'O sistema congela após o login.', 'Aberto', 1);

-- Inserir sugestões da IA
INSERT INTO SolucoesIA (ChamadoId, Sugestao, IAResponsavel) VALUES
(1, 'Verifique se o CAPS LOCK está ativado. Caso sim, redefina a senha.', 'IA v1.0'),
(2, 'Limpe o cache do navegador e reinicie a sessão.', 'IA v1.0');
```

Essas práticas resultaram em um sistema robusto, coeso e de fácil manutenção, com arquitetura preparada para integração com futuras versões Web e Mobile. O domínio dos conceitos de TEPOO permitiu aplicar a teoria na prática, fortalecendo a qualidade do software desenvolvido e o aprendizado técnico da equipe.

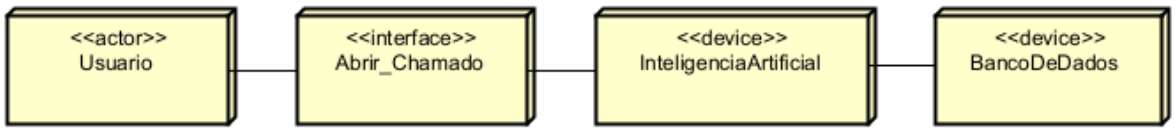
Durante o desenvolvimento, componentes foram reutilizados, agilizando o projeto e facilitando futuras manutenções. Simultaneamente, iniciou-se o levantamento de requisitos da LGPD, garantindo conformidade com a legislação vigente. O projeto UpDesk seguirá boas práticas para garantir a conformidade com a Lei Geral de Proteção de Dados (LGPD – Lei nº 13.709/2018). A coleta de dados será mínima, limitada ao essencial, como nome, e-mail e setor, evitando informações sensíveis sem necessidade legal clara. O sistema adotará consentimento e transparência, exibindo termos de privacidade e informando os usuários sobre o uso e retenção de seus dados. Para garantir segurança, haverá controle de acesso e autenticação, com níveis de permissão e senhas criptografadas. Dados sensíveis serão protegidos por criptografia, e as conexões do sistema utilizarão HTTPS para maior segurança. O projeto também estabelecerá uma política de descarte, determinando períodos de retenção e garantindo que os usuários possam solicitar a exclusão de seus dados. O sistema manterá logs de auditoria, registrando operações sensíveis para prevenir acessos indevidos e facilitar investigações. A equipe de desenvolvimento receberá treinamento sobre privacidade por design, assegurando que os princípios da LGPD sejam aplicados desde o início do projeto. Além disso, será feita a adequação aos papéis de controlador e operador, onde a empresa usuária do sistema será responsável pelos dados (controladora), enquanto a equipe técnica atuará como operadora, cumprindo obrigações específicas previstas na legislação.

Com os casos de uso relacionados à suas classes, se estruturou o diagrama de sequência de cada casos de uso

Este capítulo descreve detalhadamente um exemplo de casos de uso do sistema UpDesk, abordando os atores envolvidos, as pré e pós-condições, e os fluxos normais e alternativos em um diagrama de sequência.

## 10.2 Classe de Implementação: Abrir chamado

*Figura 13: Classe de Implementação*




Fonte: Criado pelo autor

- **Nome**: Abrir chamado
- **Atores:** Usuário, IA
- **Pré-condição:** O usuário deve estar autenticado no sistema.
- **Descrição**: Este caso de uso descreve o processo pelo qual um usuário abre um chamado no sistema para relatar um problema ou necessidade, podendo receber sugestões da IA antes de finalizar a solicitação.
- **Pós-condição**: O chamado é registrado no sistema e pode ser analisado por N1 ou IA.

### 10.2.1 Fluxo Principal (Normal):

1 O usuário acessa o sistema e seleciona a opção “Abrir chamado”.

2 O sistema exibe um formulário para preenchimento das informações do chamado.

3 O usuário preenche os campos obrigatórios, como título, descrição e categoria.

4 O usuário envia o formulário.

5 O sistema aciona a IA, que analisa as informações e sugere uma possível solução.

6 O usuário decide se quer seguir a sugestão ou continuar com a abertura do chamado.

7 O sistema registra o chamado no banco de dados.

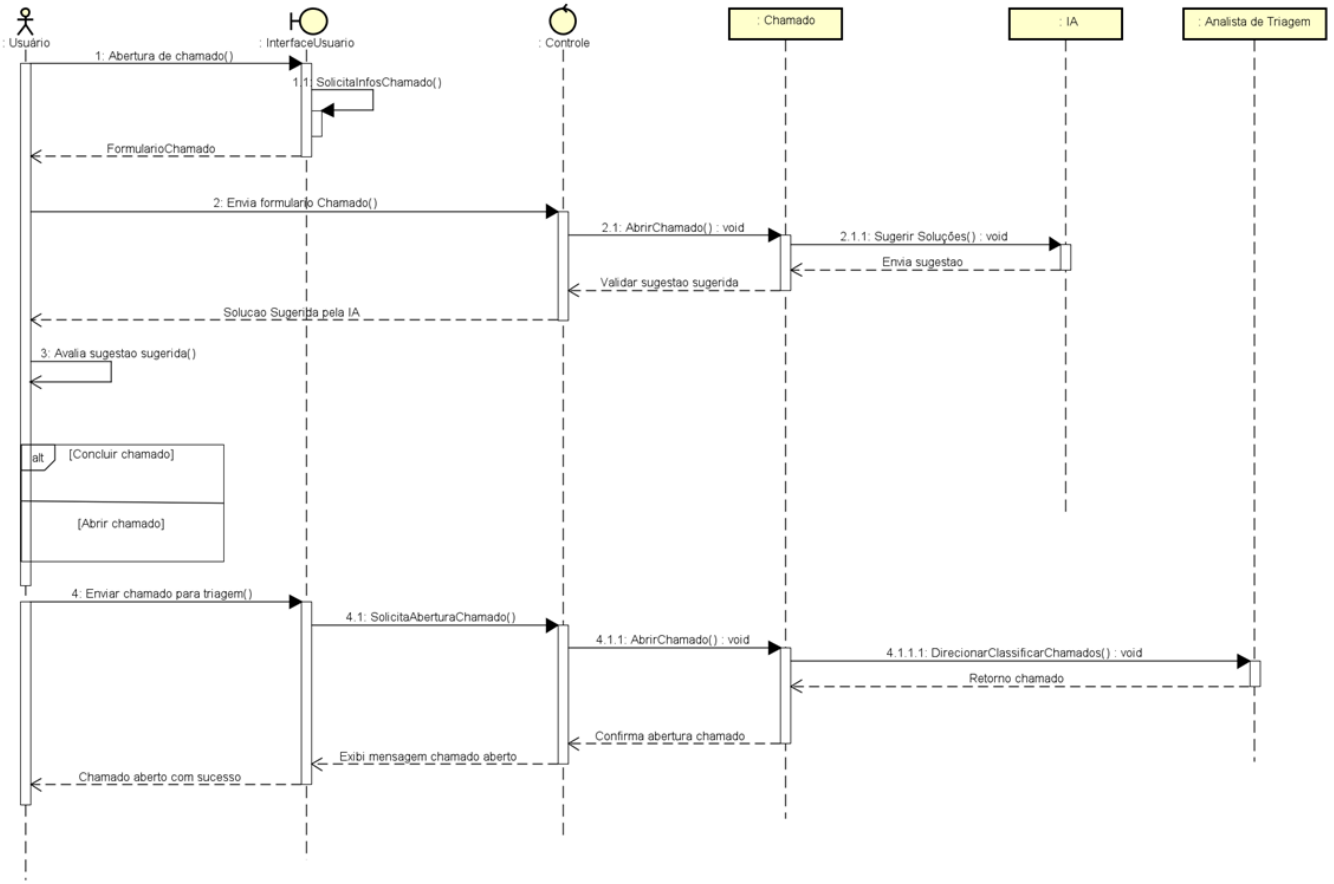
8 O sistema confirma a criação do chamado e exibe o número de protocolo ao usuário.

### 10.2.2 Fluxo de extensão:

1 IA não consegue sugerir solução: O sistema prossegue com o registro normal do chamado sem sugestão.

2 Usuário aceita a sugestão da IA: O chamado é fechado automaticamente com a solução sugerida aplicada.

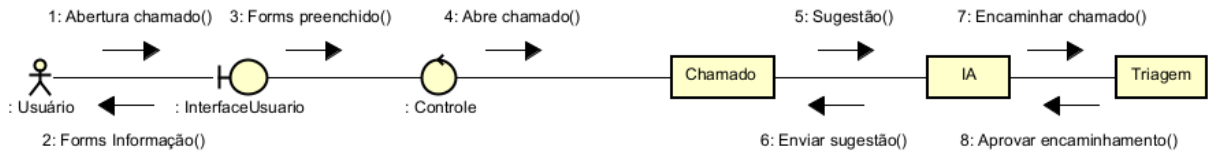
Figura *14*: Diagrama de Sequência - Abertura de chamado




Fonte 1: Criado pelo autor

### 10.2.3 Diagrama de Colaboração:

Figura 15 - Diagrama de Colaboração / Comunicação




Fonte: Gerada pelo autor

A metodologia de projeto adotada seguiu a UML (Unified Modeling Language), conforme as boas práticas apresentadas por Pressman (2016) e Fowler (2011), permitindo padronizar a documentação e facilitar a comunicação entre os membros da equipe.

### 10.2.4 Ciclo de Vida do Projeto

Iniciação: Nesta etapa, foram identificadas as partes interessadas e definida a justificativa do projeto. O problema central consistia na ausência de um sistema centralizado e inteligente de suporte técnico em uma empresa de médio porte, resultando em atrasos e retrabalhos. As partes interessadas incluíram usuários finais, equipe de TI, supervisores e gestores administrativos. O termo de abertura do projeto (TAP) formalizou o início das atividades e delimitou o escopo inicial.

Planejamento: Essa fase envolveu a definição do escopo, cronograma, riscos e comunicação. O escopo contemplou o desenvolvimento de um sistema com IA integrada para triagem e categorização de chamados, suporte multiusuário, geração de relatórios e conformidade com a LGPD. O cronograma foi estruturado em sprints quinzenais, com o uso do Trello e GitHub para controle de tarefas. Também foram identificados riscos como falhas de integração da IA, prazos curtos e possíveis

inconsistências nos testes de usabilidade. As medidas de mitigação incluíram revisões de código em grupo e validações contínuas com a equipe de triagem.

Execução: As atividades de codificação foram realizadas de forma colaborativa, utilizando C#, ASP.NET e SQL Server, de acordo com os diagramas de classes e casos de uso definidos no PIM III. A comunicação entre os membros foi mantida por meio de reuniões semanais via Microsoft Teams e relatórios no GitHub. A equipe seguiu a metodologia Scrum, com reuniões diárias curtas (daily meetings), revisões ao final de cada sprint e retrospectivas para aprimoramento do processo.

Monitoramento e Controle: O controle de progresso foi feito por meio do acompanhamento das tarefas planejadas versus concluídas, utilizando indicadores de desempenho (KPIs) como número de funcionalidades entregues, taxa de sucesso nos testes e percentual de requisitos atendidos. Foram aplicadas métricas de qualidade de software (Pressman, 2016), como densidade de defeitos e tempo médio de resolução de falhas. A gestão de riscos foi continuamente revisada e documentada, garantindo aderência ao cronograma.

Encerramento: Na etapa final, foi elaborada a documentação técnica, incluindo o manual do usuário, o roteiro de testes e o relatório final de desempenho do sistema. A equipe realizou a reunião de lições aprendidas, identificando oportunidades de melhoria, especialmente na comunicação interna e na automatização de testes. O projeto foi oficialmente encerrado após a validação de todas as funcionalidades e a aprovação do orientador.

### 10.2.5 Levantamento de Requisitos

O levantamento de requisitos representou uma etapa essencial do gerenciamento do projeto, pois garantiu que o sistema fosse desenvolvido em conformidade com as necessidades reais dos usuários. Segundo Sommerville (2019), o processo de engenharia de requisitos envolve atividades de elicitação, análise, especificação e validação.

No caso do UpDesk, a equipe adotou técnicas de entrevistas estruturadas e observação direta junto ao setor de TI da empresa parceira para compreender as principais demandas. A partir dessas informações, os requisitos foram classificados em duas categorias:

### 10.2.6 Requisitos Funcionais:

- Abertura, acompanhamento e encerramento de chamados;
- Triagem automática por meio da IA;
- Geração de relatórios de desempenho;
- Controle hierárquico de usuários (supervisor, técnico N1, técnico N2, triagem e usuário comum);
- Chatbot interativo para suporte rápido.

### 10.2.7 Requisitos Não Funcionais:

- Segurança e conformidade com a LGPD;
- Interface responsiva e acessível;
- Desempenho otimizado para múltiplos acessos simultâneos;
- Escalabilidade e integridade dos dados no SQL Server;
- Disponibilidade mínima de 99% no ambiente de testes.

Os requisitos foram documentados em um artefato específico e versionados no repositório GitHub do grupo, permitindo rastreabilidade e controle de mudanças. Cada requisito recebeu um identificador (ID) único, sendo associado a casos de uso, diagramas UML e testes unitários, conforme recomendam as boas práticas de rastreabilidade de requisitos (Dennis, Wixom & Roth, 2014).

### 10.2.8 Benefícios do Gerenciamento de Projetos

A aplicação estruturada do gerenciamento de projetos de software trouxe ganhos significativos ao desenvolvimento do sistema UpDesk. O uso combinado do PMBOK e do Scrum permitiu que a equipe otimizasse o tempo de entrega, reduzisse o retrabalho e mantivesse a motivação e a comunicação entre os integrantes. Além disso, a clara definição dos requisitos e o monitoramento contínuo asseguraram a entrega de um sistema funcional, seguro e aderente às expectativas do cliente.

### 10.2.9 Tempo estimado do projeto

O tempo total estimado para a execução do PIM foi de 8 semanas, com início em agosto e encerramento previsto para a novembro de 2025. Esse período foi subdividido em sprints quinzenais, conforme o cronograma estabelecido no GitHub do projeto. Cada sprint envolveu etapas específicas de desenvolvimento, revisão e testes, seguindo o planejamento de atividades de front-end, back-end e documentação.

### 10.2.10 Tempo foi distribuído do projeto:

- **Sprint 1** - Planejamento do backlog, definição de sprints
- **Sprint 2** - Modelagem dos diagramas (Classe de Implementação, Sequência, Colaboração)
- **Sprint 3** - Finalização do CRUD do banco de dados em SQL Server
- **Sprint 4** - Início do desenvolvimento web (estrutura, layout, rotas principais)

- **Sprint 5** - Implementação das funcionalidades principais e integração com IA
- **Sprint 6** - Início do desenvolvimento desktop e mobile (estrutura inicial e navegação)
- **Sprint 7** - Finalização dos sistemas web, mobile e desktop, testes unitários e ajustes na interface
- **Sprint 8** - Documentação técnica, revisão geral, correções finais e entrega do projeto

O esforço total estimado para a equipe foi de aproximadamente 240 horas, considerando seis integrantes com dedicação média de 4 horas semanais por membro, divididas entre desenvolvimento, testes, reuniões e documentação.

### 10.2.11 Sistema de comunicação da equipe

A comunicação entre os integrantes foi estruturada de maneira híbrida, combinando reuniões semanais via Microsoft Teams com interações diárias pelo grupo do WhatsApp. Essa organização garantiu agilidade nas decisões e fácil acompanhamento das tarefas em andamento.

Além disso, o Trello foi utilizado como ferramenta de controle visual do fluxo de trabalho, enquanto o GitHub serviu para versionamento do código, registro de issues e documentação de progresso.

### 10.2.12 A frequência de contato:

Reuniões de planejamento (Sprint Planning): Semanais, para definição das metas de cada sprint.

Reuniões rápidas (Daily meetings): Diárias via WhatsApp para acompanhamento de status.

Revisões e retrospectivas: Ao final de cada sprint, para análise do progresso e ajustes no processo.

Reuniões de validação: Quinzenais com o orientador, para avaliação dos avanços e alinhamento acadêmico.

### 10.2.13 Cronograma e cálculo estimado do projeto

Com base nas informações do Git e dos relatórios de sprint, foi elaborado um cronograma estimado de execução:

**Etapa / Sprint | Atividades Principais | Duração | Responsáveis**

- **Sprint 1** - Planejamento do backlog, definição de sprints - 1 semana - Toda a equipe
- **Sprint 2** - Protótipo de baixa fidelidade e modelagem dos diagramas (Classe de Implementação, Sequência, Colaboração) - 1 semana - Equipe de análise e back-end
- **Sprint 3** - Finalização do CRUD no banco de dados (SQL Server) - 1 semana - Equipe de back-end
- **Sprint** 4 - Início do desenvolvimento web (estrutura, layout, rotas principais) - 1 semana - Equipe de front-end
- **Sprint 5** - Implementação das funcionalidades principais e integração com IA - 1 semana - Equipe de desenvolvimento geral
- **Sprint 6** - Início do desenvolvimento desktop e mobile - 1 semana - Equipe de front-end e mobile
- **Sprint 7** - Finalização dos sistemas web, mobile e desktop; testes unitários e ajustes - 1 semana - Todos os membros

- **Sprint 8** - Documentação técnica, revisão geral e entrega do projeto - 1 semana - Líder e equipe geral

O cálculo de tempo e esforço foi realizado com base no método Three-Point Estimation (PERT), adotando-se a média ponderada entre cenários otimista, provável e pessimista para cada sprint. Assim, o tempo total do projeto foi estimado em 9,5 semanas, com uma margem de segurança de 5% para imprevistos técnicos.

### 10.2.14 Aplicação dos conceitos da disciplina

Durante todo o desenvolvimento, os conceitos da disciplina foram aplicados de forma prática, especialmente nas áreas de planejamento, controle de escopo, gestão de riscos e monitoramento de desempenho. A divisão das atividades seguiu o ciclo de vida do projeto conforme o PMBOK, garantindo o acompanhamento de entregas e a rastreabilidade de requisitos.

### 10.2.15 As métricas de desempenho (KPIs) incluíram:

- Percentual de requisitos concluídos por sprint;
- Tempo médio de resolução de falhas;
- Taxa de sucesso nos testes automatizados;
- Grau de aderência ao cronograma.

Esses indicadores possibilitaram uma visão clara do andamento do projeto, permitindo intervenções corretivas quando necessário e assegurando o cumprimento dos prazos definidos.

O desenvolvimento do sistema UpDesk – Sistema Integrado de Gestão de Chamados e Suporte Técnico com Inteligência Artificial (IA) foi conduzido com base em princípios de empreendedorismo inovador e tecnológico, evidenciando a

capacidade da equipe de transformar uma ideia em um produto funcional e escalável. Segundo Dornelas (2018), o empreendedor é aquele que identifica oportunidades e as transforma em negócios viáveis, agregando valor e promovendo inovação.

Durante o projeto, a equipe atuou de forma empreendedora ao reconhecer a necessidade de modernizar os processos de suporte técnico nas empresas, substituindo métodos manuais, como e-mails e telefonemas, por uma solução automatizada e integrada. Essa visão de oportunidade e inovação, conforme destaca Chiavenato (2020), é característica fundamental do comportamento empreendedor e foi essencial para orientar o desenvolvimento do UpDesk.

O projeto aplicou práticas de planejamento estratégico e gestão de riscos, alinhadas às diretrizes de um plano de negócios. Foram definidos missão, visão, público-alvo, diferenciais competitivos e estratégias de sustentabilidade, conforme proposto por Dolabela (2008). A utilização da metodologia Scrum reforçou a mentalidade empreendedora ao estimular a autogestão, a liderança compartilhada e a entrega de valor contínuo ao cliente.

A inovação tecnológica, conforme Drucker (2014), é o principal instrumento do empreendedor, e esteve presente na integração da IA para sugerir soluções automáticas e otimizar o fluxo de chamados. Essa abordagem gerou vantagem competitiva e alinhou o produto às tendências da Indústria 4.0. Além disso, a preocupação com a sustentabilidade e escalabilidade do modelo de negócio garantiu que o sistema pudesse crescer de forma viável e manter qualidade a longo prazo (HISRICH; PETERS; SHEPHERD, 2014).

Em síntese, o projeto UpDesk materializou o espírito empreendedor ao unir visão inovadora, planejamento estratégico, gestão de riscos e foco no cliente. A equipe demonstrou iniciativa, criatividade e capacidade de execução — competências essenciais para empreendedores de sucesso. Dessa forma, o projeto ultrapassou o

simples desenvolvimento técnico, tornando-se um exemplo prático de empreendedorismo aplicado à tecnologia e à gestão de negócios.

# 11 MANUAL DO USUÁRIO – SISTEMA UPDESK

### 11.1.1 Introdução

Este manual tem como objetivo orientar o usuário quanto à utilização do sistema UpDesk, com base no protótipo de telas desenvolvido. O sistema foi projetado para ser intuitivo, com foco em facilitar a experiência do usuário final.

### 11.1.2 2. Acesso ao Sistema

#### 2.1 Tela de Login

**Descrição:** O usuário deve informar seu e-mail e senha cadastrados para acessar o sistema.

**Ações disponíveis:**

- **Entrar no sistema:** Após preencher os campos de e-mail e senha, clique em "Entrar" para acessar.
- **Recuperar senha:** Caso tenha esquecido a senha, clique em "Recuperar senha" e siga as instruções para redefini-la.

Figura *16*: Tela de Login - upDesk


Fonte 2: Criado pelo autor

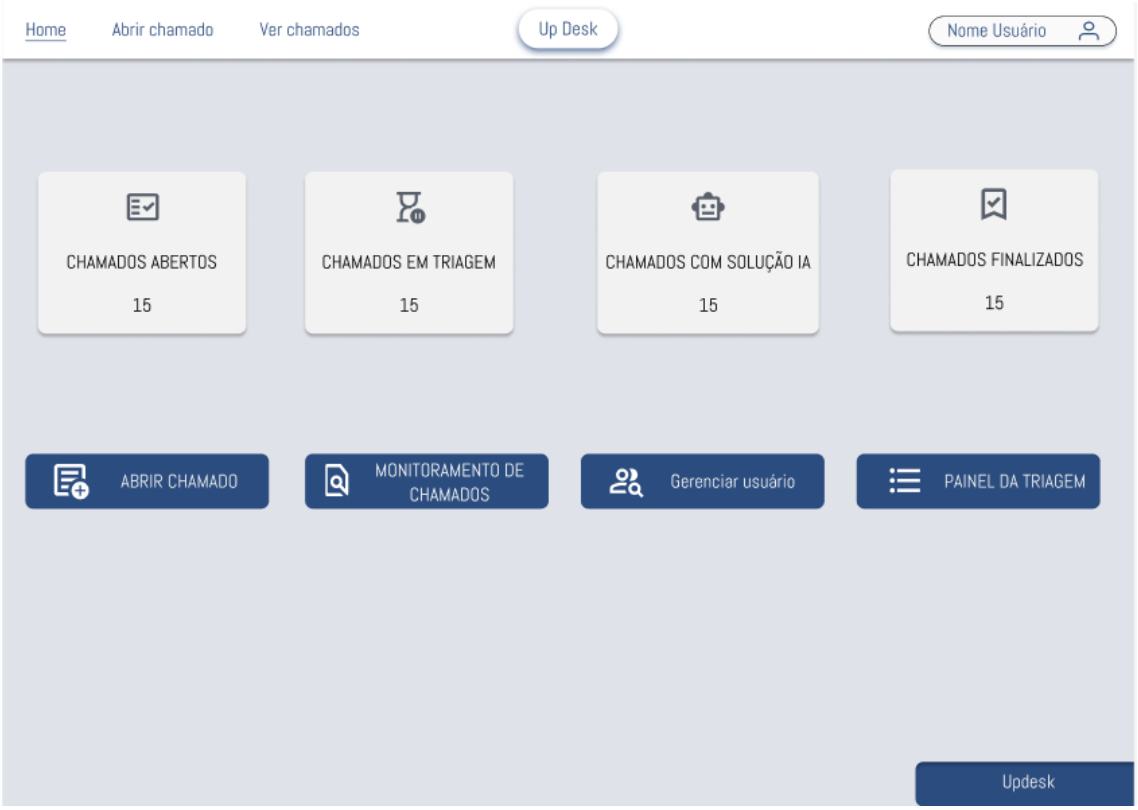
### 11.1.3 3. Tela Inicial – Home

**Descrição:** Após fazer login, o usuário será direcionado para o painel principal do UpDesk.

**Funcionalidades principais:**

- **Abertura de novo chamado**: Permite registrar solicitações de suporte técnico.
- **Visualização de chamados**: Exibe chamados abertos, em andamento e encerrados para acompanhamento.
- **Acesso ao perfil do usuário**: Disponibiliza configurações e informações do usuário.
- **Chat Bot do UpDesk**: Assistente virtual para suporte rápido e direcionamento de chamados.

Figura *17*: Home




Fonte 3: Criado pelo autor

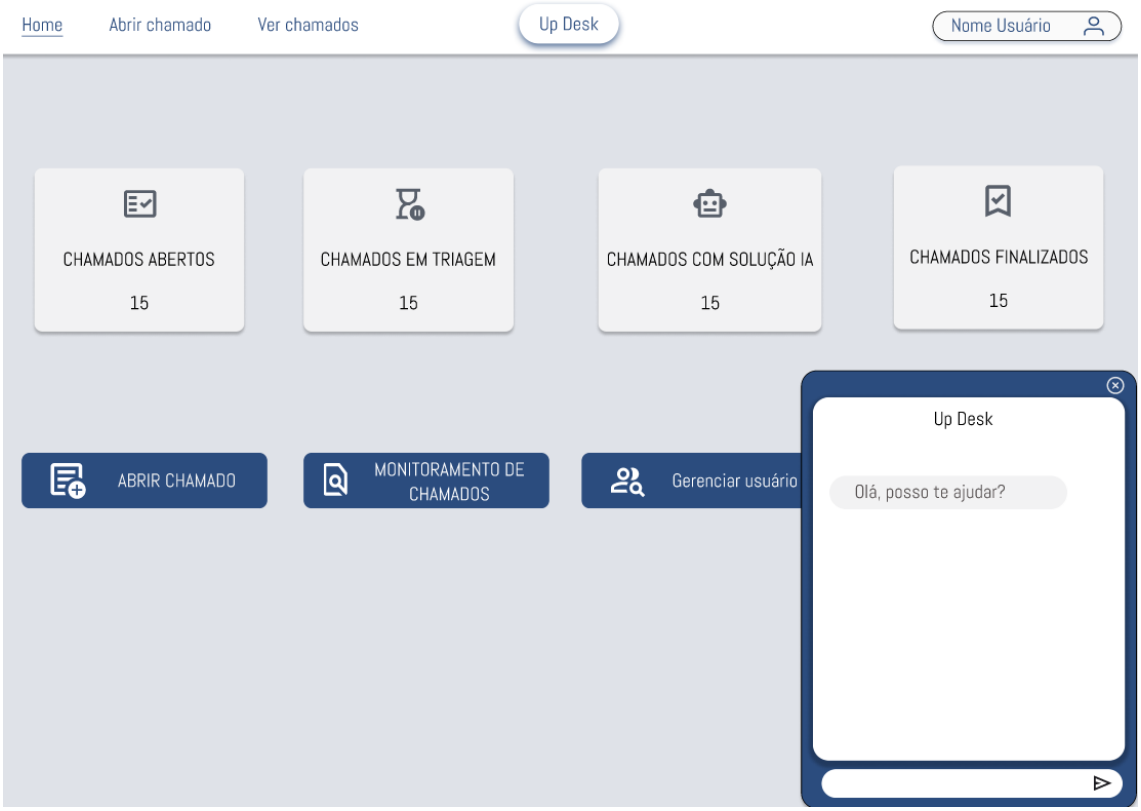
### 11.1.4 4. Chat Bot Home

A tela de Chat Bot do UpDesk permite que os usuários interajam com um assistente virtual para suporte rápido e eficiente.

**Funcionalidades Principais**

- **Envio de Mensagens**: O usuário pode conversar com o chatbot para obter ajuda.
- **Abertura de Chamados**: O bot pode sugerir a criação de um chamado com base na conversa.
- **Histórico de Conversas**: As interações ficam registradas para referência futura.
- **Direcionamento Inteligente**: Chamados são enviados automaticamente para a equipe adequada.

Figura *18*: ChatBot - Home




Fonte 4: Criado pelo autor

### 11.1.5 5. Tela de Perfil do Usuário

**Descrição:** Essa tela exibe as informações do perfil do usuário no sistema UpDesk, permitindo visualizar seus dados pessoais e seu nível de acesso no sistema.

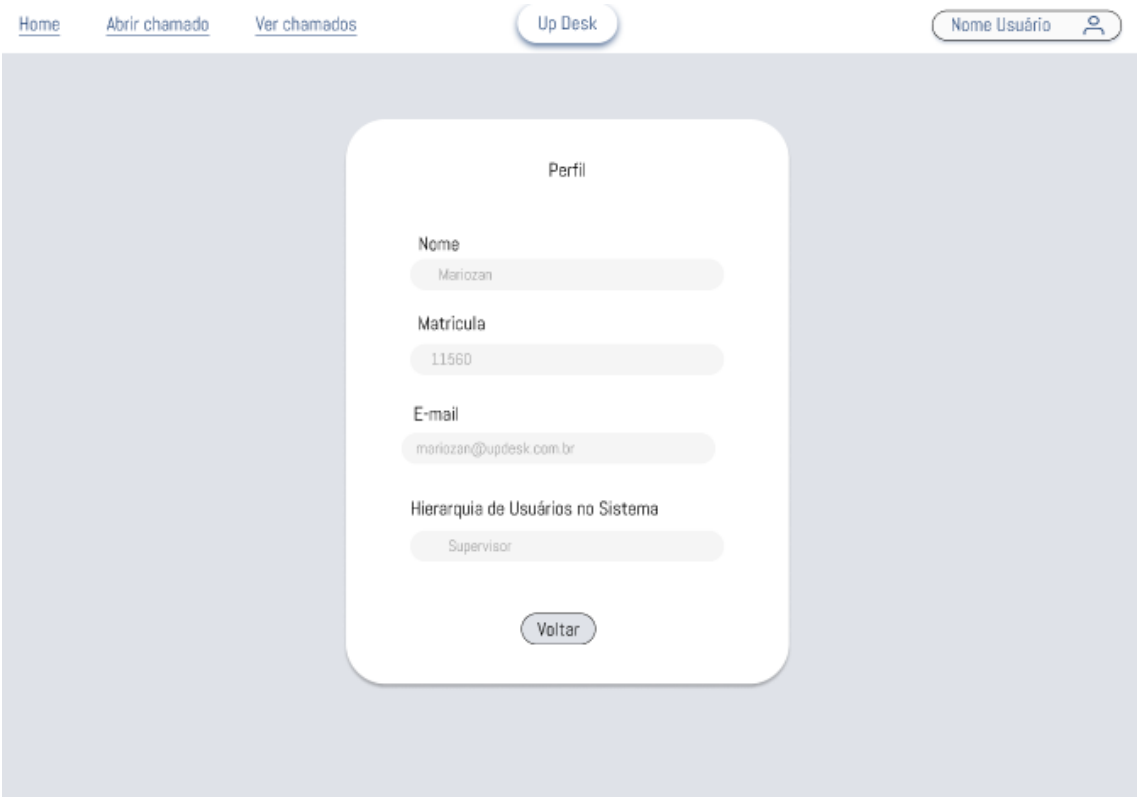
**Funcionalidades principais:**

- Nome do usuário: Exibe o nome cadastrado no sistema.
- Matrícula: Mostra o número de registro do usuário.
- E-mail: Indica o e-mail associado ao perfil.
- Hierarquia de Usuário: Exibe o nível de permissão dentro do sistema (Supervisor, por exemplo).

**Ações disponíveis:**

- Voltar: Retorna à tela anterior.
- Navegação: Acesso ao menu superior com opções como Home, Abrir chamado, Ver chamados e Nome do Usuário.

Figura *19*: Perfil Usuário




Fonte 5: Criado pelo autor

### 11.1.6 6. Abrir um Chamado

**Descrição:** Permite que o usuário registre um novo chamado técnico para suporte.

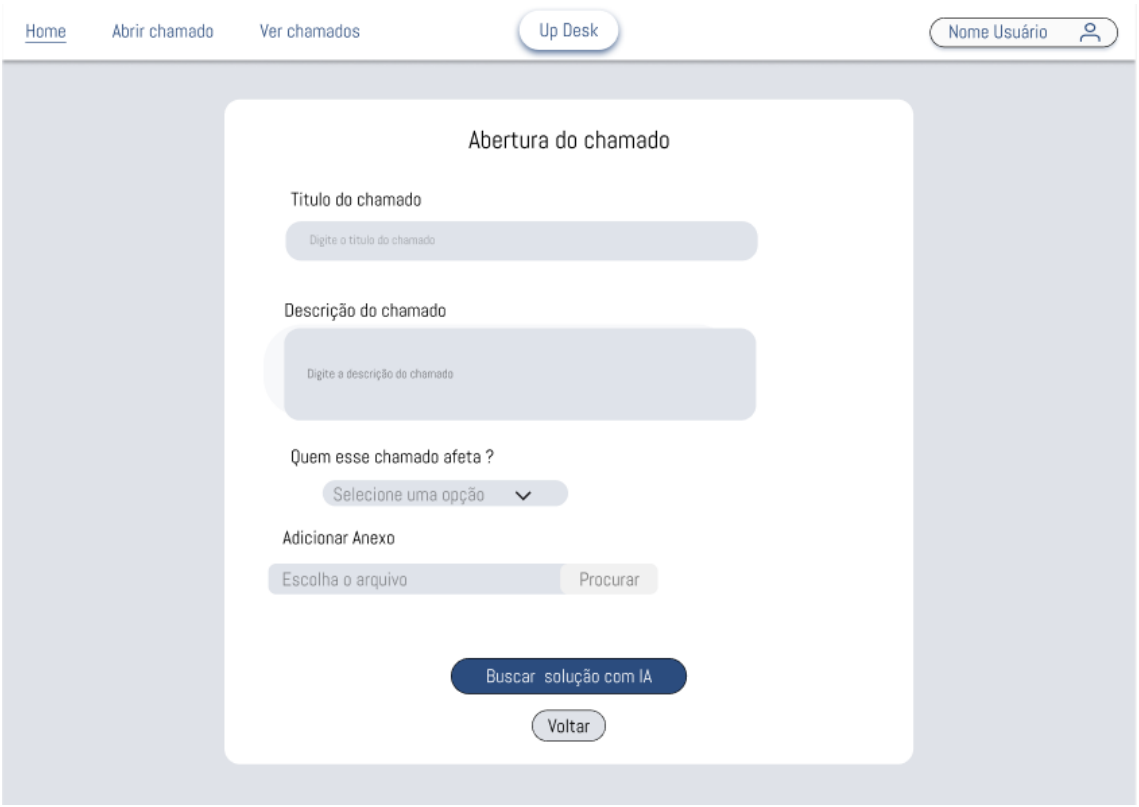
Passos para abertura do chamado:

Acesse a opção "Abrir Chamado" na tela principal.

- **Preencha os campos obrigatórios:**
- **Título do chamado**: Breve descrição do problema.

- **Categoria**: Selecione entre TI, rede, software etc.
- **Descrição**: Detalhamento do problema.
- **Quem esse chamado afeta**: Indique os usuários impactados.
- **Adicionar anexo**: Caso necessário, envie arquivos relacionados.

Figura *20*: Formulário Chamado




Fonte 6: Criado pelo autor

### 11.1.7 7. Tela de Solução Sugerida

**Descrição:** A tela apresenta sugestões de solução para problemas relatados, permitindo ao usuário decidir entre finalizar ou abrir um novo chamado.

**Funcionalidades principais:**

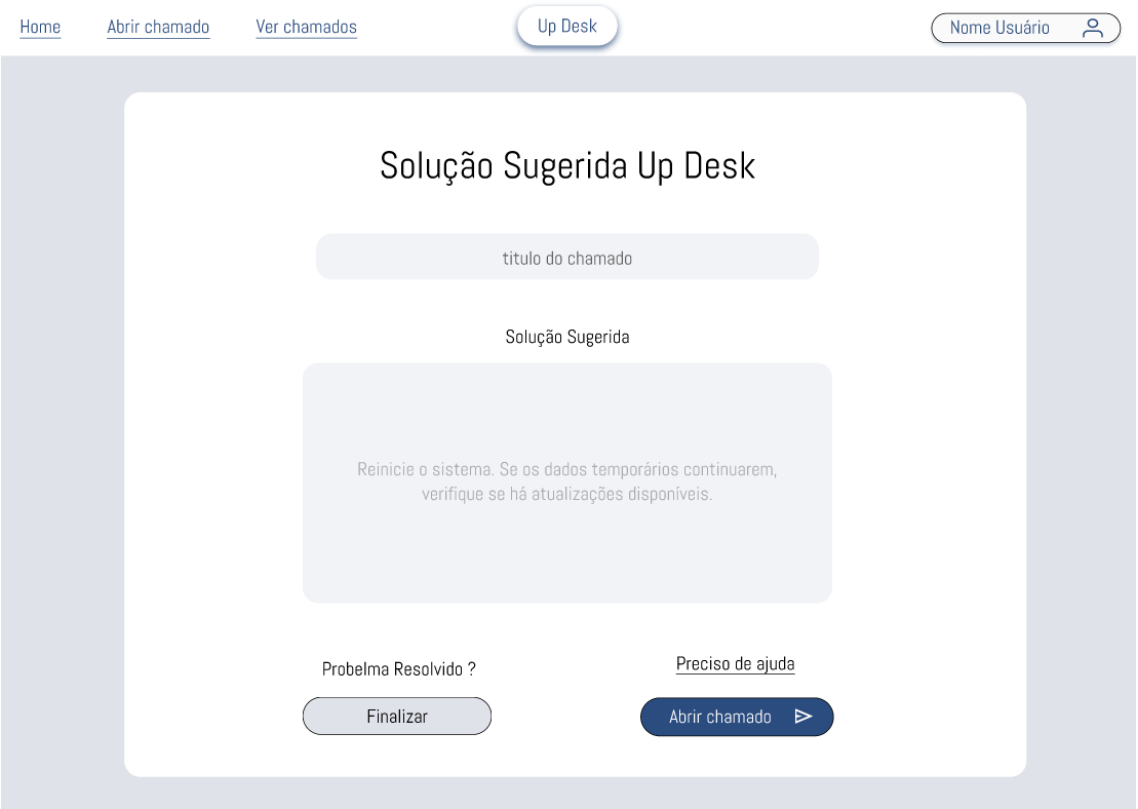
- **Sugestão de Solução**: Exibe instruções automatizadas para resolver possíveis problemas.
- **Título do chamado**: Define o problema.

- **Solução Sugerida**: Instruções recomendadas, como reiniciar o sistema ou verificar atualizações.

**Ações disponíveis:**

- **Finalizar:** Indica que o problema foi resolvido.
- **Abrir chamado:** Permite solicitar suporte adicional e iniciar um chamado para a equipe técnica.

Figura *21*: Sugestão IA




Fonte 7: Criado pelo autor

### 11.1.8 Tela de Chamado Resolvido

**Descrição:** Essa tela informa ao usuário que o chamado foi solucionado com sucesso pela inteligência artificial UPDESK e oferece opções para navegar no sistema.

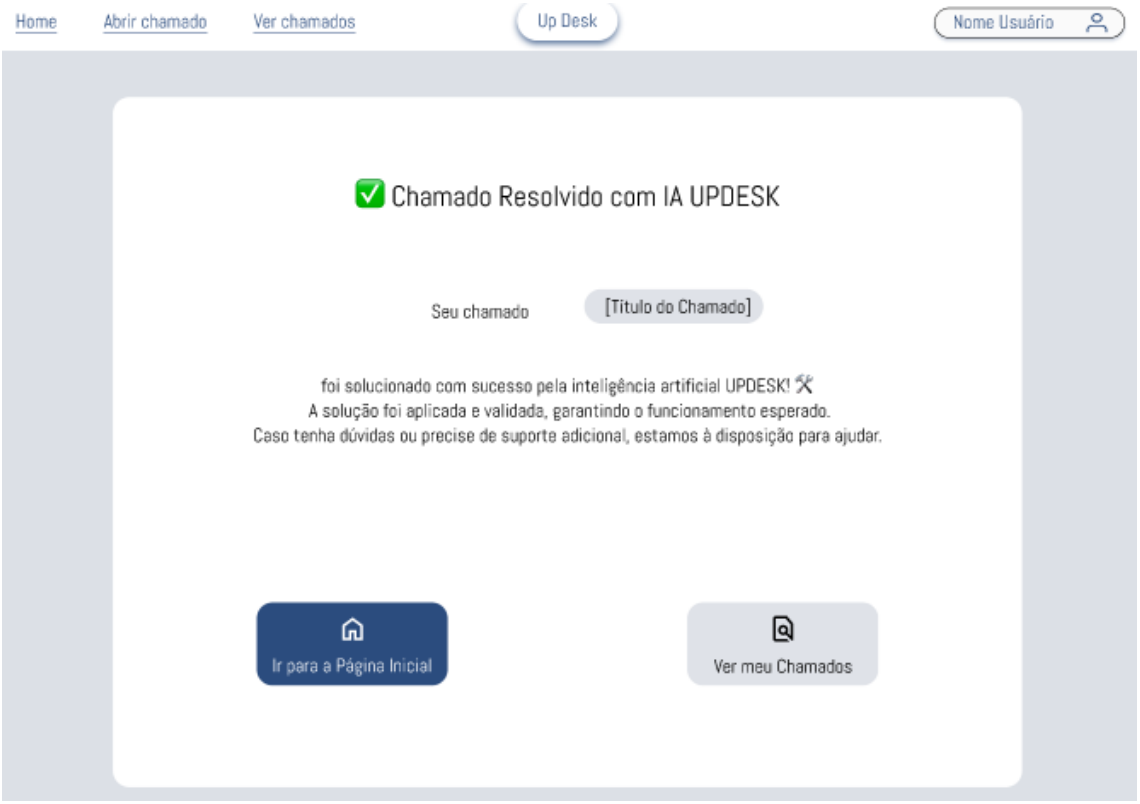
**Funcionalidades principais:**

- **Mensagem de confirmação:** Indica que o chamado foi resolvido pela IA.
- **Título do chamado:** Exibe o nome ou descrição da solicitação resolvida.
- **Validação da solução:** Confirma que a solução aplicada garante o funcionamento esperado.
- **Suporte adicional:** Caso necessário, o usuário pode buscar mais ajuda.

**Ações disponíveis:**

- **Ir para a Página Inicial:** Retorna ao painel principal do sistema.
- **Ver meus Chamados:** Acessa a lista de chamados abertos ou resolvidos.

Figura 22: Chamado resolvido




Fonte 8: Criado pelo autor

### 11.1.9 9. Tela de Confirmação de Chamado

**Descrição:** Após o envio de um chamado, esta tela informa ao usuário que a solicitação foi registrada com sucesso e fornece opções de navegação para o acompanhamento do chamado.

**Funcionalidades principais:**

- **Mensagem de confirmação:** Indica que o chamado foi enviado com sucesso.
- **Título do chamado:** Exibe o nome ou descrição do chamado registrado.
- **Setor responsável:** Informa qual equipe receberá e analisará a solicitação.
- **Urgência do chamado:** Mostra o nível de prioridade atribuído ao chamado.

**Ações disponíveis:**

- **Ir para a Página Inicial:** Retorna ao painel principal do sistema.
- **Ver meus Chamados:** Acessa a lista de chamados abertos para acompanhamento.

Figura *23*: Chamado enviado


Fonte 9: Criado pelo autor

### 11.1.10 Monitoramento de usuários

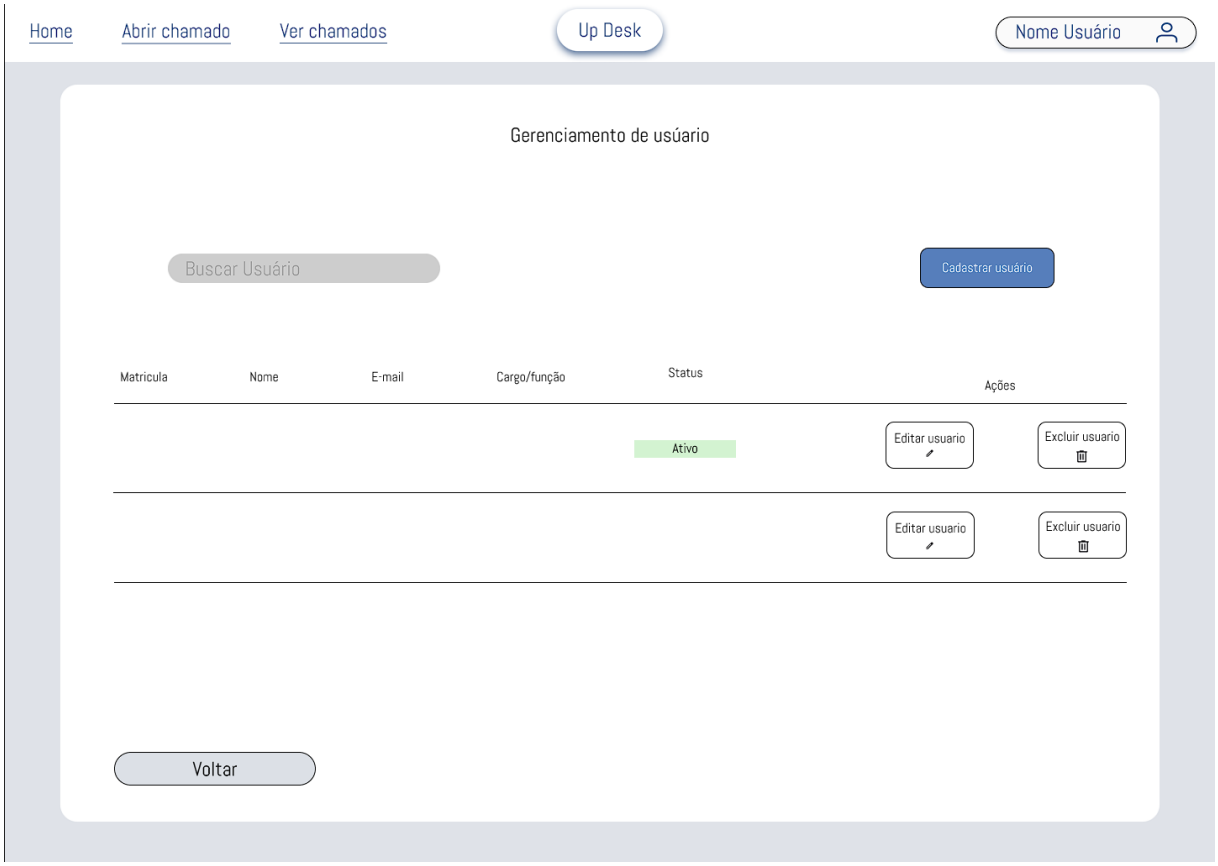
**Descrição:** Permite visualizar os usuários do sistema.

**Funcionalidades principais:**

- **Buscar usuários:** Filtra os usuários exibidos de acordo com as informações inseridas no campo de busca
- **Cadastrar usuário:** Cadastra um novo usuário, solicitando as informações necessárias.
- **Editar usuário**: Permite alterar informações salvas referente ao usuário.

- **Inativar usuário:** Inativa o usuário, o impossibilitando a acessar o sistema. Por segurança, é solicitado a senha de acesso do administrador para validar a ação.

Figura *24*: Monitoramento de usuários




Fonte 10: Criado pelo autor

### 11.1.11 Consultar Chamados

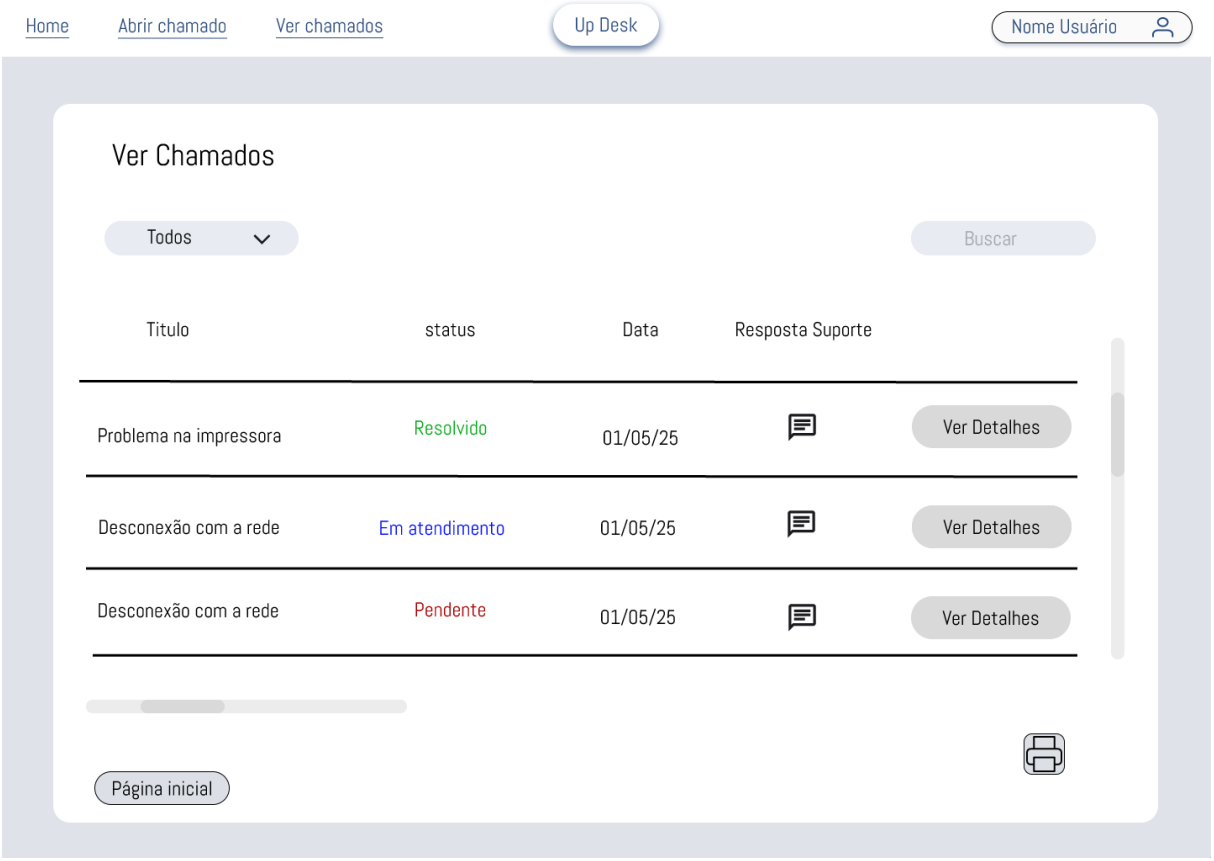
**Descrição:** Lista com os chamados abertos, em andamento ou resolvidos.

**Ações permitidas:**

1. Visualizar detalhes.
2. Acompanhar status.
3. Filtrar por data ou categoria.

4. Buscar chamado
5. Imprimir relatório

Figura *25*: Histórico de chamados




Fonte 11: Criado pelo autor

### 11.1.12 Aprendizado da IA

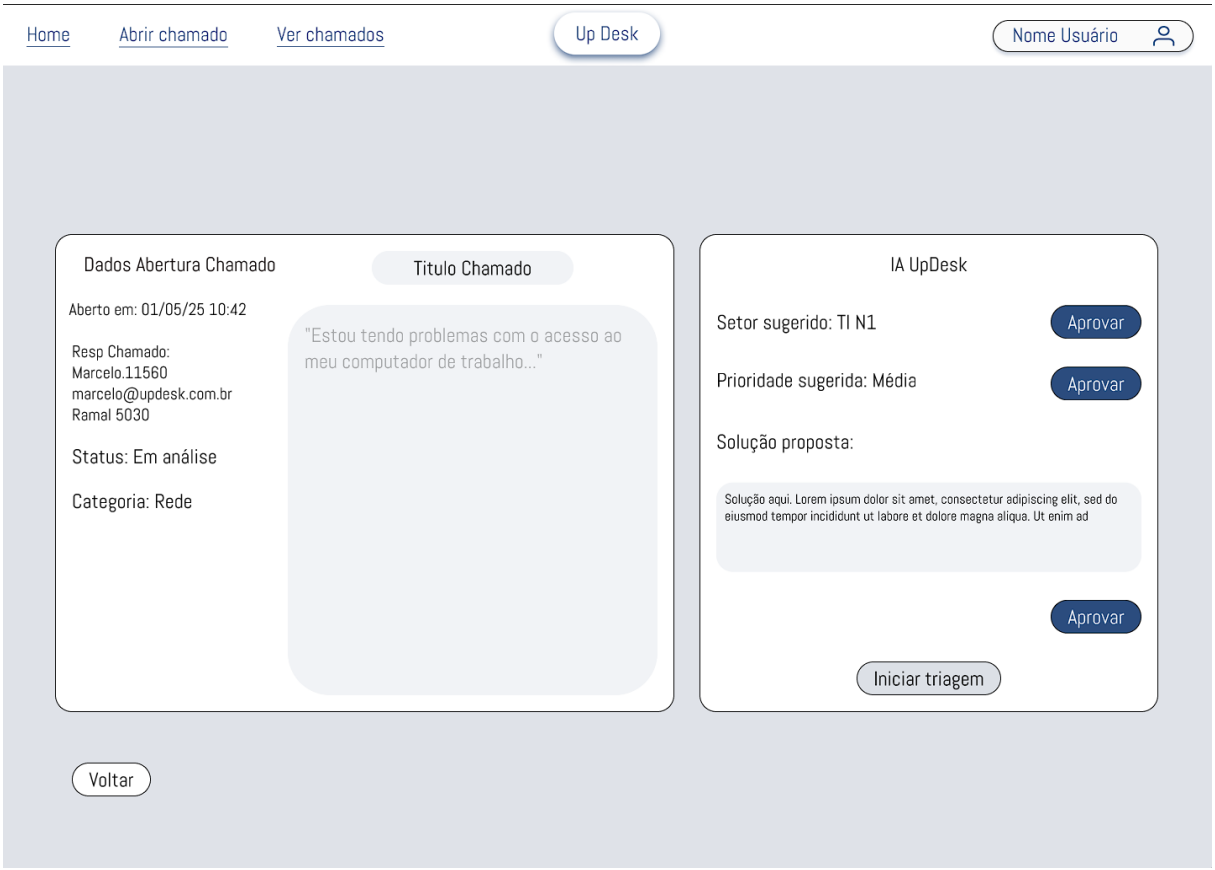
**Descrição:** O dispatcher irá validar as atitudes da IA, caso aprovado, a IA aprende que aquela abordagem fora correta e assim iniciar a triagem.

**Funcionalidades:**

- **Visualização de detalhes:** Apresenta os detalhes ao lado para auxiliar o dispatcher.

- **Aprovação das ações da IA:** Aprovando tais atitudes, a IA os utilizará mais vezes.
- **Iniciar triagem:** Procede o fluxo, seguindo para a triagem.

Figura *26*: Validação da IA

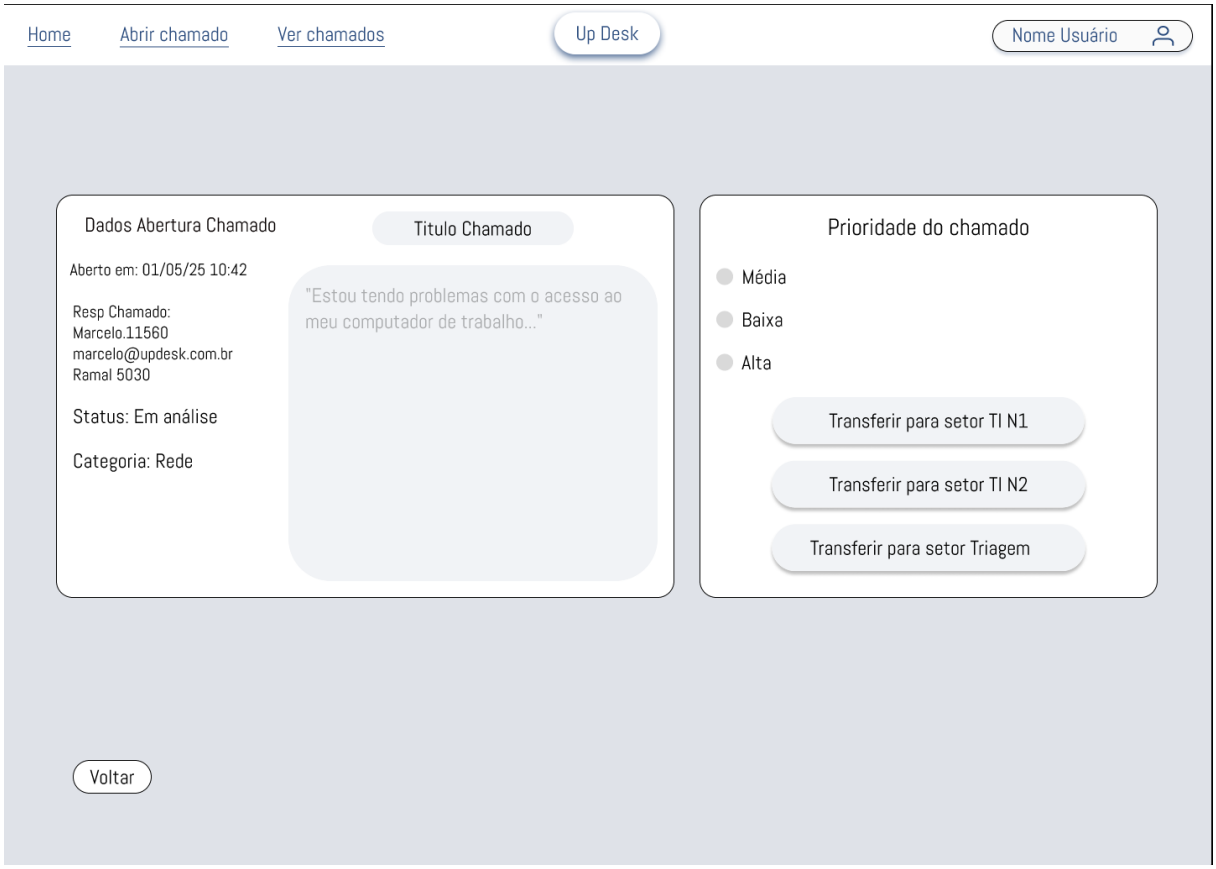



Fonte 12: Criado pelo autor

### 11.1.13 Triagem do chamado

**Descrição:** Interface para a classificação e direcionamento do chamado.

Figura 27: Interface para triagem




Fonte 13: Criado pelo autor

### 11.1.14 Considerações Finais – Manual do Usuário

Este manual será atualizado conforme novas funcionalidades forem implementadas. Recomendamos que os usuários estejam sempre atentos às notificações dentro do sistema.

# 12 CONCLUSÃO

O desenvolvimento do sistema UpDesk representou uma oportunidade concreta de aplicar os conhecimentos adquiridos ao longo do curso de Análise e Desenvolvimento de Sistemas. Com base no levantamento de requisitos realizado no semestre passado e no protótipo previamente elaborado no Figma, foi possível avançar para a implementação completa da solução, contemplando suas versões web, desktop e mobile. A proposta de integrar funcionalidades tradicionais de gerenciamento de chamados com recursos de inteligência artificial ampliou o escopo técnico do projeto e demonstrou a viabilidade de soluções inteligentes no contexto de atendimento e suporte técnico.

Durante a execução do projeto foram realizadas atividades como a modelagem do banco de dados, a elaboração da classe de implementação, além dos diagramas de sequência e colaboração, que serviram como base para estruturar o comportamento e a interação entre os componentes do sistema. Essas etapas foram fundamentais para transformar o protótipo inicial em um sistema funcional, consistente e alinhado às necessidades do usuário final.

Além dos ganhos técnicos, o projeto contribuiu significativamente para a formação prática dos envolvidos, fortalecendo a compreensão sobre o ciclo de vida do desenvolvimento de software, gestão de requisitos, modelagem UML e boas práticas em projetos de extensão universitária. A aplicação da inteligência artificial, ainda que em estágio inicial, evidenciou o potencial de automação e apoio à decisão nos processos organizacionais.

Conclui-se, portanto, que o projeto UpDesk atingiu seus objetivos pedagógicos e técnicos, consolidando-se como uma solução viável, escalável e adequada às demandas atuais de empresas que buscam maior agilidade e eficiência em seus processos de atendimento.

## REFERÊNCIAS


AGOSTINI, Marcelo N.; DECKER, Ildemar C.; SILVA, Aguinaldo S. e. Desenvolvimento e implementação de uma base computacional orientada a objetos para aplicações em sistemas de energia elétrica. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, v. 13, n. 2, p. 192-200, ago. 2002. DOI: 10.1590/S0103-17592002000200010.

BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML. 3. ed. Rio de Janeiro: LTC, 2014.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML – Guia do Usuário. 2. ed. Rio de Janeiro: Elsevier, 2006.

DENNIS, Alan; WIXOM, Barbara H.; ROTH, Roberta M. Análise e Projeto de Sistemas. Grupo GEN, 2014.

DENNIS, Alan; WIXOM, Barbara Haley; ROTH, Roberta M. Análise e Projeto de Sistemas. 5. ed. Rio de Janeiro: LTC, 2014.

FOWLER, Martin. UML Essencial: Um Breve Guia para a Linguagem Padrão de Modelagem de Objetos. 3. ed. Porto Alegre: Bookman, 2011.

GAMMA, Erich et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

IELSEN, Jakob. Usability engineering. San Francisco: Morgan Kaufmann, 2012.

ISHIKAWA, Kaoru. Controle de Qualidade Total: à maneira japonesa. 4. ed. São Paulo: IMC, 1993.

JURAN, J. M.; GODFREY, A. B. Juran's Quality Handbook. 6. ed. McGraw-Hill, 2010.

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.

Lucidchart. *O que é um diagrama de implementação UML?*. Lucid Software Inc., 2024.

Miro. *O que é diagrama UML e como fazer? Veja tipos, modelos e exemplos*. Miro Inc., 2024.gineering. San Francisco: Morgan Kaufmann, 1993

OWASP Foundation. OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks. 2021. Disponível em: https://owasp.org/www-project-top-ten/

PRESSMAN, Roger S. Engenharia de Software: uma abordagem profissional. 8. ed. McGraw-Hill, 2016.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

RAUT, Rushikesh S. Research Paper on Object-Oriented Programming (OOP). **International Research Journal of Engineering and Technology (IRJET)**, v. 7, n. 10, p. 1452, Oct. 2020. Disponível em: https://www.irjet.net/archives/V7/i10/IRJET-V7I10247.pdf. Acesso em: 11 nov. 2025.

RAUT, Rushikesh S. Research Paper on Object-Oriented Programming (OOP). **International Research Journal of Engineering and Technology (IRJET)**, v. 7, n. 10, p. 1452, Oct. 2020. Disponível em: https://www.irjet.net/archives/V7/i10/IRJET-V7I10247.pdf. Acesso em: 11 nov. 2025.

Silveira, Samara. *UML em projetos orientados a objetos*. Casa do Desenvolvedor, 23/09/2024.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. Pearson, 2019. PROJECT MANAGEMENT INSTITUTE (PMI). A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 7ª ed., 2021.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2019.

SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson, 2011.

WAZLAWICK, Raul Sidnei. Análise e Design Orientados a Objetos para Sistemas de Informação: Modelagem com UML, OCL e IFML. Rio de Janeiro: Elsevier, 2014.

## INDICE DE FIGURAS

FICHA DE CONTROLE DO PIM

Ano 2025, Período: 4° Semestre, Orientador: FRANCISCO JOSÉ MARTINS

Tema: Sistema integrado para gestão de chamados e suporte técnico baseado em IA - upDesk

Alunos:

| RA | Nome | E-mail | Curso | Visto do aluno |
|---|---|---|---|---|
| **G922CG4** | **Andrei Henrique Mancijo** | **andrei.mancijo@aluno.unip.br** | **CST em ADS** | |
| **R084353** | **Filipe Vitor dos Santos** | **filipevitor6@gmail.com** | **CST em ADS** | |
| **G9038F8** | **Jônatas dos Santos Souza** | **jonatas.souza20@aluno.unip.br** | **CST em ADS** | |
| **G03IGG0** | **Kaique Batista da Silva** | **kaique.silva107@aluno.unip.br** | **CST em ADS** | |
| **G9884G2** | **Mariozan Damasceno Lacerda Júnior** | **mariozanjunior15@gmail.com** | **CST em ADS** | |
| **G9265G4** | **Mateus Teodoro da Silva** | **mateus.silva329@aluno.unip.br** | **CST em ADS** | |

Registros:

| Data do encontro | Observações |
|---|---|
| **08 ago 2025** | **Planejamento do backlog, definição de sprints** |
| **23 ago 2025** | **Modelagem dos diagramas (Classe de Implementação, Sequência, Colaboração)** |
| **07 set 2025** | **Finalização do CRUD do banco de dados em SQL Server** |
| **22 set 2025** | **Início do desenvolvimento web (estrutura, layout, rotas principais)** |
| **07 out 2025** | **Implementação das funcionalidades principais e integração com IA** |
| **22 out 2025** | **Início do desenvolvimento desktop e mobile (estrutura inicial e navegação)** |
| **06 nov 2025** | **Finalização dos sistemas web, mobile e desktop, testes unitários e ajustes na interface** |
| **14 nov 2025** | **Documentação técnica, revisão geral, correções finais e entrega do projeto** |