

UNIVERSIDADE PAULISTA – UNIP
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ARTHUR SILVESTRE DE PAULA
JOÃO HENRIQUE SILVA DOS SANTOS
IGOR DANTAS BARROS
LAIS SILVA SANTOS
MELYSSA SOUZA ARAUJO
RENAN FRANCISCO RIBEIRO
WILLIAM GUSTAVO ABREU DE OLIVEIRA LEITE

**DESENVOLVIMENTO DE UM SISTEMA INTEGRADO PARA GESTÃO DE
CHAMADOS E SUPORTE TÉCNICO COM APOIO DE IA**

SANTOS
2025

UNIVERSIDADE PAULISTA – UNIP

CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ARTHUR SILVESTRE DE PAULA
JOÃO HENRIQUE SILVA DOS SANTOS
IGOR DANTAS BARROS
LAIS SILVA SANTOS
MELYSSA SOUZA ARAUJO
RENAN FRANCISCO RIBEIRO
WILLIAM GUSTAVO ABREU DE OLIVEIRA LEITE

DESENVOLVIMENTO DE UM SISTEMA INTEGRADO PARA GESTÃO DE CHAMADOS E SUPORTE TÉCNICO COM APOIO DE IA

PROJETO INTEGRADO MULTIDISCIPLINAR – PIM III E IV

Trabalho apresentado como exigência parcial para a avaliação das disciplinas do 3º e 4º semestre do curso de Análise e Desenvolvimento de Sistemas da Universidade Paulista – UNIP.

Professora Orientadora: **Thais** Ferauche

SANTOS

2025

FOLHA DE APROVAÇÃO

O presente trabalho foi avaliado e aprovado pela professora orientadora, como requisito parcial para a obtenção de nota nas disciplinas do 3º e 4º semestre do curso de Análise e Desenvolvimento de Sistemas da Universidade Paulista – UNIP.

Professora

Orientadora:

Thais

Ferauche

Assinatura

Santos, 2025.

RESUMO

O presente Projeto Integrado Multidisciplinar tem como objetivo o desenvolvimento de um Sistema Integrado para Gestão de Chamados e Suporte Técnico com apoio de Inteligência Artificial (IA). A solução foi projetada para otimizar o atendimento técnico, reduzindo o tempo de resposta, centralizando informações e automatizando tarefas repetitivas por meio de modelos de IA. O sistema integra três plataformas — Web, Desktop e Mobile — conectadas a uma API REST desenvolvida em ASP.NET Core, com banco de dados SQL Server e execução em contêiner Docker. Foram aplicados conceitos de programação orientada a objetos, engenharia de software, gerenciamento de projetos, usabilidade, acessibilidade, segurança da informação e conformidade com a Lei Geral de Proteção de Dados (LGPD). Os resultados incluem maior eficiência operacional, padronização do suporte e respostas automáticas em poucos segundos.

Palavras-chave: Suporte Técnico. Inteligência Artificial. Chamados. Desenvolvimento de Software. Automação.

ABSTRACT

This Multidisciplinary Integrated Project aims to develop an Integrated System for Ticket Management and Technical Support powered by Artificial Intelligence. The solution was designed to optimize technical support by reducing response time, centralizing information, and automating repetitive tasks through AI models. The system integrates three platforms — Web, Desktop, and Mobile — connected to a REST API developed in ASP.NET Core, using SQL Server and Docker containers. Concepts from object-oriented programming, software engineering, project management, usability, accessibility, information security, and compliance with the Brazilian General Data Protection Law (LGPD) were applied. The results include improved operational efficiency, standardized support processes, and automatic responses generated within seconds.

Keywords: Technical Support. Artificial Intelligence. Ticket Management. Software Development. Automation.

Sumário

1. Introdução	1
2. Desenvolvimento da Engenharia de Software	2
2.1 Contextualização da Empresa Contratante	2
2.2 Situação Atual e Desafios Enfrentados	2
2.3 Empresa Desenvolvedora Contratada	2
2.4 Conformidade com a LPGD	3
2.6 Requisitos Não Funcionais (RNF).....	4
2.7 Critérios de Qualidade.....	5
2.8 Dificuldades e Soluções Propostas	6
2.9 Caso de Teste com a IA.....	6
2.10 Testes realizados.....	7
2.11 Boas Práticas Aplicadas.....	7
2.12 Registro de Alterações e Justificativas	7
3. DIAGRAMAS	8
3.1 MODELAGEM DO SISTEMA.....	8
3.2 DIAGRAMAS UML DO SISTEMA CONNECTWAY	8
3.3 Diagrama de Casos de Uso	8
3.4 Diagrama de Classes	9
3.5 Diagrama de Sequência.....	10
4. ER BANCO DE DADOS.....	12
4.1 Descrição do Banco de Dados.....	12
5. Gerenciamento de Projetos de Software.....	13
5.1 Objetivo do Projeto	13
5.2 Descrição do Produto	13
5.3 Justificativa	14
5.4 Escopo do Projeto	14
5.4.1 Características do Projeto	14
5.4.2 Fora do Escopo	15
5.5 Estrutura Analítica do Projeto (EAP).....	16
5.6 Cronograma do Projeto	16
5.7 Orçamento do Projeto	17
5.8 Premissas e Restrições.....	18

5.9 Análise de Riscos do Projeto	18
5.10 Equipe do Projeto	19
6. Gestão da Qualidade	20
6.1 Tabela 1 – Práticas e indicadores da Gestão da Qualidade.....	21
7. Empreendedorismo	22
7.1 Tabela 2 – Aspectos Empreendedores do Projeto.....	23
8. Relações Étnico-Raciais e Afrodescendência.....	24
9. Documentação do Back-end.....	25
9.1 Arquitetura do Sistema	25
9.2 Funcionalidades e Autenticação	26
9.3 Gerenciamento de Chamados	26
Integração com IA.....	27
Execução do Projeto.....	27
Testes.....	27
10. Relatório Front-End	28
10.1 Estética e Design Visual Paleta de Cores	29
10.2 Tipografia	29
10.3 Iconografia e Elementos Gráficos	29
10.4 Consistência e Padrões de Design.....	30
10.5 Login e Registro (UX)	30
10.6 Avaliação do Atendimento Solicita:.....	31
Telas do Aplicativo.....	31
10.7 Tecnologias Utilizadas no Frontend.....	31
Execução do Projeto.....	32
Pré-requisitos	32
11. Telas das Páginas Web/Desktop.....	32
12. Telas das Páginas Mobile	36
13. Códigos Back-end:	42
15. PROMPTS UTILIZADOS COM IA (ChatGPT e Gemini)	118
Planejamento e Estrutura do Projeto	118
Desenvolvimento do Backend (API / C# / ASP.NET Core).....	119
Desenvolvimento do Frontend (React Native / Expo)	119
Design, UI e UX	119

Banco de Dados e Modelagem (DER / SQL Server)	119
Integração com IA (Gemini / ChatGPT)	120
Documentação do Projeto (Relatório / Descrição / Manual)	120
Resolução de Problemas Durante o Desenvolvimento.....	120
Testes e Qualidade.....	120
Geração das Telas e Explicação Técnica.....	121
Automatização do Documento (Word / ABNT)	121
17. CONCLUSÃO.....	122
18. REFERÊNCIAS	123

1. Introdução

O avanço das tecnologias digitais e a crescente complexidade dos ambientes corporativos têm exigido soluções cada vez mais integradas e inteligentes para o gerenciamento de processos internos. Nesse contexto, a ConnectWay Tecnologias e Soluções identificou a necessidade de aprimorar seu sistema de atendimento técnico, que até então era realizado de forma manual e descentralizada, resultando em retrabalho, atrasos e perda de informações importantes.

Para atender a essa demanda, foi proposto o desenvolvimento de um Sistema Integrado para Gestão de Chamados e Suporte Técnico baseado em Inteligência Artificial (IA). A solução visa centralizar a abertura, o acompanhamento e a resolução de chamados, automatizando tarefas repetitivas e otimizando a comunicação entre usuários e equipe técnica.

O sistema proposto contempla três plataformas — Web, Desktop e Mobile — que se integram a um módulo de Inteligência Artificial responsável por classificar solicitações, sugerir soluções automáticas e priorizar atendimentos conforme o nível de criticidade. Além disso, o projeto assegura conformidade com a Lei Geral de Proteção de Dados (LGPD), garantindo a privacidade e a segurança das informações manipuladas.

Sob o ponto de vista da Engenharia de Software, o desenvolvimento do sistema segue práticas consolidadas de levantamento de requisitos, modelagem, prototipação, versionamento e testes. Já no âmbito do Gerenciamento de Projetos de Software, foram definidos o escopo, cronograma, orçamento, análise de riscos e estrutura analítica do projeto, assegurando planejamento e execução eficientes.

Com isso, o presente trabalho busca demonstrar o processo de concepção e implementação de uma ferramenta moderna, escalável e inteligente, capaz de transformar o suporte técnico interno em um processo ágil, confiável e orientado por dados.

2. Desenvolvimento da Engenharia de Software

2.1 Contextualização da Empresa Contratante

A ConnectWay Suporte Técnico com IA é uma empresa especializada em soluções digitais voltadas exclusivamente para o suporte técnico automatizado com inteligência artificial. Fundada em São Paulo, a empresa atua no atendimento remoto e automatizado de chamados técnicos, eliminando a necessidade de equipes presenciais. A ConnectWay desenvolve e opera um sistema inteligente que integra chatbots, categorização automática de problemas e priorização de chamados, proporcionando respostas rápidas e precisas aos usuários. A equipe é composta por profissionais focados em ciência de dados e engenharia de software, que aprimoram continuamente os algoritmos de IA responsáveis por interpretar e resolver solicitações.

2.2 Situação Atual e Desafios Enfrentados

Antes da implementação do sistema atual, a abertura e o acompanhamento de chamados técnicos eram realizados manualmente, por e-mail ou telefone. Esse modelo apresentava diversas limitações, como a falta de centralização das informações e ausência de priorização automática.

Principais desafios identificados:

- Falta de registro centralizado e histórico de atendimentos.
- Priorização ineficiente dos chamados, sem critérios automatizados.
- Ausência de acompanhamento em tempo real e notificações automáticas.
- Sobrecarga dos profissionais responsáveis pela triagem manual.
- Dificuldade em extrair relatórios e indicadores de desempenho.
- Limitações na categorização manual de problemas e repetição de erros.

2.3 Empresa Desenvolvedora Contratada

Soluções Nexusoft

CNPJ: 00.123.456/0001-78

Endereço: Rua da Inovação, Nº 245 – Sala 801, Bairro Barra Funda, São Paulo – SP, CEP 04567-000

Contato: (11) 3123-4567 | WhatsApp: (11) 91234-5678 | E-mail: contato@nexusoft.com.br

A Nexusoft é uma empresa de tecnologia fundada em 2021, especializada no desenvolvimento de sistemas inteligentes para suporte técnico automatizado. Composta por uma equipe enxuta e multidisciplinar, seu foco está em entregar

soluções escaláveis e seguras, com uso de inteligência artificial aplicada à gestão de chamados e automação de respostas.

2.4 Conformidade com a LPGD

Durante todo o processo de desenvolvimento e testes, foram observadas as diretrizes da Lei Geral de Proteção de Dados (LGPD). Os dados utilizados para treinar e testar os algoritmos de IA foram devidamente anonimizados, e os acessos são restritos a usuários autorizados.

Principais ações adotadas:

- Solicitação de consentimento explícito dos usuários.
- Registro de finalidade e ciclo de vida dos dados coletados.
- Implementação de controles de acesso e anonimização.
- Criação de relatórios de impacto à proteção de dados (RIPD).

2.5 Requisitos Funcionais (RF)

Código	Descrição
RF01	Permitir o cadastro de usuários (colaboradores e administradores).
RF02	Permitir login com autenticação por nível de acesso (JWT).
RF03	Permitir que usuários abram chamados com descrição, categoria e urgência.
RF04	Permitir que administradores visualizem, filtrem e atualizem o status dos chamados.
RF05	Permitir que usuários acompanhem o andamento dos seus chamados.
RF06	Permitir que administradores cadastrem e editem categorias e configurações.
RF07	Gerar relatórios de atendimento por período, tipo e status.
RF08	Enviar notificações (push e e-mail) sobre atualizações dos chamados.
RF09	Registrar o histórico de ações realizadas em cada chamado.
RF10	Permitir o fechamento e avaliação do atendimento ao final do chamado.
RF11	Armazenar e consultar base de conhecimento com soluções anteriores.
RF12	Permitir que o aplicativo mobile abra e acompanhe chamados remotamente.

RF13	(Futuro) Sugerir automaticamente uma categoria com base na descrição do chamado (IA).
RF14	(Futuro) Sugerir possíveis soluções com base em chamados anteriores (IA).
RF15	(Futuro) Permitir integração com serviços externos de IA (OpenAI, Gemini, Azure).

2.6 Requisitos Não Funcionais (RNF)

Código	Categoria	Descrição
RNF01	Desempenho	O sistema deve responder a qualquer solicitação em até 3 segundos.
RNF02	Desempenho	Deve suportar até 200 chamados simultâneos sem perda de desempenho.
RNF03	Confiabilidade	Deve estar disponível para acesso 99% do tempo útil da semana.
RNF04	Usabilidade	A interface deve ser clara, intuitiva e com linguagem acessível.
RNF05	Acessibilidade	Deve permitir navegação por teclado e suporte a leitores de tela.
RNF06	Acessibilidade	Deve seguir contraste mínimo de 4,5:1 conforme WCAG.
RNF07	Segurança	Dados pessoais devem ser criptografados e protegidos contra acesso não autorizado.
RNF08	Segurança	Autenticação e controle de acesso por níveis de permissão.
RNF09	LGPD	Permitir consulta, edição e exclusão de dados pessoais conforme LGPD.
RNF10	Segurança	Todos os acessos e alterações devem ser registrados em log.
RNF11	Manutenibilidade	Seguir arquitetura

		modular (MVC) e boa documentação.
RNF12	Escalabilidade	Permitir adição de novos módulos (como IA) sem reestruturar o sistema.
RNF13	Qualidade	Código-fonte versionado (Git) e seguir boas práticas de documentação.

2.7 Critérios de Qualidade

Critério	Descrição	Ações Concretas no Projeto	Ferramentas/Técnicas Utilizadas
Usabilidade	Facilidade de aprendizado e operação.	Layout intuitivo; feedback visual; linguagem simples.	UX/UI, prototipação (Figma), testes com usuários
Acessibilidade	Garantir uso por pessoas com diferentes habilidades.	Alto contraste; navegação por teclado; leitores de tela.	WCAG, Axe DevTools
Desempenho	Agilidade no carregamento e resposta.	Otimização de consultas; cache local no mobile.	SQL Profiler, testes de carga (k6, JMeter)
Segurança	Proteção dos dados e controle de acesso.	Autenticação JWT; criptografia; políticas de acesso.	ASP.NET Identity, bcrypt
Testabilidade	Facilidade de testar e validar o sistema.	Código modular; testes automatizados por módulo.	xUnit, Moq, integração CI
Manutenibilidade	Facilidade de corrigir erros e evoluir.	Comentários no código; documentação técnica; API bem estruturada.	Swagger, documentação, padrões MVC
Portabilidade	Funcionar diferentes ambientes e dispositivos.	Versões para desktop, web e mobile; interface responsiva.	Xamarin, ASP.NET Core, testes em múltiplos navegadores

2.8 Dificuldades e Soluções Propostas

Dificuldade	Descrição	Solução Proposta
Mudança de requisitos durante o projeto	O cliente pode solicitar alterações mesmo após a aprovação inicial.	Adotar metodologia ágil com sprints curtos e validações contínuas.
Equipe pequena	Recursos humanos limitados para várias plataformas.	Dividir tarefas por especialidade e usar ferramentas de produtividade.
Dificuldade técnica em múltiplas tecnologias	Nem todos dominam WPF, Android, ASP.NET, SQL Server.	Promover treinamentos internos e usar repositórios com modelos prontos.
Integração entre desktop, web e mobile	Compartilhamento de dados entre plataformas pode gerar falhas.	Usar APIs RESTful padronizadas e sincronização centralizada.
Falta de testes com usuários reais	Funcionalidades podem ser entregues sem validação prática.	Criar versões MVP e realizar testes com usuários da ConnectWay.
Baixa usabilidade e acessibilidade	Dificuldade de uso por pessoas com deficiência.	Aplicar princípios de design universal e testar com leitores de tela.
Demora no feedback do cliente	Atrasos por não saber se a funcionalidade está aprovada.	Reuniões quinzenais e demonstrações de progresso.
Conformidade com LGPD	Risco de vazamento ou uso indevido de dados pessoais.	Implementar consentimento, criptografia e política de privacidade.
Dificuldade na futura implementação de IA	IA exige dados, testes e integração com serviços externos.	Planejar fase de coleta e classificação de dados; estudar APIs externas.

2.9 Caso de Teste com a IA

Exemplos de chamados com a IA

Descrição do Chamado (Input)	Categoria Esperada	Solução Esperada (Sugerida pela IA)
Não consigo imprimir na impressora da recepção	Impressão/Dispositivos	Verificar conexão USB ou rede e reiniciar

		spooler de impressão.
Minha senha de login do sistema não funciona	Acesso/Senhas	Realizar redefinição de senha via sistema interno.
A internet está oscilando muito no escritório 2	Rede/Conectividade	Verificar cabo de rede, reiniciar switch local e checar link com provedor.

2.10 Testes realizados

Durante o desenvolvimento do sistema, foram aplicados quatro tipos principais de testes para garantir sua qualidade e segurança:

- Testes Unitários: verificaram o funcionamento isolado de componentes, como autenticação e categorização de chamados.
- Testes de Integração: validaram a comunicação entre os módulos de IA, API e banco de dados.
- Testes de Segurança: analisaram a proteção de dados, autenticação e controle de acesso conforme a LGPD.
- Testes de Validação: asseguraram que o sistema final atendesse às expectativas de desempenho e usabilidade.

2.11 Boas Práticas Aplicadas

Para manter a qualidade e rastreabilidade do código, foram utilizadas boas práticas de engenharia de software e controle de versão:

- Uso de Git e GitHub para versionamento distribuído.
- Criação de branches específicas para novas funcionalidades e correções.
- Commits frequentes e descritivos seguindo convenções semânticas.
- Pull Requests revisados antes da integração ao repositório principal.
- Backups automáticos e documentação contínua das alterações.

2.12 Registro de Alterações e Justificativas

Todas as alterações deste documento foram realizadas para alinhar o conteúdo ao novo escopo da empresa e ao modelo de sistema desenvolvido. As mudanças refletem a transição de uma empresa de infraestrutura de TI para uma empresa 100% digital focada em suporte técnico com IA.

Principais modificações:

- Atualização da contextualização da empresa (seção 2.1) – agora reflete operação digital automatizada.
- Revisão completa da situação atual e desafios (seção 2.2) – adaptada à realidade de suporte virtual.
- Enfoque em IA e automação na descrição da Nexusoft (seção 2.3).
- Inclusão detalhada dos testes realizados (seção 2.8), especificando unitários, integração, segurança e validação.
- Inserção da seção 2.12 para garantir rastreabilidade documental das alterações.

3. DIAGRAMAS

O desenvolvimento de sistemas orientados a objetos (PSOO) busca representar o mundo real por meio de entidades que possuem atributos e comportamentos, promovendo modularidade, reutilização e manutenção facilitada. Essa abordagem utiliza a UML (Unified Modeling Language) como linguagem de modelagem padronizada para descrever a estrutura e o comportamento do sistema.

O projeto ConnectWay propõe um sistema de suporte técnico automatizado por inteligência artificial (IA), capaz de compreender solicitações de usuários, oferecer soluções automáticas e registrar as interações. Este documento apresenta os principais diagramas UML do sistema, atualizados com base nas orientações do professor em sala de aula, refletindo uma versão mais coerente e moderna do modelo conceitual.

3.1 MODELAGEM DO SISTEMA

A modelagem orientada a objetos busca compreender e representar as entidades e suas interações dentro do domínio do problema. No caso do sistema ConnectWay, a modelagem se apoia na UML para expressar, por meio de diagramas, tanto a estrutura estática (classes e relacionamentos) quanto o comportamento dinâmico (interações e sequências de mensagens).

3.2 DIAGRAMAS UML DO SISTEMA CONNECTWAY

Os diagramas UML foram elaborados para representar a estrutura, o comportamento e o fluxo de informações do sistema ConnectWay.

A seguir, são apresentados os principais diagramas utilizados no projeto.

3.3 Diagrama de Casos de Uso

O diagrama de casos de uso foi reformulado conforme orientação do professor, eliminando o ator “Técnico” e redefinindo o sistema como um suporte técnico totalmente automatizado por IA. Agora, o foco está nas funcionalidades

voltadas ao usuário, como cadastro, envio de prompts e análise de respostas pela inteligência artificial.

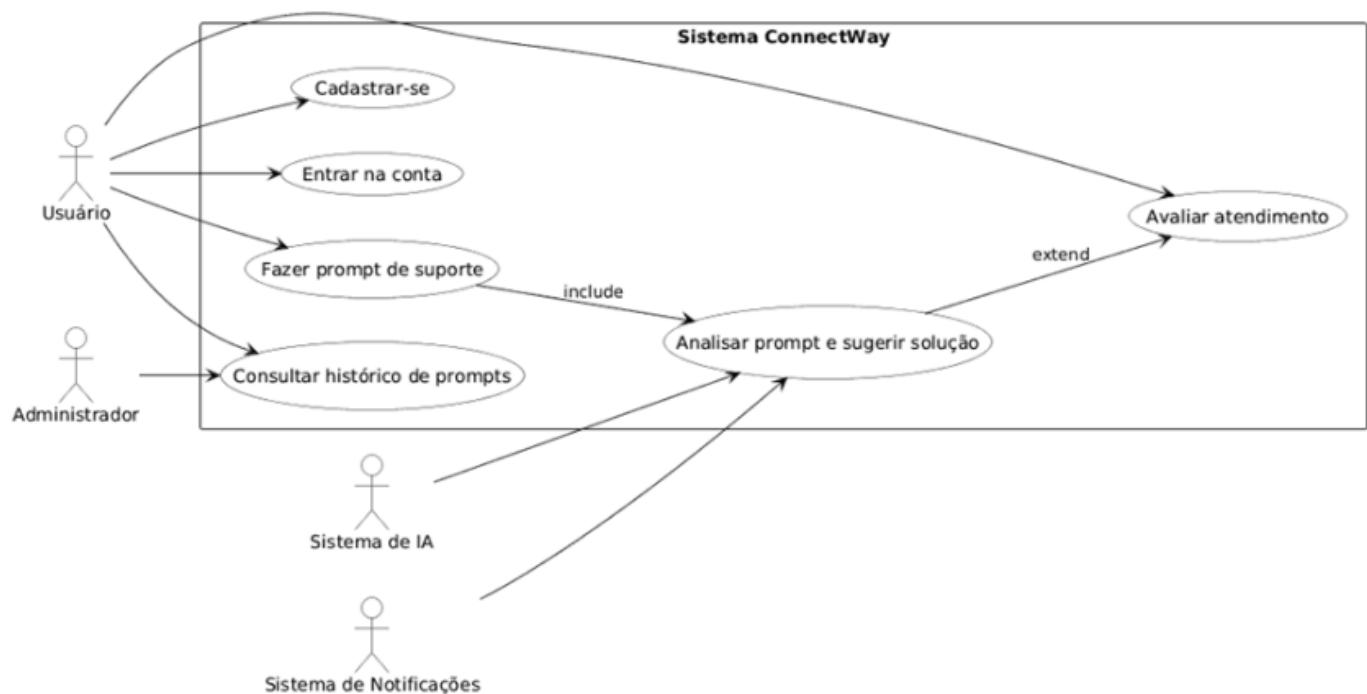


Figura 1 – Diagrama de Casos de Uso (Atualizado)

Fonte: Elaborado por PlantUML (08 nov. 2025).

3.4 Diagrama de Classes

O diagrama de classes foi atualizado para refletir a nova estrutura do sistema, composta apenas por entidades relacionadas à IA e interação com o usuário. A classe “Técnico” foi removida e novas classes, como SistemAI, Notificação e Histórico, foram adicionadas. Essas modificações foram orientadas em aula para reforçar o modelo de automação inteligente e a coerência entre os diagramas UML.

Sistema ConnectWay - Diagrama de Classes (Suporte Técnico com IA)

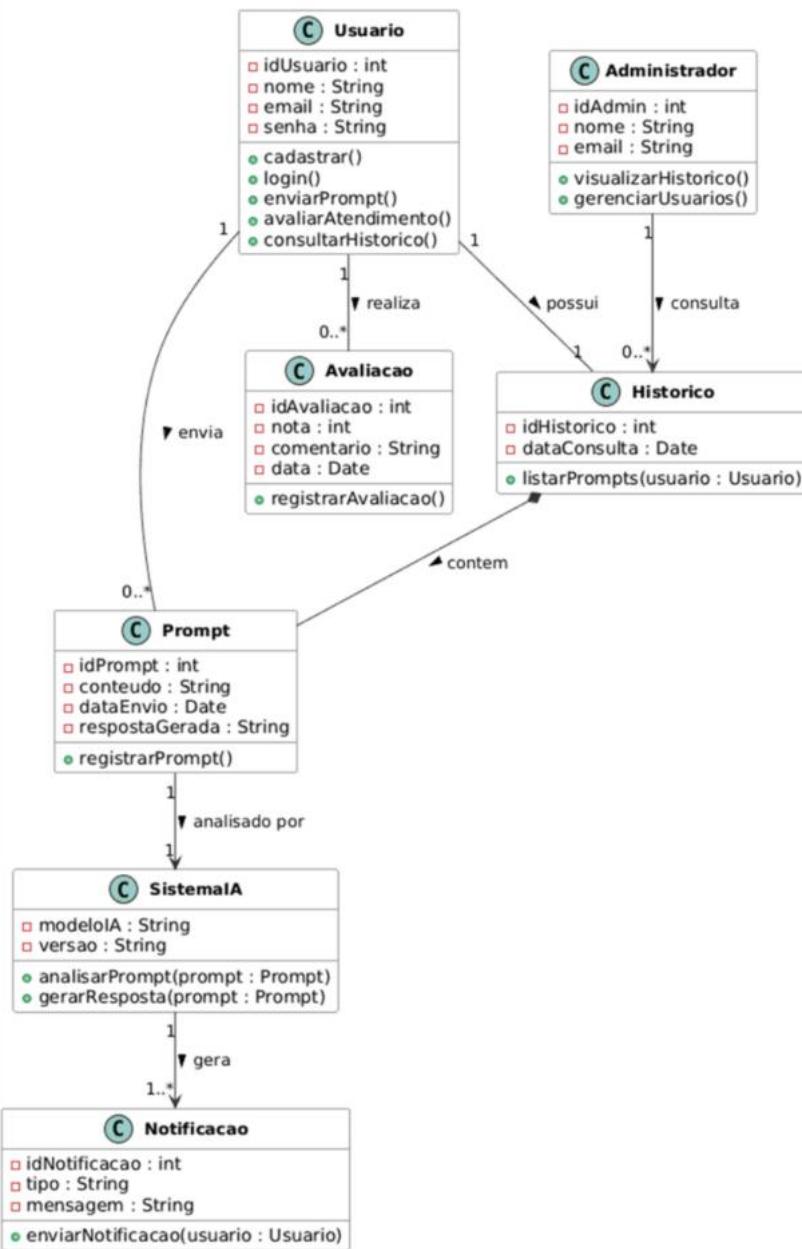


Figura 2 – Diagrama de Classes (Atualizado)

Fonte: Elaborado por PlantUML (08 nov. 2025).

3.5 Diagrama de Sequência

O diagrama de sequência foi revisado para demonstrar o fluxo dinâmico de um atendimento automatizado. Ele exibe a interação entre o usuário e a IA em todas as etapas: envio de prompt, geração de resposta, avaliação do atendimento e registro no histórico. Essa atualização segue as recomendações do professor, garantindo uma representação temporal mais clara e fiel ao comportamento do sistema.

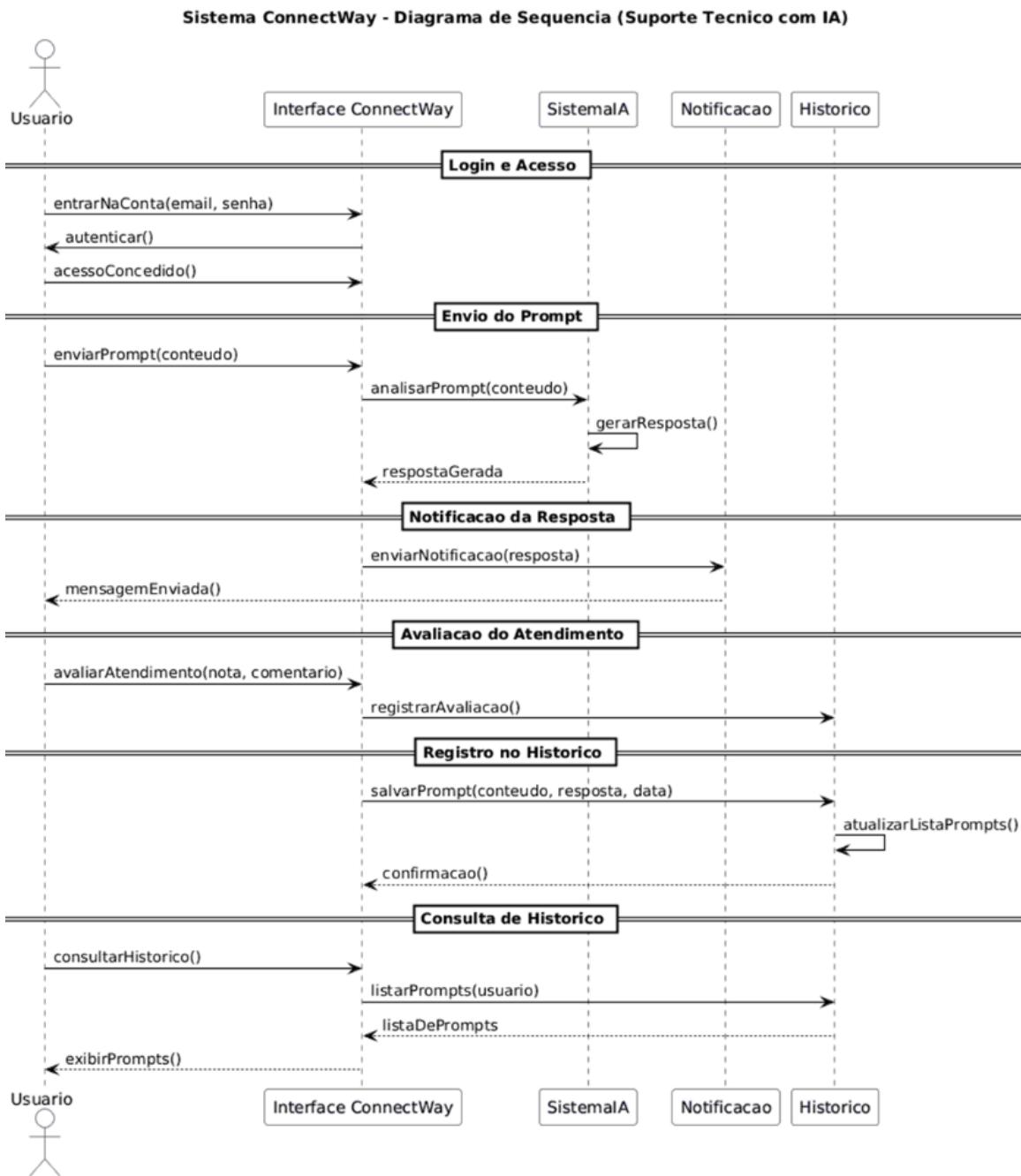


Figura 3 – Diagrama de Sequência (Atualizado)

Fonte: Elaborado por PlantUML (08 nov. 2025).

As atualizações realizadas nos diagramas UML do sistema ConnectWay resultaram em uma representação mais moderna, concisa e aderente à proposta de automação total por IA. As mudanças aplicadas a partir das orientações docentes aprimoraram a coerência entre os diagramas e reforçaram o entendimento do sistema como um ambiente inteligente e integrado, sem a necessidade de intervenção humana direta.

A adoção da modelagem orientada a objetos e da UML permitiu consolidar uma base técnica robusta, facilitando o processo de análise, projeto e documentação do sistema, em conformidade com as boas práticas de engenharia de software.

4. ER BANCO DE DADOS

O banco de dados do sistema foi desenvolvido com o Entity Framework Core 8, utilizando a abordagem Code First. Isso significa que o modelo de dados foi criado a partir das classes da aplicação, e as tabelas foram geradas automaticamente por meio das migrations. O ambiente de banco de dados roda em um contêiner Docker com SQL Server, o que oferece maior portabilidade, facilidade de configuração e consistência entre diferentes ambientes de desenvolvimento.

O diagrama de banco de dados representa as entidades principais do sistema de gestão de chamados integrados com inteligência artificial, e os relacionamentos entre elas garantem a integridade e a consistência dos dados. A tabela Chamado atua como o núcleo central, conectando as informações de solicitantes, técnicos, categorias, status e histórico de interações.

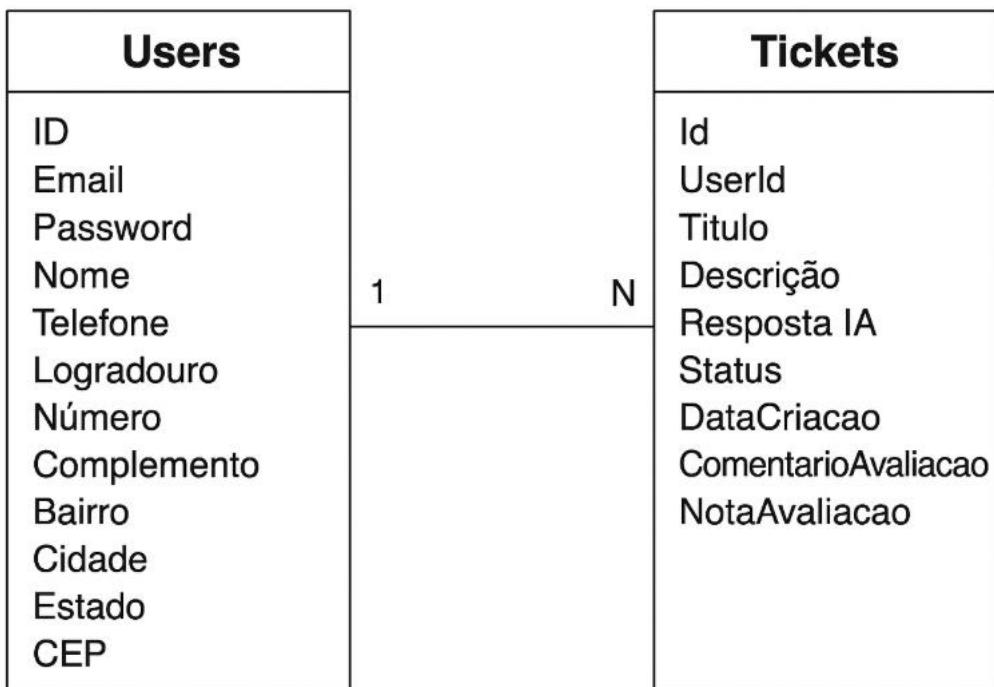
4.1 Descrição do Banco de Dados

O banco de dados do sistema de suporte técnico com IA é composto por duas tabelas principais: **Users** e **Tickets**.

A tabela **Users** armazena as informações dos usuários cadastrados, como nome, email, telefone e endereço completo. O atributo ID é a chave primária e identifica unicamente cada usuário. Cada usuário pode criar vários tickets de atendimento, estabelecendo um relacionamento de **um para muitos (1:N)** com a tabela Tickets.

A tabela **Tickets** registra os chamados abertos pelos usuários, incluindo título, descrição, status, data de criação, resposta fornecida pela inteligência artificial e avaliação do atendimento. O atributo Id é a chave primária e UserId é uma chave estrangeira que referencia Users.ID, garantindo que cada ticket esteja vinculado a um usuário específico.

O relacionamento entre as tabelas é de **1:N**, onde um usuário pode ter múltiplos tickets, mas cada ticket pertence a apenas um usuário. Esse modelo permite organizar as informações de maneira eficiente, garantindo integridade e facilitando a gestão do suporte técnico oferecido pela IA.



5. Gerenciamento de Projetos de Software

5.1 Objetivo do Projeto

Desenvolver e implantar um sistema centralizado para gestão de chamados internos, visando otimizar o processo de atendimento técnico, reduzir o tempo de resposta, aprimorar a priorização de solicitações e padronizar as informações registradas, utilizando um módulo de Inteligência Artificial para aumentar a eficiência operacional.

5.2 Descrição do Produto

O Sistema Integrado de Gestão de Chamados é uma plataforma criada para centralizar e organizar solicitações de suporte técnico dentro da ConnectWay Tecnologias e Soluções.

Ele será acessado por meio de interface Web, sistema Desktop e aplicativo Mobile, permitindo a abertura de chamados, o acompanhamento do status de atendimento e o acesso ao histórico de solicitações.

O sistema contará também com uma Base de Conhecimento, onde serão registradas as soluções aplicadas anteriormente, facilitando a consulta e padronizando a resolução de problemas.

Além disso, o sistema incluirá um módulo de Inteligência Artificial capaz de analisar os chamados recebidos, sugerir possíveis soluções, priorizar atendimentos,

e responder automaticamente demandas simples, reduzindo o tempo médio de resolução.

5.3 Justificativa

O projeto é necessário devido à falta de padronização nos atendimentos realizados pela equipe de suporte, que atualmente utiliza diversos meios não integrados para registrar solicitações. Isso gera perda de histórico, atrasos nos atendimentos, dificuldade de priorização e retrabalho, afetando diretamente a qualidade do serviço prestado.

Com a implantação do sistema, espera-se:

- Redução do tempo médio de atendimento;
- Melhor organização e priorização dos chamados;
- Registro centralizado das solicitações e soluções aplicadas;
- Automatização de tarefas simples por IA;
- Aumento da eficiência interna e da satisfação dos usuários.

5.4 Escopo do Projeto

O escopo do produto descreve as características e funcionalidades do Sistema Integrado de Gestão de Chamados desenvolvido para a ConnectWay Tecnologias e Soluções.

5.4.1 Características do Projeto

- **Tipo de Sistema:** Plataforma integrada (Web, Desktop e Mobile).
- **Função Principal:** Registro, acompanhamento e resolução de chamados de suporte técnico.
- **Interface Web (Usuários):**
 - Abertura de chamados.
 - Consulta ao status do atendimento.
 - Histórico de solicitações.
- **Sistema Desktop (Equipe Técnica):**
 - Visualização e gestão de chamados em tempo real.
 - Definição de prioridades.
 - Registro de soluções e encerramentos.
 - Acesso à base de conhecimento.
- **Aplicativo Mobile (Usuários):**
 - Abertura rápida de chamados.
 - Notificações de atualização do status.
 - Consulta ao histórico.
- **Banco de Dados Centralizado:** Armazena todos os chamados, interações e soluções.
- **Base de Conhecimento:** Artigos, procedimentos e soluções padronizadas.
- **Módulo de Inteligência Artificial:**
 - Classificação automática de chamados.

- Sugestão de solução com base em histórico.
- Priorização conforme nível de urgência.
- Respostas automáticas para problemas simples.
- **Tema/Identidade:** Interface limpa, moderna e objetiva, seguindo o padrão visual da ConnectWay.

Levantamento e Planejamento

- Reunião com a equipe.
- Mapeamento do processo atual de suporte.
- Coleta e validação de requisitos funcionais e técnicos.

Design e Arquitetura

- Modelagem do banco de dados.
- Criação do fluxo de atendimento.
- Protótipos das interfaces (Web, Desktop e Mobile).

Desenvolvimento

- Programação da interface Web para usuários.
- Desenvolvimento do sistema Desktop para técnicos.
- Desenvolvimento do aplicativo Mobile.
- Criação da base de conhecimento (estrutura inicial).
- Implementação do módulo de IA (priorização e sugestões automáticas).

Testes

- Testes funcionais por módulo.
- Testes integrados entre plataformas.
- Testes piloto com equipe interna.

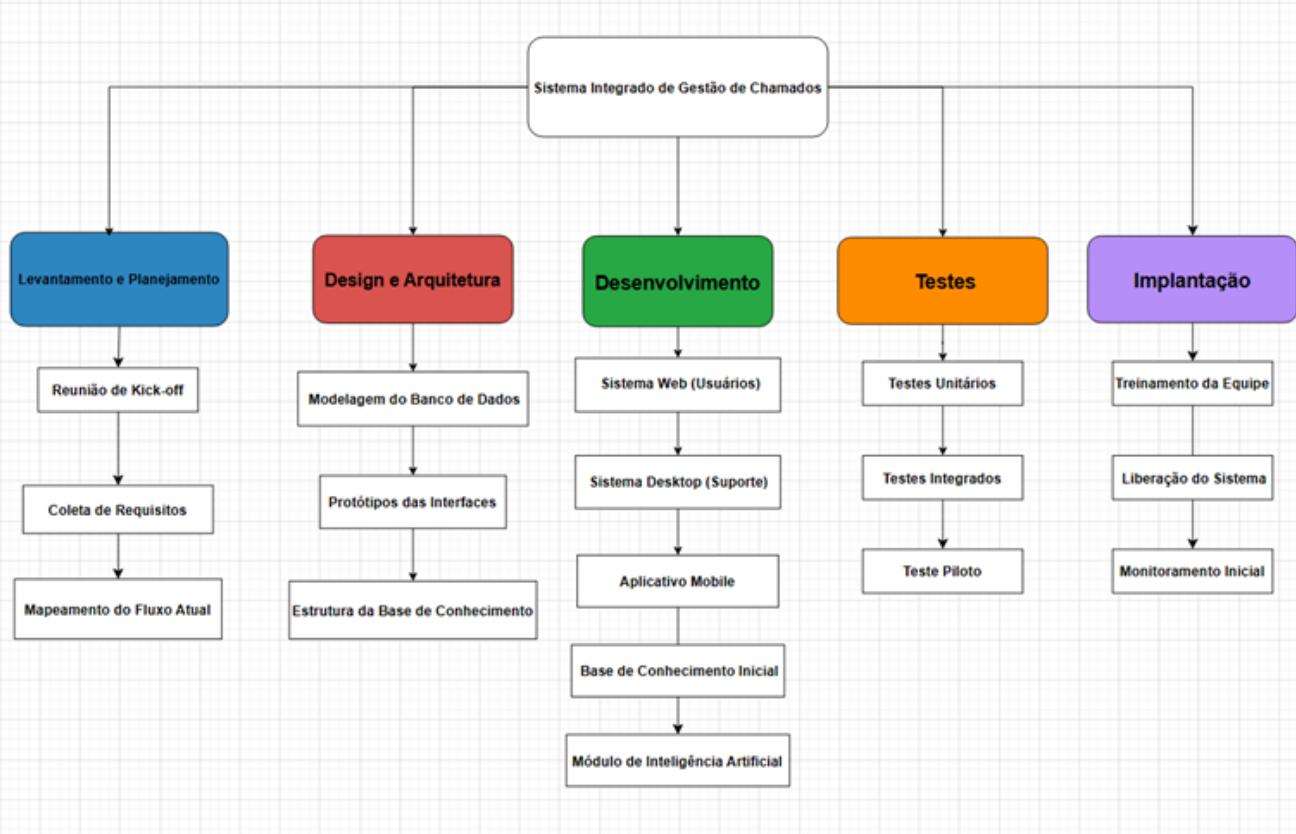
Implantação e Entrega

- Migração dos dados necessários.
- Treinamento da equipe de suporte.
- Liberação do sistema para uso interno.
- Monitoramento inicial pós-implantação.

5.4.2 Fora do Escopo

- Atendimento ao cliente externo.
- Suporte 24h.
- Integração com sistemas terceiros nesta fase.
- Customizações individuais por setor sem solicitação formal.

5.5 Estrutura Analítica do Projeto (EAP)



5.6 Cronograma do Projeto

O projeto será executado ao longo de **24 semanas (aproximadamente 6 meses)**, com as atividades organizadas de forma sequencial para garantir o desenvolvimento, testes e implantação do sistema dentro do prazo previsto.

Levantamento e Planejamento (3 semanas):

Atividades:

- Reunião de Kick-off
- Coleta de requisitos com usuários internos
- Mapeamento do processo atual de atendimento

Design e Arquitetura (4 semanas):

Atividades:

- Modelagem do banco de dados
- Criação dos protótipos das interfaces (Web, Desktop e Mobile)

- Estrutura inicial da Base de Conhecimento

Desenvolvimento (9 semanas):

Atividades:

- Desenvolvimento da Interface Web (para abertura e acompanhamento de chamados)
- Desenvolvimento do Sistema Desktop (painel da equipe de suporte)
- Desenvolvimento do Aplicativo Mobile (notificações e abertura rápida)
- Criação da Base de Conhecimento inicial
- Implementação do módulo de Inteligência Artificial (classificação automática, sugestões e priorização).

Testes (4 semanas):

Atividades:

- Testes unitários dos módulos
- Testes de integração entre Web, Desktop e Mobile
- Teste piloto com equipe interna selecionada

Implantação (4 semanas):

Atividades:

- Treinamento da equipe de suporte
- Liberação oficial do sistema
- Monitoramento inicial e ajustes finais pós-implantação

5.7 Orçamento do Projeto

O custo total estimado do projeto é de **R\$ 15.000,00**, distribuído da seguinte forma:

Infraestrutura e Hospedagem (R\$ 2.500,00)

Contratação do ambiente em nuvem, domínio, certificados e banco de dados.

Ferramentas e Serviços de Suporte (R\$ 1.500,00)

Plataformas de comunicação, APIs auxiliares e monitoramento.

Mão de Obra Técnica (R\$ 11.000,00)

Equivalente ao trabalho da equipe em análise, design, programação, testes e implantação.

5.8 Premissas e Restrições

Premissas	Restrições
<ul style="list-style-type: none"> A equipe envolvida no projeto estará disponível para as atividades de levantamento, validação e testes. Os usuários fornecerão informações claras sobre suas necessidades e forma de utilização do sistema. Os equipamentos e softwares necessários para o desenvolvimento estarão disponíveis durante todo o projeto. A comunicação entre as áreas será realizada de forma contínua, por meio de reuniões agendadas ou canais internos. O cronograma será seguido conforme o planejamento, salvo imprevistos devidamente justificados. 	<ul style="list-style-type: none"> O prazo máximo para conclusão do projeto é de 24 semanas (6 meses), sem possibilidade de extensão. O desenvolvimento será limitado aos módulos definidos no escopo, sem incluir novas funcionalidades não previstas. O sistema será disponibilizado inicialmente apenas para uso interno pelos colaboradores autorizados, sendo sua expansão para outros setores ou público externo planejada apenas em fases futuras. Qualquer alteração de requisitos após o início do desenvolvimento deverá passar por validação e aprovação da gerência. O custo total do projeto não poderá exceder o valor estimado de R\$15.000,00

5.9 Análise de Riscos do Projeto

Risco	Descrição	Impacto	Medidas Preventivas
Risco 1 - Atraso no Desenvolvimento	As etapas de desenvolvimento podem demandar mais tempo que o previsto devido à complexidade técnica.	Comprometimento do prazo final do projeto.	Reuniões semanais de alinhamento, priorização de tarefas e distribuição equilibrada das atividades.
Risco 2 – Coleta de Requisitos Incompleta	Usuários podem ter dificuldade em descrever necessidades de forma clara.	Funcionalidades inadequadas ou incompletas.	Utilização de protótipos, entrevistas guiadas e validações periódicas com usuários-chave.
Risco 3 – Baixa Efetividade do Módulo Inteligente (IA)	A IA pode não gerar sugestões ou	Diminuição da eficiência	Implementar a IA em etapas, iniciando com

	classificações com qualidade satisfatória nas primeiras versões.	atendimento e necessidade de retrabalho.	regras simples e evoluindo conforme feedback real.
Risco 4 – Resistência à Adoção do Sistema	Alguns colaboradores podem preferir continuar usando métodos informais.	Baixa utilização do sistema e perda de benefícios.	Treinamentos, comunicação clara de vantagens e acompanhamento inicial pós-implantação.
Risco 5 – Limitações da Infraestrutura da Empresa	Rede ou equipamentos podem não suportar adequadamente o sistema.	Lentidão, instabilidade ou falhas de acesso.	Testes de carga, verificação prévia de requisitos e ajustes técnicos antes da implantação.

5.10 Equipe do Projeto

Arthur S. de Paula - Gerente de Projetos

Laís Silva Santos - Desenvolvedora Back-End

Igor Dantas Barros - Engenheiro de Inteligência Artificial

Renan Ribeiro - Desenvolvedor Front-End (Web)

Melyssa Souza Araujo - Analista de Base de Conhecimento e Dados

João Henrique S. dos Santos - Analista de Implantação e Suporte ao Usuário

William Gustavo - Analista de Testes / QA

O projeto do **Sistema Integrado de Gestão de Chamados e Suporte Técnico com Inteligência Artificial (IA)** permitiu a aplicação prática dos princípios de **Engenharia de Software e Gerenciamento de Projetos de Software**, resultando em uma solução eficiente e inovadora.

O sistema desenvolvido busca centralizar os atendimentos, automatizar processos repetitivos e melhorar a comunicação entre usuários e equipe técnica, reduzindo o tempo de resposta e aumentando a qualidade dos serviços prestados. A adoção de boas práticas de desenvolvimento, versionamento e conformidade com a **Lei Geral de Proteção de Dados (LGPD)** reforça a segurança e a confiabilidade do projeto.

De forma geral, o trabalho atingiu seus objetivos, apresentando uma proposta tecnológica moderna, escalável e alinhada às necessidades organizacionais. Como

continuidade, recomenda-se o aprimoramento do módulo de IA e a ampliação do sistema para atender múltiplos contextos de suporte técnico.

6. Gestão da Qualidade

A Gestão da Qualidade desempenha papel essencial no desenvolvimento de sistemas de software, pois garante que o produto final atenda aos requisitos técnicos e funcionais estabelecidos, além de promover a melhoria contínua durante todo o ciclo de desenvolvimento. No contexto deste projeto, as práticas de versionamento, testes e controle de qualidade foram adotadas de forma integrada, visando assegurar a confiabilidade e a consistência do sistema de gestão de chamados com apoio de inteligência artificial.

O versionamento do código-fonte foi realizado por meio de um sistema de controle de versões, como o Git, que permite o acompanhamento de todas as modificações realizadas pelos integrantes da equipe. Essa prática viabiliza um ambiente colaborativo seguro, onde cada desenvolvedor pode atuar em partes específicas do sistema sem comprometer o funcionamento geral. A utilização de branches distintas para novas funcionalidades, correções e melhorias facilitou a rastreabilidade das alterações e reduziu o risco de conflitos no momento da integração do código.

A etapa de testes foi conduzida com base em critérios previamente definidos, contemplando desde a verificação individual de componentes até a análise integrada do sistema como um todo. Foram aplicados testes unitários, responsáveis por validar o comportamento de métodos e classes em C#, e testes de integração, que avaliaram a comunicação entre os módulos — como as interfaces web, desktop, mobile e o banco de dados SQL Server. Também foram realizados testes funcionais e de usabilidade, que buscaram confirmar a aderência do sistema aos requisitos do usuário e sua facilidade de utilização. Todas as etapas de teste foram devidamente documentadas, registrando os dados de entrada, resultados esperados e resultados obtidos, possibilitando reavaliações futuras.

O controle de qualidade ocorreu de forma contínua, acompanhando o progresso do desenvolvimento por meio de checklists e indicadores de desempenho, como número de falhas identificadas, tempo médio de correção e índice de retrabalho. Esses indicadores contribuíram para o monitoramento da eficiência das entregas e para a adoção de ações preventivas ou corretivas sempre que necessário. Além disso, foram observados princípios de segurança da informação e conformidade com a Lei Geral de Proteção de Dados (LGPD), garantindo que o tratamento das informações fosse realizado de forma ética, transparente e segura.

A Tabela 1 apresenta um resumo das práticas de qualidade adotadas ao longo do desenvolvimento do sistema, destacando as ferramentas e os indicadores utilizados para monitorar o desempenho e a conformidade do projeto.

6.1 Tabela 1 – Práticas e indicadores da Gestão da Qualidade

Etapa	Descrição das Atividades	Ferramentas Utilizadas	Indicadores / Resultados Esperados
Versionamento de Código	Controle de alterações e histórico de versões do projeto, permitindo colaboração segura entre os desenvolvedores.	Git / GitHub	Histórico rastreável, redução de conflitos e maior controle das versões.
Testes Unitários	Validação de métodos e classes desenvolvidos em C#, garantindo o correto funcionamento individual dos componentes.	Visual Studio/ NUnit	Redução de erros em módulos isolados.
Testes de Integração	Avaliação da comunicação entre diferentes módulos do sistema (web, desktop, mobile e banco de dados).	Postman / SQL Server / Ambientes de teste	Integração consistente entre componentes.
Testes Funcionais e de Usabilidade	Verificação do cumprimento dos requisitos e análise da experiência do usuário.	Planilhas de verificação / Prototipagem	Sistema aderente às expectativas do usuário e de fácil uso.
Controle de Qualidade Contínuo	Monitoramento por meio de checklists e indicadores de desempenho durante todo o ciclo de desenvolvimento.	Planilhas / Ferramentas de acompanhamento	Redução do retrabalho e aumento da eficiência das entregas.
Segurança e Conformidade (LGPD)	Garantia de tratamento ético e seguro das informações, com foco na privacidade dos usuários.	Políticas internas / Padrões de segurança	Conformidade com a LGPD e maior confiança do usuário.

Dessa forma, a Gestão da Qualidade aplicada ao projeto não se limitou à correção de erros, mas atuou de forma proativa na prevenção de falhas e na busca pela excelência. O resultado foi um sistema mais estável, confiável e aderente às boas práticas de engenharia de software, refletindo o compromisso do grupo com o desenvolvimento de uma solução eficiente, sustentável e de alta qualidade.

7. Empreendedorismo

A aplicação dos conceitos de Empreendedorismo neste projeto permitiu uma análise abrangente dos aspectos de viabilidade, inovação e sustentabilidade, transformando o sistema de gestão de chamados com apoio de inteligência artificial em uma solução com potencial de mercado. A abordagem empreendedora proporciona uma visão estratégica que vai além do desenvolvimento técnico, explorando a capacidade do projeto gerar valor, atender demandas reais e se manter economicamente viável a longo prazo.

O sistema proposto foi idealizado para atender empresas de pequeno e médio porte, que frequentemente enfrentam dificuldades na organização e controle de solicitações internas de suporte técnico. A partir dessa necessidade, o projeto propõe uma solução que une eficiência operacional e tecnologia inteligente, utilizando automação e inteligência artificial para agilizar atendimentos, reduzir falhas humanas e otimizar a gestão de recursos. Essas características representam um diferencial competitivo, pois reduzem custos e aumentam a produtividade das organizações.

A viabilidade econômica é sustentada pela utilização de tecnologias consolidadas e acessíveis, como C#, ASP.NET e SQL Server, que permitem um desenvolvimento de baixo custo e alta compatibilidade. O modelo de negócio pode ser estruturado de forma flexível, adotando modalidades de licenciamento, assinatura mensal ou implantação sob demanda, de acordo com o perfil de cada cliente. Essa estrutura garante escalabilidade e possibilita a expansão gradual do produto, acompanhando as mudanças e exigências do mercado.

No que se refere à sustentabilidade, o projeto busca assegurar sua continuidade técnica e operacional ao longo do tempo, adotando práticas que favorecem o crescimento estruturado. O uso de versionamento de código, documentação detalhada e arquitetura modular garante a manutenção e evolução do sistema sem perda de desempenho. Além disso, há preocupação com o impacto social e ambiental, priorizando o uso eficiente de recursos tecnológicos e

promovendo o acesso a soluções digitais que aumentem a produtividade de pequenas empresas.

A Tabela 2 apresenta um resumo dos principais aspectos empreendedores aplicados no desenvolvimento do projeto, destacando os pilares de inovação, viabilidade, modelo de negócio e sustentabilidade.

7.1 Tabela 2 – Aspectos Empreendedores do Projeto

Aspecto	Descrição	Impacto no Projeto	Resultados Esperados
Inovação	Utilização de inteligência artificial e automação nos processos de suporte técnico.	Diferencial competitivo no mercado de tecnologia.	Atendimento mais rápido e eficiente, com redução de erros.
Viabilidade Econômica	Uso de tecnologias acessíveis (C#, ASP.NET, SQL Server) e estrutura modular.	Redução de custos de desenvolvimento e manutenção.	Produto de baixo custo e alta compatibilidade.
Modelo de Negócio	Possibilidade de licenciamento, assinatura mensal ou implantação sob demanda.	Flexibilidade para atender diferentes perfis de clientes.	Expansão escalável e aumento da base de usuários.
Sustentabilidade Técnica	Versionamento, documentação e arquitetura modular garantem manutenção contínua.	Maior durabilidade e evolução do sistema.	Continuidade operacional e atualização facilitada.
Responsabilidade Social e Ambiental	Uso eficiente de recursos tecnológicos e apoio a pequenas empresas.	Contribuição para inclusão digital e redução de desperdícios.	Impacto positivo na comunidade e nas práticas empresariais.

A partir de uma visão empreendedora, o projeto demonstra potencial real de inovação e aplicabilidade, combinando tecnologia, gestão e inteligência de dados para resolver um problema comum no ambiente corporativo. Essa perspectiva amplia o alcance do trabalho e reforça a importância do pensamento estratégico na criação de soluções sustentáveis, economicamente viáveis e socialmente relevantes.

8. Relações Étnico-Raciais e Afrodescendência

A abordagem das relações étnico-raciais e afrodescendência no desenvolvimento de sistemas é fundamental para promover um ambiente digital inclusivo, ético e socialmente responsável. O projeto foi concebido considerando a importância da diversidade, do respeito às diferenças e da valorização das identidades como elementos centrais na criação de soluções tecnológicas que atendam de forma igualitária a todos os usuários.

No design e na interface do sistema, buscou-se adotar uma linguagem neutra e acolhedora, livre de expressões que possam reproduzir estereótipos ou reforçar desigualdades. A plataforma foi planejada para permitir o uso de nome social, respeitando identidades individuais e garantindo que cada pessoa seja tratada de forma digna e personalizada. Essa atenção aos detalhes demonstra o compromisso em construir um ambiente digital que reconhece e respeita as múltiplas dimensões da diversidade humana.

A perspectiva de inclusão também se manifesta na experiência do usuário. O sistema foi estruturado de modo a garantir acessibilidade e equidade de uso, proporcionando a todos os colaboradores as mesmas condições de acesso e interação, independentemente de origem, cor, gênero ou condição social. Além disso, as mensagens automáticas e conteúdos informativos gerados com apoio de inteligência artificial seguem diretrizes éticas, evitando vieses discriminatórios e promovendo uma comunicação positiva, empática e respeitosa.

A Tabela 3 apresenta os principais princípios e práticas de inclusão étnico-racial adotados durante o desenvolvimento do projeto, evidenciando o compromisso da equipe com a equidade e a valorização da diversidade.

Tabela 3 – Princípios e Práticas de Inclusão Étnico-Racial no Projeto

Princípio / Valor	Prática Adotada no Projeto	Objetivo / Impacto Esperado
Diversidade e Representatividade	Inclusão de diretrizes de design e conteúdo que respeitam diferentes origens e identidades.	Garantir que o sistema atenda a públicos diversos, sem exclusões.
Linguagem Neutra e Acolhedora	Uso de termos inclusivos e eliminação de expressões que possam reproduzir estereótipos.	Promover uma comunicação ética e respeitosa entre usuários e sistema.
Uso de Nome Social	Implementação de campo específico para nome social no cadastro e exibição do usuário.	Respeitar a identidade individual de cada pessoa.
Acessibilidade e Equidade Digital	Estrutura do sistema voltada à inclusão de	Proporcionar igualdade de acesso e uso da

	todos os usuários, independentemente de suas condições sociais ou físicas.	plataforma.
Ética e Responsabilidade na IA	Configuração de respostas automáticas e mensagens com base em princípios éticos e sem vieses.	Evitar discriminações e reforçar o uso consciente da inteligência artificial.
Valorização Cultural e Social	Incentivo à reflexão sobre diversidade e respeito nas práticas tecnológicas.	Contribuir para um ambiente digital mais humano e plural.

Com base nos princípios de valorização da diversidade e combate ao preconceito, o projeto reforça o papel da tecnologia como ferramenta de transformação social. Desenvolver um sistema que incorpore esses valores significa reconhecer que a inclusão não é apenas uma exigência legal ou moral, mas um compromisso humano e profissional com a construção de espaços mais justos, igualitários e representativos.

Assim, ao integrar as relações étnico-raciais e a afrodescendência ao processo de desenvolvimento, o trabalho reafirma que a tecnologia deve refletir e respeitar a pluralidade da sociedade. O resultado é um sistema que, além de funcional e eficiente, se alinha a princípios de ética, respeito e responsabilidade social, contribuindo para um ambiente digital mais diverso e inclusivo.

9. Documentação do Back-end

O back-end do sistema **Suporte Técnico Integrado com IA** gerencia usuários, autenticação, chamados de suporte e integração com inteligência artificial (IA) para respostas automáticas. Desenvolvido em **ASP.NET Core 8**, utiliza **Entity Framework Core** e **SQL Server** via Docker. A autenticação é feita com **JWT**, garantindo segurança e controle de acesso.

Objetivo:

Permitir que usuários registrem-se, façam login, criem chamados de suporte, recebam respostas automáticas via IA e avaliem o atendimento.

9.1 Arquitetura do Sistema

O sistema segue a arquitetura **RESTful**:

[Front-end React]

|

v

[LoginController / TicketController - ASP.NET Core]

|

v

[Services] ----> [Google Gemini API]

|

v

[AppDbContext / SQL Server]

- **Controllers:** Recebem requisições HTTP e retornam respostas.
- **Services:** Executam regras de negócio e integração com IA.
- **DbContext:** Acessa o banco de dados SQL Server.
- **IA:** Gera respostas automáticas para os chamados.

9.2 Funcionalidades e Autenticação

Endpoint	Método	Descrição	Auth
/crie-uma-conta	POST	Cria novo usuário e retorna token	✗
/entrar	POST	Login do usuário e retorna token	✗

Fluxo de Registro e Login:

Usuário -> Front-end -> LoginController

LoginController -> AppDbContext: Verifica/Busca usuário

LoginController -> PasswordHasher: Criptografa/verifica senha

LoginController -> TokenService: Gera JWT

LoginController -> Front-end: Retorna token e dados do usuário

9.3 Gerenciamento de Chamados

Endpoint	Método	Descrição	Auth
/abrir-chamado	POST	Cria chamado e gera resposta automática IA	✓
/meus-chamados	GET	Lista todos os chamados do usuário	✓
/avaliar-	POST	Avalia um	✓

chamado/{id}		chamado respondido	
--------------	--	-----------------------	--

Fluxo de Criação de Chamado com IA:

Usuário -> Front-end -> TicketController

TicketController -> ApplicationDbContext: Salva ticket

TicketController -> AIService (Google Gemini): Gera resposta automática

AIService -> TicketController: Retorna resposta

TicketController -> ApplicationDbContext: Atualiza ticket com resposta e status "Respondido"

TicketController -> Front-end: Retorna ticket com resposta da IA

Integração com IA

- Serviço AIService envia descrição do ticket para **Google Gemini**.
- Recebe resposta automática que é salva no ticket.
- Garante respostas rápidas e consistentes aos chamados.

Execução do Projeto

Configurar chave do **Google Gemini** no appsettings.Development.json.

Executar com Docker:

```
docker-compose up --build -d
```

```
docker-compose exec app dotnet ef database update
```

Acessar Swagger: <http://localhost:8080/swagger/index.html>

Comandos úteis:

- Parar contêineres: docker-compose down
- Logs em tempo real: docker-compose logs -f app

Testes

- **Unitários:** Funções isoladas (hash, token, etc.)

- **Integração:** Fluxo completo (registro → login → abertura de chamado)
- **Segurança e Validação:** Confirma autenticação e regras de modelo

O back-end do sistema **Suporte Técnico Integrado com IA** foi desenvolvido com foco em **segurança, eficiência e escalabilidade**, garantindo que os usuários possam registrar-se, autenticar-se, abrir chamados de suporte e receber respostas automáticas de forma confiável e rápida.

A escolha pelo **ASP.NET Core 8** se deu pela robustez, suporte a aplicações RESTful e fácil integração com **Entity Framework Core** para persistência de dados no **SQL Server**. Essa combinação permitiu criar um sistema com **modelo relacional consistente**, garantindo integridade das informações de usuários e chamados.

O uso de **JWT (JSON Web Tokens)** para autenticação oferece **segurança e controle de acesso**, permitindo que apenas usuários autenticados realizem ações sensíveis, como abrir ou avaliar chamados.

A integração com **Google Gemini** foi escolhida para automatizar respostas de suporte técnico, reduzindo o tempo de atendimento e proporcionando uma experiência mais ágil para o usuário. Essa abordagem também permite que o sistema seja escalável, pois a inteligência artificial pode processar múltiplos chamados simultaneamente, sem sobrecarregar o servidor principal.

Além disso, a utilização de **Docker** para execução e configuração do projeto garante **portabilidade e facilidade de implantação**, permitindo que a aplicação seja executada em diferentes ambientes de forma rápida e confiável.

Em resumo, o back-end foi estruturado para ser **modular, seguro e eficiente**, com clara separação de responsabilidades entre controllers, services e camada de dados, permitindo manutenção simplificada, integração futura com outras funcionalidades e suporte contínuo à automação de processos via inteligência artificial.

10. Relatório Front-End

A avaliação abrange aspectos fundamentais como:

Interface do Usuário (UI)

Experiência do Usuário (UX)

Estética e design visual

Arquitetura da Informação

Organização das telas

Estrutura do projeto

Tecnologias utilizadas

Integração com o backend

O objetivo é demonstrar a qualidade da implementação, justificar decisões de design e evidenciar como o frontend contribui para a eficiência e acessibilidade do sistema.

10.1 Estética e Design Visual Paleta de Cores

- O design do ConnectWay utiliza uma paleta terrosa e neutra, escolhida para transmitir segurança, estabilidade e profissionalismo. Os principais tons são:
 - **Verde Escuro / Militar**
Usado em botões principais (Entrar, Criar Conta, Avaliar Atendimento). Transmite confiança e ação.
 - **Verde Oliva / Musgo Claro**
Presente nos campos de texto e botões secundários. Suaviza a experiência e diferencia áreas de input.
 - **Bege / Creme**
Cor predominante no fundo das páginas. Evita cansaço visual e destaca os elementos interativos.
 - Essa combinação resulta em um visual moderno, acessível e coerente com a proposta da marca.

10.2 Tipografia

A tipografia é clara e funcional. São utilizados pesos variados (normal, medium, bold) para:

- destacar títulos,
- guiar o olhar do usuário,
- separar blocos de conteúdo,
- reforçar hierarquia visual.

Os textos principais como “*Bem-vindo(a)!*”, “*Crie sua conta*”, “*Resposta da IA*” foram estilizados para facilitar identificação imediata da ação principal da tela.

10.3 Iconografia e Elementos Gráficos

Elementos visuais reforçam a comunicação com o usuário:

- Ícone da IA no chat: humaniza a automação
- Ícone de check verde: comunica sucesso rapidamente
- Estrelas de avaliação: padrão universal de feedback

O uso de ícones alinhados com o tom visual do projeto melhora a compreensão e a naveabilidade.

10.4 Consistência e Padrões de Design

O ConnectWay apresenta consistência entre telas e componentes:

- mesmos estilos de botões
- campos de entrada padronizados
- cores e espaçamentos uniformes

A consistência reduz curva de aprendizagem e aumenta a eficiência.

10.5 Login e Registro (UX)

Minimalista e direta, contendo apenas:

- e-mail
- senha
- botão Entrar
- link para Criar Conta

É objetiva e reduz ruídos durante o primeiro contato do usuário.

Tela	de	Registro
Formulário mais detalhado — inclui dados pessoais e de endereço.		

Mesmo sendo extenso, a organização linear facilita o preenchimento.

Tela de Suporte e Feedback

Resposta	da	IA
Um ponto forte do sistema.		

A tela apresenta:

- título descritivo
- blocos de texto estruturados
- bullets para instruções
- sugestões da IA baseadas no problema
- possibilidade de interação contínua

O UX foi pensado para fluidez e clareza.

10.6 Avaliação do Atendimento

Solicita:

- nota (1 a 5 estrelas)
- comentário opcional

Após o envio, a tela de agradecimento permite:

- abrir novo chamado
- sair do sistema

Esse fechamento fortalece a satisfação do usuário.

O website segue uma arquitetura linear e hierárquica clara, focada em tarefas:

A navegação segue uma lógica clara e hierárquica:

1. **Login / Registro** – entrada do usuário
2. **Meus Chamados** – painel principal
3. **Abrir Chamado**
4. **Resposta da IA**
5. **Avaliação**
6. **Agradecimento**

O uso da seta de voltar "←" aparece em várias telas, permitindo retorno rápido e previsível.

Telas do Aplicativo

O sistema conta com as seguintes telas:

- **LoginScreen**
- **RegisterScreen**
- **NovoChamadoScreen**
- **MeusChamadosScreen**
- **AvaliarChamadoScreen**
- **SucessoScreen**
- **AgradecimentoAvaliacaoScreen**

Cada tela foi projetada para ser objetiva, responsiva e fácil de usar.

10.7 Tecnologias Utilizadas no Frontend

O aplicativo foi desenvolvido com:

- **React Native**

- **Expo**
- **React Navigation**
- **Hooks e Componentes reutilizáveis**

Essas tecnologias foram escolhidas por:

- compatibilidade com Android / iOS
- rapidez no desenvolvimento
- fácil integração com o backend
- ecossistema robusto

Execução do Projeto

Pré-requisitos

- Node.js
- Expo CLI
- Emulador ou celular físico

O frontend do ConnectWay apresenta uma arquitetura bem organizada, experiência de usuário intuitiva e design visual coerente. A navegação é simples e lógica, apoiando um fluxo eficiente para abertura de chamados e interação com resposta inteligente da IA.

A combinação de React Native, Expo e boas práticas de UI/UX resulta em um sistema moderno e preparado para expansão. O conjunto das telas, a clara hierarquia visual e a integração com o backend fazem do ConnectWay uma solução funcional, acessível e adequada aos objetivos do projeto.

11. Telas das Páginas Web/Desktop

Página "Login" Web:

Login

The screenshot shows a light brown background with a white login form in the center. The form has a header "ConnectWay". It contains two input fields: "E-mail" and "Senha" (Password). Below these is a large green "Entrar" (Enter) button. Underneath the button is a small link "Criar uma conta" (Create account).

← Registrar

The screenshot shows a light brown background with a white registration form in the center. The form has a header "Crie sua conta". It contains several input fields: "Nome completo", "E-mail", "Senha", "Telefone", "Logradouro", "Número", "Complemento", "Bairro", "Cidade", "Estado", and "CEP". At the bottom of the form are two buttons: a dark blue "Criar Conta" (Create Account) button and a green "Voltar" (Back) button.

Página "Registrar" Web:

Página "Novo Chamado" Web:

← Novo Chamado

Novo Chamado
Preencha os detalhes abaixo:

Título
Ex: Problema no sistema

Categoria
Ex: Suporte Técnico

Descrição
Descreva o problema com detalhes

Enviar Chamado

Ver Meus Chamados

Página "Meus Chamados" Web:

← Meus Chamados

Bem-vindo(a)!
Seus chamados registrados:

Não consigo fazer login
Status: Aberto
Data: —

Avaliar Atendimento

Erro de conexão
Status: Aberto
Data: —

Avaliar Atendimento

Erro de login
Status: Aberto
Data: —

Avaliar Atendimento

Erro de login
Status: Aberto
Data: —

Avaliar Atendimento

Problema ao mudar senha
Status: Aberto
Data: —

Avaliar Atendimento

Página "Sucesso" (IA Respondendo ao Chamado Web):

[← Sucesso](#)

[Avaliar](#)



Resposta da IA

Olá! Entendo perfeitamente o quanto é frustrante ter a internet lenta. É algo que atrapalha muito o dia a dia, seja para trabalhar, estudar, se divertir ou se manter conectado. Não se preocupe, vamos juntos investigar o que pode estar acontecendo e encontrar uma solução.

Para te ajudar da melhor forma, e antes de tentarmos algumas soluções, preciso de algumas informações rápidas. Poderia me dizer:

1. **A lentidão acontece em todos os seus dispositivos** (computador, celular, tablet) ou apenas em um específico?
2. Você está conectado via **Wi-Fi ou cabo (Ethernet)**?
3. Isso começou **recentemente** ou já vem acontecendo há algum tempo? Há algum **horário específico** em que ela fica pior?
4. A lentidão ocorre ao acessar **qualquer site/aplicativo**, ou apenas em alguns específicos (ex: apenas em streaming de vídeo, ou em jogos online)?

Enquanto você pensa nessas respostas, podemos tentar algumas das soluções mais comuns que resolvem a maioria dos casos de lentidão na internet:

Passos Iniciais para Solução de Problemas:

1. **Reinicie seu roteador e modem:**
 - Desconecte os cabos de energia de ambos (roteador e modem, se forem aparelhos separados).
 - Espere cerca de **30 segundos**.
 - Conecte-os novamente, primeiro o modem e depois o roteador.
 - Aguarde alguns minutos até que todas as luzes se estabilizem.
 - **Por que fazer isso?** Isso limpa a memória dos aparelhos e força uma nova conexão com a sua provedora, resolvendo muitos problemas temporários.
2. **Teste em outro dispositivo:**
 - Se você estiver usando um computador, tente verificar a velocidade da internet em seu celular (conectado ao Wi-Fi). Se estiver rápida no celular, o problema pode ser específico do computador.
 - Se estiver usando o celular, tente em outro celular ou computador.
 - **Por que fazer isso?** Ajuda a identificar se o problema é na rede ou em um aparelho específico.
3. **Teste com cabo (se possível):**
 - Se você estiver conectado via Wi-Fi, tente conectar com o cabo diretamente ao roteador sem usar o hub de rede (Ethernet).

Página "Avaliar Chamado" Web:

[← Avaliar Chamado](#)

Avaliação do Chamado

Como foi seu atendimento?

Nota

★★★★★

Comentário

Deixe seu comentário (opcional)

Enviar Avaliação

Cancelar

Página "Agradecimento Avaliação" Web:



12. Telas das Páginas Moblie

Página "Login" Moblie:

20:44



Login



Página "Registrar" Mobile:

20:44



← Registrar

Crie sua conta

Nome completo

E-mail

Senha

Telefone

Logradouro

Número

Complemento

Bairro

Cidade

Estado

Página "Novo Chamado" Mobile:

20:46



← Meus Chamados

Bem-vindo(a)!

Seus chamados registrados:

Não consigo fazer login

Status: Aberto
Data: —

Avaliar Atendimento

Erro de conexão

Status: Aberto
Data: —

Avaliar Atendimento

Erro de login

Status: Aberto
Data: —

Avaliar Atendimento



Página "Sucesso" (IA Respondendo ao Chamado Mobile):

20:45



← Sucesso

Avaliar



Resposta da IA

Olá! Sinto muito que esteja com dificuldades para fazer login. Sei como é frustrante quando isso acontece, mas não se preocupe, vamos tentar resolver isso juntos.

Para começar, por favor, tente seguir os passos abaixo, que resolvem a maioria dos problemas de login:

1. **Verifique a Digitação e o Caps Lock:**
 - * Parece simples, mas é a causa mais comum! Verifique se digitou seu nome de usuário e senha corretamente.
 - * Confira se a tecla **Caps Lock** (Fixa) está desativada, pois a maioria das senhas diferencia maiúsculas de minúsculas.

< 1119481.hstgr.cloud ...

Página "Avaliar Chamado" Mobile:

20:45



← Avaliar Chamado

Avaliação do Chamado

Como foi seu atendimento?

Nota

★ ★ ★ ★ ★

Comentário

Deixe seu comentário (opcional)

Enviar Avaliação

Cancelar



Página "Agradecimento Avaliação" Mobile:



13. Códigos Back-end:

Controllers: LoginController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ApiCadastro.Data;
using ApiCadastro.Models;
using ApiCadastro.Service;

namespace ApiCadastro.Controllers
{
    [ApiController]
    public class LoginController : ControllerBase
```

```
{  
    private readonly AppDbContext _context;  
    private readonly ITokenService _tokenService;  
    private readonly IPasswordHasher _passwordHasher;  
  
    public LoginController(AppDbContext context, ITokenService tokenService,  
        IPasswordHasher passwordHasher)  
    {  
        _context = context;  
        _tokenService = tokenService;  
        _passwordHasher = passwordHasher;  
    }  
  
    [HttpPost("crie-uma-conta")]  
    public async Task<IActionResult> Register([FromBody] RegisterRequest request)  
    {  
        if (!ModelState.IsValid)  
            return BadRequest(ModelState);  
  
        var existingUser = await _context.Users.FirstOrDefaultAsync(u => u.Email ==  
            request.Email);  
        if (existingUser != null)  
            return BadRequest(new ErrorResponse { Message = "Este e-mail já está em uso." });  
  
        string passwordHash = _passwordHasher.HashPassword(request.Senha);  
  
        var newUser = new User  
        {  
            Email = request.Email,  
            PasswordHash = passwordHash,  
            Nome = request.Nome,  
            Telefone = request.Telefone,  
            Logradouro = request.Logradouro,  
            Numero = request.Numero,  
            Complemento = request.Complemento,  
            Bairro = request.Bairro,  
            Cidade = request.Cidade,  
            Estado = request.Estado,  
        };  
    }  
}
```

```

    CEP = request.CEP
};

_context.Users.Add(newUser);
await _context.SaveChangesAsync();

var token = _tokenService.GenerateToken(newUser);
var response = new AuthResponse
{
    ID = newUser.ID,
    Nome = newUser.Nome,
    Email = newUser.Email,
    Token = token
};

return Ok(response);
}

[HttpPost("entrar")]
public async Task<IActionResult> Login([FromBody] LoginRequest request)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var user = await _context.Users.FirstOrDefaultAsync(u => u.Email == request.Email);

    if (user == null)
        return Unauthorized(new ErrorResponse { Message = "E-mail ou senha inválidos." });

    bool isPasswordValid = _passwordHasher.VerifyPassword(request.Senha,
        user.PasswordHash);

    if (!isPasswordValid)
        return Unauthorized(new ErrorResponse { Message = "E-mail ou senha inválidos." });

    var token = _tokenService.GenerateToken(user);
    var response = new AuthResponse
    {

```

```

ID = user.ID,
Nome = user.Nome,
Email = user.Email,
Token = token
};

return Ok(response);
}
}
}

```

TicketController.cs:

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using ApiCadastro.Data;
using ApiCadastro.Models;
using ApiCadastro.Service;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;

namespace ApiCadastro.Controllers
{
    [ApiController]
    public class TicketController : ControllerBase
    {
        private readonly AppDbContext _context;
        private readonly IAIService _aiService;

        public TicketController(AppDbContext context, IAIService aiService)
        {
            _context = context;
            _aiService = aiService;
        }

        // Criar novo ticket
        [Authorize]
        [HttpPost("abrir-chamado")]

```

```

public async Task<IActionResult> CriarChamado([FromBody] TicketRequest
request)
{
if (!ModelState.IsValid)
return BadRequest(ModelState);

var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

var ticket = new Ticket
{
Titulo = request.Titulo,
Descricao = request.Descricao,
UserId = int.Parse(userId ?? "0"),
DataCriacao = DateTime.UtcNow
};

_context.Tickets.Add(ticket);
await _context.SaveChangesAsync();

// Gera resposta automática via IA
var respostalA = await _aiService.GenerateContentAsync(request.Descricao);
ticket.RespostaIA = respostalA;
ticket.Status = "Respondido"; // Atualiza o status do chamado para respondido
semempre que a IA responder
Console.WriteLine($"Ticket Status before second save: {ticket.Status}");
await _context.SaveChangesAsync();

return Ok(new CriarChamadoResponse
{
Message = "Chamado criado com sucesso!",
Ticket = ticket
});

// Listar tickets do usuário autenticado
[Authorize]
[HttpGet("meus-chamados")]
public async Task<IActionResult> MeusChamados()

```

```

{
var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
var tickets = await _context.Tickets
    .Where(t => t.UserId.ToString() == userId)
    .ToListAsync();

return Ok(tickets);
}

[Authorize]
[HttpPost("avaliar-chamado/{id}")]
public async Task<IActionResult> AvaliarChamado(int id, [FromBody]
TicketAvaliacaoRequest request)
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

    var ticket = await _context.Tickets.FindAsync(id);
    if (ticket == null)
        return NotFound(new ErrorResponse { Message = "Chamado não encontrado." });

    if (ticket.UserId.ToString() != userId)
        return Forbid("Você só pode avaliar seus próprios chamados.");

    if (ticket.Status == "Aberto")
        return BadRequest(new ErrorResponse { Message = "O chamado ainda não foi
respondido e não pode ser avaliado." });

    ticket.NotaAvaliacao = request.Nota;
    ticket.ComentarioAvaliacao = request.Comentario;
    await _context.SaveChangesAsync();

    return Ok(new AvaliarChamadoResponse
    {
        Message = "Avaliação registrada com sucesso!",
        Id = ticket.Id,
        NotaAvaliacao = ticket.NotaAvaliacao,
        ComentarioAvaliacao = ticket.ComentarioAvaliacao
    });
}

```

```
}
```

```
}
```

```
}
```

Data: AppDbContext.cs:

```
using ApiCadastro.Models;
using Microsoft.EntityFrameworkCore;

namespace ApiCadastro.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options)
            : base(options)
        {
        }

        public DbSet<User> Users { get; set; } = null!;
        public DbSet<Ticket> Tickets { get; set; } = null!; // Coleção de Tickets

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            // Garante que o campo Email é único
            modelBuilder.Entity<User>()
                .HasIndex(u => u.Email)
                .IsUnique();
        }
    }
}
```

Models: AuthResponse.cs:

```
using System;

namespace ApiCadastro.Models
{
    public class AuthResponse
    {
        public int ID { get; set; } // Corresponde ao User.ID
        public string Email { get; set; } = string.Empty;
        public string Nome { get; set; } = string.Empty;
        public string Token { get; set; } = string.Empty;
    }
}
```

AvaliarChamadoResponse.cs:

```
namespace ApiCadastro.Models
{
    public class AvaliarChamadoResponse
    {
        public string Message { get; set; }
        public int Id { get; set; }
        public int? NotaAvaliacao { get; set; }
        public string ComentarioAvaliacao { get; set; }
    }
}
```

CriarChamadoResponse.cs:

```
namespace ApiCadastro.Models
{
    public class CriarChamadoResponse
    {
        public string Message { get; set; }
        public Ticket Ticket { get; set; }
    }
}
```

ErrorResponse.cs:

```
namespace ApiCadastro.Models
{
    public class ErrorResponse
    {
        public string Message { get; set; }
    }
}
```

LoginRequest.cs:

```
using System.ComponentModel.DataAnnotations;

namespace ApiCadastro.Models
{
    public class LoginRequest
    {
        [Required]
        [EmailAddress]
        public string Email { get; set; } = string.Empty;

        [Required]
        public string Senha { get; set; } = string.Empty;
    }
}
```

RegisterRequest.cs:

```
using System.ComponentModel.DataAnnotations;

namespace ApiCadastro.Models
{
    public class RegisterRequest
    {
        [Required]
        public string Email { get; set; } = string.Empty;
    }
}
```

```

[Required]
public string Senha { get; set; } = string.Empty; // Campo usado no Controller

[Required]
public string Nome { get; set; } = string.Empty;
public string Telefone { get; set; } = string.Empty;

// Campos de Endereço
public string Logradouro { get; set; } = string.Empty;
public string Numero { get; set; } = string.Empty;
public string Complemento { get; set; } = string.Empty;
public string Bairro { get; set; } = string.Empty;
public string Cidade { get; set; } = string.Empty;
public string Estado { get; set; } = string.Empty;

[Required]
public string CEP { get; set; } = string.Empty;
}
}

```

Ticket.cs:

```

using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ApiCadastro.Models
{
    public class Ticket
    {
        [Key]
        public int Id { get; set; }

        [ForeignKey("User")]
        public int UserId { get; set; }

        [Required]
        public string Titulo { get; set; } = string.Empty;
    }
}

```

```
[Required]
public string Descricao { get; set; } = string.Empty;

public string RespostaA { get; set; } = string.Empty;

public string Status { get; set; } = "Aberto";

public DateTime DataCriacao { get; set; }

public int? NotaAvaliacao { get; set; }

public string ComentarioAvaliacao { get; set; } = string.Empty;

public User? User { get; set; }

}
```

TicketAvaliacaoRequest.cs:

```
using System.ComponentModel.DataAnnotations;

namespace ApiCadastro.Models
{
    public class TicketAvaliacaoRequest
    {
        [Range(1, 5, ErrorMessage = "A nota deve ser entre 1 e 5.")]
        public int Nota { get; set; }

        public string Comentario { get; set; } = string.Empty;
    }
}
```

TicketRequest.cs:

```
using System.ComponentModel.DataAnnotations;
```

```
namespace ApiCadastro.Models
{
    public class TicketRequest
    {
        [Required]
        public string Titulo { get; set; } = string.Empty;

        [Required]
        public string Descricao { get; set; } = string.Empty;
    }
}
```

User.cs:

```
using System;

namespace ApiCadastro.Models
{
    public class User
    {
        public int ID { get; set; } // PK

        public string Email { get; set; } = string.Empty; // Único
        public string PasswordHash { get; set; } = string.Empty; // Hash da senha
        public string Nome { get; set; } = string.Empty;
        public string Telefone { get; set; } = string.Empty;

        // Campos de Endereço
        public string Logradouro { get; set; } = string.Empty;
        public string Numero { get; set; } = string.Empty;
        public string Complemento { get; set; } = string.Empty;
        public string Bairro { get; set; } = string.Empty;
        public string Cidade { get; set; } = string.Empty;
        public string Estado { get; set; } = string.Empty;
        public string CEP { get; set; } = string.Empty;
    }
}
```

Service: AIService.cs:

```
using System.Net.Http;
using System.Net.Http.Json;
using System.Text.Json;
using Microsoft.Extensions.Configuration;
using System;
using System.Threading.Tasks;

namespace ApiCadastro.Service
{
    public class AIService : IAIService
    {
        private readonly string _apiKey;
        private readonly HttpClient _httpClient;

        public AIService(IConfiguration configuration)
        {
            // Obtém a chave da API Gemini do appsettings.json
            _apiKey = configuration["Gemini:ApiKey"]
            ?? throw new InvalidOperationException("Gemini:ApiKey não configurada. Verifique appsettings.json ou appsettings.Development.json.");
        }

        _httpClient = new HttpClient();
    }

    public async Task<string> GenerateContentAsync(string prompt)
    {
        // Endpoint do modelo Gemini 2.5 Flash
        var url = $"https://generativelanguage.googleapis.com/v1/models/gemini-2.5-
flash:generateContent?key={_apiKey}";

        var fullPrompt = $"Você é um assistente técnico. Gere uma resposta clara, empática e útil para o usuário com base no problema: '{prompt}'";

        var requestBody = new
        {
            contents = new[]

```

```

{
new
{
parts = new[]
{
new { text = fullPrompt }
}
}
};

try
{
var response = await _httpClient.PostAsJsonAsync(url, requestBody);
var jsonResponse = await response.Content.ReadAsStringAsync();

if (!response.IsSuccessStatusCode)
{
// Retorna o erro completo para facilitar o debug
return $"Erro ao se comunicar com o Gemini ({response.StatusCode}):
{jsonResponse}";
}

using var doc = JsonDocument.Parse(jsonResponse);
var text = doc.RootElement
.GetProperty("candidates")[0]
.GetProperty("content")
.GetProperty("parts")[0]
.GetProperty("text")
.GetString();

return text ?? "A IA não retornou uma resposta.";
}
catch (Exception ex)
{
// Captura erros de rede ou de parsing
return $"Erro ao se comunicar com o Gemini: {ex.Message}";
}

```

```
}
```

```
}
```

```
}
```

IAIService.cs:

```
using System.Threading.Tasks;

namespace ApiCadastro.Service
{
    public interface IAIService
    {
        Task<string> GenerateContentAsync(string prompt);
    }
}
```

IPasswordHasher.cs:

```
using System.Security.Cryptography;

namespace ApiCadastro.Service
{
    public interface IPasswordHasher
    {
        string HashPassword(string password);
        bool VerifyPassword(string password, string hashedPassword);
    }
}
```

ITokenService.cs:

```
using ApiCadastro.Models;
using System.Threading.Tasks;

namespace ApiCadastro.Service
{
```

```
public interface ITokenService
{
    string GenerateToken(User user);
}
```

PasswordHasher.cs:

```
using System.Security.Cryptography;

namespace ApiCadastro.Service
{
    public class PasswordHasher : IPasswordHasher
    {
        private const int SaltSize = 16;
        private const int HashSize = 20;
        private const int Iterations = 10000;

        public string HashPassword(string password)
        {
            // Gera o salt aleatório
            using (var rng = RandomNumberGenerator.Create())
            {
                byte[] salt = new byte[SaltSize];
                rng.GetBytes(salt);

                // Gera o hash usando PBKDF2 com SHA256
                var pbkdf2 = new Rfc2898DeriveBytes(password, salt, Iterations,
                    HashAlgorithmName.SHA256);
                byte[] hash = pbkdf2.GetBytes(HashSize);

                // Combina salt e hash
                byte[] hashBytes = new byte[SaltSize + HashSize];
                Array.Copy(salt, 0, hashBytes, 0, SaltSize);
                Array.Copy(hash, 0, hashBytes, SaltSize, HashSize);

                // Converte para Base64 para armazenamento
                return Convert.ToString(hashBytes);
            }
        }
    }
}
```

```
}

public bool VerifyPassword(string password, string hashedPassword)
{
    // Converte Base64 de volta para bytes
    byte[] hashBytes = Convert.FromBase64String(hashedPassword);
    // Extrai o salt
    byte[] salt = new byte[SaltSize];
    Array.Copy(hashBytes, 0, salt, 0, SaltSize);

    // Calcula o hash da senha de entrada
    var pbkdf2 = new Rfc2898DeriveBytes(password, salt, Iterations,
        HashAlgorithmName.SHA256);
    byte[] hash = pbkdf2.GetBytes(HashSize);

    // Compara os hashes
    for (int i = 0; i < HashSize; i++)
    {
        if (hashBytes[i + SaltSize] != hash[i])
        {
            return false;
        }
    }
    return true;
}
```

TokenService.cs:

```
using System.IdentityModel.Tokens.Jwt;  
using System.Security.Claims;  
using System.Text;  
using ApiCadastro.Models;  
using Microsoft.IdentityModel.Tokens;  
  
namespace ApiCadastro.Service
```

```

{
public class TokenService : ITokenService
{
private readonly IConfiguration _configuration;

public TokenService(IConfiguration configuration)
{
    _configuration = configuration;
}

public string GenerateToken(User user)
{
    var tokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(
        _configuration["Jwt:Key"] ?? throw new InvalidOperationException("Jwt:Key not
configured."));
);

var tokenDescriptor = new SecurityTokenDescriptor
{
    Subject = new ClaimsIdentity(new[]
    {
        new Claim(ClaimTypes.NameIdentifier, user.ID.ToString()),
        new Claim(ClaimTypes.Email, user.Email)
    }),
    Expires = DateTime.UtcNow.AddMinutes(30),
    Issuer = _configuration["Jwt:Issuer"],
    Audience = _configuration["Jwt:Audience"],

    SigningCredentials = new SigningCredentials(
        new SymmetricSecurityKey(key),
        SecurityAlgorithms.HmacSha256Signature
    )
};

var token = tokenHandler.CreateToken(tokenDescriptor);
return tokenHandler.WriteToken(token);
}

```

```
}
```

```
}
```

```
}
```

Program.cs:

```
using ApiCadastro.Data;
using ApiCadastro.Service;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

const string myAllowSpecificOrigins = "_myAllowSpecificOrigins";
builder.Services.AddCors(options =>
{
    options.AddPolicy(name: myAllowSpecificOrigins,
    policy =>
    {
        policy.WithOrigins("http://localhost:8081", "http://srv1119481.hstqr.cloud")
        .AllowAnyHeader()
        .AllowAnyMethod();
    });
});

// Configuração do Banco de Dados
var connectionstring = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<AppDbContext>(options =>
    options.UseSqlServer(connectionstring));

// Adicionar Serviços
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
```

```

// Configuração do Token JWT
var jwtKey = builder.Configuration["Jwt:Key"] ?? throw new
InvalidOperationException("Jwt:Key not configured.");
var key = Encoding.ASCII.GetBytes(jwtKey);

builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.RequireHttpsMetadata = false;
    options.SaveToken = true;
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = new SymmetricSecurityKey(key),
        ValidateIssuer = true,
        ValidIssuer = builder.Configuration["Jwt:Issuer"],
        ValidateAudience = true,
        ValidAudience = builder.Configuration["Jwt:Audience"],
        ValidateLifetime = true,
        ClockSkew = TimeSpan.Zero
    };
});

// Injeção de dependências de serviços
builder.Services.AddScoped<ITokenService, TokenService>();
builder.Services.AddScoped<IAIService, AIService>();
builder.Services.AddScoped<IPasswordHasher, PasswordHasher>();

// Configuração do Swagger/OpenAPI
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "API Cadastro Suporte", Version =
    "v1" });
}

```

```

// Esquema de segurança para JWT
c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
{
    Description = "Insira o token JWT no formato: Bearer {seu_token_aqui}",
    Name = "Authorization",
    In = ParameterLocation.Header,
    Type = SecuritySchemeType.ApiKey,
    Scheme = "Bearer"
});

c.AddSecurityRequirement(new OpenApiSecurityRequirement
{
    {
        new OpenApiSecurityScheme
        {
            Reference = new OpenApiReference
            {
                Type = ReferenceType.SecurityScheme,
                Id = "Bearer"
            }
        },
        new string[] {}
    }
});
});

var app = builder.Build();

// Pipeline de requisição
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseCors(myAllowSpecificOrigins);

```

```

app.UseAuthentication();
app.UseAuthorization();

app.MapControllers();

app.Run();

public partial class Program { }

```

ApiCadastroTests: CustomWebApplicationFactory.cs:

```

using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc.Testing;
using Microsoft.AspNetCore.TestHost;
using Microsoft.Extensions.Hosting;
using System.IO;
using System.Reflection;

public class CustomWebApplicationFactory : WebApplicationFactory<Program>
{
    protected override void ConfigureWebHost(IWebHostBuilder builder)
    {
        // Define o diretório raiz do projeto principal para localizar arquivos como
        // appsettings.json
        builder.UseContentRoot(FindProjectRootPath("ApiCadastro"));

        builder.ConfigureServices(services =>
        {
        });
    }

    /// <summary>
    /// Localiza o diretório do projeto principal (ex.: ApiCadastro)
    /// a partir da estrutura de diretórios da solução.
    /// </summary>
    private static string FindProjectRootPath(string targetProjectName)
    {

```

```

var testAssemblyPath = Assembly.GetExecutingAssembly().Location;
var DirectoryInfo = new DirectoryInfo(Path.GetDirectoryName(testAssemblyPath)!);

// Sobe os diretórios até encontrar a raiz da solução (SuporteTecnico)
while (directoryInfo != null && directoryInfo.Name != "SuporteTecnico")
{
    directoryInfo = directoryInfo.Parent;
}

if (directoryInfo == null)
{
    throw new DirectoryNotFoundException($"Não foi possível localizar a raiz da solução 'SuporteTecnico'. Caminho atual: {testAssemblyPath}");
}

var appProjectDir = Path.Combine(directoryInfo.FullName, targetProjectName);

if (Directory.Exists(appProjectDir))
{
    return appProjectDir;
}

throw new DirectoryNotFoundException($"Não foi possível localizar o diretório do projeto: {appProjectDir}");
}
}

```

LoginControllerIntegrationTests.cs:

```

using System;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc.Testing;
using Xunit;

```

```

using ApiCadastro.Models;

namespace ApiCadastro.Tests
{
    public class LoginControllerIntegrationTests : IClassFixture<CustomWebApplicationFactory>
    {
        private readonly HttpClient _client;
        private readonly CustomWebApplicationFactory _factory;

        public LoginControllerIntegrationTests(CustomWebApplicationFactory factory)
        {
            _factory = factory;
            _client = factory.CreateClient();
        }

        // Gera um usuário com e-mail único para evitar conflitos entre testes
        private (string email, string password) GenerateUniqueUser()
        {
            var uniqueEmail = $"test-{Guid.NewGuid()}@api-cadastro.com";
            var password = "Password123!";
            return (uniqueEmail, password);
        }

        [Fact]
        public async Task Register_ShouldReturnOk_WhenUserIsCreated()
        {
            // Arrange
            var (email, password) = GenerateUniqueUser();
            var request = new RegisterRequest
            {
                Email = email,
                Senha = password,
                Nome = "Test Register",
                CEP = "00000-000"
            };

            var content = new StringContent(

```

```

JsonSerializer.Serialize(request),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json")
);

// Act
var response = await _client.PostAsync("/crie-uma-conta", content);

// Assert
response.EnsureSuccessStatusCode();
}

[Fact]
public async Task Login_ShouldReturnOk_WhenCredentialsAreValid()
{
// Arrange
using var client = _factory.CreateClient();
var (email, password) = GenerateUniqueUser();

// 1. Registro
var registerRequest = new RegisterRequest
{
Email = email,
Senha = password,
Nome = "Test Login",
CEP = "00000-000"
};

var registerContent = new StringContent(
JsonSerializer.Serialize(registerRequest),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json")
);

var registerResponse = await client.PostAsync("/crie-uma-conta", registerContent);
registerResponse.EnsureSuccessStatusCode();

// 2. Login
}

```

```

var loginRequest = new LoginRequest { Email = email, Senha = password };

var loginContent = new StringContent(
JsonSerializer.Serialize(loginRequest),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json")
);

// Act
var response = await client.PostAsync("/entrar", loginContent);

// Assert
response.EnsureSuccessStatusCode();

var responseString = await response.Content.ReadAsStringAsync();
var authResponse = JsonSerializer.Deserialize<AuthResponse>(
responseString,
new JsonSerializerOptions { PropertyNameCaseInsensitive = true }
);

Assert.NotNull(authResponse);
Assert.False(string.IsNullOrEmpty(authResponse.Token));
}

[Fact]
public async Task Login_ShouldReturnUnauthorized_WhenCredentialsAreInvalid()
{
// Arrange
var loginRequest = new LoginRequest
{
Email = "nonexistent@user.com",
Senha = "wrongpassword"
};

var loginContent = new StringContent(
JsonSerializer.Serialize(loginRequest),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json")
}

```

```
});  
  
// Act  
var response = await _client.PostAsync("/entrar", loginContent);  
  
// Assert  
Assert.Equal(HttpStatusCode.Unauthorized, response.StatusCode);  
}  
}  
}
```

LoginControllerTests.cs:

```
using Xunit;
using Moq;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Threading.Tasks;
using System;
using ApiCadastro.Controllers;
using ApiCadastro.Data;
using ApiCadastro.Models;
using ApiCadastro.Service;

namespace ApiCadastro.Tests
{
    public class LoginControllerTests : IDisposable
    {
        private readonly Mock<ITokenService> _tokenServiceMock;
        private readonly Mock<IPasswordHasher> _passwordHasherMock;
        private readonly ApplicationDbContext _dbContext;

        public LoginControllerTests()
        {
            _tokenServiceMock = new Mock<ITokenService>();
            _passwordHasherMock = new Mock<IPasswordHasher>();
            var dbContextOptions = new DbContextOptionsBuilder<ApplicationContext>()
                .UseInMemoryDatabase(databaseName: Guid.NewGuid().ToString())
        }
    }
}
```

```

.Options;
_dbContext = new AppDbContext(dbContextOptions);
}

public void Dispose()
{
    _dbContext.Database.EnsureDeleted();
    _dbContext.Dispose();
}

[Fact]
public async Task Register_ShouldReturnOk_WhenUserIsCreated()
{
    // Arrange
    var controller = new LoginController(_dbContext, _tokenServiceMock.Object,
    _passwordHasherMock.Object);
    // Os DTOs agora são resolvidos para ApiCadastro.Models.RegisterRequest
    var request = new RegisterRequest { Email = "test@example.com", Senha =
    "password", Nome = "Test User", CEP = "00000-000" };

    _passwordHasherMock.Setup(p =>
    p.HashPassword(request.Senha)).Returns("hashed_password");
    _tokenServiceMock.Setup(t =>
    t.GenerateToken(It.IsAny<User>())).Returns("test_token");

    // Act
    var result = await controller.Register(request);

    // Assert
    var okResult = Assert.IsType<OkObjectResult>(result);
    // AuthResponse está definido em TestModels.cs, no namespace ApiCadastro.Tests
    var authResponse = Assert.IsType<ApiCadastro.Models.AuthResponse>(okResult.Value);
    Assert.Equal("test_token", authResponse.Token);
}

[Fact]
public async Task Register_ShouldReturnBadRequest_WhenEmailExists()

```

```

{
// Arrange
var existingUser = new User { Email = "test@example.com", PasswordHash =
"hashed_password", Nome = "Existing", CEP = "00000-000" };
_dbContext.Users.Add(existingUser);
await _dbContext.SaveChangesAsync();

var controller = new LoginController(_dbContext, _tokenServiceMock.Object,
_passwordHasherMock.Object);
var request = new RegisterRequest { Email = "test@example.com", Senha =
"password", Nome = "Test User", CEP = "00000-000" };

// Act
var result = await controller.Register(request);

// Assert
var badRequestResult = Assert.IsType<BadRequestObjectResult>(result);
// ErrorResponse é um modelo auxiliar de teste.
var error = Assert.IsType<ApiCadastro.Models.ErrorResponse>(badRequestResult.Value);
Assert.Equal("Este e-mail já está em uso.", error.Message);
}

[Fact]
public async Task Login_ShouldReturnOk_WhenCredentialsAreValid()
{
// Arrange
var user = new User { Email = "test@example.com", PasswordHash =
"hashed_password", Nome = "Existing", CEP = "00000-000" };
_dbContext.Users.Add(user);
await _dbContext.SaveChangesAsync();

var controller = new LoginController(_dbContext, _tokenServiceMock.Object,
_passwordHasherMock.Object);
var request = new LoginRequest { Email = "test@example.com", Senha =
"password" };

```

```

_passwordHasherMock.Setup(p      => p.VerifyPassword(request.Senha,
user.PasswordHash)).Returns(true);
_tokenServiceMock.Setup(t => t.GenerateToken(user)).Returns("test_token");

// Act
var result = await controller.Login(request);

// Assert
var okResult = Assert.IsType<OkObjectResult>(result);
var authResponse = Assert.IsType<ApiCadastro.Models.AuthResponse>(okResult.Value);
Assert.Equal("test_token", authResponse.Token);
}

[Fact]
public async Task Login_ShouldReturnUnauthorized_WhenUserDoesNotExist()
{
// Arrange
var controller = new LoginController(_dbContext, _tokenServiceMock.Object,
_passwordHasherMock.Object);
var request = new LoginRequest { Email = "test@example.com", Senha =
"password" };

// Act
var result = await controller.Login(request);

// Assert
var unauthorizedResult = Assert.IsType<UnauthorizedObjectResult>(result);
var error = Assert.IsType<ApiCadastro.Models.ErrorResponse>(unauthorizedResult.Value);
Assert.Equal("E-mail ou senha inválidos.", error.Message);
}

[Fact]
public async Task Login_ShouldReturnUnauthorized_WhenPasswordIsInvalid()
{
// Arrange

```

```

var user = new User { Email = "test@example.com", PasswordHash =
"hashed_password", Nome = "Existing", CEP = "00000-000" };
_dbContext.Users.Add(user);
await _dbContext.SaveChangesAsync();

var controller = new LoginController(_dbContext, _tokenServiceMock.Object,
_passwordHasherMock.Object);
var request = new LoginRequest { Email = "test@example.com", Senha =
"wrong_password" };

_passwordHasherMock.Setup(p => p.VerifyPassword(request.Senha,
user.PasswordHash)).Returns(false);

// Act
var result = await controller.Login(request);

// Assert
var unauthorizedResult = Assert.IsType<UnauthorizedObjectResult>(result);
var error = unauthorizedResult.Value;
Assert.IsType<ApiCadastro.Models.ErrorResponse>(error);
Assert.Equal("E-mail ou senha inválidos.", error.Message);
}
}
}

```

SecurityTests.cs:

```

using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc.Testing;
using Xunit;
using ApiCadastro.Models;

namespace ApiCadastro.Tests

```

```

{
public class SecurityTests : IClassFixture<CustomWebApplicationFactory>
{
private readonly HttpClient _client;
private readonly CustomWebApplicationFactory _factory;

public SecurityTests(CustomWebApplicationFactory factory)
{
    _factory = factory;
    _client = factory.CreateClient();
}

// Gera um token JWT válido simulando o fluxo real de registro e login
private async Task<string> GetValidToken()
{
    using var client = _factory.CreateClient();
    var uniqueEmail = $"security-test-{System.Guid.NewGuid()}@api-cadastro.com";
    var password = "Password123!";

    // 1. Registro de usuário
    var registerRequest = new RegisterRequest
    {
        Email = uniqueEmail,
        Senha = password,
        Nome = "Security Test User",
        CEP = "00000-000"
    };

    var registerContent = new StringContent(
        JsonSerializer.Serialize(registerRequest),
        Encoding.UTF8,
        new MediaTypeHeaderValue("application/json")
    );

    await client.PostAsync("/crie-uma-conta", registerContent);

    // 2. Login para obter o token JWT
    var loginRequest = new LoginRequest { Email = uniqueEmail, Senha = password };
}
}

```

```

var loginContent = new StringContent(
JsonSerializer.Serialize(loginRequest),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json")
);

var response = await client.PostAsync("/entrar", loginContent);
response.EnsureSuccessStatusCode();

var responseString = await response.Content.ReadAsStringAsync();
var authResponse = JsonSerializer.Deserialize<AuthResponse>(
responseString,
new JsonSerializerOptions { PropertyNameCaseInsensitive = true }
);

return authResponse?.Token ?? string.Empty;
}

[Theory]
[InlineData("/abrir-chamado")] // Endpoint protegido (POST)
[InlineData("/meus-chamados")] // Endpoint protegido (GET)
public async Task SecuredEndpoints_ShouldReturnUnauthorized_WhenNoTokenIsProvided(string url)
{
    // Arrange
    var request = new TicketRequest { Titulo = "Título", Descricao = "Descrição" };

    var content = new StringContent(
JsonSerializer.Serialize(request),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json")
);

    // Act
    HttpResponseMessage response = url.StartsWith("/abrir-chamado")
? await _client.PostAsync(url, content)
: await _client.GetAsync(url);
}

```

```

// Assert
Assert.Equal(HttpStatusCode.Unauthorized, response.StatusCode);
}

[Fact]
public async Task SecuredEndpoint_ShouldReturnOk_WhenValidTokenIsProvided()
{
    // Arrange
    var token = await GetValidToken();
    _client.DefaultRequestHeaders.Authorization =
        new AuthenticationHeaderValue("Bearer", token);

    var request = new TicketRequest
    {
        Titulo = "Chamado Autorizado",
        Descricao = "Teste de autorização OK."
    };

    var content = new StringContent(
        JsonSerializer.Serialize(request),
        Encoding.UTF8,
        new MediaTypeHeaderValue("application/json")
    );

    // Act
    var response = await _client.PostAsync("/abrir-chamado", content);

    // Assert
    Assert.True(
        response.IsSuccessStatusCode,
        $"Esperado sucesso, mas recebeu {response.StatusCode}. Resposta: {await
        response.Content.ReadAsStringAsync()}"
    );
}

```

TestModels.cs:

```
using ApiCadastro.Models;

namespace ApiCadastro.Tests
{
    // Modelo simplificado para deserializar respostas de autenticação nos testes
    public class AuthResponse
    {
        public string Token { get; set; } = string.Empty;
        public string UserId { get; set; } = string.Empty;
    }

    // Modelo usado para validar a resposta de criação de chamados nos testes
    public class CriarChamadoResponse
    {
        public Ticket Ticket { get; set; } = new Ticket();
        public string Message { get; set; } = string.Empty;
    }
}
```

TestStartup.cs:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using ApiCadastro.Data;
using ApiCadastro.Service;
using Moq;
using System.Linq;

namespace ApiCadastro.Tests
{
    // Classe de inicialização específica para o ambiente de testes de integração
    public class TestStartup
    {
```

```
public TestStartup(IConfiguration configuration)
{
    Configuration = configuration;
}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)
{
    // Substitui o DbContext real por um em memória
    var dbContextDescriptor = services.SingleOrDefault(
        d => d.ServiceType == typeof(DbContextOptions<AppDbContext>));

    if (dbContextDescriptor != null)
        services.Remove(dbContextDescriptor);

    services.AddDbContext<AppDbContext>(options =>
        options.UseInMemoryDatabase("InMemoryDbForTesting"));

    // Registra mocks de serviços externos
    services.AddSingleton(new Mock<IAIService>().Object);
    services.AddSingleton(new Mock<IPasswordHasher>().Object);

    // Inclui os controladores da API principal
    services.AddControllers().AddApplicationPart(typeof(Program).Assembly);
}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseRouting();
    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

```
}
```

```
}
```

TicketControllerIntegrationTests.cs:

```
using System;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using ApiCadastro.Models;
using Microsoft.AspNetCore.Mvc.Testing;
using Xunit;

namespace ApiCadastro.Tests
{
    public class TicketControllerIntegrationTests : IClassFixture<CustomWebApplicationFactory>
    {
        private readonly HttpClient _client;
        private readonly CustomWebApplicationFactory _factory;

        public TicketControllerIntegrationTests(CustomWebApplicationFactory factory)
        {
            _factory = factory;
            _client = factory.CreateClient();
        }

        private (string email, string password) GenerateUniqueUser()
        {
            var uniqueEmail = $"ticket-test-{Guid.NewGuid()}@api-cadastro.com";
            var password = "Password123!";
            return (uniqueEmail, password);
        }

        private async Task<string> GetValidToken()
        {
```

```

using var client = _factory.CreateClient();
var (email, password) = GenerateUniqueUser();
var registerRequest = new RegisterRequest { Email = email, Senha = password,
Nome = "Ticket Test User", CEP = "00000-000" };
var registerContent = new StringContent(
JsonSerializer.Serialize(registerRequest),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json"))
);
var registerResponse = await client.PostAsync("/crie-uma-conta", registerContent);
registerResponse.EnsureSuccessStatusCode();

var loginRequest = new LoginRequest { Email = email, Senha = password };
var loginContent = new StringContent(
JsonSerializer.Serialize(loginRequest),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json"))
);
var response = await client.PostAsync("/entrar", loginContent);
response.EnsureSuccessStatusCode();
var responseString = await response.Content.ReadAsStringAsync();
var authResponse = JsonSerializer.Deserialize<AuthResponse>(responseString,
new JsonSerializerOptions { PropertyNameCaseInsensitive = true });
if (authResponse == null || string.IsNullOrEmpty(authResponse.Token))
{
    Assert.Fail($"Falha ao obter token durante o login. Resposta: {responseString}");
}
return authResponse.Token;
}

[Fact]
public async Task CriarChamado_ShouldReturnOk_WhenTicketIsCreated()
{
    var token = await GetValidToken();
    _client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
}

```

```

var request = new TicketRequest { Titulo = "Novo Chamado Teste", Descricao =
"Descrição detalhada do problema." };
var content = new StringContent(
JsonSerializer.Serialize(request),
Encoding.UTF8,
new MediaTypeHeaderValue("application/json")
);

var response = await _client.PostAsync("/abrir-chamado", content);

response.EnsureSuccessStatusCode();
var responseString = await response.Content.ReadAsStringAsync();
var criarChamadoResponse = JsonSerializer.Deserialize<CriarChamadoResponse>(responseString, new JsonSerializerOptions { PropertyNameCaseInsensitive = true });
Assert.NotNull(criarChamadoResponse);
Assert.True(criarChamadoResponse.Ticket.Id > 0);
Assert.Equal("Novo Chamado Teste", criarChamadoResponse.Ticket.Titulo);
}
}
}

```

TicketControllerTests.cs:

```

using ApiCadastro.Models;

namespace ApiCadastro.Tests
{
// DTOs exclusivos para testes ou respostas que não existem em
ApiCadastro.Models.
public class ErrorResponse
{
public string Message { get; set; } = string.Empty;
}

public class AvaliarChamadoResponse
{
public string Message { get; set; } = string.Empty;
public int NotaAvaliacao { get; set; }
}

```

```
public string ComentarioAvaliacao { get; set; } = string.Empty;  
}  
}
```

ValidationTests:

```
using Xunit;  
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
using ApiCadastro.Models;  
  
namespace ApiCadastro.Tests  
{  
    public class ValidationTests  
    {  
        private void ValidateModel(object model)  
        {  
            var validationContext = new ValidationContext(model, serviceProvider: null, items: null);  
            var validationResults = new List<ValidationResult>();  
            Validator.TryValidateObject(model, validationContext, validationResults, true);  
        }  
  
        [Fact]  
        public void RegisterRequest_ShouldHaveValidationErrors_WhenRequiredFieldsAreMissing()  
        {  
            var request = new RegisterRequest();  
            var validationContext = new ValidationContext(request, serviceProvider: null, items: null);  
            var validationResults = new List<ValidationResult>();  
            var isValid = Validator.TryValidateObject(request, validationContext, validationResults, true);  
  
            Assert.False(isValid);  
            Assert.Contains(validationResults, v => v.MemberNames.Contains("Email"));  
            Assert.Contains(validationResults, v => v.MemberNames.Contains("Senha"));  
            Assert.Contains(validationResults, v => v.MemberNames.Contains("Nome"));  
        }  
    }  
}
```

```

Assert.Contains(validationResults, v => v.MemberNames.Contains("CEP"));
}

[Fact]
public void LoginRequest_ShouldHaveValidationErrors_WhenRequiredFieldsAreMissing()
{
    var request = new LoginRequest();
    var validationContext = new ValidationContext(request, serviceProvider: null, items: null);
    var validationResults = new List<ValidationResult>();
    var isValid = Validator.TryValidateObject(request, validationContext, validationResults, true);

    Assert.False(isValid);
    Assert.Contains(validationResults, v => v.MemberNames.Contains("Email"));
    Assert.Contains(validationResults, v => v.MemberNames.Contains("Senha"));
}

[Fact]
public void LoginRequest_ShouldHaveValidationErrors_WhenEmailIsInvalid()
{
    var request = new LoginRequest { Email = "invalid-email", Senha = "password" };
    var validationContext = new ValidationContext(request, serviceProvider: null, items: null);
    var validationResults = new List<ValidationResult>();
    var isValid = Validator.TryValidateObject(request, validationContext, validationResults, true);

    Assert.False(isValid);
    Assert.Contains(validationResults, v => v.MemberNames.Contains("Email"));
}

[Fact]
public void TicketRequest_ShouldHaveValidationErrors_WhenRequiredFieldsAreMissing()
{
    var request = new TicketRequest();
}

```

```

var validationContext = new ValidationContext(request, serviceProvider: null, items: null);
var validationResults = new List<ValidationResult>();
var isValid = Validator.TryValidateObject(request, validationContext, validationResults, true);

Assert.False(isValid);
Assert.Contains(validationResults, v => v.MemberNames.Contains("Titulo"));
Assert.Contains(validationResults, v => v.MemberNames.Contains("Descricao"));
}

[Fact]
public void TicketAvaliacaoRequest_ShouldHaveValidationErrors_WhenNotalsOutOfRange()
{
    var request = new TicketAvaliacaoRequest { Nota = 0 };
    var validationContext = new ValidationContext(request, serviceProvider: null, items: null);
    var validationResults = new List<ValidationResult>();
    var isValid = Validator.TryValidateObject(request, validationContext, validationResults, true);

    Assert.False(isValid);
    Assert.Contains(validationResults, v => v.MemberNames.Contains("Nota"));

    request = new TicketAvaliacaoRequest { Nota = 6 };
    validationContext = new ValidationContext(request, serviceProvider: null, items: null);
    validationResults = new List<ValidationResult>();
    isValid = Validator.TryValidateObject(request, validationContext, validationResults, true);

    Assert.False(isValid);
    Assert.Contains(validationResults, v => v.MemberNames.Contains("Nota"));
}
}
}

```

14. Códigos Front-end:

Src/api/api.js:

```
// src/api/api.js
import AsyncStorage from "@react-native-async-storage/async-storage";
import axios from "axios";

const api = axios.create({
  baseURL: "https://srv1119481.hstgr.cloud/api",
  headers: {
    "Content-Type": "application/json"
  }
});

// 🔒 Interceptador opcional: adiciona o token JWT automaticamente
api.interceptors.request.use(
  async (config) => {
    const token = await AsyncStorage.getItem("token");
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => Promise.reject(error)
);

// =====
// ◇ ENDPOINTS DE LOGIN
// =====

export const criarConta = async (dados) => {
  try {
    const response = await api.post("/crie-uma-conta", dados);
    return response.data;
  } catch (error) {
    throw error.response?.data || { message: "Erro ao criar conta." };
  }
}
```

```
};

export const entrar = async (dados) => {
try {
const response = await api.post("/entrar", dados);
return response.data;
} catch (error) {
throw error.response?.data || { message: "Erro ao entrar." };
}
};

// =====
// ◇ ENDPOINTS DE CHAMADOS
// =====

export const abrirChamado = async (dados) => {
try {
const response = await api.post("/abrir-chamado", dados);
return response.data;
} catch (error) {
throw error.response?.data || { message: "Erro ao abrir chamado." };
}
};

export const listarChamados = async () => {
try {
const response = await api.get("/meus-chamados");
return response.data;
} catch (error) {
throw error.response?.data || { message: "Erro ao listar chamados." };
}
};

export const avaliarChamado = async (id, dados) => {
try {
const response = await api.post(`/avaliar-chamado/${id}`, dados);
return response.data;
} catch (error) {
throw error.response?.data || { message: "Erro ao avaliar chamado." };
}
};
```

```

}

};

export const removerChamado = async (id) => {
try {
const response = await api.delete(`/chamados/${id}`);
return response.data;
} catch (error) {
throw error.response?.data || { message: "Erro ao remover chamado." };
}
};

export default api;

```

Src/navigation/AppNavigator.js:

```

// src/navigation/AppNavigator.js
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import AgradecimentoAvaliacaoScreen from './screens/AgradecimentoAvaliacaoScreen';
import AvaliarChamadoScreen from './screens/AvaliarChamadoScreen';
import LoginScreen from './screens/LoginScreen';
import MeusChamadosScreen from './screens/MeusChamadosScreen';
import NovoChamadoScreen from './screens/NovoChamadoScreen';
import RegisterScreen from './screens/RegisterScreen';
import SucessoScreen from './screens/SucessoScreen';

const Stack = createNativeStackNavigator();
export default function AppNavigator() {
return (
<Stack.Navigator initialRouteName="Login">
<Stack.Screen name="Login" component={LoginScreen} />
<Stack.Screen name="Registrar" component={RegisterScreen} />
<Stack.Screen name="Meus Chamados" component={MeusChamadosScreen} />
<Stack.Screen name="Novo Chamado" component={NovoChamadoScreen} />
<Stack.Screen name="Avaliar Chamado" component={AvaliarChamadoScreen} />
<Stack.Screen name="Sucesso" component={SucessoScreen} />

```

```
<Stack.Screen name="AgradecimentoAvaliacao" component={AgradecimentoAvaliacaoScreen} />
</Stack.Navigator>
);
}
```

Src/screens/AgradecimentoAvaliacaoScreen.js:

```
import { StyleSheet, Text, TouchableOpacity, View } from "react-native";

export default function AgradecimentoAvaliacaoScreen({ navigation }) {
  return (
    <View style={styles.container}>
      <Text style={styles.emoji}>☑</Text>
      <Text style={styles.title}>Obrigado pela sua avaliação!</Text>

      <TouchableOpacity
        style={styles.button}
        onPress={() => navigation.navigate("Novo Chamado")}
      >
        <Text style={styles.buttonText}>Abrir Novo Chamado</Text>
      </TouchableOpacity>
      <TouchableOpacity
        style={[styles.button, styles.logoutButton]}
        onPress={() => navigation.replace("Login")}
      >
        <Text style={styles.buttonText}>Sair</Text>
      </TouchableOpacity>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#ECE5DF",
    alignItems: "center",
    justifyContent: "center",
  },
  emoji: {
    color: "#E91E63",
    font-size: 40,
  },
  title: {
    color: "#E91E63",
    font-size: 18,
    margin: 10,
  },
  button: {
    width: 150,
    height: 45,
    backgroundColor: "#E91E63",
    borderRadius: 25,
    display: "flex",
    align-items: "center",
    justify-content: "center",
    color: "white",
    font-weight: "bold",
    margin-bottom: 10,
  },
  buttonText: {
    color: "white",
    font-size: 14,
  },
  logoutButton: {
    width: 150,
    height: 45,
    backgroundColor: "#E91E63",
    borderRadius: 25,
    display: "flex",
    align-items: "center",
    justify-content: "center",
    color: "white",
    font-weight: "bold",
  },
});
```

```
padding: 20,  
},  
emoji: {  
fontSize: 80,  
marginBottom: 20,  
},  
title: {  
fontSize: 28,  
fontWeight: "bold",  
color: "#1D361F",  
textAlign: "center",  
marginBottom: 30,  
},  
button: {  
backgroundColor: "#1D361F",  
borderRadius: 12,  
paddingVertical: 14,  
paddingHorizontal: 20,  
alignItems: "center",  
marginBottom: 15,  
width: "90%",  
},  
buttonText: {  
color: "#ECE5DF",  
fontSize: 16,  
fontWeight: "bold",  
},  
logoutButton: {  
backgroundColor: "#859B48",  
},  
logoutButtonText: {  
color: "#ECE5DF",  
fontSize: 16,  
fontWeight: "bold",  
},  
});
```

Src/screens/AvaliarChamadoScreen.js:

```
import { useState } from "react";
import {
  Alert,
  ScrollView,
  StyleSheet,
  Text,
  TextInput,
  TouchableOpacity,
  View,
} from "react-native";
import { avaliarChamado } from "../api/api";

// Componente para renderizar uma estrela
const Star = ({ filled, onPress }) => (
  <TouchableOpacity onPress={onPress}>
    <Text style={filled ? styles.starFilled : styles.starEmpty}>★</Text>
  </TouchableOpacity>
);

export default function AvaliarChamadoScreen({ route, navigation }) {
  // Recebe o ID do ticket da tela anterior
  const { ticketId } = route.params || {};
  const [nota, setNota] = useState(0);
  const [comentario, setComentario] = useState("");

  const handleAvaliar = async () => {
    if (!ticketId) {
      Alert.alert("Erro", "ID do chamado não encontrado.");
      return;
    }
    if (nota === 0) {
      Alert.alert("Atenção", "Por favor, selecione uma nota de 1 a 5.");
      return;
    }
    await avaliarChamado(ticketId, nota, comentario);
    navigation.navigate("TicketList");
  };
}
```

```

try {
  // Usa a função da API, que já lida com a autenticação
  await avaliarChamado(ticketId, { nota, comentario });

  // Navega para a nova tela de agradecimento pela avaliação
  navigation.navigate("Agradecimento Avaliacao");
} catch (err) {
  console.error(err.response || err);
  Alert.alert(
    "Erro",
    err.response?.data?.message || "Falha ao registrar avaliação."
  );
}
};

return (
<ScrollView contentContainerStyle={styles.container}>
<Text style={styles.title}>Avaliação do Chamado</Text>
<Text style={styles.subtitle}>Como foi seu atendimento?</Text>

<Text style={styles.label}>Nota</Text>
<View style={styles.starsContainer}>
{[1, 2, 3, 4, 5].map((i) => (
<Star key={i} filled={i <= nota} onPress={() => setNota(i)} />
))}
</View>

<Text style={styles.label}>Comentário</Text>
<TextInput
style={[styles.input, styles.textArea]}
value={comentario}
onChangeText={setComentario}
placeholder="Deixe seu comentário (opcional)"
placeholderTextColor="#859B48"
multiline
numberOfLines={5}
/>

```

```
<TouchableOpacity style={styles.button} onPress={handleAvaliar}>
  <Text style={styles.buttonText}>Enviar Avaliação</Text>
</TouchableOpacity>

<TouchableOpacity
  style={[styles.button, styles.backButton]}
  onPress={() => navigation.goBack()}
>
  <Text style={styles.backButtonText}>Cancelar</Text>
</TouchableOpacity>
</ScrollView>
);

}

const styles = StyleSheet.create({
  container: {
    flexGrow: 1,
    backgroundColor: "#ECE5DF",
    padding: 20,
    justifyContent: "center",
  },
  title: {
    fontSize: 28,
    fontWeight: "bold",
    color: "#1D361F",
    marginBottom: 5,
    textAlign: "center",
  },
  subtitle: {
    fontSize: 16,
    color: "#859B48",
    textAlign: "center",
    marginBottom: 25,
  },
  label: {
    fontSize: 16,
    color: "#1D361F",
  }
});
```

```
marginBottom: 10,  
marginTop: 10,  
fontWeight: "600",  
},  
input: {  
backgroundColor: "#DFC8B6",  
borderRadius: 10,  
padding: 12,  
color: "#1D361F",  
fontSize: 16,  
},  
textArea: {  
height: 100,  
textAlignVertical: "top",  
},  
button: {  
backgroundColor: "#1D361F",  
borderRadius: 12,  
padding: 16,  
alignItems: "center",  
marginTop: 25,  
},  
buttonText: {  
color: "#ECE5DF",  
fontSize: 18,  
fontWeight: "bold",  
},  
backButton: {  
backgroundColor: "#859B48",  
},  
backButtonText: {  
color: "#ECE5DF",  
fontSize: 16,  
fontWeight: "bold",  
},  
starsContainer: {  
flexDirection: "row",  
justifyContent: "center",
```

```
marginBottom: 20,  
},  
starEmpty: {  
fontSize: 40,  
color: "#DFC8B6",  
marginHorizontal: 5,  
},  
starFilled: {  
fontSize: 40,  
color: "#FFC107",  
marginHorizontal: 5,  
},  
});
```

Src/screens/LoginScreen.js:

```
// src/screens/LoginScreen.js  
import { useState } from "react";  
import { Alert, StyleSheet, Text, TextInput, TouchableOpacity, View } from "react-native";  
import api from "../api/api";  
import AsyncStorage from "@react-native-async-storage/async-storage";  
  
export default function LoginScreen({ navigation }) {  
const [email, setEmail] = useState("");  
const [senha, setSenha] = useState("");  
  
const handleLogin = async () => {  
if (!email || !senha) {  
Alert.alert("Atenção", "Preencha e-mail e senha.");  
return;  
}  
  
try {  
const res = await api.post("/entrar", { email, senha });  
const auth = res.data;
```

```
const token = auth.token || auth.Token;
if (token) {
  await AsyncStorage.setItem("token", token);
}

navigation.replace("Novo Chamado", {
  token: token,
  nome: auth.nome || auth.Nome,
});
} catch (err) {
  console.error(err.response || err);
  Alert.alert("Erro ao entrar", err.response?.data?.message || "E-mail ou senha inválidos.");
}
};

return (
<View style={styles.container}>
<View style={styles.card}>
<Text style={styles.title}>ConnectWay</Text>

<TextInput
placeholder="E-mail"
placeholderTextColor="#555"
style={styles.input}
value={email}
onChangeText={setEmail}
keyboardType="email-address"
autoCapitalize="none"
/>

<TextInput
placeholder="Senha"
placeholderTextColor="#555"
style={styles.input}
value={senha}
onChangeText={setSenha}
```

```
secureTextEntry
/>

<TouchableOpacity style={styles.button} onPress={handleLogin}>
<Text style={styles.buttonText}>Entrar</Text>
</TouchableOpacity>

<TouchableOpacity onPress={() => navigation.navigate("Registrar")}>
<Text style={styles.linkText}>Criar uma conta</Text>
</TouchableOpacity>
</View>
</View>
);
}

const styles = StyleSheet.create({
container: {
flex: 1,
backgroundColor: "#DFC8B6",
justifyContent: "center",
alignItems: "center",
padding: 20,
},
card: {
width: "100%",
backgroundColor: "#ECE5DF",
borderRadius: 16,
padding: 25,
shadowColor: "#000",
shadowOpacity: 0.1,
shadowRadius: 10,
elevation: 5,
},
title: {
fontSize: 28,
color: "#1D361F",
textAlign: "center",
fontWeight: "bold",

```

```
marginBottom: 25,  
},  
input: {  
backgroundColor: "#C4C7B6",  
padding: 12,  
borderRadius: 10,  
marginBottom: 15,  
fontSize: 16,  
color: "#1D361F",  
},  
button: {  
backgroundColor: "#859B48",  
padding: 15,  
borderRadius: 12,  
alignItems: "center",  
marginVertical: 10,  
},  
buttonText: {  
color: "#ECE5DF",  
fontSize: 18,  
fontWeight: "bold",  
},  
linkText: {  
textAlign: "center",  
color: "#1D361F",  
fontSize: 16,  
marginTop: 8,  
},  
});
```

Src/screens/MeusChamadosScreen.js:

```
import { useEffect, useState } from "react";  
import {  
ActivityIndicator,  
Alert,
```

```
FlatList,  
StyleSheet,  
Text,  
TouchableOpacity,  
View,  
} from "react-native";  
import api from "../api/api";  
  
export default function MeusChamados({ route, navigation }) {  
  const { token, nome } = route.params || {};  
  const [chamados, setChamados] = useState([]);  
  const [loading, setLoading] = useState(true);  
  
  const carregarChamados = async () => {  
    try {  
      const res = await api.get("/meus-chamados", {  
        headers: { Authorization: `Bearer ${token}` },  
      });  
      setChamados(res.data || []);  
    } catch (err) {  
      console.error(err.response || err);  
      Alert.alert("Erro", "Falha ao carregar chamados.");  
    } finally {  
      setLoading(false);  
    }  
  };  
  
  useEffect(() => {  
    carregarChamados();  
  }, []);  
  
  const renderChamado = ({ item }) => (  
    <View style={styles.card}>  
      <Text style={styles.cardTitle}>{item.titulo || "Chamado sem título"}</Text>  
      <Text style={styles.cardText}>Status: {item.status || "Desconhecido"}</Text>  
      <Text style={styles.cardText}>Data: {item.dataAbertura || "—"}</Text>  
  
      <TouchableOpacity
```

```

style={styles.button}
onPress={() =>
navigation.navigate("AvaliarChamado", {
token,
chamadoid: item.id,
nome,
})
}
>
<Text style={styles.buttonText}>Avaliar Atendimento</Text>
</TouchableOpacity>
</View>
);

if (loading) {
return (
<View style={styles.container}>
<ActivityIndicator size="large" color="#1D361F" />
<Text style={styles.loadingText}>Carregando chamados...</Text>
</View>
);
}

return (
<View style={styles.container}>
<Text style={styles.title}>Bem-vindo(a), {nome}!</Text>
<Text style={styles.subtitle}>Seus chamados registrados:</Text>

{chamados.length === 0 ? (
<Text style={styles.emptyText}>Você ainda não possui chamados.</Text>
) : (
<FlatList
data={chamados}
keyExtractor={(item) => item.id?.toString()}
renderItem={renderChamado}
contentContainerStyle={styles.list}
/>
)}

```

```
</View>
);
}

const styles = StyleSheet.create({
container: {
flex: 1,
backgroundColor: "#ECE5DF",
padding: 20,
},
title: {
fontSize: 26,
fontWeight: "bold",
color: "#1D361F",
textAlign: "center",
marginBottom: 5,
},
subtitle: {
fontSize: 16,
color: "#859B48",
textAlign: "center",
marginBottom: 20,
},
list: {
paddingBottom: 20,
},
card: {
backgroundColor: "#DFC8B6",
borderRadius: 12,
padding: 16,
marginBottom: 15,
shadowColor: "#000",
shadowOpacity: 0.1,
shadowRadius: 5,
elevation: 3,
},
cardTitle: {
fontSize: 18,
```

```
fontWeight: "bold",
color: "#1D361F",
marginBottom: 6,
},
cardText: {
fontSize: 15,
color: "#1D361F",
marginBottom: 4,
},
button: {
backgroundColor: "#1D361F",
borderRadius: 10,
paddingVertical: 10,
marginTop: 10,
alignItems: "center",
},
buttonText: {
color: "#ECE5DF",
fontSize: 16,
fontWeight: "bold",
},
emptyText: {
fontSize: 16,
color: "#859B48",
textAlign: "center",
marginTop: 40,
},
loadingText: {
marginTop: 10,
color: "#859B48",
},
});
});
```

Src/screens/NovoChamadoScreen.js:

```
// src/screens/NovoChamado.js
import { useState } from "react";
```

```

import {
Alert,
ScrollView,
StyleSheet,
Text,
TextInput,
TouchableOpacity
} from "react-native";
import { abrirChamado } from "../api/api";

export default function NovoChamado({ navigation }) {
const [titulo, setTitulo] = useState("");
const [descricao, setDescricao] = useState("");
const [categoria, setCategoria] = useState("");

const handleEnviarChamado = async () => {
if (!titulo || !descricao || !categoria) {
Alert.alert("Preencha todos os campos!");
return;
}

try {
// Usa a função da API, que já trata a autenticação
const resposta = await abrirChamado({ titulo, descricao, categoria });

// Navega para a tela de sucesso com a resposta da IA
navigation.navigate("Sucesso", { resposta });
} catch (err) {
console.error(err.response || err);
Alert.alert("Erro", err.response?.data?.message || "Falha ao enviar chamado.");
}
};

return (
<ScrollView contentContainerStyle={styles.container}>
<Text style={styles.title}>Novo Chamado</Text>
<Text style={styles.subtitle}>Preencha os detalhes abaixo:</Text>

```

```

<Text style={styles.label}>Título</Text>
<TextInput
  style={styles.input}
  value={titulo}
  onChangeText={setTitulo}
  placeholder="Ex: Problema no sistema"
  placeholderTextColor="#859B48"
/>

<Text style={styles.label}>Categoria</Text>
<TextInput
  style={styles.input}
  value={categoria}
  onChangeText={setCategoria}
  placeholder="Ex: Suporte Técnico"
  placeholderTextColor="#859B48"
/>

<Text style={styles.label}>Descrição</Text>
<TextInput
  style={[styles.input, styles.textArea]}
  value={descricao}
  onChangeText={setDescricao}
  placeholder="Descreva o problema com detalhes"
  placeholderTextColor="#859B48"
  multiline
  numberOfLines={5}
/>

<TouchableOpacity style={styles.button} onPress={handleEnviarChamado}>
  <Text style={styles.buttonText}>Enviar Chamado</Text>
</TouchableOpacity>

<TouchableOpacity
  style={[styles.button, styles.backButton]}
  onPress={() => navigation.navigate("Meus Chamados")}
>
  <Text style={styles.backButtonText}>Ver Meus Chamados</Text>

```

```
</TouchableOpacity>
</ScrollView>
);
}

const styles = StyleSheet.create({
container: {
flexGrow: 1,
backgroundColor: "#ECE5DF",
padding: 20,
justifyContent: "center",
},
title: {
fontSize: 28,
fontWeight: "bold",
color: "#1D361F",
marginBottom: 5,
textAlign: "center",
},
subtitle: {
fontSize: 16,
color: "#859B48",
textAlign: "center",
marginBottom: 25,
},
label: {
fontSize: 16,
color: "#1D361F",
marginBottom: 6,
marginTop: 10,
fontWeight: "600",
},
input: {
backgroundColor: "#DFC8B6",
borderRadius: 10,
padding: 12,
color: "#1D361F",
fontSize: 16,
```

```
},
textArea: {
height: 100,
textAlignVertical: "top",
},
button: {
backgroundColor: "#1D361F",
borderRadius: 12,
padding: 16,
alignItems: "center",
marginTop: 25,
},
buttonText: {
color: "#ECE5DF",
fontSize: 18,
fontWeight: "bold",
},
backButton: {
backgroundColor: "#859B48",
},
backButtonText: {
color: "#ECE5DF",
fontSize: 16,
fontWeight: "bold",
},
});
```

Src/screens/RegisterScreen.js:

```
import { useState } from "react";
import {
Alert,
ScrollView,
StyleSheet,
Text,
TextInput,
TouchableOpacity,
```

```

} from "react-native";
import AsyncStorage from "@react-native-async-storage/async-storage";
import api from "../api/api";

export default function RegistrarScreen({ navigation }) {
  const [dados, setDados] = useState({
    nome: "",
    email: "",
    senha: "",
    telefone: "",
    logradouro: "",
    numero: "",
    complemento: "",
    bairro: "",
    cidade: "",
    estado: "",
    cep: ""
  });

  const handleChange = (campo, valor) => {
    setDados({ ...dados, [campo]: valor });
  };

  const handleCriarConta = async () => {
    console.log("🕒 Botão Criar Conta clicado!");

    const { nome, email, senha } = dados;

    if (!nome || !email || !senha) {
      Alert.alert("Atenção", "Preencha nome, e-mail e senha.");
      return;
    }

    try {
      console.log("📦 Enviando dados para registro:", dados);
      const response = await api.post("/usuarios/register", dados);

      console.log("✅ Resposta do backend:", response.data);
    }
  };
}

```

```

const token = response.data.token || response.data.Token;

if (token) {
  await AsyncStorage.setItem("token", token);
  console.log("🔗 Token salvo com sucesso!");

  Alert.alert("Sucesso!", "Conta criada com sucesso.", [
    {
      text: "OK",
      onPress: () =>
        navigation.replace("Novo Chamado", {
          nome: response.data.nome || response.data.Nome || nome,
        }),
    },
  ]);
} else {
  console.log("❌ Token não encontrado na resposta do backend.");
  Alert.alert(
    "Erro de Autenticação",
    "Registro bem-sucedido, mas não foi possível autenticar. Por favor, tente fazer o login."
  );
  navigation.navigate("Login");
}
} catch (err) {
  console.log("❌ Erro no criarConta:", err.response?.data || err.message);
  Alert.alert(
    "Erro no Cadastro",
    err.response?.data?.message ||
    "Não foi possível criar a conta. Verifique os dados e tente novamente."
  );
}

return (
  <ScrollView contentContainerStyle={styles.container}>
    <Text style={styles.title}>Crie sua conta</Text>

```

```
<TextInput
  style={styles.input}
  placeholder="Nome completo"
  placeholderTextColor="#859B48"
  value={dados.nome}
  onChangeText={(v) => handleChange("nome", v)}
/>
<TextInput
  style={styles.input}
  placeholder="E-mail"
  placeholderTextColor="#859B48"
  value={dados.email}
  onChangeText={(v) => handleChange("email", v)}
  keyboardType="email-address"
  autoCapitalize="none"
/>
<TextInput
  style={styles.input}
  placeholder="Senha"
  placeholderTextColor="#859B48"
  secureTextEntry
  value={dados.senha}
  onChangeText={(v) => handleChange("senha", v)}
/>
<TextInput
  style={styles.input}
  placeholder="Telefone"
  placeholderTextColor="#859B48"
  value={dados.telefone}
  onChangeText={(v) => handleChange("telefone", v)}
  keyboardType="phone-pad"
/>
<TextInput
  style={styles.input}
  placeholder="Logradouro"
  placeholderTextColor="#859B48"
  value={dados.logradouro}
  onChangeText={(v) => handleChange("logradouro", v)}
```

```
/>
<TextInput
  style={styles.input}
  placeholder="Número"
  placeholderTextColor="#859B48"
  value={dados.numero}
  onChangeText={(v) => handleChange("numero", v)}
  keyboardType="numeric"
/>
<TextInput
  style={styles.input}
  placeholder="Complemento"
  placeholderTextColor="#859B48"
  value={dados.complemento}
  onChangeText={(v) => handleChange("complemento", v)}
/>
<TextInput
  style={styles.input}
  placeholder="Bairro"
  placeholderTextColor="#859B48"
  value={dados.bairro}
  onChangeText={(v) => handleChange("bairro", v)}
/>
<TextInput
  style={styles.input}
  placeholder="Cidade"
  placeholderTextColor="#859B48"
  value={dados.cidade}
  onChangeText={(v) => handleChange("cidade", v)}
/>
<TextInput
  style={styles.input}
  placeholder="Estado"
  placeholderTextColor="#859B48"
  value={dados.estado}
  onChangeText={(v) => handleChange("estado", v)}
/>
<TextInput
```

```
style={styles.input}
placeholder="CEP"
placeholderTextColor="#859B48"
value={dados.cep}
onChangeText={(v) => handleChange("cep", v)}
keyboardType="numeric"
/>>

<TouchableOpacity style={styles.button} onPress={handleCriarConta}>
<Text style={styles.buttonText}>Criar Conta</Text>
</TouchableOpacity>

<TouchableOpacity
style={[styles.button, styles.backButton]}
onPress={() => navigation.navigate("Login")}
>
<Text style={styles.backButtonText}>Voltar</Text>
</TouchableOpacity>
</ScrollView>
);

}

const styles = StyleSheet.create({
container: {
flexGrow: 1,
backgroundColor: "#ECE5DF",
padding: 20,
},
title: {
fontSize: 26,
color: "#1D361F",
fontWeight: "bold",
textAlign: "center",
marginBottom: 20,
},
input: {
backgroundColor: "#DFC8B6",
borderRadius: 10,
```

```
padding: 12,  
color: "#1D361F",  
fontSize: 16,  
marginBottom: 12,  
},  
button: {  
backgroundColor: "#1D361F",  
borderRadius: 12,  
padding: 15,  
alignItems: "center",  
marginTop: 10,  
},  
buttonText: {  
color: "#ECE5DF",  
fontSize: 18,  
fontWeight: "bold",  
},  
backButton: {  
backgroundColor: "#859B48",  
},  
backButtonText: {  
color: "#ECE5DF",  
fontSize: 16,  
fontWeight: "bold",  
},  
});
```

Src/screens/SucessoScreen.js:

```
import { useEffect } from "react"; // Import useEffect  
import { ScrollView, StyleSheet, Text, TouchableOpacity, View } from "react-native";  
  
export default function Sucesso({ route, navigation }) {  
const { resposta } = route.params || {};  
  
const displayMessage =
```

```
resposta && resposta.ticket && resposta.ticket.respostaIA
? resposta.ticket.respostaIA
: "Seu chamado foi aberto e processado com sucesso.";

const ticketId = resposta?.ticket?.id;

// Configura o botão de cabeçalho dinamicamente
useEffect(() => {
if (ticketId) {
navigation.setOptions({
headerRight: () => (
<TouchableOpacity
style={styles.headerButton}
onPress={() => navigation.navigate("Avaliar Chamado", { ticketId })}
>
<Text style={styles.headerButtonText}>Avaliar</Text>
</TouchableOpacity>
),
});
} else {
// Limpa o botão do cabeçalho se não houver ticketId
navigation.setOptions({ headerRight: undefined });
}
}, [navigation, ticketId]); // Dependências do useEffect

return (
<ScrollView
style={styles.container}
contentContainerStyle={styles.scrollContent}
>
<Text style={styles.emoji}>🤖</Text>
<Text style={styles.title}>Resposta da IA</Text>
<Text style={styles.subtitle}>{displayMessage}</Text>

<TouchableOpacity
style={styles.button}
onPress={() => navigation.navigate("Meus Chamados")}
>
```

```
<Text style={styles.buttonText}>Ver Meus Chamados</Text>
</TouchableOpacity>

</ScrollView>
);
}

const styles = StyleSheet.create({
container: {
flex: 1,
backgroundColor: "#ECE5DF",
},
scrollContent: {
alignItems: "center",
padding: 20,
},
emoji: {
fontSize: 80,
marginBottom: 20,
marginTop: 20,
},
title: {
fontSize: 28,
fontWeight: "bold",
color: "#1D361F",
textAlign: "center",
marginBottom: 30,
},
subtitle: {
fontSize: 16,
color: "#555",
textAlign: "left",
marginBottom: 40,
paddingHorizontal: 10,
lineHeight: 24,
},
button: {
backgroundColor: "#1D361F",
```

```
borderRadius: 12,  
paddingVertical: 14,  
paddingHorizontal: 20,  
alignItems: "center",  
marginBottom: 15,  
width: "90%",  
},  
buttonText: {  
color: "#ECE5DF",  
fontSize: 16,  
fontWeight: "bold",  
},  
headerButton: {  
marginRight: 15,  
backgroundColor: "#859B48",  
paddingVertical: 8,  
paddingHorizontal: 12,  
borderRadius: 8,  
},  
headerButtonText: {  
color: "#ECE5DF",  
fontSize: 14,  
fontWeight: "bold",  
},  
});
```

Src/screens/TicketsScreen.js:

```
// src/screens/TicketsScreen.js  
import { useEffect, useState } from "react";  
import {  
ActivityIndicator,  
Alert,  
FlatList,  
StyleSheet,  
Text,
```

```
TouchableOpacity,  
View,  
} from "react-native";  
import api from "../api/api";  
  
export default function TicketsScreen({ route, navigation }) {  
  const { token, nome } = route.params || {};  
  const [chamados, setChamados] = useState([]);  
  const [loading, setLoading] = useState(true);  
  
  const carregarChamados = async () => {  
    try {  
      const res = await api.get("/meus-chamados", {  
        headers: { Authorization: `Bearer ${token}` },  
      });  
      setChamados(res.data || []);  
    } catch (err) {  
      console.error(err.response || err);  
      Alert.alert("Erro", "Não foi possível carregar seus chamados.");  
    } finally {  
      setLoading(false);  
    }  
  };  
  
  useEffect(() => {  
    carregarChamados();  
  }, []);  
  
  const abrirChamado = () => {  
    navigation.navigate("NovoChamado", { token, nome });  
  };  
  
  const avaliarChamado = (id) => {  
    navigation.navigate("Avaliacao", { token, chamadold: id });  
  };  
  
  return (  
    <View style={styles.container}>
```

```

<Text style={styles.title}>Olá, {nome || "usuário"} <img alt="hand icon" style={{verticalAlign: 'middle'}}/></Text>
<Text style={styles.subtitle}>Seus Chamados</Text>

{loading ? (
  <ActivityIndicator size="large" color="#1D361F" style={{marginTop: 50}} />
) : (
  <FlatList
    data={chamados}
    keyExtractor={(item) => item.id?.toString()}
    renderItem={({ item }) => (
      <View style={styles.ticketCard}>
        <Text style={styles.ticketTitle}>{item.titulo || "Sem título"}</Text>
        <Text style={styles.ticketDesc}>
          {item.descricao || "Sem descrição"}
        </Text>
        <Text style={styles.ticketStatus}>
          Status: {item.status || "Pendente"}
        </Text>
      </View>
      <TouchableOpacity
        style={styles.evalButton}
        onPress={() => avaliarChamado(item.id)}
      >
        <Text style={styles.evalButtonText}>Avaliar</Text>
      </TouchableOpacity>
    )}
    ListEmptyComponent={
      <Text style={styles.emptyText}>
        Você ainda não possui chamados abertos.
      </Text>
    }
  />
)
}

<TouchableOpacity style={styles.newButton} onPress={abrirChamado}>
  <Text style={styles.newButtonText}>+ Novo Chamado</Text>
</TouchableOpacity>

```

```
</View>
);

}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#ECE5DF",
    padding: 20,
  },
  title: {
    fontSize: 24,
    fontWeight: "bold",
    color: "#1D361F",
    marginTop: 20,
  },
  subtitle: {
    fontSize: 18,
    color: "#859B48",
    marginBottom: 15,
  },
  ticketCard: {
    backgroundColor: "#DFC8B6",
    padding: 15,
    borderRadius: 12,
    marginBottom: 15,
    shadowColor: "#000",
    shadowOpacity: 0.1,
    shadowRadius: 5,
    elevation: 4,
  },
  ticketTitle: {
    fontSize: 18,
    color: "#1D361F",
    fontWeight: "bold",
  },
  ticketDesc: {
    color: "#1D361F",
  }
});
```

```
marginVertical: 5,  
},  
ticketStatus: {  
color: "#859B48",  
fontWeight: "600",  
marginBottom: 8,  
},  
evalButton: {  
backgroundColor: "#859B48",  
borderRadius: 8,  
padding: 10,  
alignItems: "center",  
},  
evalButtonText: {  
color: "#ECE5DF",  
fontWeight: "bold",  
},  
newButton: {  
backgroundColor: "#1D361F",  
padding: 16,  
borderRadius: 12,  
alignItems: "center",  
marginTop: 10,  
},  
newButtonText: {  
color: "#ECE5DF",  
fontSize: 18,  
fontWeight: "bold",  
},  
emptyText: {  
textAlign: "center",  
color: "#1D361F",  
marginTop: 50,  
fontSize: 16,  
},  
});
```

App.js:

```
// App.js
import { NavigationContainer } from '@react-navigation/native';
import AppNavigator from './src/navigation/AppNavigator';

export default function App() {
  return (
    <NavigationContainer>
      <AppNavigator />
    </NavigationContainer>
  );
}
```

Repositórios Disponíveis para o clone:

<https://github.com/La-silva1/SuporteTecnicoIA-front-end.git>

<https://github.com/La-silva1/SuporteTecnicoIA.git>

15. PROMPTS UTILIZADOS COM IA (ChatGPT e Gemini)

(para incluir na seção “Uso de Inteligência Artificial no Desenvolvimento” do seu PIM)

Planejamento e Estrutura do Projeto

- “Me ajude a estruturar um sistema de suporte técnico com IA, incluindo módulos, funcionalidades e fluxo principal do usuário.”
- “Quais requisitos funcionais e não funcionais são adequados para um sistema de gestão de chamados?”
- “Crie um esboço de arquitetura para um app com IA integrada.”

Desenvolvimento do Backend (API / C# / ASP.NET Core)

- “Como criar um endpoint em ASP.NET Core para autenticação com JWT?”
- “Me ajude a escrever um controller de Tickets com boas práticas de clean code.”
- “Explique como implementar relacionamento entre usuários e chamados usando Entity Framework Core.”
- “Gere um exemplo de modelo C# para a tabela Tickets contendo campos de IA.”
- “Como posso consumir uma API externa de IA (Gemini) a partir do backend em C#?”

Desenvolvimento do Frontend (React Native / Expo)

- “Como criar uma tela de login em React Native com validações e navegação?”
- “Me explique como fazer uma chamada HTTP para minha API e exibir os chamados do usuário.”
- “Gere um exemplo de UI simples usando React Native estilizado com StyleSheet.”
- “Como implementar uma tela de avaliação com estrelas em React Native?”
- “Exemplo de fluxo de navegação entre telas usando React Navigation.”

Design, UI e UX

- “Sugira paletas de cores profissionais para um sistema de suporte técnico.”
- “Me ajude a estruturar um layout limpo e responsivo para telas de cadastro.”
- “Crie textos amigáveis e profissionais para mensagens de sucesso, erro e feedback.”
- “Quais boas práticas de acessibilidade devem ser aplicadas em telas mobile?”

Banco de Dados e Modelagem (DER / SQL Server)

- “Como modelar um banco de dados para um sistema de chamados?”
- “Sugira campos essenciais para a tabela Users e Tickets.”
- “Me ajude a descrever tecnicamente o DER para incluir no relatório.”

Integração com IA (Gemini / ChatGPT)

- “Gere uma resposta padrão para quando o usuário abre um chamado de baixa complexidade.”
- “Me ajude a criar uma estrutura de prompts para a IA auxiliar no diagnóstico técnico.”
- “Crie exemplos de respostas automatizadas para problemas comuns de TI.”
- “Como melhorar a precisão das respostas da IA em casos de suporte técnico?”

Documentação do Projeto (Relatório / Descrição / Manual)

- “Me ajude a escrever a introdução do PIM sobre um sistema de suporte técnico com IA.”
- “Crie um resumo acadêmico sobre o desenvolvimento de um aplicativo de atendimento automatizado.”
- “Como descrever a arquitetura do sistema de forma clara para um trabalho acadêmico?”
- “Gere uma conclusão que destaque os benefícios e melhorias futuras.”
- “Escreva uma justificativa para o uso de Inteligência Artificial no projeto.”

Resolução de Problemas Durante o Desenvolvimento

- “Meu front não está consumindo a API, quais os possíveis erros e como resolver?”
- “Erro de CORS no ASP.NET Core: como corrigir?”
- “A autenticação JWT não está validando o token, o que posso verificar?”
- “Como rodar o backend usando Docker corretamente?”

Testes e Qualidade

- “Crie casos de testes funcionais para as telas do aplicativo.”
- “Quais testes devo aplicar na API de chamados?”
- “Como documentar os testes realizados para incluir no PIM?”

Geração das Telas e Explicação Técnica

- “Me ajude a descrever cada tela do meu aplicativo para incluir no relatório.”
- “Gere um texto acadêmico explicando o fluxo da tela de resposta da IA.”
- “Crie uma explicação técnica do layout da tela de avaliação do atendimento.”

Automatização do Documento (Word / ABNT)

- “Ajuste o texto para ABNT NBR 14724.”
- “Reescreva a conclusão de forma mais acadêmica e objetiva.”
- “Me ajude a montar um sumário técnico organizado para sistema de suporte com IA.”

FICHA DE CONTROLE DE ATIVIDADES

FICHA DE CONTROLE DO PIM

Grupo Nº _____ Ano: 2025 Período: Matutino Orientador: Thais Ferauche

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

Alunos:

RA	Nome	E-mail	Curso	Visto do aluno
G14IHE0	Lais Silva Santos	Laissilvaaa300@gmail.com	ADS	<i>Lais Silva</i>
R106AC1	Arthur Silvestre de Paula	Arthurrsilvestre7@gmail.com	ADS	<i>Arthur</i>
R173CH6	João Henrique Silva dos Santos	j.henriqueess998@gmail.com	ADS	<i>João</i>
G77III4	Igor Dantas Barros	Igordantas1711@gmail.com	ADS	<i>Igor</i>
G7737I14	Renan Francisco Ribeiro	Renanghf@gmail.com	ADS	<i>Renan</i>
G9951B6	William Gustavo Abreu de Oliveira Leite	William.gustavo.wg9@gmail.com	ADS	<i>Will</i>
G73IEJ1	Melyssa Souza Araújo	Melyssaaraujo86@gmail.com	ADS	<i>Melyssa SA</i>

Registros:

Data do encontro	Observações
05/08	Planejamento Estratégico e distribuição de tarefas
03/09	Desenvolvimento do Sistema e do Banco de Dados (versão funcional)
06/10	Desenvolvimento, Testes e Validações do Sistema
20/10	Alterações necessárias no Desenvolvimento do Sistema (BackEnd e FronEnd)
31/10	Início montagem da documentação
04/11	Finalização dos Testes e implementação completa
07/11	Formatação ABNT da documentação
10/11	Criação dos Slides

17. CONCLUSÃO

O desenvolvimento do Sistema Integrado de Gestão de Chamados com Inteligência Artificial proporcionou a aplicação prática de conceitos de Engenharia de Software, Programação Orientada a Objetos, Banco de Dados, Gerenciamento de Projetos e Qualidade. O sistema alcançou seu objetivo de centralizar operações,

automatizar respostas, reduzir o tempo de atendimento e melhorar a eficiência operacional.

A solução se mostrou moderna, escalável e alinhada às necessidades reais de empresas digitais. A integração com IA foi essencial para oferecer respostas rápidas, consistentes e inteligentes aos usuários. O uso de ASP.NET Core, SQL Server, JWT, Docker e princípios de arquitetura modular garantiu segurança, desempenho e robustez.

Como trabalho futuro, destaca-se:

- aprimoramento contínuo do módulo de IA;
- ampliação da base de conhecimento;
- novos relatórios inteligentes;
- expansão do sistema para atendimento externo;
- integração com outras APIs avançadas.

O projeto demonstra maturidade tecnológica e capacidade de inovação, contribuindo para a evolução de processos corporativos e para o desenvolvimento profissional da equipe envolvida.

18. REFERÊNCIAS

PLANTUML. PlantUML – Diagramas UML simplificados. Disponível em: <https://plantuml.com/>. Acesso em: 08 nov. 2025.

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

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

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

LARMAN, Craig. Aplicando UML e Padrões: uma introdução à análise e projeto orientado a objetos e ao processo iterativo. 3. ed. Rio de Janeiro: Elsevier, 2007.

GARCIA, Vinícius da Fonseca. Modelagem de Sistemas Orientados a Objetos com UML 2.5. 3. ed. São Paulo: Novatec, 2020.

ALPAYDIN, Ethem. Introdução ao Aprendizado de Máquina. 3. ed. Porto Alegre: Bookman, 2021.

NILSSON, Nils J. Inteligência Artificial: uma nova síntese. Rio de Janeiro: LTC, 2003.

OMG – Object Management Group. Unified Modeling Language (UML) Specification, Version 2.5.1. 2017. Disponível em: <https://www.omg.org/spec/UML/2.5.1>. Acesso em: 08 nov. 2025.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018 (Lei Geral de Proteção de Dados Pessoais – LGPD). Diário Oficial da União, Brasília, DF, 15 ago. 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 08 nov. 2025.

MACHADO, Marcos. *Gestão de Projetos de Software – UNIP*, 2025.

BEZERRA, Eduardo. *Princípios de Análise e Projeto de Sistemas com UML*. 3. ed. Rio de Janeiro: Elsevier, 2022.

PMI – Project Management Institute. *Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK®)*. 7. ed. Newtown Square, PA: Project Management Institute, 2021.

MICROSOFT. Entity Framework Core Documentation. Disponível em: <https://learn.microsoft.com/ef/core>. Acesso em: 9 nov. 2025.

MICROSOFT. SQL Server Docker Image Overview. Disponível em: https://hub.docker.com/_/microsoft-mssql-server. Acesso em: 9 nov. 2025.

ALURATECH. Curso de Entity Framework Core: Introdução e Code First. São Paulo: Alura, 2024.

FREEMAN, Adam; SANDERSON, Jon. Pro ASP.NET Core 8 MVC. 10. ed. Nova York: Apress, 2024.

SILVA, José Carlos. Modelagem de Banco de Dados Relacional. 3. ed. São Paulo: Érica, 2022.

NOGUEIRA, Fábio. Banco de Dados: Teoria e Prática para Desenvolvedores. 2. ed. Rio de Janeiro: Elsevier, 2023.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de Software: Uma Abordagem Profissional. 9. ed. Porto Alegre: AMGH, 2021.

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

DORNELAS, José Carlos Assis. Empreendedorismo: Transformando Ideias em Negócios. 7. ed. Rio de Janeiro: Empreende, 2020.

CHIAVENATO, Idalberto. Empreendedorismo: Dando Asas ao Espírito Empreendedor. 5. ed. São Paulo: Saraiva, 2019.

GOMES, Nilma Lino. Educação e Identidade Negra: Entre o Silêncio e a Memória. Belo Horizonte: Autêntica, 2021.

MUNANGA, Kabengele. Rediscutindo a Mestiçagem no Brasil: Identidade Nacional versus Identidade Negra. Petrópolis: Vozes, 2019.

SLIDES:



Engenharia de Software: Melhorias



FOCO NA AUTOMAÇÃO

Sistema redesenhado para um suporte 100% digital e automatizado com IA, removendo a triagem manual.



RASTREABILIDADE E VERSÃO

Uso de Git/GitHub para controle de versão, documentação e rastreabilidade total das alterações.



TESTES INTEGRADOS

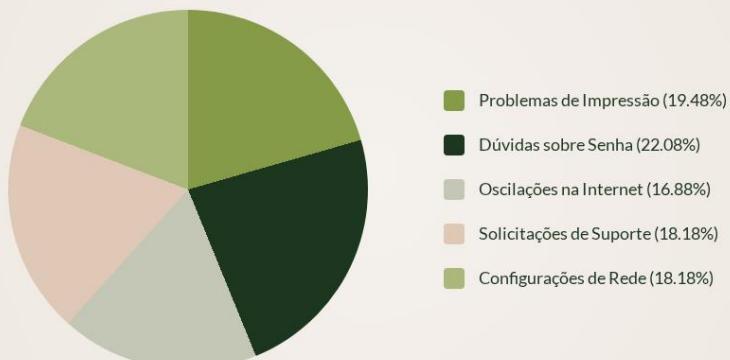
Validação de ponta-a-ponta (API, Banco de Dados, IA) para garantir a robustez e confiabilidade do sistema.



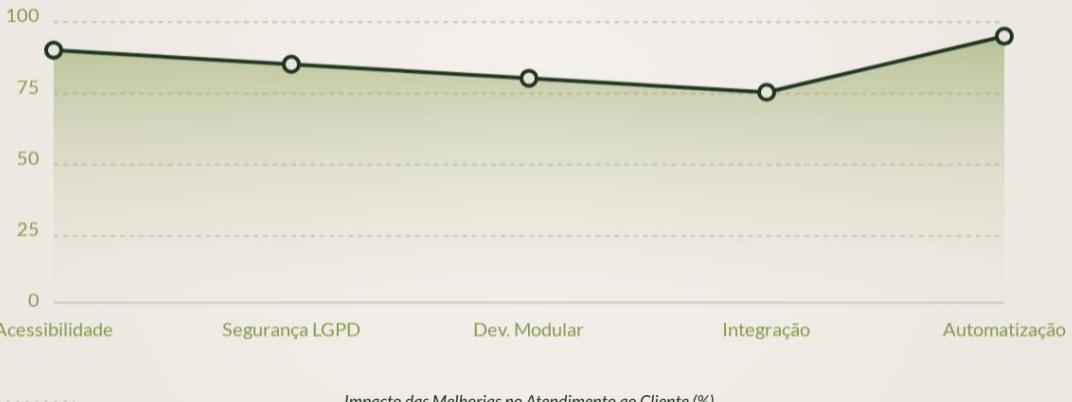
ARQUITETURA ADAPTADA (LGPD)

O design do software foi revisado para incluir anonimização de dados e conformidade total com a LGPD.

Otimização do Atendimento Técnico



Aprimoramento do Sistema de Suporte Técnico com IA



Gerenciamento de Projetos: Cronograma (24 Semanas)

Fase	Atividade Principal	Duração
1. Levantamento e Planejamento	Coleta de Requisitos e Mapeamento do Processo Atual	3 Semanas
2. Design e Arquitetura	Modelagem do Banco de Dados e Prototipação das Interfaces	4 Semanas
3. Desenvolvimento	Programação (Web, Desktop, Mobile) e Módulo de IA	9 Semanas
4. Testes	Testes Unitários, de Integração e Piloto com Equipe Interna	4 Semanas
5. Implantação e Entrega	Treinamento da Equipe e Liberação Oficial do Sistema	4 Semanas

Gestão da Qualidade: Pontos Focais



Design Inclusivo

Interface intuitiva prototipada no Figma e validação de conformidade com padrões WCAG (Axe DevTools).



Performance e Segurança

Garantia de respostas em menos de 3s (testes com k6) e proteção de dados com autenticação JWT.



Código Robusto

Arquitetura modular (MVC), testes unitários (xUnit) e documentação de API via Swagger para manutenibilidade.



Segurança de dados e conformidade com a LGPD

2. Consentimento do Usuário

Políticas de consentimento explícito para o uso de dados pessoais.

4. Controle de Acesso

Autenticação em múltiplos níveis (JWT) para proteger o sistema.

1. Proteção de Dados

Criptografia de dados pessoais para garantir a segurança das informações.

3. Auditoria de Dados

Registro de todos os acessos e alterações para monitoramento de segurança.



Diagrama de Casos de Uso

O diagrama de caso de uso apresenta as interações entre os atores (Usuário e Sistema de IA) e as funcionalidades principais do ConnectWay.

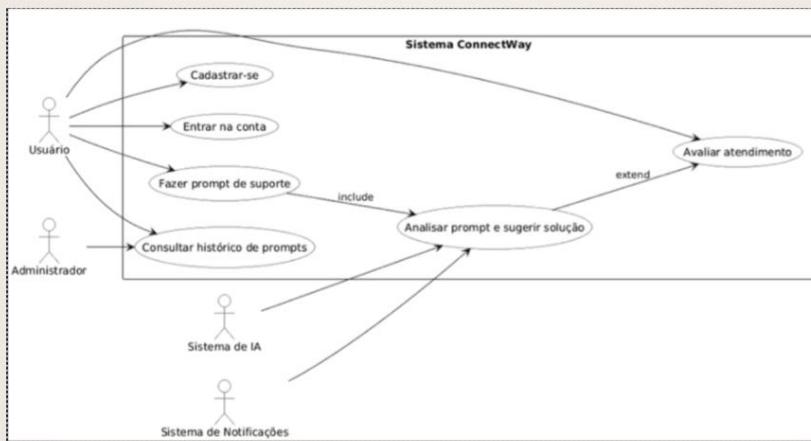


Diagrama de Classes

O diagrama de classes representa a estrutura lógica do ConnectWay, detalhando os atributos, métodos e relacionamentos entre as classes.

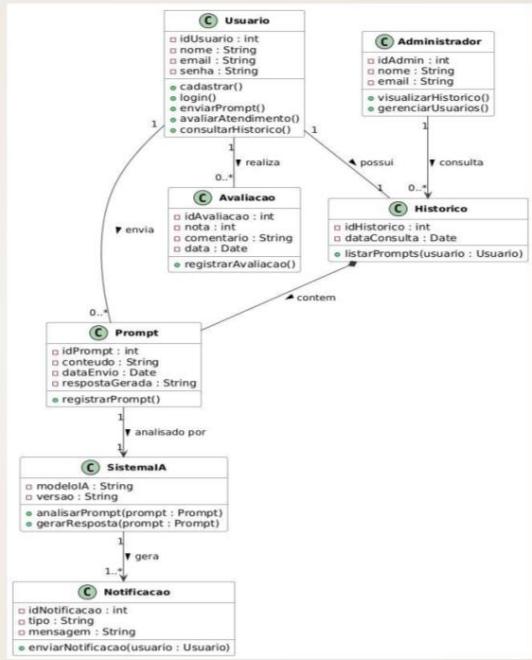
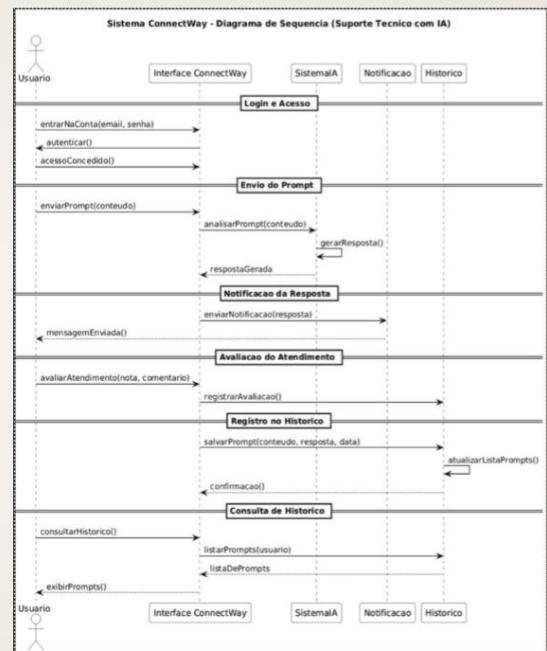


Diagrama de Sequência

O diagrama de sequência descreve o fluxo temporal da comunicação entre os objetos, passo a passo.



Arquitetura Backend



Frontend: Telas Web (1/2)

The screenshot shows the ConnectWay login interface. It features a header with the brand name 'ConnectWay'. Below it is a form with fields for 'E-mail' and 'Senha' (Password). A green 'Entrar' (Enter) button is at the bottom, followed by a link 'Criar uma conta' (Create an account).

1. Login / Registro

The screenshot displays a list of open tickets under the heading 'Meus Chamados'. Each ticket entry includes a status ('Status: Aberto'), date ('Data: ...'), and a 'Avaliar Fornecedores' (Evaluate Suppliers) button.

2. Abertura de Chamado

The screenshot shows the 'Registrar' (Create Account) page with the title 'Crie sua conta'. It contains various input fields for personal information: Nome completo, E-mail, Senha, Telefone, Logradouro, Número, Complemento, Bairro, Cidade, Estado, and CEP. At the bottom are two buttons: 'Criar Conta' (Create Account) in white on a dark background, and 'Voltar' (Back) in white on a light background.

3. Registro

The screenshot shows the 'Avaliar Chamado' (Evaluate Ticket) page. It has a section for 'Nota' (Rating) with five stars, a 'Comentário' (Comment) field with placeholder text 'Deixe seu comentário (opcional)', and two buttons at the bottom: 'Enviar Avaliação' (Send Evaluation) in white on a dark background, and 'Cancelar' (Cancel) in white on a light background.

4. Avaliação do Atendimento

131

Demonstração do Sistema



Empreendedorismo e Relações

Empreendedorismo

- **Público-Alvo:** Foco em PMEs que necessitam otimizar o suporte técnico interno.
- **Viabilidade:** Baixo custo de desenvolvimento utilizando tecnologias consolidadas (C#, SQL Server).
- **Modelo de Negócio:** Estrutura flexível (Licenciamento, Assinatura Mensal) para garantir escalabilidade.
- **Inovação:** Uso de IA como diferencial competitivo para automação e redução de custos operacionais.

Relações Étnico-Raciais e Inclusão

- **Linguagem:** Adoção de linguagem neutra e acolhedora no design e conteúdo, livre de estereótipos.
- **Identidade:** Implementação da funcionalidade de uso de **Nome Social**, respeitando identidades individuais.
- **Acessibilidade:** Conformidade com padrões WCAG (já visto em Qualidade).
- **Ética na IA:** Diretrizes para evitar vieses discriminatórios nas respostas geradas pela IA.

Obrigado.

Perguntas?

