Instituto Federal da Bahia
Departamento de Computação

Programa de Pós-Graduação em Engenharia de Sistemas de Produtos

TICKETEASE - INTEGRAÇÃO ENTRE PLATAFORMAS DE ATENDIMENTOS TÉCNICOS

Wiler Sávio Almeida Moraes

RELATÓRIO TÉCNICO

Salvador
06 de maio de 2024

WILER SÁVIO ALMEIDA MORAES

TICKETEASE - INTEGRAÇÃO ENTRE PLATAFORMAS DE ATENDIMENTOS TÉCNICOS

Este Relatório Técnico foi apresentado ao Programa de Pós-Graduação em Engenharia de Sistemas de Produtos do Instituto Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia de Sistemas e Produtos.

Orientador: Prof. Dr. Manoel Carvalho Marques Neto

Salvador
06 de maio de 2024

FICHA CATALOGRÁFICA ELABORADA PELO SISTEMA DE BIBLIOTECAS DO IFBA, COM OS DADOS FORNECIDOS PELO(A) AUTOR(A)

M827t Moraes, Wiler Sávio Almeida

Ticketease: integração entre plataformas de atendimentos técnicos / Wiler Sávio Almeida Moraes; orientador Manoel Carvalho Marques Neto -- Salvador, 2024.

82 p.

Relatório técnico (Programa de Pós-Graduação em Engenharia de Sistemas de Produtos) -- Instituto Federal da Bahia, 2024.

1. ITSM. 2. Atendimentos. 3. Aplicação. 4. Mobile. 5. Web. I. Marques Neto, Manoel Carvalho, orient. II. TÍTULO.

CDU 004.457

INSTITUTO FEDERAL DA BAHIA

PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE SISTEMAS E PRODUTOS – PPGESP

TICKETEASE - INTEGRAÇÃO ENTRE PLATAFORMAS DE ATENDIMENTOS TÉCNICOS

WILER SÁVIO ALMEIDA MORAES

Produto(s) Gerado(s): Relatório Técnico e Registro de Software.

Orientador: Prof. Dr. Manoel Carvalho Marques Neto

Banca examinadora:

Prof. Dr. Manoel Carvalho Marques Neto

Orientador – Instituto Federal da Bahia (IFBA)

Prof. Dr. Leobino Nascimento Sampaio

Membro Externo – Universidade Federal da Bahia (UFBA)

Prof. Dr. Allan Edgard Silva Freitas

Membro Interno – Instituto Federal da Bahia (IFBA)

LHA DE APROVAÇÃO DEFESA WILER SÁVIO ALMEIDA MORAES_ Defesa PPGESP em 11/06 (4221945) SEI 23279.006460/2

Trabalho de Conclusão de Curso aprovado pela banca examinadora em 11/06/2025.

Documento assinado eletronicamente por **MANOEL CARVALHO MARQUES NETO**, **Professor Titular** , em 11/06/2025, às 16:11, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **ALLAN EDGARD SILVA FREITAS**, **Professor Titular** , em 11/06/2025, às 16:13, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **Leobino Nascimento Sampaio**, **Usuário Externo**, em 17/06/2025, às 12:44, conforme decreto nº 8.539/2015.

A autenticidade do documento pode ser conferida no site http://sei.ifba.edu.br/sei/controlador_externo.php?acao=documento_conferir&acao_origem=documento_conferir&id_orgao_acesso_externo=0 informando o código verificador **4221945** e o código CRC **C6598004**.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus por ser minha fonte de força e inspiração durante a preparação deste trabalho. Sua presença em minha vida me ajuda a superar dificuldades e encontrar o caminho certo para alcançar meus objetivos.

Ao Instituto Federal da Bahia, pela contribuição fundamental para o meu crescimento, seja por meio dos ensinamentos e conselhos dos professores, seja pelas dicas e sugestões ao longo das disciplinas. As aulas foram fenomenais, e pude absorver muitos conhecimentos transmitidos pelos docentes.

Ao Prof. Dr. Manoel Carvalho Neto, por todo o suporte oferecido, desde meu ingresso na instituição até a conclusão deste trabalho. Graças às suas orientações e contribuições, consegui chegar ao final desta etapa.

À minha família, por me dar forças nas adversidades e me ajudar a conciliar os estudos com a vida profissional e pessoal. Acredito que, sem o apoio de vocês, talvez eu não tivesse conseguido finalizar o mestrado.

A mente que se abre a uma nova ideia, jamais voltará ao seu tamanho original.

—ALBERT EINSTEIN

RESUMO

O ambiente corporativo atual faz amplo uso de tecnologias que requerem monitoramento e manutenção constantes, abrangendo tanto *hardware* quanto *software*. Essas atividades devem ser executadas por profissionais com conhecimento especializado para evitar impactos negativos nos resultados do negócio, como atrasos no atendimento e problemas não resolvidos, que podem resultar em perdas financeiras e insatisfação dos clientes. Existem diversas ferramentas disponíveis para manter e gerenciar esse aparato tecnológico, muitas das quais classificam os atendentes de acordo com seu nível de conhecimento técnico, direcionando o grupo mais apropriado para resolver problemas específicos e garantindo que os atendimentos sejam realizados de forma eficiente e eficaz. No entanto, a principal limitação dessas ferramentas é a falta de integração entre elas, uma vez que muitas são concorrentes e operam de maneira isolada, dificultando a fluidez das informações e a comunicação entre as equipes. Nesse contexto, o TicketEase foi desenvolvido para oferecer uma integração entre diferentes ferramentas de ITSM, possibilitando que os técnicos tenham acesso a uma visão unificada dos chamados e demandas, além de proporcionar inúmeras facilidades de uso e alto potencial de customização, tornando a ferramenta mais alinhada às necessidades do negócio e facilitando o trabalho diário dos profissionais. A customização vai além de ajustes em dados e relatórios, permitindo integrações com outras plataformas de ITSM de forma menos impactante, o que significa que a adaptação à realidade de cada empresa pode ser realizada de maneira ágil e eficiente. O TicketEase também conta com uma interface *mobile* para técnicos, permitindo que eles acessem informações e atualizem status em tempo real, e uma interface web para gestores acompanharem indicadores e métricas de desempenho. A análise dos resultados deste relatório mostra que, embora o TicketEase tenha um significativo espaço para melhorias e customizações, ele já apresenta funcionalidades importantes que podem impactar positivamente o atendimento e a integração, oferecendo uma solução prática e eficiente para as demandas do ambiente corporativo contemporâneo. Assim, o TicketEase se posiciona como uma ferramenta inovadora, capaz de atender às exigências do mercado e contribuir para a excelência no suporte técnico.

Palavras-chave: ITSM. Atendimentos. Aplicação. Mobile. Web.

ABSTRACT

The current corporate environment extensively utilizes technologies that require constant monitoring and maintenance, covering both hardware and software. These activities must be performed by professionals with specialized knowledge to avoid negative impacts on business results, such as delays in service and unresolved issues, which can lead to financial losses and customer dissatisfaction. Various tools are available to maintain and manage this technological framework, many of which classify service providers based on their technical expertise, directing the most suitable group to address specific problems and ensuring that services are rendered efficiently and effectively. However, the main limitation of these tools is the lack of integration among them, as many are competitors and operate in isolation, hindering the flow of information and communication among teams. In this context, TicketEase was developed to provide integration between different ITSM tools, enabling technicians to access a unified view of requests and demands while offering numerous usability features and high customization potential, making the tool more aligned with business needs and facilitating the daily work of professionals. Customization goes beyond adjustments in data and reports, allowing for integrations with other ITSM platforms in a less impactful manner, meaning that adaptation to the reality of each company can be carried out quickly and efficiently. TicketEase also includes a mobile interface for technicians, allowing them to access information and update statuses in real time, as well as a web interface for managers to monitor performance indicators and metrics. The analysis of the results of this report shows that, although TicketEase has significant room for improvements and customizations, it already presents important functionalities that can positively impact service and integration, offering a practical and efficient solution for the demands of the contemporary corporate environment. Thus, TicketEase positions itself as an innovative tool capable of meeting market demands and contributing to excellence in technical support.

Keywords: ITSM. Services. Application. Mobile. Web.

SUMÁRIO

Capítulo 1—Introdução 1

1.1 Objetivos . . . 3
1.1.1 Objetivo Geral . . . 3
1.1.2 Objetivos Específicos . . . 3

Capítulo 2—Revisão Bibliográfica 5

2.1 *IT Service Management* (ITSM) . . . 5
2.1.1 GLPI . . . 5
2.1.2 CA *Service Management* . . . 7
2.1.3 OTRS . . . 8
2.1.4 *Atlassian Jira Service Desk* . . . 9
2.1.5 Compartivo entre Ferramentas de ITSM . . . 11
2.2 Arquitetura de Software . . . 12
2.2.1 Arquitetura de microsserviços . . . 13
2.2.1.1 A escolha da arquitetura . . . 13
2.2.1.2 Vantagens . . . 14
2.2.1.3 Desvantagens . . . 14
2.2.1.4 Dimensionamento . . . 15
2.2.1.5 Resiliência . . . 16
2.2.1.6 Integração contínua e implantação contínua . . . 16
2.2.2 Arquitetura Monolítica . . . 17
2.2.3 Padrão *Publish/Subscribe* (Pub/Sub) . . . 18
2.2.4 Padrão *Back-end for frontend* (BFF) . . . 19
2.2.5 *Application Programming Interface* (API) . . . 19
2.2.6 Aplicativos *Mobile* . . . 20
2.2.7 Aplicações Web . . . 21
2.3 *Cloud Computing* . . . 21
2.3.1 Azure *Cloud Computing* . . . 21
2.3.2 Configuração Remota do Firebase . . . 22

Capítulo 3—Metodologia do Sistema 25

3.1 Fluxo do Chamado . . . 26
3.2 Arquitetura da aplicação . . . 27
3.3 Microsserviço Chamado . . . 29
3.4 Microsserviço Rotas . . . 29

3.5 Microsserviço Relatórios e Configurações 29
3.6 Microsserviço Prestação de Contas 30
3.7 Microsserviço Acessos 30
3.8 Microsserviço Integração 31
3.9 ITSM Externo 31

Capítulo 4—Prova de Conceito 33

4.1 Dados quantitativos 33
4.2 Fluxo de atendimento 34
4.3 Impactos observados 35

Capítulo 5—Resultados e Discussões 37

5.1 Aplicação *Mobile* 37
5.1.1 Tela Inicial 37
5.1.2 Chamados 38
5.1.3 Meus chamados 39
5.1.4 Rotas 40
5.1.5 Prestação de contas 41
5.1.6 Checklist do veículo 41
5.1.7 Meus veículos 42
5.1.8 Meus acessos 43
5.2 Aplicação Web 44
5.2.1 Tela Inicial 44
5.2.2 Notificações 44
5.2.3 Cadastro de Clientes 45
5.2.4 Cadastro de técnicos 46
5.2.5 Prestação de contas 47
5.2.6 Checklist 48
5.2.7 Relatórios 49
5.3 Cronograma de Desenvolvimento 50
5.3.1 Sprint 1 – Estrutura Inicial 50
5.3.2 Sprint 2 – Integração com Plataformas de ITSM 51
5.3.3 Sprint 3 – Funcionalidades de "Meus Chamados 51
5.3.4 Sprint 4 – Prestação de Contas e "Meus Veículos 51
5.3.5 Sprint 5 – Checklist dos Veículos e Configurações 52
5.4 Resultados 52

Capítulo 6—Conclusão 55

LISTA DE FIGURAS

2.1 Abertura de chamado no GLPI 6
2.2 Exemplo de um chamado no CA *Service Management* 8
2.3 Abertura de chamado no OTRS 9
2.4 Abertura de um chamado no Jira Service Desk 10
2.5 Tipos de dimensionamento (LIMA, 2024) 15
2.6 CI/CD (REDHAT, 2023) 17
2.7 Funcionamento de uma fila (MICROSOFT, 2024c) 22

3.1 Fluxo dos chamados 26
3.2 Arquitetura TicketEase 27

5.1 Tela inicial do sistema 38
5.2 Tela de chamados 39
5.3 Tela Meus Chamados 40
5.4 Tela das rotas do técnico 40
5.5 Tela de prestação de contas 41
5.6 Tela de checklist do veículo 42
5.7 Tela Meus Veículos 43
5.8 Tela Meus Acessos 43
5.9 Tela Inicial 44
5.10 Tela de Notificações 45
5.11 Tela de cadastro de clientes 46
5.12 Tela de cadastro de técnicos 47
5.13 Tela de prestação de contas 48
5.14 Tela de checklist 49
5.15 Tela dos relatórios 50

LISTA DE TABELAS

2.1 Comparação entre diferentes ferramentas ITSM 12

LISTA DE SIGLAS

ITSM	*IT Service Management*	5
TI	Tecnologia da Informação	6
GLPI	*Gestionnaire Libre Parc Informatique*	5
SLA	*Service Level Agreement*	6
API	*Application Programming Interface*	55
DDD	*Domain Driven Design*	14
CI/CD	*Continuous Integration/Continuous Delivery*	16
BFF	*Back-end for frontend*	55
HTTP	*Hypertext Transfer Protocol*	19
REST	*Representational State Transfer*	19
PaaS	*Platform-as-a-Service*	21
AD	*Active Directory*	37
SSO	*Single Sing On*	27
PDF	*Portable Document Format*	30
SOAP	*Simple Object Access Protocol*	19
OTRS	*Open-source Ticket Request System*	3
IoT	*Internet of Things*	18
Pub/Sub	*Publish/Subscribe*	18

Capítulo 1

INTRODUÇÃO

Atualmente, é visível o grande uso de tecnologias nos ambientes empresariais. Por meio de celulares, notebooks, computadores de mesa, dentre tantos outros dispositivos, observa-se a dependência das instituições nesses aparelhos tecnológicos. Mas como manter toda essa tecnologia em constante funcionamento? Essa é uma indagação que as empresas têm a cada momento (PENSO, 2017).

Uma das alternativas é manter uma equipe interna de TI para atender a todas as demandas da corporação. No entanto, essa é a opção mais cara, pois, além dos custos com pessoal e encargos trabalhistas, haverá despesas com a gestão de departamentos, aquisição de insumos, deslocamentos e outros fatores que podem não ser identificados de imediato (PENSO, 2017).

A alternativa mais econômica é terceirizar o setor de TI, delegando suas tarefas a uma empresa parceira. Com essa alternativa, haverá diminuição dos custos fixos e operacionais se comparados à equipe interna. Além disso, a empresa contratante poderá contar com mais inovações tecnológicas, visto que a empresa terceirizada estará em constante contato com as novas tendências para se manter competitiva (PENSO, 2017).

Independentemente da abordagem escolhida, algumas questões surgem: como acompanhar os fluxos e avaliar se os atendimentos estão sendo realizados dentro do padrão esperado? Como garantir que os atendentes ou técnicos tenham melhor acesso aos chamados, mesmo durante deslocamentos? Como evitar uma dependência excessiva nas equipes terceirizadas?

Atualmente, essas dúvidas podem ser respondidas com a utilização de ferramentas denominadas *IT Service Management* (ITSM). Tais ferramentas fornecem funcionalidades para criar, transferir, manter e consultar chamados de atendimentos, aumentando assim a qualidade e tendo um controle de todo o fluxo do processo. Além disso, essas ferramentas criam uma base de conhecimento para resolução de problemas frequentes, reduzindo o tempo de solução (ALBERTIN; ALBERTIN, 2008).

As empresas terceirizadas de atendimentos ou até mesmo o setor interno de atendimento podem ter a necessidade de trabalhar com diversos ITSM's, aumentando a complexidade para seus técnicos. Para realizar os seus atendimentos, os técnicos deverão

acessar várias plataformas, visto que clientes diferentes podem possuir plataformas diferentes. No caso de setores internos, filiais diferentes podem possuir plataformas diferentes (ALBERTIN; ALBERTIN, 2008).

Buscando mitigar o problema apresentado, este relatório apresenta a TicketEase, uma ferramenta que atua como um centralizador de atendimentos, eliminando a necessidade de acesso direto à plataforma ITSM. Com o TicketEase, todo o fluxo de atendimento pode ser realizado de forma unificada. Além disso, a ferramenta oferece opções de personalização, permitindo a geração de relatórios e indicadores para acompanhamento de resultados.

Outra característica de destaque do TicketEase é sua arquitetura baseada em microsserviços. Essa abordagem não apenas reforça a escalabilidade e a flexibilidade da ferramenta, mas também reduz significativamente o impacto de falhas, uma vez que cada componente opera de forma independente, garantindo maior robustez ao sistema. Além disso, a separação em microsserviços facilita a implementação de novas funcionalidades e atualizações, permitindo que cada módulo seja desenvolvido, testado e implantado de forma isolada. Assim, o TicketEase adapta-se rapidamente às demandas do usuário, mantendo altos níveis de desempenho e disponibilidade.

Existem alguns pré-requisitos que devem ser seguidos pelas empresas para que elas possam utilizar o TicketEase, como possuir um *Active Directory* (AD) para controle dos usuários e criar as configurações do ITSM dentro da aplicação. Após o cumprimento dessas condições, todo o restante do processo é gerenciado pelo *software*, desde a integração até retentativas de integrações com o ITSM em caso de falhas.

Além disso, o TicketEase oferece duas interfaces de acesso. A mais utilizada será a aplicação *mobile*, que será acessada pelos técnicos para manusear os chamados. A outra interface será a aplicação web que será utilizada somente pela área administrativa para acompanhar todo o fluxo de atendimento e visualizar as métricas. Este trabalho enfatiza a aplicação mobile, embora a versão web também seja apresentada para proporcionar um entendimento geral.

Alguns termos utilizados nesse trabalho foram consultados com base no dicionário Michaelis. Tais termos são de etimologia inglesa, mas foram incorporados à língua portuguesa no decorrer dos últimos anos (MICHAELIS, 2024).

Por fim, o texto deste trabalho está organizado da forma que se segue. O Capítulo 2 apresenta a revisão bibliográfica. Neste capítulo, o leitor poderá aprofundar os seus conhecimentos em algumas ferramentas ITSM, conceitos de arquitetura de *software* e funcionalidades *cloud computing*. No Capítulo 3, o *software* TicketEase é mais detalhado em relação à sua arquitetura e microsserviços. No Capítulo 4, é apresentada a prova de conceito do TicketEase, destacando os dados quantitativos, o fluxo de atendimento e os impactos observados. No Capítulo 5, são demonstrados os resultados do desenvolvimento do *software*. Finalmente, no Capítulo 6, apresentam-se as conclusões preliminares obtidas com este trabalho.

1.1 OBJETIVOS

Para a realização deste trabalho, definiu-se um objetivo geral que guiou todas as ações e decisões tomadas e que foi base para a definição dos objetivos específicos a serem atingidos.

1.1.1 Objetivo Geral

Criar um aplicativo *mobile* para gerenciamento de atendimentos técnicos com integrações entre as plataformas *Gestionnaire Libre Parc Informatique* (GLPI) e *Open-source Ticket Request System* (OTRS).

1.1.2 Objetivos Específicos

Os objetivos específicos são apresentados a seguir:

- Automatizar a criação de rotas de atendimento;
- Possibilitar que novas integrações sejam realizadas sem grandes mudanças no sistema;
- Criar um sistema para gerenciar as métricas e relatórios dos técnicos.

Capítulo 2

Neste capítulo são apresentados os principais conceitos que fundamentaram o desenvolvimento deste trabalho. Ele está dividido em três seções: na primeira são apresentados os conceitos de ITSM, alguns exemplos de ITSM's do mercado e suas principais vantagens e desvantagens; em seguida, são apresentados alguns conceitos de arquitetura de software que foram amplamente aplicados e, por fim, são apresentados conceitos relacionados a cloud computing que foram levantados e incorporados à arquitetura do sistema.

REVISÃO BIBLIOGRÁFICA

2.1 IT Service Management (ITSM)

A sigla ITSM vem do inglês *Information Technology Service Management* que, traduzido para o português, significa "Gerenciamento de Serviços de TI". É amplamente utilizada para designar ferramentas que abrangem todo o fluxo do planejamento, execução e monitoramento dos serviços de TI (ZENDESK, 2023).

Nesta seção, serão apresentados os principais *softwares* de ITSM disponíveis no mercado, mostrando seus principais benefícios. Por fim, será apresentada uma tabela comparativa entre todos os *softwares* ITSM discutidos, destacando suas principais características.

2.1.1 GLPI

O nome *Gestionnaire Libre Parc Informatique* (GLPI) é proveniente do francês que, traduzido para o português, significa "Gerenciamento gratuito de ativos de TI". Sua história começa em 2003 com uma equipe de desenvolvedores sem fins lucrativos que, baseada em um *software* já existente de código aberto, o IRMA, iniciou uma modificação para atender requisitos específicos (GLPI, 2023).

O GLPI é desenvolvido utilizando a linguagem PHP com o banco de dados MySQL. À medida que obtinha aumento de popularidade, novas funcionalidades e recursos eram adicionados. Um dos recursos interessantes na época foi o suporte para módulos e *plug-ins* que possibilitaram um desenvolvimento versátil para uma série de organizações (GLPI, 2023).

Apesar de ser um *software* livre, possui serviços que possibilitam aos usuários apoiar financeiramente o projeto. Todo esse financiamento é usado exclusivamente para aumentar e melhorar a equipe de desenvolvimento (GLPI, 2023).

O sistema GLPI possui hoje grandes clientes em diversos ramos. Alguns exemplos são: Deezer, Airbus, Burger King France, dentre outros (GLPI, 2023).

As principais vantagens do GLPI são:

1. Possibilidade de criar catálogo de serviços, facilitando a solicitação e acesso a recursos de Tecnologia da Informação (TI);
2. Gestão de incidentes, sendo possível registrar e acompanhá-los;
3. Gestão de problemas recorrentes;
4. Gestão de níveis de serviços, também chamados de *Service Level Agreement* (SLA);
5. Comunidade ativa (LEANDRO, 2023).

Apesar de possuir inúmeras funcionalidades, esse trabalho tem como objetivo focar no controle de atendimentos. Sendo assim, toda a infraestrutura do GLPI será utilizada para controlarmos os atendimentos técnicos, visando uma rápida resolução dos problemas.

O chamado inicialmente é aberto no sistema GLPI e atribuído para o suporte inicial, chamado de N1. Na tela de abertura, o usuário requerente informa o título do chamado, bem como uma descrição do problema, podendo também anexar arquivos para uma melhor identificação do mesmo. Além disso, o usuário deverá selecionar a categoria do chamado e a sua localização para possíveis atendimentos presenciais.

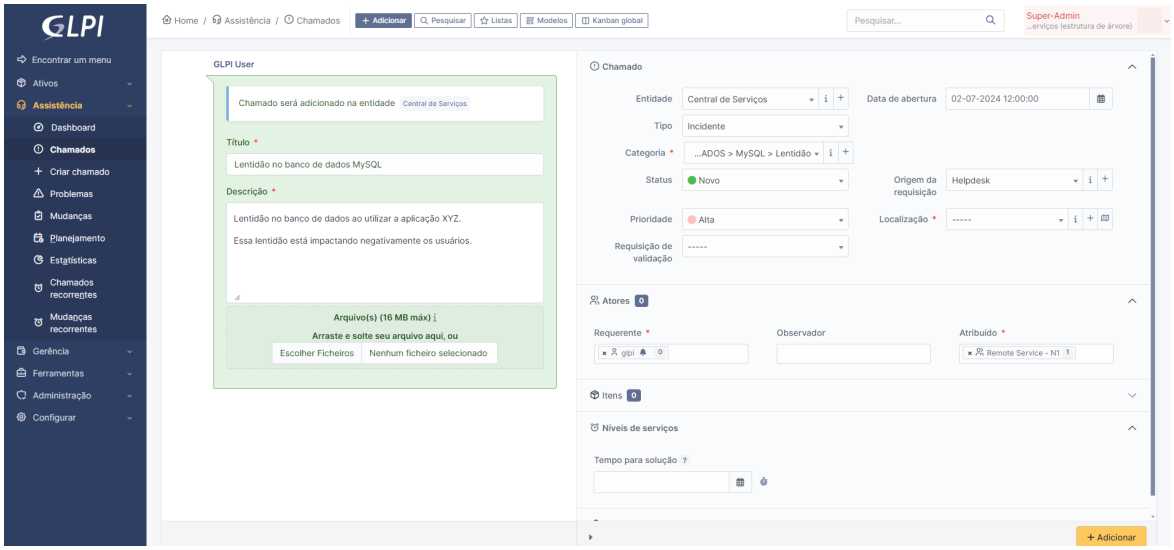

Figura 2.1 Abertura de chamado no GLPI

Após a abertura do chamado, ele é automaticamente exibido na lista de chamados do suporte N1. É exibida também uma barra de progresso informando o tempo restante para a resolução do problema, baseado no SLA informado.

A equipe de suporte N1 realizará os procedimentos necessários para tentar resolver o problema. Caso não seja possível solucioná-lo, o chamado será encaminhado para o setor especializado, acompanhado de uma descrição detalhada das tentativas de resolução e da necessidade de atendimento especializado. Se necessário, o SLA também poderá ser ajustado.

2.1.2 CA Service Management

O CA *Service Management* é um *software* desenvolvido pela empresa Broadcom que possibilita o gerenciamento de incidentes, problemas, mudanças e ativos de TI por meio de uma única aplicação. Para facilitar a integração com outros sistemas, oferece diversas APIs que permitem essa conexão. Além disso, conta com integrações pré-existentes com plataformas proprietárias, otimizando ainda mais a gestão de serviços (BROADCOM, 2022).

Alguns dos benefícios do CA *Service Management* para os usuários corporativos são:

- Interface *xFlow* - fornece exibições abrangentes para que analistas e equipes de atendimento possam oferecer um excelente serviço de atendimento;
- *Service Point* - solução utilizando inteligência artificial para criar, exibir, modificar tickets e acessar bases de conhecimento;
- *Insights* - grande variedade de relatórios e possibilidade de geração de relatórios customizados (BROADCOM, 2022).

Diferente do GLPI, o CA *Service Management* é uma solução proprietária, ou seja, o seu código-fonte não está disponível para alteração pela comunidade. Quaisquer novas funcionalidades devem partir da empresa mantenedora, no caso, a Broadcom. Além disso, o *CA Service Management* é uma ferramenta que requer licenciamento, o que representa um desafio na obtenção de exemplos práticos e imagens de implementação, já que o acesso a recursos é restrito a usuários licenciados (BROADCOM, 2022).

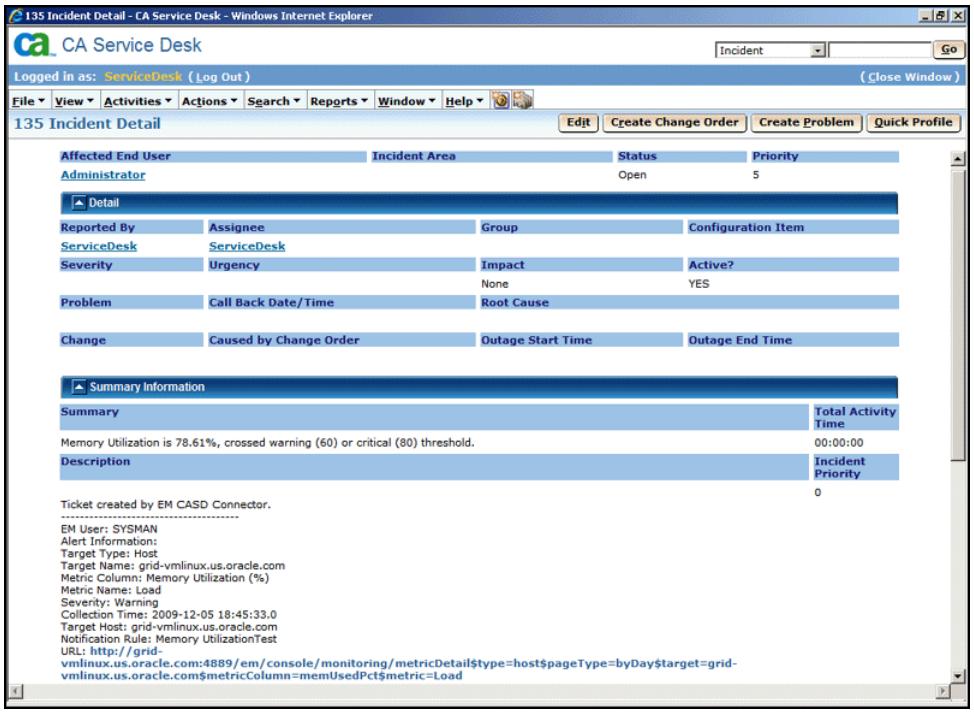

Figura 2.2 Exemplo de um chamado no CA *Service Management* (ORACLE, 2019)

2.1.3 OTRS

O nome OTRS vem do inglês *Open-source Ticket Request System* que, traduzido para o português, seria algo como "Sistema de Solicitação de Tickets de Código Aberto". Sua história começa em 2001 como uma iniciativa *open-source*, sem fins lucrativos. Em 2003, o projeto assumiu características mais profissionais, levando à fundação da empresa *OTRS GmbH*. Em 2007, essa empresa transformou-se na *OTRS Group*, proprietária e mantenedoura do código fonte do sistema OTRS (OTRS, 2020).

O OTRS possui uma estrutura flexível, fornecendo as seguintes funcionalidades:

- Filas de atendimento para diferentes times de operação;
- Organização dos fluxos de trabalho com aprovações;
- Controle de SLA;
- Respostas padronizadas;
- Relatórios e *dashboards*;
- Indicadores de negócio;
- Pesquisa de satisfação automática (INTEROP, 2018).

Por ser uma iniciativa *open-source*, o OTRS possui uma versão totalmente gratuita, sem custos de licenciamento. No entanto, se necessário, a *OTRS Group* e seus parceiros oferecem suporte pago para a ferramenta, o que pode ser ideal para usos em larga escala (INTEROP, 2018).

O fluxo de um chamado no OTRS inicia com a abertura feita pelo usuário. Na tela de criação do chamado, o usuário deve especificar o setor responsável, inserir um assunto que facilite a identificação e descrever detalhadamente o problema. Além disso, é possível anexar arquivos para auxiliar na análise do problema e definir a prioridade do chamado, permitindo que a equipe de suporte priorize casos mais urgentes.

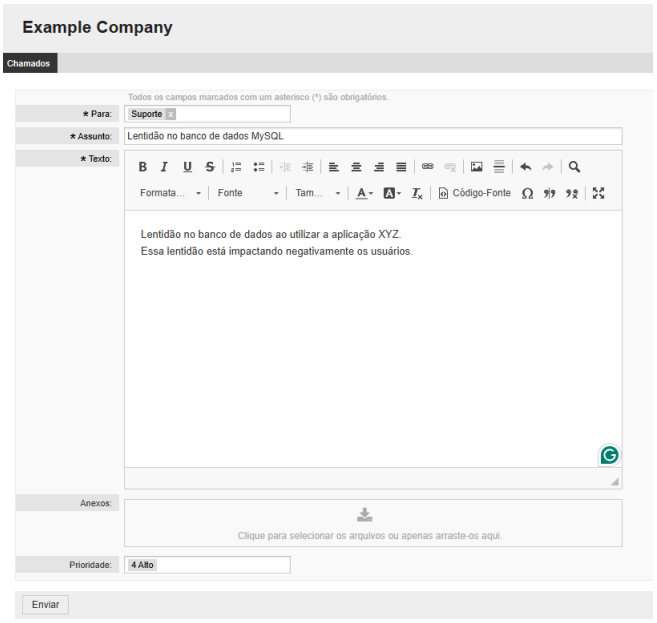

Figura 2.3 Abertura de chamado no OTRS

Após a abertura do chamado pelo usuário no OTRS, o chamado será encaminhado para a fila do setor responsável. A posição do chamado na fila depende de sua prioridade: chamados com maior urgência serão posicionados no topo, enquanto aqueles de menor prioridade ficarão na parte inferior.

O atendimento inicial será realizado pela equipe de suporte N1, responsável por resolver problemas de menor complexidade. Caso a equipe não consiga solucionar o problema, o chamado será encaminhado para a fila de atendimento especializado, acompanhado de uma descrição detalhada do problema e das tentativas de resolução realizadas. Além disso, o usuário será informado de que o atendimento será feito de forma presencial.

No OTRS, o usuário tem visibilidade completa de todas as interações realizadas pelos técnicos, incluindo as tentativas de solução e os problemas identificados. Além disso, o usuário pode, a qualquer momento, adicionar novas informações ao chamado ou encerrá-lo, caso o problema tenha sido resolvido.

2.1.4 Atlassian Jira Service Desk

A empresa Atlassian nasceu em 2002 por meio de dois colegas de faculdade determinados a criar um negócio autêntico. O nome Atlassian é inspirado no titã Atlas, da mitologia grega. Seu primeiro produto, o Jira, voltado para gerenciamento de equipes, foi lançado ainda em 2002. No ano seguinte, a empresa lançou o Confluence, uma plataforma

de colaboração destinada ao compartilhamento de documentos. Em 2010, a Atlassian expandiu sua atuação ao adquirir o BitBucket, uma ferramenta de hospedagem de código e colaboração baseada em Git (STACK, s.d.).

Em 2013, a Atlassian lançou o Jira Service Desk ao perceber que muitos clientes estavam adaptando a ferramenta Jira para gerenciar solicitações de serviço. Em um levantamento realizado, cerca de 40% dos clientes já utilizavam o Jira de forma adaptada para esse fim (ATLASSIAN, s.d.).

O Jira Service Desk possui diversas funcionalidades, entre elas:

- Gerenciamento de várias equipes;
- Gestão de SLA;
- Personalização de e-mails;
- Métricas e relatórios;
- Integração com os produtos da Atlassian;
- *Workflow* personalizável (HORLLE, 2021).

O fluxo no Jira Service Desk começa com a abertura de um chamado pelo usuário, que deve inserir um assunto e uma descrição detalhada do problema. Além disso, o usuário pode selecionar os serviços afetados, anexar imagens que ajudem na identificação do problema e definir a urgência e o impacto da situação. Essas informações são essenciais para definir o SLA adequado ao chamado.

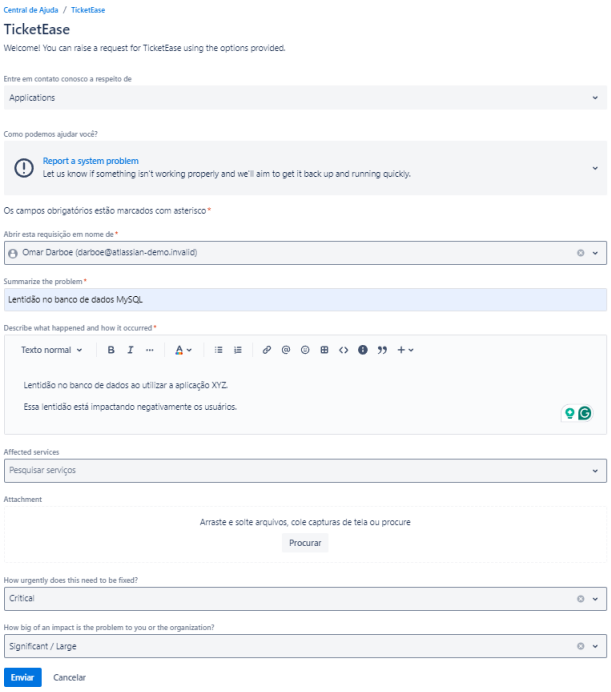

Figura 2.4 Abertura de um chamado no Jira Service Desk

Após a abertura do chamado pelo usuário no Jira Service Desk, ele é encaminhado para uma fila genérica até ser categorizado por um membro da equipe de suporte. Na interface

de suporte, os técnicos podem visualizar os chamados mais urgentes e organizá-los de acordo com o SLA, priorizando o atendimento de acordo com os prazos estabelecidos.

Quando o membro da equipe de suporte acessa o chamado, ele tem acesso a todas as informações fornecidas pelo usuário, além de um histórico das atividades realizadas por outros membros da equipe. São exibidos dados relevantes, como a urgência e o impacto do problema, o responsável atual pelo chamado, entre outras informações essenciais para o acompanhamento e resolução do problema.

Após realizar todas as verificações preliminares, o membro da equipe de suporte deve fornecer um retorno ao cliente. Esse retorno pode consistir em uma solução para o problema ou na notificação de que o chamado será transferido para um nível mais avançado de análise.

O cliente poderá visualizar todas as respostas fornecidas pelos membros da equipe de suporte, bem como as notificações sobre a mudança de status do chamado. Além disso, o cliente terá a opção de adicionar informações complementares às respostas da equipe de suporte, fornecendo detalhes adicionais ou solicitando ações específicas.

2.1.5 Compartivo entre Ferramentas de ITSM

A tabela a seguir oferece uma comparação detalhada entre quatro das principais ferramentas de ITSM: OTRS, Jira, GLPI, e CA *Service Management*. A comparação abrange aspectos essenciais, como o licenciamento de uso, a flexibilidade na customização das funcionalidades, a escalabilidade em diversos ambientes organizacionais e a facilidade de implementação e utilização. Além disso, o suporte técnico disponível para cada solução é avaliado.

O objetivo desta análise é fornecer uma visão abrangente e comparativa das principais características de cada uma dessas ferramentas, destacando seus pontos fortes e possíveis limitações. Com isso, busca-se facilitar a compreensão de como essas soluções podem ser aplicadas para atender a diferentes demandas e necessidades no contexto da gestão de serviços de TI. Essa análise auxilia gestores na escolha da plataforma mais adequada para suas operações.

Características	GLPI	OTRS	Jira	Ca Service
Licenciamento	*Open-Source*	*Open-Source* (com versão paga)	Licença paga (com versão gratuita limitada)	Licença paga
Plataforma	Web	Web	Web	Web
Gerenciamento de Incidentes	Sim	Sim	Sim	Sim
Gerenciamento de Problemas	Sim	Sim	Sim	Sim
Gerenciamento de Mudanças	Sim	Sim	Sim	Sim
Gerenciamento de Ativos	Nativo	Possível com plugins	Possível com plugins	Nativo
Automação de Fluxos de Trabalho	Sim	Sim	Sim	Sim
Customização	Alta (suporte a plugins e módulos)	Alta (plugins e integrações)	Alta (personalizável através do Jira Workflows)	Alta (integrações e customizações)
Integração com ferramentas externas	Sim (LDAP, Active Directory, API)	Sim (LDAP, Active Directory, API)	Sim (integrações com ferramentas Atlassian e outras)	Sim (API e integrações)
Interface de Usuário	Simples e funcional	Funcional mas não tão moderna	Moderna e intuitiva	Funcional mas interface clássica
Relatórios e Dashboards	Relatórios customizados	Relatórios customizados	Dashboard avançado	Relatórios detalhados
Suporte Técnico	Comunidade e suporte pago	Comunidade e suporte pago	Suporte pago (opcional)	Suporte pago
Implementação	Rápida e simples	Relativamente simples	Moderada (dependendo da customização)	Mais complexa

Tabela 2.1 Comparação entre diferentes ferramentas ITSM

2.2 ARQUITETURA DE SOFTWARE

Nesta seção serão apresentadas as escolhas arquiteturais e o motivo de ter escolhido determinada opção. Primeiramente, é apresentada a arquitetura de microsserviços. Em seguida, será discutido o padrão *Back-end for frontend* (BFF) em conjunto com a ar-

quitetura de microsserviços anterior. Por último, será discutido o desenvolvimento de aplicação incluindo a parte da API, *mobile* e web.

2.2.1 Arquitetura de microsserviços

Esta seção aborda a arquitetura de microsserviços, detalhando os motivos que fundamentaram sua escolha para o projeto. Serão apresentadas as principais vantagens desse modelo, como a diminuição do código-fonte e a escalabilidade, além de suas limitações. Também serão discutidos os desafios enfrentados na sua implementação, incluindo o dimensionamento da aplicação, a garantia de resiliência diante de falhas e a adoção de práticas como integração e implantação contínuas.

2.2.1.1 A escolha da arquitetura

Inicialmente, considerou-se a possibilidade de desenvolver uma aplicação monolítica, no qual todos os componentes estariam integrados em uma única solução. Essa abordagem é, atualmente, a menos complexa em comparação com arquiteturas mais modernas. No entanto, a necessidade de modularizar as funcionalidades e reduzir os custos de implantação tornou a adoção da arquitetura de microsserviços a opção mais adequada para o projeto.

Segundo Sam Newman (NEWMAN, 2015), microsserviços são pequenos serviços independentes que trabalham juntos. Isso significa que a aplicação é dividida em serviços menores, cada um responsável por uma única funcionalidade dentro de um contexto mais amplo. Essa abordagem facilita a manutenção, pois o volume de código é reduzido, resultando em menor complexidade e maior facilidade de evolução.

Devido ao seu tamanho reduzido, os microsserviços reforçam o princípio da responsabilidade única, descrito nos princípios do SOLID, que estabelece que cada componente deve ter uma única responsabilidade e, consequentemente, uma única razão para ser modificado (NEWMAN, 2015).

O SOLID descreve cinco princípios da orientação de objetos e design de código. Tais princípios auxiliam na melhoria da criação de códigos, possibilitando a criação de códigos mais limpos. Os princípios são:

1. Princípio da responsabilidade única: uma classe deve ter apenas uma responsabilidade e uma única razão para mudar;

2. Princípio aberto-fechado: entidades devem estar abertas para extensão, mas fechadas para modificação;

3. Princípio da substituição de Liskov: as subclasses devem ser substituíveis por sua classe base;

4. Princípio da segregação de interfaces: os clientes não devem ser forçados a depender de interfaces que não usam;

5. Princípio da inversão de dependência: os módulos devem depender de abstrações e não de implementações concretas (PAIXÃO, 2019).

2.2.1.2 Vantagens

Uma das vantagens dos microsserviços em relação às aplicações monolíticas é o tamanho do código-fonte. Como cada microsserviço é projetado para atender a uma funcionalidade específica, seu código tende a ser significativamente menor quando comparado a uma aplicação monolítica, na qual todas as funcionalidades estão acopladas. Embora uma aplicação monolítica possa ser modularizada e conter abstrações utilizando *Domain Driven Design* (DDD), a coesão obtida ainda é consideravelmente inferior à proporcionada pelos microsserviços (NEWMAN, 2015).

Outra vantagem dos microsserviços é a escalabilidade aprimorada. Como cada serviço é independente, é possível escalá-los de forma isolada, alocando recursos apenas para os componentes que realmente necessitam de mais capacidade, em vez de escalar toda a aplicação, como ocorre em arquiteturas monolíticas. Isso permite uma utilização mais eficiente dos recursos, reduzindo custos operacionais e garantindo um melhor desempenho em cenários de alta demanda. Além disso, essa abordagem facilita a adaptação a picos de uso sem comprometer o funcionamento de outros serviços (NEWMAN, 2015).

Outra vantagem significativa dos microsserviços é a maior observabilidade e monitoramento proporcionados por sua arquitetura distribuída. Cada microsserviço pode ser monitorado de forma independente, permitindo a coleta detalhada de métricas, logs e rastreamentos distribuídos. Essa abordagem oferece uma visibilidade granular do funcionamento do sistema, facilitando a identificação de gargalos, falhas e comportamentos anômalos de forma mais precisa e ágil. Ferramentas como *Prometheus*, *Grafana* e *ELK Stack* são amplamente utilizadas para fornecer *dashboards* e alertas em tempo real, contribuindo para uma melhor tomada de decisão e otimização contínua do sistema. (NEWMAN, 2015).

2.2.1.3 Desvantagens

Apesar das diversas vantagens, a adoção de microsserviços também apresenta desafios, sendo a complexidade arquitetural um dos principais. Diferentemente de uma aplicação monolítica, em que todos os componentes estão centralizados, uma arquitetura de microsserviços exige a orquestração e comunicação entre múltiplos serviços distribuídos. Isso demanda a implementação de soluções adicionais, como balanceamento de carga, descoberta de serviços, gerenciamento de falhas e comunicação assíncrona, o que pode aumentar significativamente a curva de aprendizado e os esforços de desenvolvimento. Além disso, a necessidade de coordenar múltiplas instâncias e lidar com problemas como latência e consistência eventual adiciona uma camada extra de complexidade operacional (NEWMAN, 2015).

Outra desvantagem dos microsserviços é a sobrecarga operacional e de infraestrutura. A execução de diversos serviços independentes requer a implementação e manutenção de múltiplas instâncias, pipelines de CI/CD, monitoramento individualizado e armazenamento distribuído, resultando em um consumo maior de recursos computacionais e custos mais elevados em comparação a uma aplicação monolítica. Além disso, a necessidade de gerenciar diversas bases de dados, configurações e *deploys* pode demandar uma

equipe especializada e ferramentas específicas, como *Kubernetes* e *Docker*, que adicionam complexidade à gestão da infraestrutura (NEWMAN, 2015).

2.2.1.4 Dimensionamento

Existem dois tipos de dimensionamento: o vertical e o horizontal. O dimensionamento vertical significa adicionar mais *hardware* às suas instâncias de computação. Alguns exemplos seriam: aumentar a quantidade de memória RAM, aumentar a quantidade de núcleos do processador, ativar um processador mais potente, entre outros (ORACLE, 2023).

Por outro lado, o dimensionamento horizontal consiste em adicionar mais instâncias, ou seja, mais computadores para processarem a carga de trabalho. Se a aplicação está sendo executada atualmente em um único servidor, dimensioná-la horizontalmente implica em distribuí-la entre múltiplos servidores, aumentando assim sua disponibilidade (ORACLE, 2023).

O dimensionamento em uma aplicação monolítica é sempre vertical, sendo feito para a aplicação completa. Não é possível dimensionar um módulo específico, porque tudo faz parte de uma mesma solução, tudo está acoplado. Por isso, mesmo que somente uma parte da aplicação possua a necessidade de um aumento de *hardware*, toda a aplicação irá aumentar (AMAZON, 2023).

Já os microsserviços, graças ao seu desacoplamento, podem ser dimensionados separadamente, ou seja, podem ter seu *hardware* aumentado ou diminuído. Isso se dá porque cada microsserviço é um serviço independente com publicações e funcionalidades independentes (AMAZON, 2023).

Além de dimensionar o microsserviço específico no aumento de *hardware*, nesse caso o dimensionamento vertical, é possível replicá-lo para que várias instâncias, ou seja, para que vários computadores simultâneos possam atender às requisições simultaneamente, nesse caso o dimensionamento horizontal. Então, se antes do dimensionamento tínhamos somente uma instância, após o dimensionamento poderemos ter inúmeras instâncias (AMAZON, 2023).

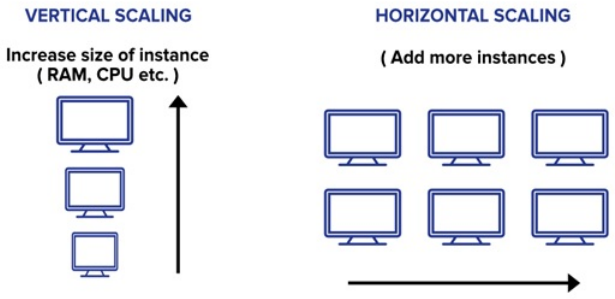

Figura 2.5 Tipos de dimensionamento (LIMA, 2024)

2.2.1.5 Resiliência

Apesar da arquitetura de microsserviços ter algumas vantagens sobre a arquitetura monolítica, existem novos desafios que devem ser enfrentados a partir desse novo paradigma. A resiliência dos microsserviços é um desses desafios (JAMSHIDI, 2018).

Comumente, existem diversas falhas que podem ocasionar a parada completa da aplicação. Elas podem ser causadas por problemas de *hardware*, rede, excesso de carga ou falhas em serviços externos. Se não tratadas, podem ter impacto significativo na disponibilidade geral do sistema, fornecendo uma percepção negativa ao cliente que acessa (JAMSHIDI, 2018).

Existem diversos padrões para diminuir os impactos das falhas. Na aplicação proposta, foi implementado o padrão *retry* que é amplamente aceito no mercado. Esse padrão, como o nome já diz, irá realizar várias retentativas de chamadas antes de retornar um erro para o usuário. Tanto a quantidade de retentativas quanto o tempo entre cada uma são configuráveis (CUSTÓDIO, 2018).

Esse padrão funcionará da seguinte forma: é definida a quantidade de retentativas e o tempo entre elas. Então, a cada falha, será acrescido o tempo definido. A primeira tentativa é realizada pelo fluxo normal. Caso receba um erro, o sistema aguardará o tempo definido e realizará uma nova tentativa. Esse procedimento se repetirá até que a quantidade de tentativas seja ultrapassada ou o retorno seja um sucesso.

2.2.1.6 Integração contínua e implantação contínua

Como os microsserviços são independentes, eles também devem ser implantados de forma independente. Entretanto, como a granularidade pode ser bem alta a depender da arquitetura da aplicação, surge um novo desafio: como gerenciar versões e implantar alterações?

É graças a esse problema que temos a implementação das práticas de desenvolvimento de *software Continuous Integration/Continuous Delivery* (CI/CD), ou em português, integração contínua/entrega contínua.

A integração contínua é o processo de integrar automaticamente as alterações de código de vários desenvolvedores em um projeto centralizado. Seu objetivo principal é detectar problemas ao mesclar um ou mais códigos o mais cedo possível. Por ser um processo automatizado, geralmente envolve a execução automática de uma série de passos, que podem incluir testes, análise de qualidade de código, entre outros. A junção desses passos é chamada de *pipeline* (REDHAT, 2023).

A entrega contínua é o processo de liberar automaticamente as alterações de um código para um ambiente específico. Esse ambiente, na maioria das vezes, é o ambiente de testes. O principal objetivo é ter um sistema que libere as funcionalidades à medida que forem disponibilizadas pelo desenvolvedor, sem a necessidade de um passo manual (REDHAT, 2023).

Existe também a implantação contínua, que é o último passo da entrega contínua. Nesse passo, depois que uma funcionalidade passou por toda *pipeline* de validação, ela está pronta para ser entregue no ambiente real do cliente, ou seja, no ambiente de produção

(REDHAT, 2023). Esse processo é realizado sem a necessidade de interação humana e pode ser acionado por alguns tipos de gatilhos. Por exemplo: quando uma *tag* do repositório for gerada, quando a *branch master* for atualizada, entre outros.

Figura 2.6 CI/CD (REDHAT, 2023)

2.2.2 Arquitetura Monolítica

A arquitetura monolítica representa um padrão onde todos os elementos da aplicação, incluindo a interface do usuário, a lógica de negócios e o acesso a dados, estão unificados em uma única estrutura. Toda a funcionalidade é administrada dentro de um único projeto, utilizando recursos comuns e compartilhando uma única base de código. A interação entre os módulos acontece internamente, sem a necessidade de interfaces externas. Esse modelo arquitetônico é frequentemente empregado em aplicações empresariais convencionais. O desenvolvimento, os testes e a implementação são executados como uma única unidade (HARRIS, s.d.).

No contexto da arquitetura monolítica, o ciclo de vida da aplicação é tratado de forma centralizada, abrangendo todas as fases desde o desenvolvimento até a implantação. A organização do código geralmente segue um modelo em camadas, separando responsabilidades internas de acordo com as funcionalidades do sistema. Alterações no código são aplicadas diretamente no repositório central, garantindo consistência em toda a aplicação. A implantação envolve a distribuição de um único artefato executável. Esse modelo permite que todas as funcionalidades estejam disponíveis em um único ambiente (HARRIS, s.d.).

Uma das vantagens da arquitetura monolítica é a simplicidade de desenvolvimento e manutenção. Devido à centralização de todos os elementos da aplicação em um único código-base, a administração do projeto se torna mais simples, simplificando a compreensão da estrutura pela equipe de desenvolvimento. Ademais, simplifica-se o processo de depuração e testes, já que todas as funcionalidades estão reunidas no mesmo ambiente, facilitando a detecção ágil de falhas e a implementação de correções de maneira unificada (AMAZON, s.d.).

Outra vantagem da arquitetura monolítica é a simplicidade de implementação. Como a aplicação é distribuída como um único objeto, a implementação é mais simples e requer menos ajustes em relação às arquiteturas distribuídas. Isso diminui a exigência de instrumentos especializados para orquestração e simplifica a administração da infraestrutura, tornando o processo de entrega contínua mais rápido e eficaz. Assim, companhias com menos capacidades técnicas conseguem implementar e gerir aplicações monolíticas com

mais facilidade (AMAZON, s.d.).

Uma das desvantagens da arquitetura monolítica é o desafio de escalar. Devido à conexão de todos os elementos da aplicação em uma única estrutura, não é viável escalar funcionalidades específicas de maneira autônoma. Para atender ao crescimento da demanda, é preciso replicar toda a aplicação, o que pode levar ao desperdício de recursos computacionais (AMAZON, s.d.).

Uma outra limitação da arquitetura monolítica é a complexidade na gestão de aplicações de grande escala. Conforme o código se expande, torna-se mais complexo estruturar e entender a estrutura básica, elevando o tempo requerido para a implementação de novas funcionalidades ou resolução de problemas. Alterações em uma parte da aplicação podem impactar outras áreas de forma imprevista, demandando testes detalhados para assegurar que o sistema continue operando adequadamente (AMAZON, s.d.).

2.2.3 Padrão Publish/Subscribe (Pub/Sub)

O padrão *Publish/Subscribe* (Pub/Sub) é um modelo de comunicação assíncrona muito utilizado em sistemas distribuídos para desacoplar os emissores de mensagens (publicadores) dos receptores (assinantes). Nesse padrão, os publicadores enviam mensagens a um intermediário, chamado de *broker* ou barramento de mensagens, não sendo necessário conhecer diretamente os assinantes que irão consumi-las. Os assinantes, por sua vez, registram seu interesse em determinados tipos de mensagens e recebem apenas aquelas que correspondem aos interesses definidos (MICROSOFT, s.d.).

Esse modelo possibilita uma comunicação escalável e eficiente, pois várias instâncias de serviços podem se inscrever para receber mensagens específicas, distribuindo a carga de trabalho de forma dinâmica. O Pub/Sub é amplamente utilizado em arquiteturas de microsserviços, sistemas de *Internet of Things* (IoT) e soluções em nuvem, onde a comunicação em tempo real e a distribuição de eventos são essenciais. (MICROSOFT, s.d.).

A implementação do padrão Pub/Sub pode ser feita por meio de diversas tecnologias, como Apache Kafka, RabbitMQ e Azure Service Bus, que oferecem recursos avançados, como persistência de mensagens, escalabilidade horizontal e garantia de entrega. Esses sistemas permitem a configuração de políticas de retenção e a definição de comportamentos para lidar com falhas, garantindo que as mensagens sejam entregues de maneira confiável, mesmo em cenários de alta demanda ou falhas temporárias na infraestrutura (MICROSOFT, s.d.).

Apesar dos benefícios, o padrão Pub/Sub apresenta desafios, como a necessidade de gerenciamento adequado do consumo de mensagens para evitar sobrecarga nos assinantes e garantir a consistência dos dados. Além disso, a identificação de problemas pode se tornar mais complexa devido à natureza assíncrona da comunicação, exigindo ferramentas de monitoramento e rastreamento para garantir a visibilidade do fluxo de mensagens (MICROSOFT, s.d.).

2.2.4 Padrão Back-end for frontend (BFF)

O padrão *Back-end for frontend* (BFF) é utilizado para customizar a entrega de recursos para cada interface de usuário. Essa customização geralmente é necessária quando clientes diferentes irão utilizar funcionalidades diferentes. Entretanto, ele pode ser usado mesmo em funcionalidades iguais para diminuir o fluxo de dados transitados em aplicações *mobile*, por exemplo. Sendo assim, seguindo esse padrão, uma aplicação *mobile* irá utilizar o seu próprio BFF, uma aplicação web irá utilizar o seu próprio BFF e assim por diante (BUGAN, 2019).

O BFF é a porta de entrada da aplicação *front-end*. É ele que ficará exposto na internet para que os demais recursos da aplicação possam ser utilizados. Essa é uma das suas vantagens: ocultar recursos essenciais da aplicação. Isso diminui consideravelmente o risco de ataques, visto que a API é acessada somente internamente no servidor (BUGAN, 2019).

Outra vantagem desse padrão é a agregação de funções. Se determinada funcionalidade precisa executar inúmeras chamadas em outros serviços, essas chamadas podem ser abstraídas no BFF e se tornarem somente uma chamada na aplicação *front-end*. Isso permite que o código da aplicação *front-end* fique mais simples e mais fácil de ser modificado.

Uma desvantagem do BFF é que muitas vezes é necessário duplicar código. Isso se dá pelo fato de que a API já respondia certas chamadas que agora deverão ser respondidas pelo BFF também (BUGAN, 2019).

Outra desvantagem é justamente o fato do BFF ser a porta de entrada para a API, criando um caminho maior para a informação percorrer. Dependendo do fluxo e do tamanho da aplicação, esse caminho pode ter um impacto negativo na questão de tempo de resposta (BUGAN, 2019).

2.2.5 Application Programming Interface (API)

Uma *Application Programming Interface* (API) fornece uma forma simplificada e padronizada de comunicação entre os diferentes componentes do *software*. Pode fornecer comunicação interna no *software*, entre uma aplicação *back-end* e uma aplicação *front-end*, ou fornecer recursos para acessos de terceiros, indicando a forma de envio e os retornos esperados (LOUZADA, 2024).

Para que a comunicação seja padronizada, as APIs utilizam protocolos. Para a web, o protocolo predominante é o *Hypertext Transfer Protocol* (HTTP), mas existem outros como o *Simple Object Access Protocol* (SOAP). Juntamente com o protocolo HTTP, hoje em dia, é muito utilizado o modelo de arquitetura *Representational State Transfer* (REST) (LOUZADA, 2024).

No modelo REST, temos a figura do servidor, que é o provedor de serviços, onde a aplicação está instalada, e o papel do cliente, que é o solicitante do serviço. Esses dois papéis são responsáveis por toda a comunicação da aplicação. No caso do TicketEase, o servidor é o computador na nuvem que a aplicação está instalada, e o cliente dependerá do dispositivo que está realizando a requisição. Se a requisição estiver sendo realizada por um celular, por exemplo, o cliente será o *mobile*. Se a requisição estiver sendo feita

por um navegador da internet, o cliente será web (LOUZADA, 2024).

O modelo REST possui duas principais vantagens, são elas:

1. É *stateless*, ou seja, o servidor não precisa ficar gravando informações de estado de cada cliente;

2. Interface uniforme, sendo compartilhada tanto pelo cliente quanto pelo servidor, facilitando assim a interoperabilidade (LOUZADA, 2024).

Além disso, o modelo REST trabalha em cima do protocolo HTTP. Isso quer dizer que todas as suas ações são sobre esse protocolo. Essas ações podem ser descritas da seguinte forma:

- *GET* – obter dados do servidor;
- *POST* – enviar ou criar dados no servidor;
- *PUT/PATCH* – atualizar dados no servidor;
- *DELETE* – excluir dados no servidor (LOUZADA, 2024).

2.2.6 Aplicativos Mobile

Os aplicativos móveis, também chamados somente de aplicativos ou "apps", são *softwares* desenvolvidos exclusivamente para dispositivos móveis como *tablets* ou celulares. Existem atualmente *softwares* que também podem ser instalados em *Smart* TVs, mas o foco aqui será os celulares (BOCARD, 2021).

Esse avanço de aplicativos começou em 2007 com o surgimento do iPhone, possuindo inicialmente apenas aplicativos nativos do sistema. Um ano mais tarde, foi lançado o iPhone 3G que trouxe o *App Store* ao sistema operacional da Apple, juntamente com 500 aplicativos que poderiam ser comprados e baixados no celular (BAISI, 2023).

O Android surgiu em 2008 juntamente com os primeiros celulares da Google e a sua loja de aplicativos, o *Android Market*. Posteriormente, em 2012, houve a unificação de vários serviços em um único novo serviço, removendo a loja *Android Market* e substituindo-a pela loja *Play Store*, mais tarde renomeada para *Google Play* (GOMES, 2021).

O Windows Phone foi lançado em 2010 e tinha como missão concorrer com os dois outros já consolidados no mercado. A sua loja de aplicativos, conhecida apenas como *Store*, não tinha grandes quantidades de aplicativos devido à falta de compatibilidade com os outros sistemas operacionais. Esse foi um dos fatores que culminaram na descontinuação desse sistema operacional em 2017 (GOMES, 2021).

Nos dias atuais, os aplicativos estão presentes cada vez mais na vida das pessoas. Seja na forma de aplicativos de envio de mensagens instantâneas como o WhatsApp ou o Telegram, seja nas redes sociais como o Instagram ou o Facebook. Existe também uma nova categoria de aplicativos muito utilizada que está relacionada à saúde e à prática de exercícios físicos. Tais aplicativos fornecem uma integração entre dispositivos *wearables* como os *smartwatches*, óculos inteligentes, entre outros (GOMES, 2021).

2.2.7 Aplicações Web

As aplicações web são *softwares* que são executados em servidores web e acessados através de um navegador da internet. Elas são diferentes de sites, pois fornecem alguma funcionalidade ou serviço específico para resolver determinados tipos de problemas. Alguns exemplos de aplicações web são: e-mails, redes sociais, plataformas de bancos, entre outros (SACRAMENTO, 2022).

A grande vantagem é que a aplicação pode ser acessada independentemente da plataforma e do sistema operacional, não sendo necessário também nenhuma instalação prévia. Basta que o usuário tenha acesso à internet e um navegador compatível. Outra vantagem é a facilidade de atualização da aplicação, não sendo necessário nenhuma ação pelo usuário. Tudo é feito pela própria aplicação e, na maioria das vezes, é totalmente transparente para o usuário final (SACRAMENTO, 2022).

As aplicações web comumente são divididas em duas partes: *front-end* e *back-end*. O *front-end* está relacionado à parte visual da aplicação. É a tela que o usuário acessa, sendo responsável por toda a apresentação, layout e facilidade de uso. Além disso, ele é responsável também pela coleta dos dados inseridos pelo usuário e pela exibição dos resultados. Já o *back-end* é a parte que opera no servidor. Ele é responsável por processar as solicitações enviadas pelo *front-end*, interagir com bancos de dados para armazenar ou obter informações, executar toda a lógica do negócio e enviar o resultado para o *front-end*, que exibirá na tela. É no *back-end* que se encontra toda a regra do negócio, sendo responsável pela aplicação de todas as validações (MACHADO, 2021).

2.3 CLOUD COMPUTING

Nesta seção serão apresentadas as funcionalidades de *cloud computing* utilizadas no projeto. Na primeira seção serão abordadas as funcionalidades da nuvem da Microsoft, o Azure. Em seguida, será abordada a configuração de funcionalidades remotas utilizando o Firebase.

2.3.1 Azure Cloud Computing

O Azure é a plataforma de computação nas nuvens da Microsoft. Ela fornece uma grande variedade de ferramentas que facilitam o desenvolvimento, a implantação e o gerenciamento soluções. Nesta seção, serão citadas as funcionalidades do Azure que foram levantadas tanto para o desenvolvimento quanto para a implantação da aplicação (MICROSOFT, 2024b).

Todos os serviços mencionados são gerenciados pela Microsoft, ou seja, são do tipo *Platform-as-a-Service* (PaaS). Isso significa que a Microsoft fornece toda a infraestrutura necessária para a sua utilização, incluindo manutenção e administração do serviço, como segurança, backup, recuperação e balanceamento de carga (MICROSOFT, 2024d).

O Serviço de Aplicativos Azure é um serviço de hospedagem para aplicativos web que fornece uma gama de recursos como integração com o Github, integração e implantação contínua (CI/CD), dimensionamento automático, autenticação e autorização integradas, entre outros (MICROSOFT, 2023d).

Para armazenamento de arquivos, a Microsoft conta com o serviço Azure *Blob Storage*. É um serviço otimizado para grandes volumes de dados, bem como fornece vários níveis de acesso que podem ser mais baratos ou mais caros conforme o uso. Além disso, suporta a replicação dos arquivos, que é uma forma de protegê-los contra falhas de *hardware*. É um serviço altamente disponível e garante que os arquivos estejam disponíveis sempre que necessário (MICROSOFT, 2023a).

Para o banco de dados, o Azure conta com o CosmosDB. É um banco de dados distribuído globalmente que permite escala elástica, ou seja, pode atender a volumes ilimitados de dados. Possui também baixa latência de leitura e gravação, além de alta disponibilidade (MICROSOFT, 2024a).

Existem processos que precisam ser executados em horários fixos do dia ou com base em algum gatilho. Tais processos podem ser executados usando a computação sem servidor do Azure *Functions*. A grande vantagem é que as funções são disparadas, executadas e depois finalizadas. Sendo assim, a cobrança é realizada somente na execução de fato da função, não onerando o custo geral da aplicação. Além disso, como essa função é executada diretamente no servidor da Azure, ela não consome recursos da aplicação (MICROSOFT, 2023c).

Para armazenamento em cache de dados frequentemente acessados, o Azure conta com o Redis. Esse cache fornece um alto desempenho, baixa latência e alta escalabilidade, podendo escolher entre várias níveis de cache. A grande vantagem do cache do Redis é que quando o tempo de vida do cache é ultrapassado, ele é automaticamente desconsiderado (MICROSOFT, 2023b).

O Azure *Service Bus* fornece uma solução para comunicação descentralizada, amplamente utilizado em comunicação entre microsserviços. Ele funciona através de mensagens e filas, onde um produtor cria uma mensagem em uma fila e um receptor, que está "escutando" aquela fila, recebe e processa. Isso cria um desacoplamento dos sistemas, fazendo com que o sistema que irá processar a mensagem não necessariamente precise conhecer o sistema que está produzindo, basta que a mensagem siga um padrão reconhecido pelos dois lados (MICROSOFT, 2024c).

Figura 2.7 Funcionamento de uma fila (MICROSOFT, 2024c)

2.3.2 Configuração Remota do Firebase

A configuração remota do Firebase permite criar valores padrão em um banco de dados, possibilitando o controle de todas as funcionalidades e da aparência do aplicativo. Esses valores podem ser usados para ativar funcionalidades específicas ou adicionar variáveis, como as cores do layout, que serão aplicadas posteriormente pela aplicação

(GOOGLE, 2023).

Outra vantagem é a possibilidade de personalizar o aplicativo para diferentes tipos de usuários, com base em requisitos específicos. Esses requisitos podem incluir uma versão específica do aplicativo, a localização geográfica do usuário e diversos outros critérios (GOOGLE, 2023).

Além disso, o Firebase facilita a execução de testes A/B, uma estratégia que aprimora o aplicativo por meio de feedback dos usuários, sem impactar sua experiência de uso. O processo consiste em liberar determinadas funcionalidades do aplicativo para um número específico de usuários. Ao interagirem com essas funcionalidades, os usuários podem fornecer respostas positivas ou negativas, oferecendo uma visão geral da aceitação dessa nova funcionalidade. Esses testes não se restringem apenas a funcionalidades específicas, podendo também abranger layouts e customizações gerais do aplicativo (GOOGLE, 2023).

Capítulo 3

Neste capítulo são apresentadas as funcionalidades do software sob a ótica da arquitetura de software, separando-as em contextos que se transformaram em microsserviços.

METODOLOGIA DO SISTEMA

Antes de iniciar o desenvolvimento da aplicação, foram realizadas etapas adicionais de planejamento. Essas etapas incluíram, principalmente, a definição das funcionalidades da aplicação e como elas seriam distribuídas no contexto de utilização de microsserviços. Após uma análise detalhada, foram identificados seis microsserviços. São eles:

1. Chamados;
2. Rotas;
3. Relatórios e configurações;
4. Prestação de contas;
5. Acesso;
6. Integração.

Os microsserviços mencionados são *Application Programming Interface* (API)s com métodos expostos para executar ações em seus respectivos contextos. Além disso, foram implementados alguns padrões como o *Back-end for frontend* (BFF), configurações remotas, entre outros. Tudo isso foi aliado a ferramentas *cloud computing*, no caso o Microsoft Azure.

Nesse capítulo serão minuciosamente descritos todos os componentes da aplicação, bem como o fluxo do chamado com a utilização do TicketEase.

3.1 FLUXO DO CHAMADO

O fluxo de atendimento começa com a abertura de um chamado na plataforma ITSM. Inicialmente, o chamado é atribuído à equipe de suporte N1, responsável pelo atendimento inicial. Caso essa equipe não consiga solucionar o problema, o chamado é reclassificado com base em seu tipo e nível de complexidade. Essa reclassificação não apenas define os SLAs aplicáveis, mas também direciona o chamado para o grupo técnico responsável. Esse grupo técnico é formado por profissionais com habilidades diversas, capacitados para lidar com uma ampla gama de problemas.

Após a classificação, o chamado fica disponível para os técnicos do grupo designado. Nesse momento, o TicketEase facilita o processo de atendimento, exibindo o chamado na lista de todos os técnicos do grupo correspondente. O técnico avaliará se deseja assumir o chamado, considerando fatores pessoais, como horário de trabalho e localização, ou restrições geográficas que definem as áreas de atendimento. Caso o técnico decida atender ao chamado, ele o adiciona à sua fila de atendimento, alterando seu status para “Em Atendimento”.

Se o técnico perceber que adicionou um chamado à sua fila de forma incorreta ou optar por não atendê-lo mais, ele pode devolvê-lo à lista de chamados disponíveis, permitindo que outro técnico assuma o atendimento. Essa ação reverte o status do chamado de “Em Atendimento” para “Pendente”.

Durante o atendimento, se o problema for resolvido, o técnico poderá finalizar o chamado, alterando seu status para “Atendido”. No entanto, caso o problema não seja solucionado ou imprevistos impeçam a conclusão, o técnico tem a opção de devolver o chamado à lista de pendentes ou mantê-lo em sua fila para uma tentativa futura. Se o chamado for devolvido, o status será alterado para “Pendente”; se optar por continuar posteriormente, o status permanecerá como “Em Atendimento”.

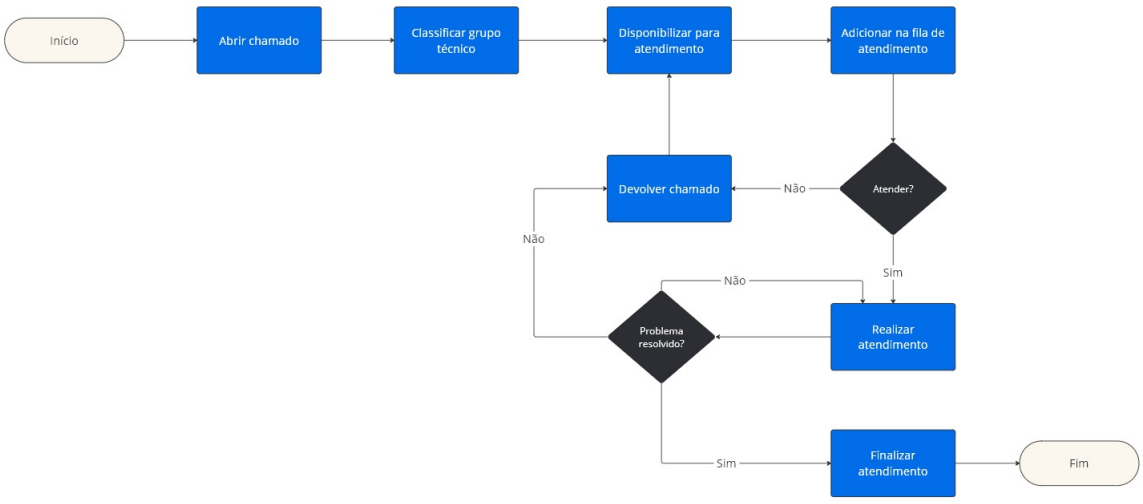

Figura 3.1 Fluxo dos chamados

3.2 ARQUITETURA DA APLICAÇÃO

A aplicação pode ser acessada de duas formas: via navegador de internet ou por meio de smartphone. O acesso via navegador é voltado para gestão, sendo restrito a usuários com permissões específicas de gerenciamento. Já o acesso via smartphone é destinado aos usuários com permissões específicas para atender os chamados, conhecidos como técnicos.

Para realizar o *login*, é necessário primeiramente uma integração direta com a ferramenta *Active Directory* (AD) da empresa. Essa integração permitirá que os usuários utilizem suas contas corporativas para acessar o sistema, através do *Single Sing On* (SSO).

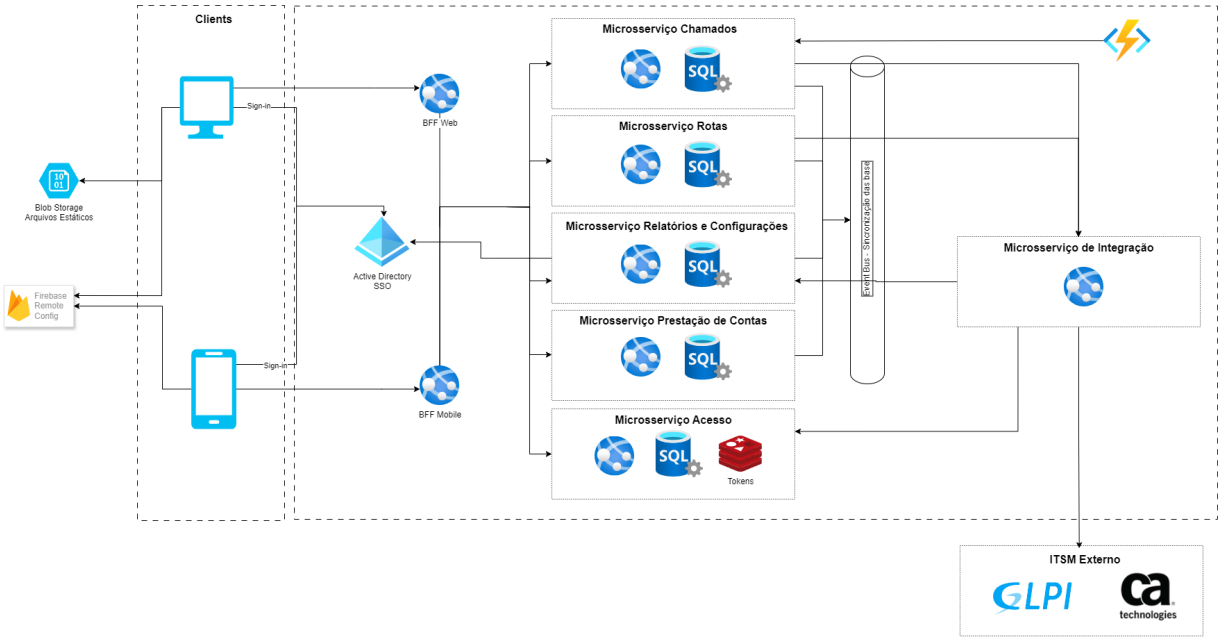

Figura 3.2 Arquitetura TicketEase

Quanto às linguagens de programação, foram escolhidas as seguintes:

- *Back-end*: Microsoft .Net com C#;
- *Front-end*: *framework* React com javascript;
- *Mobile*: *framework* React Native com javascript.

Diversas tecnologias foram utilizadas para apoiar o desenvolvimento e a implantação das aplicações. A nuvem escolhida foi o Microsoft Azure por conter mais facilidade de trabalho com ambientes Microsoft, visto que todo o *back-end* foi desenvolvido utilizando C#. No Azure foram utilizados diversos serviços relacionados a hospedagem, armazenamento de arquivos, banco de dados, funções sem servidor e cache distribuído. Além disso, foi utilizado o serviço de configuração remota do Google para o aplicativo móvel.

Para que a aplicação fique disponível para ser acessada por qualquer pessoa na internet, é necessário hospedá-la em um servidor web. Para isso, foi escolhido o Serviço de Aplicativo Azure.

Para o armazenamento de arquivos, especialmente no microsserviço de prestação de contas, onde o usuário pode enviar documentos para análise, foi mapeado o Azure *Blob Storage*. Sua utilização também foi planejada para a publicação dos ambientes web, nos quais é necessário enviar múltiplos arquivos JavaScript para serem hospedados e posteriormente acessados repetidamente pelos usuários ao utilizarem a aplicação.

O banco de dados escolhido para os microsserviços foi o Azure CosmosDB. Ele foi escolhido por ter rápida resposta nas solicitações, além de escalabilidade automática em caso de picos de requisições. Além disso, por estar na nuvem, fornece uma experiência descentralizada, podendo ser acessado de qualquer lugar do mundo.

Para garantir retentativas em caso de falhas no microsserviço de integração, era necessário que o processamento fosse disparado periodicamente. A solução adotada foi criar funções sem servidor utilizando o Azure *Functions*.

Para acessar os dados no microsserviço de integração, é necessário que o usuário se autentique. Esse processo gera um *token*, que basicamente consiste em uma *string* contendo os dados do usuário para identificá-lo. A autenticação pode consumir recursos desnecessários, pois o *token* possui um tempo de expiração. Para otimizar esse cenário, o *token* será armazenado em um cache de dados. O cache utilizado é o Redis no Azure, que permite definir um tempo de expiração e, por ser armazenado em memória, oferece alto desempenho tanto na leitura quanto na gravação. Após o tempo limite, os dados são excluídos automaticamente.

Com exceção do microsserviço de acessos, que possui utilização direta, toda a comunicação entre os microsserviços é realizada de forma assíncrona. Dessa forma, existe a necessidade de realizá-la sem que um microsserviço dependa exclusivamente do outro. Para isso, foi necessário utilizar o Azure *Service Bus*. Com ele, o microsserviço pode adicionar uma mensagem em uma fila que será processada por outro microsserviço. Um exemplo é o microsserviço de rotas que precisa de alguns dados que são gerados no microsserviço chamados. Sendo assim, quando o microsserviço chamados gera essas informações, ele também cria mensagens na fila com esses dados para que o microsserviço rotas possa recebê-los.

Na publicação de uma aplicação *mobile*, especificamente no caso de aplicações Android, todo o código-fonte é enviado ao Google *Play*. Se houver a necessidade de ocultar temporariamente uma funcionalidade após a publicação na loja, seja devido a um erro ou porque ainda está em desenvolvimento, é necessário efetuar um novo envio do código-fonte, resultando em uma nova publicação. Em tais situações, os usuários serão obrigados a atualizar o aplicativo para acessar a versão corrigida.

Diante desse problema, surgiu a necessidade de utilizar a configuração remota do Firebase. Essa ferramenta permite a criação de valores padrão em um banco de dados, possibilitando o controle de todas as funcionalidades da aplicação. Para cada item do menu, será criada uma opção no banco de dados com o valor “ativo”. Assim, caso seja necessário desativar uma opção no menu, basta alterar seu valor no banco de dados para “inativo”, e a aplicação automaticamente deixará de exibir essa informação.

3.3 MICROSSERVIÇO CHAMADO

O microsserviço Chamado possui todos os métodos que possibilitam ao técnico realizar a interação com as plataformas ITSM. Através dele, o técnico poderá realizar todo o fluxo de atendimento.

Primeiramente, o técnico listará todos os chamados pendentes de atendimento nos grupos aos quais ele pertence. Como cada chamado possui uma categoria, esse grupo é vinculado à categoria e impede que um técnico atenda a um chamado que ele não tem conhecimento técnico para resolvê-lo.

Após listar os chamados, o técnico selecionará aqueles que irá atender e mudará o status para "Em Atendimento". Internamente, o sistema preencherá o campo "Técnico responsável" com o nome do técnico. Os critérios para seleção de atendimentos podem ser estabelecidos previamente, como, por exemplo, SLA maior, distância, tempo de abertura do atendimento, entre outros. Cabe ao técnico selecionar os chamados que melhor atendam a esses critérios.

Após criar sua lista de atendimentos, o técnico poderá sair para realizá-los. Ao final de cada atendimento, o técnico deverá descrever a solução do problema no campo específico e selecionar o botão "Finalizar atendimento". Se necessário, o técnico pode voltar o atendimento para o status "Pendente", permitindo que outro técnico assuma o atendimento.

Esse microsserviço fornece todo o rastreio dos status dos atendimentos, incluindo suas datas, horários e a descrição da solução fornecida pelo técnico. O rastreamento está disponível somente para a área gestora.

3.4 MICROSSERVIÇO ROTAS

Este microsserviço é responsável por gerenciar todas as rotas de atendimentos. Quando o técnico seleciona os atendimentos que deseja atender, uma rota de atendimento é gerada. A rota consiste no trajeto completo que o técnico percorrerá a partir de sua localização atual. O caminho nem sempre será o de menor distância. No entanto, cabe ao técnico escolher, com base no seu conhecimento, o melhor trajeto. Alternativamente, a área gestora pode adicionar métricas para a geração de rotas. Essas métricas podem estar relacionadas a fatores como o trânsito no local, a necessidade de evitar estradas não asfaltadas, pedágios, entre outros.

Para viabilizar essas funcionalidades, a aplicação se integra com o Google *Maps* que fornece não apenas as funcionalidades necessárias, mas também o mapa com locais e rotas marcados. Toda a parametrização das rotas é feita diretamente na configuração do Google *Maps*.

3.5 MICROSSERVIÇO RELATÓRIOS E CONFIGURAÇÕES

A parte gerencial relacionada aos relatórios e configurações do sistema está totalmente separada neste microsserviço. Somente pessoas com perfil de administrador ou gestor terão acesso a essas funcionalidades. O acesso a essas funcionalidades é feito diretamente pela aplicação web. Os técnicos não têm acesso às configurações e relatórios no aplicativo

mobile.

Por ser um microsserviço separado, ele permite a criação de novos relatórios sem impactar o funcionamento do sistema atual. Os relatórios serão gerados inicialmente apenas em formato *Portable Document Format* (PDF), mas existe a possibilidade de adicionar novos formatos no futuro.

3.6 MICROSSERVIÇO PRESTAÇÃO DE CONTAS

Existem alguns custos que são oriundos da atividade do técnico. Os custos mais evidentes são com a manutenção do veículo e os gastos com combustíveis. Além disso, há outros custos que também ocorrem e precisam ser informados. Por exemplo: no período da pandemia, os técnicos precisavam comprar máscaras para atender certos clientes. Essas máscaras são custos relacionados à atividade do técnico.

Para facilitar o controle de reembolsos e evitar o uso de planilhas Excel ou e-mails com anexos de comprovantes, foi desenvolvida a funcionalidade de prestação de contas.

Nesta funcionalidade, o técnico pode adicionar todos os seus gastos. Ele ainda poderá acompanhar todas as despesas que ele enviou, poderá filtrar por datas e poderá também visualizar o retorno da gerência se a solicitação de reembolso foi aprovada ou reprovada. Além disso, em caso de reprovação do reembolso, a gerência irá detalhar em um campo específico o motivo daquela negativa.

Para adicionar um novo reembolso, o técnico deverá preencher os seguintes campos:

- Tipo: abastecimento, insumo ou manutenção;
- Quilometragem do veículo;
- Valor do reembolso;
- Observação (não obrigatória, mas aconselhável);
- Fotos do comprovante fiscal ou da nota fiscal.

3.7 MICROSSERVIÇO ACESSOS

Para acessar as plataformas ITSM externas, como o GLPI e o OTRS, é necessário fornecer credenciais de usuário e senha. Essas credenciais podem ser utilizadas por diversos microsserviços para interações, desde alterações pelo microsserviço Chamados até a obtenção de métricas pelo microsserviço de Relatórios.

Para evitar a necessidade de controlar essas credenciais em todos os microsserviços e centralizar a manutenção em um único ponto, foi criado o microsserviço de Acessos.

Apesar de ser utilizado na aplicação, ele é totalmente transparente. Ao acessar a aplicação, o usuário pode cadastrar suas credenciais, permitindo que o sistema se integre com os clientes que ele atende. Esse cadastro é armazenado diretamente na base de dados do microsserviço.

Para otimizar, quando o técnico interage com o ITSM, a credencial em formato de *token* é armazenada em um banco de dados de cache, no caso o Redis, por um período de 15 minutos. Dessa forma, caso o técnico realize diversas operações que necessitem das informações do ITSM, não será necessário realizar o *login* internamente nesse período. Após os 15 minutos, o *login* é realizado novamente, de forma transparente ao usuário.

3.8 MICROSSERVIÇO INTEGRAÇÃO

A aplicação está habilitada para trabalhar com dois ITSM de mercado: GLPI e OTRS. Embora sejam apenas dois, existem várias versões desses sistemas. Com isso em mente, e considerando a possibilidade de adicionar novos ITSM's no futuro, a integração foi centralizada em um microsserviço.

Além disso, esse microsserviço é responsável por transformar o modelo de dados dos ITSM's em um modelo de dados previamente estabelecido entre os outros microsserviços do TicketEase. Isso porque cada ITSM tem sua própria nomenclatura de campos, uma forma de organizar suas entidades e, sem essa transformação, tudo deveria ser feito exclusivamente no microsserviço Chamado. Além disso, esses campos também poderão mudar de versão para versão, sendo necessária a transformação para um modelo de dados conhecido.

Esse microsserviço atua como uma ponte entre o TicketEase e os ITSM's. Todas as solicitações para obtenção de informações para os ITSM's passam por esse microsserviço. Um exemplo simples seria: um técnico deseja listar todos os chamados com status "Pendente" de um determinado cliente. A solicitação será recebida pelo microsserviço Chamados, que encaminhará para o microsserviço Integração. Este, por sua vez, chamará o microsserviço Acesso para obter as credenciais do usuário e, finalmente, enviará a solicitação para o ITSM externo. Após receber o resultado, o microsserviço Integração converterá os dados para o modelo previamente estabelecido entre os microsserviços e os devolverá ao microsserviço Chamado.

3.9 ITSM EXTERNO

Essa parte do sistema não pertence ao TicketEase. A ferramenta de ITSM é fornecida do cliente, sendo o TicketEase apenas um consumidor. Essa funcionalidade representa o passo final do fluxo do sistema, que começa com a requisição do usuário, passa pelos microsserviços e chega até o ITSM externo. Nesse ponto, as informações são obtidas e, em seguida, retornadas pelo caminho inverso da solicitação.

Caso seja necessário implementar um novo ITSM, não haverá impacto em nenhum outro microsserviço, exceto o microsserviço Integração, onde será adicionada a nova forma de obter os dados.

Capítulo 4

Neste capítulo, será apresentada a prova de conceito realizada para demonstrar a eficácia da solução proposta. Inicialmente, são apresentados dados quantitativos relacionados aos atendimentos de uma empresa do setor de software, utilizada como cenário de teste. Em seguida, será descrito o fluxo de atendimento adotado pela organização. Por fim, serão discutidos os impactos observados com a aplicação do sistema.

PROVA DE CONCEITO

Com o objetivo de demonstrar a aplicabilidade prática e robustez da solução, foi realizada uma prova de conceito utilizando dados representativos de uma empresa do setor de desenvolvimento de *software*, de médio porte, com características compatíveis com o público-alvo do sistema.

O *software* foi testado em um ambiente que simula as condições reais de uso, com volumes de dados e processos similares aos encontrados em contextos corporativos. Essa etapa teve como finalidade validar o desempenho, a escalabilidade e a utilidade do sistema em um cenário próximo ao da sua aplicação final.

4.1 DADOS QUANTITATIVOS

Para o experimento, foram utilizados dados anonimizados da plataforma GLPI do ano de 2023 da empresa alvo. O objetivo principal é validar se o fluxo constante de dados, a alta demanda de interações e a necessidade cada vez maior de um controle centralizado seriam atendidos pelo sistema proposto.

Apesar de a empresa-alvo possuir outras ferramentas de ITSM, o GLPI foi escolhido como foco por conter diversos tipos de atendimentos configurados pela empresa, além de todo um controle gerencial apurado sobre os chamados abertos, sendo controlados os técnicos que realizam os atendimentos, o SLA e a satisfação geral do cliente final.

A sumarização dos dados pode ser observada logo abaixo:

- 19,82 média de chamados abertos por dia;
- 18,77 média de atendimentos por dia;
- 45 clientes cadastrados, entre eles, clientes dos setores público e privado;

- 90 atendentes cadastrados, incluindo tanto os que atuam diretamente no cliente para outros fins e os que realizam atendimentos técnicos.

Durante a execução da prova de conceito, o sistema foi utilizado de forma contínua em um ambiente controlado, que replicava os principais fluxos operacionais da empresa. A simulação contemplou o registro e o acompanhamento de chamados, bem como a análise de métricas relacionadas ao atendimento técnico. Observou-se que o sistema operou de forma estável, mantendo a coerência dos dados e respondendo adequadamente às demandas previstas no contexto real de uso. A compatibilidade com os processos já existentes também contribuiu para a fluidez na integração com as rotinas da organização.

Além disso, os resultados apontaram melhorias na consolidação das informações e no monitoramento dos indicadores de desempenho. A centralização dos dados em uma interface unificada permitiu maior visibilidade sobre os atendimentos realizados, facilitando o acompanhamento dos níveis de serviço e a identificação de pontos críticos nos processos. Essa capacidade de organização e análise reforça o potencial do sistema em apoiar a tomada de decisão e promover ganhos operacionais em empresas com características semelhantes à analisada.

4.2 FLUXO DE ATENDIMENTO

O fluxo do atendimento da empresa se inicia com os chamados do tipo N1 para problemas mais simples. Essa identificação inicial é importante para garantir que um atendimento de caráter não-presencial seja repassado para os níveis de atendimento remoto, reduzindo, assim, o deslocamento da equipe de campo. Com a identificação inicial do atendimento, o chamado é repassado para a equipe de campo, também chamada de equipe externa. A partir de então, o chamado é tratado exclusivamente pela equipe externa.

A categorização correta dos chamados logo na etapa inicial do fluxo é fundamental para otimizar os recursos disponíveis e evitar deslocamentos desnecessários da equipe externa. Para isso, a plataforma GLPI permite o uso de regras automatizadas baseadas em palavras-chave, categorias e localização do cliente, facilitando a triagem dos chamados. Esse mecanismo contribui para a redução do tempo médio de resposta e melhora a alocação das equipes conforme a complexidade e urgência do atendimento.

É importante salientar que nem todos os chamados abertos na plataforma são exclusivamente para atendimentos externos. Existem chamados que podem ser atendidos de forma remota como: configurações de servidores, configurações de cópias de segurança, atualizações de *softwares* em geral. Entretanto, existem outros chamados que devem ser atendidos exclusivamente *in loco* devido à natureza do problema. Alguns exemplos comuns são: equipamento que não está ligando, computador sem acesso à rede, reinstalação do sistema operacional, entre outros.

Após o repasse do chamado para a equipe de campo, o atendimento passa a ser conduzido exclusivamente pelos técnicos externos. Esses profissionais realizam o serviço conforme a demanda e, ao final do processo, registram a finalização do chamado na plataforma. Não há, durante a execução do atendimento, atualizações parciais ou registros

intermediários. Essa prática limita a visibilidade em tempo real do andamento das atividades, mas tem sido suficiente para manter o controle básico dos atendimentos e garantir o encerramento formal das demandas dentro dos prazos esperados.

4.3 IMPACTOS OBSERVADOS

Essa prova de conceito permitiu observar impactos significativos na organização e visualização dos dados de atendimento. A centralização das informações no sistema proposto facilitou a análise dos chamados ao longo do tempo, permitindo identificar padrões de demanda e recorrência de problemas técnicos. Mesmo com a limitação de atualizações apenas no encerramento dos chamados por parte da equipe de campo, foi possível obter uma visão mais estruturada da operação, contribuindo para o planejamento de recursos e a priorização de atendimentos em períodos de maior demanda.

Outro impacto relevante observado foi a possibilidade de geração de rotas mais eficientes para os atendimentos presenciais. Com os dados organizados e centralizados, foi possível planejar deslocamentos com maior eficiência, agrupando atendimentos por proximidade geográfica e horários disponíveis. Essa otimização permitiu reduzir o tempo de deslocamento da equipe de campo. Como consequência, o atendimento tornou-se mais efetivo, com menos ociosidade entre os técnicos externos e melhor aproveitamento da jornada de trabalho.

Também foi observado um ganho no controle gerencial dos atendimentos, mesmo com as limitações operacionais existentes. A consolidação dos dados no sistema viabilizou a construção de indicadores simples, como volume diário de chamados e média de atendimentos por técnico, permitindo que gestores acompanhassem o desempenho da equipe com maior frequência. Embora ainda exista espaço para evolução no detalhamento e na atualização em tempo real dos chamados, os dados extraídos já oferecem suporte para análises estratégicas e identificação de gargalos operacionais.

Além dos avanços já mencionados, observou-se um ganho na padronização dos registros de finalização dos atendimentos. Embora os técnicos de campo realizem as anotações apenas ao término do chamado, a utilização do sistema proposto demonstrou incentivar uma maior consistência nas informações registradas, já que o processo é facilitado pela centralização dos chamados em um único local. Essa prática poderá contribuir para a construção de um histórico técnico mais confiável e útil para futuras intervenções, além de reforçar a cultura de registro dentro da equipe. Com o amadurecimento do uso da plataforma, espera-se que essa padronização possa evoluir e trazer benefícios ainda mais expressivos para a gestão dos atendimentos.

Capítulo 5

Neste capítulo serão apresentados os resultados de todo o desenvolvimento da aplicação. Primeiramente será apresentada a aplicação mobile e, em seguida, a aplicação web. Por fim, será exibido um cronograma de desenvolvimento e serão discutidos os resultados obtidos ao longo do processo.

RESULTADOS E DISCUSSÕES

5.1 APLICAÇÃO MOBILE

Nessa seção, serão apresentadas todas as funcionalidades da aplicação mobile. Ela é acessada exclusivamente por usuários com perfil de atendente técnico, que serão tratados aqui com a nomenclatura de técnicos, e fornece todas as funcionalidades para atendimentos de chamados, geração de rotas, prestação de contas, gerenciamento de acessos e de veículos.

5.1.1 Tela Inicial

Antes de utilizar o sistema, o técnico responsável deverá ter um *login* e senha. Essas informações são obtidas através de uma integração com o *Active Directory* (AD) da empresa responsável pelos atendimentos. A empresa também é responsável pelo cadastro do técnico e por vincular todos os clientes que ele atende. Esse cadastro é realizado por usuários com perfil administrativo, através da aplicação web.

Após entrar no sistema, o técnico será redirecionado para uma tela com várias métricas relacionadas aos seus atendimentos das últimas semanas. O gráfico de barras refere-se a quantidade de atendimentos que o técnico realizou nos últimos 5 dias. Já o gráfico de pizza refere-se aos clientes que o técnico atendeu nesse mesmo período.

Na parte superior, existe o botão para mostrar o menu com todas as funcionalidades do sistema. Esse botão está presente em todas as telas e serve como navegação, permitindo ao técnico entrar no módulo que desejar a partir de qualquer tela.

As notificações, que podem ser acessadas através do botão na parte superior direita, são mensagens que o setor administrativo pode enviar para os técnicos com algumas informações adicionais, como bloqueio de pistas, feriados, entre outros. Essas notificações podem ser gerais, ou seja, para todos os técnicos, ou para um técnico específico.

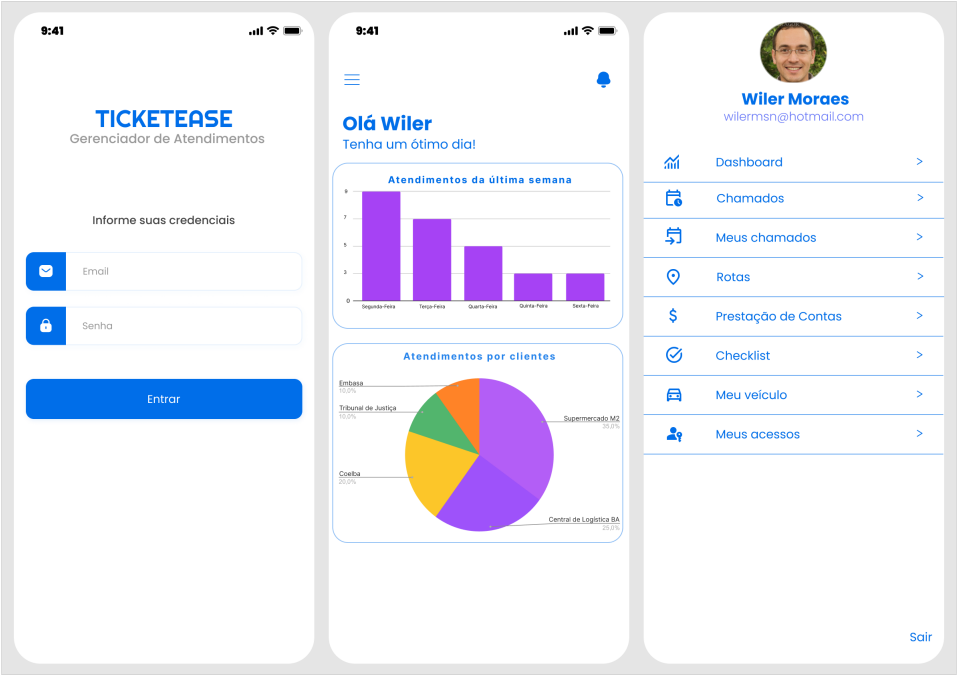

Figura 5.1 Tela inicial do sistema

5.1.2 Chamados

A funcionalidade de chamados é responsável por listar todos os chamados que se adequem aos filtros selecionados. Somente serão listados os chamados com status "Pendente", ou seja, chamados já finalizados ou que já estão no fluxo de atendimento, não serão listados. Os três filtros que podem ser preenchidos são:

- Cliente;
- Grupo Técnico;
- Período da data de abertura.

O cliente é um campo de seleção que terá todos os clientes que o técnico pode atender. O grupo técnico é uma informação do próprio chamado que informa quais tipos de técnicos têm o conhecimento para atender aquela demanda. Cada técnico pode estar vinculado a um ou vários grupos técnicos. O período da data de abertura corresponde a uma data inicial e uma data final em que o chamado foi aberto.

Após aplicar os filtros, o técnico poderá visualizar os chamados sem realizar nenhuma alteração. Caso identifique um chamado que deva atender, poderá adicioná-lo à sua fila de atendimentos clicando no botão "Atender". Essa ação atualizará o status do chamado no ITSM para "Em atendimento", designando o técnico como responsável pelo atendimento. Ao final do processo, será exibida a mensagem "Chamado adicionado à sua fila com sucesso!".

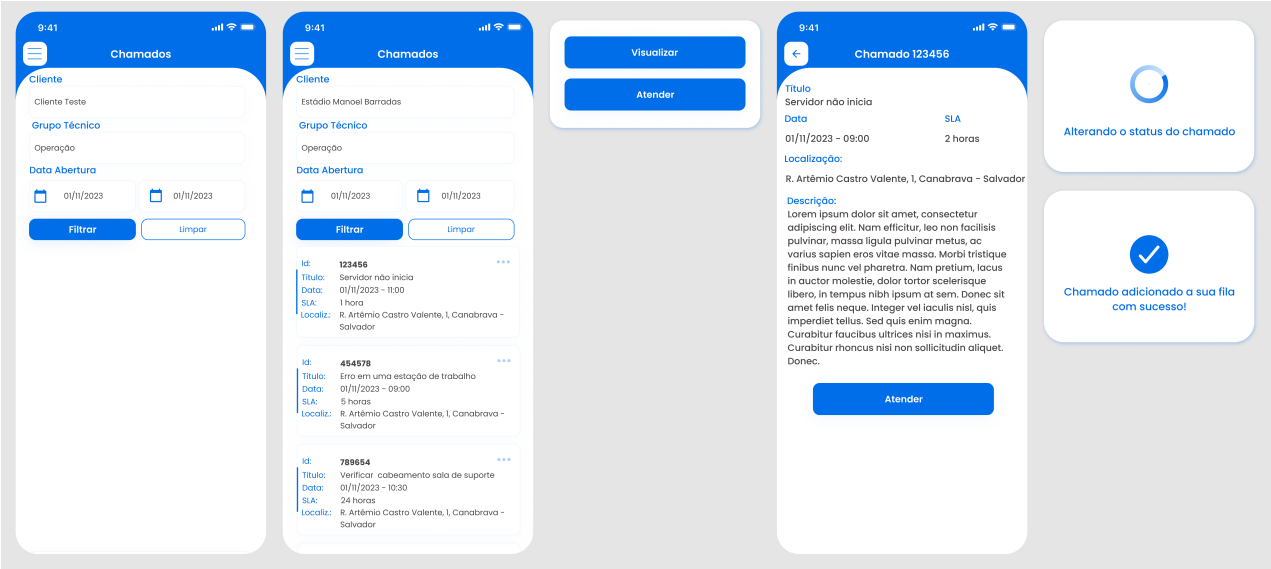

Figura 5.2 Tela de chamados

5.1.3 Meus chamados

Essa funcionalidade é responsável por mostrar todos os chamados que o técnico marcou que atenderá no dia. Além disso, ela fornece o histórico dos atendimentos realizados nos dias anteriores.

Ao abrir a tela, o período de datas selecionado será o dia atual. O técnico poderá alterá-lo conforme necessário e, em seguida, clicar em "Filtrar" para mostrar os chamados do dia selecionado. O botão "Limpar" é responsável por limpar o filtro e os resultados apresentados na parte inferior.

Ao lado de cada chamado, existe um botão que, ao ser clicado, exibirá as opções do chamado. A opção "Visualizar" permite que o técnico visualize todas as informações do chamado. Essa opção está presente em todos os itens da lista, mesmo nos chamados que já foram atendidos. Caso o chamado ainda esteja em aberto, a tela de "visualizar" apresentará um campo de texto para que o técnico informe a solução daquele atendimento. Após isso, basta clicar em "Finalizar atendimento" para que o status do chamado seja alterado para "Finalizado".

A opção "Rota" irá abrir um mapa com a rota até o cliente do chamado em questão. O aplicativo usará como ponto de partida a localização atual do técnico, mostrando o caminho que ele deverá percorrer até chegar ao destino final, no caso, o cliente do atendimento.

A opção "Devolver" possibilita ao técnico voltar o status do chamado para "Pendente" e deixá-lo disponível para que outro técnico possa realizar o atendimento. Ao utilizar essa opção, ficarão registradas nos campos de auditoria do chamado as movimentações realizadas, o atendimento e a devolução do técnico.

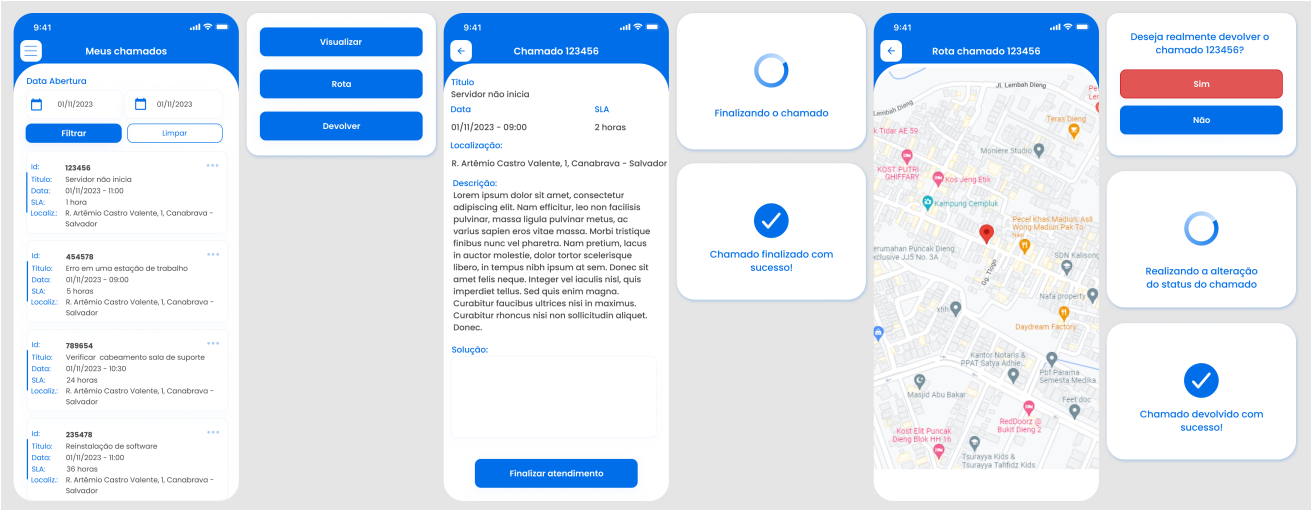

Figura 5.3 Tela Meus Chamados

5.1.4 Rotas

A funcionalidade "Rotas" é responsável por exibir no mapa todos os atendimentos que o técnico deve realizar no dia. Esses atendimentos correspondem aos chamados que o técnico selecionou para atendimento no dia em questão e são visualizados na funcionalidade "Meus Chamados".

As rotas serão exibidas como caminhos, e cada cor representará um cliente específico. Caso algum atendimento seja devolvido pelo técnico, as rotas serão automaticamente reavaliadas, mostrando as rotas atualizadas assim que o técnico acessar a funcionalidade.

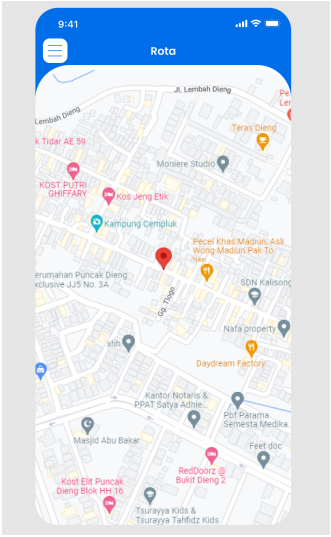

Figura 5.4 Tela das rotas do técnico

5.1.5 Prestação de contas

A funcionalidade de “Prestação de Contas” é responsável por permitir que o técnico receba o reembolso de algum tipo de gasto relacionado ao atendimento. Tais gastos podem ser compra de insumos (máscaras, luvas, entre outros), abastecimentos, estacionamentos, pedágios ou manutenções gerais.

Ao abrir a tela da funcionalidade, o filtro de data é automaticamente aplicado com base na data atual. Caso seja de interesse do técnico, ele poderá mudar essa data para consultar os registros anteriores.

Na parte inferior da tela, o botão “Nova Conta” permite que o técnico adicione uma nova prestação de contas. Ao utilizar essa funcionalidade, o técnico deverá selecionar um tipo de gasto predefinido, informar a quilometragem do veículo, o valor do gasto e anexar fotos do comprovante, como nota fiscal ou cupom fiscal. Além disso, há a opção de adicionar uma observação para justificar o gasto, embora essa observação seja opcional.

Após inserir a nova prestação de contas, ela terá o status “Pendente”. Isso quer dizer que alguém do setor administrativo avaliará aquele gasto. Caso seja aprovado, o status será alterado para “Aprovado”. Caso seja reprovado, o status será alterado para “Reprovado” e uma justificativa será adicionada pelo administrativo. O técnico pode consultar essas informações ao visualizar a prestação de contas.

A edição e a exclusão de registros só poderão ser realizadas em itens que estejam com o status “Pendente”. Registros que já foram avaliados não poderão ser editados ou excluídos, tanto pelo técnico quanto pelo setor administrativo.

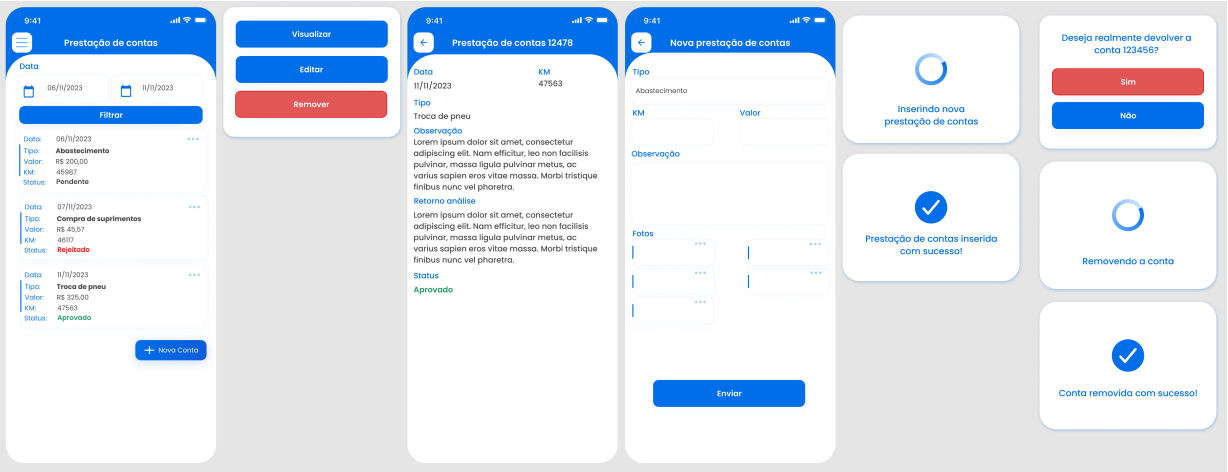

Figura 5.5 Tela de prestação de contas

5.1.6 Checklist do veículo

Para assegurar as condições adequadas dos veículos e evitar autuações de trânsito, foi definida a funcionalidade de “Checklist”. Essa funcionalidade permite ao técnico realizar a verificação de itens previamente estabelecidos pelo setor administrativo, informando se esses itens estão ou não em conformidade.

Além disso, o técnico pode visualizar todos os checklists que realizou durante o uso do sistema, proporcionando um controle detalhado das condições do veículo ao longo do tempo.

Após o registro ser adicionado, ele não pode ser modificado por nenhum usuário, incluindo os administradores, sendo acessível apenas para consulta.

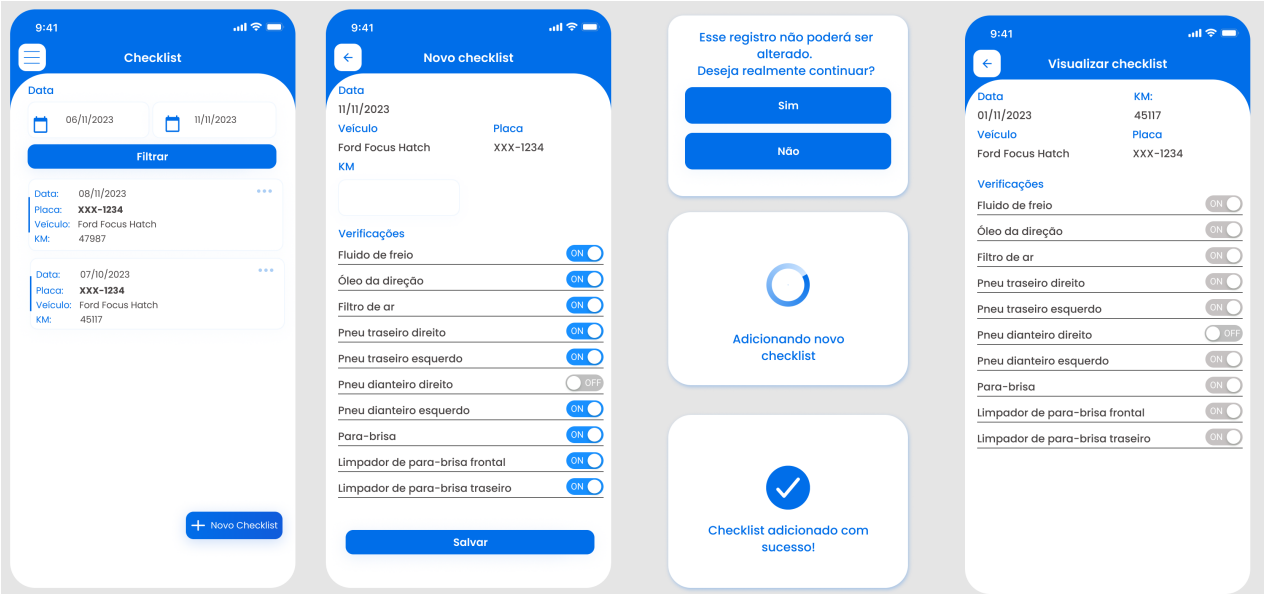

Figura 5.6 Tela de checklist do veículo

5.1.7 Meus veículos

O sistema permite que um técnico utilize diversos veículos para realizar os atendimentos. Para controlar qual veículo está sendo utilizado naquele momento, o sistema conta com a funcionalidade “Meus Veículos”.

É possível cadastrar um novo veículo com a placa, modelo, chassi, ano e outros dados relevantes. Durante a edição do veículo, a única informação que não poderá ser alterada é a placa, pois esse campo é utilizado como chave na aplicação.

Para alterar o veículo em uso, o técnico deve indicá-lo como veículo padrão. Para isso, basta clicar no submenu e selecionar a opção “Tornar padrão”. Após essa ação, o sistema passará a considerar o veículo selecionado em todas telas da aplicação.

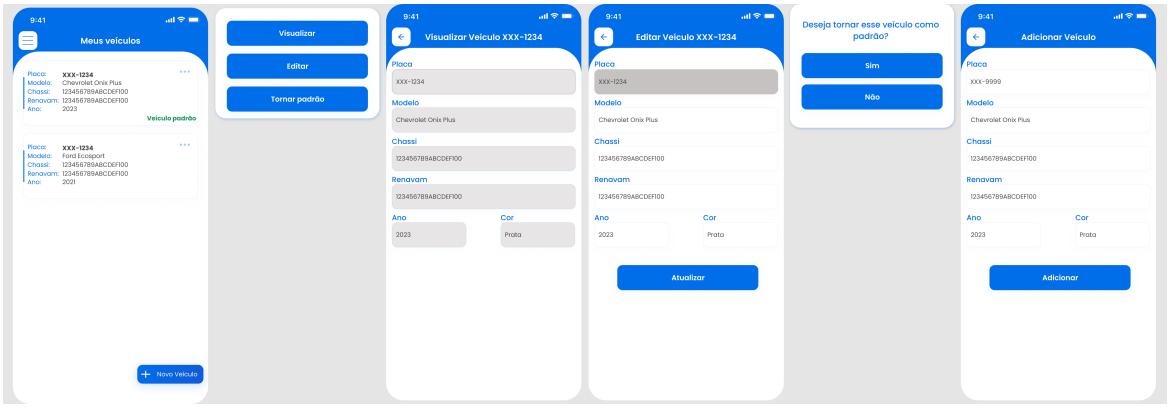

Figura 5.7 Tela Meus Veículos

5.1.8 Meus acessos

Essa funcionalidade gerencia os acessos dos técnicos aos sistemas ITSM. Nela, são exibidos todos os clientes que o técnico está autorizado a atender, junto com a opção de inserir seu próprio login e senha para cada sistema do cliente.

A vinculação entre o cliente e o técnico é realizada pelo setor administrativo na aplicação web.

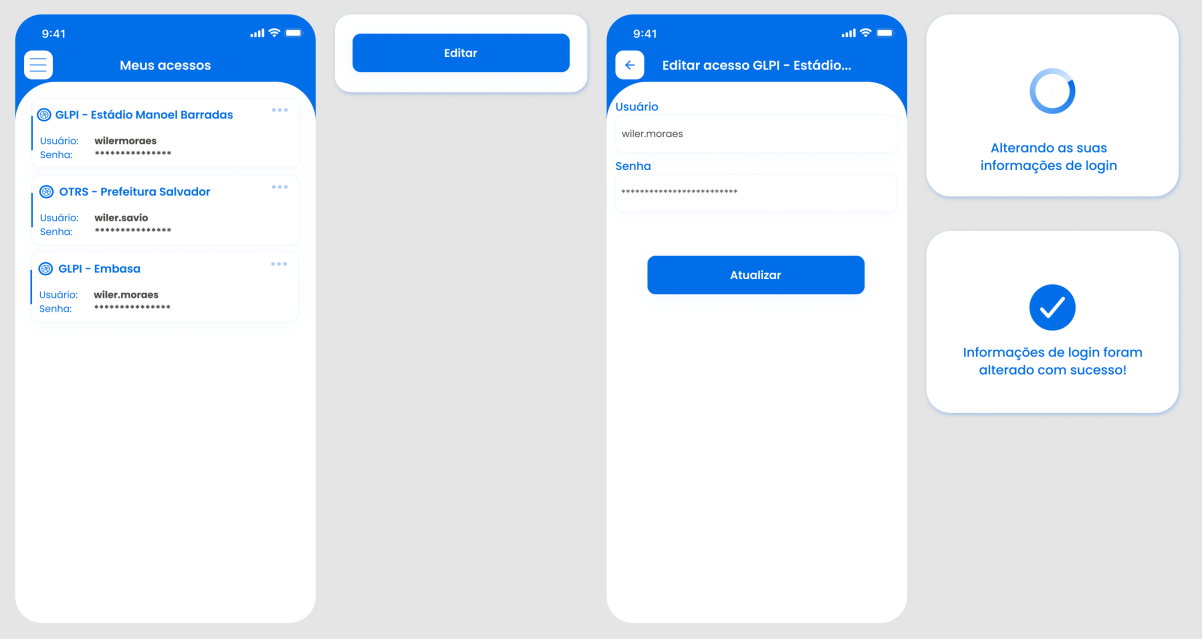

Figura 5.8 Tela Meus Acessos

5.2 APLICAÇÃO WEB

Nessa seção, serão apresentadas todas as funcionalidades da aplicação web. Ela é acessada exclusivamente por perfis administrativos, designados aqui como usuários, e provê todas as funcionalidades de gestão, como acompanhamento de métricas, notificações, cadastros, prestações de contas e relatórios.

5.2.1 Tela Inicial

Após realizar o *login* com permissões administrativas, o usuário será redirecionado para a tela inicial. Essa tela apresentará indicadores dos atendimentos, incluindo o total de atendimentos realizados na última semana, clientes mais atendidos e as regiões mais atendidas.

O menu lateral esquerdo permite navegar por todas as funcionalidades da aplicação web e está disponível em todas as telas.

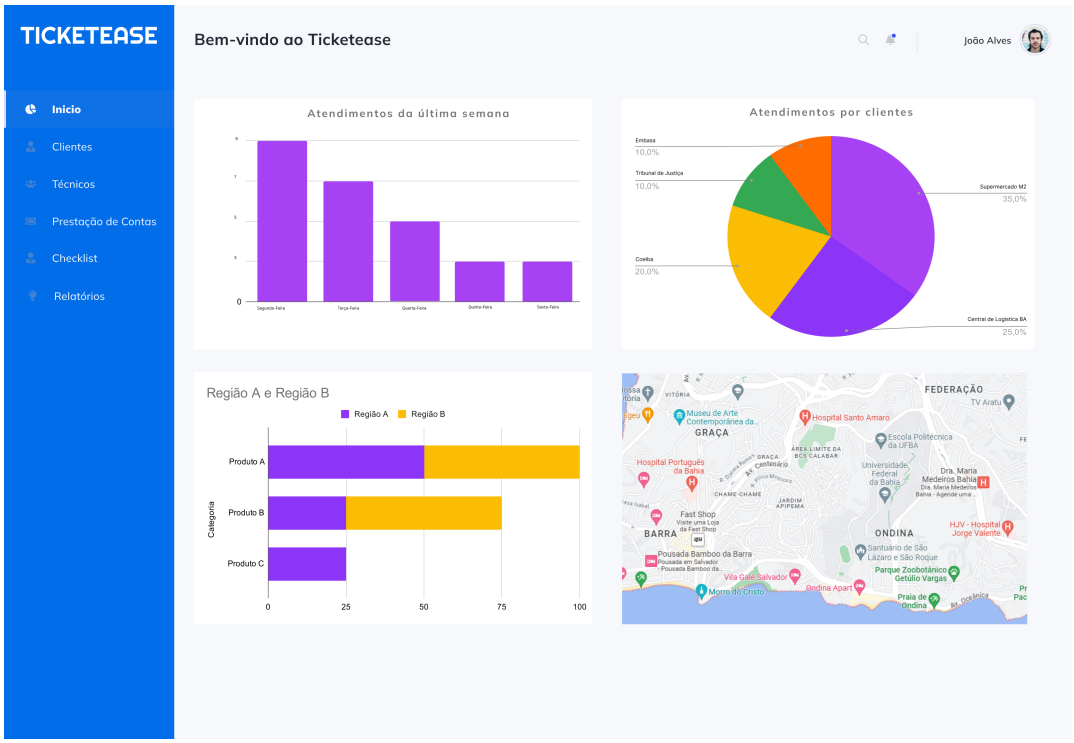

Figura 5.9 Tela Inicial

5.2.2 Notificações

Nesta tela, o usuário poderá visualizar todas as notificações cadastradas, incluindo tanto as notificações gerais, que se aplicam a todos os técnicos, e as notificações específicas, destinadas a um técnico em particular.

Para filtrar as notificações, o usuário deve clicar no botão “Filtrar”, localizado no canto superior direito. Para cadastrar uma nova notificação, basta clicar no botão “Nova Notificação”.

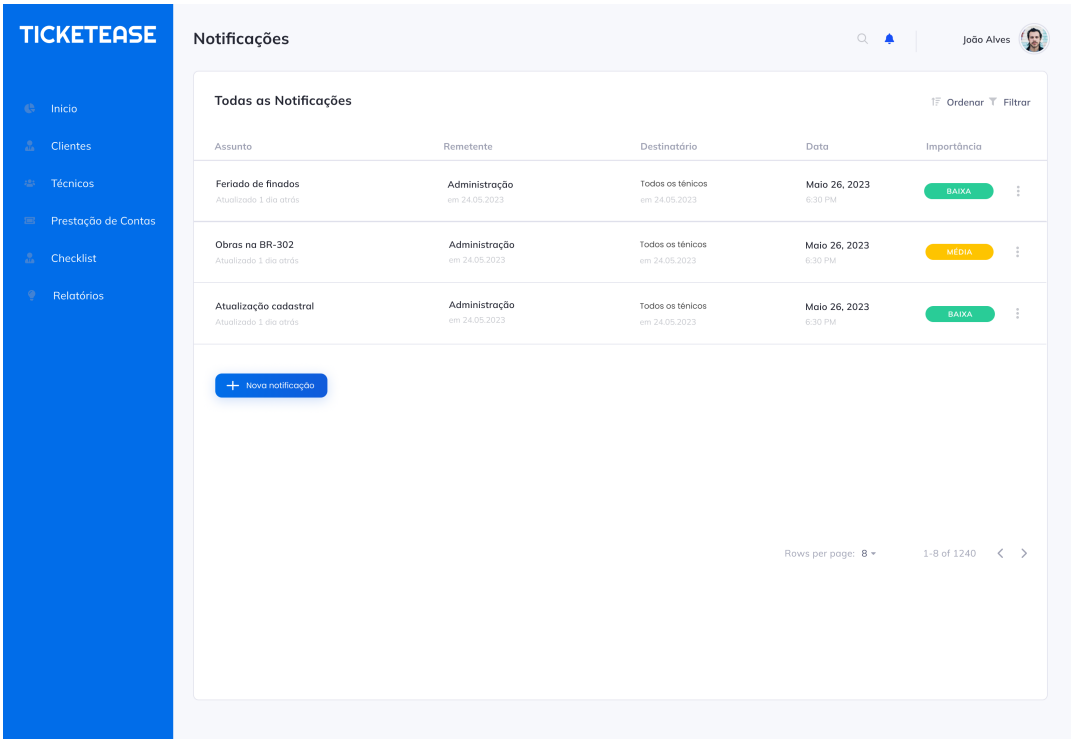

Figura 5.10 Tela de Notificações

5.2.3 Cadastro de Clientes

Na tela de cadastro de clientes, o usuário poderá registrar todos os clientes que são atendidos pelos técnicos. Além dos dados básicos, também devem ser preenchidos os dados relacionados ao ITSM dos clientes como: URL, *token*, tipo e versão. Esses dados são necessários para que o TicketEase consiga se conectar ao ITSM do cliente e realizar as operações.

Para filtrar um cliente, basta clicar no botão “Filtrar” localizado no canto superior direito. Para cadastrar um novo cliente, basta clicar no botão “Novo cliente”. Para visualizar ou editar um cliente cadastrado, basta clicar no ícone localizado à direita do item desejado e selecionar a opção “Visualizar”ou “Editar”.

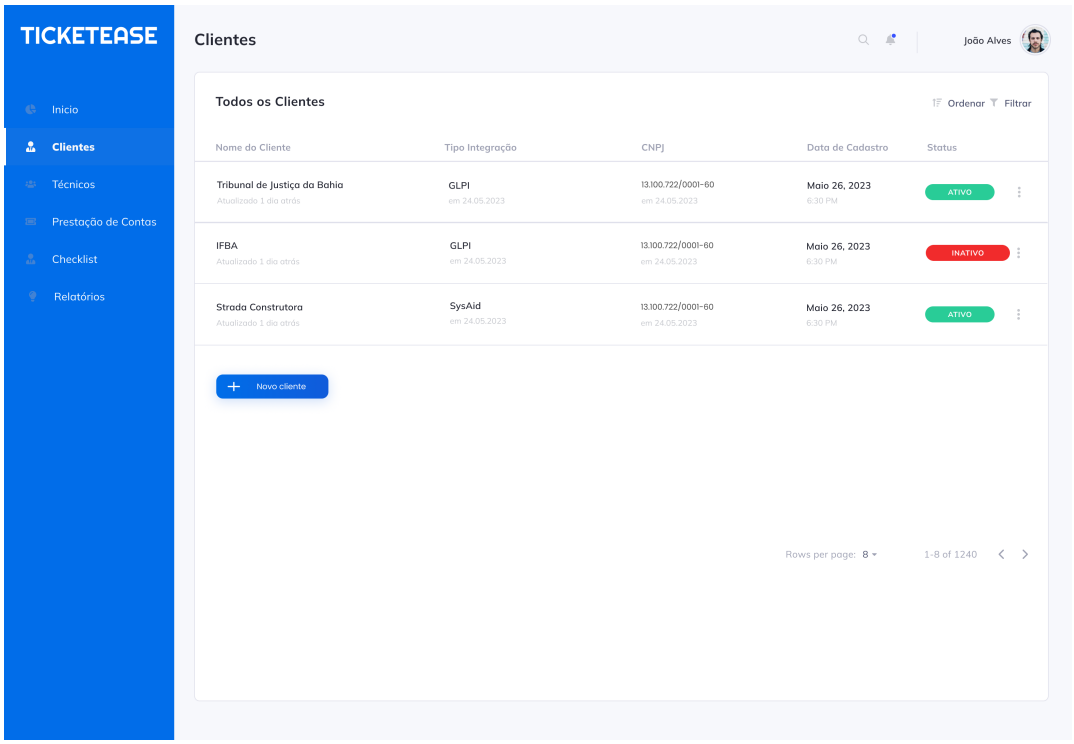

Figura 5.11 Tela de cadastro de clientes

5.2.4 Cadastro de técnicos

Na tela de cadastro de técnicos, o usuário poderá registrar todos os técnicos responsáveis pelos atendimentos. Além de inserir os dados pessoais de cada técnico, será possível vincular os clientes que ele atende ao seu perfil. Assim, o técnico conseguirá visualizar e atender apenas os chamados dos clientes que lhe foram atribuídos.

Para filtrar um técnico, basta clicar no botão “Filtrar” localizado no canto superior direito. Para cadastrar um novo técnico, basta clicar no botão “Novo técnico”. Para visualizar ou editar um técnico cadastrado, basta clicar no ícone localizado à direita do item desejado e selecionar a opção “Visualizar”ou “Editar”.

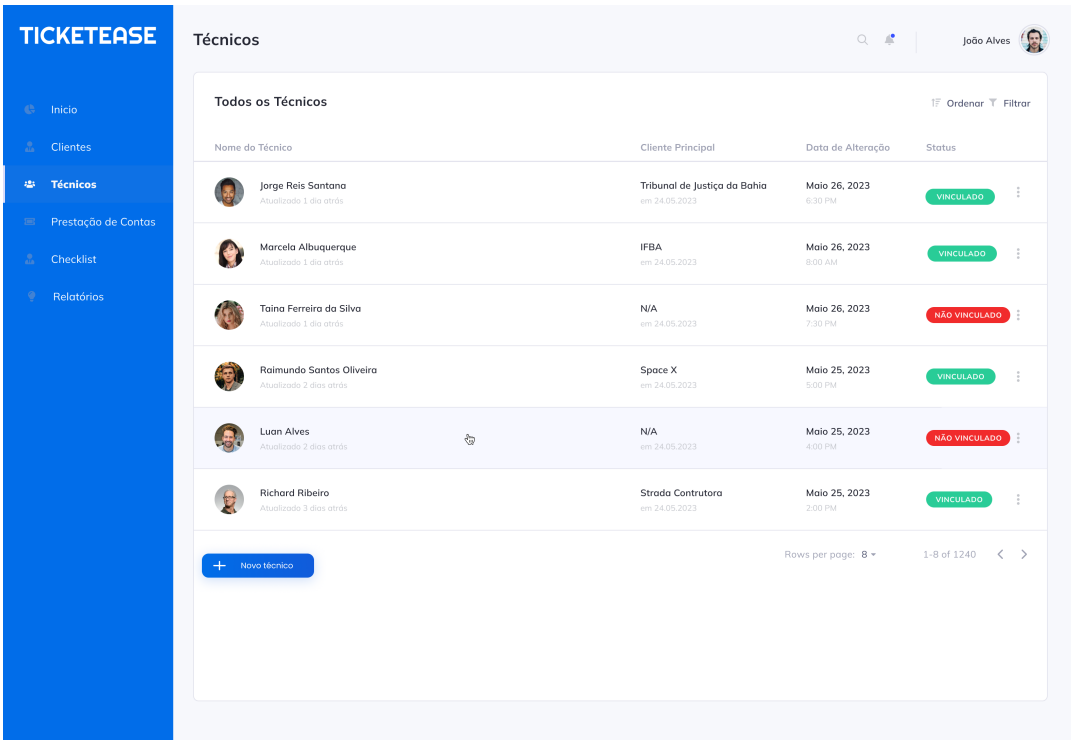

Figura 5.12 Tela de cadastro de técnicos

5.2.5 Prestação de contas

Na tela de prestação de contas, serão listados todos os registros inseridos pelos técnicos. À direita de cada registro, será exibido seu status, que poderá ser pendente, aprovado ou reprovado.

O status pendente informa que o técnico adicionou o registro, mas não obteve nenhum retorno até o momento. O status aprovado informa que a solicitação de reembolso foi aprovada, enquanto o status reprovado indica que a solicitação de reembolso foi reprovada.

Para filtrar um registro, basta clicar no botão “Filtrar” localizado no canto superior direito. Para visualizar, aprovar ou reprovar um registro, basta clicar no ícone localizado à direita do item desejado e selecionar a opção “Visualizar”, “Aprovar”ou “Reprovar”. Ao reprovar, é obrigatório inserir uma observação do motivo da reprovação.

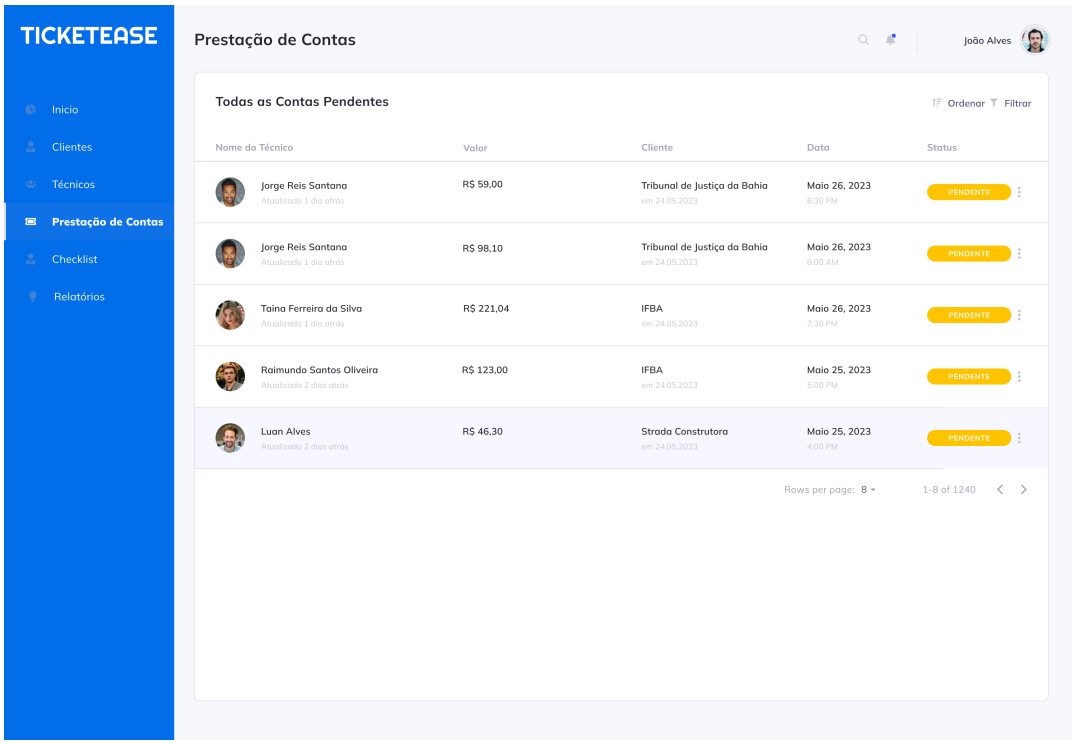

Figura 5.13 Tela de prestação de contas

5.2.6 Checklist

Na tela de checklist, o usuário pode visualizar todos os registros inseridos pelos técnicos ao longo do tempo. Essa interface foi projetada para fornecer uma visão clara e organizada de todas as checagens realizadas, permitindo que os usuários acompanhem as condições dos veículos de forma eficaz. É importante ressaltar que esta tela oferece apenas a opção de visualização, o que significa que não é possível editar ou excluir nenhum registro. Essa restrição foi implementada para garantir a integridade e a precisão das informações registradas, assegurando que todos os dados reflitam fielmente as condições dos veículos no momento das avaliações.

Para filtrar um checklist, basta clicar no botão “Filtrar” localizado no canto superior direito. Para visualizar algum dos registros da tela, basta clicar no ícone localizado à direita e, então, selecionar a opção “Visualizar”.

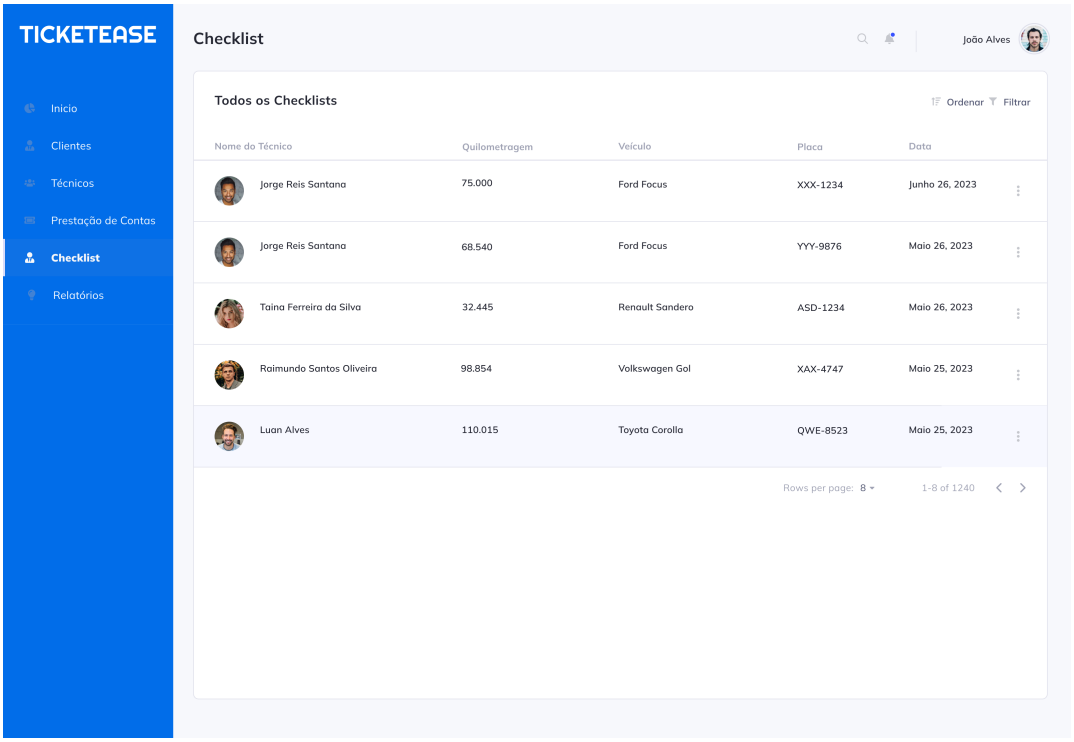

Figura 5.14 Tela de checklist

5.2.7 Relatórios

Por fim, temos a tela de relatórios que atualmente conta com apenas um relatório. Esse relatório exibe o total de cada status dos atendimentos. Além disso, fornece um gráfico com os atendimentos do dia, oferecendo ainda outras métricas como tempo médio de atendimento, porcentagem de resolução, entre outros.

Atualmente, esse relatório pode ser exportado apenas para PDF. Entretanto, com a adição futura de novos relatórios, novos formatos de exportação poderão ser incluídos.

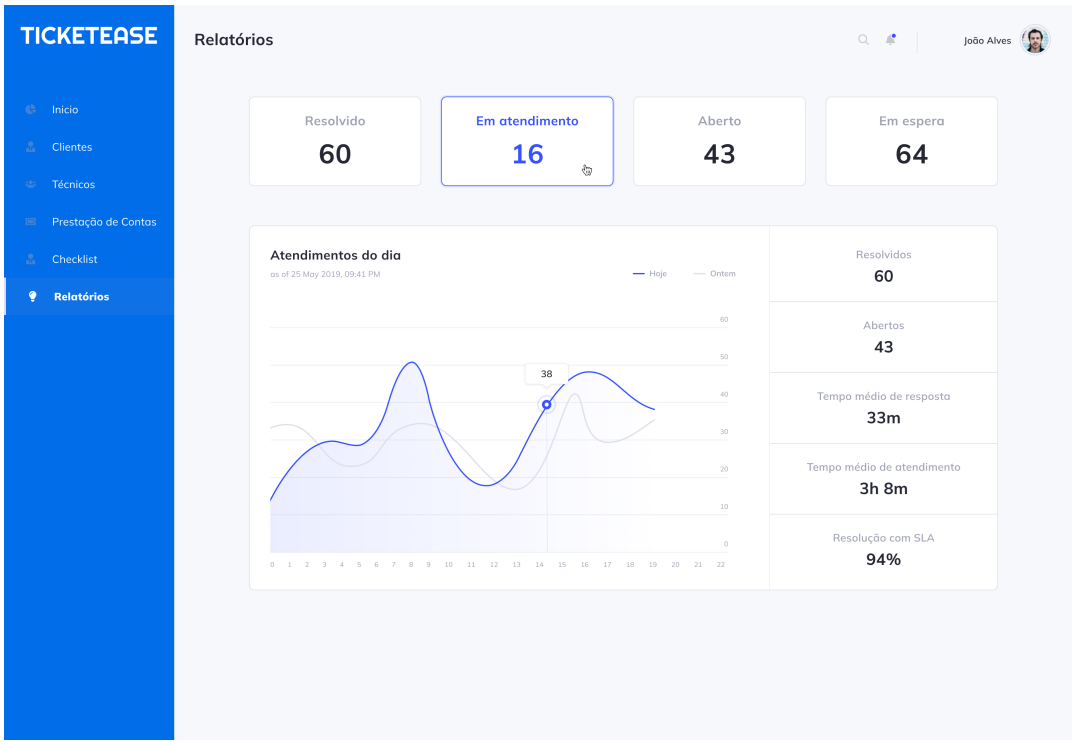

Figura 5.15 Tela dos relatórios

5.3 CRONOGRAMA DE DESENVOLVIMENTO

As atividades deste projeto foram organizadas em cinco *sprints*, cada uma com duração de duas semanas. As tarefas abrangem tanto o desenvolvimento do *back-end* quanto da aplicação *mobile*. Abaixo estão descritas as tarefas planejadas para cada *sprint*:

5.3.1 Sprint 1 – Estrutura Inicial

Nas duas primeiras semanas, as atividades serão direcionadas à construção da estrutura inicial do projeto. As tarefas incluem:

- Criação do repositório da aplicação;
- Configuração da *pipeline* de publicação;
- Desenvolvimento do *boilerplate*, que será a base para todos os microsserviços do *back-end*;
- Configuração inicial do projeto da aplicação *mobile*, incluindo ajustes básicos;
- Criação do projeto inicial do BFF, com as configurações necessárias de acesso e bibliotecas;
- Configuração do ambiente de testes do OTRS, garantindo permissões de acesso à API;
- Configuração do ambiente de testes do GLPI, garantindo permissões de acesso à

API;
- Configuração inicial dos microsserviços, incluindo bancos de dados e permissões.

5.3.2 Sprint 2 – Integração com Plataformas de ITSM

Na terceira e quarta semanas, o foco será nas funcionalidades relacionadas à integração com plataformas de ITSM. As tarefas incluem:

- Desenvolvimento da funcionalidade para listar chamados vinculados ao técnico, utilizando filtros como cliente, grupo técnico e período de abertura;
- Implementação da funcionalidade para visualizar chamados vinculados ao técnico;
- Criação de uma funcionalidade para permitir que o técnico atenda aos chamados.

5.3.3 Sprint 3 – Funcionalidades de "Meus Chamados"

A quinta e a sexta semanas serão dedicadas às funcionalidades associadas aos chamados atribuídos ao técnico. As tarefas são:

- Implementação da funcionalidade para listar os chamados do técnico logado na aplicação;
- Desenvolvimento de uma funcionalidade para finalizar um atendimento, com possibilidade de adicionar observações de finalização;
- Criação de uma funcionalidade para visualizar rotas até os chamados, utilizando integrações externas;
- Desenvolvimento da funcionalidade para devolver chamados, removendo-os da listagem do técnico e disponibilizando-os para outro profissional;
- Criação da funcionalidade de rotas gerais, permitindo a visualização do itinerário completo baseado nos chamados do dia.

5.3.4 Sprint 4 – Prestação de Contas e "Meus Veículos"

Na sétima e oitava semanas, as tarefas serão direcionadas à prestação de contas e às funcionalidades relacionadas aos veículos. As atividades incluem:

- Criação de uma funcionalidade para listar prestações de contas com base em filtros de datas do técnico logado;
- Desenvolvimento da funcionalidade para envio de prestações de contas com imagens;
- Implementação de uma funcionalidade para excluir prestações de contas que ainda não foram analisadas pelo *backoffice*;
- Desenvolvimento da funcionalidade para visualizar prestações de contas;
- Criação da funcionalidade para listar veículos cadastrados do técnico logado;
- Implementação da funcionalidade para visualizar dados de um veículo cadastrado;
- Desenvolvimento da funcionalidade para adicionar novos veículos;
- Criação de uma funcionalidade para editar veículos cadastrados;
- Implementação da funcionalidade para alternar entre veículos, permitindo definir um veículo padrão.

5.3.5 Sprint 5 – Checklist dos Veículos e Configurações

Nas duas últimas semanas, o foco será nas funcionalidades relacionadas ao *checklist* de veículos e às configurações de acesso dos técnicos. As tarefas incluem:

- Desenvolvimento da funcionalidade para visualizar *checklists* baseados em uma data do técnico logado;
- Criação da funcionalidade para adicionar novos *checklists*;
- Implementação da funcionalidade para visualizar *checklists* adicionados anteriormente;
- Desenvolvimento de uma funcionalidade para listar os clientes que o técnico está autorizado a atender;
- Criação de uma funcionalidade para editar dados de acesso de clientes vinculados ao técnico.

5.4 RESULTADOS

Um dos principais resultados alcançados foi o registro oficial do *software*. Esse registro é um passo fundamental, pois valida a conformidade e a integridade da solução, bem como proporciona uma base legal e técnica para futuras atualizações e integrações. Essa conquista não apenas confirma a viabilidade da aplicação, mas também aumenta a confiança das partes interessadas em sua utilização.

Além disso, a conclusão do protótipo é também um resultado extremamente positivo. O protótipo permite uma validação ágil das funcionalidades propostas, além de possibilitar a realização de ajustes antes do desenvolvimento completo. Essa flexibilidade é crucial para evitar que mudanças no escopo do projeto comprometam sua viabilidade.

Outro resultado significativo foi a definição das funcionalidades planejadas, como a visualização dos chamados e a geração automática de rotas para os técnicos. Embora essas funcionalidades ainda não tenham sido implementadas, sua inclusão no escopo do projeto representa um avanço na concepção da aplicação. A visualização dos chamados é essencial para que os técnicos tenham uma compreensão clara das demandas em aberto, enquanto a geração automática de rotas facilitará o deslocamento dos técnicos, garantindo que cheguem de forma eficiente e rápida aos clientes. Essas funcionalidades, uma vez implementadas, prometem melhorar a eficiência operacional e a qualidade do atendimento, um aspecto crucial no contexto de ITSM.

Por fim, a documentação do desenvolvimento e dos processos envolvidos também foi uma conquista importante. A criação de documentação abrangente não só facilita a compreensão da aplicação por novas partes interessadas, mas também serve como um guia valioso para futuras manutenções e atualizações. Essa prática assegura que o conhecimento adquirido durante o desenvolvimento seja preservado e acessível, contribuindo para a sustentabilidade da solução a longo prazo.

Esses resultados, que incluem o registro do *software*, a finalização do protótipo, a definição de funcionalidades essenciais e a documentação do desenvolvimento, refletem um progresso significativo na criação de uma solução eficaz para a integração de chamados de ITSM. Com esses marcos alcançados, a aplicação está bem posicionada para avançar nas

próximas fases de desenvolvimento e implementação, promovendo melhorias importantes nas operações de gerenciamento de serviços de TI da organização.

Capítulo 6

CONCLUSÃO

O presente trabalho abordou temas relevantes utilizados no desenvolvimento de *software* atualmente. Desde a arquitetura da aplicação, utilização de ferramentas de *cloud computing*, padrões e design de código.

Na arquitetura da aplicação pôde-se demonstrar alguns conceitos relacionados a microsserviços, tais como vantagens e desvantagens, dimensionamento, resiliência, integração contínua e implantação contínua. Foi demonstrado também o padrão *Back-end for frontend* (BFF) utilizado juntamente com *Application Programming Interface* (API) para aplicativos *mobile* e web.

Na parte de *cloud computing*, foram demonstradas algumas funcionalidades da nuvem Azure como *Blob Storage* para gravação de arquivos, o banco de dados CosmosDB, a execução de funções com o Azure *Functions*, o armazenamento em cache com o Redis e o enfileiramento de mensagens com o Azure *Service Bus*. Além disso, foi demonstrada também a configuração remota utilizando o Firebase.

Além disso, foram apresentadas as diversas maneiras de estruturar os microsserviços, juntamente com o desenho arquitetônico da aplicação. Cada microsserviço foi minuciosamente detalhado na seção de metodologia do sistema, destacando seu papel e funcionalidade dentro da arquitetura geral. Essa abordagem não apenas ilustra a organização do sistema, mas também evidencia como cada componente contribui para o funcionamento eficaz da aplicação como um todo.

A proposta deste trabalho foi desenvolver uma ferramenta que facilitasse o cotidiano dos técnicos e otimizasse a capacidade de gerenciamento do setor administrativo, por meio de métricas e relatórios. O objetivo era criar uma solução integrada que permitisse aos clientes rastrear suas demandas, proporcionando uma previsão mais precisa para a resolução de problemas. Com essa abordagem, espera-se não apenas melhorar a eficiência operacional, mas também aumentar a satisfação do cliente ao oferecer um acompanhamento transparente e eficaz de suas solicitações.

Como resultado, foi proposto um protótipo que pode atender, mesmo que inicialmente, à demanda de setores que utilizam plataformas de ITSM. Além disso, a ferramenta apresenta um alto potencial de crescimento, uma vez que a adição de novas funcionalidades

não é um processo lento e burocrático. Pelo contrário, ao utilizar microsserviços e o desacoplamento das funcionalidades, a incorporação de novos recursos é realizada de forma descomplicada.

A principal contribuição do TicketEase, além da integração com as plataformas ITSM, é a facilidade de uso que os técnicos experimentarão ao manusear o sistema. Essa facilidade abrange não apenas a centralização de todos os chamados em uma única interface, mas também a implementação de funcionalidades que não estão disponíveis nos ITSM, como a geração de rotas em tempo real, a possibilidade de registrar e gerenciar gastos, e a verificação das condições dos veículos por meio de um checklist. Essas características visam otimizar o trabalho dos técnicos, proporcionando uma experiência mais eficiente e intuitiva.

Evidentemente, além da adição de novas funcionalidades, é fundamental melhorar e expandir alguns recursos existentes para aumentar a usabilidade da aplicação. Entre as melhorias que podem ser consideradas, destacam-se a inclusão de novos relatórios, a criação de integrações com outras plataformas ITSM e a implementação de um sistema de permissões de acesso para usuários na aplicação web. Essas melhorias não apenas aprimorarão a experiência do usuário, mas também permitirão uma gestão mais eficiente e flexível dos processos de atendimento e suporte.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBERTIN, A. L.; ALBERTIN, R. M. d. M. *Benefícios do uso de tecnologia de informação para o desempenho empresarial*. 2008. ⟨https://www.scielo.br/j/rap/a/37B3xwwvmvw5bnVzJRXyXLD/?lang=pt⟩. Acessado em: 28 de Abril de 2024.

AMAZON, I. *Qual é a diferença entre arquitetura monolítica e de microsserviços?* 2023. ⟨https://aws.amazon.com/pt/compare/the-difference-between-monolithic-and-microservices-architecture/⟩. Acessado em: 28 de Abril de 2024.

AMAZON, I. *Principais diferenças: monolítico versus microsserviços*. s.d. ⟨https://aws.amazon.com/pt/compare/the-difference-between-monolithic-and-microservices-architecture/⟩. Acessado em: 20 de Janeiro de 2025.

ATLASSIAN, I. *Jira Service Desk x Jira Service Management*. s.d. ⟨https://www.atlassian.com/br/software/jira/service-management/features/service-desk⟩. Acessado em: 20 de Setembro de 2024.

BAISI, A. *Google Play: como surgiu a loja de apps do Android*. 2023. ⟨https://www.mobiletime.com.br/museu-movel/22/09/2023/google-play-como-surgiu-a-loja-de-apps-do-android/⟩. Acessado em: 22 de Abril de 2024.

BOCARD, T. *O que são aplicativos? Definição da desenvolvedora Usemobile*. 2021. ⟨https://usemobile.com.br/aplicativo-movel/⟩. Acessado em: 22 de Abril de 2024.

BROADCOM, I. *CA Service Management - Guia de Introdução*. 2022. ⟨https://techdocs.broadcom.com/br/pt_br/ca-enterprise-software/business-management/ca-service-management/17-3/getting-started.html⟩. Acessado em: 21 de Abril de 2024.

BUGAN, R. *Arquitetura BFF — Back-end for Front-end*. 2019. ⟨https://medium.com/digitalproductsdev/arquitetura-bff-back-end-for-front-end-13e2cbfbcda2⟩. Acessado em: 01 de Abril de 2024.

CUSTÓDIO, T. *Conhecendo o Retry Pattern*. 2018. ⟨https://imasters.com.br/back-end/conhecendo-o-retry-pattern⟩. Acessado em: 28 de Abril de 2024.

GLPI, I. *GLPI 20 anos*. 2023. ⟨https://glpi-project.org/pt-br/glpi-20-anos/⟩. Acessado em: 01 de Abril de 2024.

GOMES, L. *A história dos apps, uma breve linha do tempo*. 2021. ⟨https://www.showmetech.com.br/a-historia-dos-apps/⟩. Acessado em: 22 de Abril de 2024.

GOOGLE, I. *Configuração remota do Firebase*. 2023. ⟨https://firebase.google.com/docs/remote-config?hl=pt-br⟩. Acessado em: 22 de Abril de 2024.

HARRIS, C. *Microsserviços versus arquitetura monolítica*. s.d. ⟨https://www.atlassian.com/br/microservices/microservices-architecture/microservices-vs-monolith⟩. Acessado em: 20 de Janeiro de 2025.

HORLLE, W. *Jira Service Desk: Entenda os benefícios e praticidades que ele entrega*. 2021. ⟨https://blog.csptecnologia.com/jira-service-desk-o-que-e/⟩. Acessado em: 20 de Setembro de 2024.

INTEROP, I. *O que é o OTRS e como funciona esse sistema de Service Desk*. 2018. ⟨https://www.interop.com.br/o-que-e-o-otrs/#:~:text=Por%20que%20utilizar%20software%20livre,de%20um%20de%20nossos%20especialistas⟩. Acessado em: 20 de Setembro de 2024.

JAMSHIDI, P. e. a. *Microservices: The journey so far and challenges ahead*. [S.l.]: IEEE Software, 2018.

LEANDRO. *Gerenciamento de Serviços com GLPI: Principais funcionalidades e vantagens 2023*. 2023. ⟨https://4infra.com.br/gerenciamento-de-servicos-com-glpi/⟩. Acessado em: 01 de Abril de 2024.

LIMA, A. *Projeto do sistema - escala horizontal e vertical*. 2024. ⟨https://acervolima.com/projeto-do-sistema-escala-horizontal-e-vertical/⟩. Acessado em: 01 de Abril de 2024.

LOUZADA, V. e. a. *API: o que é, para quê serve e qual é a sua importância*. 2024. ⟨https://www.alura.com.br/artigos/api⟩. Acessado em: 28 de Abril de 2024.

MACHADO, A. *Qual a diferença entre front-end e back-end?* 2021. ⟨https://tecnoblog.net/responde/qual-a-diferenca-entre-front-end-e-back-end/⟩. Acessado em: 22 de Abril de 2024.

MICHAELIS. *Dicionário Brasileiro da Língua Portuguesa*. 2024. ⟨https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro⟩. Acessado em: 05 de Maio de 2024.

MICROSOFT, I. *Introdução ao Armazenamento de Blobs do Azure*. 2023. ⟨https://learn.microsoft.com/pt-br/azure/storage/blobs/storage-blobs-introduction⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *O que é o Cache do Azure para Redis?* 2023. ⟨https://learn.microsoft.com/pt-br/azure/azure-cache-for-redis/cache-overview⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *Visão geral do Azure Functions*. 2023. ⟨https://learn.microsoft.com/pt-br/azure/azure-functions/functions-overview⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *Visão geral do Serviço de Aplicativo*. 2023. ⟨https://learn.microsoft.com/pt-br/azure/app-service/overview⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *Banco de dados para Era de IA*. 2024. ⟨https://learn.microsoft.com/pt-br/azure/cosmos-db/introduction⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *O que é Azure?* 2024. ⟨https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-azure⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *O que é o Barramento de Serviço do Azure?* 2024. ⟨https://learn.microsoft.com/pt-br/azure/service-bus-messaging/service-bus-messaging-overview⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *O que é PaaS?* 2024. ⟨https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-paas⟩. Acessado em: 22 de Abril de 2024.

MICROSOFT, I. *Padrão Publisher-Subscriber*. s.d. ⟨https://learn.microsoft.com/pt-br/azure/architecture/patterns/publisher-subscriber⟩. Acessado em: 20 de Janeiro de 2025.

NEWMAN, S. *Building Microservices - Designing fine-grained systems*. [S.l.]: O'Reilly Media, Inc., 2015. ISBN 1491950358.

ORACLE, I. *Configuration Guide for CA Service Desk Ticketing Connector*. 2019. ⟨https://docs.oracle.com/cd/E91266_01/GCCAD/GCCAD.pdf⟩. Acessado em: 22 de Setembro de 2024.

ORACLE, I. *Desing de Escalabilidade*. 2023. ⟨https://docs.oracle.com/pt-br/solutions/oci-best-practices/design-scalability1.html⟩. Acessado em: 01 de Abril de 2024.

OTRS, I. *The history of OTRS*. 2020. ⟨https://corporate.otrs.com/history/⟩. Acessado em: 20 de Setembro de 2024.

PAIXÃO, J. R. d. *O que é SOLID: O guia completo para você entender os 5 princípios da POO*. 2019. ⟨https://medium.com/desenvolvendo-com-paixao/o-que-%C3%A9-solid-o-guia-completo-para-voc%C3%AA-entender-os-5-princ%C3%ADpios-da-poo-2b937b3fc530⟩. Acessado em: 28 de Abril de 2024.

PENSO, T. *Qual a Melhor Opção: Terceirizar a TI ou Investir Em Uma Equipe Interna?* 2017. ⟨https://www.penso.com.br/ti-terceirizada-ou-equipe-interna/⟩. Acessado em: 28 de Abril de 2024.

REDHAT, I. *O que é CI/CD?* 2023. ⟨https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd⟩. Acessado em: 21 de Abril de 2024.

SACRAMENTO, G. *O que são aplicativos? Definição da desenvolvedora Usemobile*. 2022. ⟨https://rockcontent.com/br/talent-blog/aplicacao-web/⟩. Acessado em: 22 de Abril de 2024.

STACK, I. B. *Atlassian: softwares de gerenciamento de projetos*. s.d. ⟨https://www.b2bstack.com.br/empresa/atlassian⟩. Acessado em: 20 de Setembro de 2024.

ZENDESK, I. *ITSM: O que é e as vantagens dele para o TI da sua empresa*. 2023. ⟨https://www.zendesk.com.br/blog/itsm-o-que-e/⟩. Acessado em: 01 de Abril de 2024.