UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

FELIPE KOSLOSKI KOGA

DESENVOLVIMENTO DE UM SISTEMA DE GEOLOCALIZAÇÃO E COMUNICAÇÃO PARA SERVIÇOS DE TRANSPORTES

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2021

FELIPE KOSLOSKI KOGA

DESENVOLVIMENTO DE UM SISTEMA DE GEOLOCALIZAÇÃO E COMUNICAÇÃO PARA SERVIÇOS DE TRANSPORTES

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Diego Roberto Antunes

PONTA GROSSA

2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UM SISTEMA DE GEOLOCALIZAÇÃO E COMUNICAÇÃO PARA SERVIÇOS DE TRANSPORTE

por

FELIPE KOSLOSKI KOGA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 14 de maio de 2021 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Diego Roberto Antunes
Orientador(a)

_____
Prof. MSc. Vinicius Camargo Andrade
Membro titular

_____
Prof. Dr. Richard Duarte Ribeiro
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. André Pinz Borges
Coordenador do curso

RESUMO

KOGA, Felipe Kosloski. **Desenvolvimento de um sistema de geolocalização e comunicação para serviços de transporte.** 2021. 94 p. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2021.

Com o avanço da tecnologia da informação e o aumento considerável dos serviços no setor de transporte, as ferramentas de logística podem fornecer uma grande vantagem competitiva. Dentre essas ferramentas, sistemas de gestão de frota possuem extrema importância para o sucesso de uma empresa, pois podem fornecer benefícios como a maximização da produtividade e a centralização do planejamento estratégico. A gestão de frota, aliada a uma comunicação de qualidade com os funcionários, fornece agilidade e praticidade no gerenciamento e em tomadas de decisões. No entanto, o mercado atual não possui uma ampla variedade de soluções de monitoramento com meios de comunicação utilizando dispositivos móveis. Portanto, o presente trabalho apresenta um sistema que auxilia na gestão de frota, fornecendo recursos para monitoramento em tempo real dos veículos e meios de comunicação entre os funcionários. Para isso, foi desenvolvido uma aplicação *web*, utilizando o *framework* Angular, para a visualização dos veículos da frota em um mapa, e um aplicativo móvel Android, utilizando a linguagem Kotlin, para o envio da geolocalização dos veículos a partir de dispositivos móveis. Além disso, foram desenvolvidos meios de comunicação em tempo real como, *chats* de mensagens assíncronas e *streaming* de áudio, simulando um rádio *Push to talk*. Como resultado, as funcionalidades desenvolvidas no sistema auxiliam na gestão de frota de veículos, fornecendo maior produtividade e centralização estratégica, além da flexibilidade para ser utilizado em diversos contextos, como serviços de agentes de saúde, técnicos em manutenção de Internet e agentes de defesa ambiental.

Palavras-chave: Frota. Tecnologia da Informação. Monitoramento. Comunicação. *Streaming*.

ABSTRACT

KOGA, Felipe Kosloski. **Development of a geolocation and communication system for transport services.** 2021. 94 p. Course Conclusion Work (Technology in Analysis and Development of Systems) – Federal University of Technology – Paraná, 2021.

With the advancement of information technology and the considerable increase in transportation segment, logistics softwares can provide a competitive advantage. Among these services, a fleet management system can be extremely important for the success of a company, as they can provide benefits such as maximizing productivity and centralizing strategic planning. Combined with quality communication with employees, agility and practicality in management, become part of the processes and many decision-making. However, the current market does not have a wide variety of monitoring solutions with means of communication using mobile devices. Therefore, the present work has the objective of developing a system that assists in fleet management, providing resources for real-time monitoring of vehicles and means of communication between employees, coordinators and managers. For this, a web application was developed, using the Angular framework, to view the fleet's vehicles on a map, and an Android mobile application, using the Kotlin language, to send the vehicle's geolocation from mobile devices. In addition, real-time communication means were developed, such as asynchronous message chats and audio streaming, simulating a Push to talk radio. As a result, the features developed assist in fleet management, increasing productivity and strategic centralization, in addition to the flexibility to be used in different contexts, such as health care agents, Internet maintenance technicians and environmental agents.

Keywords: Fleet. Information Technology. Monitoring. Communication. Streaming.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AJAX	*Asynchronous Javascript and XML*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
FAAS	*Function as a Service*
GPS	*Global Positioning System*
HTML	*Hypertext Markup Language*
IDE	*Integrated Development Environment*
PTT	*Push to Talk*
REST	*Representational State Transfer*
SNS	*Simple Notification Service*
SPA	*Single Page Application*
SQL	*Structure Query Language*
S3	*Amazon Simple Storage Service*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO

Ao longo dos anos, os serviços no setor de transporte cresceram consideravelmente (AMARAL, 2015). Para manter esses serviços em alto nível para o cliente, é importante aplicar serviços logísticos, tais como a gestão de frota de veículos (VIVALDI; PIRES; SOUZA, 2012). Essa gestão pode levar a desenvolvimentos adicionais em outros campos, como político, social e econômico (SAGHAEI, 2016). Nesse contexto, os serviços de logística, aplicados de forma eficaz e rápida, são destacados pelas empresas como um elemento que pode fornecer uma vantagem competitiva (LPU, 2013).

Devido a essa importância, é fundamental que as empresas adotem medidas de prevenção de riscos em sua gestão de frota, com o intuito de evitar perdas e danos (ROSA, 2007). Para isso, uma das principais medidas utilizadas é o rastreamento por meio da geolocalização, sendo possível verificar informações sobre a localização do veículo a partir de um aparelho móvel (SEST SENAT, 2018). Além disso, ferramentas que realizam a comunicação entre os condutores e os gestores de frota tornam-se aliadas a essa prevenção (SEST SENAT, 2018), devido à facilidade da troca de informações.

Atualmente, para realizar essa comunicação, dois dos principais modos utilizados são a radiocomunicação (BELIZARIO *et al*., 2003) e o envio de mensagens assíncronas[1] por aplicativos móveis. Na radiocomunicação, os custos de aquisição e manutenção dos equipamentos necessários são elevados, se tornando inviável sua utilização para a maioria das empresas (FLORENCIO, 2019). Porém, utilizando aplicativos móveis, é apenas necessário que o condutor do veículo possua um dispositivo móvel com acesso à Internet.

O mercado atual não possui uma ampla variedade de soluções que unam o monitoramento em tempo real de uma frota, em uma aplicação *web*, com a comunicação assíncrona entre os funcionários. Além disso, como a comunicação instantânea é fundamental no mundo corporativo (FLORENCIO, 2019), mostra-se importante um aplicativo que simule um rádio PTT (*Push to talk*) – pressione para falar.

---

[1] Comunicação que consiste no envio e o recebimento de informações entre um remetente e um destinatário que não ocorrem necessariamente ao mesmo tempo (CACCIAGRANO; CORRADINI, 2001)

Portanto, com o avanço da tecnologia da informação e que, estatisticamente, há mais de um dispositivo móvel por habitante no Brasil (MEIRELLES, 2018), mostra-se relevante o desenvolvimento de um sistema que permita visualizar a localização em tempo real dos veículos em uma aplicação *web*, a partir dos dispositivos móveis dos condutores, e que possa criar canais de comunicação entre os funcionários, por meio de envios de mensagens assíncronas ou mensagens instantâneas em um aplicativo móvel, simulando um rádio PTT.

Além disso, em virtude de sua flexibilidade, o sistema desenvolvido poderá ser aplicado em outros cenários. Atualmente, diversos funcionários prestam serviços fora da empresa, como técnicos em iluminação, agentes de saúde, técnicos em manutenção de Internet e agentes de defesa ambiental. A utilização do sistema proposto, pode auxiliar na logística desses serviços.

Neste contexto, este trabalho propõe o desenvolvimento de um sistema de monitoramento de frota, a partir da geolocalização de dispositivos móveis, e comunicação entre funcionários da frota, por meio de mensagens assíncronas ou simulando um rádio PTT.

Para isso, foi utilizado o modelo de diagrama de casos de uso e esquemas que demonstram o fluxo entre todos os componentes envolvidos no sistema, para uma melhor visão e compreensão geral. Em seguida, foi desenvolvido uma aplicação *web*, utilizando o *framework* Angular, para visualização da frota em um mapa, e um aplicativo móvel Android, utilizando a linguagem Kotlin, para envio da geolocalização do veículo a partir de um dispositivo móvel.

Além disso, determinados serviços são necessários para o funcionamento do sistema, como banco de dados, armazenamento de arquivos e hospedagem dos códigos que contém a lógica de negócio. Para isso, foi utilizado um provedor de serviços em nuvem denominado *Amazon Web Services* (AWS).

Como resultado, as funcionalidades desenvolvidas no sistema auxiliam na gestão de frota de veículos, fornecendo maior produtividade e centralização estratégica. A comunicação em tempo real é realizada de forma rápida e eficaz, permitindo tomadas de decisões rápidas.

1.1 OBJETIVOS

Nesta Seção serão descritos o objetivo geral e os específicos do trabalho.

1.1.1 Objetivo Geral

Desenvolver um sistema de monitoramento de frota e comunicação em tempo real entre os funcionários dos serviços de transporte, utilizando uma aplicação *web* e um aplicativo móvel.

1.1.2 Objetos Específicos

Em conjunto ao objetivo geral, foram definidos os seguintes objetivos específicos:

- Definir a arquitetura do sistema e como cada componente se relaciona, utilizando um provedor de serviços em nuvem;
- Desenvolver uma aplicação *web* que permita visualizar a frota de veículos em um mapa e iniciar um *chat* com os funcionários;
- Desenvolver um aplicativo nativo Android que possibilite o envio da geolocalização do veículo a partir do dispositivo móvel do funcionário e permita iniciar *chats* com outros funcionários;
- Desenvolver uma solução para *streaming*[2] de áudio, similar a um rádio PTT, para a comunicação entre a plataforma *web* e *mobile*.

1.2 JUSTIFICATIVA

O transporte é um campo complexo que envolve vários níveis de decisões, incertezas e consideráveis gastos de capital, sendo esse setor que sustenta a maioria das atividades econômicas sociais (CRAINIC; LAPORTE, 2012). Com isso, para permanecer competitivo, o setor deve contar com soluções de TI para a gestão dos veículos (CRAINIC; LAPORTE, 2012).

[2] Método de transmissão e recebimento de dados contínuos e constantes, através da Internet.

Empresas que utilizam essas soluções podem obter vantagens competitivas, como a otimização das rotas de viagens, maximização da sua produtividade e centralização do planejamento estratégico (AOE, 2010). Porém, as que não possuem esses sistemas, utilizam controles elaborados e executados de forma manual, o que pode causar uma queda em sua produtividade (SOUZA, 2017).

Portanto, soluções que auxiliam no gerenciamento de frota tornam-se aliadas das empresas que necessitam desse controle. Por meio de uma comunicação de qualidade entre os funcionários e monitoramento em tempo real dos veículos, essas soluções fornecem agilidade e praticidade no gerenciamento e em tomadas de decisões.

Na literatura da Ciência da Computação, dois sistemas correlatos foram encontrados. O primeiro, desenvolvido por Heineck (2012), teve o objetivo de desenvolver um sistema de monitoramento em tempo real utilizando dispositivos móveis com o intuito de auxiliar na logística. O segundo, desenvolvido por Meister, Farhat e Trento (2013), consiste em um sistema de monitoramento a partir da localização de um dispositivo portátil, com o objetivo de atuar na área de telemetria de posição.

No mercado, segundo Ferreira (2018), os motoristas da empresa Uber utilizam, em conjunto com o Zello, o Life360 para realizar o monitoramento dos motoristas. Portanto, os motoristas utilizam dois aplicativos: um para comunicação e outro para monitoramento.

Sendo assim, pela falta de documentação e de soluções no mercado, a escolha do tema justifica-se pelo interesse em desenvolver e documentar uma aplicação que aborde essas funcionalidades. Com tais recursos e dados disponíveis em único sistema, será possível auxiliar na gestão de frota de forma prática e efetiva.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho foi dividido da seguinte forma: o Capítulo 2 apresenta o levantamento teórico, com conceitos sobre logística, gestão de frotas e sistemas correlatos no mercado. O Capítulo 3 descreve o desenvolvimento do trabalho, contendo o levantamento de requisitos, arquitetura definida e os resultados obtidos.

Por fim, o Capítulo 4 apresenta as considerações finais do projeto e indica possíveis melhorias e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentada a fundamentação teórica para o desenvolvimento do projeto. A Seção 2.1 discute os serviços de transporte e seus modais. A seção 2.2 aborda a logística, suas definições, conceitos e sua presença na tecnologia da informação e gestão de frota. A Seção 2.3 aborda trabalhos que possuem funcionalidades similares ao projeto desenvolvido. Por fim, a Seção 2.4 apresenta as ferramentas e as tecnologias utilizadas para o desenvolvimento do projeto

2.1 SERVIÇOS DE TRANSPORTE

Segundo Lima Júnior (1995), transportar é uma atividade essencial que torna possível a realização de atividades humanas. O transporte é como um facilitador das interações dos indivíduos nas trocas comerciais de bens e interações humanas, ou seja, é o provedor de movimentação de bens físicos ou pessoas. A necessidade das pessoas de participarem de atividades na sociedade torna necessário os transportes, devido a distribuição espacial e por características temporais (LIMA JÚNIOR, 1995).

No Brasil, os serviços de transporte têm um papel fundamental na economia, devido as empresas de setores distintos atuarem em território nacional e integradas às cadeias de produção e distribuição de bens (CNT, 2017), além de viabilizar a movimentação de insumos para as plantas industriais e de produtos para os consumidores (RODRIGUES, 2014).

2.1.1 Modais de Transporte

O crescimento econômico de uma nação está diretamente relacionado a capacidade e a facilidade de mobilidade, além da acessibilidade da população no deslocamento urbano e o escoamento da produção de mercadorias de forma segura (COLAVITE; KONISHI, 2015).

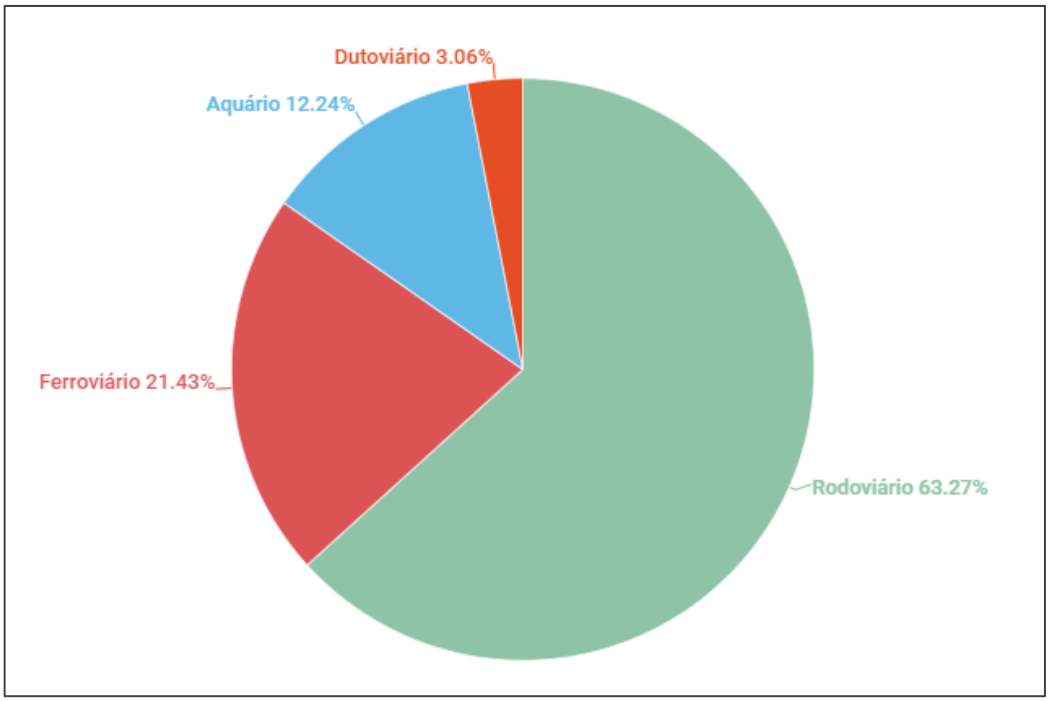
Segundo Rodrigues (2014), os modos utilizados para efetuar um transporte são:

- Rodoviário: transporte de carga em caminhões e carretas pelas rodovias;

- Ferroviário: transporte de carga em vagões fechados pelas ferrovias;
- Hidroviário: transporte de carga em embarcações através de rios, lagos;
- Marítimo: transporte de carga em embarcações através dos mares e oceanos;
- Aquaviário: abrange os modais marítimo e hidroviário;
- Aéreo: transporte de carga em aviões, através do espaço aéreo;
- Dutoviário: transporte de carga através de dutos.

No Brasil, o desequilíbrio do uso dos modais de transporte é prejudicial ao país, devido à dificuldade de escoamento da produção (COLAVITE; KONISHI, 2015). A Figura 1 apresenta a composição da matriz do transporte de carga e passageiros no Brasil.

Figura 1 – Matriz de transporte de cargas do Brasil, em 2016



Fonte: Adaptado de ILOS (2017)

2.2 LOGÍSTICA

A palavra “logística” é derivada da palavra francesa *"Loger*”, um conceito militar que está relacionado ao movimento e suprimento de exércitos (LPU, 2013). A logística também possui origem grega e significa contabilidade e organização (SILVA,

2019). Essa expressão não é uma ideia recente, pois desde a construção das pirâmides, o fluxo efetivo de materiais e informações para atender às exigências dos clientes tiveram pequenas mudanças (CHRISTOFER, 2011).

Atualmente, logística é o processo de gerenciar estrategicamente o movimento, a aquisição e o armazenamento de materiais, peças, estoques e fluxos de informações (CHRISTOFER, 2011), para satisfazer os clientes e aumentar a competitividade dos negócios (TSENG; YUE; TAYLER, 2005). É o *design* e gerenciamento de um sistema para controlar o fluxo de material de toda a corporação, sendo um dos segmentos mais importantes do fenômeno do marketing nos negócios (LPU, 2013).

A logística é uma orientação e estrutura de planejamento que busca criar um plano único para o fluxo de produtos e informações por meio de um negócio (CHRISTOFER, 2011). Esse processo abrange atividades como serviços de transporte, troca de informações, gerenciamento de estoque, armazenamento e manuseio de carga (KONDRATJEV, 2015), fornecendo uma potencial vantagem competitiva para muitas empresas (BITRE, 2001).

O desenvolvimento da tecnologia permitiu a remoção dos limites no comércio mundial e o aumento da concorrência trouxe a necessidade de entrega do produto ao cliente na hora certa, ao melhor preço e a custo mínimo (LPU, 2013). A logística e o processo de distribuição são tão importantes quanto a produção do produto, sendo isso notado pelas empresas como o elemento competitivo mais eficaz nas condições do aumento da concorrência (LPU, 2013).

No século XXI, as empresas destacam à realização rápida e eficaz dos serviços da logística, podendo superar o próprio produto na competição entre as empresas (LPU, 2013). Para isso, é necessário um planejamento e gerenciamento logístico adequado (MASO, 2018).

2.2.1 Gerenciamento Logístico

No século XXI, as mudanças estratégicas impulsionam o gerenciamento logístico nas empresas (LPU, 2013), devido a busca de maneiras eficientes de atender as atuais exigências do mercado, como atendimento rápido, qualidade e cumprimento

de prazos (MASO, 2018). Com um gerenciamento logístico de qualidade, as empresas conquistam rapidez, flexibilidade e eficiência em suas funções logísticas (LPU, 2013).

A missão do gerenciamento logístico é coordenar e planejar as atividades necessárias para atingir os níveis desejados de serviços e qualidade, almejando o menor custo possível (CHRISTOFER, 2011). É o meio pelo qual as necessidades dos clientes são satisfeitas por meio da coordenação dos fluxos de materiais e informações da empresa e fornecedores (CHRISTOFER, 2011).

O gerenciamento logístico é uma função de integração, que coordena e otimiza todas as atividades da logística, além de integrar as atividades da logística à outras funções, tais como o marketing e a tecnologia da informação (CSMP, 2020). As atividades do gerenciamento de logística incluem o gerenciamento de transportes de entrada e saída, gerenciamento de frota, armazenamento, manuseio de materiais, atendimento de pedidos, entre outros (CSMP, 2020).

2.2.2 Tecnologia da Informação

A Tecnologia da Informação (TI) mudou a maneira como as empresas operam e obtém sucesso na economia (TURBAN, 2004). O uso da TI nas empresas começou nos anos 1960, com aplicativos de processamento de dados (KALKAN, 2018). Atualmente, novas abordagens comerciais e operacionais começam a mudar com a integração de tecnologias, como troca eletrônica de dados, Internet[3], *Intranet*[4] e *Extranet*[5], facilitando a comunicação entre os usuários (KALKAN, 2018).

O uso de TI é o fator mais importante para as empresas manterem sua vantagem competitiva (AZEVEDO; FERREIRA; LEITÃO, 2007). Ferramentas de TI facilitam os processos de obtenção de informações (SZYMONIK, 2012), sendo amplamente disponíveis pela Internet, reduzindo a dificuldade de compra e distribuição (PORTER, 2001). Além disso, com a abertura da Internet, combinada com avanços na arquitetura de software e ferramentas de desenvolvimento, tornou-se mais

[3] Rede global de computadores que se comunicam utilizando um protocolo comum.
[4] Rede privada que atende às necessidades de uma empresa (SIEGEL, J, H; HARTMAN, S, W; QURESHI, A, 1998).
[5] Rede similar a *Intranet*, porém é possível ser acessado pela Internet, permitindo conexão entre várias empresas (SIEGEL, J, H; HARTMAN, S, W; QURESHI, A, 1998).

fácil para as empresas realizarem a projeção e implementação de aplicações próprias (PORTER, 2001).

Nesse sentido, as empresas devem desenvolver metodologias com o intuito de adotar novas tecnologias no campo da logística para se tornar ainda mais competitivas (AZEVEDO; FERREIRA; LEITÃO, 2007). Um número crescente de empresas está substituindo modelos de gerenciamento tradicionais por sistemas integrados que ajudam a reduzir custos e aumentar a velocidade e fluidez dos fluxos físicos e de informações (AZEVEDO; FERREIRA; LEITÃO, 2007; LPU, 2013). Neste contexto, o sucesso dos negócios de logística depende de sua capacidade de criar soluções que proporcionem satisfação ao cliente, estruturando a tecnologia que pode contribuir para o desempenho dos negócios (KALKAN, 2018).

2.2.3 Transporte

Transporte é a área operacional da logística que move e posiciona geograficamente o estoque (LPU, 2013). A logística é responsável por administrar todas as operações envolvidas nessa área operacional (SILVA, 2019). Devido à sua importância fundamental e custos visíveis, o transporte tradicionalmente recebe considerável atenção gerencial (LPU, 2013).

O transporte ocupa um terço da quantidade nos custos de logística e sistemas de transporte influenciam o desempenho do sistema de logística, sendo um elemento chave na cadeia logística (TSENG; YUE; TAYLER, 2005). O transporte é necessário em todos os procedimentos de produção, desde a fabricação até a entrega aos consumidores finais e devoluções (TSENG; YUE; TAYLER, 2005).

No sistema logístico, três fatores são essenciais para o desempenho do transporte (LPU, 2013):

- Custo: é o pagamento pela remessa entre duas localizações geográficas e as despesas relacionadas à manutenção do estoque em trânsito. Os sistemas logísticos devem utilizar o transporte que minimize o custo total do sistema, ou seja, o método de transporte mais barato pode não resultar no menor custo total;
- Velocidade: é o tempo necessário para concluir um movimento específico;

- Consistência: é o tempo necessário para executar um movimento específico em várias remessas, refletindo na confiabilidade do transporte.

Os requisitos de transporte podem ser satisfeitos de três maneiras básicas (LPU, 2013):

- Uma frota privada de equipamentos pode ser operada;
- Contratos podem ser firmados com especialistas em transportes dedicados;
- Uma empresa pode contratar os serviços de uma ampla variedade de transportadoras que fornecem diferentes serviços de transporte, conforme necessário, por remessa.

O setor de transporte requer mais investimentos devido ao fato de trazer lucratividade e resultados imediatos à empresa (SILVA, 2019). Com isso, a gestão de transportes é uma área de atuação de grande complexidade, devido a vários fatores, tais como, o controle de tempo entrega, e segurança, treinamento de motoristas, controle de consumo de combustível e sistemas de rotas (SILVA, 2019).

2.2.4 Comunicação

Atualmente, um dos principais meios utilizados para a comunicação em serviços de transporte é via rádio, no qual os motoristas se comunicam com uma central de monitoramento (BELIZARIO *et al*., 2003) ou trocam informações entre si. Grandes empresas que mantêm frotas de veículos equipados com rádios ou telefones, podem estabelecer um contato rápido entre os veículos ou entre um veículo e uma central de atendimento (STOSCHEK, 1978).

Contudo, o custo para a aquisição e manutenção dos equipamentos necessários para utilização da radiocomunicação é elevado (FLORENCIO, 2019). Somado a isso, frequentemente existem parcelas da frota de veículos que não estão em uso, causando inatividade para esse dispendioso equipamento nos veículos que não são utilizados suficientemente (STOSCHEK, 1978).

Com o lançamento de aplicativos que realizam a comunicação assíncrona, ou seja, comunicação que consiste no envio e o recebimento de informações entre um remetente e um destinatário que não ocorrem necessariamente ao mesmo tempo

(CACCIAGRANO; CORRADINI, 2001), a troca de informações entre os indivíduos se tornou mais simples.

A comunicação assíncrona não requer que os usuários estabeleçam uma sessão de comunicação e permaneçam sincronizados durante toda a troca de informações (SHORE, 2016). Devido a isso, surge uma nova alternativa para comunicação em serviços de transporte.

2.2.5 Gestão de Frota

Na logística, o gerenciamento de frota é fundamental para manter um serviço de alto nível (VIVALDI; PIRES; SOUZA, 2012). Para isso, com o avanço da tecnologia, a utilização de sistemas de TI gera novas possibilidades para a melhoria dos serviços de logística (VIVALDI; PIRES; SOUZA, 2012).

Esses sistemas identificam a localização e a direção do movimento de cada veículo em tempo real e relatam automaticamente essas informações diretamente ao gerente de frota (COFFE *et al*, 2003). Esses veículos podem ser carros, vans, caminhões, aeronaves, navios e vagões (SAGHAEI, 2016).

Para que o relatório dos veículos seja efetivo e seguro, cada veículo da frota possui um intervalo de tempo designado para transmitir suas informações por meio de uma rede de comunicações sem interferir nas transmissões de outros veículos (COFFE *et al*., 2003).

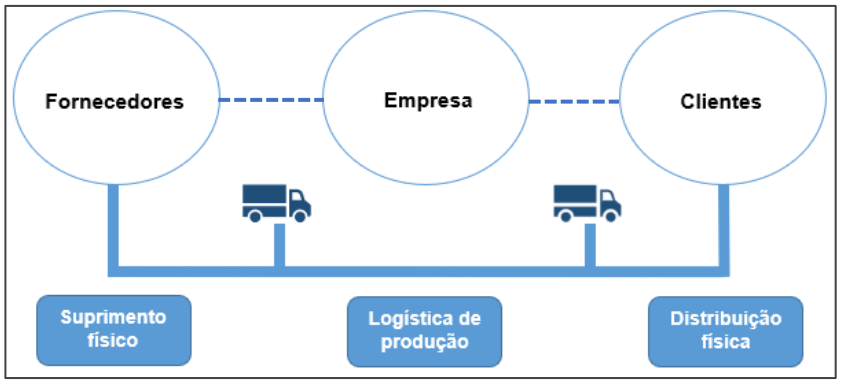
Devido à importância dos serviços de transporte, a gestão de frota é de extrema importância e pode proporcionar desenvolvimentos adicionais em outros campos, como político, social e econômico (SAGHAEI, 2016). As vantagens desse gerenciamento são (SAGHAEI, 2016):

- Gerenciamento eficiente e exato da frota de veículos, e o aumento da capacidade de supervisão;
- Promoção da eficiência do sistema e redução considerável dos custos de controle e monitoramento em comparação com os sistemas tradicionais de supervisão baseados em operadores;
- Recebimento de informações exatas de desempenho e operação dos veículos;

- Aumento de eficiência do sistema de gerenciamento de frota;
- Redução considerável de violações de direção durante períodos em serviço;
- Aumento da satisfação do cliente e a transparência da equipe;
- Possibilidade de avaliar o desempenho das organizações afiliadas;
- Padronização dos conceitos e formas implementados nas organizações executivas com o objetivo de corrigir os métodos existentes e impedir tendências subjetivas para governar vários processos.

O transporte nacional e internacional possui uma movimentação de cargas volumosa, tendo um aumento de situações de avarias, roubos e assaltos a mercadorias, impedindo a entrega da carga (SEST SENAT, 2018). A Figura 2 mostra a movimentação de carga até a distribuição física.

Figura 2 – Movimentação de carga até a distribuição física



Fonte: Adaptado de Sest Senat (2018).

Neste contexto, é importante que empresas adotem medidas de prevenção a risco, tendo controle de perdas e de reparações financeiras dos danos (ROSA, 2007). Segundo Rosa (2007), essas medidas compreendem:

- Retenção de perdas;
- Transferências de perdas;
- Prevenção de riscos com rastreamento;
- Redução de riscos por meio de equipamentos que aumentem a segurança do transporte.

Uma das principais tecnologias aliadas ao gerenciamento de risco é a geolocalização, devido ao fato de que qualquer dispositivo conectado à Internet pode ser facilmente localizado, sendo possível verificar informações sobre a localização do

dispositivo (SEST SENAT, 2018). Para isso, existem ferramentas de monitoramento, como o GPS (sistema de navegação espacial americano que fornece informações de localização e tempo para um aparelho receptor (HOQUE, 2016), e comunicação, como telefonia celular e radiocomunicação, com o intuito de obter o posicionamento dos veículos com o acompanhamento de um dispositivo móvel, sendo possível efetuar ligações telefônicas para postos de controle (SEST SENAT, 2018).

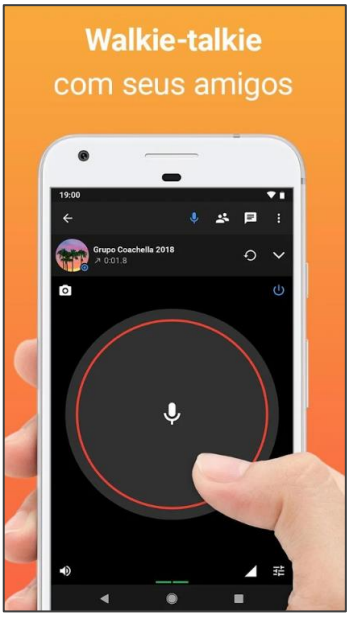
2.3 TRABALHOS CORRELATOS

Existem diversas aplicações que possibilitam a comunicação assíncrona e monitoramento em tempo real. Cinco exemplos, que englobam funcionalidades similares para este tipo de aplicativo são: Zello (2020), *WhatsApp* (2020), *Telegram* (2020), *Life360* (2020) e *MyCarTracks* (2020).

2.3.1 Zello

O aplicativo Zello (2020), lançado oficialmente em 2014, possui a função de transformar o dispositivo móvel em um transceptor de rádio sendo necessário apenas a conexão com a Internet e um dispositivo móvel com sistema operacional Android ou iOS (ZELLO, 2019). Segundo Ferreira (2018), com o intuito de tomar medidas contra assaltos, motoristas da empresa Uber relataram o uso do aplicativo como canal de comunicação entre outros motoristas da empresa. A Figura 3 mostra o aplicativo Zello.

Figura 3 – Aplicativo Zello



Fonte: Zello (2019)

Para realizar a comunicação em tempo real no Zello, é necessário escolher um contato ou um canal e, logo em seguida, pressionar e segurar o botão para falar (ZELLO, 2020). Com isso, o áudio captado no dispositivo móvel do emissor é executado em tempo real no dispositivo receptor. Finalizada essa etapa, é possível que o receptor fale no dispositivo do emissor, mantendo uma comunicação similar a um transceptor de mão.

Esse tipo de comunicação utiliza a tecnologia PTT*,* que funciona de forma bidirecional (ZELLO, 2020). Portanto, o usuário que estiver transmitindo não pode ouvir comunicações e quem está recebendo não pode se comunicar (ZELLO, 2020).

O Zello também possui uma solução *web,* denominada ZelloWork. Essa solução é uma central de gerenciamento, na qual é possível adicionar, remover, editar e silenciar usuários a qualquer momento (ZELLO, 2020). Também é possível criar canais para comunicação com um grupo de usuários, canais ocultos para conversas particulares e canais dinâmicos para grupos sem contato constante (ZELLO, 2020).

Como um complemento opcional, o ZelloWork oferece um recurso de mapeamento em tempo real que exibe a localização atual dos usuários no *OpenStreetMap*, uma base de dados georreferenciados exibida na forma de um mapa (ZELLO, 2020). No entanto, para utilizar os recursos de histórico de localização, tráfego em tempo real e Google Maps[6], é necessário adquirir um plano de 3 dólares

[6] https://www.google.com/maps

por usuário no mês (ZELLO, 2020). Atualmente, na documentação do ZelloWork, não consta que o sistema fornece formas de exibir as rotas dos veículos em serviços de transporte.

2.3.2 WhatsApp

O aplicativo *WhatsApp* (2020), fundado em 2009, e posteriormente incorporado ao *Facebook* em 2014, surgiu como uma alternativa ao uso dos serviços de mensagens curtas (SMS), sendo possível acessá-lo por um dispositivo móvel com sistema operacional Android ou iOS (WHATSAPP, 2019).

Esse aplicativo é gratuito e tem a função de realizar a comunicação entre os usuários, por meio de troca de mensagens de texto, fotos, áudios, vídeos e também uma opção de enviar a localização em tempo real para outro usuário (WHATSAPP, 2019). Além disso, com o recurso de chamadas de voz, o aplicativo utiliza apenas a conexão com a Internet em vez de utilizar o plano do celular do usuário (WHATSAPP, 2019).

As mensagens e chamadas nesse aplicativo são criptografadas de ponta-a-ponta, não permitindo que terceiros, ou o próprio *WhatsApp,* possa lê-las ou ouvi-las (WHATSAPP, 2019). A Figura 4 mostra o aplicativo *WhatsApp*.

Figura 4 – Aplicativo *WhatsApp*

Fonte: WhatsApp (2019)

É possível acessar o *WhatsApp* pela *web* ou com um computador Windows ou Mac, sincronizando as conversas facilmente (WHATSAPP, 2019). Além disso, o

WhatsApp fornece uma solução para proprietários de pequenas, médias ou grandes empresas, denominado *WhatsApp Business*. Essa solução fornece recursos que automatizam, classificam e agilizam as respostas aos clientes, oferecendo suporte e envio de notificações importantes (WHATSAPP, 2019). No entanto, diferentemente do ZelloWork, o *WhatsApp Business* não fornece meios para realizar o mapeamento com localizações em tempo real para múltiplos usuários.

2.3.3 Telegram

O *Telegram* (2020), lançado para a plataforma iOS em 2013 e, posteriormente, para as plataformas Android, Windows, Mac e Linux, é um aplicativo de mensagens gratuito baseado em armazenamento em nuvem, com foco na velocidade e segurança (TELEGRAM, 2020).

Com esse aplicativo, o usuário pode enviar mensagens, fotos, vídeos e arquivos de qualquer tipo, realizar chamadas de voz criptografadas de ponta a ponta, além de compartilhar a localização em tempo real (TELEGRAM 2020). A Figura 5 mostra o aplicativo na plataforma Android.

Figura 5 – Aplicativo *Telegram* para Android

Fonte: Telegram (2020)

Diferentemente do *WhatsApp*, o *Telegram* é um aplicativo de mensagens baseado em nuvem, podendo acessar as mensagens de vários dispositivos móveis

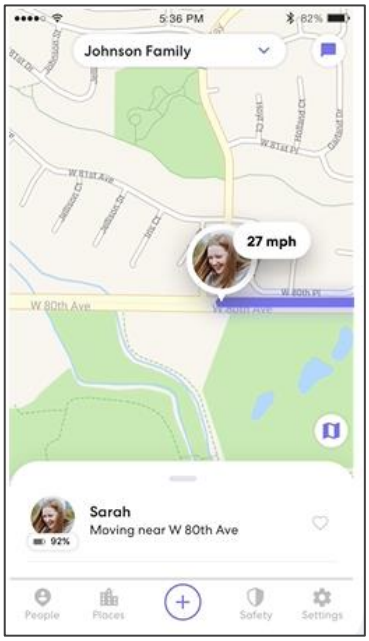
ao mesmo tempo, como celulares, *tablets* e computadores (TELEGRAM, 2020). Além disso, o *Telegram* fornece uma API (*Application Programming Interface*), com o intuito de fornecer aos desenvolvedores de software a possibilidade de criar seus próprios aplicativos de mensagens (TELEGRAM, 2020). Porém, da mesma forma que ao *WhatsApp Business*, o *Telegram* não possui meios de exibir a localização em tempo real dos usuários em um mapa.

2.3.4 Life360

O *Life360* (2020) é um aplicativo de monitoramento em tempo real dos usuários, disponível para Android e iOS (LIFE360, 2020). Seu propósito é ser um localizador familiar, utilizando tecnologias de rastreamento por GPS (LIFE360, 2020).

No aplicativo, um grupo de usuários é denominado círculo (LIFE360, 2020). Em um círculo, é possível disponibilizar a localização em tempo real, iniciar um bate papo com um usuário e enviar mensagens de texto e arquivos para o mesmo (LIFE360, 2020). Além disso, o aplicativo gera alertas no momento em que um membro do círculo chega a um destino determinado (LIFE360, 2020). A Figura 6 mostra o aplicativo *Life360*.

Figura 6 – Aplicativo *Life360*



Fonte: Life360 (2020)

O *Life360* (2020) oferece uma opção *premium* por 164 reais ao ano, em que é possível realizar o monitoramento de velocidade máxima e uso de mensagens de texto, análise de direção segura, 30 dias de histórico de localização e alertas ilimitados de chegada e partida (LIFE360).

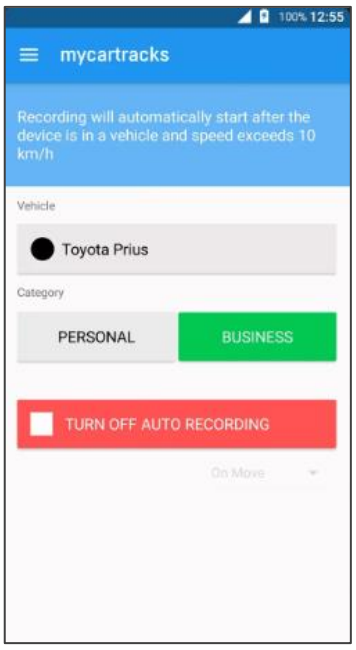
No entanto, o aplicativo possui algumas limitações. Diferentemente do *WhatsApp* e *Telegram*, não é possível efetuar chamadas e enviar áudios no *Life360*. Além disso, não é possível acessar o aplicativo em uma plataforma *web*, ou seja, a localização dos usuários ocorre apenas no aplicativo, dificultando a visualização em um círculo com diversos usuários.

2.3.5 MyCarTracks

O aplicativo *MyCarTracks* (2020) é uma solução para rastreamento de veículos a partir de um *smartphone* (GROW WITH GOOGLE, 2017). Essa solução é usada por empresas para gerenciar efetivamente suas frotas de veículos e otimizar suas operações, registrando as viagens dos motoristas e enviando dados para uma aplicação *web* (GROW WITH GOOGLE, 2017).

O aplicativo, disponível para Android e iOS, permite o uso com diferentes veículos e fornece opções para distinguir se o veículo é para uso pessoal ou profissional (MYCARTRACKS, 2020). Além disso, caso o dispositivo móvel do usuário não possua acesso à Internet, o aplicativo permite a sincronização posterior dos dados a partir de uma conexão Wi-fi com o objetivo de reduzir os custos para o usuário (MYCARTRACKS, 2020). A Figura 7 mostra o aplicativo *MyCarTrack*.

Figura 7 – Aplicativo *MyCarTrack*



Fonte: MyCarTrack (2020)

O MyCarTrack possui uma aplicação *web* na qual é possível administrar a frota de veículos e gerar relatórios (MYCARTRACK, 2020). Uma das funcionalidades principais da aplicação *web* é exibir a localização em tempo real da frota de veículos em um mapa, mostrando informações a respeito da viagem, como, velocidade do veículo, temperatura, umidade e altitude, e o histórico de localização do veículo (MYCARTRACK, 2020).

No entanto, essa solução possui uma limitação. Diferentemente dos aplicativos citados, o *MyCarTrack* não fornece meios de comunicação com a frota de veículos, como mensagens de texto e áudio, dificultando na troca de informações entre a central que realiza a gestão da frota e os motoristas.

2.3.6 Considerações e Limitações

Os cinco aplicativos abordados possuem vantagens e desvantagens no contexto de gerenciamento de frota. *WhatsApp* e *Telegram* são aplicações similares que permitem a comunicação assíncrona entre os usuários, seja por mensagem de texto, imagens, vídeos ou áudios, e o monitoramento em tempo real a partir da geolocalização do dispositivo móvel do usuário. No entanto, é inviável utilizar estas aplicações para realizar a gestão de frota, dado que não é possível visualizar, em uma

aplicação *web,* a localização em tempo real de diversos dispositivos móveis em um único mapa.

As aplicações *MyCarTracks* e *Life360* possuem como principal objetivo o monitoramento em tempo real a partir de geolocalização do dispositivo móvel. Porém, uma diferença entre essas duas aplicações é a possibilidade de envio de mensagens. O *MyCarTracks* não possui meios de efetuar uma comunicação direta entre os motoristas e o gestor da frota, dificultando a troca de informação.

No *Life360* é possível realizar essa comunicação entre os usuários. Porém, a desvantagem dessa aplicação*,* no contexto de gestão de frota, é a ausência da funcionalidade de realizar chamadas e envios de áudios.

O Zello, principal trabalho correlato, possui diversas funcionalidades que auxiliam na gestão de frota, como monitoramento em tempo real em uma aplicação *web* e comunicação via áudio, sendo útil para gestores de frota e motoristas. A funcionalidade de monitoramento é um complemento opcional para a aplicação *web*, sendo restrito ao uso da tecnologia *OpenStreetMap*.

2.4 TECNOLOGIAS

Nesta seção é apresentada a fundamentação teórica sobre as tecnologias para o desenvolvimento do projeto.

2.4.1 Aplicativos Móveis Nativos

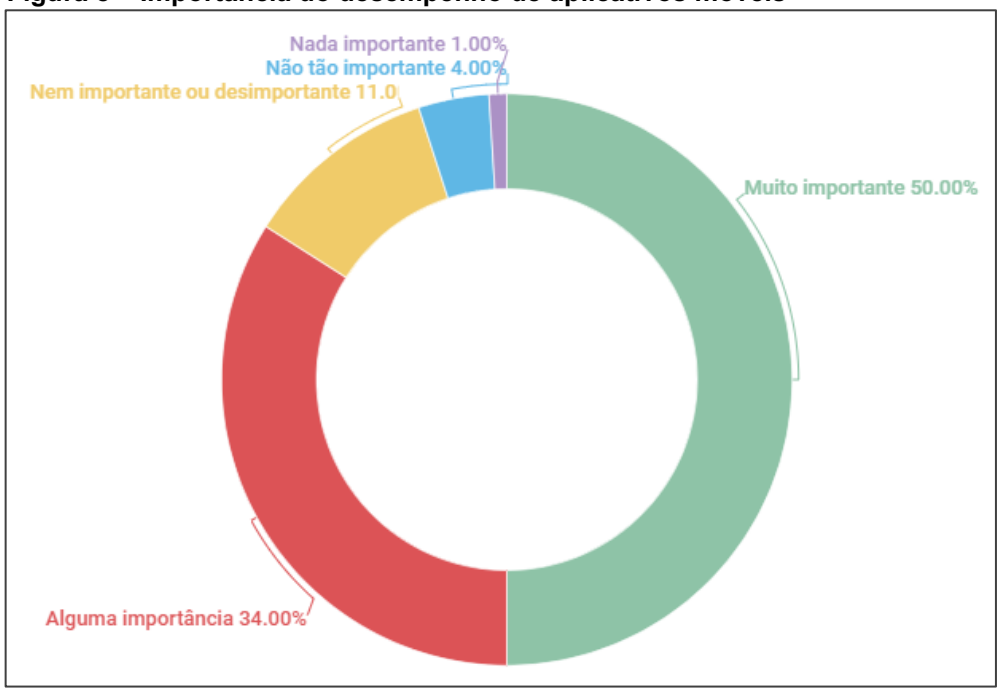
Os aplicativos nativos são desenvolvidos especificamente para um sistema operacional, como iOS e Android (MORE; CHANDRAN, 2016). Esses aplicativos podem utilizar todos os recursos do dispositivo móvel, como câmera, GPS, acelerômetro, bússola, lista de contatos e gestos (MORE; CHANDRAN, 2016). Esses recursos são essenciais para um aplicativo que exija dados do dispositivo, como a localização geográfica e os movimentos do dispositivo móvel (MORE; CHANDRAN, 2016).

Em comparação com aplicativos híbridos, ou seja, aplicativos que podem ser executados em diversas plataformas com um único código-fonte, os aplicativos

nativos possuem melhor performance, segurança e fornece uma melhor experiência para o usuário (YML, 2019).

À medida que os usuários navegam em um aplicativo nativo, a estrutura e elementos visuais já estão disponíveis no dispositivo móvel, possibilitando um carregamento instantâneo e proporcionando uma melhor experiência para o usuário (YML, 2019). A Figura 8 mostra que 84% dos usuários consideram o desempenho um importante fator no uso de um aplicativo (YML, 2019).

Figura 8 – Importância do desempenho de aplicativos móveis



Fonte: Adaptado de YML (2019)

O Android é um sistema operacional criado para uma ampla variedade de dispositivos com diferentes formatos, tendo como principal objetivo criar uma plataforma de software aberta disponível para que operadoras e desenvolvedores possam apresentar um produto que aprimora a experiência dos usuários (ANDROID, 2020). Esse sistema operacional consta com código-fonte aberto, podendo qualquer usuário baixar e realizar personalizações (ANDROID, 2020).

A *Integrated Development Environment* (IDE) oficial para desenvolvimento de aplicativos para Android é o Android Studio (ANDROID, 2020). Essa IDE fornece um editor de código, ferramentas avançadas para o desenvolvedor e recursos para aumentar a produtividade na compilação de aplicativos Android.

Atualmente, a linguagem de programação oficial para desenvolvimento Android é o Kotlin (MEVADA, 2017), substituindo a linguagem Java. O Kotlin é essencial para desenvolvimento de aplicativos Android, pois possui diversas vantagens, como (KOTLIN, 2020):

- Compatibilidade: aplicativos desenvolvidos em Kotlin podem ser executados em versões mais antigas do Android, pois as ferramentas do Kotlin são compatíveis com o sistema de compilação e as ferramentas são suportadas pelo Android Studio;
- Desempenho: um aplicativo desenvolvido em Kotlin é tão rápido quanto um desenvolvido em Java. Além disso, com a utilização de funções *lambda*[7], o aplicativo é executado ainda mais rápido em comparação ao Java;
- Interoperabilidade: o Kotlin é interoperável com o Java, permitindo que um aplicativo desenvolvido em Kotlin possa utilizar todas as bibliotecas existentes do Android.

A vantagem do sistema Android é que ele não está vinculado a um único fabricante de dispositivo ou a um preço relativamente limitado, e por isso é popular em países de baixa renda (MAHAPATRA, 2013). Para consumidores que não são sensíveis ao preço, empresas como Samsung e HTC produzem telefones com sistema operacional Android com a tecnologia mais recente no mercado (MAHAPATRA, 2013).

2.4.2 Aplicações *Web*

Aplicações *web* são sistemas desenvolvidos para serem executados em um navegador, utilizando linguagens de programação como JavaScript e HTML[8] (AL-SAGHADI, 2011). Essas aplicações são acessadas por meio de uma conexão de rede, em vez de existir na memória do dispositivo (TECHNOPEDIA, 2020).

Os sites tradicionais introduzidos no começo da Internet, tinham o objetivo de fornecer páginas estáticas ao cliente, utilizando HTML e CSS (JADHAV; SAWANT; DESHMUKH, 2015). Porém, com o passar dos anos, as pessoas necessitaram de

[7] Funções que não são declaradas, mas são transmitidas como expressão (KOTLIN, 2020). Podem ser passadas como parâmetro ou retornadas de outras funções.
[8] Linguagem de marcação para criação de páginas na *web*.

sites dinâmicos com acesso a uma camada de persistência de dados, e com isso, tecnologias como Javascript e AJAX começaram a ser utilizados (JADHAV; SAWANT; DESHMUKH, 2015).

Para atingir a mesma experiência que os usuários possuem em aplicações *desktop*, surgiu a *Single Page Application* (SPA) (JADHAV; SAWANT; DESHMUKH, 2015). A SPA é composta por componentes individuais que podem ser atualizados e substituídos, sem a necessidade de recarregar a página a cada ação do usuário (JADHAV; SAWANT; DESHMUKH, 2015). Isso faz com que a experiência do mesmo seja aprimorada, devido a menos recargas e melhor desempenho geral, sendo necessário menos largura de banda em relação a métodos tradicionais (ANGULAR, 2019).

Para o desenvolvimento *web* de uma SPA, existem diversos *frameworks* javascript, como Angular (2020), React (2020) e Vue (2020). Angular é um *framework* SPA para construção de aplicações *web* utilizando tecnologias como HTML e TypeScript (ANGULAR UNIVESITY, 2019), sendo popular para desenvolvimento *front-end* (KUMAR, 2018).

Diferentemente dos *frameworks* React e Vue, o Angular fornece todos os pacotes necessários para desenvolver uma aplicação SPA, como validação de formulários e roteador de páginas (KALUZA; TROSKOT; VUKELIC, 2018). Além disso, esse *framework* permite que códigos complexos sejam escritos e mantidos (KALUZA; TROSKOT; VUKELIC, 2018), permitindo que aplicações se tornem manuteníveis e escaláveis. Portanto, para o desenvolvimento de uma aplicação SPA, o Angular é o mais adequado (KALUZA; TROSKOT; VUKELIC, 2018).

Angular aborda alguns dos melhores aspectos do desenvolvimento *front-end* e os utiliza para aprimorar o HTML no navegador, tornando mais simples e fácil a criação de aplicações (FREEMAN, 2017). Devido a isso, o Angular é popular entre os desenvolvedores e usado por organizações como Google[9], *WhatsApp* e Instagram[10] (KUMAR, 2018).

As aplicações em Angular são desenvolvidas utilizando um padrão de projeto de software chamado *Model-View-Controller* (MVC) (FREEMAN, 2017). Esse padrão

[9] https://www.google.com/
[10] https://www.instagram.com/

enfatiza que as aplicações sejam extensíveis, manuteníveis, testáveis e padronizados (FREEMAN, 2017).

Para o desenvolvimento de uma aplicação em Angular, existem diversas IDEs, mas a principal e amplamente utilizada atualmente é o *Visual Studio Code*[11]. Essa IDE é um editor de código-fonte leve, disponível para Windows, macOS e Linux, e possui suporte interno para Javascript e Typescript (VISUAL STUDIO CODE, 2020).

2.4.3 Banco de Dados NoSQL

NoSQL, ou *non-relational databases*, é um sistema de gerenciamento de banco de dados (SGBD) não relacional de código aberto que possui a função de armazenar grandes quantidades de dados (SHARMA; DAVE, 2012). Bancos de dados NoSQL se tornaram populares com o advento de aplicativos móveis e *web*, *big data* e armazenamento em nuvem (IBM, 2019).

NoSQL permite maior agilidade e flexibilidade, tendo um *design* mais simples, maior escabilidade e controle (IBM, 2019). Esse tipo de SGBD abrange dados estruturados, que permite que mecanismos de pesquisa entendam o formato do código, e dados semiestruturados, contendo *tags* para separação de elementos semânticos e hierarquias nos registros e dados (IBM, 2019).

Com o avanço da Internet e disponibilidade de armazenamento, grandes quantidades de dados são armazenadas para diversas aplicações. Realizar o processamento desses dados requer velocidade, esquemas flexíveis e banco de dados distribuídos. Para isso, os bancos de dados NoSQL se tornaram a principal opção (LI; MANOHARAN, 2013).

Os bancos de dados relacional são baseados no modelo ACID (*Atomicty, Consistency, Isolation, Durability*) com o objetivo de garantir a consistência dos dados, enquanto o NoSQL é baseado no modelo BASE (*Basically Available, Soft-state, Eventualy consistent*) para obter maior escalabilidade, disponibilidade e desempenho (KABAKUS; KARA, 2017). Desse modo, diferentemente do NoSQL, o modelo relacional apresenta problemas de escalabilidade, prejudicando o desempenho à medida que o volume de dados aumenta (NAYAK; PORIYA; POOJARY, 2013).

[11] https://code.visualstudio.com/

Além disso, em comparação com banco de dados relacionais, o NoSQL não segue os princípios dos sistemas de gerenciamento de banco de dados relacional, ou seja, não armazena dados em tabelas (GYÕRÖDI *et al*, 2015). Com isso, os bancos NoSQL podem ser classificados em quatro tipos:

- Chave-valor: os registros consistem em pares de chave e valor (STONEBRAKER, 2011). Cada valor armazenado no banco de dados possui uma chave de identificação (GYÕRÖDI *et al*, 2015). DynamoDB[12] e Redis[13] são exemplos de bancos de dados chave-valor;
- Orientado a documentos: similar ao chave-valor, porém requer que os dados sejam estruturados e codificados por JSON[14], XML ou outros (LI, 2018). Exemplos desse tipo de banco de dados inclui o MongoDB[15] e Apache CouchDB[16];
- Orientado a grafos: consiste em armazenar os dados na forma de um grafo (NAYAK; PORIYA; POOJARY, 2013). Um grafo é formado por nós e arestas, em que os nós são entidades e as arestas são relacionamentos (LI, 2018). *Neo4j*[17] é o principal exemplo de banco de dados orientado a grafos;
- Família de colunas: possui uma estrutura de dados distribuída e orientada a colunas, podendo armazenar diversos atributos em uma chave (MONIRUZZAMAN; HOSSAIN, 2013). Em comparação com os bancos de dados baseados em linhas, as consultas são mais rápidas e apenas as colunas necessárias são lidas (SHARMA; DAVE, 2012). Cassandra[18] e HBase[19] são exemplos de banco de dados orientado a colunas.

Levando em consideração os dados apresentados, será utilizado um banco de dados NoSQL, devido ao seu melhor desempenho quando comparado a bancos de dados SQL. O mesmo será do tipo chave-valor, pois possui recuperação rápida e altamente escalável dos dados necessários para uma aplicação (MONIRUZZAMAN; HOSSAIN, 2013).

---

[12] https://aws.amazon.com/pt/dynamodb/
[13] https://redis.io/
[14] JSON é um texto escrito com notação de um objeto Javascript.
[15] https://www.mongodb.com/
[16] https://couchdb.apache.org/
[17] https://neo4j.com/
[18] https://cassandra.apache.org/
[19] https://hbase.apache.org/

2.4.4 Computações em Nuvem

A computação em nuvem é um modelo para permitir acesso onipresente e sob demanda a um conjunto compartilhado de recursos de computação, como armazenamento em banco de dados, servidores, aplicativos e serviços (MELL; GRANCE, 2011). Uma plataforma de serviços em nuvem fornece acesso rápido a recursos de TI por meio da Internet, definindo o preço desses recursos conforme o uso (AWS, 2020).

Existem cinco características essenciais que definem a computação em nuvem (MELL; GRANCE, 2011):

- Autoatendimento sob demanda: Um consumidor pode provisionar recursos de computação de forma automática, como tempo do servidor e armazenamento de dados, sem exigir interação humana com cada provedor de serviços;
- Amplo acesso à rede: Os recursos estão disponíveis na rede e acessados por mecanismos padrão que promovem o uso por plataformas heterogêneas de clientes, como celulares, *tablets* e *notebooks*;
- Agrupamento de recursos: Os recursos de computação do provedor são agrupados para atender vários consumidores usando um modelo multilocatário, com diferentes recursos físicos e virtuais dinamicamente atribuídos e reatribuídos de acordo com a demanda do consumidor. Existe uma sensação de independência de localização em que o cliente geralmente não tem controle ou conhecimento sobre a localização exata dos recursos fornecidos. Exemplos de recursos incluem armazenamento, processamento, memória e largura de banda da rede;
- Rápida elasticidade: Os recursos podem ser provisionados elasticamente e liberados para escalar rapidamente, de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento geralmente parecem ilimitados e podem ser apropriados em qualquer quantidade a qualquer momento;
- Serviço medido: Os sistemas em nuvem controlam e otimizam automaticamente o uso de recursos, por meio da capacidade de medição em

> algum nível de abstração apropriado ao tipo de serviço, como armazenamento, processamento e contas ativas de usuário. O uso desses recursos pode ser monitorado, controlado e relatado, fornecendo transparência ao fornecedor e ao consumidor do serviço utilizado.

Um dos principais benefícios da computação em nuvem é a oportunidade de substituir as despesas iniciais de infraestrutura de capital por baixos custos variáveis (AWS, 2020). Com a nuvem, as empresas podem ativar instantemente milhares de servidores em minutos e entregar resultados rapidamente, não necessitando de um planejamento com infraestruturas de TI (AWS, 2020). Uma plataforma de serviços em nuvem, como a *Amazon Web Services* (2020), possui e mantém essa infraestrutura, enquanto o cliente provisiona e usa o que precisa por meio de uma aplicação *web.*

A *Amazon Web Services* (AWS) é uma plataforma em nuvem que oferece mais de 175 serviços em todo mundo (AWS, 2020). A AWS abrange 69 zonas de disponibilidade em 22 regiões geográficas em todo mundo, com planos anunciados para mais 16 zonas de disponibilidade e mais cinco regiões da AWS na Indonésia, Itália, Japão, África do Sul e Espanha (AWS, 2020). A Figura 9 representa a rede global de regiões da AWS.

Figura 9 – Rede global de regiões da AWS

Fonte: AWS (2020)

Em 2006, a AWS iniciou a oferecer serviços de computação em nuvem (AWS, 2020). Atualmente, essa plataforma fornece uma infraestrutura altamente confiável,

escalável e de baixo custo, que abastece centenas de milhares de empresas em 190 países (AWS, 2020).

A AWS oferece uma quantidade maior de serviços em comparação com outros provedores de nuvem, como armazenamento de dados e tecnologias emergentes como *machine learning* e inteligência artificial (AWS, 2020). Para o presente trabalho, foram utilizados os serviços: DynamoDB, Cognito, Lambda, API Gateway, S3 e SNS.

O DynamoDB é um serviço de banco de dados chave-valor com desempenho rápido e escalável (AWS, 2020). Ele replica automaticamente os dados em vários servidores, fornecendo alta disponibilidade e durabilidade (AWS, 2020). Além disso, pode suportar 10 trilhões de requisições por dia e 20 milhões de requisições por segundo (AWS, 2020). O DynamoDB conta com um nível gratuito de uso, disponibilizando 25 GigaByte (GB) de armazenamento físico de dados e 1 GB de transferência de dados agregado com os serviços da AWS (AWS, 2020). Portanto, ele foi utilizado para a persistência dos dados do trabalho.

Lambda é um serviço que permite que um código seja executado sem gerenciar ou provisionar servidores (AWS, 2020). Esse serviço processa cada acionamento individualmente e executa o código paralelamente, escalando de acordo com a necessidade (AWS, 2020). Dessa forma, é necessário apenas escrever o código *back-end* e o serviço realiza toda a administração (AWS, 2020). Esse serviço também conta com um nível gratuito que inclui 1 milhão de solicitações gratuitas por mês e 400.000 GB/segundos de tempo de computação por mês (AWS, 2020).

No entanto, para desenvolver o *back-end*, é necessário utilizar uma ferramenta que seja compatível com a *Lambda*. Para isso, foi utilizado o NodeJS, uma plataforma javascript para criação de código *back-end* (MADSEN; TIP; LHOTÁK, 2015) altamente performática e confiável (BUTTIGIEG; JEVDJENIC, 2015).

Para que seja possível que o *front-end* e *back*-end possam se comunicar, é necessária uma API. Para isso, foi utilizado o API *Gateway,* serviço que permite a criação, publicação, monitoramento e proteção de APIs (AWS, 2020). Com esse serviço, é possível criar APIs que permitem a comunicação em tempo real, utilizando o RESTFul (serviço que implementa o padrão REST[20]) e *WebSocket* (serviço que

[20] Conjunto de restrições arquiteturais que visam minimizar a latência e comunicação de rede e maximizar a escalabilidade e a independência de componentes rede (FIELDING; TAYLOR, 2000).

utiliza o protocolo TCP, permitindo a comunicação bidirecional cliente-servidor, sendo ideal para aplicativos de bate-papo) (AWS, 2020).

Além disso, esse serviço possui um nível gratuito que inclui um milhão de chamadas de API REST e um milhão de mensagens e 750.000 minutos de conexão para APIs do *WebSocket* por mês durante até 12 meses (AWS, 2020).

Para realizar a segurança dos recursos do sistema, foi utilizado o Cognito, serviço para autenticação de usuários em aplicações móveis e *web*, sendo altamente escalável (AWS, 2020). Esse serviço permite o cadastro de usuários, autenticação por meio de provedores externos, como *Facebook*, *Twitter* e *Amazon*, e a possiblidade de disponibilizar credenciais para acesso temporário de recursos *back-end* (AWS, 2020). Esse serviço também faz parte de nível gratuito da AWS, disponibilizando diversos recursos, como 10 GB de armazenamento e um máximo de 50 mil usuários ativos mensais (AWS, 2020).

O S3 é um serviço de armazenamento de objetos (AWS, 2020). Esse serviço possui alta escalabilidade, segurança e desempenho, sendo o líder do setor no mercado (AWS, 2020). Além disso, o S3 cria e armazena automaticamente cópias de todos os objetos com o objetivo de disponibilizar os dados sempre que necessário, protegidos contra falhas e erros (AWS, 2020).

O S3 possui um nível gratuito que inclui 5 GB de armazenamento, 20.000 solicitações para buscar objetos, 2.000 solicitações para inserir ou atualizar um objeto e 15 GB de transferência de dados a cada mês (AWS, 2020). Portanto, foi utilizado o S3 para armazenamento dos arquivos do sistema.

Em um sistema de *chat*, é necessário o uso de notificações para informar ao usuário o recebimento de uma mensagem. Para isso, a AWS possui um serviço chamado *Simple Notification Service* (SNS)

Esse serviço tem o objetivo de enviar notificações para celular, SMS e e-mail para os usuários (AWS, 2020). Essas notificações podem aparecer como alerta de mensagens, alertas sonoros e atualizações de *badges*[21] (AWS, 2020).

O nível gratuito do SNS possui diversos recursos, como 1.000.000 de notificações para dispositivos móveis, 100 envios de SMS para qualquer lugar do mundo e 1.000 envios de e-mail (AWS, 2020). Portanto, devido aos seus recursos

[21] Indicações numéricas que representam quantos itens estão associados a um *link* (W3SCHOOLS, 2020).

gratuitos e ser altamente disponível e seguro (AWS, 2020), foi utilizado o SNS para gerenciar os envios de notificações no sistema.

3 DESENVOLVIMENTO

Neste capítulo é apresentado o resultado do projeto. A Seção 3.1 descreve a análise de requisitos, com especificação de tipos de usuários, diagrama de casos de uso e definição de requisitos funcionais e não funcionais. A Seção 3.2 descreve a arquitetura do sistema e como cada componente se comunica, com diagramas exemplificando o fluxo das principais funcionalidades. A Seção 3.3 descreve os resultados obtidos.

3.1 ANÁLISE DE REQUISITOS

Os requisitos são atributos definidos para um sistema antes mesmo de desenvolvê-lo. Para isso, é realizada uma análise com intuito de identificar um conjunto de recursos que satisfaçam as necessidades do sistema (GRADY, 2010).

Nesta seção é apresentada essa análise para o desenvolvimento do projeto, como definição de grupos de usuários e suas respectivas permissões, diagrama de caso de uso e arquitetura.

3.1.1 Grupos de Usuário

Três grupos de usuários foram definidos para acesso ao sistema: funcionário, operador e administrador. O funcionário possui a função de enviar informações e relatórios aos operadores da frota, como dados de rota ou requisitar auxílio para uma situação específica. Esse grupo não possui permissão de acesso aos dados da frota, ou seja, a localização de cada veículo é restrita apenas ao operador e administrador.

O operador é responsável por monitorar a frota de veículos e manter uma comunicação assíncrona com os funcionários. O administrador tem permissão total no sistema, possibilitando a criação de novos usuários, definindo seus pré-cadastros e o grupo ao qual os mesmos pertencem.

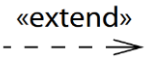
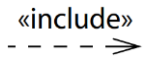
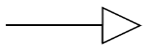
3.1.2 Diagrama de Casos de Uso

O Diagrama de Casos de Uso é um modelo que faz parte de uma linguagem para documentação e modelagem denominada UML (*Unified Modeling Language*). Sua utilização permite uma visão e compreensão geral do sistema a ser desenvolvido.

Esse diagrama possui três componentes principais: Ator, Caso de Uso e Relacionamentos. O Ator é representado por uma pessoa externa, um processo ou algo interagindo com o sistema (RUMBAUGH, JACOBSON, BOOCH; 2004). No presente trabalho, os atores são identificados como funcionários, operadores e administradores.

O Caso de Uso, simbolizado como uma elipse, representa uma funcionalidade interativa do sistema, podendo ter diversos relacionamentos e associações com atores (RUMBAUGH, JACOBSON, BOOCH; 2004). A Tabela 1 define os diferentes tipos de relacionamentos.

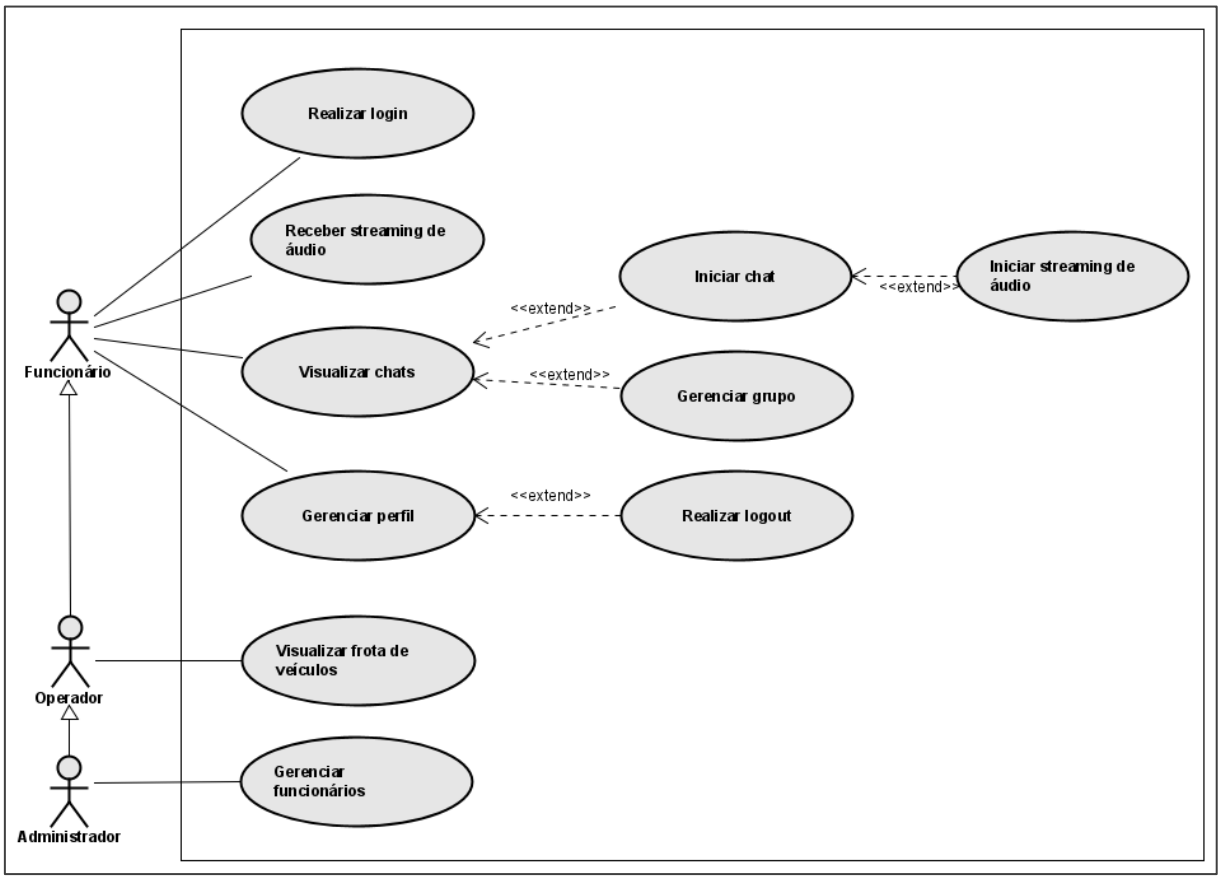
Tabela 1 - Tipos de relacionamentos

Relacionamentos	Função	Notação
Associação	O caminho de comunicação entre um ator e um caso de uso.	
Extensão	Define um relacionamento opcional que pode ser inserido no comportamento do caso de uso base.	
Inclusão	Define a inclusão de um comportamento de um caso de uso no caso de uso base.	
Generalização	Relacionamento entre um caso de uso geral com um caso de uso específico que herda seus comportamentos.	

Fonte: RUMBAUGH, JACOBSON, BOOCH; 2004

Além disso, é possível realizar um relacionamento de generalização entre atores (RUMBAUGH, JACOBSON, BOOCH; 2004). Portanto, um ator específico pode herdar os atributos e relacionamentos de um ator geral. A Figura 10 apresenta o Diagrama de Caso de Uso do projeto, definindo os relacionamentos entre os atores e os casos de uso.

Figura 10 – Diagrama de Caso de Uso



Fonte: Autoria própria

Os atores “funcionário” e “operador” possuem um relacionamento de generalização, ou seja, o operador herda os atributos do funcionário, possibilitando interagir com os casos de usos em que o funcionário está se relacionando. Da mesma forma, o ator “operador” e “administrador” possuem esse relacionamento, permitindo que o administrador interaja com casos de usos que o operador possui relacionamento.

Os casos de uso presente no diagrama são:

1) Realizar *login*: primeira funcionalidade a ser executada. É necessário que o ator realize seu *login* para interagir com os demais casos de usos do sistema;
2) Visualizar *chats*: funcionalidade para visualizar todos os *chats* em que o ator pertence, seja ele privado ou um grupo;
3) Iniciar *chat*: funcionalidade opcional para iniciar um *chat* entre dois ou mais atores;

4) Gerenciar grupo: funcionalidade opcional para editar os dados de um *chat* com vários atores;
5) Iniciar *streaming* de áudio: funcionalidade opcional que permite que o ator grave sua voz e execute-a instantaneamente no dispositivo de atores que pertencem a um *chat* ou grupo;
6) Receber *streaming* de áudio: O ator pode receber em seu dispositivo móvel os dados de um áudio instantaneamente para ser executado em seu dispositivo, sem a necessidade de realizar outra ação;
7) Gerenciar perfil: funcionalidade que permite o ator adicionar, editar ou remover seus dados cadastrais;
8) Realizar logout: funcionalidade para sair do sistema;
9) Visualizar frota de veículos: funcionalidade para visualizar, em tempo real, a frota de veículos em um mapa;
10) Gerenciar funcionários: funcionalidade para adicionar, editar e remover funcionários do sistema.

A descrição detalhada de cada caso de uso consta no Apêndice A.

3.2 ARQUITETURA

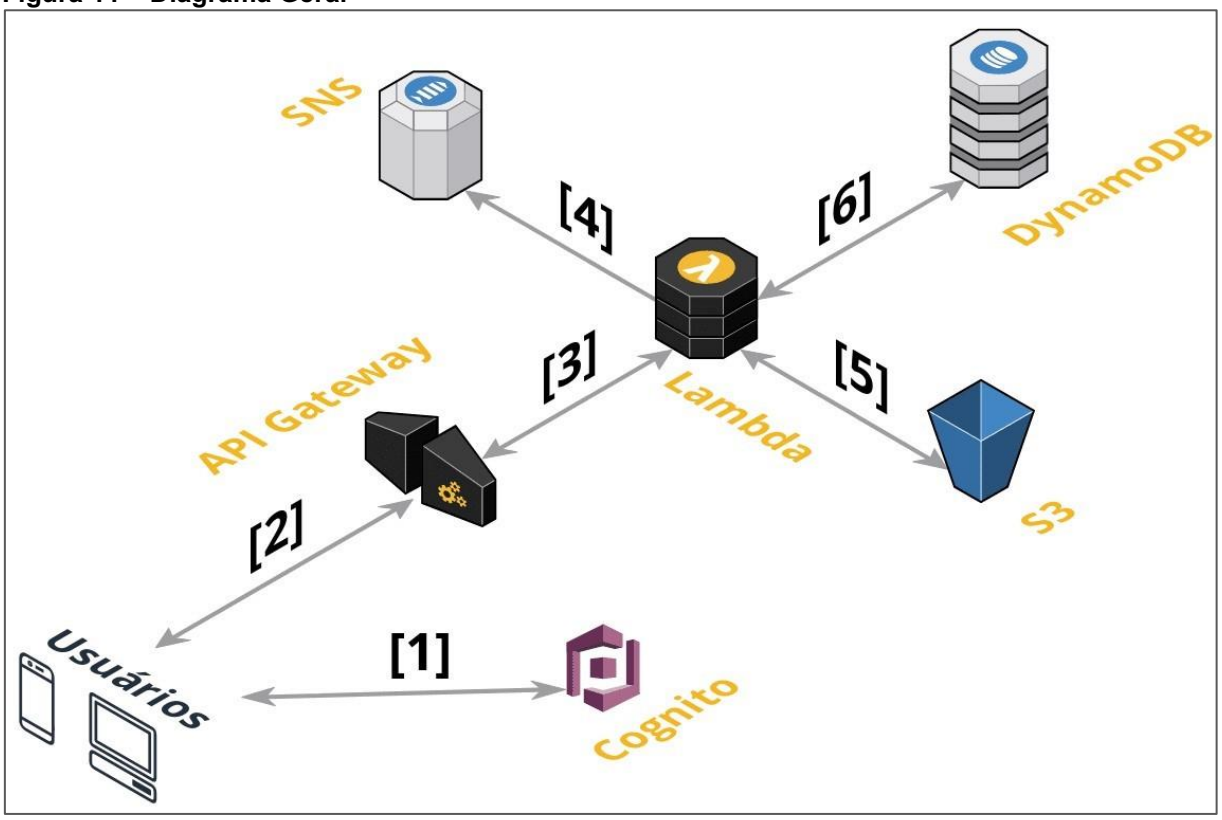
Segundo CLEMENTS *et al*. (2002), a arquitetura de um sistema se refere a um conjunto de estruturas necessárias para dar logicidade ao sistema, especificando as propriedades e relacionamentos dos componentes envolvidos. Portanto, nesta seção é apresentada a arquitetura definida para o presente projeto.

A Seção 3.2.1 apresenta o Diagrama Geral, com todos os serviços utilizados e suas comunicações. A Seção 3.2.2 descreve como é realizada a funcionalidade PTT, por meio de *streaming* de áudio. A Seção 3.2.3 apresenta o fluxo de como a geolocalização dos veículos da frota chegam até a aplicação *web*, para que os gestores possam visualizar a frota em tempo real. A Seção 3.2.4 apresenta a comunicação entre o operador de frota e um funcionário, por meio de um *chat* privado.

3.2.1 Diagrama Geral

A Figura 11 representa o Diagrama Geral do Sistema, mostrando como os componentes e serviços se relacionam. Esse diagrama possui operações numeradas de 1 a 6 que serão descritas a seguir.

Figura 11 – Diagrama Geral



Fonte: Autoria Própria.

A operação [1] é a primeira interação dos usuários com o sistema: realizar *login*. Os usuários preenchem suas credenciais na parte do cliente (aplicação *web* ou aplicativo móvel) para realizar uma autenticação e liberar o acesso aos recursos do sistema.

Após ter o acesso, a operação [2] pode ser executada. O usuário realiza uma requisição HTTP ou *websocket* para a API, iniciando assim uma comunicação com o *back-end*. Nesta operação, a API valida a requisição feita pelo usuário e se o mesmo possui permissão de acesso para que a operação [3] seja executada.

Na operação [3], uma função FaaS[22] é executada. Nesta operação, a função recebe os parâmetros recebidos da API para interpretar e validar a requisição do usuário. Além disso, são nessas funções em que está concentrada toda a lógica de negócio do sistema.

Na operação [4], a função pode requisitar o envio de notificações para usuários específicos. Ao executar um serviço de notificação, como o SNS, é o usuário receberá em seu dispositivo móvel uma notificação.

Na operação [5], algumas requisições podem necessitar que um arquivo seja armazenado, como fotos de perfil de funcionários e áudios em *chats*. Para isso, é utilizado um serviço para armazenamento desses objetos.

Na operação [6], a função realiza a comunicação com o banco de dados não-relacional, podendo realizar listagem, inserção, atualização e deleção de registros. Após receber a resposta do banco de dados, a função manipula e processa os dados para ser retornado ao cliente.

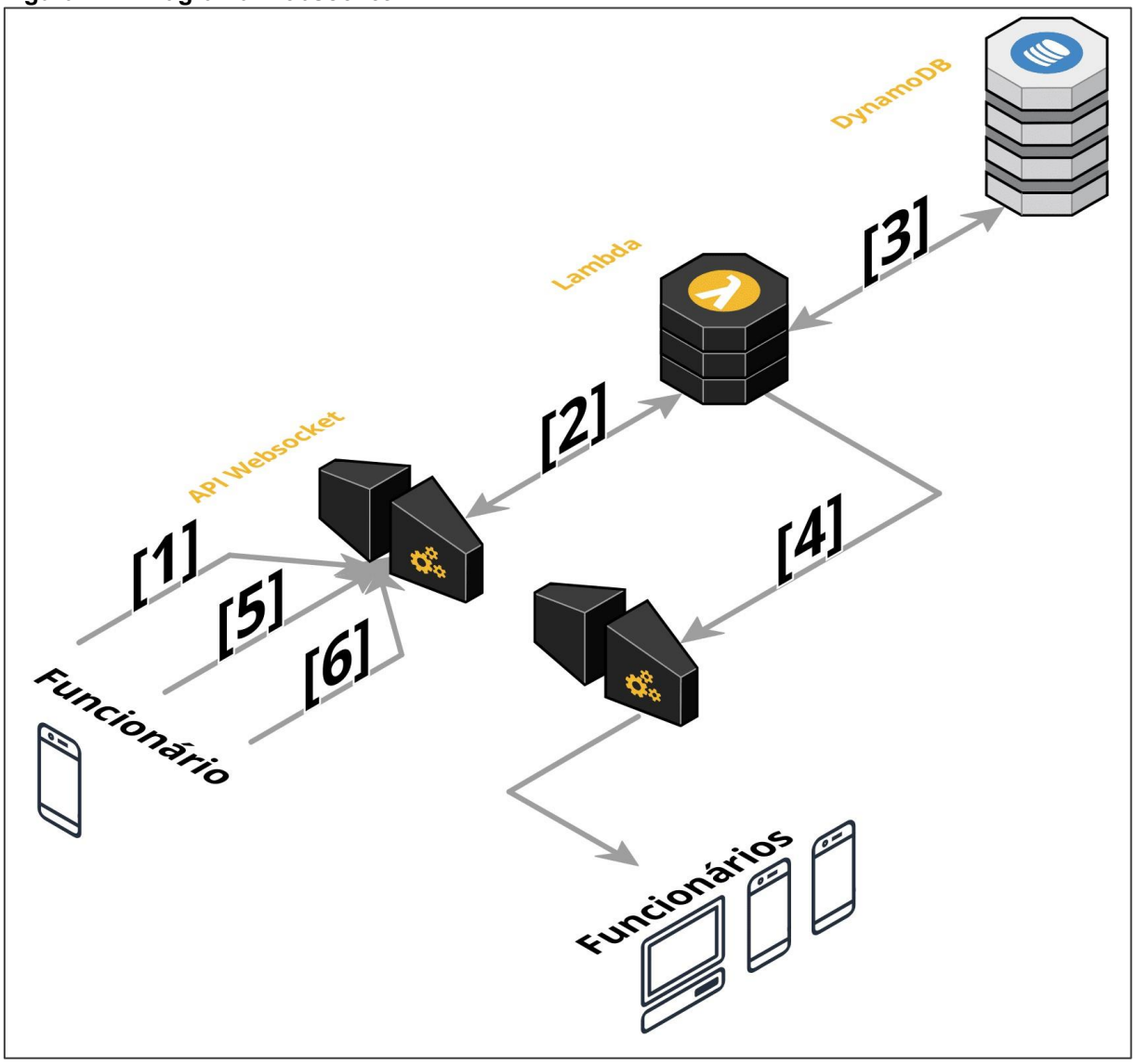
A partir do Diagrama Geral, é possível especificar como é realizada uma conexão *websocket* e as principais funcionalidades do sistema, como o *streaming* de áudio, envio da geolocalização dos veículos e o *chat* em tempo real.

3.2.2 Diagrama Websocket

A Figura 12 representa o fluxo completo com uma API *Websocket*, como conexão, desconexão e envio de uma mensagem a partir de uma rota customizada.

[22] *Function as a Service*, ou FaaS, é um serviço de computação em nuvem que permite executar código sem precisar gerenciar servidores (AWS, 2021).

Figura 12 – Diagrama *Websocket*



Fonte: Autoria própria.

Na operação [1], o funcionário realiza uma tentativa de conexão por meio da rota *$connect*. Para poder ser efetuada essa conexão, é necessário validar se o funcionário está autenticado e possui permissões de acesso.

Para isso, a operação [2] invoca uma função responsável por gerenciar a conexão e desconexão da API. Na conexão, a função verifica se o usuário está autenticado e armazena seu *connectionId*[23] (operação [3]).

[23] Identificador único de conexão utilizado para que o cliente possa receber atualizações em tempo real (AWS, 2020).

Na operação [4], é enviada uma atualização a todos os funcionários ativos no sistema, ou seja, todos os que possuem um *connectionId* ativo. Nesse caso, será enviada uma atualização informando que o funcionário está *online*.

Na operação [5], o usuário envia uma mensagem utilizando uma rota customizada da API *Websocket*. Para o presente projeto, foi definido as seguintes rotas:

1) *SEND_MESSAGE*: responsável por enviar uma mensagem de áudio ou texto em um *chat*;
2) *SEND_LOCATION*: responsável por enviar a localização do veículo a partir do dispositivo móvel do funcionário;
3) *PUSH_TO_TALK*: responsável pelo *streaming* de áudio;
4) *OPEN*_MESSAGES: responsável por sinalizar que o funcionário recebeu as mensagens enviadas em um *chat*.

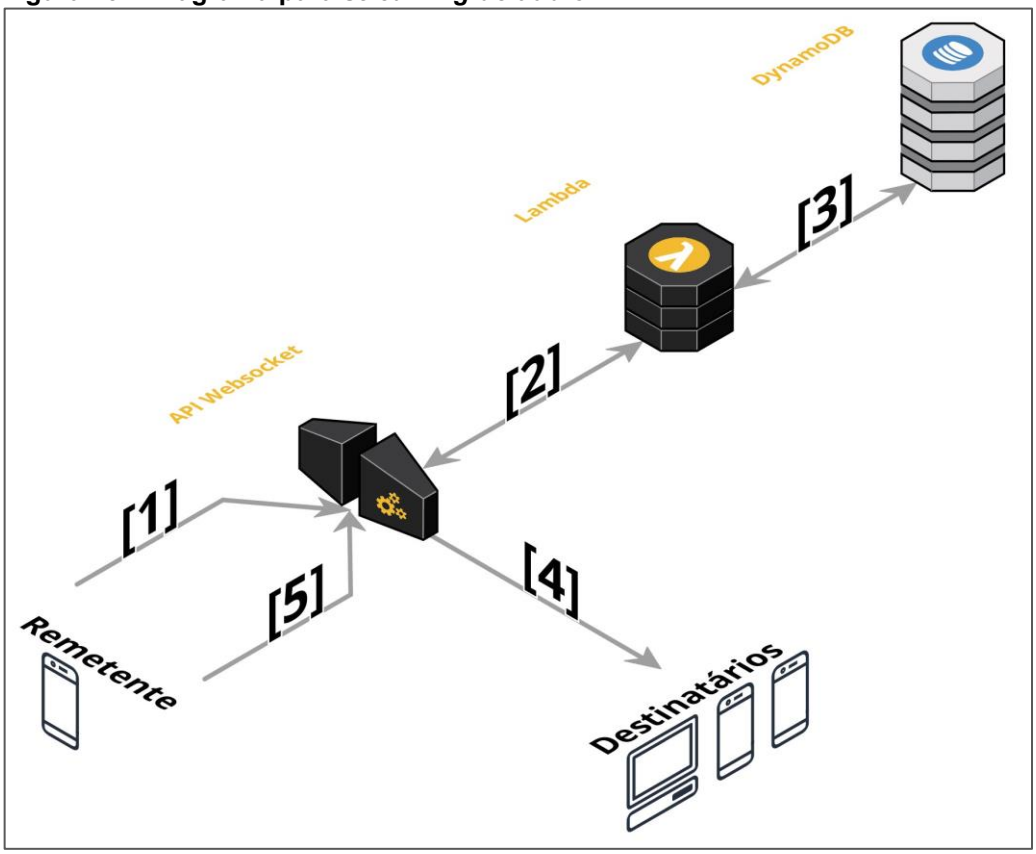
Caso uma mensagem enviada a API não esteja dentre as rotas definidas acima, a rota *$default* será executada. Essa rota será sempre executada se a mensagem estiver em um formato indevido, ou seja, não está em formato JSON, ou não possui nenhuma rota correspondente (AWS, 2021).

No momento em que o funcionário deixa de usar o sistema, a operação [6] é executada, sendo realizada uma requisição a rota *$disconnect*, para desconexão do *websocket*. Essa requisição segue o mesmo fluxo da conexão, no entanto, a operação [4] envia para os funcionários ativos a atualização de que o usuário está *offline*.

3.2.3 Diagrama para *streaming* de áudio

A Figura 13 exemplifica o fluxo para ser realizado *streaming* de áudio, ao utilizar a funcionalidade PTT.

Figura 13 – Diagrama para *streaming* de áudio



Fonte: Autoria própria.

Na operação [1], o remetente segura o botão de microfone na interface do sistema para iniciar a funcionalidade. Após essa etapa, é enviado a API *Websocket* uma mensagem sinalizando que o *streaming* de áudio foi iniciado por um usuário.

Na operação [2], a função processa a requisição do usuário e, logo após, realiza uma listagem de todas as conexões *websocket* ativas dos destinatários (operação [3]). Ao ser retornado às conexões, uma mensagem é enviada por meio de *websocket* para todos os destinatários, sinalizando que um usuário iniciou o *streaming* de áudio, sendo necessário aguardar que o mesmo finalize a sua ação para que outro usuário possa iniciar o PTT.

A API *Websocket* do serviço API *Gateway* aceita apenas dados em formato de JSON (AWS, 2021). Portanto, a voz captada do remetente é transformada em formato *float array*[24], de tamanho máximo de 1024 *byte*s, e convertida para texto. Com isso, os destinatários recebem os dados do áudio em formato de texto, realizam uma conversão para *float array* e executam o áudio.

[24] Tipo de dado que pode guardar números de ponto flutuante em um array.

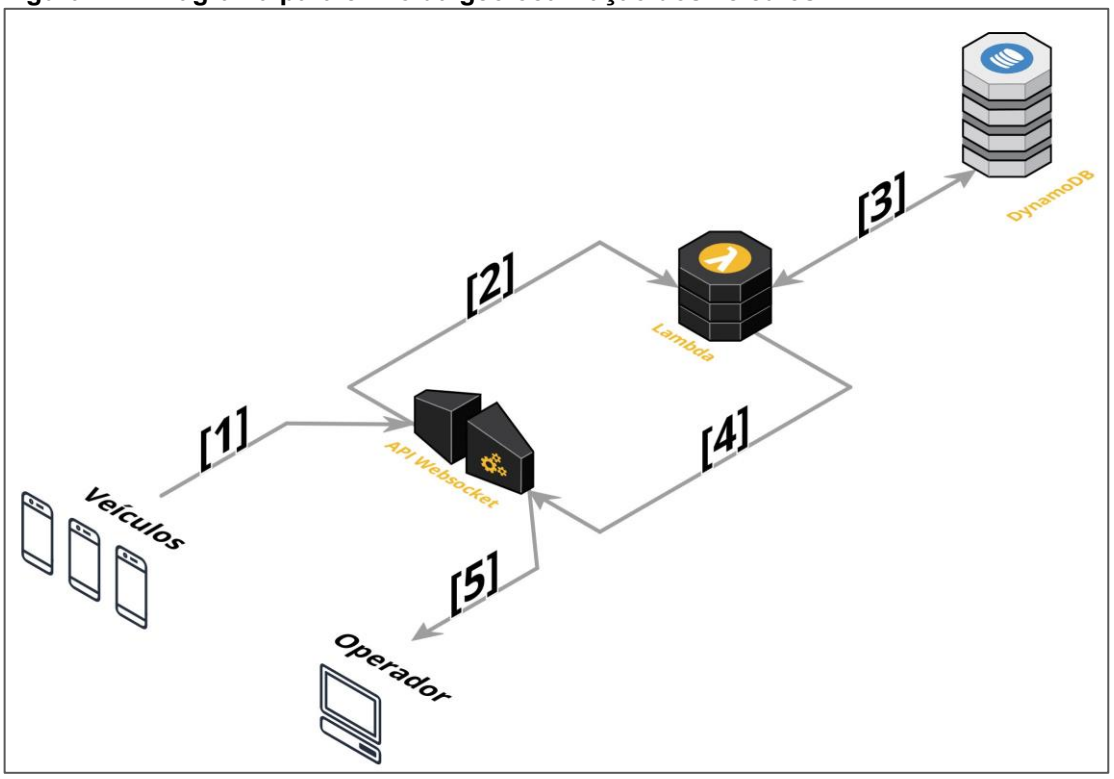
Na operação [4], os dados convertidos são enviados para todos os destinatários, executando em tempo real o áudio em seus dispositivos móveis ou em aplicação *web*.

Na operação [5], o remetente liberou o botão de microfone, finalizando o *streaming* de áudio. Após essa etapa, todos os destinatários são notificados e que o botão de microfone está liberado para poder ser realizado um novo *streaming* de áudio.

3.2.4 Diagrama para envio da geolocalização dos veículos

A Figura 14 apresenta o fluxo para envio da geolocalização dos veículos da frota para que o operador da frota possa visualizar em tempo real em um mapa.

Figura 14 – Diagrama para envio da geolocalização dos veículos



Fonte: Autoria própria.

Na operação [1], a partir dos dispositivos móveis dos funcionários, é possível enviar a geolocalização dos veículos para o *back-end*. Dessa forma, é possível enviar uma mensagem à API *Websocket* com os dados da localização do veículo, como a latitude e a longitude.

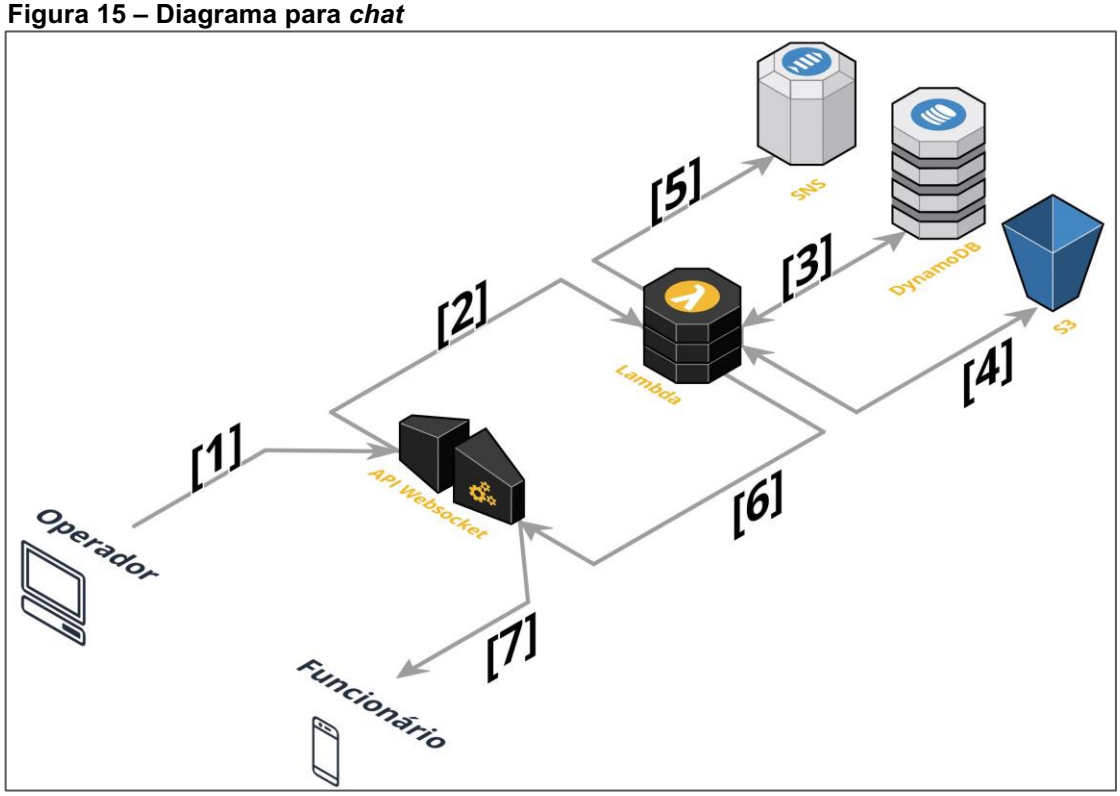
Na operação [2], a API recebe a mensagem e executa a função para processamento dos dados enviados. Após essa etapa, a função armazena os dados da localização no banco de dados e busca as conexões ativas dos gestores de frota (operação [3]). Em seguida, as operações [4] e [5] enviam os dados da localização do veículo ao operador da frota.

Dessa forma, o operador recebe na aplicação *web*, em tempo real, a geolocalização do veículo do funcionário. Para visualizar de uma forma interativa, a localização do veículo é inserida em um mapa.

3.2.5 Diagrama para comunicação em tempo real

A Figura 15 apresenta o fluxo de comunicação entre o operador da frota com um funcionário, por meio de um *chat* privado. Para isso, o diagrama demonstra o envio de uma mensagem de texto ou áudio pelo operador, até o momento em que o funcionário receberá em seu dispositivo móvel.

Figura 15 – Diagrama para *chat*



Fonte: Autoria própria.

Na operação [1], o operador deseja enviar uma mensagem de texto ou áudio para um funcionário. Para isso, é enviada uma mensagem à API *Websocket* contendo os dados que o operador deseja enviar. Após essa etapa, a API realiza a validação da requisição e invoca a função para processar e enviar a mensagem (operação [2]).

Na operação [3], a função armazena a mensagem do operador no banco de dados. Em seguida, caso a mensagem contenha um áudio, a operação [4] é executada. O arquivo de áudio, em formato .wav[25], é armazenado em um serviço de armazenamento de objetos, para que seja possível recuperá-lo posteriormente.

Na operação [5], a função executa o serviço de notificações, para que o funcionário possa receber uma notificação em seu dispositivo móvel que uma mensagem foi enviada para ele. Após essa etapa, a função executa uma operação para que a API *Websocket* envie a mensagem do operador ao funcionário, em tempo real.

3.2.6 Estrututa do banco de dados NoSQL

As principais vantagens de utilizar bancos NoSQL é sua alta flexibilidade, escalabilidade e desempenho (AWS, 2021). Em sistemas que possuem centenas de milhares de requisições de leitura e escrita, os bancos NoSQL se tornam mais apropriados (AWS, 2021).

Entre diversos tipos de bancos de dados NoSQL no mercado, o DynamoDB se encaixa nos requisitos do presente projeto, devido a sua alta escalabilidade com grandes quantidades de dados e usuários (AWS, 2021). Como o sistema possui capacidade para ter vários funcionários e a localização de cada veículo da frota será enviada periodicamente (padrão 30 segundos), o banco de dados realizará diversas requisições de leitura/escrita em uma grande quantidade de dados.

Diferentemente de bancos de dados relacionais, no DynamoDB é definida uma estrutura específica para a tabela com objetivo de tornar as consultas comuns e importantes mais rápidas e de menor custo possível (AWS, 2021).

Uma tabela no DynamoDB é composta por alguns componentes, como chave de partição e chave de classificação. A junção dessas duas chaves é denominada

[25] Formato de arquivo para armazenar dados de áudio.

chave primária composta, sendo necessário informar as duas chaves, obrigatoriamente, para realizar operações no banco de dados. A Tabela 2 representa a estrutura definida para a tabela e a especificação da chave primária composta.

Tabela 2 - Estrutura da tabela do banco de dados

Chave primária		**Atributos**
Chave de partição	**Chave de classificação**	
COMPANY#<companyId>	CONFIG	nome
USER#<username>	CONFIG#<companyId>	usuário, nome completo, email, celular, cargo, status (online e offline) e configurações da conta
	LOCATION#<timestamp>	Latitude, longitude e última atualização (*timestamp)*.
	CONNECTION#<connectionId>	Conexão *Websocket* do usuário
CHAT#<chatId>	CONFIG	Identificador se é um *chat* privado, avatar (opcional), nome do grupo (opcional)
	MEMBER#<username>	Campo que armazena o número de mensagens que não foram visualizadas pelo usuário
	MESSAGE#<timestamp>#<messageId>	Mensagem de texto, identificador de aúdio e a data em que foi enviada a mensagem

Fonte: Autoria própria.

A chave de partição “COMPANY#<companyId>” é utilizada para identificar uma empresa ou organização, em que “companyId” é um identificador único. Para buscar os atributos de uma empresa, basta utilizar a chave de classificação “CONFIG”.

Os usuários são armazenados com chave de partição “USER#username”, em que “username” é um identificador único. Para realizar uma busca pelos atributos de um usuário, basta informar na chave de partição o “USER#” seguido de seu “username” e “CONFIG#” na chave de classificação.

Para consultar todos os funcionários de uma empresa, é necessário definir um *Global Secondary Index* (GSI), em que a chave de partição assume a função da

chave de classificação e vice-versa. Portanto, basta informar “CONFIG#” seguido do identificar único da empresa na chave de partição e “COMPANY#” na de classificação.

Para buscar as últimas localizações vinculadas ao veículo de um funcionário, basta informar a chave de classificação como “LOCATION#”. Dessa forma, basta efetuar uma consulta no banco ordenando de forma descendente e limitando a um único registro.

As conexões ativas de um funcionário podem ser consultadas informando a chave de partição e a chave de classificação como “CONNECTION#”. Com isso, é possível enviar, via *websocket,* atualizações em tempo real para os usuários.

Um *chat* é identificado com chave de partição “CHAT#<chatId>”, em que “chatId” é um identificador único. Para consultar os atributos de um *chat*, é preciso informar seu identificador na chave de partição e “CONFIG” na de classificação.

Os membros de um *chat* em específico podem ser consultados informando a chave de partição com valor “CHAT#”, acompanhado de seu identificador e na de classificação informar “MEMBER#”. Caso deseje realizar o processo inverso, consultando todos os *chats* em que um funcionário fictício pertence, basta utilizar a GSI e definir a chave de partição como “MEMBER#” acompanhada do *username* do funcionário e a chave de classificação como “CHAT#”.

Para consultar todas as mensagens de um *chat*, é necessário informar a chave de partição “CHAT#” acompanhada de seu identificador e definir a chave de classificação como “MESSAGE#<timestamp>#<messageId>”. Em um *chat*, é necessário que as mensagens estejam ordenadas de forma ascendente, para que a última mensagem seja sempre a mais recente. Para isso, é definido o campo “<timestamp>” em sua chave de classificação. Além disso, para evitar duplicações, é definido um identificador único no campo “<messageId>”.

3.3 RESULTADOS

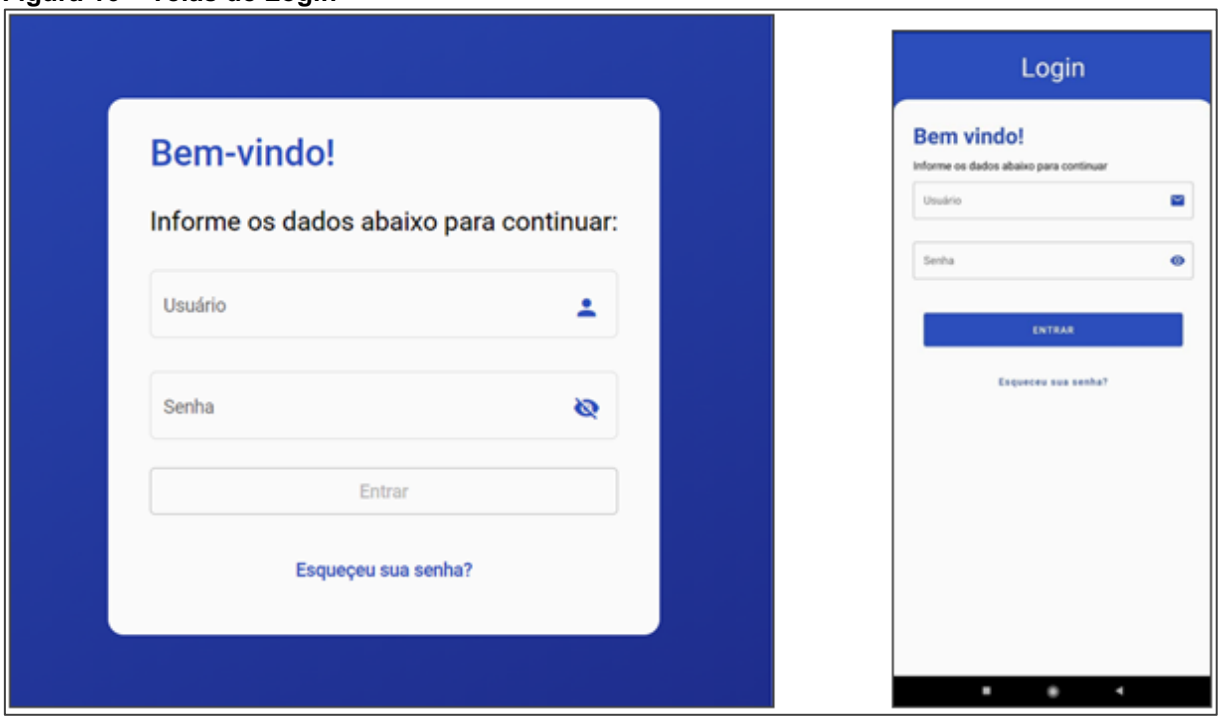
Esta seção apresenta os resultados obtidos no desenvolvimento da aplicação *web* e aplicativo. A Seção 3.3.1 apresenta as telas para autenticação do funcionário no sistema. Além disso, mostra como realizar o gerenciamento de funcionários da empresa.

A Seção 3.3.2 aborda como foi desenvolvido o *chat* em tempo real. A Seção 3.3.3 apresenta o fluxo da funcionalidade PTT e como o *streaming* de áudio é realizado. Por fim, a Seção 3.3.4 mostra como é possível realizar o monitoramento da frota de veículos.

3.3.1 Autenticação e gerenciamento de funcionários.

A autenticação do funcionário é a primeira requisição do usuário ao sistema, sendo ela obrigatória para acessar os demais recursos e funcionalidades. Para isso, é necessário ter uma conta previamente cadastrada por um administrador. A Figura 16 apresenta as telas na aplicação *web* e aplicativo móvel para realizar o *login.*

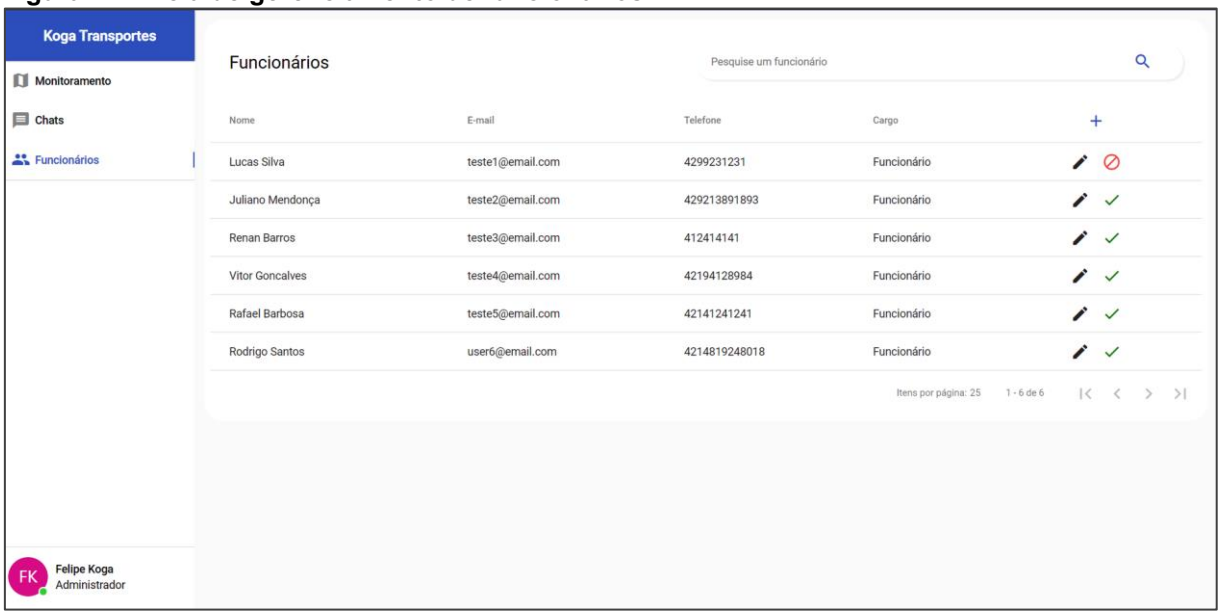
Figura 16 – Telas de Login



Fonte: Autoria própria.

O gerenciamento de funcionários é realizado por usuários que possuem permissão de administrador. A Figura 17 apresenta a tela para realizar esse gerenciamento. Para acessá-la, é necessário estar autenticado e selecionar a opção “Funcionários” no *menu* lateral esquerdo. Após o redirecionamento, o usuário pode visualizar a relação de todos os funcionários que pertencem àquela empresa.

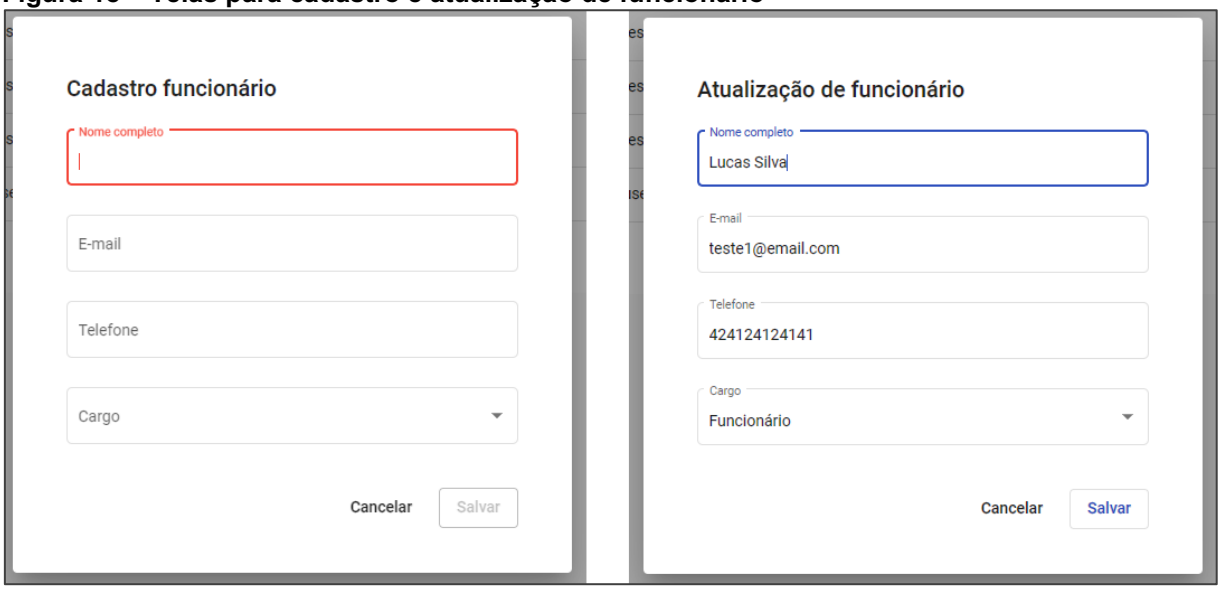
Figura 17 – Tela de gerenciamento de funcionários



Fonte: autoria própria.

A tabela mostrada na Figura 17, exibe o nome, *e-mail*, telefone e cargo dos funcionários. A Figura 18 apresenta as telas para realizar um novo cadastro de um funcionário ou atualizar os dados de um previamente cadastrado. É necessário informar os quatro campos para realizar as operações.

Figura 18 – Telas para cadastro e atualização de funcionário

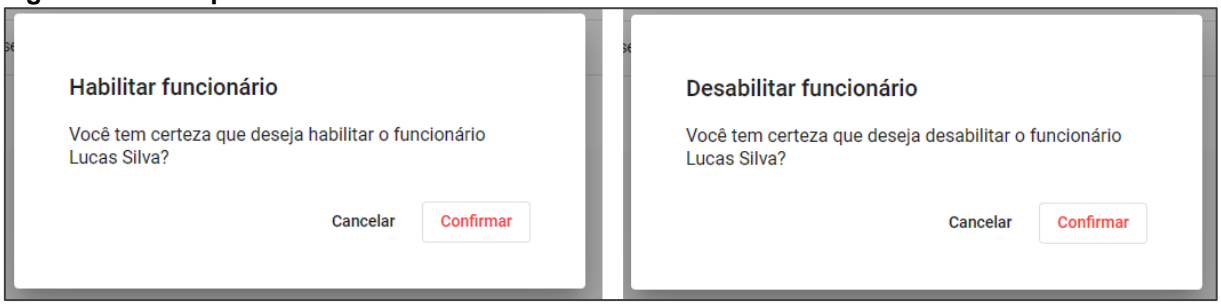


Fonte: Autoria própria.

Na Figura 17, é possível visualizar que o funcionário “Lucas Silva” possui um botão na cor vermelha na sua linha da tabela. Esse botão indica que o funcionário está desabilitado, ou seja, ele não pode acessar os recursos do sistema. Os demais

funcionários estão habilitados e podem realizar acesso normalmente. A Figura 19 apresenta as telas para alterar o *status* do funcionário.

Figura 19 – Tela para habilitar ou desabilitar um funcionário



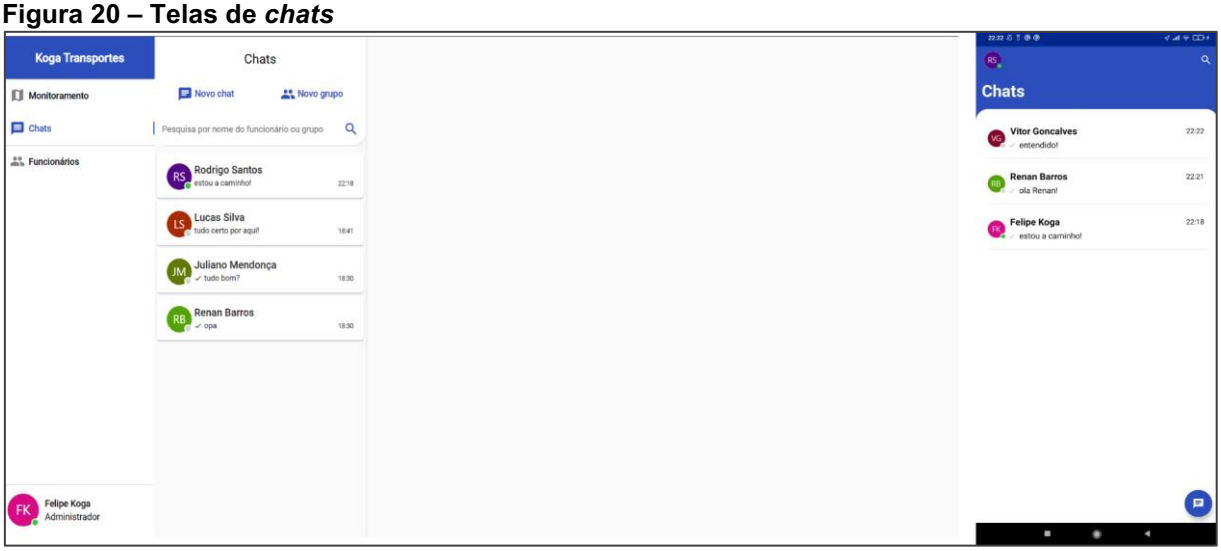
Fonte: Autoria própria.

3.3.2 *Chat* em tempo real

A Figura 20 apresenta as telas de *chats* nas duas plataformas. Na aplicação *web*, ao selecionar no menu a opção "*Chats"*, o funcionário é redirecionado para a tela de *chats*, onde é possível visualizar e iniciar novos *chats* privados ou grupos com múltiplos funcionários.

No aplicativo, ao realizar o *login*, o funcionário será redirecionado para a tela de *chats*, sendo possível visualizar e iniciar um novo *chat* clicando no botão que se encontra no canto inferior direito.

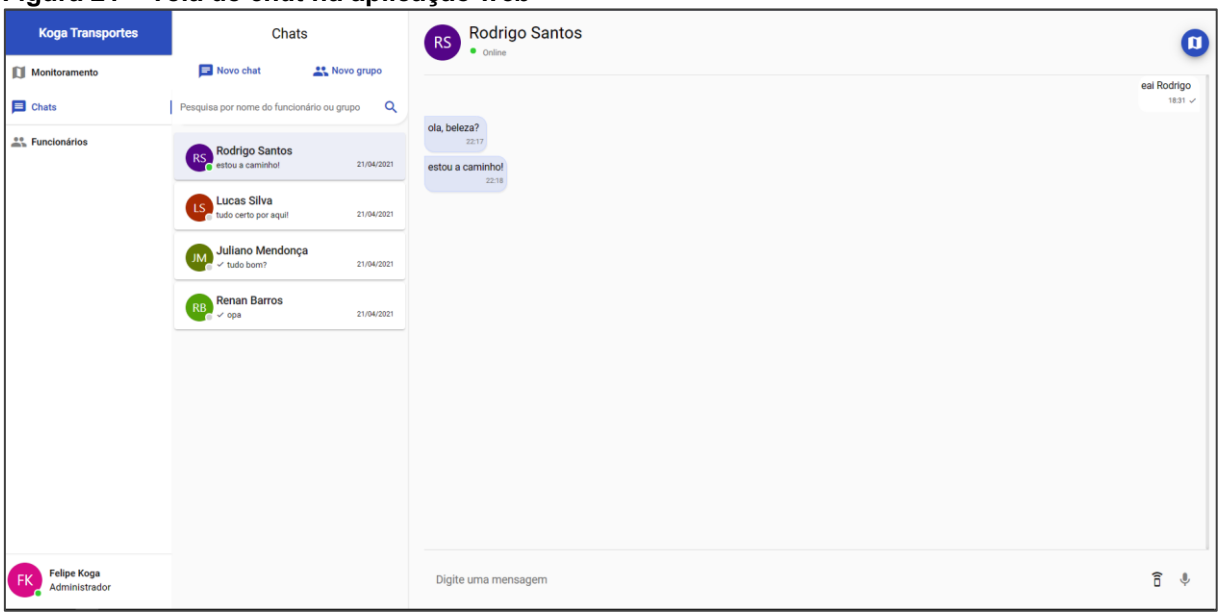
Figura 20 – Telas de *chats*



Fonte: Autoria própria.

Para acessar um *chat* já iniciado, basta clicar sobre o mesmo na lista. A Figura 21 apresenta a tela na aplicação *web* que permite visualizar e enviar mensagens para um funcionário ou um grupo.

Figura 21 – Tela de *chat* na aplicação *web*

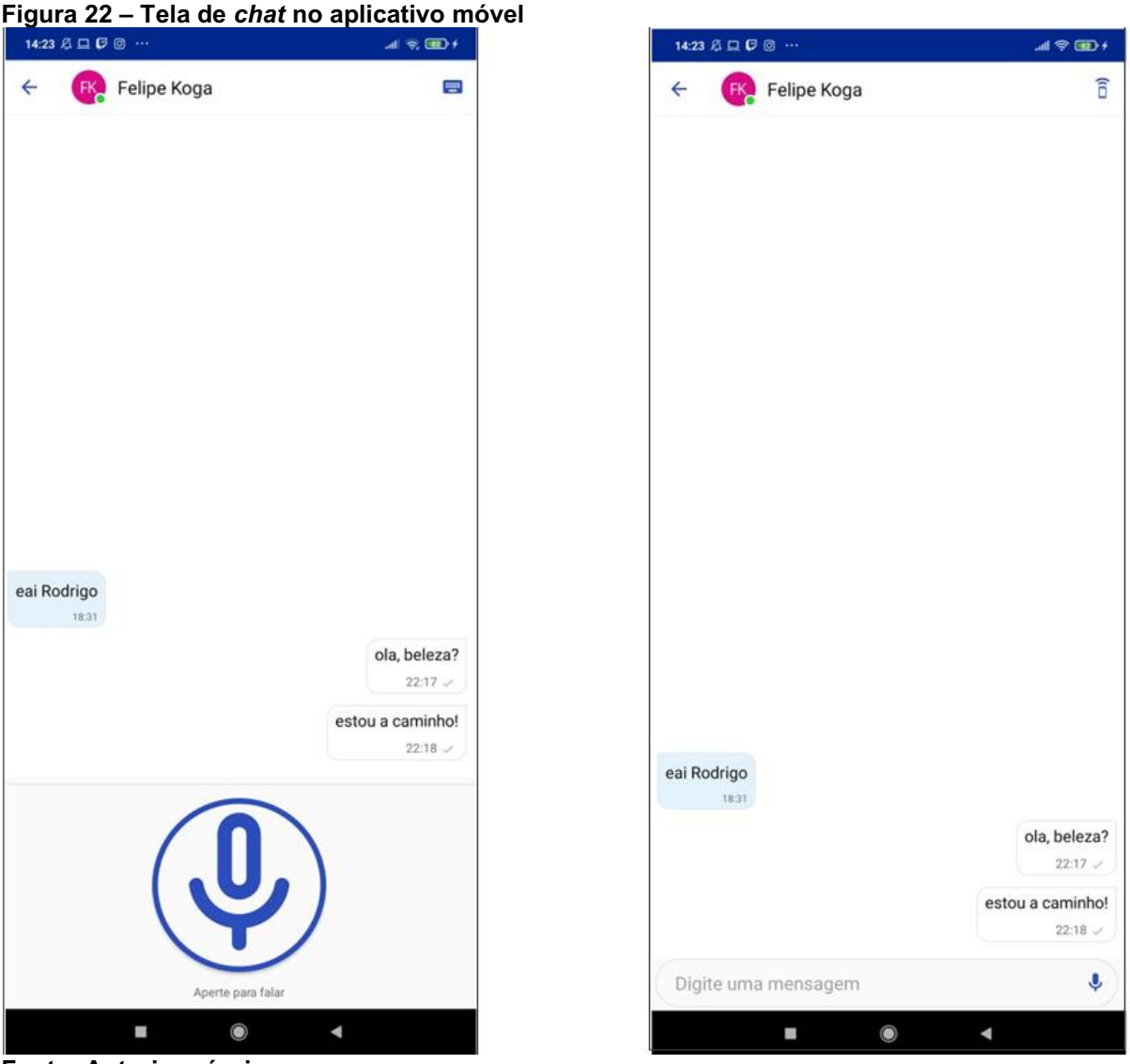


Fonte: Autoria própria.

No canto inferior direito, existe dois botões de ação. O botão esquerdo é responsável por iniciar um *streaming* de áudio, enquanto o da direita é responsável por gravar um áudio para ser enviado como mensagem no *chat*.

Existem algumas diferenças entre a tela da aplicação *web* e o aplicativo móvel. A Figura 22 apresenta a tela de *chat* no aplicativo, sendo possível identificar duas formas de envio de mensagem.

Figura 22 – Tela de *chat* no aplicativo móvel



Fonte: Autoria própria.

Na primeira imagem, a tela exibe um botão relativamente grande, sendo responsável por iniciar o *streaming* de áudio. Seu tamanho é justificável, pois auxilia o motorista do veículo a enviar uma mensagem de voz a outros funcionários, sem perder o foco na corrida.

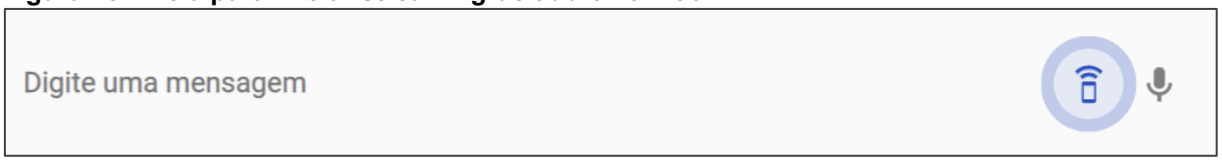
No canto superior direito, existe uma opção para habilitar o teclado do celular. Ao clicar nessa opção, o botão de *streaming* muda para um teclado virtual, para ser possível enviar mensagens de texto. Além disso, existe a opção de gravar um áudio, segurando no botão de microfone, no canto inferior direito.

3.3.3 *Streaming* de áudio

Para iniciar o *streaming* de áudio na aplicação *web*, é necessário acessar a página do *chat* com o funcionário ou o grupo e, posteriormente, segurar o botão esquerdo que se encontra no canto inferior direito da tela.

Nesse momento, uma animação inicia-se ao redor do botão, identificando que a voz do usuário está sendo captada e enviada para os participantes do *chat*. A Figura 23 apresenta a tela para iniciar o *streaming*.

Figura 23 – Tela para iniciar *streaming* de áudio na *web*



Fonte: Autoria própria.

Para iniciar o *streaming*, um método representado na Figura 24 é executado. Esse método recebe três parâmetros:

1) *chatId*: identificador único de um *chat;*
2) *user*: usuário autenticado que está iniciando o *streaming*;
3) *receivers*: funcionários que receberão o áudio do *streaming*. Pode ser apenas um destinatário, caso seja um *chat* privado, ou múltiplos destinatários, no caso de um grupo.

Figura 24 – Código para iniciar *streaming* de áudio na *web*

```
1  public async start(chatId: string, user: User, receivers: string[]) {
2    this.recording$.next(true);
3    this.setupVariables({ user, chatId, receivers });
4    const payload = {
5      user: this.user,
6      receivers,
7      chatId,
8    };
9    this.webSocketService.sendMessage({
10     action: Actions.PUSH_TO_TALK,
11     body: {
12       ...payload,
13       type: Actions.START_PUSH_TO_TALK,
14     },
15   });
16   this.setupStreaming(payload);
17 }
```

Fonte: Autoria própria.

A linha 9 envia uma mensagem, via *websocket*, sinalizando que um funcionário iniciou o *streaming*. Dessa forma, nenhum outro funcionário relacionado àquele *chat* pode iniciar outro *streaming* de áudio.

A linha 16 executa um método denominado *setupStreaming*. Esse método é responsável por configurar os recursos necessários para efetuar o *streaming* de áudio. A Figura 25 apresenta o código de configuração.

Figura 25 – Código para configurar o *streaming* na *web*

```
1  private setupStreaming = async (payload) ⇒ {
2    this.context = new AudioContext({
3      sampleRate: 8000,
4    });
5    this.processor = this.context.createScriptProcessor(1024, 1, 1);
6    const stream = await navigator.mediaDevices.getUserMedia({
7      audio: true,
8    });
9    this.source = this.context.createMediaStreamSource(stream);
10   this.source.connect(this.processor);
11   this.processor.addEventListener(
12     "audioprocess",
13     this.audioProcessingListener
14   );
15   this.processor.connect(this.context.destination);
16   this.audioProcessingListener = (e) ⇒ {
17     this.webSocketService.sendMessage({
18       action: Actions.PUSH_TO_TALK,
19       body: {
20         ...payload,
21         inputData: e.inputBuffer.getChannelData(0).toString(),
22       },
23     });
24   };
25 };
```

Fonte: Autoria própria.

Na linha 2, é inicializado um objeto do tipo *AudioContext*[26], que possui a função de captar a voz do funcionário. Em seguida, é executado o método *createScriptProcessor,* para criação de um processador de áudio. Esse método aceita três parâmetros (MDN, 2021):

1) *bufferSize*: o tamanho do *buffer*[27];

[26] https://developer.mozilla.org/en-US/docs/Web/API/AudioContext
[27] Região da memória física que armazena os dados temporariamente para serem transferidos a outro local.

2) *numberOfInputChannels*: número que especifica a quantidade de canais para entrada de dados;

3) *numberOfOutputChannels*: número que especifica a quantidade de canais de saída de dados.

Na linha 6, é executado o método *getUserMedia(),* que possui a função de solicitar a permissão do usuário para gravação de áudio utilizando seu dispositivo (MDN, 2021). Após conceder a permissão, o método retorna um objeto do tipo *MediaStream*, responsável pelo *stream* dos dados de áudio.

Na linha 9, o método *createMediaStreamResource* é executado. Esse método recebe como parâmetro o *stream* de áudio e retorna um objeto que fornece recursos para manipulação dos dados.

Na linha 16, ao especificar o evento *audioprocess* no método *addEventListener*, é possível receber atualizações do processamento do áudio, permitindo o acesso aos dados em formato *Float32Array,* utilizando o método *getChannelData*. Em seguida, é necessário converter os dados do áudio em *string,* utilizando o método *toString*, para ser possível enviar uma mensagem via *websocket*.

Após enviar a mensagem, uma função *Lambda* será executada para o processamento no *back-end*. A Figura 26 apresenta o código dessa função.

Figura 26 – Código de streaming de áudio no *backend*

```
5  const apigwManagementApi = new ApiGatewayManagementApi({
6      apiVersion: '2018-11-29',
7      endpoint: 'https://87davwn2wl.execute-api.us-east-1.amazonaws.com/dev',
8  });
9
10 const merge = array => array.reduce((a, b) => a.concat(b), []);
11
12 const getAction = type => {
13     if (type === METHODS.START_PUSH_TO_TALK)
14         return RESPONSE.STARTED_PUSH_TO_TALK;
15     if (type === METHODS.STOP_PUSH_TO_TALK)
16         return RESPONSE.STOPPED_PUSH_TO_TALK;
17     return RESPONSE.RECEIVED_PUSH_TO_TALK;
18 };
19
20 exports.handler = async event => {
21     const payload = JSON.parse(event.body);
22     const { type, receivers, ...body } = payload.body;
23     const connectionIdsPromises = [];
24     receivers.forEach(receiver => {
25         connectionIdsPromises.push(fetchConnectionIds(receiver));
26     });
27     const connectionIds = merge(await Promise.all(connectionIdsPromises));
28     const action = getAction(type);
29     const requests = [];
30     connectionIds.forEach(connectionId => {
31         requests.push(
32             apigwManagementApi
33                 .postToConnection({
34                     ConnectionId: connectionId,
35                     Data: JSON.stringify({
36                         action,
37                         body,
38                     }),
39                 })
40                 .promise(),
41         );
42     });
43
44     await Promise.all(requests);
45 };
46
```

Fonte: Autoria própria.

Na linha 21, os dados da mensagem são processados e transformados de JSON para objeto. Em seguida, na linha 25, é realizada uma consulta no banco de dados para buscar as conexões *websocket* ativas dos destinatários. A Figura 27 apresenta o código para realizar essa consulta.

Figura 27 - Código para consulta de conexões no banco de dados

```
1  async function fetchConnectionIds(username) {
2      const { Items } = await dynamoDB
3          .query({
4              TableName: process.env.TABLE,
5              KeyConditionExpression:
6                  'partitionKey = :pk and begins_with(sortKey, :sk)',
7              ExpressionAttributeValues: {
8                  ':pk': `USER#${username}`,
9                  ':sk': 'CONNECTION#',
10             },
11             ProjectionExpression: 'connectionId',
12         })
13         .promise();
14
15     return Items.map(res ⇒ res.connectionId);
16 }
17
```

Fonte: Autoria própria.

Na linha 3, é executado um método para realizar a consulta ao banco de dados. Esse método aceita como parâmetro um objeto que possui as seguintes propriedades:

1) *TableName*: nome da tabela que será realizada a consulta;
2) *KeyConditionExpression*: expressão que especifica os valores dos itens a ser consultados;
3) *ExpressionAttributeValues*: valores que são substituídos na expressão da consulta;
4) *ProjectionExpression*: especifica os atributos que serão consultados

Após realizar a consulta e receber as conexões, é possível enviar os dados de áudio por meio de um método chamado *postToConnection*. Na Figura 26, esse método é executado na linha 44 e recebe como parâmetro um objeto com as seguintes propriedades:

1) *ConnectionId*: identificador único da conexão do destinatário;
2) *Data*: dados remetente e do áudio.

Nesse momento, os destinatários são notificados que um funcionário está realizando o *streaming* de áudio. A Figura 28 apresenta essa notificação, podendo ser visualizada em qualquer tela da aplicação.

Figura 28 – Tela para recebimento de streaming de áudio na aplicação web

Fonte: Autoria própria.

Para que os destinatários possam ouvir efetivamente o áudio, é necessário processar os dados recebidos da mensagem *websocket*. A Figura 29 apresenta o código para realizar esse processamento.

Figura 29 – Código para recebimento de streaming de áudio na web

```
1  public playAudio(audioAsString: string) {
2    const audioAsFloatArray = audioAsString
3      .split(",")
4      .map((value: string) => +value);
5    const buf = this.context.createBuffer(1, 1024, 8000);
6    const player = this.context.createBufferSource();
7    buf.copyToChannel(new Float32Array(audioAsFloatArray), 0);
8    player.buffer = buf;
9    player.connect(this.context.destination);
10   player.start(0);
11 }
```

Fonte: Autoria própria.

Na linha 2, os dados do áudio são convertidos de *string* para *Float32Array*. Em seguida, é realizada a criação de um *buffer* ao executar um método do objeto *context*, instância da classe *AudioContext*. Esse método é denominado *createBuffer* e aceita três parâmetros (MDN, 2021):

1) *numOfChannels*: número de canais do *buffer*;

2) *length*: número que representa o tamanho do *buffer*;

3) *sampleRate*: número que representa a taxa de amostragem dos dados do áudio. Esse parâmetro aceita valores entre 8.000 Hz a 96.000 Hz.

Na linha 6, é realizada a criação do *player*, utilizando o método *createBufferSource*. O *player* possui a função de reproduzir os dados do áudio contidos no *buffer* (MDN, 2021).

Na linha 7, é utilizado o método *copyToChannel,* que possui a função de adicionar os dados de áudio em formato *Float32Array* ao canal especificado do *buffer* (MDN, 2021). Após essa etapa, é necessário atribuir o *buffer* previamente criado ao *buffer* do *player* e conectá-lo ao destino do *context*, para seja possível ouvir o áudio.

No aplicativo móvel, o método para iniciar e receber um *streaming* de áudio é similar, como mostrado na Figura 30. É possível notar que, no momento que o funcionário pressiona o botão, a borda se torna verde, indicando que o *streaming* está ativo.

Figura 30 – Tela para iniciar streaming de áudio no aplicativo

Fonte: Autoria própria.

Nesse momento, uma função é executada no aplicativo. A Figura 31 apresenta o código responsável pelo gerenciamento do *streaming* de áudio.

Figura 31 – Classe para streaming de áudio no aplicativo

```
class PTTStream {
    object Metadata {
        var bufferSize = 1024
        const val sampleRate = 8000
        const val channel = AudioFormat.CHANNEL_IN_MONO
        const val encoding = AudioFormat.ENCODING_PCM_FLOAT
    }

    private var recorder: AudioRecord

    @Volatile
    private var isRecording = false

    init {
        val min = AudioRecord.getMinBufferSize(
            Metadata.sampleRate,
            Metadata.channel,
            Metadata.encoding
        )
        Metadata.bufferSize = min
        recorder = AudioRecord(
            MediaRecorder.AudioSource.MIC, Metadata.sampleRate,
            Metadata.channel, Metadata.encoding, min
        )
    }

    fun start(onReceive: (audioAsString: String) -> Unit) {
        isRecording = false
        recorder.startRecording()
        Thread {
            while (!isRecording) {
                val data = FloatArray(Metadata.bufferSize)
                recorder.read(data, 0, data.size, AudioRecord.READ_BLOCKING)
                onReceive(data.joinToString())
            }
        }.start()
    }

    fun stop() {
        isRecording = true
        recorder.stop()
    }
}
```

Fonte: Autoria própria.

A classe *PTTStream* possui um método *init* que possui a mesma funcionalidade de um método construtor, ou seja, é invocado no momento da criação do objeto. Nesse método, é inicializado o gravador de áudio, sendo uma instância da classe *AudioRecord*[28].

[28] https://developer.android.com/reference/android/media/AudioRecord

Similar a aplicação *web*, a função *start* inicia o gravador de áudio, captando os dados em formato *FloatArray*, devido à configuração previamente adicionada na instanciação do objeto *recorder*, na linha 21.

Na linha 34, ao receber os dados em formato *FloatArray*, um *callback*[29] denominado *onReceive* é executado na função *start*. O código que gerencia a chamada desse *callback* está na Figura 32.

Figura 32 – Código para iniciar streaming de áudio no aplicativo

```
1  private val pttStream = PTTStream()
2
3  fun startPushToTalk() {
4      val receivers = getReceivers()
5      pushToTalkRepository.start(chatId, receivers)
6      pttStream.start { audioAsString ->
7          pushToTalkRepository.send(audioAsString)
8      }
9  }
10
11 fun stopPushToTalk() {
12     pttStream.stop()
13     pushToTalkRepository.stop()
14     _isRecordingPushToTalk.postValue(false)
15 }
```

Fonte: Autoria própria.

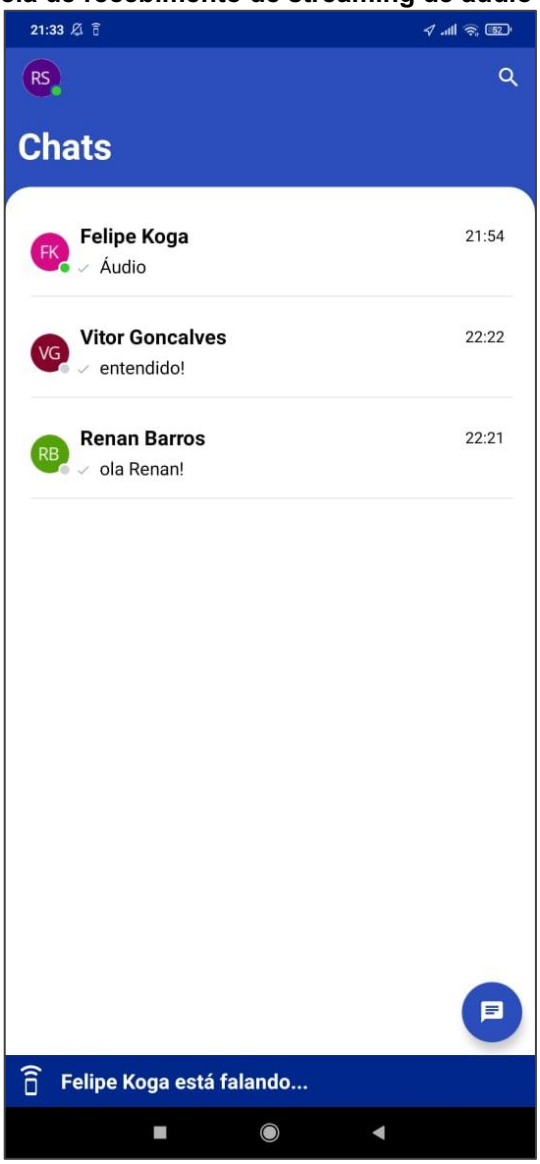
A função *startPushToTalk* é executada no momento em que o funcionário segura no botão na interface do aplicativo. Na linha 5, é enviada uma mensagem *websocket* para os destinatários, indicando que o funcionário está iniciando o *streaming.*

A linha 7 representa o código da implementação do *callback onReceive*, que possui a função de enviar os dados do áudio aos destinatários por *websocket.* No momento em que o funcionário deixa de pressionar o botão na interface, a função *stopPushToTalk* é executada, finalizando o *streaming* de áudio.

Receber o *streaming* de áudio no aplicativo também é similar a aplicação *web*. A Figura 33 apresenta a tela do momento em que um funcionário recebe o *streaming* de áudio.

[29] Função passada como argumento a outra função.

Figura 33 – Tela de recebimento de streaming de áudio no aplicativo



Fonte: Autoria própria.

É possível observar, na parte inferior da tela, que um funcionário está realizando o *streaming*. Para isso ser possível, é necessário definir constantes responsáveis por receber e processar os dados de áudio. A Figura 34 apresenta a definição dessas constantes.

Figura 34 – Definição de constantes para recebimento de streaming de áudio no aplicativo

```
1  private val audioAttributes = AudioAttributes.Builder()
2      .setUsage(AudioAttributes.USAGE_MEDIA)
3      .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
4      .build()
5  private val audioFormat = AudioFormat.Builder()
6      .setSampleRate(8000)
7      .setEncoding(AudioFormat.ENCODING_PCM_FLOAT)
8      .setChannelMask(AudioFormat.CHANNEL_OUT_MONO)
9      .build()
10
11 private val audioTrack =
12     AudioTrack(audioAttributes, audioFormat, 1024, AudioTrack.MODE_STREAM, 0)
```

Fonte: Autoria própria.

A constante *audioAttributes* define os atributos básicos a respeito do áudio a ser recebido pelo *streaming*, como o tipo do conteúdo. A constante *audioFormat* define os atributos do formato do áudio, como taxa de amostragem e a codificação.

A constante *audioTrack* é uma instância da classe *AudioTrack*[30], que recebe os dados em formato *FloatArray* e processa para ser possível ouvir o áudio. O código necessário para converter os dados recebidos do *streaming* de *string* para *FloatArray* é apresentado na Figura 35.

Figura 35 – Código para recebimento de streaming de áudio no aplicativo

```
1  val audioAsFloatList = response.body.inputData.split(',')
2      .map { floatString -> floatString.toFloat() }
3  val audioAsFloatArray = audioAsFloatList.toFloatArray()
4  audioTrack.write(audioAsFloatArray, 0, 1024, AudioTrack.WRITE_BLOCKING)
5  if (!hasStarted) {
6      audioTrack.play()
7      hasStarted = true
8  }
```

Fonte: Autoria própria.

Na linha 4, é executado um método da constante *audioTrack* denominado *write*. Esse método grava os dados do áudio convertido em *FloatArray* para ser possível ouvi-lo no dispositivo móvel, utilizando o método *play*.

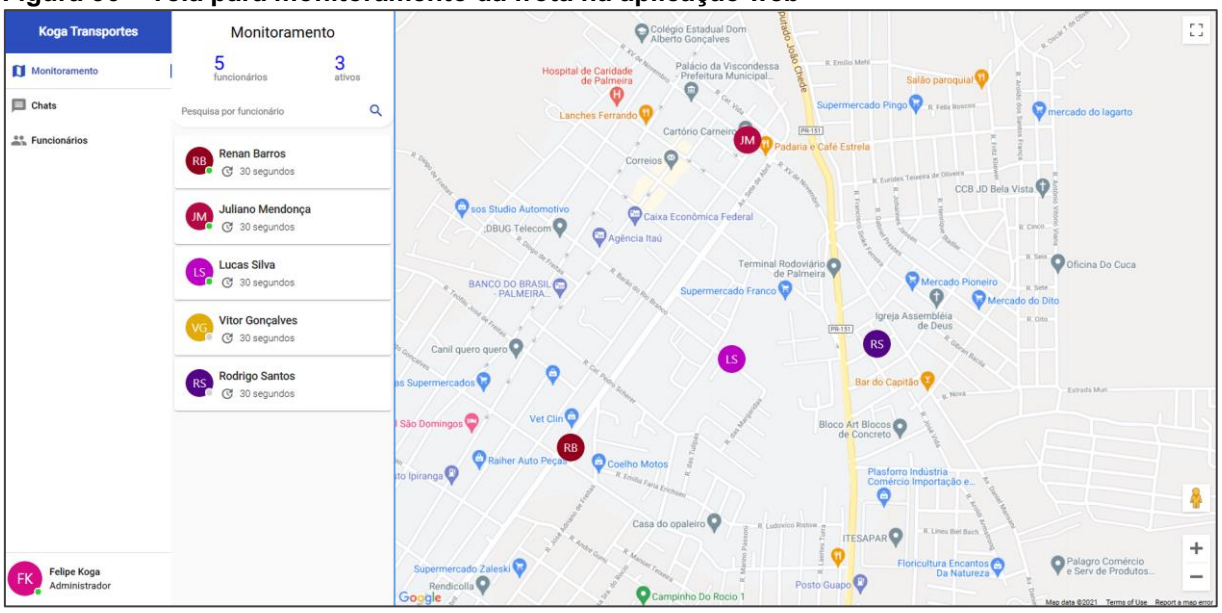
[30] https://developer.android.com/reference/android/media/AudioTrack

3.3.4 Monitoramento da frota

Para visualizar a localização dos veículos da frota no mapa, basta acessar a opção do *menu* “Monitoramento”. Essa é a página padrão da aplicação *web*, ou seja, o operador e administrador serão redirecionados a ela no momento da sua autenticação.

A Figura 36 apresenta a tela para o monitoramento da frota. É possível observar que existe uma coluna ao lado do mapa que informa quantos funcionários a frota possui e quantos funcionários estão com seus veículos ativos.

Figura 36 – Tela para monitoramento da frota na aplicação web

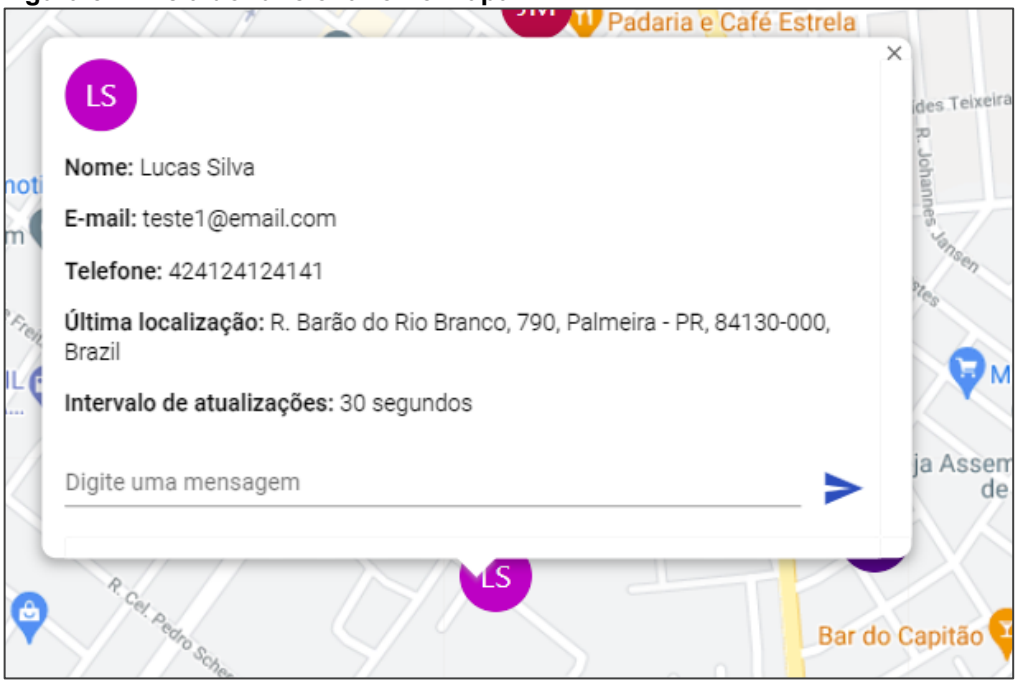


Fonte: Autoria própria.

É possível observar, que existem 5 funcionários ao todo na frota e 3 deles estão ativos (*online)*. No mapa, é possível identificar a localização dos veículos da frota a partir do avatar dos funcionários.

Para obter mais informações a respeito da localização de um veículo em específico, é necessário clicar sobre o avatar do funcionário no mapa. A Figura 37 apresenta a tela que exibe mais informações sobre o funcionário.

Figura 37 – Tela do funcionário no mapa



Fonte: Autoria própria.

É possível observar os dados do funcionário, como nome, *e-mail* e telefone, e a última localização detalhada do veículo. Além disso, para evitar que o operador ou administrador da frota saia da tela de monitoramento para enviar uma mensagem de texto ao funcionário, é possível enviar uma mensagem pelo mapa.

Para exibir o mapa na aplicação *web*, foi adicionado um componente do Angular chamado *AgmMap*[31], que renderiza o *Google Maps*. Além disso, para renderizar a tela que exibe as informações do funcionário e a última localização do veículo, foi utilizado o componente *AgmOverlay* e *AgmInfoWindow*.

Para exibir os marcadores no mapa, é executada uma função no momento em que a página é iniciada, ou seja, no método *ngOnInit*. A Figura 38 apresenta o código para adicionar informações na tela de monitoramento.

[31] https://angular-maps.com/api-docs/agm-core/components/agmmap#info

Figura 38 – Código para adicionar informações na tela de monitoramento

```
1  ngOnInit() {
2    this.getCurrentLocation();
3    this.usersService.fetch();
4    this.usersService.usersState$.subscribe((state) => {
5      this.users = state.users
6        .filter((user) => this.isEmployee(user))
7        .sort(this.orderByOnline);
8      this.buildMarkers();
9    });
10
11   this.webSocketService.messages.subscribe(({ action, body }) => {
12     if (
13       action === Actions.USER_CONNECTED ||
14       action === Actions.USER_DISCONNECTED ||
15       action === Actions.USER_NEW_LOCATION
16     ) {
17       this.usersService.replaceUser(body);
18     }
19   });
20 }
```

Fonte: Autoria própria.

Na linha 3, é requisitada a lista dos funcionários da empresa ao *back-end*. Na linha 4, existe uma inscrição para verificar mudanças na variável *userState$*, ou seja, no momento em que for adicionado, atualizado ou removido um funcionário, o código será executado novamente.

Em seguida, é realizado a manipulação no *array* de funcionários, filtrando apenas àqueles que não são operadores e administrador. Após isso, o *array* é ordenado para exibir primeiramente os funcionários ativos.

Na linha 8, é realizada a construção dos marcadores no mapa. Para isso, é executada uma função que adiciona os marcadores em outro *array*, denonimado *markers*, que está vinculada ao mapa. A Figura 39 apresenta o código para adicionar os marcadores.

Figura 39 – Código para adicionar os marcadores no mapa

```
1  private buildMarkers() {
2    this.markers = [];
3    this.users.forEach((user) => {
4      if (user.location) {
5        this.markers.push({
6          latitude: +user.location.latitude,
7          longitude: +user.location.longitude,
8          icon: {
9            url: user.avatar ? user.avatar : this.getUserAvatar(user),
10         },
11         user,
12       });
13     }
14   });
15 }
```

Fonte: Autoria própria.

Na linha 11 da Figura 38, a inscrição é responsável por observar novas localizações dos veículos, em tempo real. Dessa forma, sempre que um funcionário ficar ativo, inativo ou a localização do seu veículo mudar, a função *replaceUser*, na linha 17, será executada. Dessa forma, o código da inscrição do *usersState$* será executada novamente, adicionando novos marcadores.

No aplicativo, para enviar a localização do dispositivo móvel, é utilizado um *Service*[32]. A Figura 40 apresenta o código responsável por iniciar esse componente.

Figura 40 – Código para iniciar o service de localização no aplicativo

```
1  private fun startService() {
2      if (isServiceStarted) return
3      isServiceStarted = true
4      setLocationServiceState(this, ServiceState.STARTED)
5      GlobalScope.launch {
6          while (isServiceStarted) {
7              launch(Dispatchers.IO) {
8                  sendLocation()
9              }
10             delay(clientRepository.user().locationUpdate.times(1000).toLong())
11         }
12     }
13 }
```

Fonte: Autoria própria.

[32] Componente que executa operações longas em segundo plano (ANDROID, 2021). https://developer.android.com/guide/components/services

Na linha 8, é possível observar que a função *sendLocation* é executada. Além disso, devido à linha 10, essa função será executada periodicamente, seguindo a preferência do funcionário no envio da localização do dispositivo móvel. Por padrão, a periodicidade é 30 segundos. A Figura 41 apresenta código da função *sendLocation*.

Figura 41 – Código para envio da localização do dispositivo móvel no aplicativo

```
1  private fun sendLocation() {
2      if (!hasPermission() || !isLocationEnabled()) return
3
4      fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
5      fusedLocationClient?.lastLocation?.addOnSuccessListener { location ->
6          if (location ≠ null) {
7              repository.sendLocation(location.latitude, location.longitude)
8          }
9      }
10 }
```

Fonte: Autoria própria.

A linha 2 verifica se o funcionário concedeu permissão para envio da localização do dispositivo móvel e se o mesmo possui habilitado o envio, em suas configurações. Para captar a localização do veículo, é utilizando um recurso nativo do Android chamado *LocationService*[33], que permite recuperar a latitude e longitude do dispositivo móvel.

No momento em que é possível recuperar a localização, a linha 7 é executada. Sua função é enviar os dados para o *back-end*, a partir de uma requisição a API *websocket*.

No *back-end*, uma função *Lambda* é executada. A *Lambda* processa a requisição e executa a função *addLocation*, responsável por adicionar no banco de dados a localização do veículo e enviar as atualizações para os operadores e administradores de frota. A Figura 42 apresenta o código da função *addLocation*.

---

33 https://developers.google.com/android/reference/com/google/android/gms/location/LocationServices

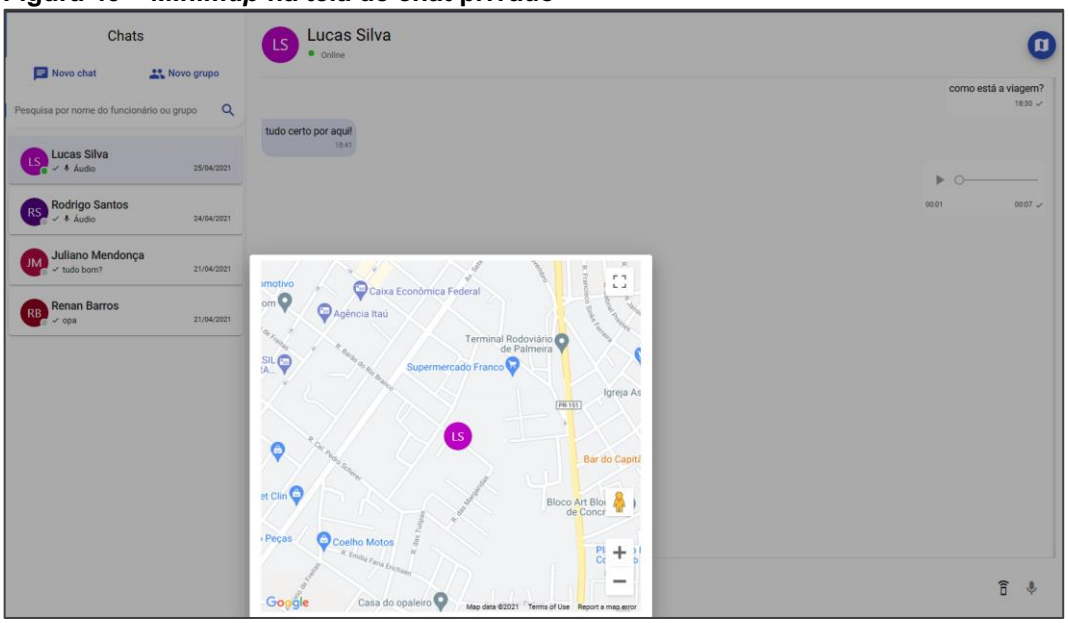
Figura 42 – Código para armazenar a localização no banco de dados

```
async function addLocation({ companyId, username, latitude, longitude }) {
    await Database.location.add(username, {
        latitude,
        longitude,
        lastUpdate: Date.now(),
    });
    const user = await get(username, companyId);
    await postMessage(user, 'USER_NEW_LOCATION');
    return true;
}
```

Fonte: Autoria própria.

Por fim, é possível abrir um minimapa na tela do *chat*, o qual permite visualizar a localização do veículo do funcionário sem precisar navegar na tela de monitoramento. A Figura 43 ilustra a opção, situada no canto superior direito, da tela de um *chat* privado.

Figura 43 – *Minimap* na tela de chat privado



Fonte: Autoria própria.

4 CONSIDERAÇÕES FINAIS

Baseado nos requisitos e arquiteturas definidas, o objetivo principal do presente projeto foi concluído com sucesso. Todas as funcionalidades especificadas foram desenvolvidas, como monitoramento em tempo real da frota de veículos, *chats* entre funcionários e *streaming* de áudio.

A utilização dessas funcionalidades permitem que o operador possua mais controle sobre a frota. Além disso, com a comunicação entre os funcionários, seja ela assíncrona ou por meio de *streaming*, somado ao monitoramento em tempo real da frota, as informações podem ser transmitidas com agilidade e praticidade.

As tecnologias e ferramentas utilizadas fornecem documentações que facilitam no desenvolvimento do código e resoluções de dúvidas. Além disso, são amplamente utilizadas no mercado de trabalho, como o Angular (STATE OF JS, 2021).

A utilização de um provedor de serviços em nuvem colaborou para um rápido desenvolvimento do sistema. O provedor fornece toda a infraestrutura necessária para ser possível focar no desenvolvimento da lógica de negócio.

A principal dificuldade encontrada foi a compatibilidade do *streaming* de áudio entre a aplicação *web* e o aplicativo móvel. Com a utilização de linguagens de programação diferentes para as duas plataformas, foi necessário buscar soluções em diversas documentações para padronização do envio e recebimento do áudio em um único formato.

No entanto, é possível destacar duas limitações nessa funcionalidade. Para que o *streaming* de áudio funcione corretamente e não tenha atrasos no seu conteúdo, é necessário ter uma baixa latência de rede.

Além disso, a cada processamento do áudio captado pelo microfone do dispositivo, a função *Lambda* executa uma vez. Portanto, em um *streaming* que possui a duração de 3 segundos, a função *Lambda* executará, em média, 35 vezes. Isso resulta em diversas invocações da função e consultas ao banco de dados, podendo ocasionar em um custo elevado.

No envio de áudio por meio de *chat*, o formato de arquivo utilizado é .wav. No entanto, esse tipo de formato possui um tamanho elevado (DEMPSEY, 2020), podendo impactar no consumo de banda de rede. Como uma alternativa, é possível

realizar uma conversão para o formato .mp3, que possui um tamanho menor comparado ao .wav (DEMPSEY, 2020).

Um dos diferenciais do sistema desenvolvido é a sua capacidade de ser utilizado em diversos cenários, pois permite que a solução não fique restrita apenas a serviços de transporte, mas auxilie na logística de serviços que demandam atividades em campo.

4.1 TRABALHOS FUTUROS

Devido à importância do sistema desenvolvido, é possível implementar diversas melhorias e funcionalidades em um trabalho futuro, como visualização da frota de veículos no aplicativo móvel, geração de relatórios de viagens e envio de arquivos e documentos no *chat*.

Além disso, uma funcionalidade possível de ser implementada é o módulo de rotas. O funcionário adiciona sua origem e destino em um mapa no aplicativo móvel e transmite os dados da rota ao operador de frota, em tempo real. Isso possibilita um maior controle ao operador, auxiliando em tomadas de decisões.

Atualmente, a popularidade de aplicativos que possuem *streaming* de áudio vem aumentando expressivamente. Portanto, mostra-se relevante implementar melhorias nessa funcionalidade, como soluções de baixa latência e custo.

Como o aplicativo móvel está apenas disponível para Android, é importante desenvolver o aplicativo para iOS, sistema operacional da *Apple* para dispositivos móveis, com o objetivo de atingir um número maior de usuários. Para isso, pode-se utilizar as linguagens *Swift* ou *Objective-C*.

REFERÊNCIAS

AL-FEDAGHI, S. **Developing Web Applications**. 2011. International Journal of Software Engineering and Its Applications. Computer Engineering Department. Kuwait University.

AMARAL, L. B. **A Qualidade dos Serviços no Transporte sob a Logística aplicada em empresas que trabalham no sistema onshore**. 2015. Congresso Nacional de Excelência em Gestão.

ANDROID. **Meet Android Studio**. Disponível em: https://developer.android.com/studio/intro. Acesso em: 17 jan. 2020.

ANDROID. **Set up for Android Development**. Disponível em: https://source.android.com/setup/. Acesso em: 15 jan. 2020.

ANGULAR UNIVERSITY. **Angular Single Page Applications (SPA): What are the Benefits?**. 2019. Disponível em: https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/. Acesso em: 17 jan. 2020.

ANGULAR. Disponível em: https://angular.io/. Acesso em 23 nov. 2019.

AOE, G. X. **Gestão da Informação no Gerenciamento de Frotas de uma Empresas de Prestação de Serviços**. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Engenharia de Produção) – Universidade Estadual de Maringá, Maringá, 2010. Disponível em: http://www.dep.uem.br/gdct/index.php/dep_tcc/article/view/976. Acesso em: 14 jan. 2020.

AWS. **About WebSocket APIs in API Gateway.** Disponível em: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api-overview.html. Acesso em: 12 abr. 2021.

AWS. **Amazon API Gateway**. Disponível em: https://aws.amazon.com/pt/api-gateway. Acesso em: 14 jun. 2020.

AWS. **Amazon Cognito**. Disponível em: https://aws.amazon.com/pt/cognito/. Acesso em 12 jun. 2020.

AWS. **Amazon DynamoDB**. Disponível em: https://aws.amazon.com/pt/dynamodb/. Acesso em: 11 jun. 2020.

AWS. **Amazon S3**. Disponível em: https://aws.amazon.com/pt/s3/. Acesso em: 12 jun. 2020.

AWS. **API Gateway WebSocket API mapping template reference.** Disponível em: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api-mapping-template-reference.html. Acesso em 16 abr. 2021.

AWS. **AWS Lambda**. Disponível em: https://aws.amazon.com/pt/lambda/. Acesso em: 12 jun. 2020.

AWS. **Cloud Computing with AWS**. Disponível em: https://aws.amazon.com/what-is-aws. Acesso em: 12 fev. 2020.

AWS. **Invoking your backend integration:** $default Route and custom routes**.** Disponível em https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api-routes-integrations.html. Acesso em 15 abr. 2021.

AWS. **Relational (SQL) or NoSQL?**. Disponível em: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SQLtoNoSQL.WhyDynamoDB.html. Acesso em: 16 abr. 2021.

AWS. **What is cloud computing?**. Disponível em: https://aws.amazon.com/what-is-cloud-computing. Acesso em: 12 fev. 2020.

AWS. **What is NoSQL?.** Disponível em: https://aws.amazon.com/nosql/. Acesso em 15 abr. 2021.

AZEVEDO, S, G; FERREIRA, J; LEITÃO, J. **The Role of Logistics Information and Communication Technologies in Promoting Competitive Advantages of the Firm**. 2008. The IUP Journal of Managerial Economics. University of Beira Interior.

BELIZARIO, T *et al*. **Difusão da tecnologia da informação aplicada ao transporte rodoviário de cargas**. 2002. Encontro Nacional de Engenharia de Produção, XXII.

BITRE. **Logistics in Australia:** A Preliminary Analysis. 2001. Disponível em:zhttps://www.bitre.gov.au/publications/2001/wp_049. Acesso em: 26 dez. 2019.

BUTTIGIEG, S; JEVDJENIC, M. **Learning Node. js for Mobile Application Development**. Birmingham: Packt Publishing. 2015.

CACCIAGRANO, D; CORRADINI, F; **On Synchronous and Asynchronous Communication Paradigms.** 2001. Theoretical Computer Science. ICTCS 2001. Lecture Notes in Computer Science, vol 2202. Springer, Berlin, Heidelberg.

CHRISTOFER, M. **Logistics & Supply Chain Management**. Edinburgh: Pearson Education Limited. 2011.

CLEMENTS, P *et al*. **Documenting Software Architectures:** Views and Beyond. Boston: Addison-Wesley Professional, 2010.

CNT. **Transporte é o maior segmento do setor de serviços.** Disponível em: http://cms.cnt.org.br/Imagens%20CNT/PDFs%20CNT/Economia%20em%20foco/ECONOMIA_EM_FOCO_ dez2017.pdf. Acesso em: 26 nov. 2019.

COFFEE, J, R. *et al*. **Vehicle tracking, communication and fleet management system**. U.S. Patent No. 6,611,755. 26 ago. 2003. Washington, USA. Disponível em: https://patents.google.com/patent/WO2001046710A2/en. Acesso em: 14 jun. 2020.

COLAVITE, A. S; KONISHI, F. **A matriz do transporte no Brasil:** uma análise comparativa para a competitividade. Simpósio de Excelência em Gestão e Tecnologia, XII, 2015. Resende.

CRAINIC, G. T; LAPORTE, G. **Fleet Management and Logistics**. New York: Springer Science & Business Media. 2012.

CSMP. **CSMP Supply Chain Management Definitions and Glossary**. Disponível em:https://cscmp.org/CSCMP/Educate/SCM_Definitions_and_Glossary_of_Terms.aspx. Acesso em: 13 jan. 2020.

DEMPSEY, J. **How do MP3 and WAV Files Differ:** Your Questions Asked. 2020. Disponível em: https://www.dawsons.co.uk/blog/how-do-mp3-and-wav-files-differ. Acesso em: 17 mai. 2021.

FERREIRA, R. Para se defender de assaltos, motoristas de Uber criam rede de monitoramento 24h. ACritica. 2018. Disponível em: https://www.acritica.com/channels/manaus/news/motoristas-da-uber-criam-rede-de-monitoramento-para-se-defender-de-assaltos. Acesso em: 25 nov. 2019.

FIELDING, R. T; TAYLOR, R. N. **Principled design of the modern Web Architecture**. 2000. International Conference on Software Engineering, Limerick, Ireland.

FLORENCIO, P. **Apps de Comunicação Instantânea são Adequados para Equipes de Trabalho?**. 2019. Disponível em: http://blog.biptt.com/2019/09/20/aplicativos-equipes-de-trabalho/. Acesso em: 17 mai. 2021.

FLORENCIO, P. **Polícia Federal Certifica o BiPTT**. 2019. Disponível em: http://blog.biptt.com/2019/02/08/policia-federal-certifica-biptt/. Acesso em: 05 mai. 2021.

FREEMAN, Adam. **Pro Angular**. New York: Apress, 2017.

GRADY, J. O. **System Requirements Analysis**. Amsterdam: Elsevier, 2010

GROW WITH GOOGLE. **MyCarsTracks – Android drives success for MyCarsTrack**. 2017. Disponível em: https://www.youtube.com/watch?v=5XUGR9Um9GY. Acesso em: 24 fev. 2020.

GYÕRÖDI, C. *et al*. **A Comparative Study: MongoDB vs. MySQL**. 2015. The 13th International Conference on Engineering of Modern Electric Systems. Oradea.

HEINECK, A. **Um sistema de monitoramento de logística em tempo real utilizando dispositivos móveis**. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Ciência da Computação) – Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Santa Rosa, 2012. Disponível em: https://bibliodigital.unijui.edu.br:8443/xmlui/bitstream/handle/123456789/1308/TCC_Adrieli _Heineck.pdf?sequence=1. Acesso em: 14 jan. 2020.

HOQUE, M. Z. **Basics concepts of GPS and its Applications**. IOSR Journal of Humanities and Social Science. 2016.

IBM. **NoSQL Databases**. 2019. Disponível em: https://www.ibm.com/cloud/learn/nosql-databases. Acesso em: 25 nov. 2019.

ILOS. **Transporte de cargas e a encruzilhada do Brasil para o futuro**. 2017. Disponível em: https://www.ilos.com.br/web/tag/matriz-de-transportes/. Acesso em: 23 fev. 2020.

JADHAV, M; SAWANT, B, R; DESHMUKH, A. **Single Page Application using AngularJS**. 2015. International Journal of Computer Science and Information Technologies. Lokmanya Tilak College of Engineering. Koparkhairane. Navi Mumbai.

KABAKUS, A. T; KARA, R. A. **A performance evaluation of in-memory databases. Journal of King Saud University**. 2016. Computed and Information Sciences. Departament of Computer Engineer. Duzce University. Duzce. Turkey.

KALKAN, B. **The Relation between Use of Information Technologies in Logistics Firms, Customer Satisfaction and Business Performance**. 2018. International Journal of Trade, Economics and Finance.

KALUZA, M; TROSKOT, K; VUKELIC, B. **Comparison of front-end frameworks for web applications development**. Zbornik Veleučilišta u Rijeci, 2018.

KONDRATJEV, J**. Logistics. Transportation and warehouse in supply chain**. 2015. Centria University of Applied Sciences.

KOTLIN. **High-Order Functions and Lambda**. Disponível em: https://kotlinlang.org/docs/reference/lambdas.html. Acesso em: 12 mai. 2020.

KOTLIN. **Using Kotlin for Android Development**. Disponível em: https://kotlinlang.org/docs/reference/android-overview.html. Acesso em: 16 jan. 2020.

KUMAR, A. **React vs. Angular vs. Vue.js: A Complete Comparison Guide**. 2018. DZone. Disponível em: https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu. Acesso em: 20 jan. 2020.

LI, Y; MANOHARAN, S. **A performance comparison of SQL and NoSQL databases**. 2013. Communications, Computer and Signal Processing. Department of Computer Science. University of Auckland. New Zealand.

LIFE360. Disponível em: https://www.life360.com/. Acesso em: 15 jan. 2020.

LIFE360. **Life360 – Localizador Familiar e Celular**. Disponível em: https://play.google.com/store/apps/details?id=com.life360.android.safetymapd&hl=pt_BR. Acesso em: 15 jan. 2020.

LIMA JÚNIOR, O. F. **Qualidade em serviços de transportes:** conceituação e procedimento para diagnóstico. 1995. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 1995.

LORENCIO, P. **Push-to-talk:** Comunicação de Voz Instantânea para Equipes de Trabalho. 2017. Disponível em: http://blog.biptt.com/2017/02/10/tecnologia-push-to-talk/. Acesso em: 03 jul. 2020.

LPU. **Logistics and Supply Chain Management**. Phagware: Excel Boks Privated Limited. 2013.

MADSEN, M; TIP, F; LHOTÁK, O. **Static analysis of event-driven Node.js JavaScript applications**. 2015. International Conference on Object-Oriented Programming, Systems, Languages and Applications. Association for Computing Machiery. New York. USA.

MAHAPATRA, L. **Android Vs. iOS: What's The Most Popular Mobile Operating System In Your Country?**. 2013. International Business Times. Disponível em: https://www.ibtimes.com/android-vs-ios-whats-most-popular-mobile-operating-system-your-country-1464892. Acesso em: 12 jan. 2020.

MASO, A. B. D. **Logística de Cargas:** Desenvolvimento e validação de uma ferramenta para a gestão de controle de cargas a mercados agroindustriais. 2018. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Engenharia de Produção) – Universidade Federal de Campina Grande, Sumé, 2018. Disponível em: http://dspace.sti.ufcg.edu.br:8080/jspui/handle/riufcg/5006. Acesso em: 04 jan. 2020.

MDN. **AudioBuffer.copyToChannel().** 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/AudioBuffer/copyToChannel. Acesso em: 03 mai. 2021.

MDN. **BaseAudioContext.createBuffer().** 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/BaseAudioContext/createBuffer. Acesso em: 03 mai. 2021.

MDN. **BaseAudioContext.createBufferSource().** 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/BaseAudioContext/createBufferSource. Acesso em: 03 mai. 2021.

MDN. **BaseAudioContext.createScriptProcessor().** 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/BaseAudioContext/createScriptProcessor. Acesso em: 03 mai. 2021.

MDN. **MediaDevices.getUserMedia().** 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia. Acesso em: 03 mai. 2021.

MEIRELLES, F. S. **29ª Pesquisa Anual de Administração e Uso de Tecnologia da Informação nas Empresas**. 2018. Disponível em: https://eaesp.fgv.br/sites/eaes. Acesso em: 29 set. 2019.

MEISTER, A; FARHAT, J; TRENTO, N. **Sistema de Monitoração de localização com dispositivo portátil**. 2013. 125 f. Trabalho de Conclusão de Curso (Graduação em Bacharelo em Engenharia Industrial Elétrica) – Univesidade Tecnológica Federal do Paraná, Curitiba, 2013. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2206/1/CT_ENGELN_2013_1_01.pdf. Acesso em: 14 jan. 2020.

MELL, P; GRANCE, T. **The NIST Definition of Cloud Computing**. 2011. Computer Security Division. Department of Commerc. Gaithersburg. Maryland. USA.

MEVADA, D. **The Latest Announcement at Google I/O 2017 Developer Conference**. 2017. Mind Inventory. Disponível em: https://www.mindinventory.com/blog/the-latest-announcement-at-google-io-2017

MONIRUZZAMAN, A. B. M; HOSSAIN, S. A. **NoSQL Database:** New Era of Databases or Big Data Analytics – Classification, Characteristics and Comparison. 2013. International Journal of Database Theory and Application. Departament of Computer Science and Engineering. Daffodil International University. Dhaka.

MORE, K. A; CHANDRAN, M. **Native Vs Hybrid Apps**. 2016. International Journal of Current Trends in Engineering & Research. Bharati Vidyapeeth's Institute of Management Information Technology, University of Mumbai, Maharashtra, India.

NAYAK, A; PORIYA, A; POOJARY, D. **Type of NOSQL Databases and its Comparison with Relational Databases**. 2013. International Journal of Applied Information System. Foundation of Computer Science. New York. USA.

PORTER, M. **Strategy and the Internet**. Harvard Business Review. 2001.

RODRIGUES, P. R. A. **Introdução aos Sistemas de Transporte no Brasil e à Logística Internacional.** 5. ed. São Paulo: Aduaneiras, 2014.

ROSA, A. C. **Gestão do transporte na logística de distribuição física**: uma análise da minimização do custo operacional. 2007. Dissertação (Mestrado) – Universidade de Taubaté, Departamento de Economia, Contabilidade e Administração, 2007.

RUMBAUGH, J; JACOBSON, I; BOOCH, G. **The Unified Modeling Language Reference Manual**. Boston: Addison-Wesley Professional. 2004.

SAGHAEI, H. **Design and Implementation of a Fleet Management System Using Novel GPS/GLONASS Tracker and Web-Based Software**. 2016. Department of Electrical Engineering, Islamic Azad University, Shahrekord.

SEST SENAT. **Gerenciamento de riscos no transporte internacional de cargas**. 2018. Conselho Regional do Rio Grande do Sul. Disponível em: http://cmsintranet.sestsenat.org.br/. Acesso em: 27 nov. 2019.

SHARMA, V; DAVE, M. **SQL and NoSQL Databases**. 2012. International Journal of Advanced Research in Computer Science and Software Engineering. Jagan Nath University, Jaipur.

SHORE, J. **Synchronous vs. asynchronous communication:** The differences. 2016. TechTarget. Disponível em: https://searchapparchitecture.techtarget.com/tip/Synchronous-vs-asynchronous-communication-The-differences. Acesso em: 25 nov. 2019.

SIEGEL, J. G; HARTMAN, S. W; QURESHI, A. **The intranet and extranet**. The CPA Journal. 1998.

SILVA, M. G. **A eficácia da logística:** um estudo de caso no transporte de uma empresa home center. 2019. Trabalho de Conclusão de Curso (Graduação em

Bacharelado em Administração de Empresas) – Uni-ANHANGUERA, Goiânia, 2019. Disponível em: http://repositorio.anhanguera.edu.br:8080/handle/123456789/277. Acesso em: 27 dez. 2019.

SOUZA, D. A. **INFOTRUCK:** Gerenciamento de frota. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Ciência da Computação) – FACVEST, Lages, 2017. Disponível em: https://www.unifacvest.net/assets/uploads/files/arquivos. Acesso em: 16 jan. 2020.

STATE OF JS. **Front-end Frameworks.** Disponível em: https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/. Acesso em: 17 mai. 2021.

STONEBRAKER, M. Stonebraker on NoSQL and enterprises. **Communications of the ACM**. New York. v. 52, n. 8, p. 10-11. 2011.

STOSCHEK, Michael. **Radio and/or telephone communication system for a fleet of vehicles**. U.S. Patent n. 4,124,815, 7 nov. 1978.

SZYMONIK, A. **Logistics and Supply Chain Management**. Edinburgh: Pearson Education Limited. 2012.

TECHOPEDIA. **Web-Based Application**. 2017. Disponível em: https://www.techopedia.com/definition/26002/web-based-application. Acesso em: 18 jan. 2020.

TELEGRAM. Disponível em: https://telegram.org. Acesso em 14 jan. 2020.

TELEGRAM. **Telegram FAQ**. Disponível em: https://telegram.org/faq. Acesso em 14 jan. 2020.

TSENG, Y. Y; VUE, W; TAYLOR, M. **The role of transportation in logistics chain**. 2005. Proceedings of the Eastern Asia Society for Transportation Studies. 5. 1657-1672.

TURBAN, E. **Information Technology for Management**. New Jersey: John Wiley & Sons INC. 2004.

VISUAL STUDIO CODE. **Overview**. Disponível em: https://code.visualstudio.com/docs. Acesso em: 21 jan. 2020.

VIVALDINI, M. PIRES, S, R. SOUZA, F. B. **Improving logistics services through the technology used in fleet management**. Journal of Information Systems and Technology Management. 2012.
WHATSAPP. Disponível em: https://www.whatsapp.com/. Acesso em: 25 nov. 2019.

WHATSAPP. **WhatsApp Business App**. Disponível em: https://www.whatsapp.com/business/. Acesso em: 25 nov. 2019.

W3SCHOOLS. **Bootstrap Badges and Labels**. Disponível em: https://www.w3schools.com/bootstrap/bootstrap_badges_labels.asp. Acesso em: 14 jun. 2020.

YML. **Hybrid vs Native Mobile Apps? The Answer is Clear**. 2019. Disponível em: https://ymedialabs.com/hybrid-vs-native-mobile-apps-the-answer-is-clear. Acesso em: 14 jan. 2020.

ZELLO. **Central Management Console.** Disponível em: https://zello.com/product/central-management-console/. Acesso em: 22 fev. 2020.

ZELLO. Disponível em: https://play.google.com/store/apps/details?id=com.loudtalks&hl=pt_BR. Acesso em: 25 nov. 2019.

ZELLO. **Zello Push-to-Talk (PTT) Mobile App**. Disponível em: https://zello.com/product/push-to-talk-app/. Acesso em: 25 nov. 2019.

APÊNDICE A - Especificação de caso de uso

Tabela 3 - UC001 Realizar *login*

ID: UC001
Nome: Realizar Login
Atores: Funcionário, Operador e Administrador
Descrição: Permite realizar a autenticação do Ator no sistema
Pré-condições: O sistema deve estar conectado à Internet
Pós-condições: O sistema redireciona para a página principal
Fluxo Principal
1. Sistema solicita o preenchimento dos campos “usuário” e “senha” 2. Ator informa os campos 3. Ator pressiona o botão *Entrar* 4. Sistema valida os campos informados 5. Sistema libera o acesso
Fluxo alternativo
4ª. Usuário e/ou senha incorretos 1. O sistema exibe uma mensagem de erro e adiciona foco no campo “usuário” 2. Retorna ao passo 1 do fluxo principal

Fonte: Autoria própria.

Tabela 4 - UC002 Visualizar *chats*

ID: UC002
Nome: Visualizar *chats*
Atores: Funcionário, Operador e Administrador
Descrição: Permite visualizar a lista de *chats* que o Ator participa
Pré-condições: O caso de uso UC001 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator seleciona a opção “Chats” no menu 2. Sistema exibe a lista de *chats* do Ator
Fluxo alternativo
Não há

Fonte: Autoria própria.

Tabela 5 - UC003 Iniciar *chat*

ID: UC003
Nome: Inicar *chat*
Atores: Funcionário, Operador e Administrador
Descrição: Permite enviar e receber mensagens dos funcionários da empresa
Pré-condições: O caso de uso UC002 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator seleciona um funcionário ou grupo da lista 2. Sistema inicia o *chat* 3. Ator envia uma mensagem. 4. Sistema valida e adiciona a mensagem ao *chat* 5. Sistema gera uma notificação para o(s) destinatário(s) da mensagem
Fluxo alternativo
1a Ator não seleciona um funcionário da lista e recebe uma mensagem 1. Sistema gera uma notificação para o Ator com a mensagem e o remetente 2. Ator clica na notificação 3. Sistema inicia o *chat*

Fonte: Autoria própria.

Tabela 6 - UC004 Iniciar *streaming* de áudio

ID: UC004
Nome: Inicar *streaming* de áudio
Atores: Funcionário, Operador e Administrador
Descrição: Permite iniciar *streaming* de áudio para os destinatários, utilizando a tecnologia *Push to talk* (PPT)
Pré-condições: O caso de uso UC003 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator pressiona o botão de microfone 2. Ator começa a falar 3. Sistema capta e executa instantaneamente a voz do Ator no dispositivo móvel do destinatário 4. Ator libera o botão de microfone. 5. Sistema encerra o *streaming* de áudio.
Fluxo alternativo
Não há

Fonte: Autoria própria.

Tabela 7 - UC005 Receber *streaming* de áudio

ID: UC005
Nome: Receber *streaming* de áudio
Atores: Funcionário, Operador e Administrador
Descrição: Permite receber *streaming* de um remetente que utilizou o *Push to talk* (PTT)
Pré-condições: O caso de uso UC003 foi executado
Pós-condições: Não há
Fluxo Principal
1. Sistema capta a voz do remetente e executa no dispositivo móvel do Ator 2. Sistema encerra o *streaming* de áudio
Fluxo alternativo
Não há

Fonte: Autoria própria.

Tabela 8 - UC006 Visualizar frota de veículos

ID: UC006
Nome: Visualizar frota de veículos
Atores: Operador e Administrador
Descrição: Permite visualizar em um mapa a frota de veículos da empresa
Pré-condições: O caso de uso UC001 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator seleciona a opção “Mapa” no menu 2. Sistema exibe os veículos com a imagem de perfil dos motoristas no mapa
Fluxo alternativo
Não há

Fonte: Autoria própria.

Tabela 9 - UC007 Gerenciar grupo

ID: UC007
Nome: Gerenciar grupo
Atores: Funcionário, Operador e Administrador
Descrição: Permite criar e excluir grupos com funcionários
Pré-condições: O caso de uso UC002 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator seleciona a opção “Novo grupo” no menu 2. Sistema exibe a lista de funcionários da empresa 3. Ator seleciona os funcionários que deseja inserir no grupo 4. Ator clica no botão “Criar grupo” 5. Sistema valida os funcionários selecionados 6. Sistema cria o grupo 7. Sistema redireciona para o *chat* do grupo
Fluxo alternativo
Não há

Fonte: Autoria própria

Tabela 10 - UC008 Gerenciar funcionários

ID: UC008
Nome: Gerenciar funcionários
Atores: Administrador
Descrição: Permite cadastrar, atualizar, habilitar e desabilitar os usuários do sistema
Pré-condições: O caso de uso UC001 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator seleciona a opção “Funcionários” no menu 2. Sistema exibe a lista de usuários da empresa 3. Ator clica no botão “Novo usuário” 4. Sistema exibe os campos obrigatórios: “nome”, “e-mail”, “telefone” e “cargo” 5. Ator preenche os campos 6. Ator clica no botão “Cadastrar” 7. Sistema valida os dados 8. Sistema gera uma senha aleatória 9. Sistema envia um e-mail para o usuário com suas credenciais
Fluxo alternativo
Não há

Fonte: Autoria própria

Tabela 11 - UC009 Gerencial perfil

ID: UC009
Nome: Gerenciar perfil
Atores: Funcionário, Operador e Administrador
Descrição: Permite modificar os dados do perfil
Pré-condições: O caso de uso UC001 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator clica no item “Meu perfil” no menu 2. Sistema redireciona para o perfil do Ator 3. Sistema exibe os dados e configurações personalizadas do Ator 4. Ator preenche o campo “nome de exibição” 5. Ator clica no botão “Editar” 6. Sistema valida o campo informado 7. Sistema realiza a edição
Fluxo alternativo
4a Ator clica no botão para inserir uma foto de perfil 1. Sistema solicita o *upload* de uma imagem 2. Ator aperta no botão “Browse” 3. Sistema abre as imagens do dispositivo do Ator 4. Ator seleciona uma imagem 5. Ator clica no botão “Confirmar” 6. Sistema valida a imagem 7. Sistema insere a imagem no perfil 4b Ator clica no botão “Nova senha” 1. Sistema solicita os campos obrigatórios: “senha atual”, “nova senha” e “confirmar nova senha” 2. Ator preenche os campos 3. Sistema valida os campos 4. Sistema realiza a edição

Fonte: Autoria própria

Tabela 12 - UC010 Realizar *logout*

ID: UC010
Nome: Realizar *logout*
Atores: Funcionário, Operador e Administrador
Descrição: Permite sair do sistema, impossibilitando o acesso aos recursos do sistema
Pré-condições: O caso de uso UC009 foi executado
Pós-condições: Não há
Fluxo Principal
1. Ator acessa a tela de perfil 2. Ator seleciona a opção “Sair” 3. Sistema encerra o acesso ao sistema 4. Sistema redireciona para o Login
Fluxo alternativo
Não há

Fonte: Autoria própria