CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RIVALDO SOARES DO NASCIMENTO

DESENVOLVIMENTO DE UM SOFTWARE PARA GESTÃO DE RESERVA DE VEÍCULOS

PALMAS – TO
2023

Rivaldo Soares do Nascimento

DESENVOLVIMENTO DE UM SOFTWARE PARA GESTÃO DE RESERVA DE VEÍCULOS

Projeto Tecnológico II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Me Jackson Gomes de Souza.

Palmas – TO

2023

Rivaldo Soares do Nascimento

DESENVOLVIMENTO DE UM SOFTWARE PARA GESTÃO DE RESERVA DE VEÍCULOS

Projeto Tecnológico II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Me Jackson Gomes de Souza.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Me. Jackson Gomes de Souza

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Fernanda Pereira Gomes

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Robson Ferreira Gomes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2023

Dedico este trabalho a todos aqueles que contribuíram diretamente ou indiretamente para a conclusão desta formação. Suas palavras de encorajamento, recursos, apoio e amizade foram fundamentais para a minha jornada acadêmica.

Em especial, gostaria de expressar minha gratidão aos meus amigos e colegas de classe que estiveram ao meu lado durante esses anos. Suas discussões, *insights* e colaboração enriqueceram minha experiência acadêmica, e sou grato por cada momento compartilhado.

Minha mãe, pessoa que me impulsiona a buscar pelo constante conhecimento e aperfeiçoamento, merece um agradecimento especial. Sua dedicação, amor incondicional e incentivo constante foram os pilares que me levaram a alcançar meus objetivos. Sou verdadeiramente abençoado por tê-la como minha mãe.

Há também pessoas que fizeram parte da minha jornada, mas cujos caminhos se separaram dos meus. A vida é feita de ciclos e nem todos duram ou terminam como desejávamos, mas reconheço a contribuição que essas pessoas trouxeram para a minha vida e para o meu crescimento pessoal. Apesar de não mencionar nomes específicos, carrego comigo as lembranças dos momentos compartilhados e agradeço pelas lições aprendidas.

Neste momento, quero reservar um espaço especial para honrar a memória de um amigo muito querido, cuja partida prematura deixou um vazio em nossos corações. Fernando, você foi um companheiro leal, generoso e sempre presente em minha vida. Suas memórias e nosso tempo juntos continuarão a ser uma fonte de inspiração e motivação para mim. Sinto sua falta todos os dias e serei eternamente grato por ter tido a honra de chamá-lo de amigo.

Este trabalho não seria possível sem o apoio, incentivo e colaboração de todos aqueles que mencionei, incluindo aqueles que já partiram, e de muitos outros que, de alguma forma, cruzaram meu caminho. Expresso aqui minha profunda gratidão a todos vocês.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a Deus e aos inspiradores professores que guiaram meu caminho ao longo desses anos e, em especial, àqueles que ofereceram o Norte para o desenvolvimento deste trabalho. Vocês foram mais do que transmissores de conhecimento; foram mentores e guias, incansáveis em sua dedicação em ajudar-nos a despertar todo o nosso potencial. Suas aulas envolventes, divertidas, orientações valiosas e o incentivo constante foram fundamentais para o meu crescimento acadêmico e pessoal.

Agradeço por compartilharem seu conhecimento, experiência e paixão pela disciplina que lecionam. Vocês nos desafiaram a pensar de forma crítica, a explorar novas ideias e a superar obstáculos por meio da tecnologia. Seus conselhos e feedback construtivos moldaram minha abordagem de estudo e me ajudaram a desenvolver habilidades essenciais para minha futura carreira.

Este trabalho não seria possível sem o apoio, incentivo e colaboração de todos aqueles que mencionei anteriormente, incluindo meus amigos, colegas de classe, colegas de trabalho, minha mãe e aqueles que já partiram, e de muitos outros que, de alguma forma, cruzaram meu caminho. Expresso aqui minha mais sincera gratidão a todos vocês por terem desempenhado um papel fundamental em minha jornada acadêmica e pessoal.

“A tecnologia é só uma ferramenta. No que se refere a motivar as crianças e conseguir que trabalhem juntas, um professor é o recurso mais importante”

Bill Gates.

RESUMO

NASCIMENTO, Rivaldo Soares do. **Desenvolvimento De Um Software Para Gestão De Reserva De Veículos**. 2023. 113 f Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2023.

A gestão eficiente de uma frota corporativa de veículos apresenta desafios relacionados à reserva de veículos, disponibilidade, ocupação e manutenção. Atualmente, muitas organizações lidam com esses processos de forma manual, o que pode ser demorado, propenso a erros e ineficiente. Portanto, existe a necessidade de um sistema automatizado que melhore a gestão de reservas de veículos em frotas corporativas. Apesar da existência de algumas soluções no mercado voltadas para a gestão de frotas de veículos, a maioria delas está direcionada para empresas de logística, indústrias e multinacionais, deixando uma lacuna em relação a um software mobile adaptável para as diversas realidades de negócio que possuem frota veicular interna. Neste sentido, há a falta de uma ferramenta que atenda às necessidades de gestão de reservas de veículos para diferentes tipos de organizações. Este trabalho propõe o desenvolvimento de um software mobile como uma ferramenta que visa auxiliar a gestão de reservas de veículos em frotas corporativas. O software visa funcionalidades que permitem o controle das reservas e a visualização da disponibilidade dos veículos. Além disso, busca-se oferecer uma solução alternativa para otimizar as atividades de gestores de frotas, reduzindo custos operacionais e trazendo maior precisão no registro de informações. A metodologia utilizada para o desenvolvimento do software inclui a identificação das funcionalidades essenciais por meio de entrevistas e análise de dados, bem como a utilização do software de prototipagem Figma para visualizar e validar o fluxo das funcionalidades. Um banco de dados online foi modelado e estruturado para suportar as necessidades do sistema. O desenvolvimento do software será realizado utilizando linguagens e frameworks como Dart e Flutter, o que permite sua compilação para diferentes plataformas em trabalhos futuros. A implementação do software, com foco em desenvolvimento mobile, tendo potencial para se tornar um software multiplataforma, busca trazer benefícios como otimização de operações e melhoria na tomada de decisões relacionadas à gestão de frotas corporativas de veículos.

Palavras-chave: gestão de frotas, reserva de veículos, frotas corporativas, banco de dados online, software mobile, desenvolvimento de software.

ABSTRACT

NASCIMENTO, Rivaldo Soares do. **Desenvolvimento De Um Software Para Gestão De Reserva De Veículos**. 2023. 113 f Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2023.

The efficient management of a corporate fleet of vehicles presents challenges related to vehicle reservation, availability, occupancy, and maintenance. Currently, many organizations handle these processes manually, which can be time-consuming, error-prone, and inefficient. Therefore, there is a need for an automated system that improves vehicle reservation management in corporate fleets. Despite the existence of some solutions in the market focused on vehicle fleet management, most of them are targeted at logistics companies, industries, and multinational corporations, leaving a gap for an adaptable mobile software for the diverse business realities that have an internal vehicle fleet. In this sense, there is a lack of a tool that meets the vehicle reservation management needs for different types of organizations. This work proposes the development of a mobile software as a tool aimed at assisting the management of vehicle reservations in corporate fleets. The software aims at functionalities that enable reservation control and visualization of vehicle availability. Additionally, it seeks to provide an alternative solution to optimize fleet manager activities, reducing operational costs, and bringing greater accuracy in information recording. The methodology used for software development includes the identification of essential functionalities through interviews and data analysis, as well as the use of the Figma prototyping software to visualize and validate the flow of functionalities. An online database was modeled and structured to support the system's needs. Software development will be carried out using languages and frameworks such as Dart and Flutter, enabling compilation for different platforms in future work. The software implementation, focusing on mobile development, with the potential to become cross-platform, aims to bring benefits such as operational optimization and improvement in decision-making related to the management of corporate vehicle fleets.

Keywords: fleet management, vehicle reservation, corporate fleets, online database, mobile software, software development.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicação
APIs	Interfaces de Programação de Aplicações
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
GPS	Sistema de Posicionamento Global
HTML	Linguagem de Marcação de Hipertexto
HTML5	Quinta versão da Linguagem de Marcação de Hipertexto
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
ID	Idenrificador Exclusivo
IDs	Identificadores Exclusivos
MG	Minas Gerais
MVC	Model-View-Controller
MVP	Minimum Viable Product
REST	Transferência de Estado Representacional
RENAVAM	Registro Nacional de Veículos Automotores
TI	Tecnologia da Informação
UI	Interface do Usuário
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UX	User Experience
WWW	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

A gestão eficiente de uma frota corporativa de veículos é um desafio constante para organizações e empresas de diversos setores, devido à complexidade que envolve. Essa gestão abarca diversos aspectos, desde assegurar que haja veículos suficientes para atender às demandas dos colaboradores até monitorar a manutenção dos veículos para garantir segurança e eficiência operacional. Nesse contexto, é fundamental abordar processos como a disponibilidade de veículos e sua preparação para uso, incluindo atividades corriqueiras como ações administrativas externas, visita a clientes, encaminhamento veicular para manutenções corretivas, revisões e lavagens. Tais procedimentos envolvem a necessária troca de informações entre os usuários solicitantes e os membros do setor de transportes das organizações.

Um exemplo tangível ocorre no caso de uma reserva de veículo para tarefas externas. O usuário que demanda o veículo deve interagir com o gestor de transportes, fornecendo informações vitais como datas, horários e motoristas designados. O gestor, por sua vez, precisa recorrer às suas anotações, planilhas ou agendas para avaliar a disponibilidade do veículo que atenda à solicitação. Esse processo envolve a confrontação com outras reservas ou manutenções planejadas, e só após essa análise, o gestor pode confirmar a reserva ao solicitante. Esse processo, frequentemente, ocorre de maneira manual. No entanto, um software pode automatizar esses trâmites, otimizando significativamente as atividades do gestor de frotas.

Além da automação proporcionada, esse tipo de software também apresenta como um de seus benefícios a economia de recursos e tempo. Ademais, o uso de um sistema confiável garante a integridade e a precisão das informações, o que é fundamental ao lidar com responsabilidades relacionadas a ocorrências no trânsito, como infrações, colisões e sinistros.

Clemente (2008) afirma que a gestão de frotas de veículos envolve a administração dos automóveis de uma organização, sendo possibilitada por meio de softwares e ferramentas tecnológicas, de modo a viabilizar o controle e o monitoramento das informações pertinentes aos veículos. Estas atividades são entendidas como ações operacionais. Segundo Lima et al (2014), a Tecnologia da Informação (TI) traz notáveis vantagens às operações empresariais, abarcando qualidade, desempenho e disponibilidade de recursos. Essas soluções tecnológicas permitem que as organizações otimizem suas operações e atendam às demandas de maneira eficaz, o que se reflete em resultados ampliados, além do desenvolvimento de produtos, serviços e mercados.

Observando o cenário atual, o site Capterra (2023) lista 148 empresas especializadas em softwares de gestão de frotas, com 10 delas atuando no Brasil. Entre essas, há empresas focadas

em monitoramento veicular, trajetos via Sistema de Posicionamento Global (GPS), métricas de velocidade, consumo de combustível e metas de entregas. Algumas se concentram na manutenção de maquinário industrial. Dessas empresas, apenas 1 oferece sistematização para uso interno de veículos por parte dos funcionários, atendendo a outras empresas como ALD *Automotive*, MRV Engenharia, Kroton, Petrobrás, Raízen, Honda, Hyundai e Unidas (Joycar, 2023).

No entanto, essa análise aponta que os softwares para gestão de frotas empresariais se direcionam predominantemente a empresas industriais, multinacionais e de logística. Nesse contexto, identifica-se uma oportunidade para uma solução informatizada que facilite o controle institucional de solicitações para viagens, manutenções e outras ações rotineiras relacionadas à gestão de veículos. Essa solução precisa ser adaptável a diferentes contextos de negócios, especialmente aqueles que envolvem frota veicular interna.

O desenvolvimento de uma ferramenta capaz de se ajustar às práticas usuais na utilização de veículos por parte dos colaboradores, englobando desde a verificação de disponibilidade até a devolução, revela um nicho com grande potencial. Por meio da tecnologia da informação, essa proposta de desenvolvimento busca beneficiar uma ampla gama de organizações, preservando as atividades cotidianas dos gestores de frotas, abrangendo a gestão de reservas, ocupação veicular e responsabilização por infrações. Tal abordagem não apenas reduz custos operacionais associados, mas também aprimora o registro preciso das informações tratadas, independentemente do porte da organização.

Dessa forma, o desenvolvimento considerou fatores como o fluxo de trabalho, segurança e integridade das informações, bem como o acesso ao sistema. Para otimizar o fluxo, foi essencial identificar as funcionalidades essenciais e, para validá-las, foi crucial conceber um protótipo visual. Para garantir a integridade dos dados, foi implementado um banco de dados online interligado ao sistema. Essa abordagem é crucial para operacionalizar as funções e facilitar a funcionalidade geral do aplicativo. Quanto ao acesso dos usuários, a adoção de uma alternativa mobile, com potencial de escalonamento multiplataforma, permite que essa interação ocorra em dispositivos diversos, além dos tradicionalmente utilizados em ambientes estacionários.

A conjuntura atual oferece ferramentas gratuitas para a criação de protótipos, a exemplo do Figma. Outras ferramentas viabilizam a compilação de códigos para criar aplicações multiplataforma, como o Flutter. Em relação à modelagem de banco de dados, existem opções Open Source com uma comunidade em crescimento, como o Supabase. Vale notar que a

simplicidade de implantação de softwares mobile, potencialmente multiplataforma, se traduz em menos exigências de configuração local nos dispositivos das organizações.

Nesse contexto abrangente, este trabalho busca que resultados obtidos possam ser aplicados em empresas de diversos setores, contribuindo para a otimização de suas operações e aperfeiçoamento de seus resultados. Além disso, procura impulsionar a geração de novas ideias e a apresentação de tecnologias capazes de expandir a aplicação do conhecimento voltado para o desenvolvimento de softwares.

Portanto, o objetivo desta pesquisa baseia-se na compreensão do processo de desenvolvimento de um software mobile como ferramenta na gestão de reservas de veículos, visando abordar as complexidades inerentes a esse desafio. Em contextos mais específicos, a ferramenta projetada proporciona informações detalhadas sobre as reservas, os status das reservas, os status dos veículos, bem como otimiza o cálculo das quantidades de veículos em uso e disponíveis. Além disso, o estudo propõe um protótipo e traz discussão quanto a modelagem, a estruturação do banco de dados e a implementação das funcionalidades subsidiados pelos requisitos identificados após a coleta de informações por meio de questionários e entrevistas.

Durante a pesquisa, ficou evidente que a gestão eficaz de frotas corporativas é uma atividade desafiadora com repercussões significativas. Garantir a disponibilidade de veículos, usá-los eficientemente e manter sua manutenção são fatores críticos. A adoção de um software pode melhorar drasticamente essa gestão, proporcionando acesso a informações em tempo real sobre reservas, disponibilidade e uso de veículos. Isso facilita a tomada de decisões sobre alocação de veículos, planejamento de manutenções e controle de custos.

Além disso, a automação de diversas ações administrativas correlacionadas com a utilização veicular é possível com a utilização da tecnologia da informação, como por exemplo o agendamento de novas reservas, ou a listagem de reservas por usuários. Cita-se que o rastreamento de veículos e a geração de relatórios abrangentes sobre a utilização da frota pode ser abordado em trabalhos futuros paralelamente com o desenvolvimento de outras funcionalidades. Vale dizer que essas automações resultam em economia de tempo e recursos, permitindo que a organização concentre esforços em outras áreas cruciais.

Portanto, o desenvolvimento de um software para a gestão de reservas de veículos em frotas corporativas oferece uma vantagem competitiva, melhorando a eficiência da gestão, reduzindo custos e aprimorando demais processos decisórios. Para aqueles interessados em explorar ou contribuir com o código-fonte deste software, ele está disponível publicamente no GitHub, através do link https://github.com/oirivaldo/linkar. Com isso em mente, este projeto

busca não apenas otimizar as operações empresariais por meio da tecnologia, mas também incentivar a participação da comunidade de desenvolvedores no aprimoramento contínuo do software, ou até mesmo em abordagens similares.

2 REFERENCIAL TEÓRICO

2.1 Frotas Corporativas de Veículos

Toda empresa moderna precisa de estratégias gerenciais montadas, em princípio, pelos gestores dessa organização, já que o seu desenvolvimento requer compreensão não só do funcionamento do setor, mas de controle da eficiência, previsão de mercado futuro, conhecimento técnico, entre outros. Nesse sentido, Barbosa e Brondani (2005) argumentam que o planejamento estratégico é uma ferramenta valiosa da alta gestão, por permitir que as ações de gerenciamento sigam um plano previamente determinado com metas e estratégias claras, diminuindo a possibilidade de que se cometam erros.

Segundo Ackoff (1976, *apud* Santana, 2021) a elaboração do Planejamento Estratégico tem cinco etapas essenciais:

- **Fins**: onde se colocam as metas e os objetivos;
- **Meios**: onde se estabelecem as políticas, os programas, os procedimentos e as práticas que levarão a empresa a alcançar seus objetivos;
- **Recursos**: os meios e quantidades necessários, bem como, a forma como serão gerados e distribuídos;
- **Implantação**: os procedimentos para a tomada de decisão com base em um plano de execução;
- **Controle**: os procedimentos que irão apontar as falhas e os erros do plano de execução, bem como, as formas de prevenção e correção continuadas que devem ser aplicadas.

Desse modo, o autor demonstra que as políticas da empresa devem estar contidas no Planejamento Estratégico, representando o meio que a empresa irá se valer para alcançar os objetivos propostos. Seguindo o mesmo raciocínio, Oliveira (1997) afirma que a utilização das políticas corporativas funciona como balizas ou guias que irão conduzir o gestor nas suas tomadas de decisões. No momento da tomada de decisão, a política servirá de apoio ao administrador, para isso, afirma Leontiades (1982, apud Mundstock 2008), haveria três degraus a serem superados:

- **Formulação de Políticas**: onde se estabelecem o objetivo e a direção geral da organização;
- **Formulação da Estratégia**: onde se desenvolvem as estratégias alternativas que irão ser enxertadas nas políticas;
- **Formulação de Planos**: onde irão ser criados os programas táticos específicos dos setores, bem como, as etapas de cumprimento das estratégias propostas no item anterior.

Um procedimento muito importante para que o gestor defina as diretrizes de sua empresa, suas políticas operacionais e as orientações para o controle e avaliação dos resultados é a elaboração de um planejamento estratégico detalhado. Esse planejamento deve levar em consideração a missão, visão e valores da empresa, além de identificar os objetivos a serem alcançados e as estratégias a serem adotadas para atingir tais metas. Além disso, deve ter meios de correção das falhas apresentadas, por meio de um sistema de controle interno eficiente que permita identificar desvios em relação ao planejado e tomar medidas corretivas para corrigi-los. Essa abordagem ajuda a garantir o alinhamento entre as ações da empresa e os resultados desejados, promovendo a eficiência e o sucesso organizacional (Schmidt, 2002). No mesmo sentido, as ideias de Coase (1937) e Williamson (2002), citados por Silva Filho (2006), defendem que as empresas deveriam descentralizar a execução de atividades que poderiam ser contratadas a baixo custo, focando na especialização de sua atividade principal.

Esses autores propuseram que o gestor analise o mercado e considere a contratação terceirizada de serviços. Essa abordagem visa reduzir custos e permitir que o gestor concentre sua energia institucional na atividade-fim da organização, ou seja, nas atividades principais e estratégicas que são o cerne do negócio. Dessa forma, ao terceirizar serviços secundários, a empresa pode se beneficiar da especialização e expertise de outras empresas, otimizando recursos e maximizando a eficiência operacional. Suas intenções buscavam a propositura de se pensar além do que, à época, seria o conveniente.

> "A necessidade em se ir além de uma análise conveniente, às vezes até adequada, da concepção da firma como uma mera unidade funcional de produção, que é a construção da tecnologia, para se conceber esta como uma estrutura de governança, que é a construção organizacional, e com efeitos de propósitos e econômicos." (Williamson, 2002, p. 602 apud Silva Filho, 2006)

Portanto, a partir dos anos 2000, empresas passam a terceirizar serviços secundários, contratando outras empresas especializadas, de modo que possam se focar em potencializar os

ganhos com sua atividade principal. Assim, inicia-se uma nova demanda: empresas que surgem como solução para minimizar custos de produtos e serviços diretamente ao mercado, intermediando os processos da estrutura hierárquica. Buscava-se agora a eficiência operacional, de modo que a produção das atividades tivesse mais desempenho, com menores custos e menores recursos (Potter, 2004, p. 2). Alguns exemplos de empresas especializadas em serviços secundários são serviços de limpeza, serviços de segurança, serviços de aluguel de impressoras, serviços de terceirização de frotas, dentre outros.

De acordo com a lógica da teoria de Williamson (2002), todas as decisões empresariais deveriam passar por uma análise comparativa de custos totais a fim de identificar o menor custo para a organização, escolhendo entre: ter ou terceirizar. Neste sentido, quanto às questões de terceirização de serviços, especificamente à mobilidade dos funcionários, a terceirização de frotas impactou as tomadas de decisões realizadas pelos profissionais envolvidos com a gestão do setor de transportes de uma empresa.

Por exemplo, supondo haver duas empresas, Empresa A e Empresa B, ambas dedicadas a fornecer uma frota para atender às necessidades operacionais de seus colaboradores em suas respectivas atividades comerciais. No caso em que a Empresa A possui uma frota própria composta por 30 veículos, ela assume a responsabilidade por diversas tarefas, incluindo manutenção, contratação de seguros veiculares, quitação de impostos, gerenciamento de multas, instalação de rastreadores, procedimentos para a alienação de veículos desvalorizados, e outros contextos pertinentes. Isso, é claro, sem mencionar a necessidade de mão de obra especializada para gerenciar todas essas demandas.

Por outro lado, a Empresa B, ao identificar a necessidade de uma frota semelhante à da Empresa A, pode optar por uma abordagem mais econômica e prática. Nesse cenário, a Empresa B tem a opção de terceirizar parte ou até mesmo toda a sua frota, contratando uma única empresa que oferecerá todos os serviços mencionados anteriormente. Esta empresa intermediária, por sua vez, pode decidir terceirizar parte desses serviços com outras empresas, como Empresa C, D, E, e assim por diante. Essa estratégia permite à Empresa B concentrar-se em suas operações principais, enquanto desfruta dos benefícios da terceirização para otimizar eficiência e recursos.

Além disso, caso o gestor de transporte estiver direcionando sua atenção para atender às demandas internas, a terceirização se revela como uma alternativa viável. Nesse cenário, o gestor pode concentrar seus esforços na coordenação eficiente das solicitações de uso de veículos, garantindo a mobilidade dos colaboradores com base na disponibilidade da frota. Além disso, ao optar pela terceirização, o gestor pode focalizar suas atividades na apuração de

responsabilidades relacionadas a avarias, infrações e até mesmo no monitoramento de possíveis utilizações indevidas dos veículos para fins pessoais por parte dos colaboradores. Esse enfoque permite uma gestão mais especializada e eficaz das operações de transporte, liberando o gestor para lidar de maneira mais estratégica com aspectos cruciais do seu papel.

Contudo, vale dizer que não se trata de uma análise simples, ensina o autor, já que devem ser considerados os custos aparentes e os invisíveis, além dos potenciais problemas advindos do contrato de terceirização ou da aquisição do recurso, das garantias e salvaguardas necessárias, o grau de confiança, entre outros (Silva Filho, 2006). De todo modo, uma coisa é certa, mesmo com todos estes casos, o gestor do transporte precisa estar atento às demandas que o contexto de mobilidade empresarial exige.

De modo a auxiliar nesta gestão, conforme for a demanda, um software que traga informações íntegras e precisas pode ser uma ferramenta útil para que o gestor possa estudar gargalos que precisam ser solucionados. Este estudo só é possível pela realização da análise dos dados registrados e gerados conforme as necessidades. Por exemplo, um sistema que auxilie no agendamento de viagens pode ser útil para promover a manutenção preventiva de modo antecipado, bem como a limpeza veicular. Ainda, após um período de utilização do próprio sistema, pode ser possível realizar a previsão de quais períodos estas requisições são mais demandadas e os contextos que as justifiquem.

2.2 Gestão de Reserva de Veículos

A gestão de reserva de veículos de uma frota colaborativa de uma empresa é o processo de gerenciamento de veículos compartilhados entre os funcionários da empresa, em que os colaboradores podem reservar um veículo para uso em suas atividades de trabalho. Este processo envolve o planejamento e a organização de todas as solicitações de reserva de veículos feitas pelos usuários. Isso inclui a verificação da disponibilidade dos veículos, a alocação de veículos para as solicitações de reserva, a manutenção dos veículos, a gestão dos motoristas, entre outros aspectos.

Para a gestão eficiente de reservas de veículos, é necessário que a empresa tenha um sistema adequado que permita o monitoramento e controle de todas as solicitações de reserva. Esse sistema deve ser capaz de armazenar informações sobre os veículos, usuários e motoristas, além de permitir a realização de reservas, consultas e relatórios. O sistema pode ser gerenciado por meio de uma plataforma online ou aplicativo, em que os colaboradores podem visualizar os

veículos disponíveis, fazer reservas, verificar a disponibilidade e status dos veículos, bem como gerenciar as devoluções e manutenções.

Alguns dos principais benefícios da gestão de reserva de veículos de uma frota colaborativa incluem a redução de custos, a otimização do uso dos veículos, a redução do tempo ocioso, o aumento da produtividade e a melhoria da sustentabilidade ambiental. Além disso, pode ajudar a empresa a monitorar o uso dos veículos e a identificar oportunidades de melhorias na gestão da frota.

2.3 Software para Gestão de Reservas de Veículos

Para que seja possível explanar de forma prática as questões de utilização de softwares para a gestão de frotas, é preciso abordar alguns casos existentes que se enquadrem neste contexto. Deste modo, cita-se os estudos de Pereira (2016), que criou um sistema de gerenciamento de veículo onde se pode controlar a frota pela internet; gerar relatórios de consumo médio, distância percorrida, viabilidade, despesas diversas e impacto financeiro.

Ao fim do estudo, Pereira (2016) traz a conclusão de que o sistema consegue auxiliar na gestão de frotas, no controle de operações, na geração de relatórios, na responsividade, na visão ampla da gestão. Porém, pontua algumas melhoras necessárias ao sistema:

- Ter um módulo de rastreamento de veículos;
- Ter um módulo de controle dos pneus;
- Ter relatório dos motoristas;
- Ter opção de envio dos relatórios ao e-mail do usuário;
- Ter consulta de multas diretamente com o órgão responsável;
- Ter mensagens de boas práticas.

Há também a experiência do trabalho de Rodrigues (2019) que viabilizou a implantação de um sistema de gestão de frotas de veículos pela Prefeitura de Capitólio - MG, além de cuidar da manutenção da frota e auxiliar na tomada de decisões.

Ao cabo dos trabalhos, Rodrigues et a (2019) confirma a capacidade do sistema em gerenciar informações referentes a manutenção e limpeza de automóveis, sendo o destaque para a disponibilidade de informações e de ser canal de comunicação efetivo entre os usuários. Eles também atestaram a capacidade para gerar relatórios de manutenção, sendo uma plataforma de web com boa responsividade.

Ter acesso a relatório é uma função chave dentro de sistemas de gestão, pois visualiza a saúde e o desempenho dos negócios, sendo a bússola para as tomadas de decisões da administração. Nas palavras do autor:

> "Percebeu-se que a implantação de um sistema de gestão de frota permite definir especificações adequadas à execução dos serviços de manutenção e limpeza, racionalizar o processo de aquisição de peças de reposição e reduzir a contratação de serviços de terceiros. Viabiliza ainda, a diminuição do número de veículos ociosos, assim como, disponibilizar informações para um melhor planejamento sobre a manutenção preventiva dos mesmos" (RODRIGUES, 2019).

Vale considerar os resultados dos estudos de Silva, R., (2020) que criou um sistema de controle, gerenciamento, armazenamento e publicação dos dados pela Prefeitura Municipal de Lajes - RN, analisando o SYSFROTA.

No relato final, Silva, R., (2020) confirma que o Sistema é capaz de gerir, armazenar e publicar os dados referentes aos gastos com combustíveis da Prefeitura, apontando os erros de controle, sendo de fácil utilização e com capacidade de gerar relatórios legíveis e de qualidade. Dentro dos pontos negativos, foram pontuados que o sistema não permite acesso via internet, apenas uso local e que o sistema não tem a opção de relatório dinâmico.

Porém, a experiência serviu para contribuir com apontamentos importantes aos que pretendem desenvolver uma aplicação de gerenciamento de frotas, já que são melhorias propostas ao próprio SYSFROTA, quais sejam:

- Ter relatórios dinâmicos;
- Gerar relatórios em pasta indicada pelo usuário;
- Apontar as anomalias de desempenho da frota;
- Ter versão com acesso via internet;
- Ter opção de exportar os dados diretamente no Portal da Transparência;
- Ter relatórios com gráficos de fácil visualização;
- Ter layout facilitado e integrado com o usuário do sistema;
- Ter mais testes de usabilidade;
- Melhorar a definição do público-alvo.

Comparando os resultados, pode-se concluir que o SYSFROTA viabilizou mais agilidade ao sistema, principalmente na execução de tarefas, na comunicação entre os setores, além de armazenar informações importantes à gestão de frotas, dando confiabilidade aos dados.

2.4 Desenvolvimento de Software

O desenvolvimento de Software está inserido na disciplina de Engenharia de Software, conjuntamente com processos técnicos de produção, especificação e manutenção de sistemas (Pressman, 2016). Complementa Sommerville (2011) que, dentro do desenvolvimento de software, o foco pode-se estar no gerenciamento de projeto, desenvolvimento de ferramentas ou de métodos que apoiem a produção de um sistema de alta qualidade.

Nas palavras de Keyes (2002) a Engenharia de Software abarca métodos e ferramentas de desenvolvimento que identificam as diferentes etapas para desenvolver um sistema, desde a escolha do sistema, do estudo de viabilidade, do design, do plano de teste completo até a implantação satisfatória para os usuários finais. Essa estrutura é reafirmada por Sommerville (2011) quando elenca as atividades fundamentais para o desenvolvimento de software nos mais diversos processos: Especificação de Software, Projeto de Implementação de Software, Validação de Software e Evolução do Software.

No primeiro, são identificadas e mapeadas as funcionalidades do sistema e suas restrições, no segundo, é desenvolvido o sistema seguindo as especificações. No terceiro, é verificado se os requisitos do cliente foram atendidos pelo sistema, e, no quarto, devem ser feitos os aprimoramentos para atender as necessidades novas do cliente.

Existem diversos modelos de processo para desenvolvimento de Software. Cada um deles representa uma abordagem diferente, que viabiliza a conexão dos agentes internos e externos da relação, a garantia de estabilidade, o controle e a organização do sistema.

Na Engenharia de Software o foco é definir a sequência de atividades procedimentais que serão realizadas, existindo uma diversidade de modelos de processo (paradigmas) que buscam estruturar de forma útil a execução dos trabalhos. Cada modelo irá referir-se a uma abordagem usada na criação de softwares, integrando os usuários que irão desempenhar papéis nesses processos, garantindo estabilidade, controle e organização.

Há o modelo Cascata, que aborda um desenvolvimento de software sequencial, iniciando-se o processo na fase de requisitos e análise, seguindo pelo projeto, implementação, integração, operação e manutenção (Pressman, 2016). Ele geralmente se aplica a processos que focam em planejamentos e que são definidos antes da execução. Então, para esse modelo primeiro atende-se às necessidades específicas, analisa-se as restrições e as funções identificadas no sistema.

Sendo essa fase um pré-requisito para seguir com a definição da arquitetura do sistema, dos subsistemas do software e suas correlações. Depois, vem a codificação do software e os

testes de validação para a implementação. Cumprindo essa etapa, segue-se para a integração do sistema e a entrega ao cliente, sendo a fase final de manutenção e correção dos erros do sistema, além de atualização e aperfeiçoamento do sistema.

Os problemas inerentes desse modelo são: o fluxo sequencial quase nunca é utilizado em projetos reais; há dificuldade do cliente em definir os requisitos de forma direta; e, o tempo de entrega do produto final ao cliente deve esperar a modelagem. Sendo um defeito crônico do modelo a inflexibilidade de adaptação para correção de erros cometidos em fases já concluídas (Pressman, 2016).

Já o modelo Incremental vem como uma melhoria ao modelo anterior, ele cria pequenas partes (incrementos) dentro do software entregando cada um de uma vez, desenvolvendo-se de forma linear. Ele combina elementos do modelo Cascata com uma interatividade que visa o progresso por meio de novos incrementos, melhorando a cada novo resultado (Pressman, 2016). Esse modelo se adequa em ambiente onde não há mão-de-obra disponível para a implementação do projeto, a técnica será aperfeiçoada pelo desenvolvimento ágil.

Metodologias ágeis são as mais novas técnicas de produção de software, que levam em consideração processos mais leves e flexíveis, com ciclos cada vez mais curtos. Ela surge para adaptar-se aos novos fatores de desenvolvimento de projetos, com previsão de mudanças (Pontes et al, 2018). Dentro desse conceito há pontos essenciais:

- Foco no indivíduo e nas interações e menos em processos e ferramentas;
- Foco maior no software mais do que na documentação;
- Foco na Colaboração do cliente mais do que nos contratos;
- Foco em respostas rápidas às mudanças mais do que aos planos rígidos.

A utilização de metodologias ágeis é útil também nos contextos de conectividade e desenvolvimento. Para Cechinel et al (2017), o surgimento do World Wide Web (WWW), em 1989, viabilizou o desenvolvimento de software para a internet. O novo sistema possibilitava acesso fácil às informações, de qualquer lugar, desde que conectadas na rede e apresentadas em formato de linguagem de marcação de hipertexto (HTML). O sistema WWW evoluiu bastante ao longo dos anos, tendo três ou quatro ondas, nos ensinos de Choudhury (2014): A WWW apenas de leitura, a WWW de leitura e escrita, a WWW programável e a nova fase com a WWW simbiótica.

A versão web 1.0 nasce com Tim Burners-Lee, ela é considerada apenas de leitura, pois permitia que as aplicações fornecessem informações estatísticas limitadas para comunicação e

interação com os leitores. Apesar de que havia capacidade de variar os protocolos da web, sendo o mais comum o Hypertext Transfer Protocol (HTTP).

Na segunda versão (web 2.0) tem-se a chamada web Social ou web leitura e escrita, nela havia um conjunto de padrões de design e arquiteturas, além da migração dos negócios para a internet. Foca na interação e participação entre comunidades, pessoas e software, principalmente nas aplicações de web (Cechinel et al, 2017).

Com a terceira onda, chamada de web 3.0, era a web Executável que era focada na facilidade de desenvolver sistemas completos e otimizar a experiência online, principalmente pelo surgimento da computação em Nuvem (Kulesza et al, 2018). Por fim, a quarta onda era a web simbiótica com o objetivo de integrar homem e máquina (Aghael et al, 2012), toda essa evolução mudou radicalmente o desenvolvimento de aplicações web. O que vai além de softwares que podem ser utilizados apenas em navegadores.

Existem vários sistemas que servem para o desenvolvimento multiplataforma. Para desenvolver-se aplicações nativas, deve-se ter em foco as linguagens e ambientes específicos de cada plataforma. A produtividade e manutenção de aplicações com foco comercial influenciou o surgimento de alternativas com maior produtividade e abrangência de plataformas com menor esforço, o que foi chamado de abordagens híbridas, interpretada e generativa (Hanschke et al, 2013).

Framework passaram a ser utilizados para que o desenvolvimento e alcance fossem agilizados. Eles são uma estrutura de construção de aplicações e dentre outros projetos digitais, com soluções prontas para desenvolvedores, presente um conjunto de códigos já pré-definidos que buscam auxiliar novos projetos. Ou seja, são constituídos de padrões, com aplicação de template que podem ser seguidos para criar projetos, que permitem a obtenção de todo o poder do framework, contendo um método de desenvolvimento específico e espaços vazios para serem preenchidos pelos desenvolvedores (Duarte, 2015).

Segundo Fayad et al (1999) utilizar frameworks traz os seguintes benefícios:

- Melhora a modelagem – implementação dos componentes voláteis são envoltos em interfaces estáveis e que o usuário não consegue modificá-lo;
- Aumenta o reuso – desenvolve componentes genéricos que podem ser reaplicados na produção de novos sistemas;
- Extensão – uso de metodologias *hook* que permitem que as aplicações utilizem interfaces estáveis;

- Inversão de controle – o framework controla o fluxo global de execução dos programas em vez de ser o desenvolvedor.

Existe uma série de frameworks para o desenvolvimento híbrido, interpretativo, generativo que se vale de uma tecnologia para várias plataformas, como por exemplo: Cordova, PhoneGap, IBM Worklight, APPMOBI, Titanium, LiveCode, Rhomobile, React Native, Haxe, Kotlin, Flutter, entre tantos outros. Cada um tem suas características específicas para desenvolvimento de aplicações, seja o ambiente de desenvolvimento integrado, a arquitetura de desenvolvimento, as permissões, o acesso aos recursos, dentre outros.

Especificamente no desenvolvimento de aplicações web, usam-se ferramentas como a quinta versão do HTML (HTML5), Cascading Style Sheets (CSS) e JavaScript (Amatya et al, 2014). Os sistemas web também são chamados de aplicativos HTML5, pois diferem-se dos híbridos em algumas características. Uma dessas diferenças é o fato de que eles não são vendidos em lojas de aplicativos, mas por meio de endereço de Uniform Resource Locator (URL).

Para desenvolvimento de software web deve-se observar a adaptação aos diferentes tamanhos de tela para as multiplataformas, sendo o parâmetro *viewport* do HTML5 o mais utilizado, nele se pode controlar a exibição da página no dispositivo do usuário, essa adequação foi conceituada como responsividade (Marcotte, 2010). Para atividades mais complexas, para formulários com grandes volumes de dados, utilização de mapas e imagens, dar-se-á preferências aos dispositivos com tela maior. Contudo, desde que o sistema contenha o devido tratamento para simplificar a exibição destes dados, ele pode ser também operado em dispositivos móveis.

Outra preocupação do desenvolvedor é que o acesso à aplicação seja possível nos diferentes dispositivos. Isto leva ao aumento da produtividade e da manutenção do desenvolvimento, já que uma única aplicação pode ser utilizada em qualquer dispositivo que suporte um navegador, o que não ocorre em aplicativos híbridos (Hoffa et al, 2008). Uma ferramenta que ajuda na responsividade em diferentes dispositivos e plataformas, e que é largamente utilizada, é o Bootstrap. Ele provê bibliotecas que adaptam a interface de sites nos mais diversos tamanhos, orientando a tela conforme a necessidade. Também tem recursos que tornam a interface mais apresentável sem muito esforço do desenvolvedor. Por fim, cita-se que as aplicações de web não necessitam de instalação em dispositivos, a carga de memória é baixa, pois armazena e processa praticamente tudo no servidor.

As arquiteturas do sistema de software que trazem a adaptação de conteúdo permitem a alteração da exibição quando direcionada a diferentes computadores pessoais e tamanhos de

telas. A melhor solução para a adaptação do conteúdo que pode ser realizada, segundo os estudos de Buchholz e Schill (2003):

- No servidor de origem do site: de modo que o servidor deve ter versões do mesmo conteúdo, após a identificação do dispositivo envia-se o produto dentro das limitações daquele.
- No dispositivo do usuário: as adaptações ocorrem conforme os recursos do dispositivo, mas pode gerar uma sobrecarga de processamento ou uma demora no tempo de resposta. Ensinam Roudaki et al (2015) que o sistema HTML5 viabiliza o ocultamento ou não de elementos conforme o tamanho e a orientação da tela do dispositivo.
- No servidor dedicado: neste há uma ferramenta de adaptação que funciona em um servidor dedicado, captando os requisitos de um dispositivo e adaptando para outro. De certo modo não sobrecarrega a comunicação, segundo Hofman et al (2000), esses sistemas podem escanear vírus, classificar e filtrar conteúdo, traduzir linguagens, decodificar imagens e inserções de publicidade, entre outras funções.

2.4.1 Desenvolvimento Front-End Moderno

A arquitetura de um software deve considerar três camadas, que são:

- Camada de apresentação (ou camada de interface do usuário): é responsável por apresentar a informação ao usuário e receber as entradas do usuário. É a camada com a qual o usuário interage diretamente e geralmente é composta por elementos como menus, botões, janelas e formulários.
- Camada de lógica de negócios (ou camada de processamento): é responsável por processar as informações recebidas da camada de apresentação e realizar as operações necessárias para atender aos requisitos de negócios do software. Essa camada contém a lógica do programa e as regras de negócios.
- Camada de acesso a dados (ou camada de armazenamento): é responsável por gerenciar o acesso aos dados do software, como informações do usuário, configurações e dados de aplicativos. Essa camada pode interagir com bancos de dados, sistemas de arquivos e outras fontes de dados.

Figura 1. Arquitetura de um sistema

Fonte: Elaborado pelo Autor

A camada de apresentação (ou camada de interface do usuário) é o front-end, pois é a parte da aplicação que o usuário interage diretamente. Essa camada inclui todos os elementos visuais e de interação com o usuário, como a interface gráfica, as animações, o layout e o design em geral.

A camada de lógica de negócios (ou camada de processamento) e a camada de acesso a dados (ou camada de armazenamento) juntas são conhecidas como back-end. O back-end é a parte da aplicação que geralmente não é visível para o usuário e é responsável pelo processamento e armazenamento dos dados da aplicação. A camada de lógica de negócios lida com a lógica da aplicação, enquanto a camada de acesso a dados lida com a persistência dos dados em um banco de dados ou outro sistema de armazenamento.

Portanto, são vários fatores que devem ser considerados quando se pensa em tratativa de dados e as funções que operacionalizarão as demandas para as quais o software deverá lidar. Neste sentido, conforme anteriormente citado, considera-se que a utilização de bibliotecas, frameworks e interfaces de programação de aplicações (APIs) facilitam o desenvolvimento e as tratativas a serem consideradas. Inclusive, para Stein Junior (2019), estas ferramentas têm a

finalidade de otimizar o tempo do desenvolvedor. Por exemplo, na linguagem de CSS, existe o SASS5 e o LESS6, que vão fazer a reutilização do CSS; na linguagem HTML5 haverá diversos frameworks, sendo o Bootstrap o mais difundido. Para o JavaScript há o Angular e o React.

Para se discutir meios de renderização de páginas de web, criação de aplicações, arquitetura de software, similaridades e interações entre cliente-servidor, é preciso ter uma visão ampla de temas contemporâneos. Ainda mais, com as novas responsabilidades que a camada front-end ganhou a partir da maior capacidade dos navegadores modernos (Stein Junior, 2019).

Pelo exposto, quando se fala em desenvolvimento moderno, entende-se que um software pode ser desenvolvido por meio de bibliotecas, frameworks e API's, pois podem ajudar a elaborar interfaces mais robustas, organizadas e na preservação de procedimentos que envolvam a tratativa de seus dados. Dessa forma, também deve ser considerada a relação dos usuários, seja na qualidade de cliente ou desenvolvedor, sendo ela decisória para cumprir com as necessidades de ambos os lados, pois a utilização do software é o que mantém a relação entre ambos e que dão sentido aos conceitos de front e back-end.

2.4.2 Desenvolvimento de API HTTP REST

Durante o desenvolvimento de API's HTTP REAT, de acordo com Tarkowska et al., 2018, devem ser consideradas as seguintes onze dicas:

1 Use verbos HTTP de maneira consistente para a realização de operações Create, Read, Update, Delete (CRUD): é importante usar os verbos HTTP corretos (GET, POST, PUT, DELETE) para as operações CRUD em seus recursos. Isso ajuda a tornar sua API mais intuitiva e fácil de usar.

2 Use plural para nomes de recursos: usar nomes no plural para seus recursos ajuda a tornar sua API mais consistente e fácil de entender. Por exemplo, em vez de usar "/user" como endpoint para obter informações do usuário, use "/users".

3 Use sub-recurso para relacionamentos entre recursos: o uso de sub-recursos é uma maneira eficaz de representar relacionamentos entre recursos em sua API de Transferência de Estado Representacional (REST). Por exemplo, se você tiver um recurso "user" e um recurso "post", pode usar "/users/{userId}/posts" como endpoint para obter todos os posts associados a um usuário específico.

4 Forneça filtros, classificação e paginação para coleções de recursos: se você tiver coleções grandes de recursos em sua API RESTful, é importante fornecer

mecanismos de filtragem, classificação e paginação para ajudar os usuários a encontrar o que estão procurando com mais facilidade.

5 Use códigos de status HTTP apropriados e forneça mensagens de erro úteis: o uso correto dos códigos de status HTTP ajuda os usuários da sua API a entenderem o resultado da solicitação que fizeram. Além disso, fornecer mensagens claras e úteis em caso de falha na solicitação pode ajudar os usuários a corrigir o problema.

6 Adote o formato JSON como padrão para troca de dados: o formato JSON é amplamente utilizado em APIs RESTful e é fácil de entender e usar. Adotar o formato JSON como padrão para troca de dados entre a API e seus clientes pode ajudar a tornar sua API mais acessível.

7 Forneça documentação clara e completa da API: fornecer documentação clara e completa da API é essencial para ajudar os usuários a entenderem como usar sua API. A documentação deve descrever todos os endpoints disponíveis, parâmetros aceitos e exemplos claros de como usar a API.

8 Adote o padrão OpenAPI (anteriormente conhecido como Swagger) para documentação da API: o OpenAPI é um padrão aberto para documentação de APIs RESTful. Adotar o OpenAPI como uma maneira padronizada de documentar sua API pode ajudar a torná-la mais fácil de entender e usar.

9 Forneça exemplos claros e completos de solicitações e respostas da API: fornecer exemplos claros e completos de solicitações e respostas da API pode ajudar os usuários a entenderem como usar sua API. Os exemplos devem incluir todos os parâmetros necessários e as respostas esperadas.

10 Use autenticação e autorização adequadas para proteger a API contra acesso não autorizado: é importante usar autenticação e autorização adequadas para proteger sua API contra acesso não autorizado. Isso pode incluir o uso de tokens de acesso, autenticação baseada em OAuth ou outras técnicas.

11 Verifique se o seu framework web pode ajudá-lo a implementar essas dicas automaticamente: muitos frameworks web modernos têm recursos integrados que podem ajudá-lo a implementar essas dicas automaticamente. Verifique se o seu framework web tem esses recursos disponíveis para economizar tempo e esforço no desenvolvimento da sua API RESTful.

Vale dizer que seguir essas diretrizes pode ajudar pesquisadores, desenvolvedores e gerentes de dados durante a implementação de algum projeto de software web.

> "Muitos dos frameworks compatíveis com REST já implementaram muitas das dicas discutidas acima, em uma variedade de linguagens de programação, como Spring (Java); Django ou Flask (Python); Restify, hapiJS, Express ou Loopback (Node.js); e Catalyst ou Mojolicious (Perl). Se sua API REST é construída usando um desses frameworks, muitas dessas dicas já estarão implementadas ou disponíveis por meio de um plugin para simplificar a implementação." (Tarkowska et al., 2018 - tradução do autor).

Com base nestes pontos, entende-se que o desenvolvimento de APIs HTTP Rest é fundamental para a otimização de projetos científicos que envolvem troca de dados via web. Dessa forma, ao adotar uma API que já faz uso destas dicas em sua concepção, o processo de criação de um software online se torna mais simples. Portanto, pode se afirmar que, durante o desenvolvimento de um projeto que se enquadre nestes contextos, deve ser considerada a utilização de APIs HTTP Rest, visando simplificar sua implementação e, consequentemente, aprimorar a usabilidade, escalonamento e acessibilidade.

2.4.3 Desenvolvimento de Software Multiplataforma

Desenvolvimento em multiplataforma ou cross-plataforma refere-se à criação de um único código-fonte que possa ser compilado em código nativo para diversos sistemas operacionais diferentes. Isso tem a ver com o fato de que as empresas de tecnologia desejam obter o maior número de usuários, precisando ofertar seu produto nas mais diversas plataformas (El-Kassas et al, 2015).

Com esse recurso, as empresas conseguem otimizar a distribuição dos seus produtos, reduzir seus gastos, economizar tempo e não perder em qualidade. Além disso, a demanda por novas aplicações cresceu, entre 2016 e 2019, em 45% (Matos, 2016).

As vantagens e desvantagens do uso de multiplataforma são os seguintes:

- Vantagens:
    - Custos de Desenvolvimento Reduzidos: são baixos e com apenas um código-fonte é possível distribuir em vários dispositivos e plataformas, sem maiores investimentos;
    - Grande Alcance de Mercado: a possibilidade de atingir um maior número de clientes se deve ao grande número de plataformas a disposição

  - Simplicidade: as alterações e manutenções desses sistemas são mais fáceis e sincronizadas automaticamente entre as plataformas

- Desvantagens:
  - Bloqueio de Fornecedor: ao mudar de plataforma seu código pode ser bloqueado, já que muitas estruturas de multiplataformas possuem sua própria sublinguagem de Java Script
  - Problemas de Integração: há possibilidade de incompatibilidade de preferências e notificações nativas, mas é possível a adesão aos sistemas de nuvem com diversas opções de armazenamento.
  - Demora de atualização: a estrutura usual pode não ter todas as ferramentas operacionais para que as mudanças sejam prontamente executadas, precisando de adaptações ou até frameworks mais atualizados.

Sob o pressuposto de contornar as desvantagens, em 2018, a Google lançou o Flutter 1.0, um framework de aplicativos que permite o desenvolvimento multiplataforma através de um único código-fonte em linguagem Dart. Ou seja, por meio dele é possível desenvolver aplicativos para iOS, Android, web e desktop. Ele fornece um conjunto de ferramentas e componentes para a criação de interfaces de usuário responsivas, além de suportar recursos avançados, como animações personalizadas, acesso a APIs e integração com serviços em nuvem. O Flutter também possui um compilador próprio que gera código nativo otimizado para cada plataforma, resultando em um alto desempenho dos aplicativos desenvolvidos. (Flutter, 2023).

Na seção de Perguntas Frequentes de sua documentação oficial, é afirmado que já foram construídos mais de 400.000 aplicativos para os mais diversos dispositivos. Vale destacar que nesta mesma seção é dito que em 2021 ele já se encontrava na versão conhecida como Flutter 2.

De acordo com os dados de Stack Overflow, 2021, Flutter tem sido conquistado espaço entre desenvolvedores de todo o mundo, tornando-o uma das opções mais populares para o desenvolvimento de aplicativos móveis multiplataforma. A seguir segue recorte de grafo elaborado pela pesquisa realizada pelo Stack Overflow:

Figura 2: Loved x Dreaded

Fonte: Loved vs. Dreaded. Stack Over Flow, 2021.

Portanto, por todo o exposto, é entendido que o desenvolvimento multiplataforma tem ganhado foco ao passo que já possui um framework e assim por dizer, uma nova roupagem. Isto abre um cenário competitivo para a produção de softwares pois viabiliza sua utilização por parte de seus usuários das mais variadas plataformas.

3 MATERIAIS E MÉTODOS

Nesta seção, são apresentados os materiais e métodos que foram utilizados no desenvolvimento deste trabalho. Dentre elas, as ferramentas, as tecnologias e as técnicas utilizadas para a realização da pesquisa e coleta de dados, as ferramentas utilizadas para concepção de protótipos e desenvolvimento de softwares, além dos métodos de avaliação e validação do próprio software desenvolvido neste trabalho.

3.1 Materiais

Aqui estão descritos os materiais empregados no desenvolvimento do trabalho, destacando as ferramentas de software e tecnologias específicas utilizadas. Cada uma dessas ferramentas é detalhada a seguir.

3.1.1 Figma

Uma ferramenta utilizada neste trabalho é o Figma, um software de design de interface do usuário (UI) e de experiência do usuário (UX), permitindo que os layouts de sites aplicativos móveis sejam testados, ajudando desenvolvedores e gerentes de projetos a construir feedbacks e a tomar decisões. O Figma é uma ferramenta baseada na nuvem que permite a colaboração em tempo real entre equipes de design, bem como o compartilhamento de projetos (Figma, 2023).

O Figma foi utilizado neste projeto para a criação de protótipos de interface do usuário para o software proposto. Com o Figma, foi possível criar e iterar rapidamente as diversas telas de interface do usuário gestor, testando diferentes layouts, fluxos de navegação e interações.

Além disso, o Figma possui uma ampla biblioteca de componentes de interface do usuário que foram utilizados para acelerar o processo de design. Com esses componentes, é possível criar interfaces mais facilmente e de forma consistente, garantindo uma experiência do usuário mais agradável e coesa.

Nesta pesquisa, o Figma está vinculado ao processo inicial de desenvolvimento, de modo que a refletir os fluxos e funcionalidades previstas no código do software. Dessa forma, é possível garantir que a interface do usuário gestor seja consistente com a visão do projeto e que atenda às suas necessidades.

3.1.2 Supabase

O Supabase é um serviço de banco de dados e autenticação baseado na nuvem que oferece uma alternativa open source - ou código aberto - e escalável aos serviços tradicionais de banco de dados. Ele é construído com base no PostgresSQL, um sistema de gerenciamento de banco de dados relacional amplamente utilizado (Supabase, 2023).

Neste projeto, o Supabase foi utilizado como o sistema de gerenciamento de banco de dados do software proposto. Ele oferece recursos destinados à autenticação de usuários, armazenamento de dados e notificações em tempo real. Com a ajuda do Supabase, foi possível criar um banco de dados escalável e seguro para o software.

Além disso, o Supabase também oferece uma API para interagir com o banco de dados, permitindo que, em estudos futuros, os desenvolvedores possam criar endpoints personalizados para a comunicação entre o software e o banco de dados. Isso torna a comunicação mais eficiente e rápida, já que será possível adaptar a API de acordo com as necessidades do software, pois, segundo sua biblioteca de clientes e produtos oferecidos, sua aplicação é voltada a seis linguagens de programação, incluindo recursos para JavaScript e Flutter (Supabase, 2023).

Portanto, o Supabase está presente no processo de desenvolvimento deste trabalho, de modo que as atualizações feitas no banco de dados sejam diretamente refletidas no software. Dessa forma, é possível garantir que o sistema tenha acesso aos dados atualizados e que as alterações no banco de dados sejam realizadas de forma segura e consistente.

3.1.3 Flutter

O desenvolvimento do software proposto envolveu a utilização de a linguagem de programação Dart e o framework Flutter para a criação da interface do usuário e a funcionalidade do sistema. Ambos estão presentes no pacote do Android Studio para Windows.

O Flutter foi escolhido também por este trabalho visar a possibilidade de um desenvolvimento multiplataforma no futuro, através de um código responsivo. Como framework desenvolvido pelo Google, ele permite a criação de aplicativos nativos para Android, iOS e outros dispositivos com uma única base de código. Considerando a data deste trabalho, o Flutter é uma tecnologia relativamente nova, mas vem ganhando cada vez mais popularidade entre os desenvolvedores de aplicativos móveis.

No contexto deste projeto, este supracitado framework foi utilizado para desenvolver o aplicativo móvel que permite ao usuário realizar a gestão das reservas de veículos da frota

corporativa. Ressalta-se que com o Flutter, é possível criar um aplicativo com um visual atraente, responsivo e moderno, o que potencializa uma experiência positiva ao usuário.

Consequentemente, ele permite a criação de aplicativos nativos com performance e escalabilidade. Menciona-se que ele abarca várias possibilidades de personalização e diversidade de componentes - ou widgets – pré-construídos, e ainda a incorporação de bibliotecas, que podem ser consultadas e acessadas pelo endereço https://pub.dev/, repositório gratuito, público e oficial de pacotes para aplicativos Dart e Flutter. Estas características são facilitadores que contribuem para o processo de desenvolvimento.

Portanto, frisa-se que o uso dessas tecnologias e framework garantiu a criação de uma interface de usuário intuitiva e agradável, além de permitir o desenvolvimento de um software robusto e escalável. Deixando como oportunidade futura seu aperfeiçoamento e escalonamento para as plataformas web e desktop.

3.2 Métodos

A seguir, são descritos e detalhados os métodos utilizados na pesquisa e no desenvolvimento do software proposto.

3.2.1 Entrevista

Para Lakatos e Marconi (2010), a entrevista é uma técnica qualitativa de coleta de dados que tem como objetivo obter informações aprofundadas sobre as percepções, atitudes, crenças e experiências dos entrevistados. Para este trabalho, foram realizadas entrevistas com profissionais dos mais diversos setores que pertencem a empresas que possuem frotas corporativas de veículos, com o objetivo de entender as necessidades e desafios enfrentados por essas empresas no gerenciamento de reservas de veículos.

De acordo com Minayo (2008), as entrevistas podem ser classificadas em vários tipos. Logo abaixo, é possível verificar suas características de acordo com suas tipagens:

Tabela 1: Tipos de Entrevistas

Tipos de Entrevistas	Características
Entrevista totalmente	Um questionário totalmente estruturado é utilizado, onde a seleção do entrevistado depende da sua capacidade de responder às perguntas formuladas

estruturada /questionário/sondagem de opinião	pelo investigador. Tipo de método de coleta de dados mais comumente utilizado em pesquisas quantitativas
Entrevista semiestruturada	Pode conter perguntas fechadas, que geralmente são usadas para identificação ou classificação, mas, principalmente, contém perguntas abertas, permitindo que o entrevistado fale de forma mais livre sobre o assunto em questão.
Entrevista aberta ou em profundidade	O entrevistado é encorajado a expressar suas ideias de forma aberta sobre um determinado tema, enquanto o entrevistador pode fazer perguntas adicionais para obter respostas mais detalhadas e aprofundadas.
Entrevista focalizada	Direcionada exclusivamente para um problema específico.
Entrevista projetiva ou narrativa	Geralmente são empregadas para lidar com assuntos ou temas desafiadores de serem discutidos. Pode-se utilizar recursos visuais, como filmes, vídeos, ilustrações, etc., como uma forma de convidar o entrevistado a participar. Dentro dessa abordagem, se incluem as narrativas de vida e os grupos focais.

Fonte: Minayo, 2008 (adaptado pelo autor).

Para esta pesquisa, as entrevistas foram conduzidas de forma estruturada, semi-estruturada e focalizada, com perguntas específicas e abertas para permitir a obtenção de informações detalhadas e aprofundadas sobre o assunto em estudo. As entrevistas foram realizadas presencialmente e por meio de chamadas de vídeo, de acordo com a disponibilidade dos entrevistados, com aplicação de questionário elaborado pela Ferramenta Google Forms, que pode ser consultado nos apêndices deste trabalho.

3.2.2 Análise de dados

As informações coletadas por meio das entrevistas foram analisadas e categorizadas para identificar as funções prioritárias a serem desenvolvidas que se adequem às necessidades relacionados à gestão de reservas de veículos. Esses resultados foram utilizados para orientar a definição do escopo de desenvolvimento do software proposto, conforme resultados apresentados no próximo capítulo.

3.2.3 Prototipação e Elaboração de Fluxograma

A prototipação permite a identificação de recursos necessários no desenvolvimento e a definição das tratativas a serem abordadas. Dessa forma, com base na análise das respostas obtidas durante as entrevistas, o projeto seguiu para a criação do protótipo no Figma. Ele recebeu o nome de “Linkar”, uma fusão das palavras em inglês *link* (ligação) e *car* (carro). A segunda palavra também pode ser interpretada como uma possível abreviação de sua tradução. De todo modo, a nomenclatura revela um trocadilho verbal, pois “Linkar” refere-se à capacidade do aplicativo de estabelecer vínculos entre seus elementos, criando rotas intuitivas e dinâmicas, com uma rede de interconexões e fluxos entre as diversas telas.

O protótipo forneceu subsídios para a elaboração de um fluxograma que traz a representação visual das rotas definidas pelas funcionalidades. Esse fluxograma pode ser encontrado no capítulo destinado à análise dos resultados obtidos com este projeto. Após a prototipação, foi possível realizar implementação. Vale dizer que o nome “Linkar”, outrora pertencente ao protótipo, também foi mantido como nome do sistema durante a sua implementação.

3.2.4 Implementação

Seguindo com a proposta deste projeto, mediante a definição das funções que estariam dentro do escopo, o módulo do gestor da frota foi desenvolvido em Flutter, por meio da linguagem de programação Dart. Ambos estão presentes no Android Studio, ferramenta disponibilizada pela Google para o desenvolvimento de aplicações.

A implementação se apoiou em pacotes disponibilizados no site https://pub.dev, repositório oficial para aplicações criadas pelo Flutter. Devido às tratativas da prototipação e do banco de dados utilizado pelo Supabase, percebeu-se que bibliotecas deveriam ser importadas, além de que poderiam auxiliar no processo de desenvolvimento. Como, por exemplo, o fato de o aplicativo lidar com o gerenciamento de dados que evolvem o registro de horários e datas, o auxílio de um calendário já construído poderia facilitar o tratamento dos dados e a visualização das informações.

Segundo pesquisa realizada diretamente no Pub.dev (2023), sendo este o repositório oficial de pacotes para o framework Flutter, as seguintes bibliotecas foram escolhidas para serem incorporadas ao código:

- flutter_localizations:

Este pacote é usado para internacionalizar aplicativos Flutter. Segundo o repositório, ele permite que seja adicionado suporte a vários idiomas no aplicativo. Portanto, este é uma biblioteca fundamental para a tradução de alguns componentes que exibem opções de seleção de data e horário;

- intl:
  Segundo Dart.dev (2023), este pacote fornece funcionalidades para lidar com mensagens, formatação e análise de datas e números, texto bidirecional e outras questões de internacionalização. Necessário para traduzir para o português do Brasil o componente "TableCalendar";
- table_calendar:
  Este é um componente que apresenta um calendário completo altamente personalizável e repleto de recursos para Flutter, como personalização de decorações, formatação de exibição, tradução de dias da semana, exibição de objetos vinculados ao dia selecionado, criação de funções, alteração de estados, dentro outros. Especificamente, este componente foi fundamental para que as reservas ainda não encerradas sejam exibidas como eventos conforme o dia selecionado;
- flutter_expandable_fab:
  Segundo, Zuvola.com (2023), esta biblioteca é útil para criar botões flutuantes animados e expansíveis, com vários recursos de personalização: estender outros botões em forma de leque, ou em linha reta, de forma horizontal ou vertical, manipular a quantidade de botões extensíveis ou chamar funções.
- supabase_flutter:
  Segundo Supabase.io (2023), este pacote é necessário para a integração do Flutter com o Supabase e permite simplificação no processo de criação de produtos escaláveis e seguros aos desenvolvedores.

A instalação destas bibliotecas pode ser feita por meio da adição de linha de código dentro de "dependencies", constante no arquivo pubspec.yaml. Este arquivo é parte essencial do desenvolvimento de aplicativos Flutter, pois é nele que as dependências são declaradas e gerenciadas. Portanto, essas dependências representam bibliotecas e pacotes externos instalados ao framework e permitem suas utilizações através de importação realizada em linha do código durante a execução do aplicativo.

4 RESULTADOS E DISCUSSÃO

Considerando os objetivos delineados, este estudo fundamentou-se nas informações obtidas por meio de entrevistas aplicadas a potenciais usuários, incluindo gestores e desenvolvedores, com o intuito de viabilizar o desenvolvimento. As entrevistas foram registradas por meio de um questionário online, empregando a ferramenta Google Formulários, intitulado “Pesquisa para TCC II - Funções úteis em um aplicativo de Reserva de Veículos”. Esse método simulou uma análise de requisitos, permitindo a definição do escopo das funcionalidades essenciais para a pesquisa.

A partir da análise das respostas obtidas, foi concebido o protótipo do software utilizando a plataforma Figma. Esse protótipo delineou o fluxo entre as telas e funcionalidades do aplicativo. Com o protótipo validado, a pesquisa evoluiu para a fase de implementação, seguindo o escopo estabelecido e os conceitos do referencial teórico. Para garantir a versatilidade do código em futuros projetos e escalonamentos, optou-se pelo desenvolvimento do software por meio do Flutter, possibilitando a compilação para diversas plataformas, como mobile, web e desktop.

O desenvolvimento se beneficiou de tecnologias com ênfase na estruturação de classes, funções e seus relacionamentos. Após a conclusão do desenvolvimento, este trabalho apresenta os resultados obtidos, incluindo discussões relevantes ao tema, e delineia perspectivas para trabalhos futuros. Desta forma, o projeto impulsiona a criação de um software que atua como uma ferramenta de gestão na reserva de veículos em empresas com frotas corporativas, destinadas ao uso de seus colaboradores em suas atividades operacionais.

4.1 Questionário e Análise das Respostas

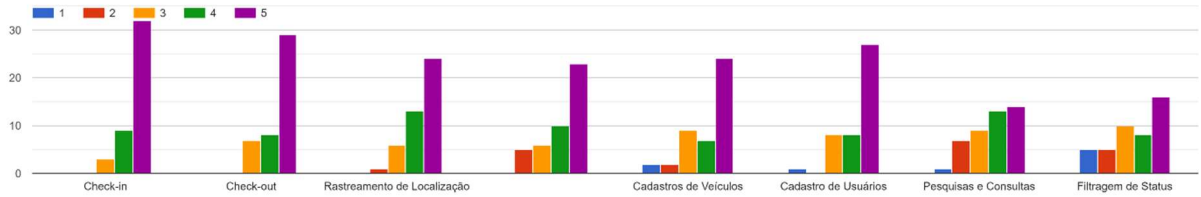
O formulário, intitulado "Pesquisa para TCC II - Funções úteis em um aplicativo de Reserva de Veículos" e divulgado através do Google Formulários, compreendeu um total de 8 perguntas, das quais 5 eram obrigatórias e 3 opcionais. As questões obrigatórias visavam capturar *insights* sobre a importância percebida e a prioridade atribuída às diferentes funcionalidades, ao mesmo tempo em que exploravam as preferências em relação à plataforma de utilização. As perguntas opcionais foram reservadas para a identificação dos entrevistados e para observações adicionais, caso optassem por compartilhar maiores informações.

É relevante destacar que os participantes que optaram por se identificar concederam autorização para o uso de seus dados neste projeto específico. Ao longo dos meses de agosto e

setembro de 2023, 44 participantes registraram suas respostas, fornecendo valiosos retornos para este trabalho.

O gráfico a seguir apresenta os resultados obtidos das funções avaliadas pelos participantes, as quais foram pontuadas com notas 5 (muito importantes) e 2 (pouco importantes):

Figura 3: Importância das Funções



Os entrevistados foram incentivados a atribuir livremente pontuação a essas funções, as quais foram apresentadas como opções de resposta múltipla. Dessa forma, as 5 funcionalidades consideradas mais importantes pelos entrevistados incluem Check-in, Check-out, Cadastro de Usuários, Cadastro de Veículos e Rastreamento e Localização

No entanto, uma abordagem diferente foi adotada em relação à ordenação da prioridade das funções. Quando questionados a respeito de priorização das funções, ou seja, sem que pudessem atribuir livremente pontuações ou notas para as funções, os entrevistaram puderam reordenar as funções que considerassem com prioritárias. As respostas coletadas permitiram a seguinte ordenação:

Tabela 2: Ranking de Prioridade de Funções

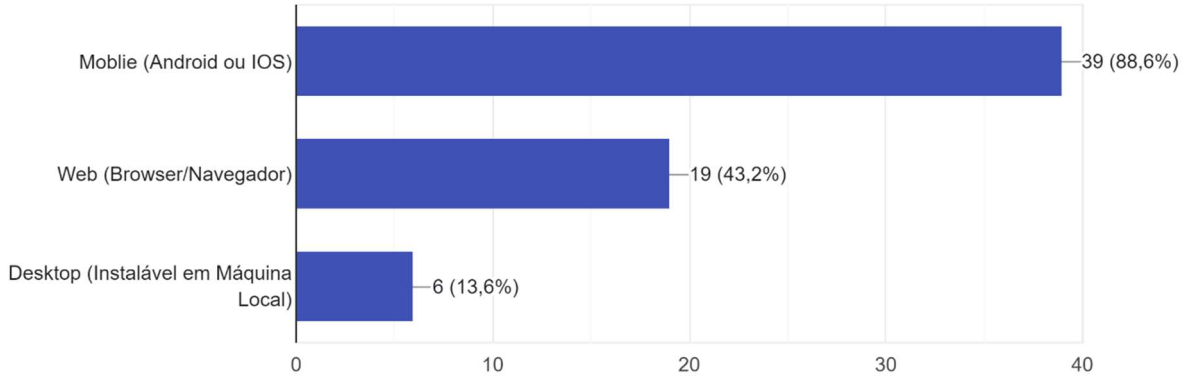
Rank	Função	Pontuação de Prioridade	Quantidade de Funções	Média
1º	Check-in	302	8	37,75
2º	Check-out	272	8	34,00
3º	Rastreamento e Localização	234	8	29,25
4º	Indicação de Status de Veículo	215	8	26,88
5º	Cadastro de Veículos	197	8	24,63

Diante disso, foram definidas as prioridades no escopo de desenvolvimento, concentrando-se em três funcionalidades principais: Check-in, Check-out e Indicação do Status do Veículo. A função Rastreamento e Localização foi excluída para garantir o cumprimento do cronograma estipulado, evitando tratativas complexas envolvendo latitude e longitude. Para

facilitar os testes de maneira objetiva, optou-se por pré-cadastrar os veículos no banco de dados, eliminando a função de Cadastrar Veículos e substituindo-a pelo Cadastro de Reservas.

Quanto à preferência de plataforma para a utilização do software, 88,60% dos entrevistados expressaram maior interesse no desenvolvimento mobile, mesmo considerando a possibilidade de escolher mais de uma plataforma, conforme ilustrado na abaixo:

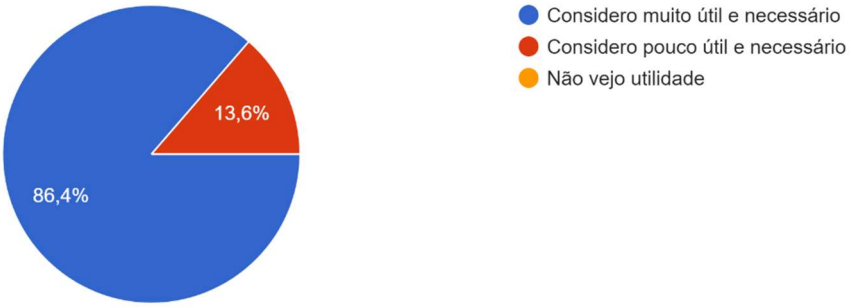
Figura 4: Indicação Preferência de Plataforma



O termo "Mobile" foi utilização para representar a escolha quanto a preferências para dispositivos móveis que possuem como sistema operacional o Android ou IOS. Já o termo "Web" faz referência a sistemas que são executados em navegadores web e o termo "Desktop" foi utilizado para representar plataformas que são instaláveis em sistemas operacionais em máquinas e dispositivos locais.

Os entrevistados puderam expressar seu pensamento quanto a viabilidade de um software mobile de reservas de veículos de uma frota corporativa. Assim, 86,40% dos entrevistados manifestaram-se que consideram a viabilidade como algo muito útil e necessário, conforme ilustrado na figura abaixo:

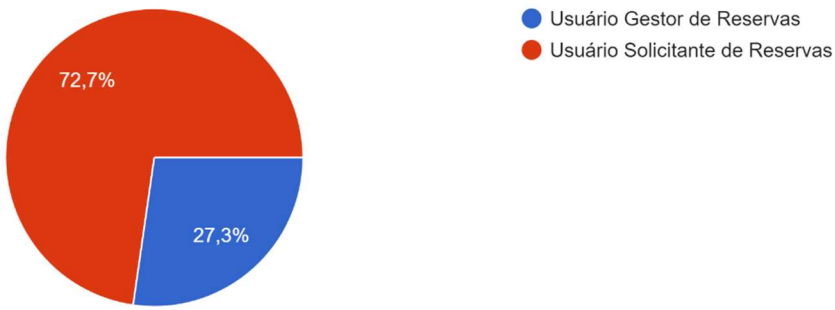
Figura 5: Pensamento Quanto a Viabilidade de Software Mobile de Reservas de Veículos



De acordo com o gráfico, verifica-se que 13,6% do público entrevistado considera que o software abordado nesta pesquisa é pouco útil e necessário. Vale dizer que nenhuma resposta o indica como algo sem utilidade.

A pesquisa também explorou a perspectiva dos entrevistados como usuários gestores ou usuários solicitantes. Apenas 27,30% se identificaram como gestores, como mostrado a seguir:

Figura 6: Perspectiva de Perfil de Usuário



Este resultado já era esperado uma vez que a função desempenhada por um gestor de reservas internas de veículos é um trabalho essencialmente específico, e, numa organização empresarial, dos demais colaboradores seriam aqueles que mais demandariam solicitações ao gestor.

Adicionalmente, ao considerar os cargos ou funções laborativas dos entrevistados, foram identificadas 28 diferentes respostas, abrangendo variedade de profissões, desde analistas de diversas áreas até empreendedores, taxistas e profissionais do setor financeiro, como listado a seguir:

1. Acionista
2. Analista
3. Analista de Processos
4. Analista de Redes
5. Analista de Suporte a Gestão
6. Analista e Desenvolvedora de Sistemas
7. Analista Técnico
8. Analista Técnico - Sebrae/TO
9. Arquiteto de Software
10. Assistente Administrativo

11. Data Security Analyst
12. Desenvolvedor Node
13. Desenvolvedora web
14. Empreendedor
15. Engenheiro Civil
16. Estudante de Contabilidade
17. Gerente Administração e Finanças
18. Gerente de Equipes
19. Gerente da Regional Sudeste do Sebrae/TO
20. Gestor de Contas
21. Médico Veterinário
22. Professora
23. Profissional da Educação Física
24. Profissional do Setor Financeiro
25. Servidor Público
26. Servidor Público e Analista Controle Interno
27. Suporte Técnico
28. Taxista

Dessa forma, o formulário não apenas visou o levantamento de dados, mas preocupou-se em extrair a visão de cada área de atuação, mediante o confronto de todas as respostas. Essa diversidade reflete a abrangência e a aplicabilidade do software proposto em diversos contextos profissionais.

O público entrevistado também pôde expressar livremente suas observações quanto a este projeto. E assim, alinhados diretamente com o teor dos objetivos propostos, destacam-se os seguintes dizeres:

> "Sou gerente do departamento de transportes e o uso dessa ferramenta é de extrema importância para o controle dos veículos, identificação de condutores, status e localização de veículos, facilitando a gestão, manutenções e resguardando a segurança dos nossos colaboradores." (Gerente de Administração e Finanças)

De acordo com a citação acima, percebe-se que o desenvolvimento proposto por este trabalho vai ao encontro das atividades desenvolvidas pela pessoa entrevistada. Dessa forma,

sob a perspectiva de um gerente de transportes, as funcionalidades que permitam a identificação de condutores, status dos veículos e suas localizações são bem-vindas e estão diretamente ligadas a segurança dos colaboradores.

Ainda dentro do contexto de funcionalidades, seguindo outras perspectivas, pôde-se destacar a seguinte observação:

> "Priorizei as funcionalidades partindo da minha percepção de complexidade de desenvolvimento e do que poderia ser contemplado no MVP do Software." (Analista Técnico)

Sob esta ótica, a prioridade das funções deve ser um ponto a ser considerado mediante a complexidade de seu desenvolvimento e ao MVP do software. De acordo com o Zendesk (2022), o MVP é uma sigla que significa "Produto Mínimo Viável". Ele é a versão mais simples de um produto que apresenta suas funcionalidades mais básicas. Segundo a supracitada plataforma, o MVP é criado para testar a aderência e aplicabilidade de um produto no mercado, antes de adicionar itens mais complexos e de custo mais elevado. Ele funciona como um teste para validar a ideia e, assim, evitar que se perca tempo e recursos financeiros em uma ideia que não ganhará a aderência de seu público. Além disso, ajuda a definir uma direção para o desenvolvimento do produto, incluir melhorias e ajustes para que seu lançamento seja bem-sucedido.

Portanto, utilizando-se oportunamente destas definições e da observação dada pelo entrevistado, correlacionando-as, frisa-se que este trabalho possui as características de um MVP, uma vez que objetiva a ser uma ferramenta na gestão de reservas de veículos de uma frota corporativa, trazendo funções elencadas como prioritárias, nos padrões ditos anteriormente.

Este trabalho também vislumbra a abordagem multiplataforma, e, para um dos entrevistados, a responsividade do sistema, se utilizado em uma abordagem web, deve ser considerada, inclusive numa abordagem web, conforme a seguinte contribuição:

> "Considero que um app mobile poderia ser uma estratégia vantajosa. No entanto, no cenário em que o público-alvo abrange usuários que realizam reservas de veículos de forma ocasional, é possível a implementação de uma plataforma web, especialmente se tiver uma interface responsiva semelhante à de um app mobile, tende a ser mais propícia e amplamente adotada pelos usuários." (Analista e Desenvolvedora de Sistemas)

Segundo estes apontamentos, adotar uma abordagem web para a utilização e exibição do software pode ser atrativo para aqueles que realizariam reservas de forma esporádica. Além disso, sugere-se que a responsividade da interface, semelhante à experiência oferecida por aplicativos mobile, pode contribuir para a usabilidade do sistema. Nesse contexto, é relevante observar que o Flutter dispõe de recursos que possibilitam a adaptação do código desenvolvido para atender a essa demanda de interface responsiva.

Continuando a análise abordada neste tópico, uma consideração adicional, apresentada por um dos entrevistados, cita contextos de resguardo que podem ser subsidiados pelo desenvolvimento proposto:

> "Todos os tópicos são importantes, tanto pela gestão e controle, pois são importantes para o gestor saiba quem utilizou os veículos, [sendo útil] para resguardar a empresa por possíveis multas de trânsito [...]" (Assistente Administrativo)

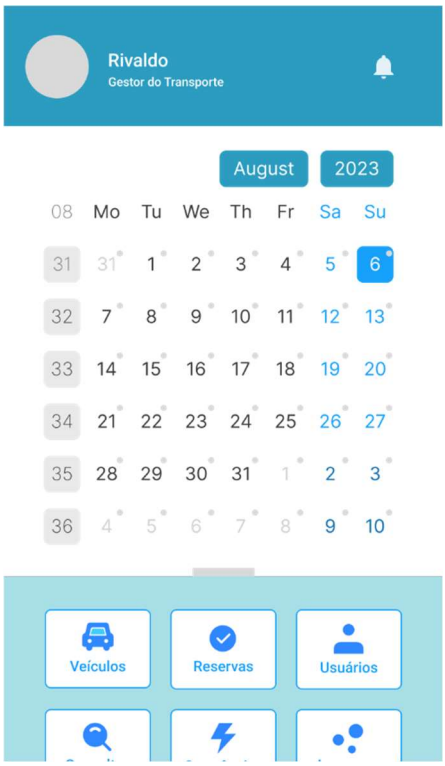
O profissional ressaltou a importância do desenvolvimento dessas funções para a gestão eficaz, controle da frota e identificação dos condutores. Além disso, destacou que essas funcionalidades desempenham um papel na proteção da empresa, podendo ser essenciais em processos que envolvem a cobrança de infrações de trânsito.

Assim, com base na análise abrangente das respostas obtidas, este projeto delineou o escopo a ser abordado. Alinhado com os objetivos definidos, a pesquisa progrediu para a elaboração da arquitetura por meio da configuração do banco de dados em nuvem, a organização do código e o desenvolvimento da aplicação, contendo as funcionalidades de "Realizar Check-in", "Realizar Check-out", exibição dos Status dos Veículos e Reservas, e Cadastro de Reservas.

4.2 Protótipo

Como dito no capítulo anterior no tópico 3.2.3, o nome "Linkar" foi atribuído ao protótipo. Por meio dele foi possível imaginar a organização dos componentes, correções de cores e funcionalidades que foram implementadas. A figura a seguir exibe o protótipo da tela do gestor.

Figura 7: Linkar: Protótipo - Tela Inicial



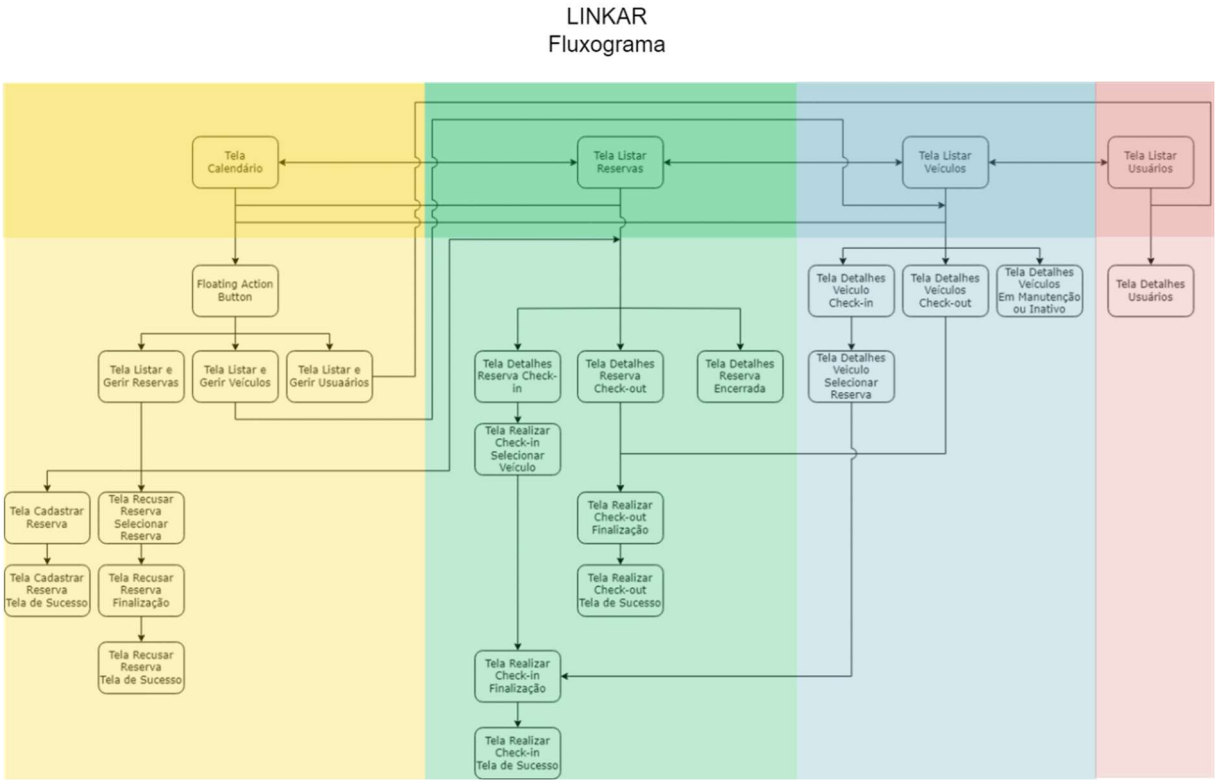
Frisa-se que, na implementação, a paleta de cores para o tema foi alterada para uma coloração arroxeada. Fluxos e deslizamentos também foram implementados, reorganizando o aplicativo, diferenciando-se mais uma vez do que fora inicialmente imaginado. Contudo, a ideia de um calendário dinâmico e interativo fora preservada, bem como toda a lógica dos fluxos definidos. Nesta oportunidade, é válido dizer que nome “Linkar” foi mantido durante a implementação.

Para que a discussão abordada neste capítulo não se torne repetitiva, considerando que a implementação manteve a ideia proposta pelo protótipo em sua essência, respeitando as definições realizadas pela análise das respostas do questionário e que, ao decorrer deste estudo, consta-se a explicação dos processos do desenvolvimento do aplicativo, ao que tange aos resultados obtidos pelo protótipo, a análise prosseguirá para a explicação das rotas e fluxos definidos por meio dele.

4.3 Fluxo das Rotas Definidas

Conforme dito anteriormente, o protótipo proporcionou a visualização das rotas definidas desenvolvidas neste trabalho[1]. Após a conclusão da fase de prototipação, considerando o quantitativo de rotas e interações entre as telas, o seguinte fluxograma foi desenvolvido:

Figura 8: Linkar: Fluxograma Simplificado



A primeira tela carregada pelo sistema é subdivida em 4 seções: Tela Calendário, Tela Listar Reservas, Tela Listar Veículos e Tela Listar Usuários. Considerando a figura anterior, a área amarela traz a ideia simplificada de fluxos partindo da seção Tela Calendário, considerando a utilização de um botão flutuante de ação. Este botão funciona como um menu e leva a outras telas que são destinadas a tratar questões especificadas conforme o objeto selecionado, sendo elas a Tela Gerir reservas, Tela Gerir Veículos e a Tela Gerir Usuários. Estas telas são destinadas a conter funções de recusa e cadastros de reservas, por exemplo.

[1] A visualização detalhada de todos os caminhos e fluxos definidos após a implementação, contendo a captura de cada tela do software, pode ser acessada por meio do link: https://www.figma.com/file/QP2K46iTNfWfQMFcth64Nr/Linkar-Fluxograma-TCC---Ci%C3%AAncia-da-Computa%C3%A7%C3%A3o-Ceulp-Ulbra?type=whiteboard&node-id=0%3A1&t=OTqLq1sPZrdhn8CH-1.

A área verde exibe as rotas entre as telas considerando a Tela Listar Reservas. Esta tela lista as reservas e leva a outras telas voltas ao detalhamento de reservas que trarão informações destrinchadas de cada reserva selecionada.

Já os fluxos a partir da Tela Listar Veículos estão destacadas na área azul, que por sua vez levará a tela de detalhamento dos veículos. Por fim, a área rosa traz o fluxo da Tela Listar Usuários, que, conforme o padrão adotado, levará a tela de detalhamento de usuários.

Após todas estas análises e fluxos, o projeto seguiu para sua implementação. Inicialmente, partiu-se para a criação e configuração do banco de dados. Após, estabeleceu-se a organização do código, a conectividade com o banco por meio da API do Supabase e então o desenvolvimento de cada tela no framework Flutter.

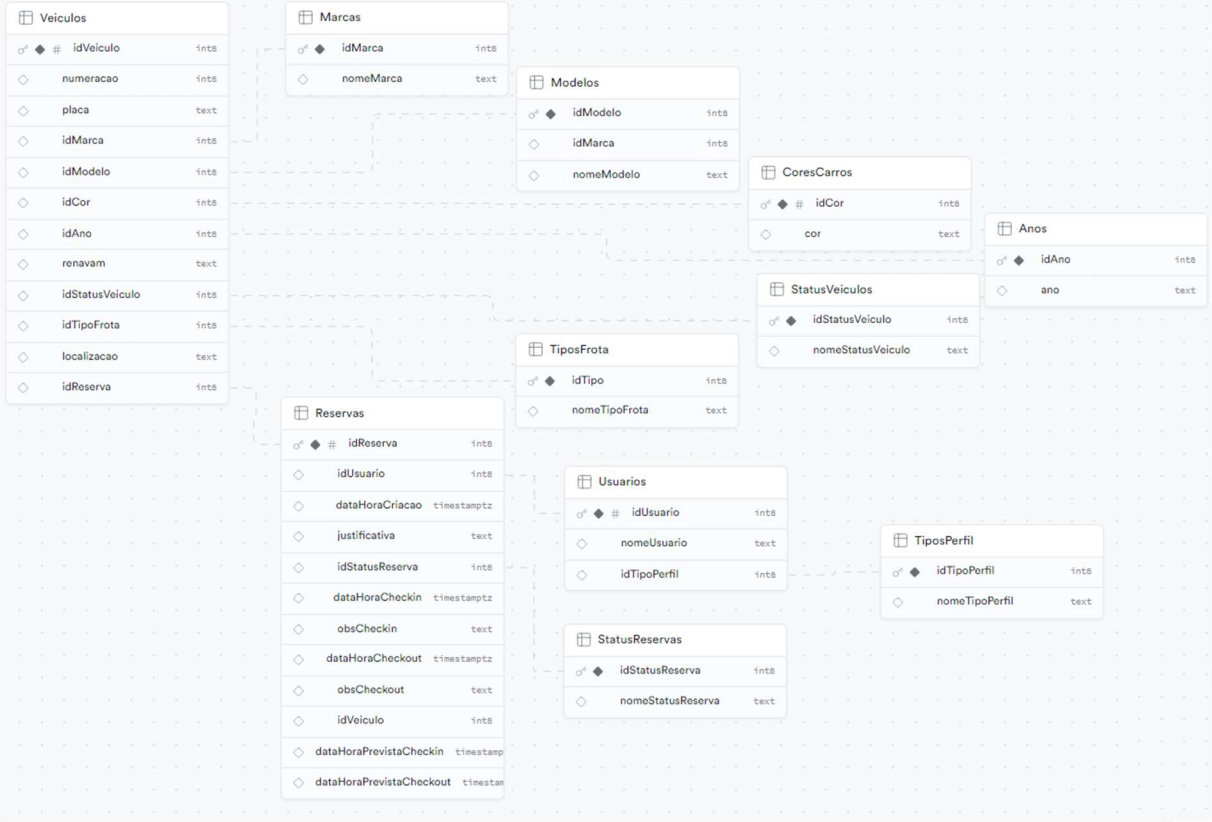
4.4 Criação e Configuração do Banco de Dados

O modelo de banco de dados adotado para o sistema “Linkar” foi estruturado com tabelas que refletem as entidades essenciais do domínio, oferecendo um entendimento organizado e coeso. A arquitetura planejada visa proporcionar uma representação fiel e eficiente das interações complexas no contexto de um sistema de reserva de veículos.

A criação e configuração do banco de dados no Supabase foram realizadas por meio da própria ferramenta de dashboard oferecida aos usuários do serviço. Dentre as diversas funcionalidades disponíveis, destacam-se o *Table Editor*, o *SQL Editor*, *Project Settings* e o *Scheme Visualizer*, este último sendo encontrado na seção dedicada ao *Database*.

As tabelas foram estruturadas e alimentadas com informações padronizadas quanto às características dos veículos, como tipo de frota, ano, cor, marca, modelo, status da reserva e do veículo, além de tipos de perfis de usuários. Os relacionamentos entre as tabelas visam oferecer uma representação organizada das interações complexas no contexto de um sistema de reserva de veículos, conforme ilustração a seguir:

Figura 9: Supabase - Relacionamento entre Tabelas



Quanto à abordagem que permite categorizações e a recuperação de informações quanto a identidade dos veículos, cita-se como exemplo o relacionamento entre a tabela “Veiculos” e todas as demais tabelas que fazem referência a características dos veículos, sendo elas as tabelas “Anos”, “Marcas”, “Modelos”, “TiposFrota” e “CoresCarros”. Através das chaves estrangeiras, é possível estabelecer uma ligação direta entre um veículo específico e seu ano de fabricação, sua marca, modelo, tipo de frota e cor correspondentes.

Algo semelhante ocorre na relação entre a tabela “Veiculos” e a tabela “StatusVeiculos”, que permite monitorar o estado operacional dos veículos, com a chave estrangeira idStatusVeiculo. Dessa forma, é possível realizar a classificação dos veículos em diferentes padronizações, sendo elas “Disponível”, “Em Utilização, “Em manutenção” ou “Inativo”.

Outro ponto de destaque é o relacionamento entre a tabela “Veiculos” e a tabela “Reservas” por meio da chave estrangeira idReserva, permitindo associar um veículo a uma reserva específica. Desse modo, é possível estabelecer uma conexão direta entre a disponibilidade do veículo e as solicitações de reserva por parte dos usuários.

Esclarece-se que o veículo não é associado diretamente ao usuário. Entretanto, é possível rastrear as atividades de reserva associadas a cada usuário específico, pois a chave primária idUsuario na tabela “Usuarios” é referenciada como chave estrangeira na tabela

“Reservas”. Isto conecta indiretamente os veículos associados aos usuários que efetuaram as reservas.

Sob outro aspecto, a tabela “Usuarios” armazena informações sobre os usuários e estabelecendo relacionamento com a tabela “TiposPerfil” através da chave estrangeira idTipoPerfil. Essa conexão permite a classificação dos usuários de acordo com seus perfis, sendo útil para a atribuição de permissões e funcionalidades específicas a cada tipo de usuário, contribuindo para a segurança e a personalização da experiência no sistema.

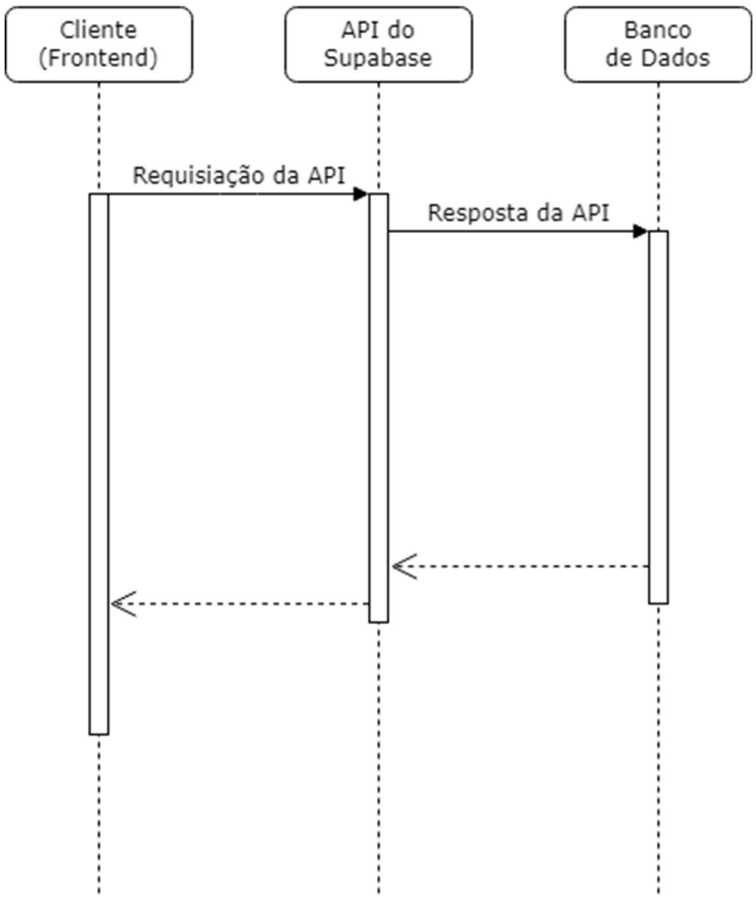
Deste modo, os relacionamentos interligados entre “Veiculos”, “Reservas” e “Usuarios” criam uma estrutura organizada e funcional, permitindo consultas complexas e operações integradas no sistema. A modelagem proposta demonstra uma abordagem cuidadosa para garantir a integridade referencial e oferecer flexibilidade na gestão de dados relacionados a veículos, reservas, usuários e perfis, de modo que termos de tipagem, cores, anos, marcas e modelos, e outras descrições possam estar padronizados.

Não foram realizadas configurações ou automações de inserção, atualização, conversão e validação de dados. Estas operações são gerenciadas por funções no código em Flutter.

4.5 API do Supabase

Para este trabalho, o banco de dados foi configurado para permitir a conexão e o acesso aos dados de forma simplificada, dispensando autenticação de usuários, conforme as configurações da própria API do Supabase que fornecesse URL e chave anônima geradas automaticamente. O diagrama a seguir ilustra a sequência das requisições:

Figura 10: API do Supabase - Diagrama de Sequência



Portanto, os dados são acessados publicamente de forma anônima, não sendo necessário realizar demais configurações de *backend*. Menciona-se que a abordagem de leitura pública de forma anônima só será efetivada se a tabela também permitir a leitura anônima de seus dados. Estas práticas não seriam aconselhadas caso houvesse a intenção de restringir mais rigidamente o acesso aos dados no banco, conforme explicitado pela própria plataforma.

Finalizadas as devidas configurações no banco de dados, é possível partir para a implementação do código propriamente dito na linguagem e framework desejados. E no caso de se utilizar o banco de dados em desenvolvimento Flutter de forma integrada, é preciso inicialmente instalar o pacote da biblioteca Supabase no ambiente Flutter, conforme orientações oficiais (SUPABASE.IO, 2023).

4.6 Estrutura e implementação do Aplicativo

A estrutura adotada por esta pesquisa reflete uma abordagem de desenvolvimento organizada e orientada a objetos, onde classes são utilizadas para representar entidades do

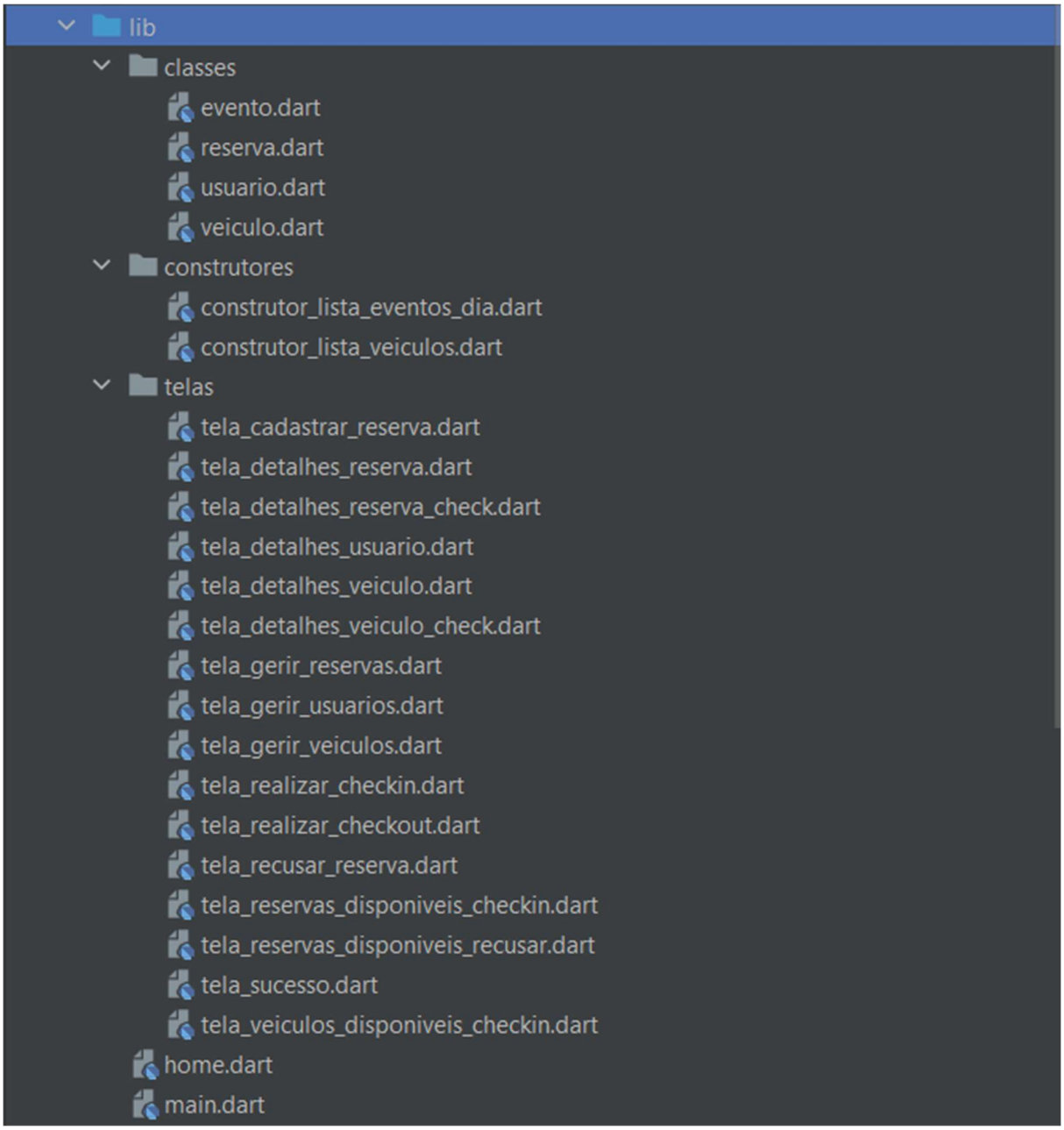
mundo real. Considerando que este projeto se concentra exclusivamente no módulo de gestão da frota veicular, o código desenvolvido aborda especificamente questões lógicas relacionadas ao gerenciamento de reservas e disponibilidade de veículos para que o usuário gestor da frota de veículos esteja assessorado em suas atribuições.

4.6.1 Organização do Código

No que diz respeito à organização do código, foram criados dois arquivos Dart no diretório principal "\lib" do Flutter: "main.dart" e "home.dart". Adicionalmente, foram estabelecidos três diretórios: "\classes", "\construtores" e "\telas". O arquivo "main.dart" contém as configurações do aplicativo, juntamente com as funções principais acessadas ao longo da execução do sistema. Já o arquivo "home.dart" concentra as lógicas da primeira tela, exibida imediatamente após o carregamento inicial e a validação da conexão.

Os mencionados diretórios contêm arquivos Dart dedicados à estruturação das classes, à definição de construtores específicos e à implementação das demais telas secundárias. Além disso, essas telas abrigam a criação de algumas funções específicas. A distribuição destes arquivos pode ser verificada a seguir:

Figura 11: Estruturação dos Arquivos



O diretório “\classes” guarda o desenvolvimento das classes de objetos fundamentais para o funcionamento deste sistema e as tratativas de informações e questão correlacionadas com os dados guardados pelas tabelas do banco. O diretório “\construtores” guarda o desenvolvimento em Dart de componentes específicos trabalhados na construção da seção “Calendário” da tela inicial. As telas a serem carregadas de acordo com as interações do usuário gestor foram guardadas no diretório “\telas”. Elas serão acessadas após a execução do arquivo “home.dart”.

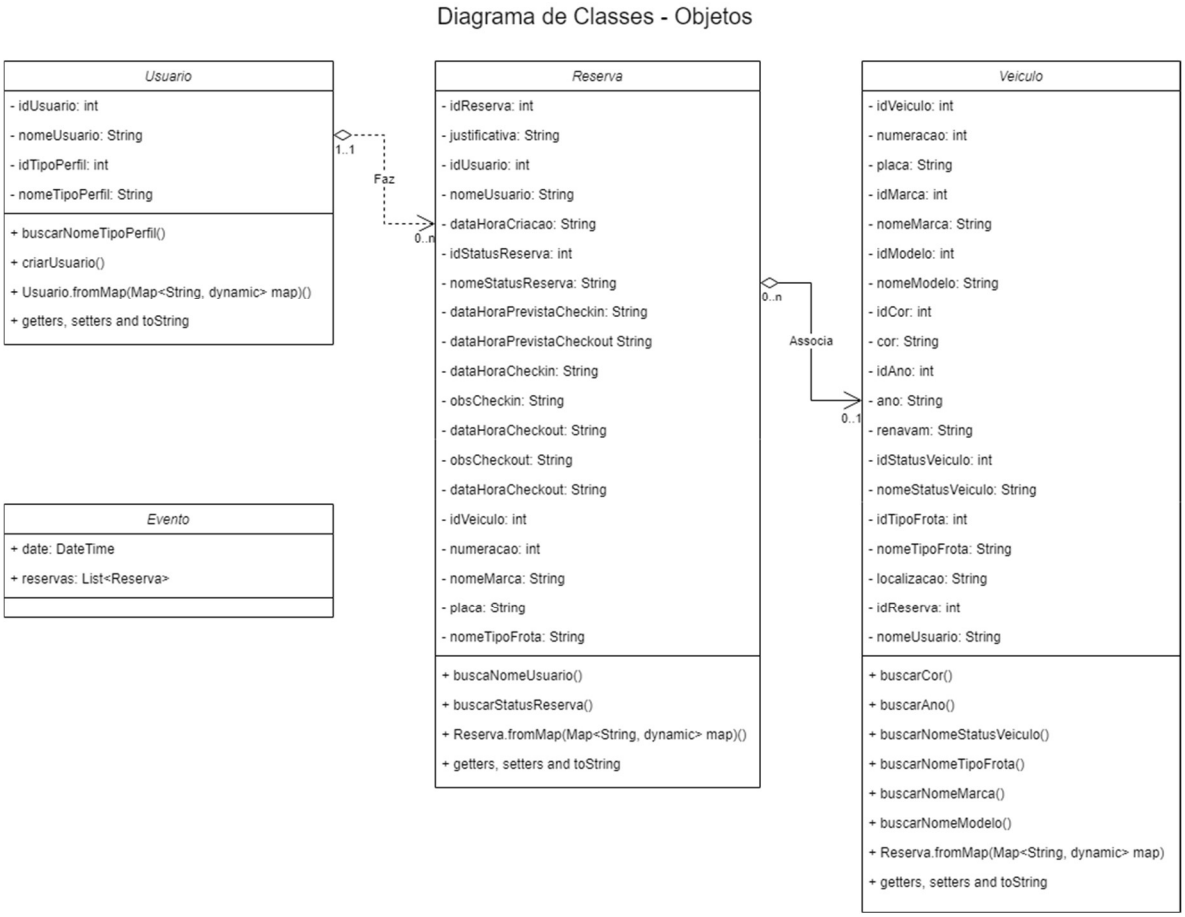
Essa abordagem pôde facilitar a manutenção do código, uma vez que as responsabilidades são distribuídas de forma lógica entre os diferentes arquivos e diretórios. A separação de algumas estruturas e funcionalidades pode contribuir para uma compreensão mais intuitiva e eficiente do sistema, promovendo um desenvolvimento modular e escalável.

4.6.2 Definição de Classes

Com base na organização de toda a estrutura, inicialmente foram delineadas as classes de objetos que se correlacionam as informações armazenadas no banco, bem como as funções responsáveis por manipular essas informações dentro do código. Assim, visando uma potencial simplificação, dado que este trabalho focou na modelagem das relações entre reservas e veículos, foram concebidas quatro classes: “Evento”, “Reserva”, “Usuario” e “Veiculo”.

Para uma representação visual, foi elaborado o Diagrama de Classes que reflete a implementação em Fluter. O diagrama pode ser visualizado logo abaixo:

Figura 12: Linkar - Diagrama de Classes de Objetos



Conforme evidenciado no diagrama, a classe “Evento” não se relaciona diretamente com as demais classes. Contudo, ela é fundamental para utilização dentro do componente

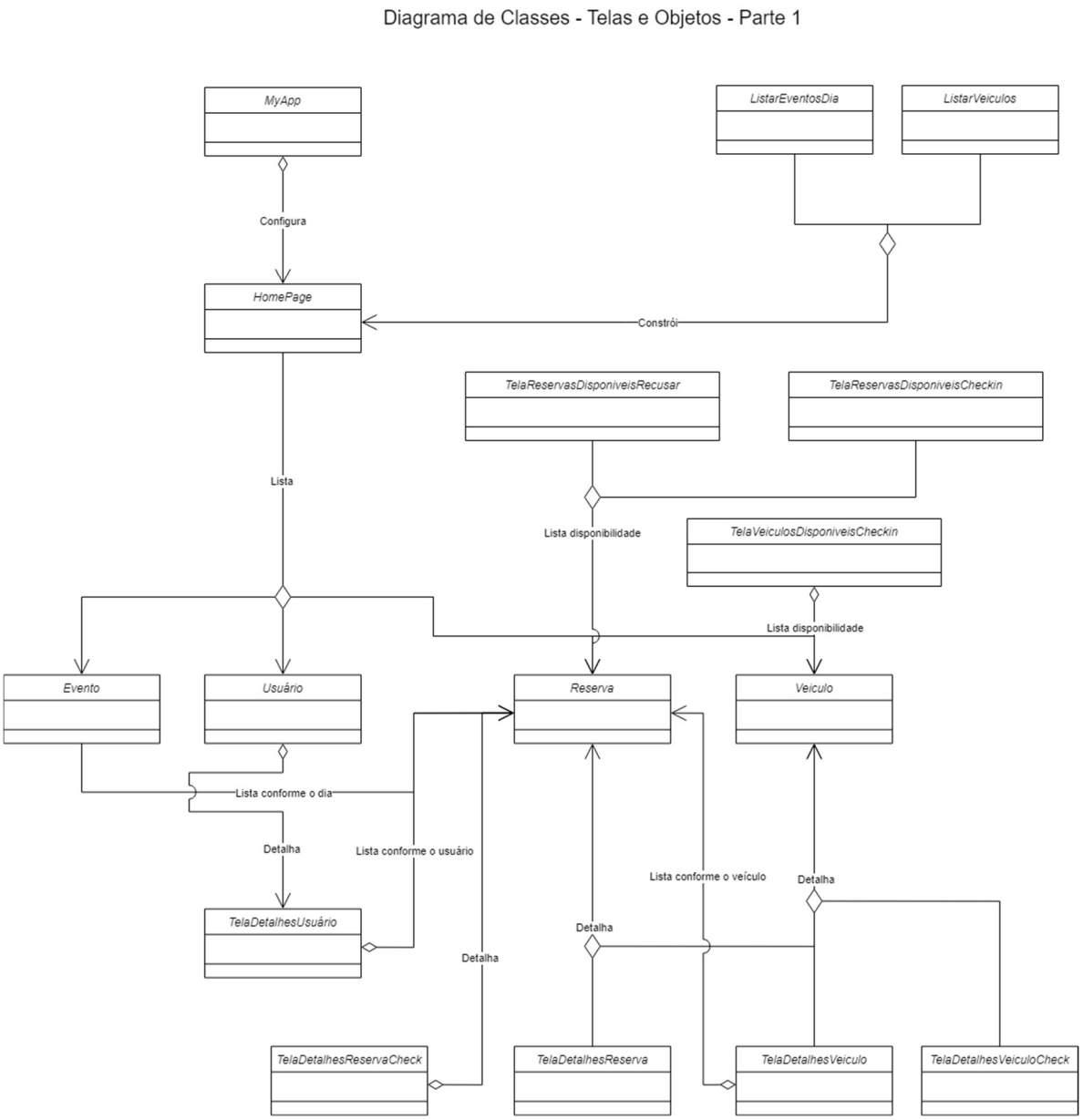
TableCalendar que a utiliza para organizar e referenciar os eventos conforme o dia selecionado, permitindo sua tratativa, que no caso deste projeto, visa sua visualização de acordo com o dia selecionado e ações de gestão.

A modelagem das classes define atributos privados inicializados por meio de um construtor, promovendo encapsulamento e contribuindo para a coesão das classes. Destaca-se que os valores podem ser acessados, alterados e impressos por meio dos conhecidos *getters*, *setters* e do método toString. No contexto das funções de cada classe, para viabilizar a construção dos objetos, o código prevê a criação de instâncias a partir de mapas, utilizando o método fromMap. Este método tem como objetivo mapear dados provenientes de fontes externas, como no caso dos bancos de dados adotados nesta pesquisa.

As funções de busca, tais como “buscarNomeUsuario” e “buscarNomeStatusReserva”, foram desenvolvidas visando à integração com o banco de dados por meio da biblioteca Supabase. Essas funções acessam dados relacionados a IDs em outras tabelas, oferecendo uma abordagem modular para a obtenção de informações específicas dentro das classes.

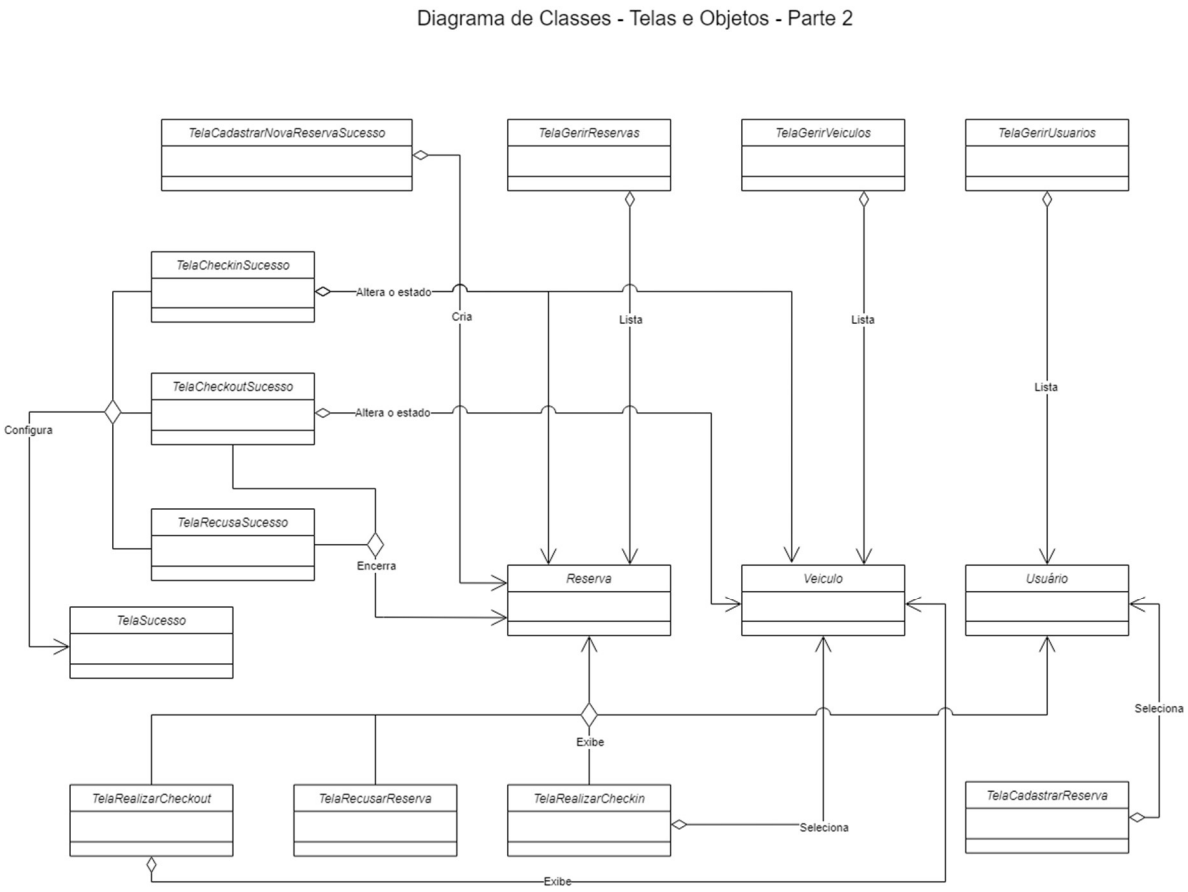
Quanto às demais classes, elas se referem a componentes do próprio Flutter, responsáveis pela construção das telas e destinam-se ao tratamento de toda a estrutura lógica e organizam as exibições e fluxos, abrangendo desde configurações de customização, listagens de objetos e são responsáveis também pela construção de outros componentes. A seguir, a primeira parte da implementação das relações destes construtores de tela, partindo das execuções da tela inicial mediante as funcionalidades e relacionamentos com as demais telas construídas pelos seus próprios componentes:

Figura 13: Linkar - Diagrama de Classes de Objetos de Telas e Objetos Parte 1



Quanto as demais classes dos objetos de telas, seus relacionamentos estão voltados à consumação das ações do aplicativo no que tange a realização de Check-in, Check-out, gestão de reservas, veículos e usuários, bem como o cadastro de reservas, encerramento de reservas por recusa e o retorno da finalização de lançamentos que alteram o estado dos objetos reserva e veículos. Estas relações podem ser verificadas no diagrama a seguir:

Figura 14: Linkar - Diagrama de Classes de Objetos de Telas e Objetos Parte 2

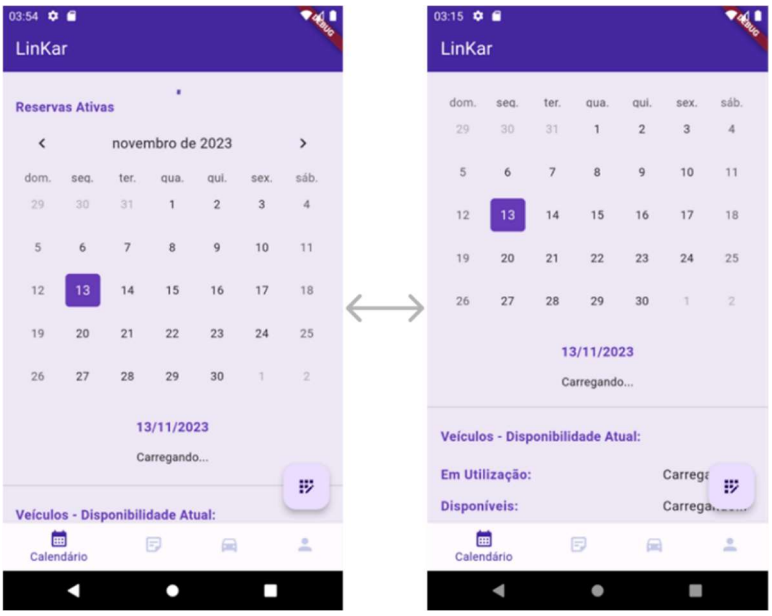


4.7 Implementação de Telas de Funcionalidades

Nesta seção, são detalhados os fluxos e as funcionalidades das telas que foram implementadas, abordando as ações de cada componente essencial. A discussão foi organizada de acordo com as funções vinculadas a sua tela correspondente e as rotas oriundas destas funcionalidades.

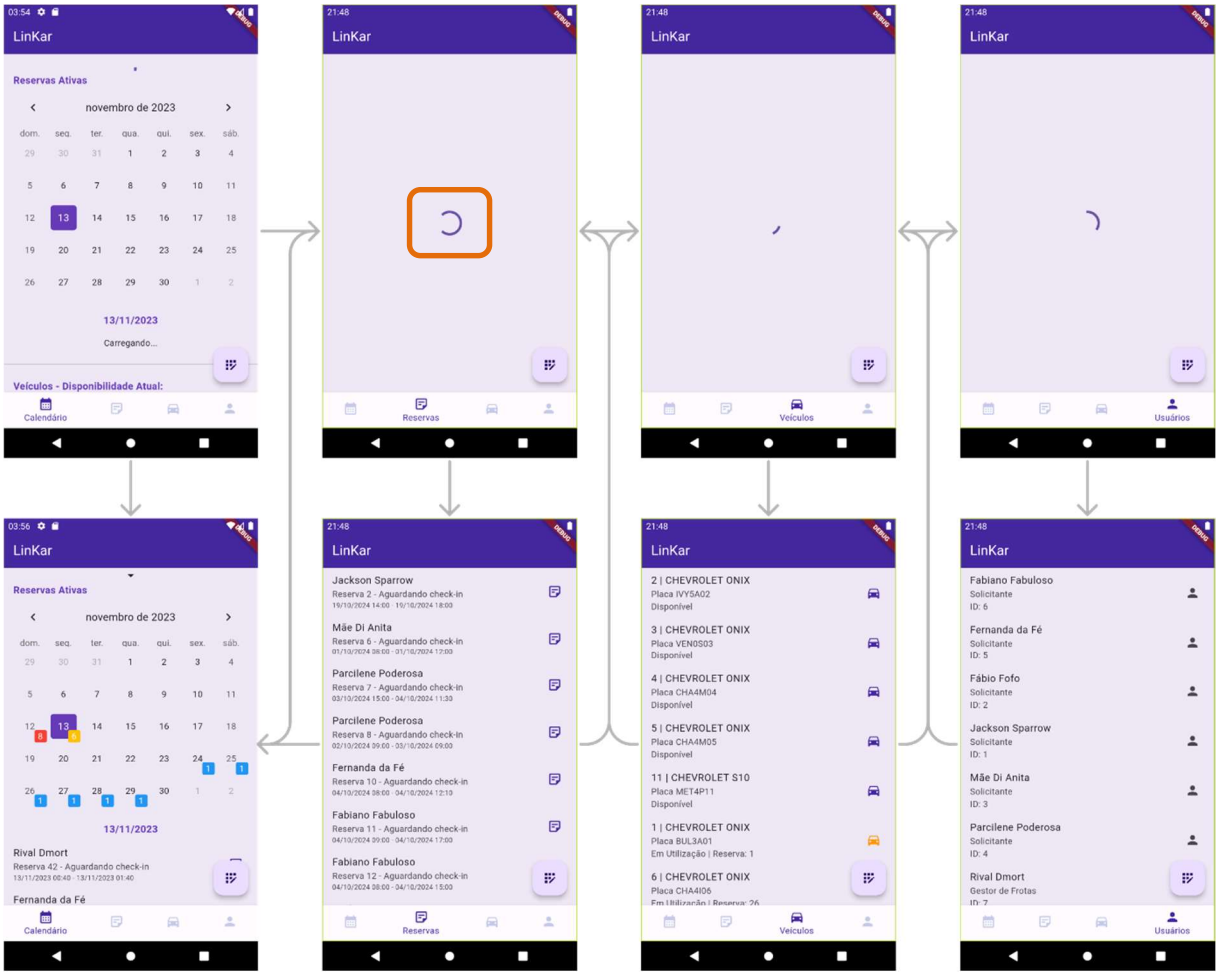
Dessa forma, o aplicativo inicia carregando a Tela Principal, composta por quatro seções designadas como "Calendário", "Reservas", "Veículos" e "Usuários", além de um botão flutuante que exibe o ícone definido no Flutter como Icons.app_registration. Por padrão, o estado inicial da tela está na seção "Calendário", como ilustrado a seguir:

Figura 15: Linkar - Tela Inicial



No que se refere à navegação entre as suas seções, os usuários podem alternar entre elas deslizando lateralmente na tela ou selecionando um dos ícones disponíveis na barra inferior, conforme o fluxo abaixo:

Figura 16: Linkar - Fluxo da Tela Inicial



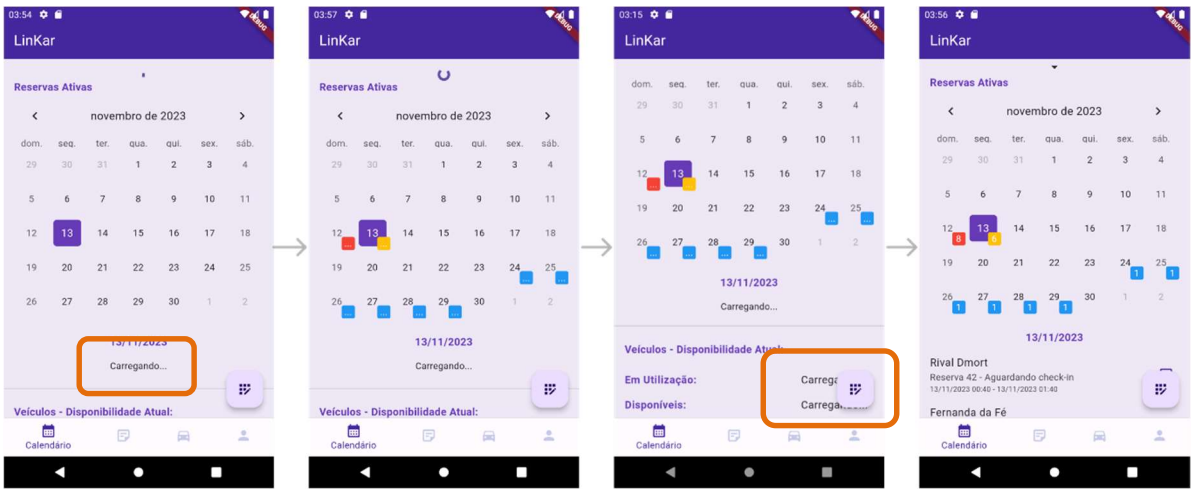
Exceto pela seção "Calendário", sempre que a seção é alterada para as demais, ocorre automaticamente uma requisição ao banco de dados. Durante esse processo, o CircularProgressIndicator, destacado acima, é exibido, indicando que a consulta está em andamento.

Esse procedimento garante que os objetos estejam sempre atualizados, prevenindo erros nos fluxos subsequentes. Após a conclusão dessa consulta e organização dos objetos de acordo com a seção selecionada, a tela exibe a listagem completa de Reservas, Veículos ou Usuários.

4.7.1 Tela Calendário

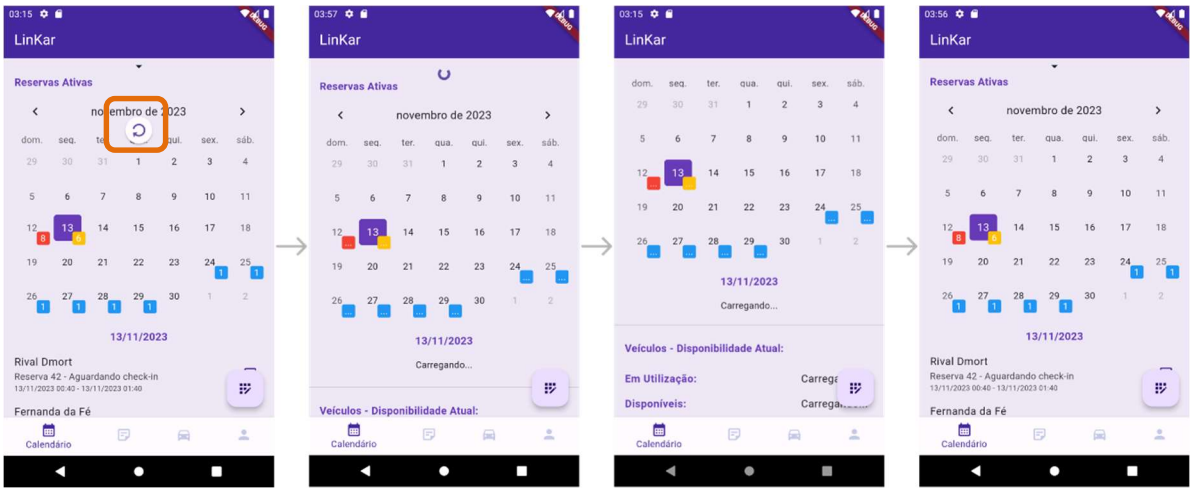
Esta seção, ao ser iniciada pela primeira vez, realiza consulta ao banco de dados para apresentar as informações de reservas e veículos, as quais serão listadas logo abaixo do calendário. Durante a espera pela conclusão da requisição, a mensagem "Carregando...", é exibida, conforme destaque laranja e ilustração a seguir**:**

Figura 17: Linkar - Tela Calendário Carregamento



Se o usuário realizar o gesto de deslizar para baixo, do topo em direção ao final da tela, um ícone representativo será exibido, indicando a possibilidade de atualização. Ao completar o deslizamento, a tela realizará uma nova consulta ao banco de dados, e os componentes serão atualizados, como mostrado na figura a seguir, com destaque para o ícone da atualização:

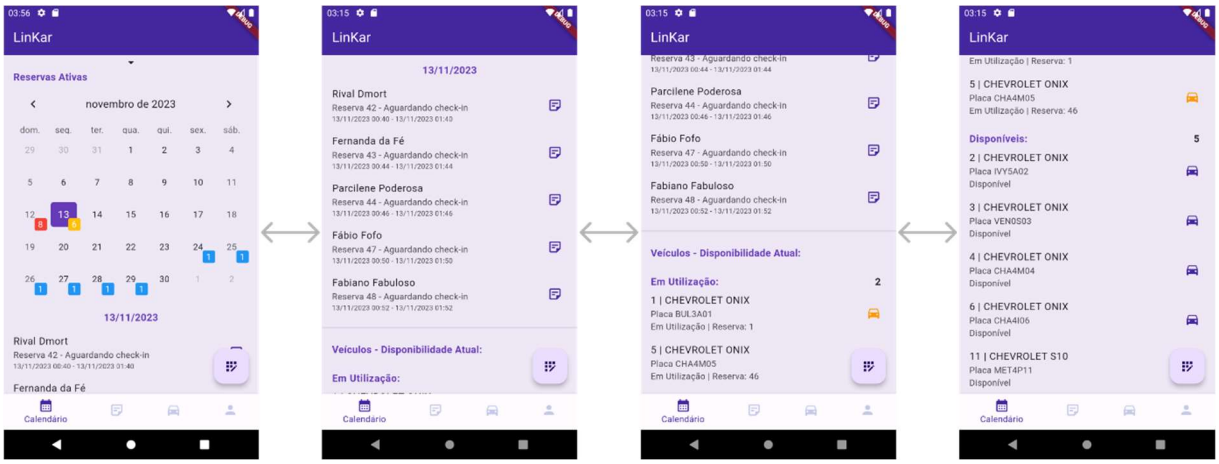
Figura 18: Linkar - Tela Calendário Atualização

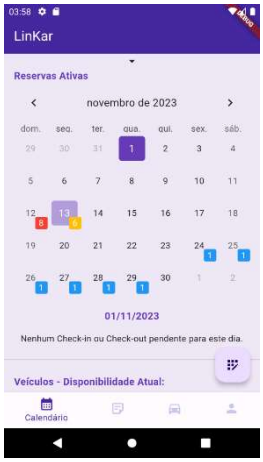


A seção "Calendário" é subdividida em áreas principais. Inicialmente, ela apresenta o calendário que exibe a quantidade de reservas pendentes de encerramento. Logo abaixo, encontra-se a lista destas reservas, de acordo com sua programação mediante o dia selecionado no calendário, não considerando aquelas que já se encontram encerradas. Caso não haja reservas ativas que envolvam aquele dia, a lista não será carregada e em seu lugar será exibido o texto “Nenhum Check-in ou Check-out pendentes para este dia”. Por fim, a tela exibe a lista de veículos ativos que não estão em manutenção.

Ressalta-se que durante a execução da consulta ao banco de dados, as listas aguardam para serem construídas. Após a conclusão da requisição, ao se rolar a tela, as listagens são exibidas, conforme a seguinte representação da rolagem de tela:

Figura 19: Linkar - Tela Calendário Informações Carregadas





O carregamento das reservas presentes no banco de dados é iniciado por meio da função _carregarEventosReservas(). Sua execução acontece no início do carregamento da tela, acionando a função buscarEvenosReservas(), conforme o trecho logo abaixo:

Figura 20: Linkar - Trecho de Código Função _carregarEventosReservas()

```
void initState() {
[...]
  _carregarEventosReservas();
[...]
Future<void> _carregarEventosReservas() async {
  buscarEventosReservas().then((novosEventos) {
    if(mounted) {
      setState(() {
        eventos = novosEventos!;
        eventosCarregando = false;
      });
    }}
  );
}
```

A função buscarEventosReservas() tem como objetivo buscar e organizar eventos de reservas em um formato apropriado para exibição no calendário. A implementação abrange a transformação de reservas em uma lista para posterior manipulação, a verificação do status da reserva e a criação de eventos correspondentes, considerando as datas de Check-in e Check-out. Seu trecho de código pode ser visualizado logo a seguir:

Figura 21: Linkar - Trecho de Código Função buscarEventosReservas()

```
Future<Map<DateTime, List<Evento>>?> buscarEventosReservas() async {
  Map<DateTime, List<Evento>> novosEventos = {};

  final reservas = await transformarReservaToListCalendario();
  for (Reserva reserva in reservas) {
    if (reserva.idStatusReserva == 1 || reserva.idStatusReserva == 2) {
      String checkinString = reserva.dataHoraPrevistaCheckin;
      String checkoutString = reserva.dataHoraPrevistaCheckout;

      DateTime checkin = DateTime.parse(checkinString);
      DateTime checkout = DateTime.parse(checkoutString);

      checkin = checkin.toLocal();
      checkout = checkout.toLocal();

      int diasDeUtilizacao = checkout.difference(checkin).inDays;

      for (int i = 0; i <= diasDeUtilizacao; i++) {
        DateTime dataEvento = checkin.add(Duration(days: i));
        dataEvento =
            DateTime.utc(dataEvento.year, dataEvento.month,
dataEvento.day);

        if (novosEventos[dataEvento] == null) {
          novosEventos[dataEvento] = [
            Evento(date: dataEvento, reservas: [reserva])
          ];
        } else {
          Evento eventoExistente = novosEventos[dataEvento]!.firstWhere(
            (evento) => evento.date == dataEvento,
            orElse: () => Evento(date: dataEvento, reservas: []),
          );
          eventoExistente.reservas.add(reserva);
        }
      }
    }
  }
  return novosEventos;
}
```

A função inicia criando um mapa vazio chamado novosEventos para armazenar eventos de reservas. Em seguida, utiliza a função assíncrona transformarReservaToListCalendario() para obter a lista de reservas. Para cada reserva, é verificado se seu status é 1 (Aguardando Check-in) ou 2 (Aguardando Check-out). Caso positivo, as datas previstas com período de utilização são convertidas de strings para objetos DateTime e ajustadas para o fuso horário local.

Com as datas convertidas, o código calcula a quantidade de dias de utilização da reserva. Em seguida, um laço percorre cada dia, criando um objeto DateTime para representar a data do evento. Essa data é normalizada para o fuso horário *Coordinated Universal Time* (UTC). O código verifica se já existe um evento para essa data no mapa novosEventos. Se não existir, um

novo evento é criado com a reserva associada. Se já existir, a reserva é adicionada ao evento correspondente.

Dessa forma, o código gerencia as informações de reservas armazenadas no banco de dados, organizando-as com base nas datas previstas de utilização, criando objetos da classe "Evento", que vinculam as reservas de acordo com o dia correspondente. Concluídas todas estas etapas, finalizando assim o carregamento que, por sua vez, também envolve a verificação de quais veículos estão ativos, o código exibirá o calendário plenamente construído e carregado, por meio do componente TableCalendar.

Esse componente apresenta diversos recursos, destacando-se neste projeto a seleção de dia, indicação da data atual, marcadores próximos aos dias e a capacidade de lidar com objetos referenciando-os a datas. A estrutura principal está encapsulada em um Container, fornecendo margens e centralizando o conteúdo, conforme o trecho a seguir:

Figura 22: Linkar - Trecho de Código Encapsulamento do TableCalendar

```
Container(
  margin: EdgeInsets.fromLTRB(16, 0, 16, 16),
  child: Center(
    child: TableCalendar(...)
```

O componente permite a definição do intervalo de datas visíveis, especificando o primeiro e o último dia. Também permite tratativas quanto sua capacidade de pular entre páginas do calendário, que no caso foi desabilitada (pageJumpingEnabled: false). Para que ele seja exibido em português, sua localização deve ser definida para 'pt_BR'. Além disso, o código controla o dia focado (focusedDay) e a lógica de atualização quando o usuário muda de página (onPageChanged). Essas configurações formam a base para a interatividade e funcionalidade do calendário, conforme o trecho a seguir:

Figura 23: Linkar - Trecho de Código Customização do TableCalendar Parte 1

```
TableCalendar(
      pageJumpingEnabled: false,
      firstDay: DateTime.utc(2000, 1, 1),
      lastDay: DateTime.utc(2099, 12, 31),
      focusedDay: focusedDay,
      locale: 'pt_BR',
      headerStyle: HeaderStyle(...),
      calendarFormat: format,
      onPageChanged: (_focusedDay) {
        focusedDay = _focusedDay;
      },
      startingDayOfWeek: StartingDayOfWeek.sunday,
      daysOfWeekVisible: true,

[...]
)
```

Este calendário permite a manipulação de eventos associados a datas específicas, como por exemplo a possibilidade de se indicar o dia selecionado e o dia que será focalizado. Além disso, o atributo eventLoader é utilizado para se acessar os eventos (as reservas) carregados para o dia selecionado, conforme o trecho a seguir:

Figura 24: Linkar - Trecho de Código Customização do TableCalendar Parte 2

```
[...]
onDaySelected: (DateTime selectDay, DateTime focusDay) {
  setState(() {
    print(selectDay);
    selectedDay = selectDay;
    focusedDay = focusDay;
  });
},
selectedDayPredicate: (DateTime date) {
  return isSameDay(selectedDay, date);
},
eventLoader: _getEventosDoDia,
[...]
```

Com o auxílio da função _getEventosDoDia, o código acessa as reservas que envolvem o dia selecionado mapeadas pelas funções detalhadas anteriormente:

Figura 25: Linkar - Trecho de Código Função _getEventosDoDia()

```
List<Evento> _getEventosDoDia(DateTime date) {
  return eventos[date] ?? [];
}
```

Outro fator do TabelCalendar, é a presença de um construtor personalizado chamado de CalendarBuilders, responsável por exibir marcadores visuais no calendário. Esses marcadores indicam a quantidade de reservas que necessitam de Check-in ou Check-out, conforme o carregamento de eventos outrora executado, proporcionando uma visualização rápida e informativa dos eventos associado. Além disto, esta personalização viabiliza automações como a alteração de cor do marcador que, dependendo da proporção da disponibilidade de veículos ativos, será diferente. As condições que orientam a mudança de cor podem ser visualizadas a seguir:

Tabela 3: Condicionamento de Cores mediante o Quantitativo de Reservas no Dia

Cor	Condição
	Se o número de reservas for menor ou igual a 70% do quantitativo de veículos ativos, a cor retornada ao usuário gestor de frotas é azul (Colors.blue).
	Se o número de reservas for maior que 70% e menor ou igual a 90% do quantitativo de veículos ativos, a cor retornada ao usuário gestor de frotas é âmbar (Colors.amber).
	Se o número de reservas for maior que 90% e menor que o quantitativo de veículos ativos, a cor retornada ao usuário gestor de frotas é laranja profundo (Colors.deepOrange.shade300).
	Se nenhuma das condições acima for atendida, ou seja, se o número de reservas for maior ou igual ao quantitativo de veículos ativos, a cor retornada ao usuário gestor de frotas é vermelha (Colors.red).

Figura 26: Linkar - Tela Inicial Destaque Calendário

Essa estratégia oferece um alerta prático e operacional para o gestor de frotas dentro do sistema. Ao empregar essa tratativa, o usuário gestor recebe indicações visuais instantâneas sobre a ocupação da frota em comparação com o limite estabelecido, permitindo ao gestor a identificação de situações críticas, como o excesso de demanda em contraste com a indisponibilidade veicular e reservas não cumpridas, ou ainda, a necessidade de ações proativas, como alocar recursos adicionais ou recusar solicitações de novas reservas. Essa abordagem prática e visual contribui para a eficiência e tomada de decisões no contexto da gestão da frota.

Ainda sobre esta seção, no que diz respeito à organização da listagem de reservas carregadas pelos eventos, ela segue uma lógica de prioridade. O código leva em consideração os critérios de data de criação, a hora prevista para o Check-in e o estado da reserva. Em outras palavras, além de manter a ordenação refletindo a expectativa da hora de utilização ao longo do dia e de exibir as reservas que aguardam Check-in antes daquelas que já estão em utilização, a

implementação dá preferência à data de criação das reservas. Dessa forma, as reservas cadastradas primeiro têm preferência quando os horários de Check-in são idênticos.

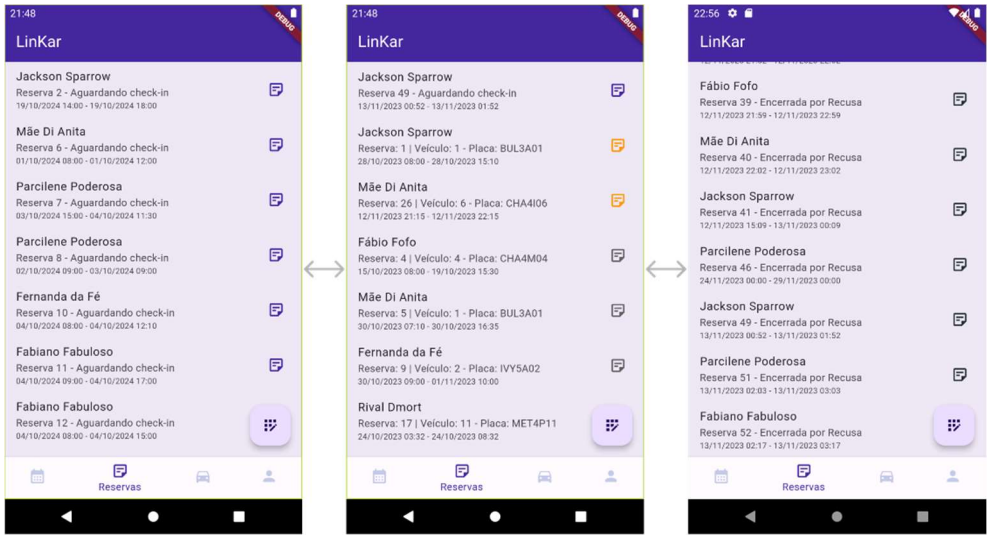
A listagem de veículos considera o estado atual desses objetos, independentemente do dia selecionado, proporcionando a realização de gestão e previsões de utilização ao longo dos dias, subdividindo-os em Veículos Em Utilização e em Veículos Disponíveis, ordenando-os de acordo com sua numeração cadastrada no sistema. Essa listagem do estado atual dos veículos também indica quais reservas estão associadas a eles, quando aplicável, posicionando-os próximos das reservas em utilização, facilitando a identificação de vínculos. Veículos em manutenção e inativados não são exibidos.

Mediante ao clique nos objetos das listas, o usuário gestor é direcionado à tela de detalhamento destes objetos. Estes detalhamentos serão abordados logo mais adiante.

4.7.2 Tela Reservas

Após o carregamento da tela, a listagem completa das reservas é apresentada, seguindo critérios de ordenação baseados no estado da reserva e em seu identificador. Desta maneira, as reservas são organizadas, priorizando aquelas que aguardam a realização do Check-in, seguidas por aquelas que aguardam a realização de Check-out, as que foram encerradas e, por fim, aquelas que foram encerradas por recusa. Dentro de cada estado de reserva, a ordenação é feita de acordo com o identificador, proporcionando uma visualização estruturada, conforme representado abaixo:

Figura 27: Linkar - Tela Reservas



A coloração dos ícones reflete o estado da reserva, mantendo um padrão em todo o código sempre que uma lista de reservas é exibida. As condições que determinam a aplicação das cores nos ícones podem ser verificadas a seguir:

Tabela 4: Condicionamento de Cores mediante o Estado de Reservas

Cor	Condição
	Se o estado da reserva for igual a Aguardando Check-in, a cor roxa será aplicada (Colors.deepPurple[800])
	Se o estado da reserva for igual a Aguardando Check-out, a cor laranja será aplicada (Colors.orange)
	Se o estado da reserva for igual a Aguardando Check-in, a cor cinza será aplicada (Colors.grey[700]).
	Se o estado da reserva for igual a Aguardando Check-in, a cor azul-acinzentado será aplicada (Colors.blueGrey[900]).
	Para qualquer outro valor de estado da reserva, a cor aplicada preta será aplicada (Colors.black).

Mediante ao clique nos objetos da lista, o usuário gestor é direcionado à tela de detalhamento destes objetos. O detalhamento traz a informações detalhadas das reservas, como Data de Criação, Justificativa e Veículo Associado e permite a realização de Check-in ou Check-out, conforme for o caso de seu estado.

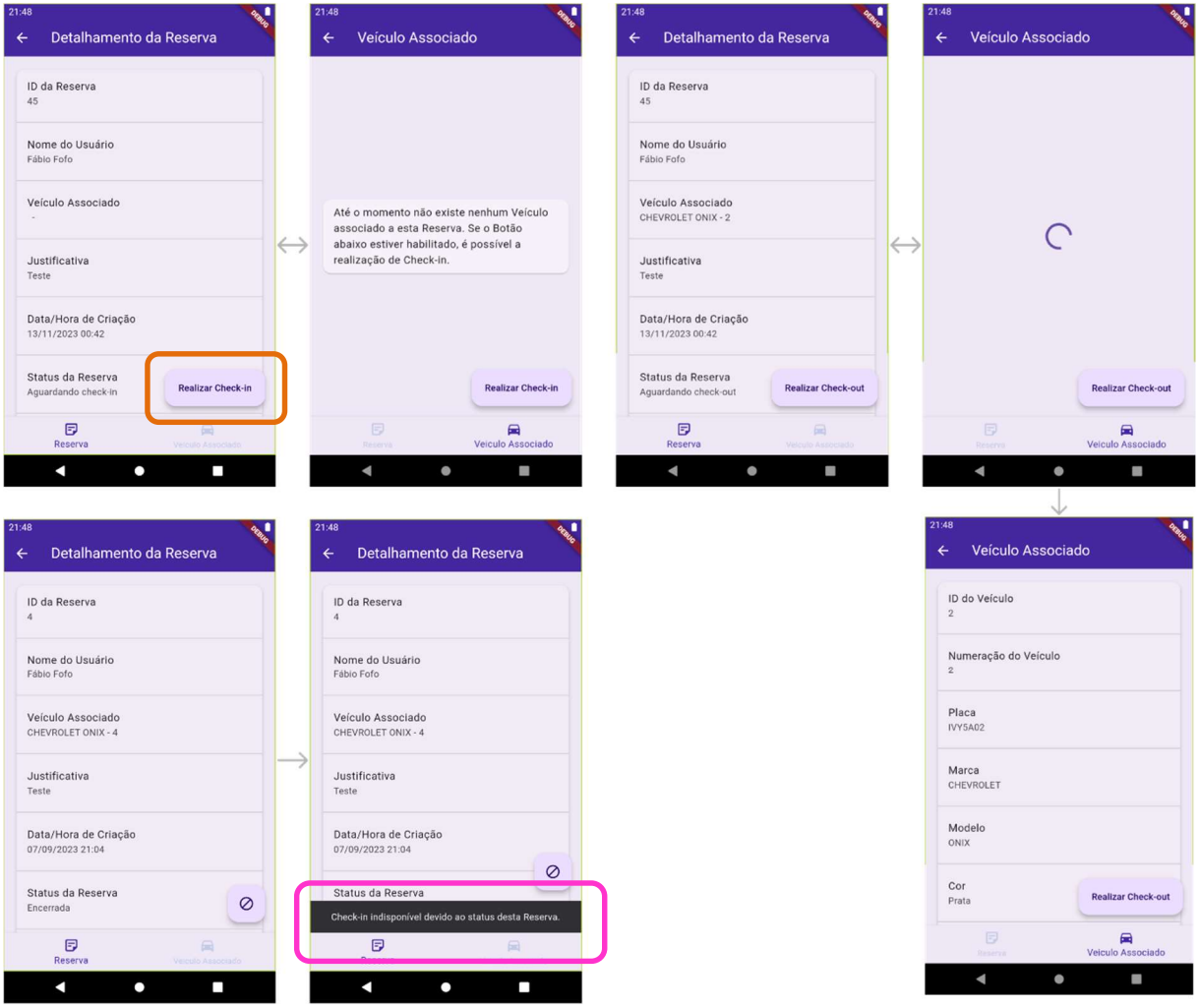
4.7.3 Tela Detalhamento de Reserva

Esta tela apresenta todas as informações armazenadas no banco de dados relacionadas à reserva selecionada, incluindo detalhes do veículo associado, quando aplicável. Quanto às reservas sem vínculo veicular, a área destinada ao detalhamento do veículo é preenchida por uma mensagem indicativa da ausência de associação entre a reserva e algum veículo. Caso a reserva tenha já sido vinculada, as informações do veículo associado serão carregadas conforme dados constantes no banco de dados.

No canto inferior da tela consta-se um FloatingActionButton extensível, indicando as rotas possíveis para as reservas que aguardam a realização de Check-in ou Check-out. Caso a reserva se encontre encerrada, o botão que levaria a estas rotas estará desabilitado e, se acionado, exibirá um SnackBar com a mensagem indicativa de que não será possível realizar estas ações devido ao estado da reserva. A seguir, os fluxos que representam as operações

descritas, bem como destaque em laranja para identificação do FloatingActionButton e em rosa para identificação do SnackBar:

Figura 28: Linkar - Fluxo Tela Detalhamento da Reservas



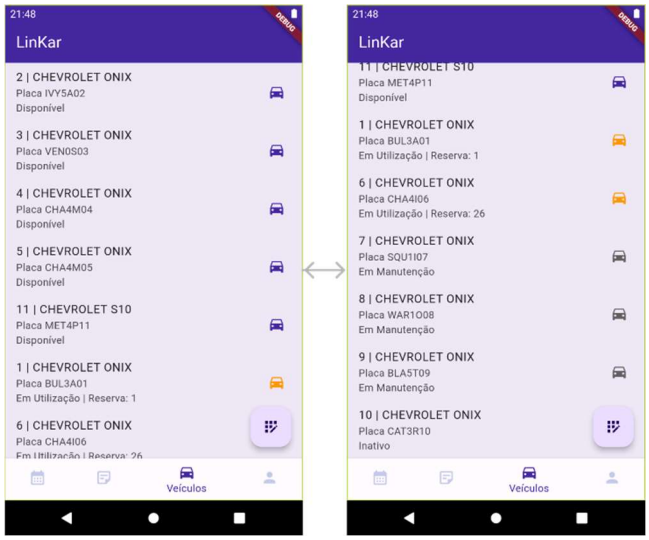
As rotas das funções Realizar Check-in e Realizar Check-out partindo de uma reserva detalhada serão abordadas no decorrer deste projeto. Vale dizer que o acesso a esta funções também é possível através das listagens de objetos da classe “Veículo”.

4.7.4 Tela Veículos

Após o carregamento da tela, a listagem completa dos veículos é apresentada, seguindo critérios de ordenação baseados no estado do veículo e em sua numeração. Desta maneira, semelhantemente à Tela Reservas, os veículos são organizados, priorizando aqueles que aguardam a realização do Check-in, seguidos por aqueles que aguardam a realização de Check-out, os que foram enviados para a manutenção e, por fim, aqueles que foram inativados do

sistema. Dentro de cada estado de veículo, a ordenação é feita de acordo com sua numeração, proporcionando uma visualização estruturada, conforme representado abaixo:

Figura 29: Linkar - Tela Veículos



A coloração dos ícones reflete o estado do veículo, mantendo um padrão em todo o código sempre que uma lista de veículo é exibida. As condições que determinam a aplicação das cores nos ícones podem ser verificadas a seguir:

Tabela 5: Condicionamento de Cores mediante o Estado de Veiculos

Cor	Condição
	Se o estado do veículo for igual a Disponível, a cor roxo profundo será aplicada (Colors.deepPurple[800])
	Se o estado do veículo for igual a Em Utilização, a cor laranja será aplicada (Colors.orange)
	Se o estado do veículo for igual a Em Manutenção, a cor cinza será aplicada (Colors.grey[700]).
	Se o estado do veículo for igual a Aguardando Check-in, a cor azul-acinzentado será aplicada (Colors.blueGrey[900]).
	Para qualquer outro valor de estado de veículo, a cor aplicada preta será aplicada (Colors.black).

Mediante ao clique nos objetos da lista, o usuário gestor é direcionado à tela de detalhamento destes objetos. O detalhamento traz a informações detalhadas dos veículos, como Numeração, Cor do Veículo, Placa, Registro Nacional de Veículos Automotores (RENAVAM),

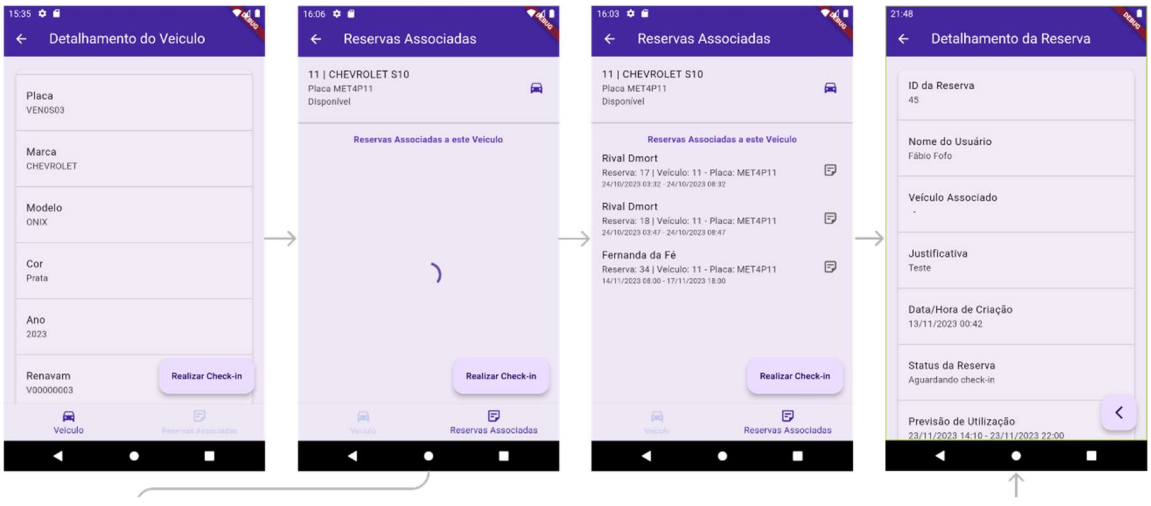
e listagem de todas as reservas vinculadas e permite a realização de Check-in ou Check-out, conforme for o caso de seu estado.

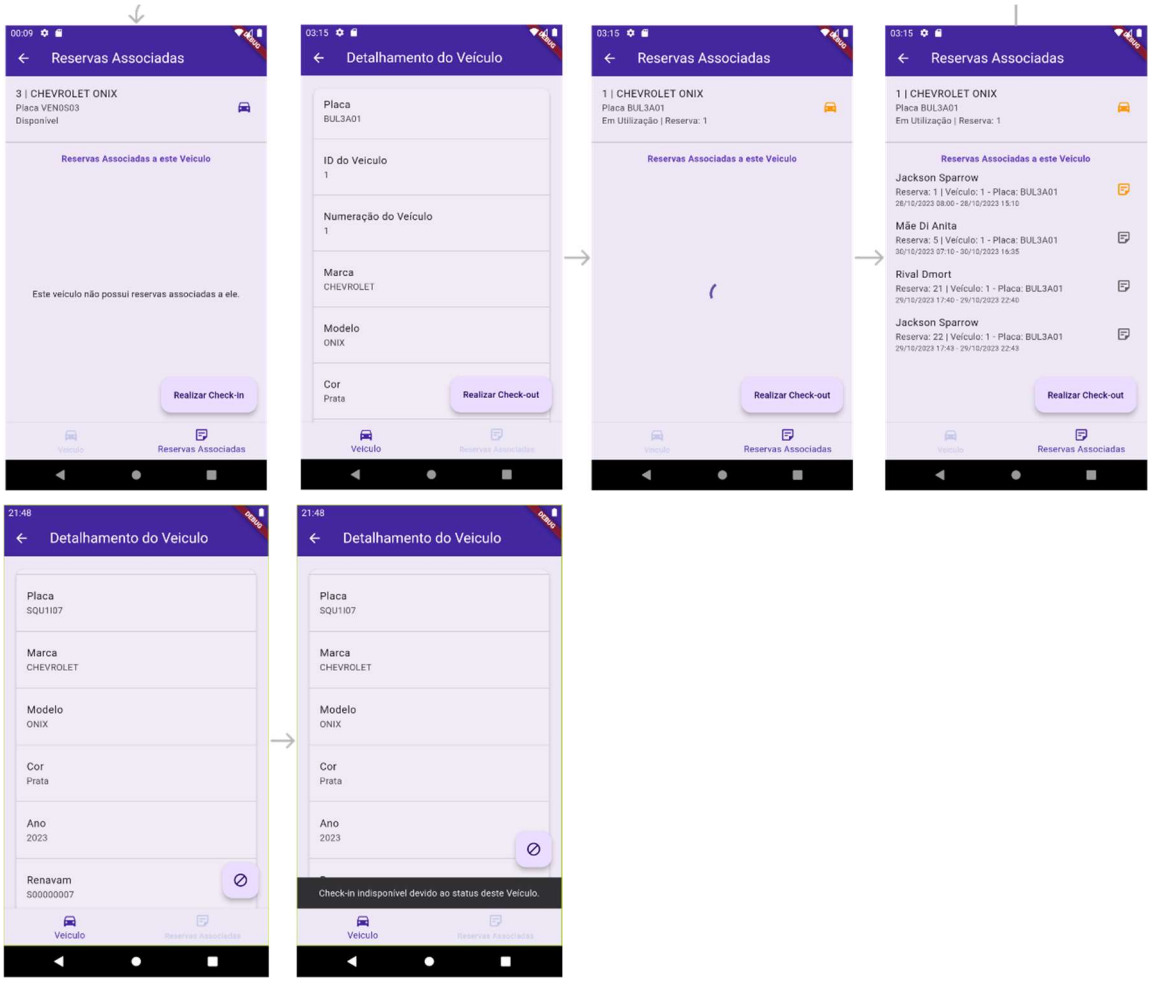
4.7.5 Tela Detalhamento de Veículo

Esta tela apresenta todas as informações armazenadas no banco de dados relacionadas o veículo selecionado, incluindo listagem das reservas que já foram associadas a este veículo, quando aplicável. Quanto aos veículos que não foram vinculados a reservas, a área destinada a listagem das reservas é preenchida por uma mensagem indicativa da ausência de associação entre o veículo e a alguma reserva. Caso o veículo já tenha sido vinculado a pelo menos uma reserva, a listagem das reservas associadas ao veículo será carregada conforme dados constantes no banco de dados. Antes de listar as reservas o código sempre irá consultar o banco de dados e verificar o quantitativo de reservas.

No canto inferior da tela consta-se um FloatingActionButton extensível, indicando as rotas possíveis para os veículos que se encontrem disponíveis para a realização de Check-in ou em utilização para a realização de Check-out. Caso o veículo se encontre em manutenção ou inativo, o botão que levaria a estas rotas estará desabilitado e, se acionado, exibirá um SnackBar com a mensagem indicativa de que não será possível realizar estas ações devido ao estado atual do veículo. A seguir, os fluxos que representam as operações descritas:

Figura 30: Linkar - Fluxo Tela Detalhamento do Veículo





As rotas das funções Realizar Check-in e Realizar Check-out partindo de um veículo selecionado serão abordadas no decorrer deste projeto. Vale dizer que as reservas que foram associadas ao veículo podem ser detalhadas após o carregamento da listagem de reservas exibidas por esta tela. Contudo, este detalhamento de reserva acessadas por estas funcionalidades não levará a demais rotas. Esta abordagem impede que ocorram laços de repetições entre as rotas de detalhamentos de reservas e veículos.

4.7.6 Tela Usuários e Detalhamento de Usuários

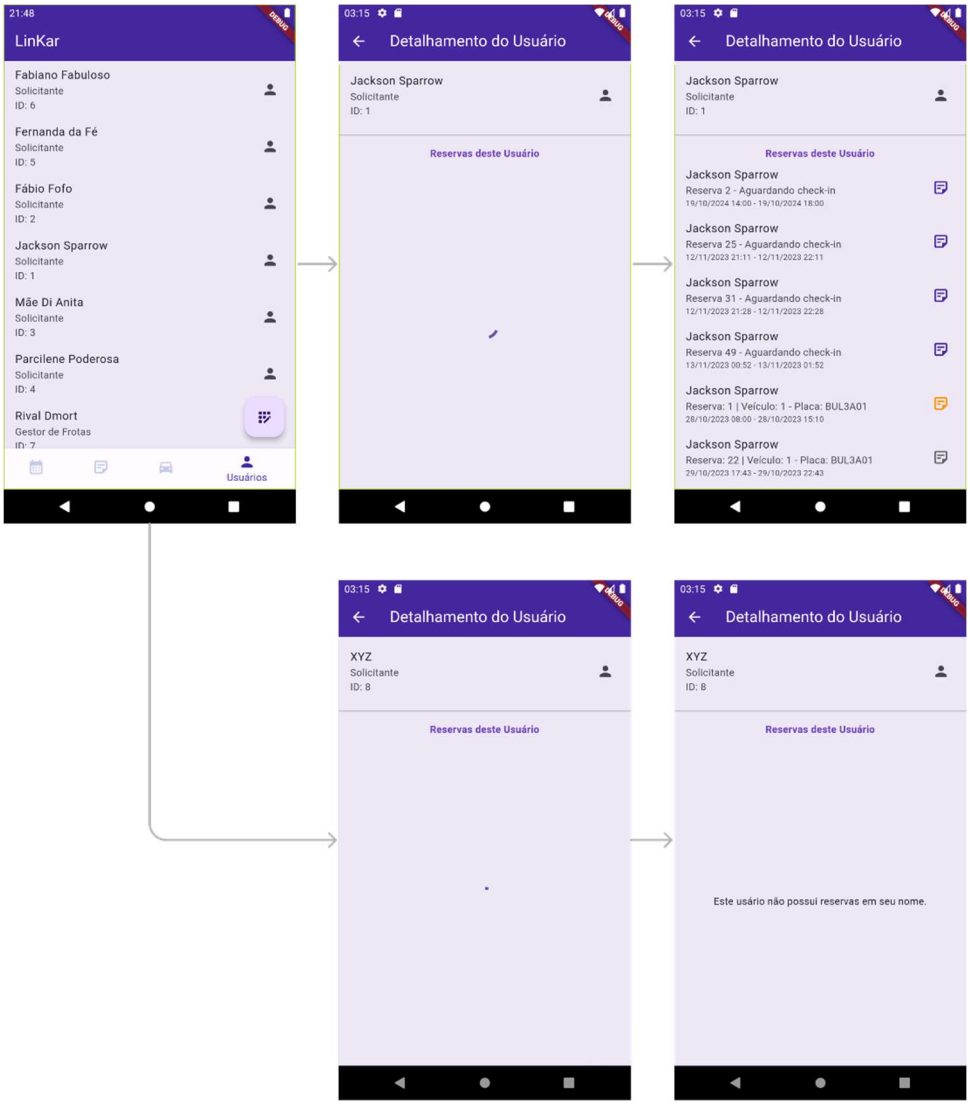
Após o carregamento da tela, a listagem completa dos usuários é apresentada, seguindo critérios de ordenação em ordem alfabética. Se o usuário gestor clicar em um dos objetos da classe “Usuarios” aqui listados, ele será conduzido a Tela Detalhamento de Usuários.

A Tela Detalhamento de Usuários traz todas as informações do usuário selecionado conforme seu cadastro realizado no banco de dados, contendo seu número identificador e seu perfil de acesso. Logo a abaixo destas informações, o aplicativo realiza consulta para listar todas

as reservas elaboradas pelo usuário detalhado. Caso o esse usuário não tenha feito solicitações de reservas, será exibida mensagem indicativa que ele não possui reservas em nome no lugar destinado à listagem.

A organização da listagem de reservas segue a mesma lógica de organização da lista apresentada na Tela Reservas. A seguir, os fluxos que representam as operações descritas:

Figura 31: Linkar - Fluxo Tela Usuários e Tela Detalhamento de Usuário



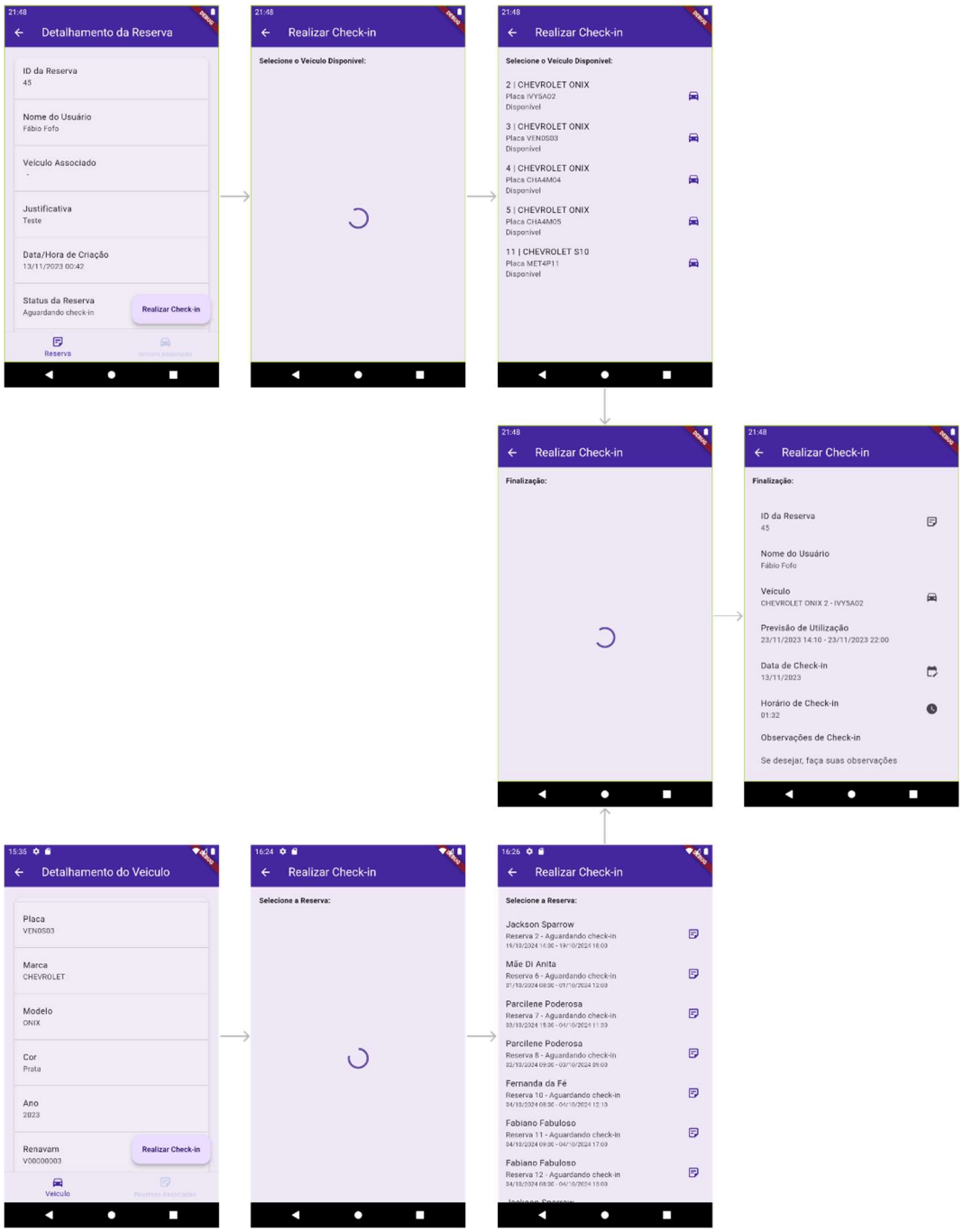
Se o usuário gestor escolher uma das reservas listadas no detalhamento do usuário, ele será redirecionado para a Tela de Detalhamento de Reservas. A lógica para as operações de Check-in, Check-out ou a impedimento dessas ações, com base no estado da reserva, seguirá a mesma abordagem da Tela de Reservas. Essa abordagem garante intuitividade na utilização do aplicativo, pois aciona as mesmas funcionalidades, proporcionando uma experiência fluida e

familiar ao usuário. Esclarece-se que funções de inclusão, alteração, ou exclusão de usuários não estavam previstas no escopo desta pesquisa.

4.7.7 Tela Realizar Check-in

A tela Realizar Check-in parte de duas abordagens diferentes, tendo cada uma delas, duas etapas diferentes. Caso o usuário gestor intente realizá-lo partindo da Tela Detalhamento de Reservas, a Tela Realizar Check-in iniciará a primeira etapa com a Tela de Seleção de Reservas que aguardam Check-in, que contém uma lista de reservas neste estado, após consulta realizada no banco de dados. Caso o usuário gestor parta da Tela Detalhamento de Veículos, a Tela Realizar Check-in iniciará a primeira etapa com a Tela de Seleção de Veículos Disponíveis que contém uma lista de veículos neste estado, após consulta realizada no banco de dados. A segunda etapa consiste com o carregamento dos objetos selecionados e o preenchimento das informações conforme formulário constante na finalização do Check-in. A seguir, o fluxo aqui descrito pode ser visualizado:

Figura 32: Linkar - Fluxo Tela Realizar Check-in Etapa 1

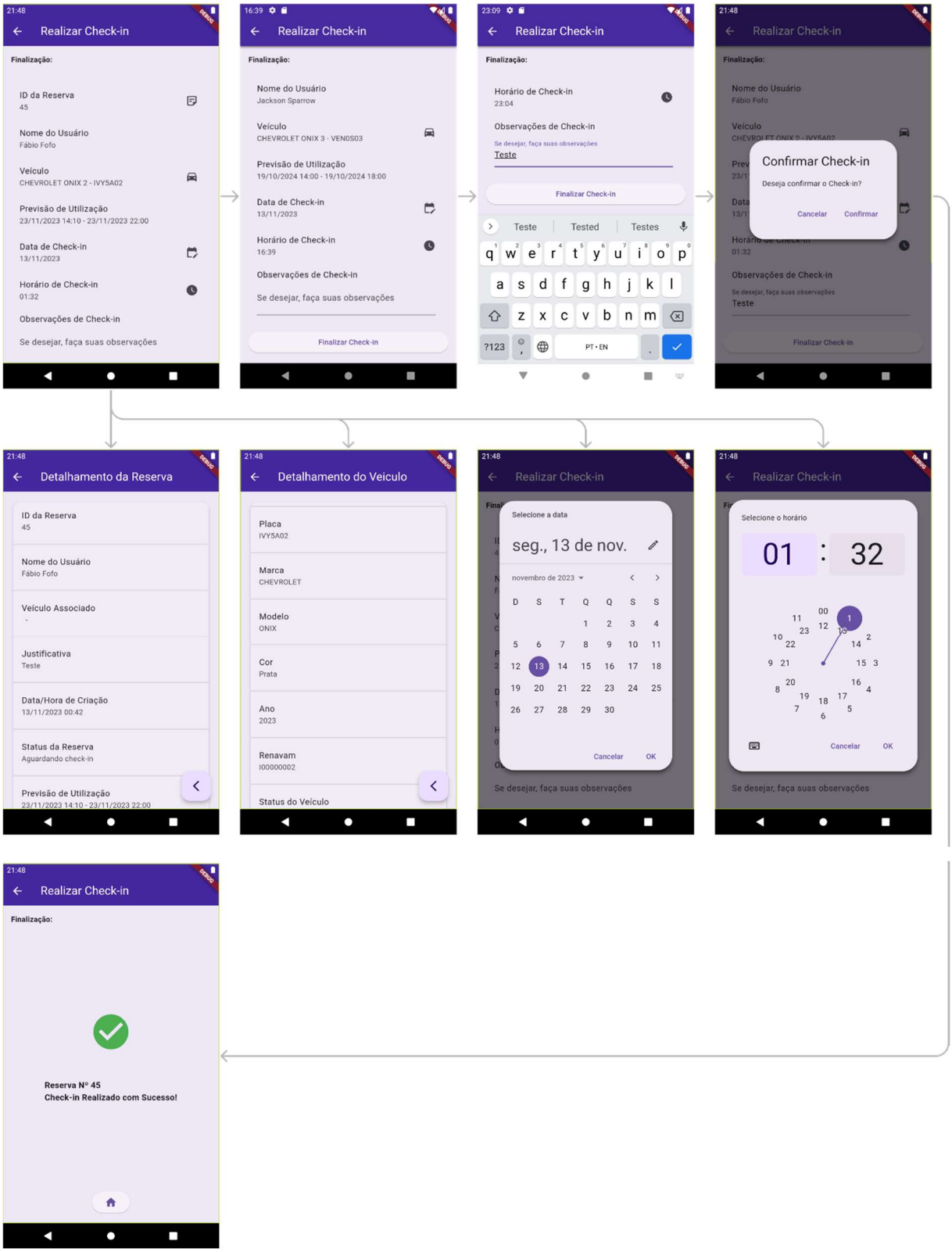


O formulário nesta segunda etapa oferece recursos interativos ao usuário. É possível consultar os objetos selecionados, utilizar componentes constantes no sistema do dispositivo para indicar a data e a hora efetiva da realização do Check-in e registrar alguma observação, caso assim o gestor acredite ser oportuno.

Conforme configurado no código, por padrão, o sistema lança a data e o horário atual referente ao momento de carregamento da tela. Acionando o botão “Finalizar Check-in” localizado no final do formulário, será exibido uma caixa de diálogo questionando o usuário se ele deseja finalizar este procedimento. Se sim, o código lançará o Check-in no banco de dados,

associando a reserva ao veículo, alterando seus estados, e a tela exibirá mensagem de sucesso constando o número da reserva, zerando a rota trilhada para a instancia inicial do sistema. A ilustração a seguir, mostra os fluxos discutidos neste tópico:

Figura 33: Linkar - Fluxo Tela Realizar Check-in Etapa 2 Finalização

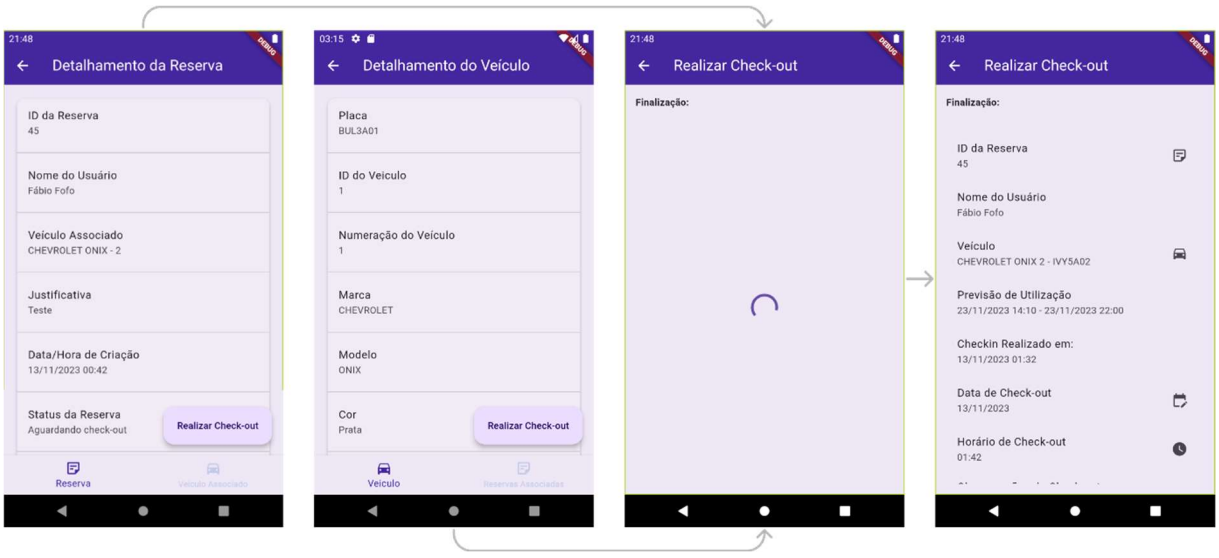


Após a interação com a tela responsável pela confirmação de sucesso na realização do Check-in, ela é removida da rota e o aplicativo retorna à Tela Inicial. Em seguida, o código realiza uma nova consulta ao banco de dados para que o calendário exiba as alterações resultantes da conclusão do Check-in. Esse processo assegura que as informações apresentadas ao usuário gestor estejam sempre atualizadas e refletindo o estado atual das reservas no sistema.

4.7.8 Tela Realizar Check-out

Independente de se partir pelo detalhamento de reservas ou veículos, a tela Realizar Check-out seguirá para sua única etapa que consiste com o carregamento dos objetos selecionados e o preenchimento das informações conforme formulário constante na finalização do Check-in. A seguir, o fluxo aqui descrito pode ser visualizado:

Figura 34: Linkar - Fluxo Tela Realizar Check-out Etapa 1

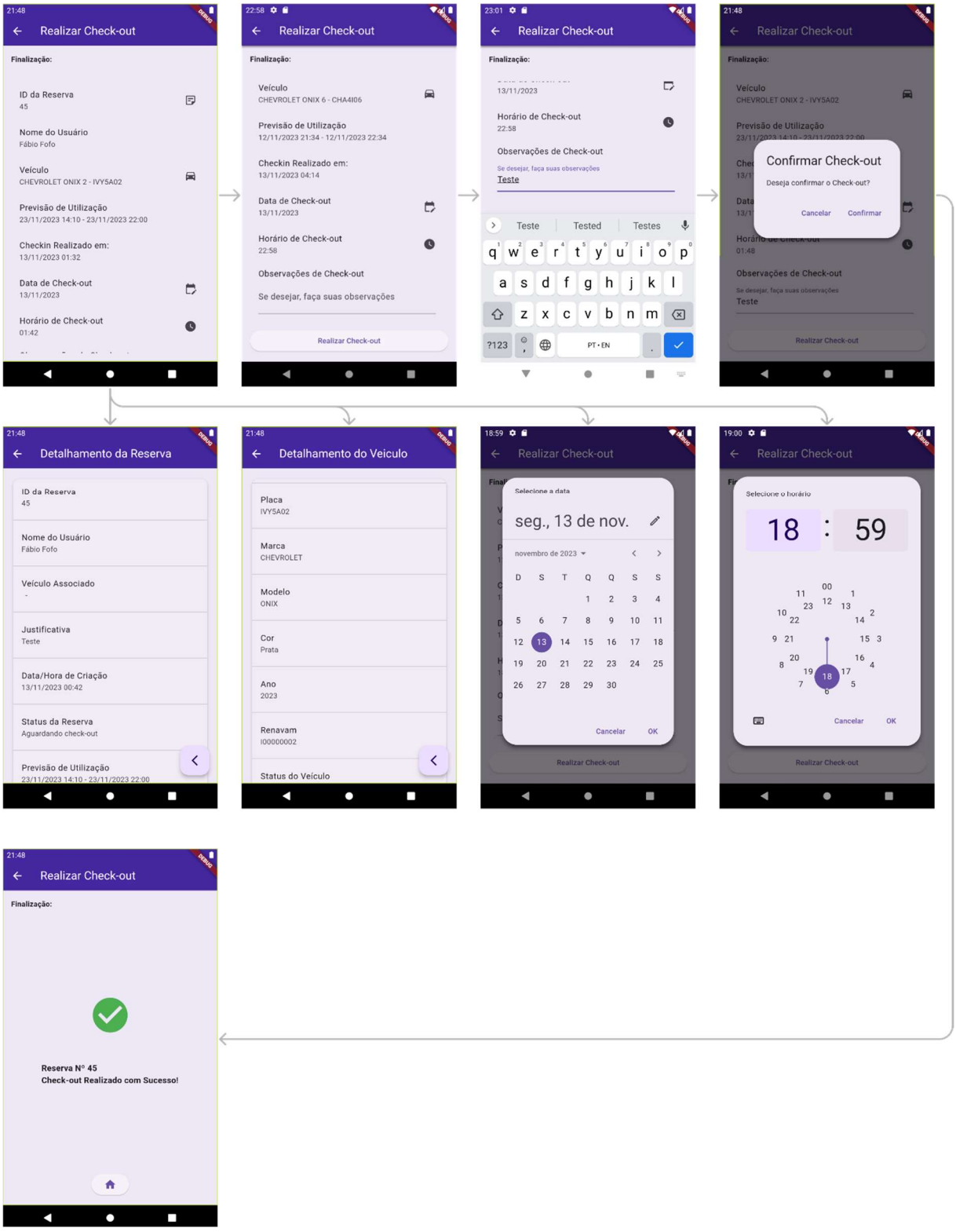


O formulário oferece recursos interativos ao usuário. É possível consultar os objetos selecionados, utilizar componentes constantes no sistema do dispositivo para indicar a data e a hora efetiva da realização do Check-out e registrar alguma observação, caso assim o gestor acredite ser oportuno.

Conforme configurado no código, por padrão, o sistema lança a data e o horário atual referente ao momento de carregamento da tela. Acionando o botão “Realizar Check-out” localizado no final do formulário, será exibido uma caixa de diálogo questionando o usuário se ele deseja finalizar este procedimento. Se sim, o código lançará o Check-out no banco de dados,

desassociando a reserva na tabela “Veículos”, mas mantendo a o veículo associado na tabela “Reservas”, alterando seus estados, e a tela exibirá mensagem de sucesso constando o número da reserva, zerando a rota trilhada para a instancia inicial do sistema. A ilustração a seguir, mostra os fluxos discutidos neste tópico:

Figura 35: Linkar - Fluxo Tela Realizar Check-out Etapa 2



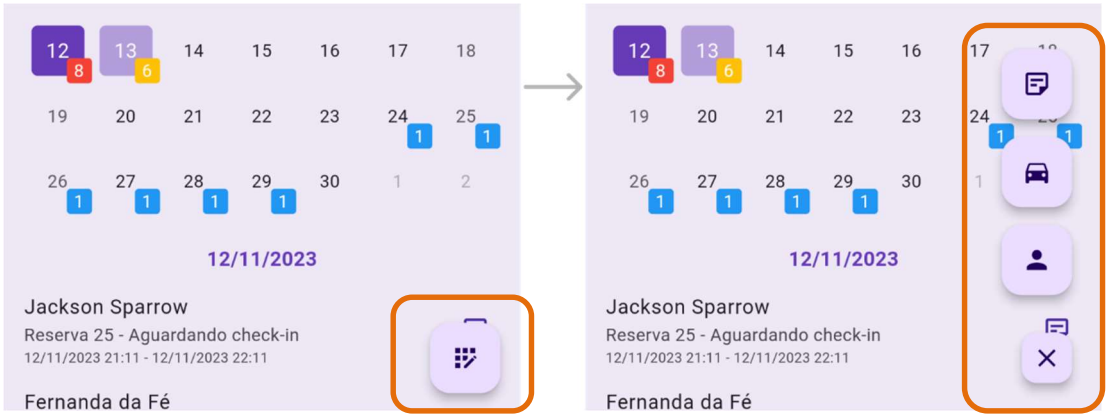
Após a interação com a tela responsável pela confirmação de sucesso na realização do Check-out, ela é removida da rota e o aplicativo retorna à Tela Inicial. Em seguida, o código realiza uma nova consulta ao banco de dados para que o calendário exiba as alterações resultantes da conclusão do Check-out. Esse processo assegura que as informações apresentadas ao usuário gestor estejam sempre atualizadas e refletindo o estado atual das reservas no sistema.

4.7.9 Componente ExpandableFab

Presente na Tela Inicial, em todas as seções, há o componente ExpandableFab, ou botão flutuante expansível. Este componente, quando acionado pelo usuário, se expande exibindo um menu de botões. Por meio dele é possível implementar os mais diversos recursos.

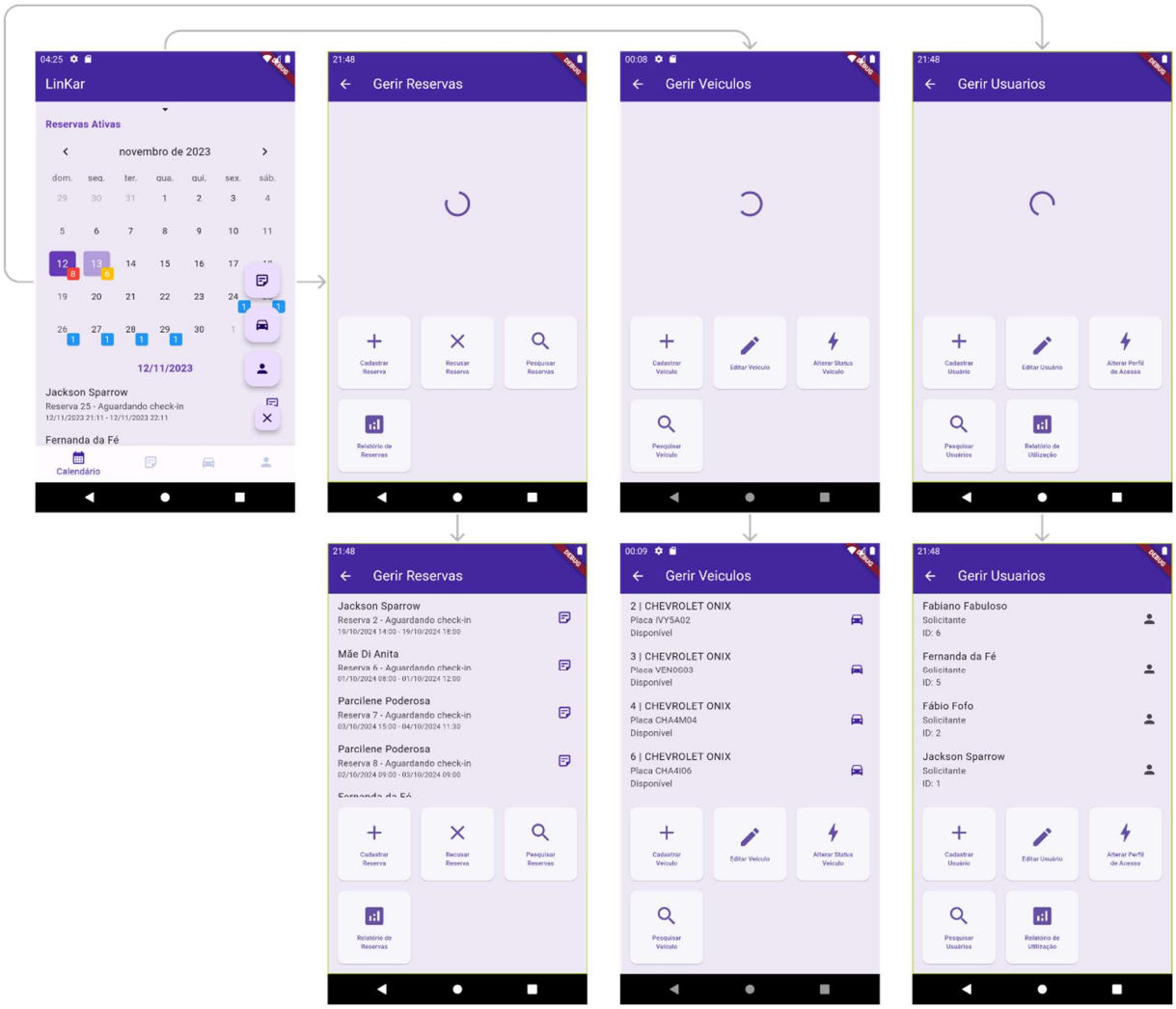
Neste trabalho, o botão foi inserido visando oferecer intuitividade em ações voltadas ao cadastro de objetos e a alteração de informações. Quando pressionado, são exibidos 3 outros botões, em linha vertical, com os ícones adotados nesta pesquisa para as classes de “Reservas”, “Veículos” e “Usuários”, conforme demonstrado no destaque alaranjado e na ilustração abaixo:

Figura 36: Linkar - Componente ExpandableFab



De acordo com o ícone pressionado, o usuário gestor poderá ser direcionado a seguintes telas: Tela Gerir Reservas, Tela Gerir Veículos e Tela Gerir Usuário, conforme mostrado pela figura a seguir:

Figura 37: Linkar - Componente ExpandableFab



A estrutura destas telas segue um padrão visual, apresentando uma subdivisão entre a parte superior e inferir do aplicativo. Na parte superior se encontra um construtor que carrega os objetos das classes equivalentes aos ícones selecionados, listando-os com todos os dados constantes no banco de dados, seguindo a mesma lógica implementada para os modelos listados na Tela Inicial, conforme sua seção equivalente. Na parte inferior, é apresentado um menu de funcionalidades, de acordo com a classe da tela em questão.

4.7.10 Tela Gerir Reservas

Dentre as funções apresentadas pelo menu, apenas duas delas estavam no escopo desta pesquisa, pois elas foram úteis para a validação dos testes realizados que envolveram o lançamento e o envio de requisições ao banco de dados. A função Cadastrar Reserva e a função

Recusar Reserva foram implementadas para validar as operações lógicas e atualizações sistêmicas quanto as abordagens apresentadas pelo calendário implementado.

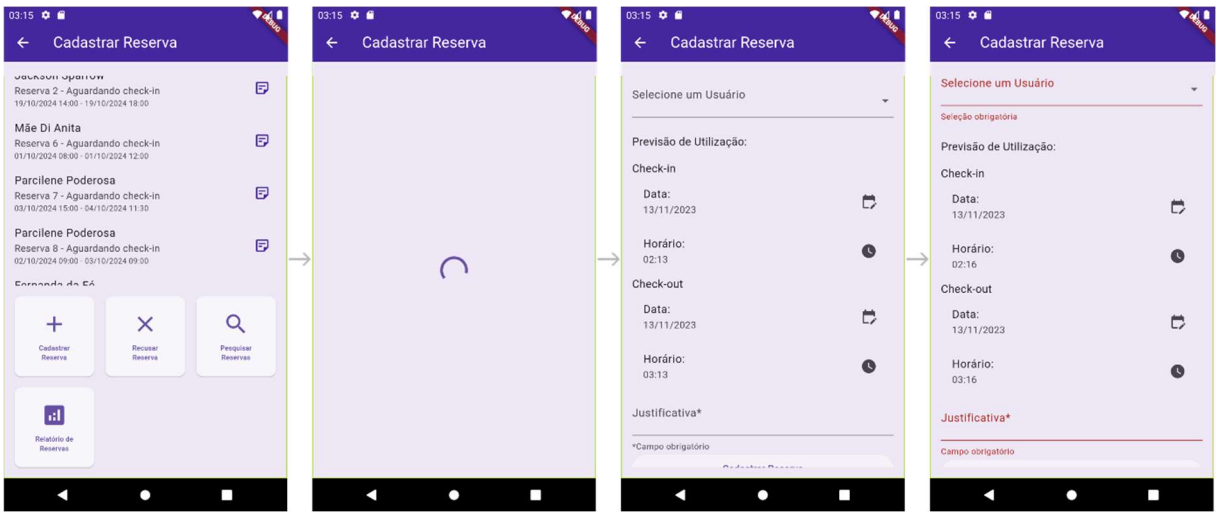
As funções implementadas levam a telas específicas, sendo elas a Tela Cadastrar Reserva e a Tela Recusar Reserva. A seguir serão apresentadas a discussão quanto as suas implementações.

4.7.11 Tela Cadastrar Reserva

Esta tela consiste no preenchimento de um formulário que requer as informações necessárias para validação do cadastramento de reservas no banco de dados. Dessa forma, obrigatoriamente, devem ser informados o nome do usuário ao qual a reserva pertencerá, a previsão do período de utilização e a justificativa desta utilização, conforme atributos e campos modelados para esta pesquisa.

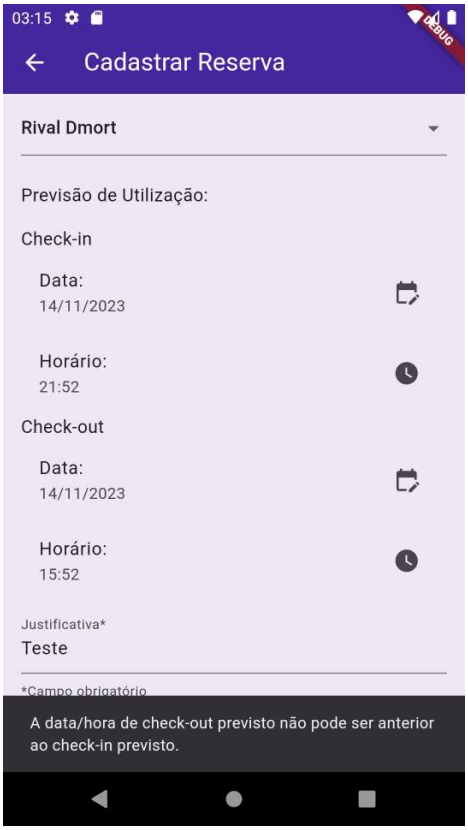
Para facilitar o aplicativo carrega os dados dos usuários e os exibe em um DropdownButtonFormField (destacado em laranja na Figura 34), que lista o nome de todos os usuários. Contudo, caso não seja informado um usuário ou algum dos citados espaços não estejam preenchidos o sistema acusará destaque em vermelho aos campos no momento que o botão Cadastrar Reserva for acionado. Isto ocorre pois o formulário está sujeito a uma validação, sendo este um recurso interessante para se evitar erros, conforme visualizado a seguir:

Figura 38: Linkar - Fluxo Tela Cadastrar Reserva: Validador de Campos Vazios



Além disso, o validador inclui uma verificação entre as datas de Check-in e Check-out. Como não é possível realizar um Check-out sem ter feito primeiro um Check-in, se o usuário inserir uma data ou horário inferior para o Check-out em comparação com a data ou horário informado para o Check-in, ao acionar o botão "Cadastrar Reserva", um SnackBar será exibido, indicando que essa ação não é possível, conforme a seguinte captura:

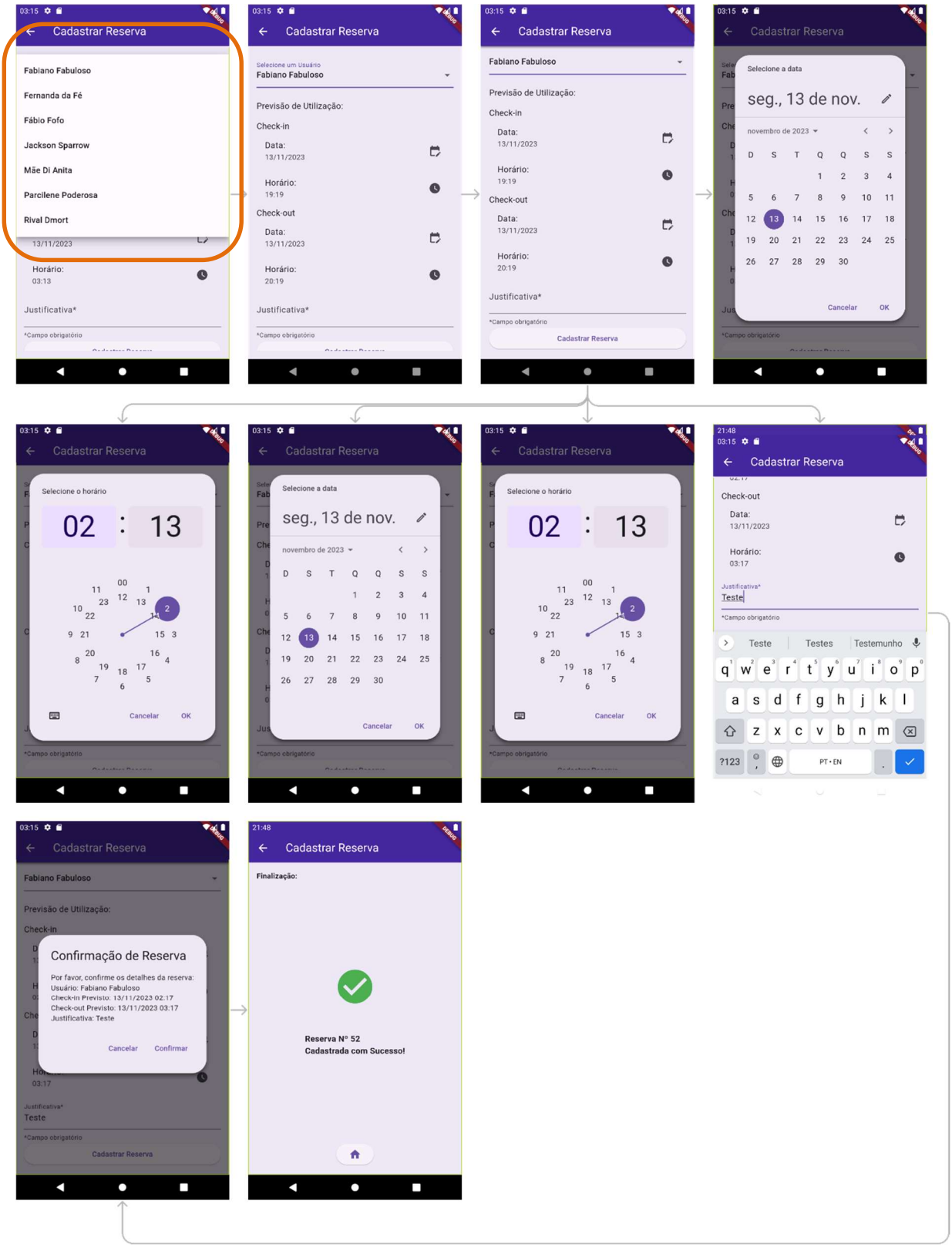
Figura 39: Linkar - Fluxo Tela Cadastrar Reserva: Validador Previsão de Utilização



Neste sentido é oportuno dizer que o código já preenche automaticamente as informações de previsão de utilização de maneira padronizada. A data atual é utilizada como parâmetro. A partir da data atual captada pelo sistema no momento do carregamento da tela, são adicionados 5 minutos para o Check-in e 65 minutos para o Check-out, resultando em um período previsto de utilização de 1 hora. No entanto, o usuário tem a liberdade de modificar essas informações conforme necessário.

Após a validação do formulário, a tela exibirá uma caixa de diálogo com o resumo das informações lançadas, questionando o usuário se ele deseja finalizar este procedimento. Se sim, o código cadastrará a reserva no banco de dados, associando a reserva ao usuário, e a tela exibirá mensagem de sucesso constando o número da reserva, zerando a rota trilhada para a instancia inicial do sistema, conforme a rota ilustrada abaixo:

Figura 40: Linkar - Fluxo Tela Cadastrar Reserva



Após a interação com a tela responsável pela confirmação de sucesso no cadastro da reserva, ela é removida da rota e o aplicativo retorna à Tela Inicial. Em seguida, o código realiza

uma nova consulta ao banco de dados para que o calendário exiba as alterações resultantes da conclusão do cadastro. Esse processo assegura que as informações apresentadas ao usuário gestor estejam sempre atualizadas e refletindo o estado atual das reservas no sistema.

4.7.12 Tela Recusar Reserva

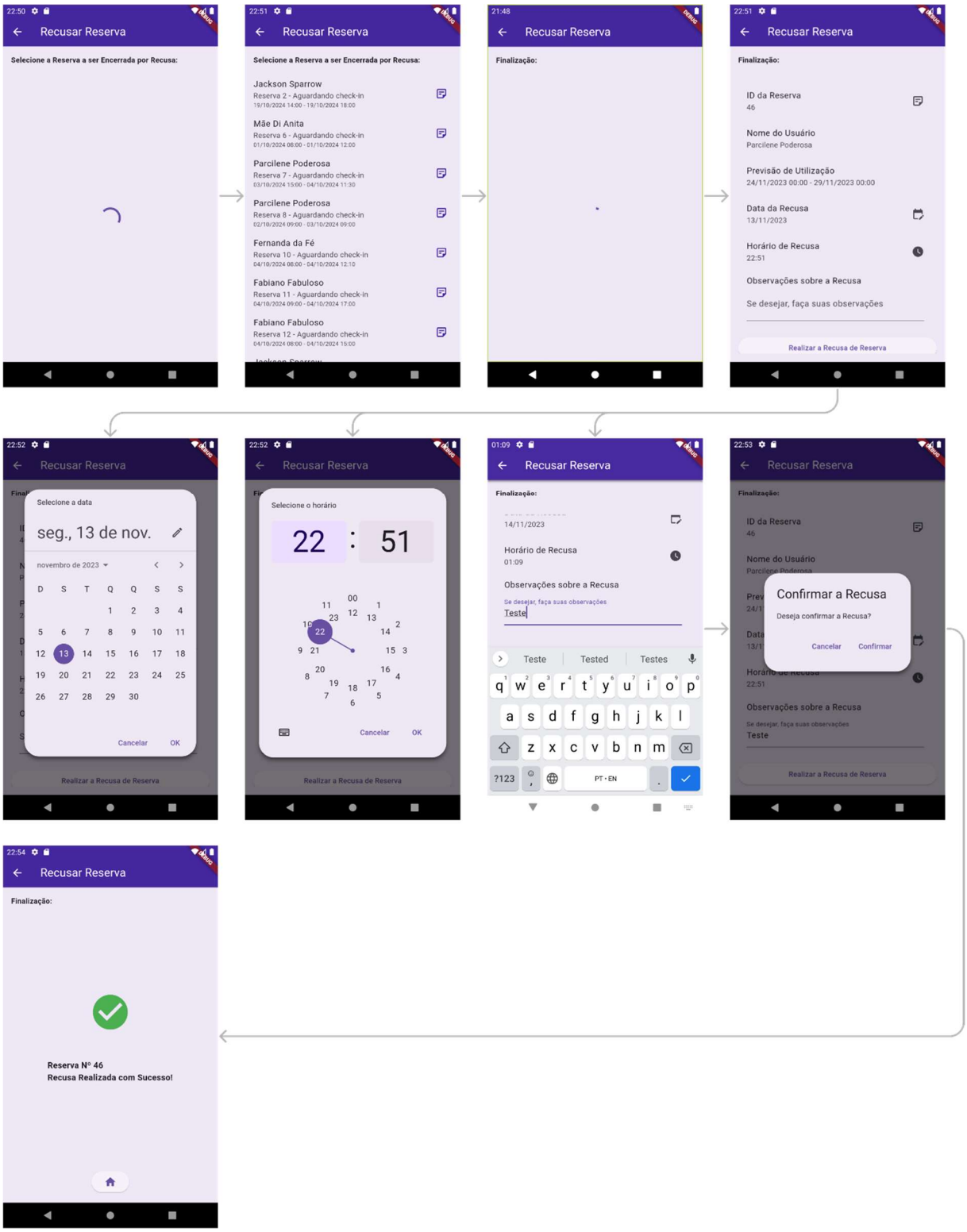
Caso o gestor opte por recusar um reserva, ele poderá fazer isso acessando o menu que leva a Tela Gerir Reservas. Acionando a devida funcionalidade ele será encaminhado para a Tela Recusar Reserva. A recusa consiste em duas partes. Primeiramente, após o carregamento da listagem das reservas que aguardam Check-in, o usuário gestor deverá selecionar a reserva que irá ser recusada. Após, isso ele será direcionado ao formulário que receberá as informações da recusa, semelhantemente ao que ocorre na finalização da Tela Realizar Check-in e na Tela Realizar Check-out.

No formulário desta tela em específico, mesmo que o usuário gestor não digite observações, por padrão, elas serão lançadas automaticamente contendo a informação de que nenhum veículo foi associado. Caso o gestor digite algo, seu texto será concatenado a informação de que nenhum veículo fora associado. Estas observações serão lançadas tanto nos campos voltados às observações de Check-in quanto de Check-out na tabela “Reservas” do banco de dados. O horário informado da recusa, também preencherá os horários de Check-in e Check-out. Esta abordagem enfatiza a percepção de que a reserva fora recusada.

Conforme configurado no código, por padrão, o sistema lança a data e o horário atual referente ao momento de carregamento da tela. Acionando o botão “Realizar a Recusa de Reserva” localizado no final do formulário, será exibido uma caixa de diálogo questionando o usuário se ele deseja finalizar este procedimento. Se sim, o código lançará a recusa no banco de dados, alterando os campos pertinentes as tratativas aqui adotadas, e a tela exibirá mensagem de sucesso constando o número da reserva, zerando a rota trilhada para a instancia inicial do sistema.

A ilustração de toda a rota aqui descrita, pode ser conferida por meio da ilustração logo a seguir:

Figura 41: Linkar - Fluxo Tela Recusar Reserva



Após a interação com a tela responsável pela confirmação de sucesso no cadastro da reserva, ela é removida da rota e o aplicativo retorna à Tela Inicial. Em seguida, o código realiza uma nova consulta ao banco de dados para que o calendário exiba as alterações resultantes da

conclusão do cadastro. Esse processo assegura que as informações apresentadas ao usuário gestor estejam sempre atualizadas e refletindo o estado atual das reservas no sistema.

4.7.13 Tela Gerir Veículos e Tela Gerir Usuários

O desenvolvimento das Telas Gerir Veículos e Gerir Usuários, bem como suas funcionalidades, não estavam abarcadas pelo escopo deste projeto. Elas apenas foram inseridas no protótipo como vislumbre para um desenvolvimento futuro. Dessa forma, as funções de cadastro, alteração e exclusão, não foram implementadas. A seguir, segue-se representação visual de suas telas após o carregamento das listas contendo os objetos de suas classes correspondentes:

Figura 42: Linkar - Fluxo Tela Gerir Veículos

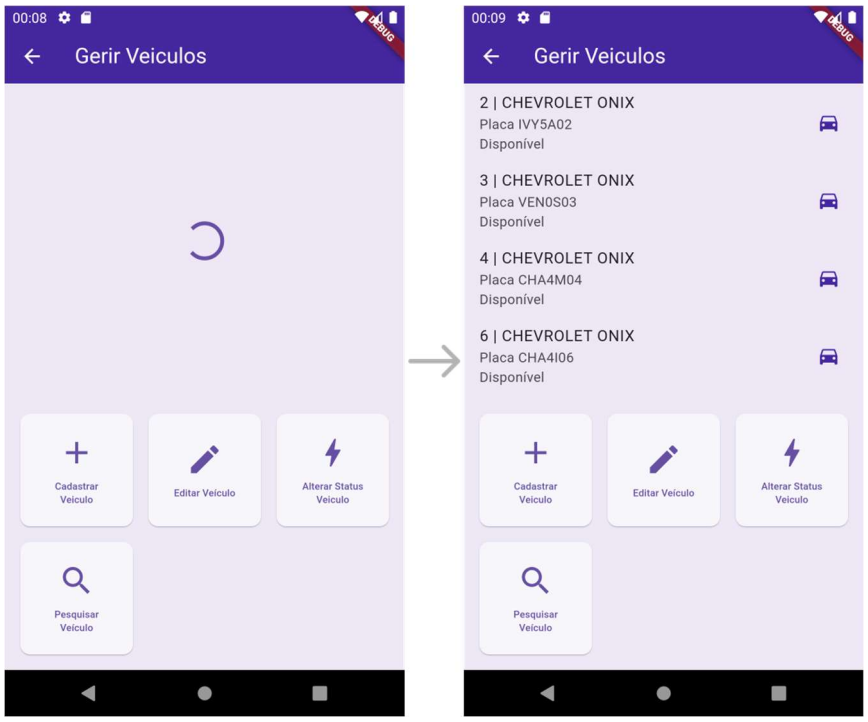
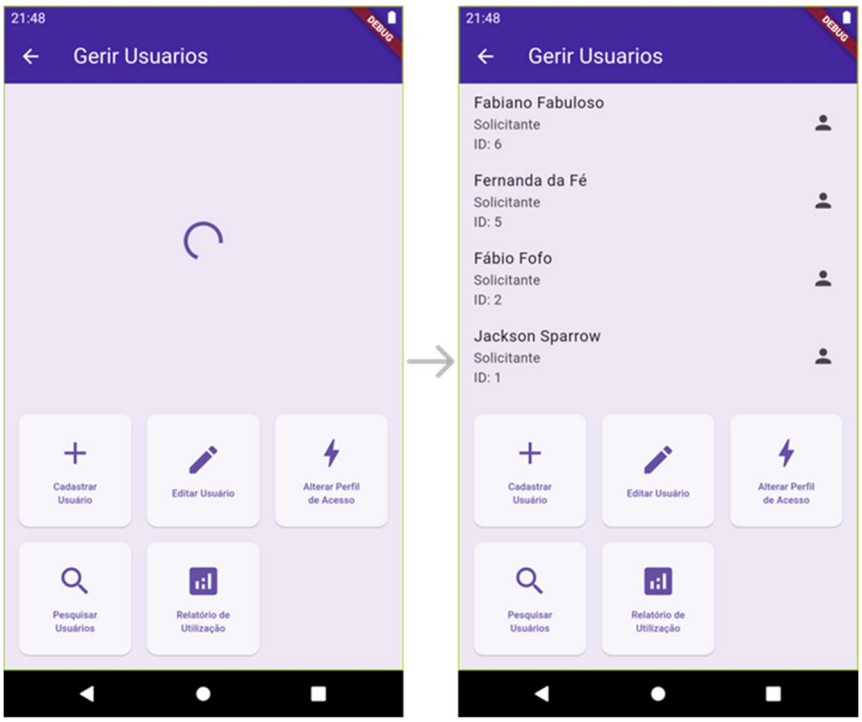


Figura 43: Linkar - Fluxo Tela Gerir Usuários



Com base nestes dizeres, considerando o desenvolvimento proposto encerra-se o capítulo de análise dos resultados. A diante, este projeto deixa suas considerações finais e contribuições para trabalhos futuros.

5 CONCLUSÃO E CONSIDERAÇÕES FINAIS

O desenvolvimento do sistema apresentado neste trabalho reflete uma abordagem moderna para a gestão de reservas e veículos, integrando uma interface intuitiva, funcionalidades práticas e frameworks. Ao longo deste projeto, foram exploradas diferentes camadas, desde a lógica de negócios até a implementação visual, demonstrando um entendimento abrangente do ciclo de vida de um sistema de gestão de frotas.

5.1 Interface Intuitiva

A implementação de uma interface intuitiva foi prioridade para proporcionar uma experiência usuário positiva ao usuário. Com o auxílio da prototipagem, conforme discutido no capítulo anterior em seu tópico 4.2, foi possível desenvolver as seguintes implementações visuais:

- Calendário Interativo: A inclusão de um calendário interativo oferece ao gestor uma visão consolidada e visualmente compreensível das reservas. A capacidade de navegar facilmente entre datas e identificar rapidamente o status das reservas contribui para a eficiência na tomada de decisões.
- Fluxos de Navegação Lógicos: Os fluxos de navegação foram projetados de maneira lógica, proporcionando ao gestor uma jornada coesa ao realizar operações como check-in, check-out, cadastro de reservas, entre outras. Isso pode reduzir a curva de aprendizado e aumentar a eficiência do usuário;
- Identificação Visual de Status: as reservas e os veículos são diferenciados por cores com base no seu estado, facilitando a sua identificação. Isso permite que o gestor possa reconhecer visualmente as reservas que estão aguardando check-in ou check-out, aquelas que estão concluídas ou recusadas, bem como aqueles veículos que estão disponíveis, em utilização, em manutenção ou inativos;
- Retornos comunicativos ao usuário: após a finalização de algum lançamento, o usuário sempre é notificado, evitando equívocos em seu entendimento quanto aos procedimentos já realizados;
- Adoção de Material Design Kit: o Flutter permite a adaptação de seus frameworks ao padrão estabelecido pelo Material Design Kit. Isto proporciona

uma experiência alinhada com os padrões de design de acordo com a versão informada.

5.2 Desenvolvimento Moderno

Utilizando frameworks, importações e configurações descritas no capítulo anterior, o desenvolvimento se beneficia de outras APIs e pacotes que otimizam as execuções, favorecendo a estruturação de um código seguro e organizado. Como estratégias de execução, citam-se:

- Importação de Bibliotecas: os pacotes otimizam o desenvolvimento, auxiliando na construção de abordagens lógicas complexas, como a tradução de componentes, tratamento de eventos e associações de dados e informações com outras tecnologias, como por exemplo a criação de um calendário completo traduzido e que respeita fuso horários, permitindo operações que envolvem diferentes datas e horários;
- Botões Flutuantes Expansíveis: A presença dos botões flutuantes expansíveis adiciona uma camada de praticidade à interface. Esses botões fornecem acesso direto a funcionalidades-chave, como gerenciamento de reservas, veículos e usuários, simplificando a navegação e a execução de tarefas, bem como a realização de check-ins e check-outs, de acordo com o caso;
- Validação e Feedbacks: A implementação de validadores e feedbacks visuais evitam que lançamentos indevidos aconteçam, como por exemplo informar um check-out com uma data, ou horário, inferior ao check-in;
- Funções assíncronas: as execuções e construções de componentes respeitam a prioridade de buscas e retornos, evitando erros ou associações indevidas. Alguns componentes não são carregados até que a consulta ao banco de dados seja finalizada, como por exemplo a lista de objetos, e consequentemente alguma interação com um estado indevido;
- Utilização de um banco de dados online: por meio do Supabase os dados são armazenados em nuvem, sempre disponíveis, segundo os critérios de autenticação e acesso;
- Funções que organizam os dados: a exibição dos dados é tratada e organizada conforme a abordagem definida no espoco, trazendo, por exemplo, reservas que respeitam a expectativa da utilização e a sua data de criação por parte do usuário.

Cita-se inclusive as tratativas de listar primeiramente os veículos disponíveis e as reservas que aguardam check-in e listagem dos usuários em ordem alfabética.

- Funções que detalham os objetos: várias funções detalham os objetos e exibem informações quanto aos seus status, previsões de datas de utilização, disponibilidade veicular, dentre outras.

Desse modo, entende-se que todos os objetivos propostos neste projeto foram cumpridos. O que não significa que outras perspectivas podem ser adotadas, conforme detalhado a seguir.

5.3 Desafios e Oportunidades Futuras

Imaginando-se um contexto futuro, pode ser interessante criar funções voltadas a inclusão e alteração de dados, como o cadastro de veículos e alteração de seus estados entre ativos e inativos. Há margem para a inclusão de recursos que tragam relatórios detalhados, customização de pesquisas, notificações quanto a velocidade de veiculos e integração com demais tecnologias e sistemas externos, como por exemplo a utilização de APIs que são provenientes de serviços voltados ao rastreamento veicular para que sua localização seja informada em tempo real.

Considerando que esta pesquisa se concentra no módulo do usuário gestor de reservas de veículo, se aproveitando da funcionalidade que leva ao cadastro de uma nova reserva, é possível otimizar o desenvolvimento do módulo de usuários. Neste módulo podem estar presentes as funcionalidades de cadastramento de novos usuários a ser realizado pelo próprio novo usuário, a alteração de seus cadastros, a definição de perfis de acesso, o carregamento de fotos ou imagens de perfil, dentre outras.

Neste sentido, vale dizer que o zelo pela segurança dos dados é uma preocupação central em sistemas de gestão. Considerações adicionais sobre criptografia, autenticação robusta e autorizações adequadas devem ser abordadas para garantir a integridade e a privacidade dos dados e assim podem ser projetadas.

Conforme as tratativas de carregamento e busca de dados cadastrados no banco de dados definidas nesta pesquisa, não há operações lógicas voltadas a otimização desta execução. Em trabalhos futuros, é oportuno realizar abordagens que possam reduzir o tempo de retorno da consulta.

A customização do aplicativo pode ser testada, por meio de outros efeitos visuais, por meio de técnicas de colorações e iluminação, customização de bordas, contrastes, ícones e implementação de temas. No caso de uma futura pesquisa voltada a estes conceitos, ela poderá colher a perspectiva de usuários após estas adequações e alterações.

O código desenvolvido aborda somente as tratativas de telas voltadas a dispositivos móveis, não tendo tratativas para telas e plataformas diferentes. A responsividade em desenvolvimento multiplataforma pode ser objeto de estudo e a criação de recursos lógicos para que a execução da linha de código preveja várias situações e adequações de layout conforme o tamanho da tela em execução.

Por fim, apesar dos testes realizados durante o desenvolvimento, é viável conduzir testes mais abrangentes, incluindo testes de carga e teste de usabilidade com usuários finais. Isso ajudará a identificar e corrigir possíveis problemas e aprimorar a experiência do usuário.

5.4 Considerações Finais

Este projeto revelou uma oportunidade para aplicar de forma prática os conhecimentos adquiridos ao longo do curso de Ciência da Computação. A abordagem adotada, desde a concepção da lógica de negócios até a implementação visual, não apenas reflete compreensão sólida dos princípios fundamentais do desenvolvimento de software, mas também destaca a capacidade de transformar conceitos teóricos em produtos tangíveis, com ênfase na praticidade e estética. A habilidade demonstrada de criar soluções que são não apenas funcionais, mas também visualmente intuitivas, ressalta a competência em traduzir teoria em produtos que atendem não apenas às exigências iniciais do escopo, mas que também estabelecem base para evolução e expansão futuras.

O processo de desenvolvimento enfatizou a importância da análise de requisitos, teste e interação entre os objetos, o banco de dados e suas tratativas lógicas. A compreensão profunda da interconexão entre as diferentes partes do sistema e a habilidade de torná-las utilizáveis frente às demandas da gestão foram habilidades desenvolvidas durante este projeto.

Assim, o sistema de gestão de reservas e veículos transcende a categorização de um projeto acadêmico, sendo uma aplicação prática dos conceitos de tecnologia da informação para a modelagem de um aplicativo multiplataforma. Contribui não apenas para a formação profissional, mas também prepara para os desafios e oportunidades emergentes no mercado. Este trabalho, portanto, oferece subsídios para a construção de sistemas similares em contextos

do mundo real, sublinhando a importância do pensamento analítico, design centrado no usuário e implementação eficiente.

Numa análise mais abrangente, o código revela a aplicação prática de conceitos voltados a programação orientada a objetos, design de banco de dados e operações assíncronas em Dart. A estrutura modular e as práticas de codificação adotadas sugerem uma abordagem organizada e escalável para o desenvolvimento de aplicativos Flutter integrados a bancos de dados externos. O código desenvolvido neste projeto pode ser acessado através do link https://github.com/oirivaldo/linkar e se posiciona como um exemplo para os estudantes das áreas da tecnologia da informação, proporcionando uma compreensão profunda e prática de conceitos presentes no desenvolvimento de software moderno.

REFERÊNCIAS

ACKOFF, R. L. **Planejamento empresarial**. Rio de Janeiro: Livros Técnicos e Científicos, 1976.

AMATYA S.; ARIANIT K. **Cross-platform mobile development:** challenges and opportunities. ICT Innovations 2013, pp. 219-229. Springer, Heidelberg, 2014.

BARBOSA, E. R.; Brondani, G. **Planejamento estratégico organizacional**. Revista eletrônica de contabilidade – UFSM, v. 1, n. 2, 2005.

BUCHHOLZ, S.; SCHILL, A**. Adaptation-aware web caching: Caching in the future pervasive web**. In: SPRINGER. Kommunikation in Verteilten Systemen (KiVS), 2003. p. 55–66.

CAPTERRA. **Fleet Management Software**. Disponível em: https://www.capterra.com.br/directory/20007/fleet-management/software. Acesso em: 24 março de 2023.

CECHINEL, A. et al. **Avaliação do framework angular e das bibliotecas react e knockout para o desenvolvimento do front-end de aplicações web**. Florianópolis, SC, 2017.

CHOUDHURY, N. **World wide web and its Journey from web 1.0 to web 4.0**. International Journal of Computer Science andInformation Technologies, Citeseer, v. 5, n. 6, p. 8096–8100, 2014.

CLEMENTE, Q. K. **Gestão de frota de veículos rodoviários**. Instituto Superior Técnica Universidade Técnica de Lisboa, Lisboa, Portugal, 2008.

COASE, R. H. **The Natureof the Firm**. Economica VI. 4, n. 16, 1937, p. 386-405.

DART.DEV. **Intl 0.18.1**. Disponível em: https://pub.dev/packages/intl. Acesso em: outubro de 2023.

DUARTE, Alessandro Silveira**. Processo de sistematização de ambientes de residência em software brasileiros**. 2015. 134 f. Dissertação (Mestrado em Informática) - Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2015.

EL-KASSAS, W. S. et al. **Taxonomy of Cross-Platform Mobile Applications Development Approaches. Ain Shams Engineering Journal**, 2015. ISSN 2090-4479. Disponível em: http://www.sciencedirect.com/science/article/pii/S2090447915001276. Acesso em: nov. 2020.

FAYAD, M. E.; SCHIMIDT, D. C.; JOHNSON, R. **Building application frameworks**: Object-oriented foundations of framework design. John Wiley & Sons, 1999.

FIGMA. **What is Figma?** Disponível em: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma-. Acesso em: maio de 2023.

FLUTTER. **Flutter documentation**. Disponível em: https://flutter.dev/docs. Acesso em: 29 de abril de 2023.

HANSCHKE, S.; HEITKOTTER, H.; MAJCHRZAK, T. A. **Evaluating crossplat form development approaches for mobile applications**. In: web information systems and technologies. Springer, 2013. p. 120–138.

HOFFA, C., MEHTA, G., FREEMAN, T., DEELMAN, E., KEAHEY, K., BERRIMAN, B., GOOD, J. (2008). **On the use of cloud computing for scientic workows**. In eScience, 2008.

HOFMANN, M.; BECK, A.; CONDRY, M. **Example services for network edge proxies**. 2000.

JOYCAR. **Homepage**. Disponível em: https://joycar.com.br/. Acesso em: 24 mar. 2023.

KEYES, J. **Software engineering handbook**. CRC Press. 2002.

KULESZA, R. et al. **Evolução das arquiteturas de software rumo à web 3.0**. Sociedade Brasileira de Computação, 2018.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Fundamentos de metodologia científica**. 7 ed. São Paulo: Atlas, 2010.

LEONTIADES, M. **Management policy, strategy, andplans**. Boston: Litle, Brown. 1982.

LIMA, L. M. S. de; OLIVEIRA, S.; OLIVEIRA, A. P. de. **Gestão de transportes**. Editora e Distribuidora Educacional S.A. Londrina, 2014.

MARCOTTE, E. **Responsive Web Design**. A Book Apart, 2010.

MATOS, B. R. D.; SILVA, J. G. de B. **Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando abordagem nativa e multiplataforma**. Estudo comparativo, Brasília, 2016.

MINAYO, Maria Cecília de Souza. **O desafio do conhecimento**. 11 ed. São Paulo: Hucitec, 2008.

MUNDSTOCK, P. **Relação entre planejamento estratégico e desempenho superior**. Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul, 2008.

OLIVEIRA, D. de P. R. de. **Excelência na administração estratégica: a competitividade para administrar o futuro das empresas:** com depoimentos de executivos. 3 ed. São Paulo: Atlas, 1997.

PEREIRA, R. A. **Sistema para Gestão de Frotas de Veículos**. Universidade Regional de Blumenau, 2016.

PONTES, T. B.; ARTHAUD, D. D. B. **Metodologias ágeis para o desenvolvimento de softwares**. Ciência E Sustentabilidade, v. 4, n. 2, 2018.

POTTER, M. E. **Estratégia competitiva:** técnicas para análise de indústrias e da concorrência. Rio de Janeiro: Elsevier, 2004.

PRESSMAN, R. S. **Engenharia de Software**: uma abordagem profissional. McGraw Hill Editora, 2016.

PUB.DEV. **Flutter_localization 0.1.14**. Disponível em: https://pub.dev/packages/flutter_localization. Acesso: em outubro de 2023.

PUB.DEV. **Table_calendar 3.0.9**. Disponível em: https://pub.dev/packages/table_calendar. Acesso: em outubro de 2023.

RODRIGUES, T. V. et al. **Sistema de informações para gestão e manutenção de frota de veículos**. 2019.

ROUDAKI, A.; KONG, J.; YU, N. **A classification of web browsing on mobile devices**. Journal of Visual Languages & Computing, Elsevier, v. 26, p. 82–98, 2015.

SANTANA, A. G.; FERREIRA, V. R.; SILVA, R. C. da. **Planejamento empresarial e tributário de Startups**: definições e características específicas. Revista do Direito Público, Londrina, v. 16, n. 2, p. 126-143, ago. 2021.

SCHMIDT, P. **Controladoria**: agregando valor para a empresa. Porto Alegre: Bookman, 2002.

SILVA FILHO, E. B. **Teoria da firma e a abordagem dos custos de transação**: elementos para uma crítica institucionalista. Revista Pesquisa & Debate, v. 17, n. 2, pg. 259-277, São Paulo, 2006.

SILVA, Rauan Hiago da et al. **Um sistema para o controle de gastos com a frota de veículos da prefeitura municipal de Lajes/RN**. 2020.

SOMMERVILLE, I. **Engenharia de Software**. 9ª Edição, 2011. ed. [S.l.]: Pearson Education - BR, 2011.

STACK OVERFLOW. **Technology Most Loved Dreaded And Wanted Languages Loved**. 2021. Disponível em: https://insights.stackoverflow.com/survey/2021#technology-most-loved-dreaded-and-wanted-languages-loved. Acesso em: 29 de abril de 2023.

STEIN JUNIOR A. R. M.; SANTOS, M DOS. **Arquitetura REST API E Desenvolvimento De Uma Aplicação Web Service**. Projetos e relatórios de estágios, V.1 n. 1, 2019, p. 1-59.

SUPABASE. **Supabase Documentation**. Disponível em: https://supabase.com/docs. Acesso em: maio de 2023.

SUPABASE.IO. **Supabase_flutter 1.10.16.** Disponível em: https://pub.dev/packages/supabase_flutter/versions/1.10.16. Acesso em: novembro de 2023.

TARKOWSKA A, CARVALHO-SILVA D, COOK C. E, TURNER E, FINN RD, YATES A.D. **Eleven quick tips to build a usable REST API for life sciences**. PLoS Comput Biol v. 14, n. 12, 2018. Disponível em https://doi.org/10.1371/journal.pcbi.1006542. Acessado em março de 2023.

WILLIAMSON, O. E. **The Theoryof the Firm as Governance Structure**: From Choice to Contract. Journal of Economic Perspectives, v.16, n. 3, 2002.

ZENDESK. **O que é MVP?** Conceito, importância e como criar um Mínimo Produto Viável. 2022. Disponível em https://www.zendesk.com.br/blog/o-que-e-mvp/. Acessado em novembro de 2023.

ZUVOLA.COM. **Flutter_expandable_fab 2.0.0**. Disponível em: https://pub.dev/packages/flutter_expandable_fab. Acesso em: outubro de 2023.

APÊNDICE A - DETALHAMENTO DAS TABELAS DO BANCO DE DADOS

DETALHAMENTO DAS TABELAS DO BANCO DE DADOS

1. Tabela “Anos”

- Objetivo: Armazenar informações sobre os anos de fabricação de veículos.
- Atributos:
    - idAno (bigint): Identificador único do ano.
    - ano (text): Representação do ano de fabricação.
- Restrições:
    - Chave primária (Anos_pkey) definida pelo idAno.

2. Tabela “CoresCarros”

- Objetivo: Registrar as cores dos carros.
- Atributos:
    - idCor (bigint): Identificador único da cor.
    - cor (text): Descrição da cor do carro.
- Restrições:
    - Chave primária (CoresCarros_pkey) definida pelo idCor.

3. Tabela “Marcas”

- Objetivo: Armazenar informações sobre as marcas de veículos.
- Atributos:
    - idMarca (bigint): Identificador único da marca.
    - nomeMarca (text): Nome da marca do veículo.
- Restrições:
    - Chave primária (Marcas_pkey) definida pelo idMarca.

4. Tabela “Modelos”

- Objetivo: Registrar os modelos de veículos associados às marcas.
- Atributos:

- idModelo (bigint): Identificador único do modelo.
- idMarca (bigint): Identificador da marca associada.
- nomeModelo (text): Nome do modelo do veículo.

- Restrições:
  - Chave primária (Modelos_pkey) definida pelo idModelo.
  - Chave estrangeira (Veiculos_idMarca_fkey) referenciando a tabela “Marcas” pelo idMarca.

5. Tabela “Reservas”

- Objetivo: Registrar reservas de veículos.
- Atributos:
  - idReserva (bigint): Identificador único da reserva.
  - idUsuario (bigint): Identificador do usuário associado a reserva.
  - dataHoraCriacao (timestamp with time zone): Data e hora de criação da reserva.
  - justificativa (text): Justificativa da reserva.
  - idStatusReserva (bigint): Identificador do status da reserva.
  - dataHoraCheckin (timestamp with time zone): Data e hora de check-in.
  - obsCheckin (text): Observações do check-in.
  - dataHoraCheckout (timestamp with time zone): Data e hora de check-out.
  - obsCheckout (text): Observações do check-out.
  - idVeiculo (bigint): Identificador do veículo associado a reserva.
  - dataHoraPrevistaCheckin (timestamp with time zone): Data e hora prevista para o check-in.
  - dataHoraPrevistaCheckout (timestamp with time zone): Data e hora prevista para o check-out.
- Restrições:
  - Chave primária (Reservas_pkey) definida pelo idReserva.
  - Chave estrangeira (Reservas_idStatusReserva_fkey) referenciando a tabela “StatusReservas” pelo idStatusReserva.
  - Chave estrangeira (Veiculos_idReserva_fkey) referenciando a tabela “Veiculos” pelo idVeiculo.

6. Tabela “StatusReservas”

- Objetivo: Armazenar os diferentes status de uma reserva.
- Atributos:
    - idStatusReserva (bigint): Identificador único do status da reserva.
    - nomeStatusReserva (text): Descrição do status da reserva.
- Restrições:
    - Chave primária (StatusReservas_pkey) definida pelo idStatusReserva.

7. Tabela “StatusVeiculos”

- Objetivo: Registrar os diferentes status de um veículo.
- Atributos:
    - idStatusVeiculo (bigint): Identificador único do status do veículo.
    - nomeStatusVeiculo (text): Descrição do status do veículo.
- Restrições:
    - Chave primária (StatusVeiculos_pkey) definida pelo idStatusVeiculo.

8. Tabela “TiposFrota”

- Objetivo: Armazenar os tipos de frota de veículos.
- Atributos:
    - idTipo (bigint): Identificador único do tipo de frota.
    - nomeTipoFrota (text): Descrição do tipo de frota.
- Restrições:
    - Chave primária (Tipos_pkey) definida pelo idTipo.

9. Tabela “TiposPerfil”

- Objetivo: Registrar os diferentes tipos de perfil de usuários.
- Atributos:
    - idTipoPerfil (bigint): Identificador único do tipo de perfil.
    - nomeTipoPerfil (text): Descrição do tipo de perfil.

- Restrições:
    - Chave primária (TiposPerfil_pkey) definida pelo idTipoPerfil.

10. Tabela “Usuarios”

- Objetivo: Armazenar informações sobre os usuários do sistema.
- Atributos:
    - idUsuario (bigint): Identificador único do usuário.
    - nomeUsuario (text): Nome do usuário.
    - idTipoPerfil (bigint): Identificador do tipo de perfil do usuário.
- Restrições:
    - Chave primária (Usuarios_pkey) definida pelo idUsuario.
    - Chave estrangeira (Usuarios_idTipoPerfil_fkey) referenciando a tabela “TiposPerfil” pelo idTipoPerfil.

11. Tabela “Veiculos”

- Objetivo: Registrar informações detalhadas sobre os veículos.
- Atributos:
    - idVeiculo (bigint): Identificador único do veículo.
    - numeracao (bigint): Número identificador do veículo.
    - placa (text): Placa do veículo.
    - idMarca (bigint): Identificador da marca do veículo.
    - idModelo (bigint): Identificador do modelo do veículo.
    - idCor (bigint): Identificador da cor do veículo.
    - idAno (bigint): Identificador do ano de fabricação do veículo.
    - renavam (text): Número do RENAVAM do veículo.
    - idStatusVeiculo (bigint): Identificador do status do veículo.
    - idTipoFrota (bigint): Identificador do tipo de frota do veículo.
    - localizacao (text): Localização atual do veículo.
    - idReserva (bigint): Identificador da reserva associada ao veículo.
- Restrições:
    - Chave primária (Veiculos_pkey) definida pelo idVeiculo.

- Chave estrangeira (Veiculos_idMarca_fkey) referenciando a tabela “Marcas” pelo idMarca.
- Chave estrangeira (Veiculos_idModelo_fkey) referenciando a tabela “Modelos” pelo idModelo.
- Chave estrangeira (Veiculos_idAno_fkey) referenciando a tabela “Anos” pelo idAno.
- Chave estrangeira (Veiculos_idStatusVeiculo_fkey) referenciando a tabela “StatusVeiculos” pelo idStatusVeiculo.
- Chave estrangeira (Veiculos_idTipoFrota_fkey) referenciando a tabela “TiposFrota” pelo idTipo.
- Chave estrangeira (Veiculos_idReserva_fkey) referenciando a tabela “Reservas” pelo idReserva.

APÊNDICE B - PESQUISA PARA TCC II - FUNÇÕES ÚTEIS EM UM APLICATIVO DE RESERVA DE VEÍCULOS

Pesquisa para TCC II - Funções úteis em um aplicativo de Reserva de Veículos

Este formulário visa coletar respostas anônimas para servirem de parâmetro para as funções que irão ser desenvolvidas para um Software de Reserva de Veículos, visando se tornar uma ferramenta que auxilie na gestão de frotas corporativas. As funções estão sendo abordadas hipoteticamente sob a ótica de Usuário Gestor de Reserva de Veículos.

* Indica uma pergunta obrigatória

1. Em um grau de 1 a 5, em que 1 signifique pouco importante e 5 muito importante, **escalone a relevância destas funções** a serem desenvolvidas para um software de gestão de reservas de veículos de uma frota corporativa: *

Marcar apenas uma oval por linha.

	1	2	3	4	5
Check-in	◯	◯	◯	◯	◯
Check-out	◯	◯	◯	◯	◯
Rastreamento de Localização	◯	◯	◯	◯	◯
Indicação de Status de Veículos	◯	◯	◯	◯	◯
Cadastros de Veículos	◯	◯	◯	◯	◯
Cadastro de Usuários	◯	◯	◯	◯	◯
Pesquisas e Consultas	◯	◯	◯	◯	◯
Filtragem de Status	◯	◯	◯	◯	◯

2. Em um grau de 1 a 8, em que 1 signifique pouco importante e 8 muito importante, **redistribua a prioridade destas funções** a serem desenvolvidas para um software de gestão de reservas de veículos de uma frota corporativa: *

Marcar apenas uma oval por linha.

	1	2	3	4	5	6	7	8
Check-in	◯	◯	◯	◯	◯	◯	◯	◯
Check-out	◯	◯	◯	◯	◯	◯	◯	◯
Rastreamento de Localização	◯	◯	◯	◯	◯	◯	◯	◯
Indicação de Status de Veículos	◯	◯	◯	◯	◯	◯	◯	◯
Cadastros de Veículos	◯	◯	◯	◯	◯	◯	◯	◯
Cadastro de Usuários	◯	◯	◯	◯	◯	◯	◯	◯
Pesquisas e Consultas	◯	◯	◯	◯	◯	◯	◯	◯
Filtragem de Status	◯	◯	◯	◯	◯	◯	◯	◯

3. Caso você tivesse a oportunidade de utilizar um software de gestão para reservas de veículos de uma frota corporativa, quais destas plataformas estariam dentro de suas preferências? *

Selecione quantas plataformas preferir.

Marque todas que se aplicam.

- [ ] Moblie (Android ou IOS)
- [ ] Web (Browser/Navegador)
- [ ] Desktop (Instalável em Máquina Local)

4. Escolha a opção que melhor expressa seu pensamento quanto a viabilidade de um software mobile de reservas de veículos de uma frota corporativa: *

Marcar apenas uma oval.

- ◯ Considero muito útil e necessário
- ◯ Considero pouco útil e necessário
- ◯ Não vejo utilidade

5. Como usuário de um software de gestão de reservas de veículos em uma frota corporativa, você seria: *

Selecione a opção que melhor se associa com suas funções.

Marcar apenas uma oval.

- ◯ Usuário Gestor de Reservas
- ◯ Usuário Solicitante de Reservas

Opcionalmente, você pode se identificar a seguir.
Ao deixar seu nome e cargo ou ocupação, você autoriza a utilização destas informações no Trabalho de Conclusão de Curso do aluno Rivaldo Soares do Nascimento no Curso de Ciência da Computação do CEULP/ULBRA.

6. Nome Completo:

_____

7. Cargo/Ocupação:

_____

8. Caso deseje, utilize o espaço a seguir para deixar suas observações:

_____
_____
_____
_____
_____

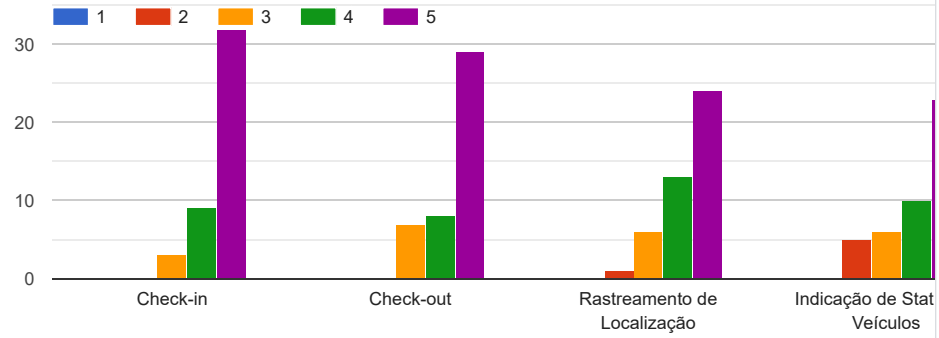
APÊNDICE C - PESQUISA PARA TCC II - FUNÇÕES ÚTEIS EM UM APLICATIVO DE RESERVA DE VEÍCULOS (RESUMO RESPOSTAS)

Pesquisa para TCC II - Funções úteis em um aplicativo de Reserva de Veículos

44 respostas

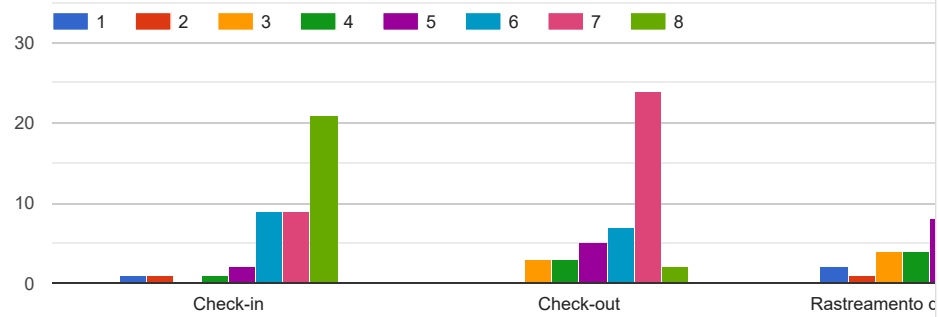
Em um grau de 1 a 5, em que 1 signifique pouco importante e 5 muito importante, **escalone a relevância destas funções** a serem desenvolvidas para um software de gestão de reservas de veículos de uma frota corporativa:

Copiar



Em um grau de 1 a 8, em que 1 signifique pouco importante e 8 muito importante, **redistribua a prioridade destas funções** a serem desenvolvidas para um software de gestão de reservas de veículos de uma frota corporativa:

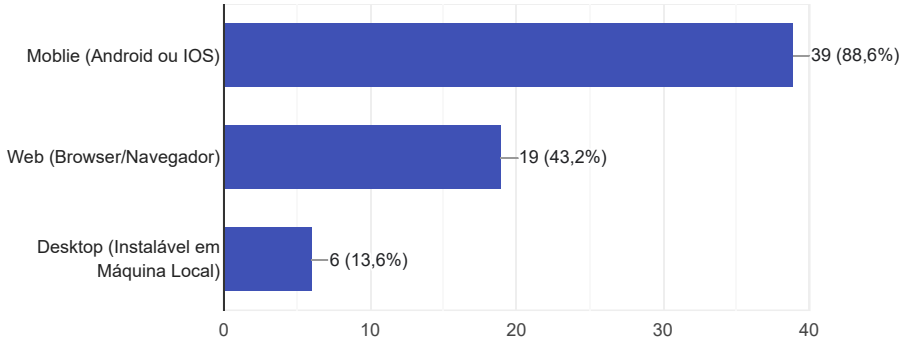
Copiar



Caso você tivesse a oportunidade de utilizar um software de gestão para reservas de veículos de uma frota corporativa, quais destas plataformas estariam dentro de suas preferências?

Copiar

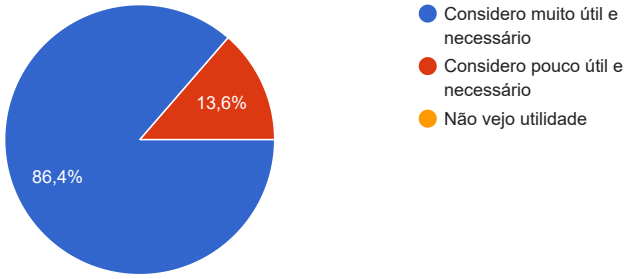
44 respostas



Escolha a opção que melhor expressa seu pensamento quanto a viabilidade de um software mobile de reservas de veículos de uma frota corporativa:

Copiar

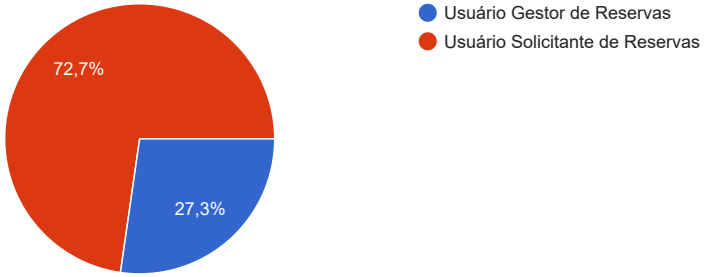
44 respostas



Como usuário de um software de gestão de reservas de veículos em uma frota corporativa, você seria:

Copiar

44 respostas



Opcionalmente, você pode se identificar a seguir.

Nome Completo:

29 respostas

Davi Henrique

Ana Paula Alves Cunha

Andrea Azevedo Cavalheiro

Carlos Capistrano

Jandérik Silva Marins

Nivonaldo Francisco Alves

Thereza Rejane de Souza Almeida

Newton Lopes de F. Neto

benedito martiniano

Pedro Araújo

Luzigleidson Carneiro de Sousa

Flávia Andréia Cappellesso

Vilton Ribeiro da Silva

Antônio Louça Curcino

Odilo Junior Oliveira Carvalho

Anderson Donizeth

Higor de Oliveira

Débora Ribeiro Pereira

Jonathan Escobar

EDUARDO MONTEIRO

Jansley Carvalho Mendes Corrêa

Rafael Liberato de França Lima

Ana Clara da Conceição Macêdo da Silva

Anderson Victor Silva Araujo

Mariana Oliveira

Janaina Miranda Xavier

Matheus Almeida

welligton dos passos silva

Madianita Bogo Marioti

Cargo/Ocupação:

29 respostas

Acionista

Gerente Administração e Finanças

Gestor de contas

Analista de Redes

Data Security Analyst

Financeiro

Analista de Suporte a Gestão

Suporte Técnico

servidor público/analista controle interno

Analista Técnico - Sebrae/TO

Funcionário público.

Analista de Processos

Taxista

Gerente Regional Sudeste Sebrae Tocantins

Analista Técnico

Gerente

Profissional da educação física

Engenheiro Civil

Desenvolvedora Web

Empreendedor

Arquiteto de Software

Estudante de Contabilidade

Analista e Desenvolvedora de Sistemas

Medico Veterinário

Analista

Analista Técnico - Sebrae

Desenvolvedor Node

assistente administrativo

Professora

Caso deseje, utilize o espaço a seguir para deixar suas observações:

6 respostas

Sou gerente do departamento de transportes e o uso dessa ferramenta é de extrema importância para o controle dos veículos, identificação de condutores, status e localização de veículos, facilitando a gestão, manutenções e resguardando a segurança dos nossos colaboradores.

sem considerações.

Obrigado

Priorizei as funcionalidades partindo da minha percepção de complexidade de desenvolvimento, e também do que poderia ser contemplado no MVP do Software.

Considero que um app mobile poderia ser uma estratégia vantajosa. No entanto, no cenário em que o público-alvo abrange usuários que realizam reservas de veículos de forma ocasional, é possível a implementação de uma plataforma web, especialmente se tiver uma interface responsiva semelhante à de um app mobile, tende a ser mais propícia e amplamente adotada pelos usuários.

todos tópicos são importante, tanto pela gestão, controle, pois são importantes para o gestor saber quem utilizou os veículos, para resguardar a empresa por possíveis multas de trânsito e também ter o controle da disponibilidade de veículos, e também com essas informações levantar a necessidade de adquirir novos veículos para suprir a demanda.