UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DIANA GOMES COUTINHO
THAIS CRISTINA BERTOLDO

SISTEMA PARA GERENCIAMENTO DE LOCADORAS DE VEÍCULOS

PATO BRANCO
2017

DIANA GOMES COUTINHO
THAIS CRISTINA BERTOLDO

SISTEMA PARA GERENCIAMENTO DE LOCADORAS DE VEÍCULOS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso II, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA PARA GERENCIAMENTO DE LOCADORAS DE VEÍCULOS

por

DIANA GOMES COUTINHO
THAIS CRISTINA BERTOLDO

Este trabalho de conclusão de curso foi apresentado no dia 05 de julho de 2017, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Profª Drª Beatriz Terezinha Borsoi
Orientador

Profª Me. Andreia Scariot Beulke

Profª Esp. Adriana Ariati

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª Drª Beatriz Terezinha Borsoi
Responsável pela Atividade de Trabalho de Conclusão de Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

AGRADECIMENTOS

Agradecemos primeiramente a Deus, por nos dar saúde e energia para concluir o curso e o TCC.

Agradecemos à professora Beatriz Borsoi por toda a contribuição e incentivo durante o curso.

Agradecemos a todos os professores que nos proporcionaram um bom aprendizado durante nossa vida acadêmica, especialmente ao professor Vinícius Pegorini, que nos auxiliou no desenvolvimento deste trabalho.

Agradecemos aos nossos pais pelo apoio e incentivo nesses anos de luta.

Eu, Diana, agradeço ao meu namorado Darlan pelo amor e incentivo prestado antes da escolha do curso, ao qual foi responsável, e depois por não me deixar desistir. Agradeço a minha colega de curso, de trabalho e de vida, Thais, que além de tudo é minha grande amiga.

Eu, Thais, agradeço a minha amiga Diana por todo apoio e compreensão. Agradeço também a minha prima Yascara pelas sugestões e melhorias durante o desenvolvimento do projeto.

Agradecemos a todas as pessoas que fizeram parte desta etapa de nossas vidas.

RESUMO

BERTOLDO, Thais Cristina; COUTINHO, Diana Gomes. Sistema para gerenciamento de locadoras de veículos. 2017. 74f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

A locação de veículos é uma atividade que é realizada por diversos motivos como, por exemplo, trabalho e lazer. Locações relacionadas ao trabalho ocorrem quando pessoas se deslocam para locais diferentes da localização da empresa e lá precisam utilizar veículos. A locação pode estar relacionada ao lazer, quando a viagem até o destino é realizada por meio aéreo, trem ou metrô, por exemplo, e veículos podem ser necessários para deslocamentos no local de destino ou para complementar o deslocamento para chegar ao local de destino. Dados estatísticos sustentam que a locação de veículo é uma atividade que tem crescido no País e isso, consequentemente, aumenta a quantidade de empresas que prestam serviços de locação. Assim, aumenta a necessidade de controle dos veículos locados, a exigência por qualidade e quantidade dos requisitos dos sistemas utilizados para a escolha e locação dos veículos pelos clientes e a necessidade de um gerenciamento efetivo da empresa locadora para a redução de custos e a maximização de lucros, mantendo-as frente à concorrência. Considerando esse contexto de existência de um mercado de locação e de clientes mais exigentes, neste trabalho é apresentado o desenvolvimento de um sistema para gerenciamento de locadoras de veículos. O sistema é para *web* e com acesso a partir de dispositivos móveis. Por meio do sistema, o cliente poderá consultar opções e valores de locações, além de realizar reservas. E a empresa mais facilmente controlará os veículos locados, as condições de empréstimos e devolução de cada veículo, assim sendo possível realizar também o acompanhamento financeiro.

Palavras-chave: Locadoras de veículos. Sistema de gerenciamento de locadoras de veículos. Sistema web.

ABSTRACT

BERTOLDO, Thais Cristina; COUTINHO, Diana Gomes. Car rental system. 2017. 74f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

Leasing vehicles is an activity that is performed for various reasons such as work and leisure. Rent a car related to work occurs when people move to locations other than the location of the company and need to use vehicles there. The lease may be related to leisure, when the trip to the destination is carried out by air, train or subway, for example, and vehicles may be necessary for commuting at the destination or to complement the commute to reach the destination. Statistical data maintain that vehicle leasing is an activity that has grown in the country and this, consequently, increases the number of companies that provide rental services. This increases the need for control of leased vehicles, the requirement for quality and quantity of the requirements of the systems used to choose and lease the vehicles by the customers and the need for effective management of the rental company for cost reduction and maximization of profits, keeping them against the competition. Considering this context of existence of a leasing market and more demanding clients, this work presents the development of a system for management of car rental companies. The system is for web and with access from mobile devices. Through the system, the customer can consult options and values of locations, as well as make reservations. And the company will more easily control the leased vehicles, the conditions of loans and the return of each vehicle, thus also being able to carry out the financial monitoring.

Keywords: Rental car. Car rental management system. Web system.

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTAGENS DE CÓDIGO

LISTA DE ABREVIATURAS E SIGLAS

ABLA	Associação Brasileira das Locadoras de Automóveis
ABS	*Antilock Braking System*
AJAX	*Asynchronous Javascript and XML*
CNH	Carteira Nacional de Habilitação
CPF	Cadastro de Pessoas Físicas
DER	Diagrama de Entidade e Relacionamentos
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
JSP	*Java Server Pages*
RG	Registro Geral
RIA	*Rich Internet Application*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa da realização deste trabalho. No final do capítulo é apresentada a organização do texto por meio de uma breve apresentação dos seus capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

Entre os motivos de locação de veículos podem ser destacados os relacionados ao trabalho e ao lazer. Há ainda outros motivos como os relacionados à saúde, quando as pessoas se deslocam para outros locais para tratamento, para ajudas humanitárias e eventos religiosos.

Em termos de trabalho, a locação pode ser realizada para atender necessidade de deslocamento dentro da cidade que o locatário se encontra, especialmente se é uma cidade de grande porte, ou quando há deslocamento para outras cidades, Estados e Países. Nesses casos, geralmente o deslocamento até o destino foi realizado por outro meio de transporte como, por exemplo, avião, mas no destino é necessário utilizar um veículo.

As empresas podem beneficiar-se da locação de veículos quando funcionários se deslocam para outras cidades (do mesmo ou de outro Estado), entre matriz e filiais, entre filiais de uma empresa ou para visitar clientes e fornecedores. Com a locação de veículos, os custos diminuem significativamente em relação ao pagamento de táxis, despesas com manutenção de veículos próprios e, além disso, o aluguel para que funcionários possam deslocar-se garante autonomia e autenticidade à empresa (RENT A CAR, 2016).

Quando a locação está relacionada a lazer, pode trazer mais conforto e facilidade de deslocamento para quem está de viagem de férias, por exemplo. Nesse tipo de viagem as pessoas podem beneficiar-se de um veículo locado para o deslocamento entre os pontos de passeio, especialmente se são viagens que envolvem pontos turísticos naturais e que podem ser distantes da cidade que foi o ponto de chegada e se há pontos distantes para serem visitados. É relativamente comum em viagens que possuem como destino locais turísticos que haja necessidade de deslocamentos longos.

O mercado de locação de automóveis, sem identificar a razão da locação, tem movimentado valores bastante expressivos, como mostram os dados apresentados na Associação Brasileira de Locadoras de Automóveis (Tabela 1). Nessa tabela estão dados

sobre o faturamento anual, frota, usuários, emprego e impostos gerados pelo setor de locadoras de automóveis.

Tabela 1- Locação de Automóveis – 2001 a 2013

Ano	Faturamento (em bilhões de R$)	Frota do setor	Usuários (em milhões de R$)	Geração de Empregos (diretos e indiretos)	Geração de impostos (em milhões de R$)
2001	1,89	155.000	7,3	144.000	0,58
2002	2,26	178.000	8,3	165.000	0,67
2003	2,35	181.900	8,7	165.500	0,7
2004	2,68	203.650	10,1	168.200	0,79
2005	2,91	223.811	12,2	178.240	0,87
2006	3,17	250.204	14,1	185.560	0,94
2007	3,49	283.562	15,1	194.838	1,06
2008	3,99	318.865	16,2	209.061	1,27
2009	4,37	363.456	16,8	240.644	1,44
2010	5,11	414.340	17,7	264.708	1,69
2011	5,67	445.470	18,6	277.943	1,86
2012	6,23	489.548	20,2	293.715	2,05
2013	6,52	529.890	21,7	309.017	2,15

Fonte: Associação Brasileira de Locadoras de Automóveis (2016a, p. 1).

De acordo com a Associação Brasileira das Locadoras de Automóveis (ABLA), em 2015 havia 7.455 empresas locadoras de veículos, somando 8.626 pontos de locação e gerando um faturamento de R$ 16.275,69 bilhões, além de proporcionar 472.113 empregos diretos e indiretos, conforme pode ser verificado na Figura 1.

Figura 1 - Estatísticas 2015
Fonte: Associação... (2016b, p.24-25).

No Brasil o setor de locação de veículos é bastante pulverizado, com empresas locadoras de pequeno, médio e grande porte atuando no mercado. Aproximadamente 10% das empresas do setor possuem faturamento anual superior a R$ 15 bilhões, enquanto 26% faturam entre R$ 300 mil e R$ 1 milhão por ano (ABLA, 2016). Na Figura 2 é apresentada esta divisão com base no faturamento anual das locadoras que atuam no Brasil.

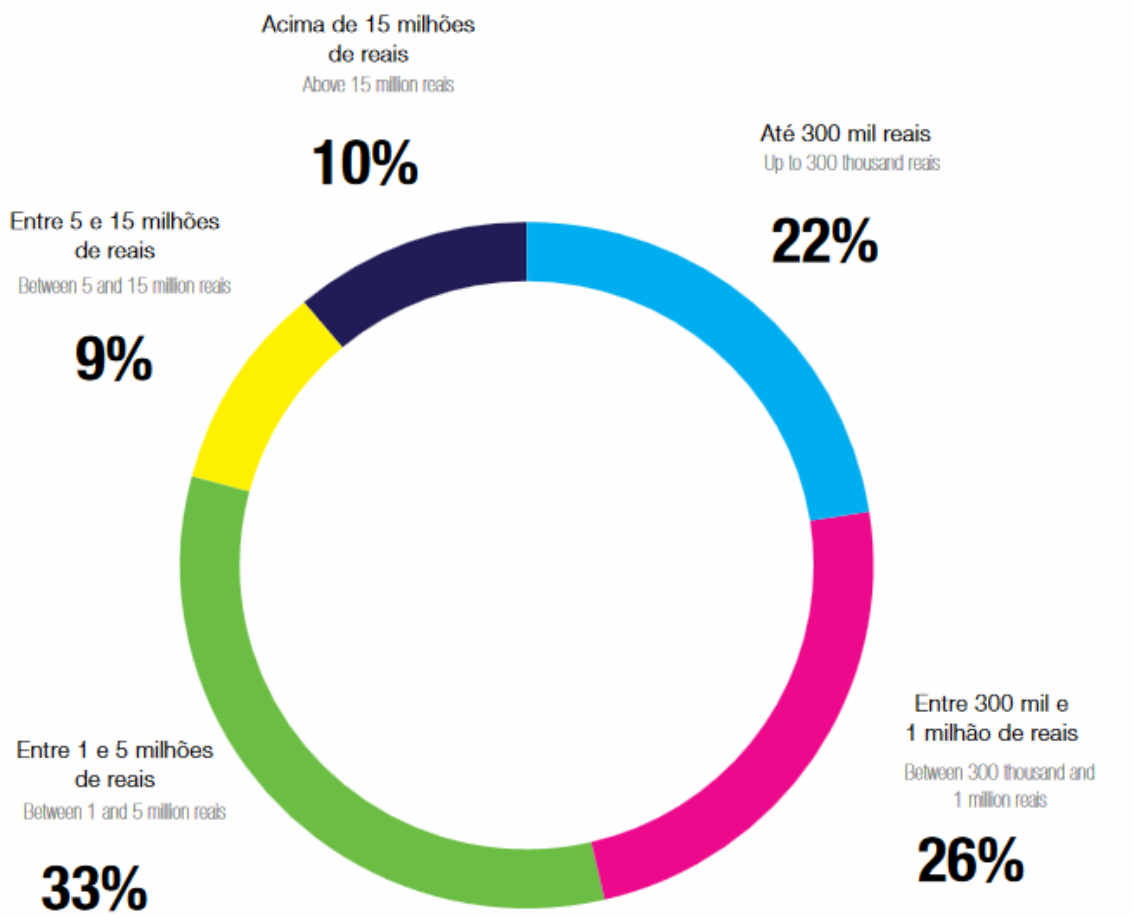

Figura 2 - Faixas de faturamento anual
Fonte: Associação... (2016b, p. 26).

O volume movimentado fornece uma ideia do mercado de locação de veículos. E no processo de locação de veículos há diversos profissionais envolvidos: gerentes, atendentes, pessoal de limpeza e manutenção dos veículos. O processo de locação requer o registro de vários dados, como os relacionados ao locador, ao período e à forma de locação, ao tipo de veículo e recursos disponíveis e sobre seguros, dentre outros. O grande volume de dados e os controles necessários tornam propício que esse tipo de negócio seja controlado por um sistema informatizado. Visando agilizar as tarefas do locatário, por meio do desenvolvimento

deste trabalho será implementado um sistema que tem o objetivo de auxiliar no gerenciamento das locações e devoluções de veículos bem como de fornecer uma versão *mobile* para maior praticidade e comodidade ao locador.

1.2 OBJETIVOS

O objetivo geral está relacionado ao resultado principal que é esperado da realização deste trabalho. Os objetivos específicos complementam o objetivo geral em termos de funcionalidades do sistema.

1.2.1 Objetivo Geral

Implementar um sistema *web*, complementado por versão *mobile*, para gerenciamento das atividades de locadoras de veículos.

1.2.2 Objetivos Específicos

- Facilitar a locação de veículos pelo cliente com acesso ao sistema por meio da plataforma *mobile*.
- Agilizar a devolução dos veículos pela possibilidade de realizar a verificação do veículo por meio de um *checklist*.
- Melhorar o gerenciamento da empresa pela disponibilidade de relatórios e gráficos com informações relacionadas às movimentações financeiras e outras.

1.3 JUSTIFICATIVA

A escolha por complementar o sistema com um aplicativo de acesso *mobile* visa facilitar o uso, especialmente, do cliente. Porque o cliente pode realizar locações e acompanhar o *status* das suas locações por meio de um dispositivo móvel Os atendentes da locadora também poderão realizar a verificação dos itens do veículo (por exemplo: vidros,

para-choques e pintura) antes da entrega e após a devolução por meio do sistema *web*. Nele serão apresentadas checagem de itens do veículo e o atendente marca os pontos de verificação e faz observações caso haja falta de conformidade no momento da devolução ou alguma avaria no momento da entrega para o cliente.

O sistema possibilitará o cadastramento de clientes, fornecedores, funcionários e veículos, entre outros. Além do gerenciamento de reservas, locações e devoluções. Também será possível realizar o gerenciamento administrativo e financeiro da empresa e para isso o sistema disponibilizará o controle de receitas (locações) e despesas (manutenção de veículos e pagamento de funcionários). O sistema terá relatórios financeiros e administrativos para controles recorrentes, além de outros documentos necessários para a locação e devolução do veículo, como contratos, comprovantes de reservas e emissão de recibos de pagamento. O sistema disponibilizará gráficos das principais informações do sistema.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. O Capítulo 2 apresenta o referencial teórico que é sobre aplicações Internet denominadas como ricas. O sistema desenvolvido como resultado deste trabalho é para ambiente Internet e utiliza tecnologias e conceitos que o caracterizam como *Rich Internet Applicaiton* (RIA). No Capítulo 3 são apresentados os materiais e o método utilizados para o desenvolvimento do trabalho. No Capítulo 4 está o resultado da realização do trabalho, que é a modelagem e a implementação do sistema obtido como resultado da realização deste trabalho. Por fim estão as considerações finais, seguidas das referências utilizadas no texto.

2 APLICAÇÕES INTERNET RICAS

Diferentemente das aplicações *web* baseadas em tags *HyperText Markup Language* (HTML) e códigos de *script* como elementos básicos de uma aplicação *web*, uma nova geração de aplicações para *web* integram novas tecnologias que proporcionam alto grau de interatividade (AGUSTIN, 2015). Entre essas novas tecnologias estão *Asynchronous Javascript and XML* (AJAX) (GARRET, 2005) e *Rich Internet Applications* (RIA) (PAULSON, 2005) suportando amplamente o novo tipo de aplicação para a Internet.

RIAs são aplicações cliente/servidor que fazem a convergência de duas culturas de desenvolvimento concorrentes: *desktop* e aplicações *web* (MELIÁ et al., 2010). Para esses autores elas provêm a maioria dos benefícios de desenvolvimento e manutenibilidade das aplicações *web* ao mesmo tempo em que elas suportam um cliente com interface com o usuário rica. E, além disso, as RIAs introduzem novas características arquiteturais no campo das aplicações *web* tradicionais, como, por exemplo, a manutenção de estado da interface com estados conectado e desconectado e comunicação cliente/servidor inteligente com requisições assíncronas.

Pang et al. (2010) destacam que as RIAs promovem a integração da semântica tradicional do hipertexto com as aplicações *desktop*. Para esses autores as aplicações *desktop* possuem vantagens em relação à interface com o usuário e as aplicações *web* possuem características de distribuição rápida e de baixo custo e comunicação multimídia em tempo real, mas as RIAs integram as melhores vantagens das aplicações *desktop* e *web*.

Tecnologias posteriores ao padrão de tecnologias HTML/http para desenvolvimento *web* estão moldando a *web*, entre essas tecnologias estão as RIAs que desempenham um papel proeminente nessa nova geração de aplicações *web* (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010). Para esses autores RIA refere-se a uma heterogênea família de soluções, caracterizadas por um objetivo comum de adicionar novas potencialidades às aplicações *web* baseadas em hipertexto.

Classificação das características do lado cliente de aplicações RIA (MELIÁ et al., 2010):

a) Trabalho *offline* – possibilidade de trabalhar enquanto desconectado por meio de *download* de lógica de negócio e dos dados no lado cliente. Essa característica requer que mecanismo de armazenamento no lado cliente e lógica de negócio definida e implementada para isso.

b) Armazenamento – RIAs provêem novas facilidades de manipulação dos dados armazenados no lado cliente que vem do servidor. Esse armazenamento pode ser persistente ou volátil.
c) *Cache* – cliente RIA tem a habilidade de manter informação do servidor durante um período de tempo melhorando o desempenho da aplicação e responsividade da interface com o usuário. O cache dessas aplicações possui um atributo *fetching* que suporta valores *immediate* or *lazzy*.
d) Lógica de negócio - o cliente RIA possui uma capacidade de processo melhorado permitindo desempenhar processos complexos. Essa característica possui um atributo *location* que determina se a lógica de negócio é do cliente, do servidor ou uma mistura de ambos.
e) Manipulação de eventos – RIAs possuem uma coreografia entre diferentes componentes de interface com o usuário que podem ser sincronizadas com "*bus*" de eventos centralizados ou com um padrão *observer* descentralizado.
f) Validação – RIA podem conter regras de validação para entradas de usuário bem como para regras de negócio no cliente.
g) *Template* – possibilidade de suportar a criação de visões e leiaute de apresentação em tempo de execução.
h) Plataforma – determina qual a plataforma usada para implementar a camada cliente.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a realização deste trabalho. Os materiais estão relacionados às tecnologias e ferramentas utilizadas e o método apresenta a sequência das principais atividades realizadas.

3.1 MATERIAIS

O Quadro 1 apresenta as ferramentas e as tecnologias que foram utilizadas para modelar e implementar o sistema.

Ferramenta / Tecnologia	**Versão**	**Referência (site)**	**Finalidade**
Vertabelo		http://www.vertabelo.com/	Modelagem Banco de Dados
Astah Community	7.1	http://astah.net/download	Modelagem *Unified Modeling Language* (UML)
Java	8	http://www.java.com/pt_BR/download/win10.jsp	Linguagem de programação
Eclipse	Mars.2	http://www.eclipse.org/downloads	Ambiente de desenvolvimento para o sistema *web*
PostgreSQL	9.3	http://www.postgresql.org/download/windows/	Banco de Dados
PgAdmin	1.18.1	http://www.postgresql.org/ftp/pgadmin3/release/v1.18.1/win32/	Administrador do banco de dados
Android Studio	1.5.1	http://tools.android.com/download/studio/builds/1-5-1	Ambiente de desenvolvimento para o aplicativo Android.
Bootstrap		http://getbootstrap.com/	Framework web front-end para design de aplicações web para desenvolver a interface da aplicação.
Tomcat		http://tomcat.apache.org/	Servidor de Aplicação
Trello		https://trello.com/	Gerenciamento de Tarefas
Spring		https://spring.io/	*Framework* de desenvolvimento
DataTables		https://datatables.net/	Plug-in para a biblioteca jQuery JavaScript
Tortoise SVN	1.9.3	https://tortoisesvn.net/downloads.html	Versionamento de código.
Maven		https://maven.apache.org/	Gerenciamento de dependências

Quadro 1 – Ferramentas e tecnologias utilizadas

3.2 MÉTODO

Na fase inicial do levantamento de requisitos foram pesquisados trabalhos acadêmicos sobre sistemas semelhantes, visitados sites de locadoras e conversado com funcionários do ramo, buscando entender os processos e atividades realizadas na locadora. Assim, os requisitos foram definidos e organizados em funcionais e não funcionais.

Na fase de análise do sistema tendo como base os requisitos pré-estabelecidos, definiram-se os casos de uso do sistema. Esses casos de uso foram documentados gerando informações para a modelagem do banco de dados.

Após a modelagem dos casos de uso, foi elaborado o diagrama de entidades e relacionamentos do banco gerando as tabelas com seus campos, tipos, tamanhos e relacionamentos.

A implementação foi realizada utilizando o *Integrated Development Environment* (IDE) Eclipse para o desenvolvimento do sistema *web* e Android Studio para o aplicativo *mobile*.

Os testes realizados foram informais, sem um plano de testes específico, e visaram a identificação de erros de codificação e a verificação do atendimento dos requisitos definidos e modelados.

4 RESULTADO

Este capítulo apresenta o resultado deste trabalho que é o desenvolvimento de um sistema *web* complementado por um aplicativo *mobile* com o objetivo de auxiliar no gerenciamento de locadoras de veículos.

4.1 ESCOPO DO SISTEMA

O sistema para gerenciamento de locadoras de veículos possibilitará o cadastro de clientes, fornecedores, funcionários, veículos, entre outros. Além do gerenciamento de reservas, locações e devoluções, o sistema também possibilitará o gerenciamento administrativo e financeiro da empresa.

No cadastro de clientes será necessário informar os dados pessoais como Cadastro de Pessoas Físicas (CPF), Registro Geral (RG), endereço, Carteira Nacional de Habilitação (CNH), *e-mail*, senha de acesso, entre outros.

O cadastro de fornecedores disponibilizará os campos de dados de identificação e contatos da prestadora de serviços e produtos, para posteriormente possibilitar o lançamento de despesas.

No cadastro de funcionários serão armazenados dados pessoais e profissionais, além de campos de configurações de acesso ao sistema, como dados de *login* e senha.

No cadastro de veículos será necessário informar o grupo de classificação (exemplo: utilitário, passeio, familiar, entre outros) previamente cadastrados, além dos dados de identificação do veículo (placa, cor, chassi, renavan e foto). Esse cadastro também possuirá dados do veículo referentes à existência de *Global Positioning System* (GPS), cadeiras e assentos de elevação, entre outros. E haverá também o gerenciamento de quilometragem, nível de combustível e disponibilidade.

As reservas poderão ser realizadas pelo aplicativo *mobile* e pelo sistema *web*. Para realizar reservas, o cliente deverá fazer seu cadastro, selecionar as opções de reserva, tipo de seguro e período de locação.

Para efetuar a locação, o cliente deverá ter o seu cadastro no sistema aprovado pelos critérios estabelecidos pela empresa, fazer as escolhas dos adicionais e entre as opções de seguro oferecidas. A locação será efetuada apenas se o funcionário possuir as permissões de acesso ao sistema. E para realizar a devolução do veículo, o responsável deverá entregá-lo nas

mesmas condições de recebimento, que serão avaliadas por uma vistoria aprovada pela empresa com o uso de um *checklist* fornecido pelo sistema. Se o veículo não estiver nas condições esperadas, ele será enviado para manutenção e os custos serão acrescidos ao valor final da locação. O veículo somente será disponibilizado novamente para locação após o seu retorno à empresa e as informações decorrentes de uso, como quilometragem, deverão ser atualizadas.

O sistema possui, ainda, funcionalidade relacionada ao gerenciamento de receitas (locações, reservas, entre outros) e despesas (manutenção de veículos, pagamento de funcionários, entre outros). Podendo ser atribuídos descontos e categorias de classificação definidas pela empresa, a fim de facilitar o controle de caixa.

O sistema emitirá relatórios financeiros e administrativos, sendo possível a visualização de gráficos das principais informações armazenadas. Além desses relatórios, o sistema emitirá documentos necessários para a locação e a devolução do veículo, como contratos, comprovantes de reservas, emissão de recibos de pagamento, entre outros.

4.2 MODELAGEM DO SISTEMA

Os Quadros 2 a 25 apresentam a descrição dos requisitos funcionais e os respectivos requisitos não funcionais relacionados.

F1 - Manter clientes				
Descrição: O sistema deve permitir o cadastro de clientes contendo nome, telefone, CPF/CNPJ e as demais informações pessoais. Além disso, deve ser permitido editar e excluir cadastros.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF1.1 Validar CPF/CNPJ	O sistema deve executar uma função de validação do CPF/CNPJ.	Regra de negócio	( )	(X)
NF1.2 Autenticar o número do CPF	O sistema não deve permitir que seja cadastrado CPF igual, para não haver duplicação.	Regra de Implementação	( )	(X)
NF1.3 Validar campo de data	O sistema deve permitir apenas a inserção de data no formato válido.	Interface	(X)	( )

Quadro 2 - Requisito manter clientes

F2 - Manter funcionários				
Descrição: O sistema deve permitir o cadastro de usuários. Além disso, deve ser permitido editar e inativar esses cadastros.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF2.1 Permissões	O formulário de inclusão deve disponibilizar em sua interface uma opção de seleção do tipo de usuário, que pode ser funcionário ou administrador.	Interface	( )	(X)
NF2.2 Excluir	O sistema não deve permitir a exclusão de usuários com vínculos no sistema, apenas a inativação destes cadastros.	Integridade	( )	(X)

Quadro 3 - Requisito manter funcionários

F3 - Manter fornecedores				
Descrição: O sistema deve oferecer ao funcionário a possibilidade de cadastrar os fornecedores de serviços. Além disso, deve ser permitido editar e excluir cadastros.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF3.1 Cadastros	O formulário de cadastro de fornecedor deve disponibilizar em sua interface campos para o endereço e contatos do fornecedor.	Interface	( )	(X)

Quadro 4 - Requisito manter fornecedores

F4 Manter reservas				
Descrição: O cliente poderá fazer uma reserva apenas se estiver previamente cadastrado.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF4.1 Aviso de cliente inválido	O sistema deve avisar e bloquear a reserva se o cliente não estiver cadastrado.	Regra de Implementação	( )	(X)

Quadro 5 - Requisito manter reservas

F5 Manter veículos				
Descrição: O sistema deve oferecer ao usuário a possibilidade de cadastrar os veículos. Além disso, deve ser permitido editar e excluir os cadastros.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF7.1 Cadastros	O formulário de cadastro de veículos deve permitir inserir a placa, o modelo, cor e ano.	Interface	(X)	( )
NF7.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de veículos com a mesma placa.	Integridade	( )	(X)

Quadro 6 - Requisito manter veículos

F6 Manter cidades				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de cidades para serem utilizadas nos cadastros de clientes, funcionários e fornecedores.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF8.2 Excluir	O sistema não deve permitir a exclusão de cidades utilizadas em cadastros.	Integridade	( )	(X)

Quadro 7 - Requisito manter cidades

F7 Manter cargos				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de cargos de funcionários.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF10.1 Excluir	O sistema não deve permitir a exclusão de cargos utilizados em cadastros de funcionários.	Integridade	( )	(X)
NF10.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de cargos com mesmo nome.	Integridade	( )	(X)

Quadro 8 - Requisito manter cargos

F8 Manter tipos de fornecedores				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de tipos de fornecedores.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF11.1 Excluir	O sistema não deve permitir a exclusão de tipos de fornecedores com vínculos.	Integridade	( )	(X)
NF11.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de tipos de fornecedores com mesmo nome.	Integridade	( )	(X)

Quadro 9 - Requisito manter tipos de fornecedores

F9Manter seguros				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de seguros oferecidos nas locações.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF12.1 Excluir	O sistema não deve permitir a exclusão de seguros com vínculos.	Integridade	( )	(X)
NF12.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de seguros com mesmo nome.	Integridade	( )	(X)

Quadro 10 - Requisito manter seguros

F10 Manter grupos de veículos				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de grupos de veículos.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF13.1 Excluir	O sistema não deve permitir a exclusão de grupos de veículos com vínculos.	Integridade	( )	(X)
NF13.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de grupos de veículos com mesmo nome.	Integridade	( )	(X)

Quadro 11 - Requisito manter grupos de veículos

F11 Manter receitas				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e quitação de parcelas a receber para clientes cadastrados. Além disso, deve ser permitido editar essas parcelas.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF14.1 Incluir	O formulário de inclusão deve disponibilizar em sua interface campos para selecionar o cliente, valor, data de vencimento, categoria do plano de contas e forma de cobrança.	Interface	( )	(X)

Quadro 12 - Requisito manter receitas

F12 Manter despesas				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e quitação de parcelas a pagar para fornecedores ou funcionários cadastrados. Além disso, deve ser permitido editar essas parcelas.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF15.1 Incluir	O formulário de inclusão deve disponibilizar em sua interface campos para selecionar o fornecedor ou funcionário, valor, data de vencimento, categoria do plano de contas e forma de cobrança.	Interface	( )	(X)

Quadro 13 - Requisito manter despesas

F13 Manter tipos de cobranças e recebimentos				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de tipos de cobranças e recebimentos.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF16.1 Excluir	O sistema não deve permitir a exclusão de tipos de cobranças e recebimentos com vínculos.	Integridade	( )	(X)

NF16.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de tipos de cobranças e recebimentos com mesmo nome.	Integridade	( )	(X)

Quadro 14 - Requisito manter tipos de cobranças e recebimentos

F14 Manter tipos de cobranças extras				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de tipos de cobranças extras.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF17.1 Excluir	O sistema não deve permitir a exclusão de tipos de cobranças extras com vínculos.	Integridade	( )	(X)
NF17.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de tipos de cobranças e recebimentos com mesmo nome.	Integridade	( )	(X)

Quadro 15 - Requisito manter tipos de cobranças extras

F15 Manter tipos de adicionais				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de tipos de adicionais contratados na locação.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF19.1 Excluir	O sistema não deve permitir a exclusão de tipos de adicionais com vínculos.	Integridade	( )	(X)
NF19.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de tipos de adicionais com mesmo nome.	Integridade	( )	(X)

Quadro 16 - Requisito manter tipos de adicionais

F16 Manter itens do *checklist*				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de itens do *checklist*.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF20.1 Excluir	O sistema não deve permitir a exclusão itens do *checklist* com vínculos.	Integridade	( )	(X)
NF20.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de itens do *checklist* com mesmo nome.	Integridade	( )	(X)

Quadro 17- Requisito manter itens do *checklist*

F17 Manter plano de contas				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer a inclusão, exclusão e edição de plano de contas.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF21.1 Excluir	O sistema não deve permitir a exclusão itens do plano de contas associadas a receitas e despesas.	Integridade	( )	(X)
NF21.2 Validar a duplicidade de cadastro	O sistema não deve permitir o cadastro de plano de contas com mesmo nome.	Integridade	( )	(X)

Quadro 18 - Requisito manter plano de contas

F18 Manter caixa				
Descrição: O sistema deve oferecer ao usuário a possibilidade de fazer exclusão de lançamentos do caixa.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF22.1 Excluir	O sistema deve permitir o estorno de lançamentos referentes a quitação de receitas e despesas.	Integridade	( )	(X)

Quadro 19 - Requisito manter caixa

F19 Manter log
Descrição: O sistema deve possuir um armazenamento de log de trabalhos, que armazene as ações efetuadas nos principais módulos.

Quadro 20 - Requisito manter log

F20 Gerar relatórios
Descrição: O sistema deve oferecer ao usuário administrador a possibilidade de gerar modelos de relatórios dos principais módulos.

Quadro 21 - Requisito manter relatórios

O diagrama de casos de uso apresentado na Figura 3 contém as funcionalidades essenciais do sistema realizadas pelos seus atores que são: clientes, funcionários e administrador. O administrador é responsável pelos cadastros de fornecedores, veículos, controle de despesas e gerenciamento da empresa. O funcionário é quem realiza o processo de cadastro de clientes, reservas, locações e devoluções de veículos, bem como o gerenciamento de recebimentos. O cliente poderá efetuar seu cadastro por meio do aplicativo e reservar veículos.

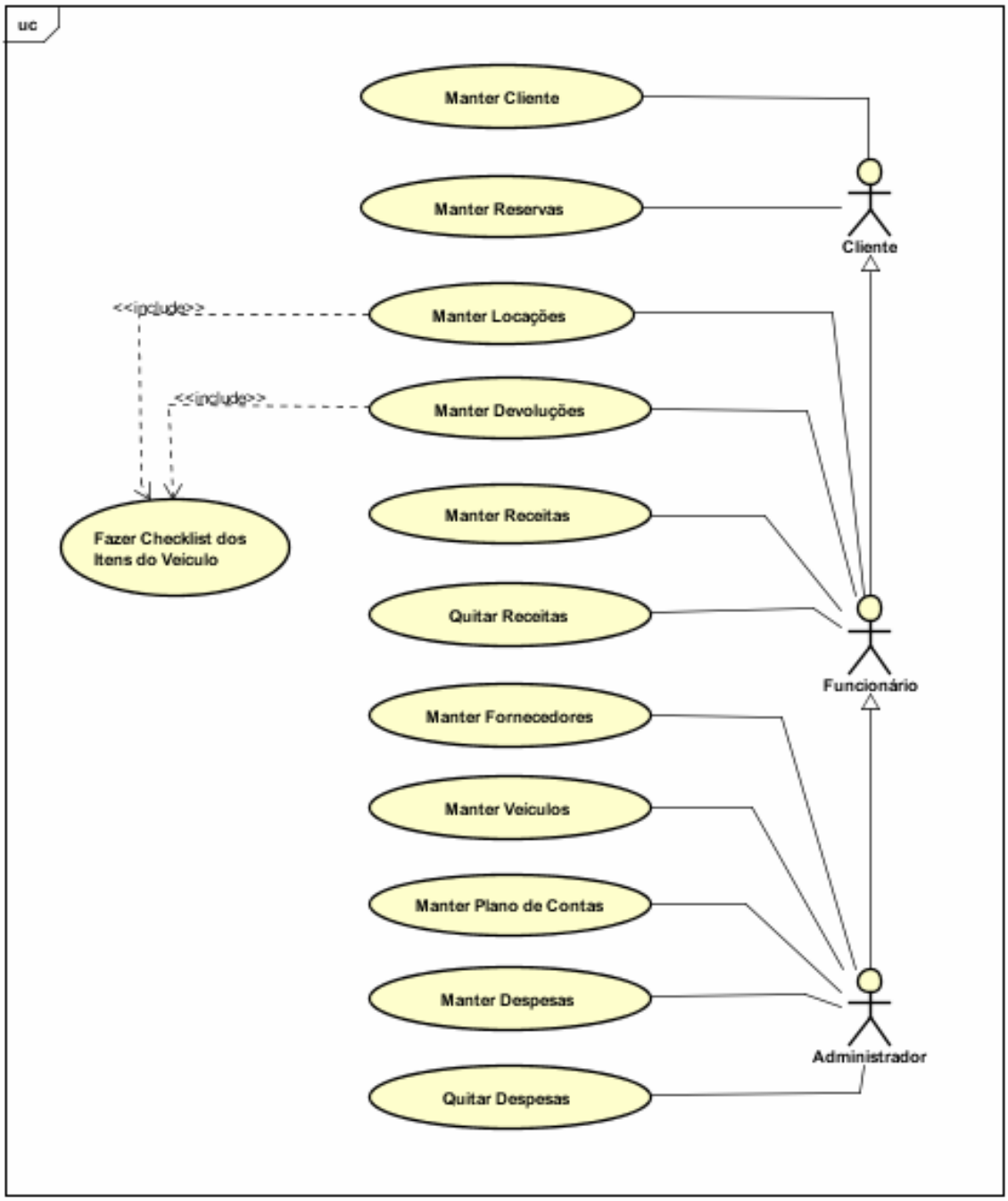

Figura 3 - Diagrama de casos de uso

Os Quadros 22 a 27 apresentam a descrição do caso de uso cadastrar. Essa descrição se refere à operação de inclusão de todos os casos de usos identificados como “manter”.

Caso de uso:

Incluir (refere-se à operação de inclusão de todos os casos de usos identificados como "manter").

Descrição:

Inclusão dos dados cadastrais no sistema.

Evento Iniciador:

Ator solicita inclusão de um registro no sistema.

Atores:

Cliente, funcionário ou administrador de acordo com suas funções definidas no caso de uso.

Pré-condição:

Não há.

Sequência de Eventos:

1. Ator acessa a tela para realizar o cadastro inserindo as informações necessárias.
2. O sistema insere as informações no banco de dados e informa ao usuário o *status* do procedimento.

Pós-Condição:

Registro inserido no banco de dados.

Extensões:

Informações provenientes de outros cadastros, como clientes, fornecedores, veículos e plano de contas não cadastrados.

Nome do fluxo alternativo (extensão)	Descrição
1. Campos obrigatórios não informados.	1.1. O cliente deixa de informar dados obrigatórios e clica em salvar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.
2. Campos informados em formato incorreto.	2.1. O cliente informa dados em um formato incorreto e clica em salvar. 2.2. O sistema valida que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.

Quadro 22 - Operação "incluir" dos casos de uso de cadastro

Caso de uso:

Alterar (refere-se à operação de alteração de todos os casos de usos identificados como "manter").

Descrição:

Alteração dos dados cadastrais no sistema.

Evento Iniciador:

Ator solicita alteração de um registro no sistema.

Atores:

Cliente, funcionário ou administrador de acordo com suas funções definidas no caso de uso.

Pré-condição:

Registro estar incluso no sistema.

Sequência de Eventos:

1. Ator acessa a tela para visualização dos dados do registro.
2. O sistema apresenta o registro selecionado para alteração.
3. Usuário altera os dados do registro.
4. O sistema altera as informações no banco de dados e informa ao usuário o *status* do procedimento.

Pós-Condição:

Registro alterado no banco de dados.

Extensões:

Informações provenientes de outros cadastros, como clientes, fornecedores, veículos e plano de contas não cadastrados.	
Nome do fluxo alternativo (extensão)	Descrição
1. Campos obrigatórios não informados.	1.1. O cliente apaga dados obrigatórios e clica em salvar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de edição mantendo as alterações realizadas.
2. Campos informados em formato incorreto.	2.1. O cliente altera os dados deixando em um formato incorreto e clica em salvar. 2.2. O sistema valida que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de edição mantendo as alterações realizadas.

Quadro 23 - Operação "alterar" dos casos de uso de cadastro

Caso de uso:
Excluir (refere-se à operação de exclusão de todos os casos de usos identificados como "manter").
Descrição:
Exclusão dos dados cadastrais no sistema.
Evento Iniciador:
Ator solicita exclusão de um registro no sistema.
Atores:
Funcionário ou administrador de acordo com suas funções definidas no caso de uso.
Pré-condição:
Registro estar incluso no sistema.
Sequência de Eventos:
1. Ator acessa a tela para exclusão do registro.
2. O sistema exclui as informações no banco de dados e informa ao usuário o *status* do procedimento.

Pós-Condição:
Registro excluído no banco de dados.
Extensões:
Registro possuir vínculo com outros cadastros

Nome do fluxo alternativo (extensão)	Descrição
1. Exclusão de registro com vínculos no sistema	1.1. Cliente clica em excluir um registro que possui vínculos no sistema. 1.2. O sistema verifica que o registro tem vínculos, não o exclui e exibe mensagem de alerta ao usuário.

Quadro 24 - Operação "excluir" dos casos de uso de cadastro

Caso de uso:
Consultar (refere-se à operação de consulta de todos os casos de usos identificados como "manter").
Descrição:
Consulta dos dados cadastrais dos registros do sistema.
Evento Iniciador:
Ator solicita consulta de um registro no sistema.
Atores:
Cliente, funcionário ou administrador de acordo com suas funções definidas no caso de uso.

Pré-condição: Registro estar incluso no sistema. **Sequência de Eventos:** 1. Ator acessa a tela para visualização dos dados do registro. 2. O ator indica os filtros desejados para consulta. 3. O sistema apresenta os dados da consulta ao usuário. **Pós-Condição:** Dados da consulta apresentados ao usuário.

Quadro 25 - Operação "consultar" dos casos de uso de cadastro

O Quadro 26 apresenta a descrição do caso de uso quitar parcelas. Essa descrição se refere à operação de quitação de parcelas em todos os casos de usos identificados como "quitar".

Caso de uso: Quitar (refere-se à operação de quitação de parcelas em todos os casos de usos identificados como "quitar"). **Descrição:** Quitar as parcelas inclusas no sistema. **Evento Iniciador:** Ator solicita a quitação de uma parcela do sistema. **Atores:** Funcionário ou administrador de acordo com suas funções definidas no caso de uso. **Pré-condição:** Registro estar incluso no sistema e não estar quitado. **Sequência de Eventos:** 1. Ator acessa a tela para quitação das parcelas. 2. O ator informa os dados para quitação (Ex: valor, data pagamento). 3. O sistema quita a parcela e informa ao usuário o *status* do procedimento. **Pós-Condição:** Parcela quitada.	
Nome do fluxo alternativo (extensão)	Descrição
1. Campos obrigatórios não informados.	1.1. O cliente deixa de informar dados obrigatórios e clica em quitar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de quitação mantendo os dados informados anteriormente.
2. Campos informados em formato incorreto.	2.1. O cliente informa dados em um formato incorreto e clica em quitar. 2.2. O sistema valida que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de quitação mantendo os dados informados anteriormente.

Quadro 26 - Operação "quitar" dos casos de uso

O Quadro 27 apresenta a descrição do caso de uso fazer *checklist* dos itens do veículo.

Caso de uso:

Fazer *checklist* (refere-se à operação de verificação dos itens do veículo que devem ser vistoriados ao efetuar e/ou devolver uma locação). **Descrição:** Informar situação de itens do veículo com base em uma lista cadastrada. Para os itens que não estiverem em conformidade é necessário descrever o motivo. **Evento Iniciador:** Ator solicita realização do *checklist*. **Atores:** Funcionário ou administrador de acordo com suas funções definidas no caso de uso. **Pré-condição:** Locação estar inclusa no sistema. **Sequência de Eventos:** 1. Ator acessa a tela para inclusão/edição do *checkist*. 2. O ator marca a situação dos itens e caso necessário informa o motivo. 3. O sistema salva o *checklist* e informa ao usuário o *status* do procedimento. **Pós-Condição:** *Checklist* inserido/editado no banco de dados.

Quadro 27 - Operação "fazer *checklist*" dos casos de uso

A Figura 4 representa o Diagrama de Entidades e Relacionamentos (DER) do banco de dados utilizando a PostgreSQL. Os campos de chave primária são do tipo serial para que sejam gerados automaticamente pelo banco de dados.

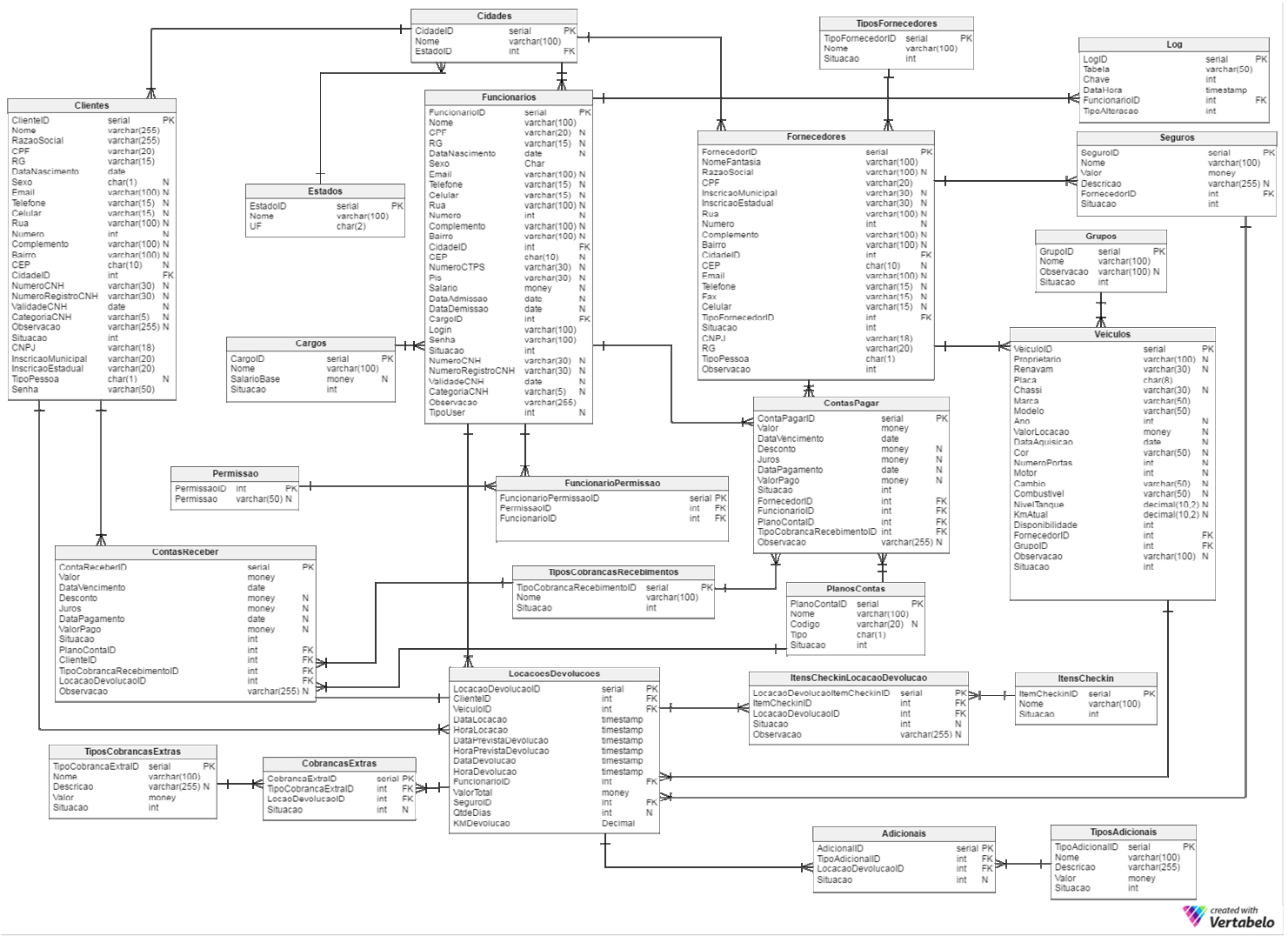

Figura 4 - Diagrama de entidades e relacionamentos

No Quadro 28 estão os campos da tabela de estados. Um estado poderá estar relacionado a uma ou muitas cidades.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
EstadoID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
UF	Char	Não	Não	Não

Quadro 28 - Campos da tabela Estados

No Quadro 29 estão os campos da tabela de cidades. Uma cidade poderá estar relacionada a um ou muitos clientes, funcionários ou fornecedores.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
CidadeID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
EstadoID	Integer	Não	Não	Sim

Quadro 29 - Campos da tabela Cidades

No Quadro 30 estão os campos da tabela de cargos. O cargo é utilizado no cadastro de funcionários.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
CargoID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
SalarioBase	Money	Sim	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 30 - Campos da tabela Cargos

No Quadro 31 estão os campos da tabela de tipos de fornecedores.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
TipoFornecedorID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 31 - Campos da tabela TiposFornecedores

No Quadro 32 estão os campos da tabela de tipos de cobranças e recebimentos. Um tipo de cobrança ou recebimento poderá estar relacionado a uma ou muitas contas a receber e a pagar.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
TipoCobrancaRecebimentoID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 32 - Campos da tabela TiposCobrancasRecebimentos

No Quadro 33 estão os campos da tabela de grupos de veículos. Um grupo poderá estar relacionado a um ou muitos veículos.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
GrupoID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Observacao	Varchar	Sim	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 33 - Campos da tabela Grupos

O Quadro 34 apresenta os campos da tabela de seguros de veículos. Um seguro poderá estar relacionado a um ou muitos veículos e será fornecido por apenas um fornecedor.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
SeguroID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Valor	Money	Não	Não	Não
Descricao	Varchar	Sim	Não	Não
FornecedorID	Integer	Não	Não	Sim
Situacao	Integer	Não	Não	Não

Quadro 34 - Campos da tabela Seguros

Os campos da tabela de tipos de adicionais das locações estão no Quadro 35. Um adicional poderá estar relacionado a uma ou muitas locações.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
TipoAdicionalID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Descricao	Varchar	Sim	Não	Não
Valor	Money	Não	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 35 - Campos da tabela TiposAdicionais

Um cliente poderá estar relacionado a uma ou muitas locações/devoluções, reservas e contas a receber. No Quadro 36 estão os campos da tabela de clientes.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
ClienteID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
RazaoSocial	Varchar	Sim	Não	Não
CPF	Varchar	Sim	Não	Não
CNPJ	Varchar	Sim	Não	Não
RG	Varchar	Sim	Não	Não
InscricaoMunicipal	Varchar	Sim	Não	Não
InscricaoEstadual	Varchar	Sim	Não	Não
DataNascimento	Date	Sim	Não	Não
Sexo	Char	Sim	Não	Não
TipoPessoa	Char	Sim	Não	Não
Email	Varchar	Sim	Não	Não
Telefone	Varchar	Sim	Não	Não
Celular	Varchar	Sim	Não	Não

Rua	Varchar	Sim	Não	Não
Numero	Integer	Sim	Não	Não
Complemento	Varchar	Sim	Não	Não
Bairro	Integer	Sim	Não	Não
CEP	Char	Sim	Não	Não
CidadeID	Integer	Sim	Não	Sim
NumeroCNH	Varchar	Sim	Não	Não
NumeroRegistroCNH	Varchar	Sim	Não	Não
ValidadeCNH	Date	Sim	Não	Não
CategoriaCNH	Varchar	Sim	Não	Não
Senha	Varchar	Sim	Não	Não
Observacao	Varchar	Sim	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 36 - Campos da tabela Clientes

No Quadro 37 estão os campos da tabela de funcionários. Um funcionário poderá estar relacionado a uma ou muitas locações/devoluções e contas a pagar.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
FuncionarioID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
CPF	Varchar	Sim	Não	Não
RG	Varchar	Sim	Não	Não
DataNascimento	Date	Sim	Não	Não
Sexo	Char	Sim	Não	Não
Email	Varchar	Sim	Não	Não
Telefone	Varchar	Sim	Não	Não
Celular	Varchar	Sim	Não	Não
Rua	Varchar	Sim	Não	Não
Numero	Integer	Sim	Não	Não
Complemento	Varchar	Sim	Não	Não
Bairro	Integer	Sim	Não	Não
CEP	Char	Sim	Não	Não
CidadeID	Integer	Sim	Não	Sim
NumeroCTPS	Varchar	Sim	Não	Não
Pis	Varchar	Sim	Não	Não
Salario	Money	Sim	Não	Não
DataAdmissao	Date	Sim	Não	Não
DataDemissao	Date	Sim	Não	Não
CargoID	Integer	Não	Não	Não
Login	Varchar	Sim	Não	Não
Senha	Varchar	Sim	Não	Não
Situacao	Integer	Não	Não	Não
NumeroCNH	Varchar	Sim	Não	Não
NumeroRegistroCNH	Varchar	Sim	Não	Não
ValidadeCNH	Date	Sim	Não	Não
CategoriaCNH	Varchar	Sim	Não	Não
Observacao	Varchar	Sim	Não	Não
TipoUser	Integer	Não	Não	Não

Quadro 37 - Campos da tabela Funcionarios

No Quadro 38 estão os campos da tabela de veículos. Um veículo poderá estar relacionado a uma ou muitas reservas e locações.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
VeiculoID	Serial	Não	Sim	Não
Proprietario	Varchar	Sim	Não	Não
Renavan	Varchar	Sim	Não	Não
Placa	Char	Sim	Não	Não
Chassi	Varchar	Sim	Não	Não
Marca	Varchar	Não	Não	Não
Modelo	Varchar	Não	Não	Não
Ano	Integer	Sim	Não	Não
ValorLocacao	Money	Sim	Não	Não
DataAquisicao	Date	Sim	Não	Não
Cor	Varchar	Sim	Não	Não
Motor	Integer	Sim	Não	Não
Cambio	Varchar	Sim	Não	Não
Combustivel	Varchar	Sim	Não	Não
NivelTanque	Decimal	Sim	Não	Não
KmAtual	Decimal	Sim	Não	Não
FornecedorID	Integer	Não	Não	Sim
GrupoID	Integer	Não	Não	Sim
Observacao	Varchar	Sim	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 38 - Campos da tabela Veiculos

No Quadro 39 estão os campos da tabela de fornecedores. Um fornecedor poderá estar relacionado a um ou muitos seguros, veículos e contas a pagar.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
FornecedorID	Serial	Não	Sim	Não
NomeFantasia	Varchar	Não	Não	Não
RazaoSocial	Varchar	Sim	Não	Não
CPF	Varchar	Sim	Não	Não
CNPJ	Varchar	Sim	Não	Não
InscricaoMunicipal	Varchar	Sim	Não	Não
InscricaoEstadual	Varchar	Sim	Não	Não
RG	Varchar	Sim	Não	Não
TipoPessoa	Char	Sim	Não	Não
Rua	Varchar	Sim	Não	Não
Numero	Integer	Sim	Não	Não
Complemento	Varchar	Sim	Não	Não
Bairro	Varchar	Sim	Não	Não
CidadeID	Integer	Não	Não	Sim
CEP	Char	Sim	Não	Não
Email	Varchar	Sim	Não	Não
Telefone	Varchar	Sim	Não	Não
Fax	Varchar	Sim	Não	Não
Celular	Varchar	Sim	Não	Não
Observacao	Varchar	Sim	Não	Não
TipoFornecedorID	Integer	Não	Não	Sim

Situacao	Integer	Não	Não	Não

Quadro 39 - Campos da tabela Fornecedores

No Quadro 40 estão os campos da tabela de planos de contas. Um plano de conta poderá estar relacionado a uma ou muitas receitas e despesas. Os tipos serão padrões: E – Receitas e S - Despesas.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
PlanoContaID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Codigo	Varchar	Sim	Não	Não
Tipo	Char	Não	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 40 - Campos da tabela PlanosContas

No Quadro 41 estão os campos da tabela de tipos de cobranças extras que serão poderão ser adicionados ao valor de locações. Um tipo de tarifa poderá estar relacionado a uma ou muitas locações.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
TipoCobrancaExtraID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Descricao	Varchar	Sim	Não	Não
Valor	Money	Não	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 41 - Campos da tabela TiposCobrancasExtras

No Quadro 42 estão os campos da tabela de Itens para *Checkin*. Um item poderá estar vinculado a uma ou muitas locações / devoluções.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
ItemCheckinID	Serial	Não	Sim	Não
Nome	Varchar	Não	Não	Não
Situacao	Integer	Não	Não	Não

Quadro 42 - Campos da tabela ItensCheckin

No Quadro 43 estão os campos da tabela de adicionais das reservas e locações. Esta tabela foi gerada por meio de relacionamento de muitos para muitos entre locações/reservas e tipos de adicionais. Uma locação/reserva pode possuir um ou muitos adicionais.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
AdicionalID	Serial	Não	Sim	Não
TipoAdicionalID	Integer	Não	Não	Sim
LocacaoDevolucaoID	Integer	Não	Não	Sim
Situacao	Integer	Não	Não	Não

Quadro 43 - Campos da tabela Adicionais

No Quadro 44 estão os campos da tabela de cobranças extras das locações. Esta tabela foi gerada por relacionamento de muitos para muitos entre locações e tipos de cobranças extras. Uma locação pode possuir uma ou muitas cobranças extras.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
CobrançaExtraID	Serial	Não	Sim	Não
TipoCobrancaExtraID	Integer	Não	Não	Sim
Situacao	Integer	Não	Não	Não
LocacaoDevolucaoID	Integer	Não	Não	Sim

Quadro 44 - Campos da tabela CobrancaExtraLocacaoDevolucao

No Quadro 45 estão os campos da tabela de Itens para *checkin* que serão verificados nas locações e devoluções. Esta tabela foi gerada por meio do relacionamento de muitos para muitos entre locações/devoluções e itens *checkin*. Uma locação/devolução pode possuir um ou muitos itens a serem verificados.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
LocacaoDevolucaoItemCheckinID	Serial	Não	Sim	Não
ItemCheckinID	Integer	Não	Não	Sim
LocacaoDevolucaoID	Integer	Não	Não	Sim
Situacao	Integer	Não	Não	Não
Observação	Varchar	Sim	Não	Não

Quadro 45 - Campos da tabela ItensCheckinLocacaoDevolucao

No Quadro 46 estão os campos da tabela de Locações / Devoluções.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
LocacaoDevolucaoID	Serial	Não	Sim	Não
ClienteID	Integer	Não	Não	Sim
VeiculoID	Integer	Não	Não	Sim
DataLocacao	TimeStamp	Não	Não	Não
HoraLocacao	TimeStamp	Não	Não	Não
DataPrevistaDevolucao	TimeStamp	Não	Não	Não
HoraPrevistaDevolucao	TimeStamp	Não	Não	Não
DataDevolucao	TimeStamp	Sim	Não	Não
HoraDevolucao	TimeStamp	Sim	Não	Não
FuncionarioID	Integer	Não	Não	Sim
ValorTotal	Money	Não	Não	Não
QtdeDias	Integer	Não	Não	Não
KmDevolucaoo	Decimal	Sim	Não	Não
SeguroID	Integer	Não	Não	Sim
Situacao	Integer	Não	Não	Não

Quadro 46 - Campos da tabela LocacoesDevolucoes

No Quadro 47 estão os campos da tabela contas a pagar. A situação das parcelas será padrão: 0 - pendente, 1 – quitada e 2 – cancelada.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
ContaPagarID	Serial	Não	Sim	Não
Valor	Money	Não	Não	Não
DataVencimento	Date	Não	Não	Não
Desconto	Money	Sim	Não	Não
Juros	Money	Sim	Não	Não
DataPagamento	Date	Sim	Não	Não
ValorPago	Money	Sim	Não	Não
Situacao	Integer	Não	Não	Não
FornecedorID	Integer	Não	Não	Sim
FuncionarioID	Integer	Não	Não	Sim
PlanoContaID	Integer	Não	Não	Sim
TipoCobrancaRecebimentoID	Integer	Não	Não	Sim
Observacao	Varchar	Sim	Não	Não

Quadro 47 - Campos da tabela ContasPagar

No Quadro 48 estão os campos da tabela contas a receber. A situação das parcelas será padrão: 0 - pendente, 1 – quitada e 2 – cancelada.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
ContaReceberID	Serial	Não	Sim	Não
Valor	Money	Não	Não	Não
DataVencimento	Date	Não	Não	Não
Desconto	Money	Sim	Não	Não
Juros	Money	Sim	Não	Não
DataPagamento	Date	Sim	Não	Não
ValorPago	Money	Sim	Não	Não
Situacao	Integer	Não	Não	Não
ClienteID	Integer	Não	Não	Sim
PlanoContaID	Integer	Não	Não	Sim
TipoCobrancaRecebimento ID	Integer	Não	Não	Sim
LocacaoDevolucaoID	Integer	Sim	Não	Sim
Observacao	Varchar	Sim	Não	Não

Quadro 48 - Campos da tabela ContasReceber

No Quadro 49 estão os campos da tabela de log. Nesta tabela será armazenada as principais alterações que os usuários realizarem no sistema. O tipo da alteração é padrão: 1 – inclusão, 2 – edição e 3 – exclusão.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
LogID	Serial	Não	Sim	Não
Tabela	Varchar	Não	Não	Não
Chave	Integer	Não	Não	Não
Data Hora	TimeStamp	Não	Não	Não
FuncionarioID	Integer	Não	Não	Não
TipoAlteracao	Integer	Sim	Não	Não

Quadro 49 - Campos da tabela Log

No Quadro 50 estão os campos da tabela de permissões dos funcionários. Esta tabela foi gerada por meio de relacionamento de muitos para muitos entre funcionários e permissões. Um funcionário pode possuir uma ou muitas permissões.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
FuncionarioPermissaoID	Serial	Não	Sim	Não
PermissaoID	Integer	Não	Não	Sim
FuncionarioID	Integer	Não	Não	Sim

Quadro 50 - Campos da tabela FuncionarioPermissao

No Quadro 51 estão os campos da tabela de permissões.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira
PermissaoID	Integer	Não	Sim	Não
Permissao	Varchar	Não	Não	Não

Quadro 51 - Campos da tabela Permissao

4.3 APRESENTAÇÃO DO SISTEMA

A Subseção 4.3.1 apresenta o sistema *web* desenvolvido e na 4.3.2 é apresentado o aplicativo *mobile* implementado.

4.3.1 Interface Web

O leiaute do sistema apresentado foi desenvolvido com o *Bootstrap,* que é um *framework front-end* que auxilia na criação de interface de sites e sistemas *web*. A interface é composta por menus laterais e páginas de cadastros.

A Figura 5 apresenta a tela inicial do sistema para os usuários, após realizada a autenticação no sistema. Nela é possível realizar a navegação entre os menus e são apresentados dados por meio de indicativos e gráficos exibindo informações em tempo real. Os dados exibidos são consultados dinamicamente do banco de dados e são atualizados quando são realizadas operações que os alterem.

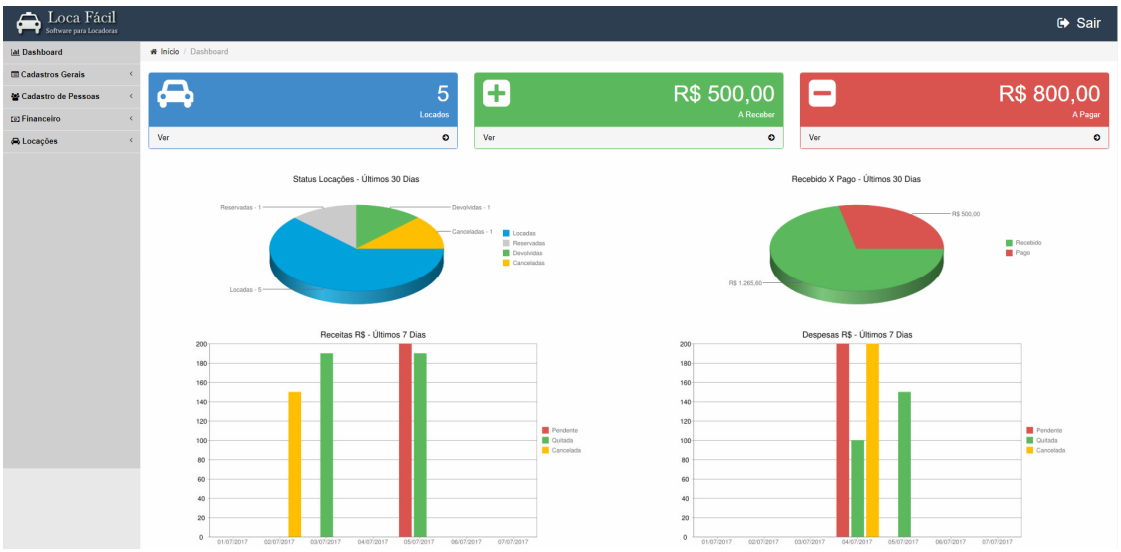

Figura 5 - Tela de Início

Ao clicar em qualquer um dos menus de cadastros, é possível verificar as listas de itens cadastrados e algumas informações mais significativas, como mostrado na Figura 6, possibilitando realizar filtros, além das telas de listas serem responsivas.

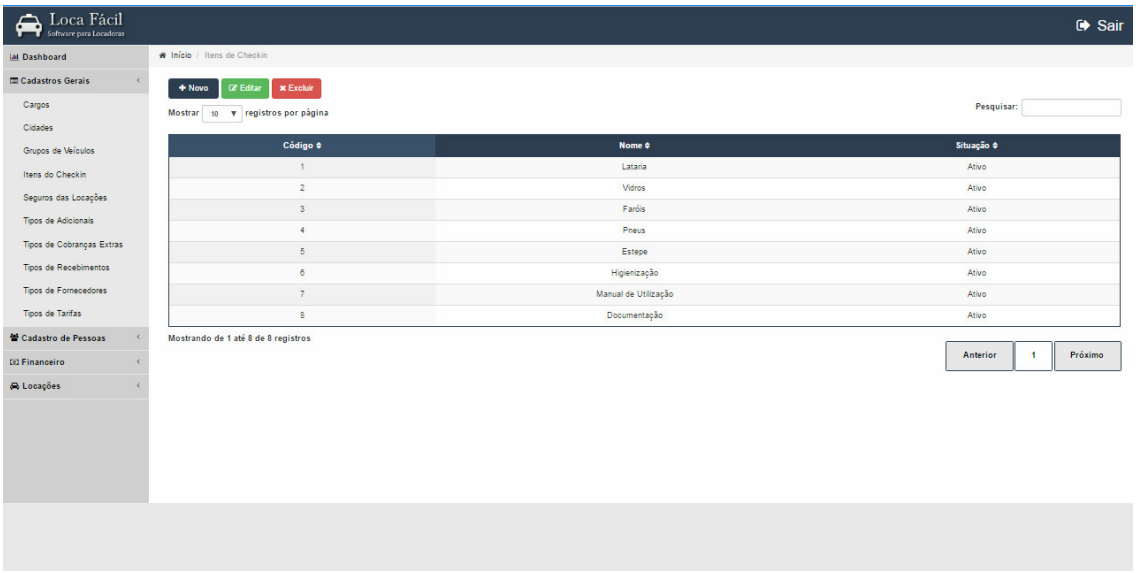

Figura 6 **- Tela de Listagem**

Nas telas de listagem dos cadastros gerais há funções para incluir, editar e excluir, com as validações necessárias, como de bloquear exclusões de itens com vínculos e confirmação da opção de excluir, como é possível verificar na Figura 7.

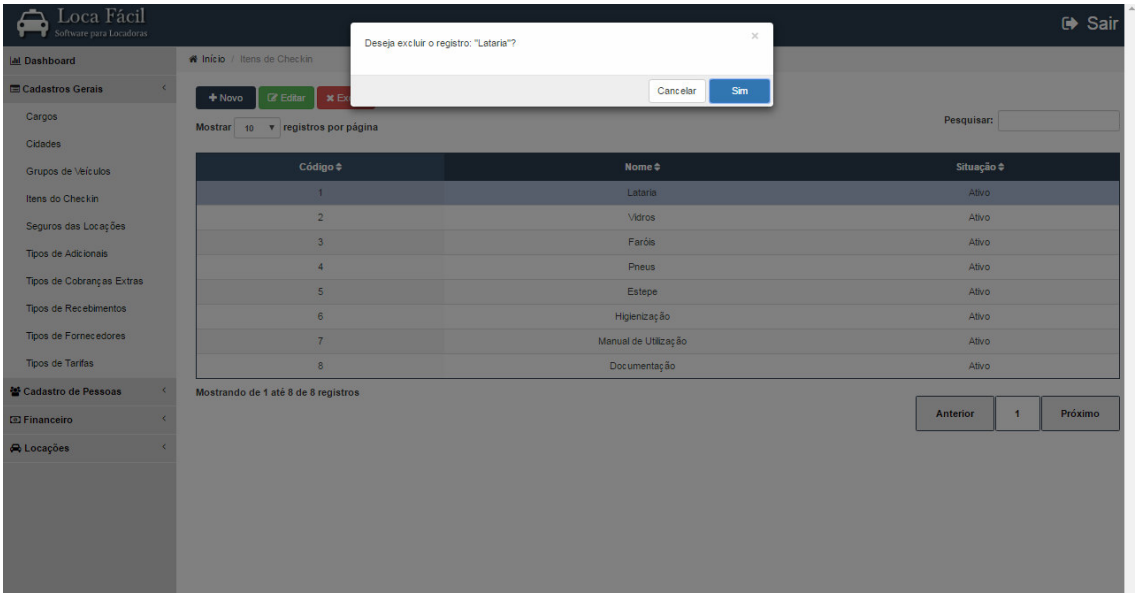

Figura 7 - Tela de Exclusão de Cadastro

Nas telas de formulários, são realizadas algumas validações e verificações de campos obrigatórios, não sendo possível realizar a inclusão ou alteração do cadastro sem que essas verificações e validações sejam realizadas, como mostra a Figura 8.

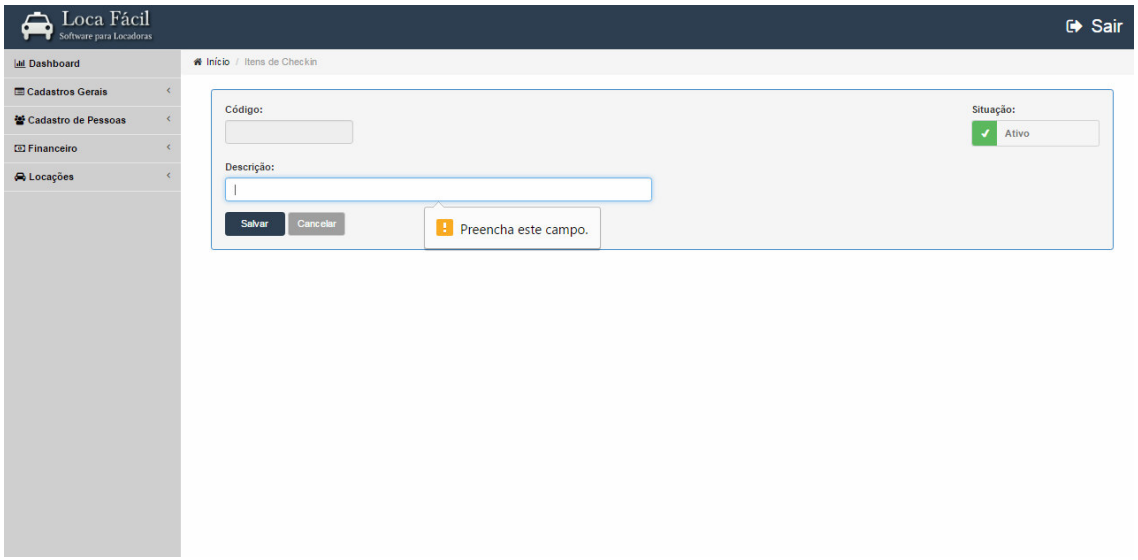

Figura 8 - Tela de Inclusão de Novo Cadastro

Os cadastros possuem campos de situação, para que possam ser inativados os itens que não podem ser excluídos, como mostra a Figura 9. Esses itens podem estar vinculados a outras tabelas, como, por exemplo, itens de *check in* que podem não ser mais disponíveis por se tornarem obsoletos para os veículos atuais, mas é necessário mantê-los pela consistência com dados dos veículos que foram locados quando esses itens estavam ativos.

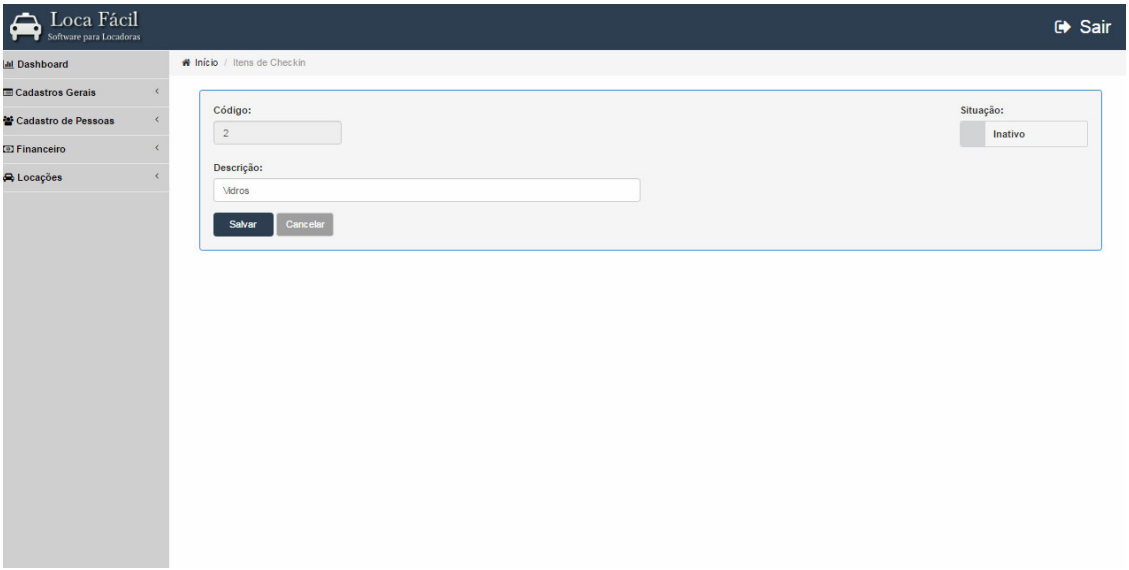

Figura 9 - Tela de Inativação de Cadastro

Nas telas de cadastros de pessoas existem algumas validações diferentes, como o preenchimento correto de cada campo e busca automática do endereço por meio do CEP, como é possível ver na Figura 10.

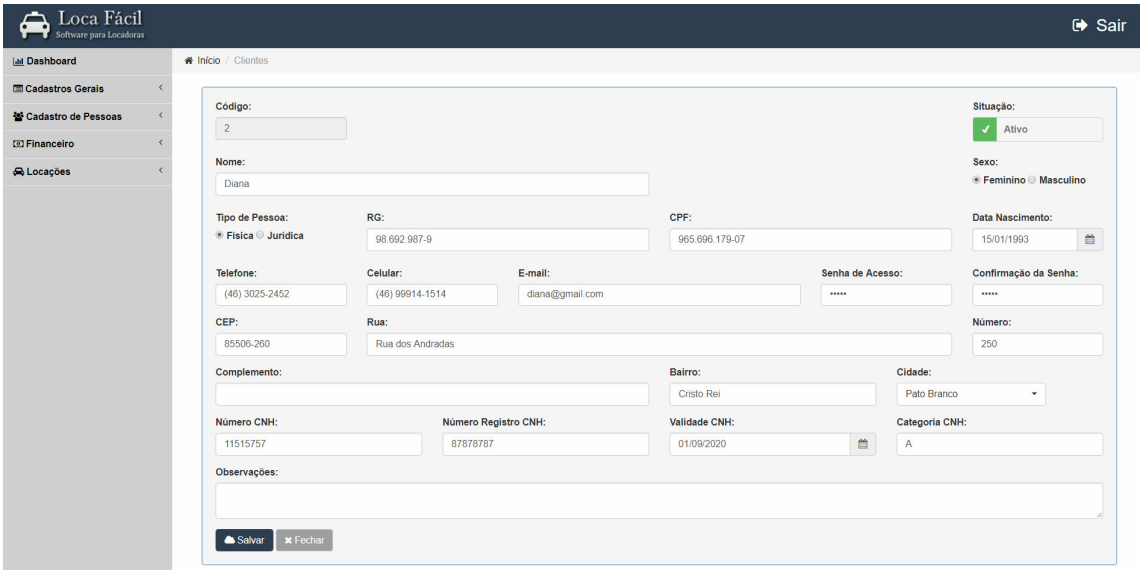

Figura 10 - Tela de Cadastro de Pessoas

Nas telas do menu Financeiro é possível ver a lista de parcelas em diferentes situações, e ao posicionar o mouse sobre as mesmas é exibido a descrição da situação, também há as opções de incluir, excluir, quitar e cancelar uma parcela, como mostra a Figura 11, além da possibilidade de gerar o recibo da parcela selecionada.

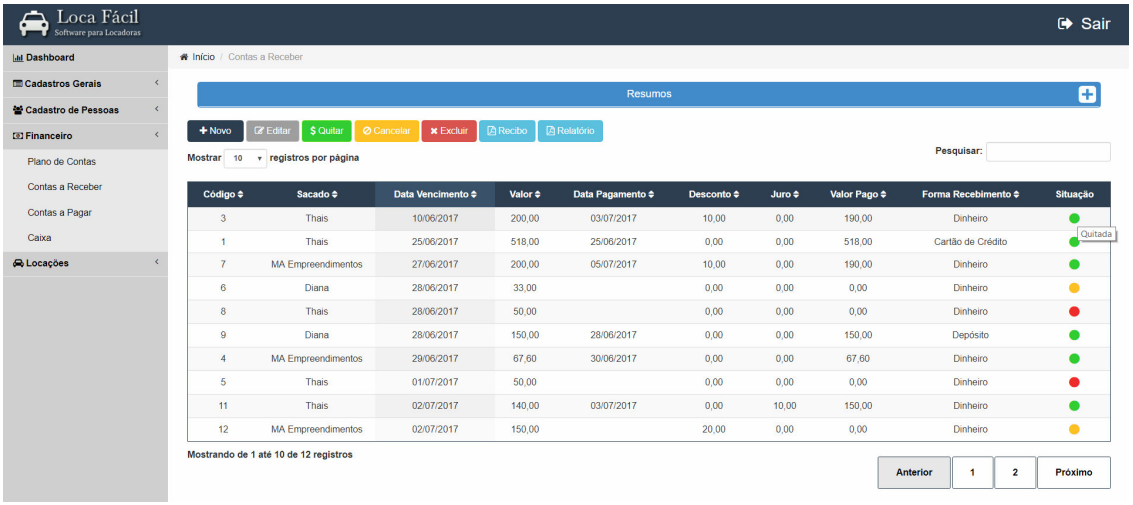

Figura 11 - Tela de Listagem do Financeiro

Nas telas de inclusão e edição das parcelas, foram realizadas as validações necessárias dos campos de preenchimento e bloqueio de alterações dos campos reservados para dados após a quitação, como é mostrado na Figura 12.

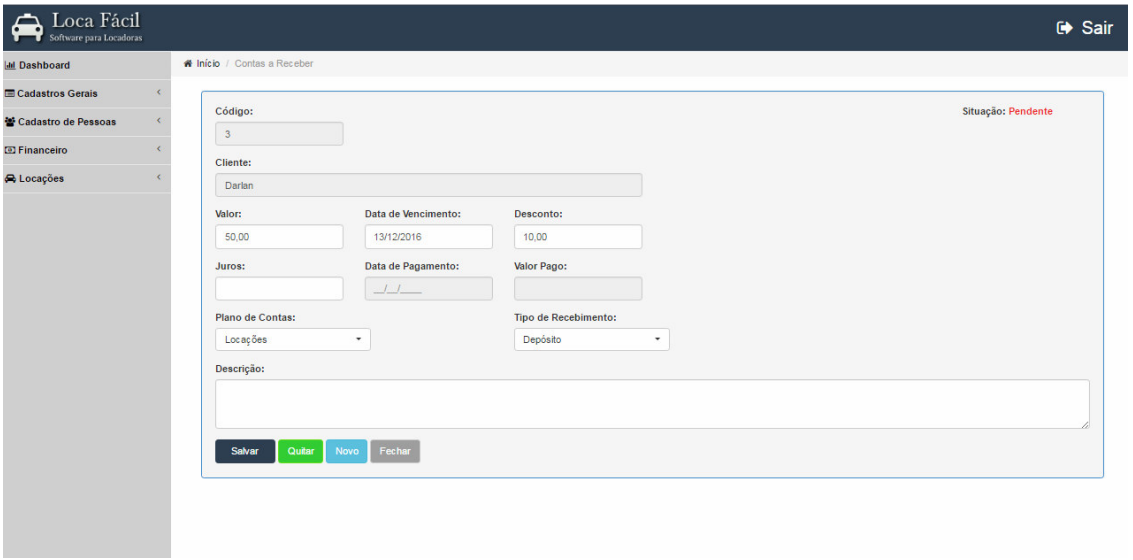

Figura 12 - Tela de Edição de Parcela

Na tela de quitação os campos são preenchidos automaticamente com as informações da parcela e há a possibilidade de alterações se necessário, como é possível verificar na Figura 13. Após a quitação, a situação da parcela é alterada e o valor adicionado ao caixa.

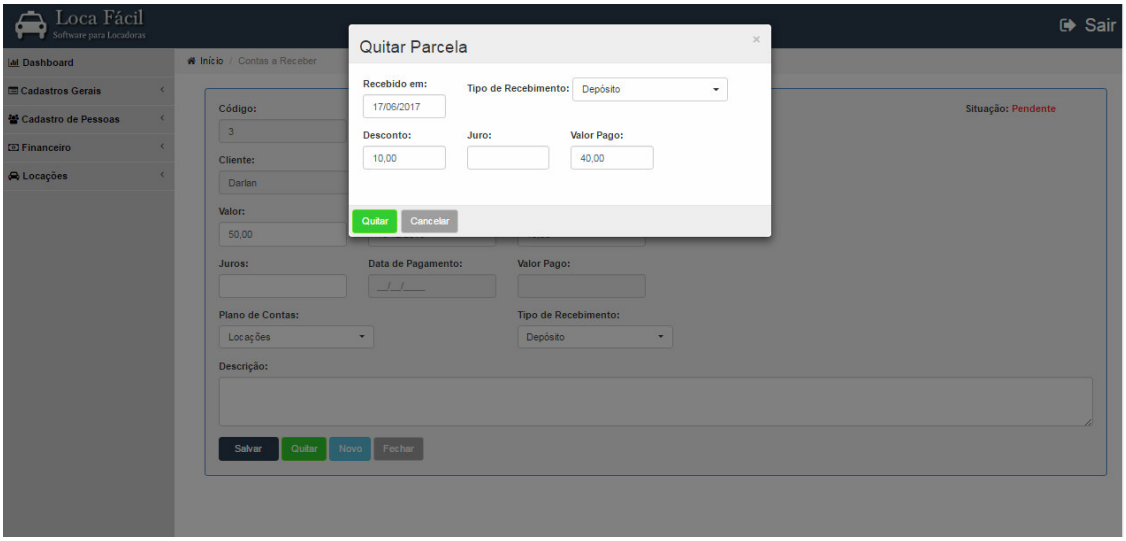

Figura 13 - Tela de Quitação de Parcela

Na tela do caixa, mostrada na Figura 14, são exibidos os resumos das entradas e saídas que são os valores totais das receitas e despesas quitadas, além da listagem das parcelas. O tipo 'S' corresponde à despesa e o tipo 'E' corresponde à receita.

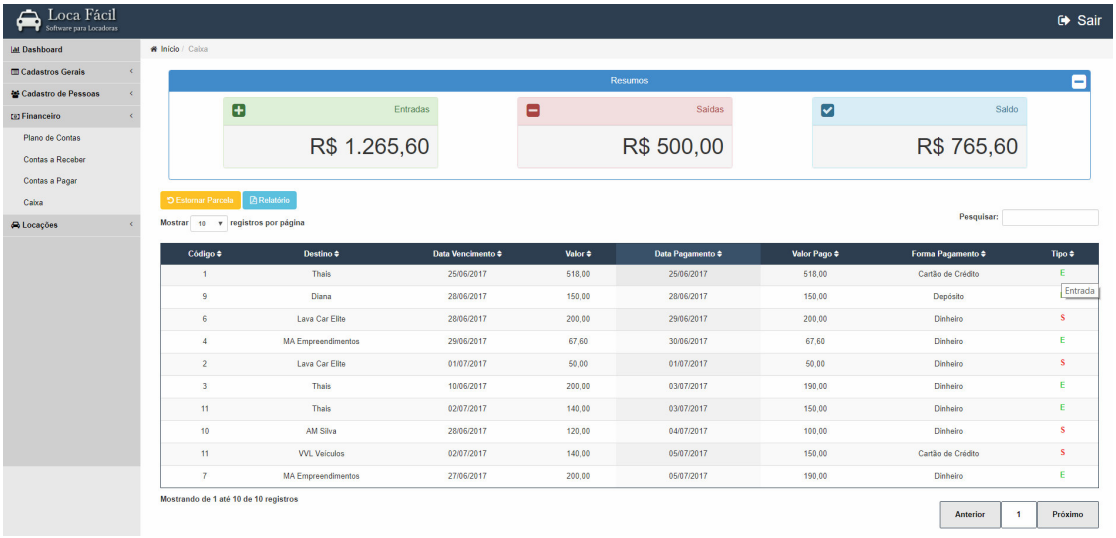

Figura 14 - Tela do Caixa

No menu de Locações, há a opção de cadastro de veículos. Na Figura 15, é possível verificar a tela de listagem dos veículos cadastrados, com as opções de incluir um novo veículo, editar ou excluir o veículo selecionado. Um veículo em situação indisponível, não pode ser locado. Isso ocorre porque o veículo já está locado, em manutenção ou outras situações que não permitam que, no momento, possa ser disponibilizado para locação.

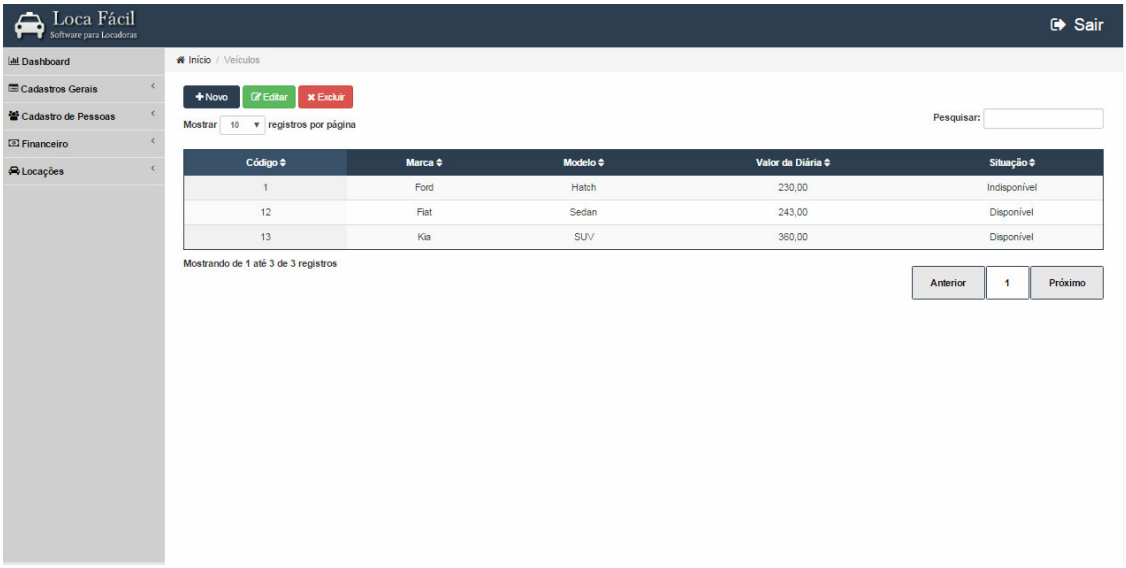

Figura 15 - Tela de Listagem de Veículos

Na tela de inclusão e edição dos veículos são exibidos os campos de informações de fábrica e informações atuais do veículo, além de apresentar se o mesmo está disponível ou não para locações, como é possível verificar na Figura 16.

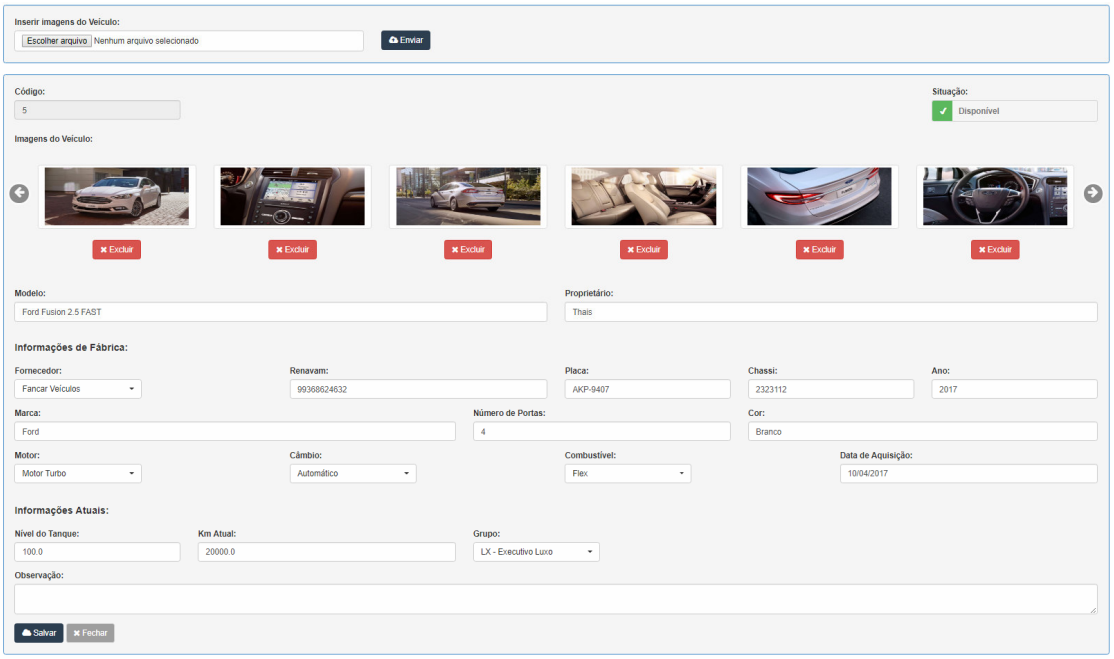

Figura 16 - Tela de Edição de Veículos

A Figura 17 apresenta a listagem das locações e devoluções. Nessa tela são exibidas informações cadastrais. Além das opções de incluir uma nova locação, editar ou excluir a locação selecionada. Além disso, é possível imprimir um relatório com as locações.

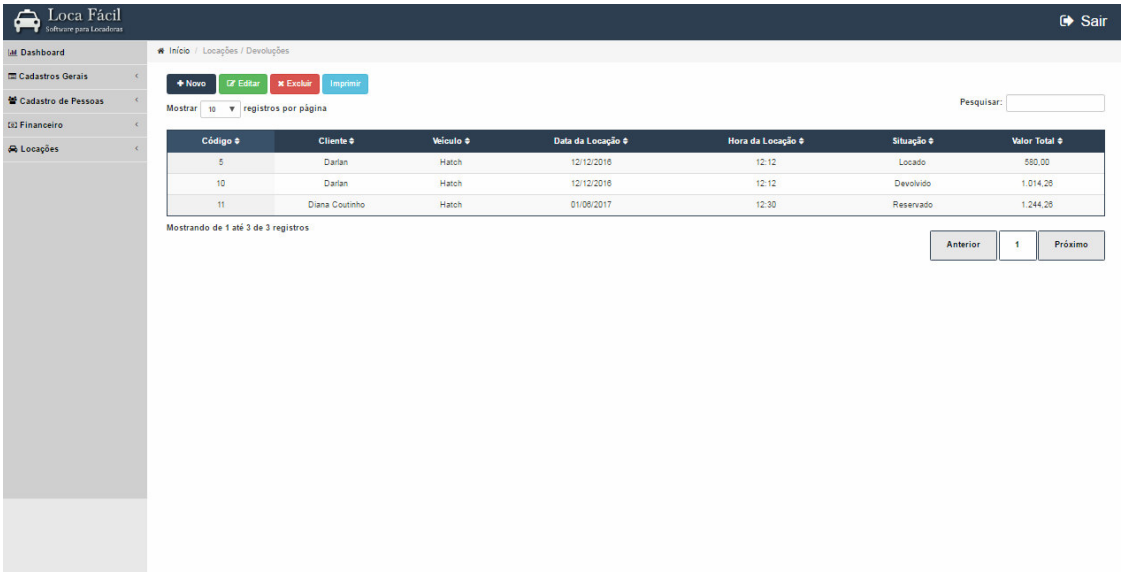

Figura 17 - Tela de Listagem das Locações e Devoluções

Na tela de inclusão e edição da locação, é possível informar os dados da locação ou reserva e posteriormente registrar os dados da devolução do veículo. Nesta tela também é possível criar uma conta a receber com os dados da locação selecionada, como é mostrado na Figura 18.

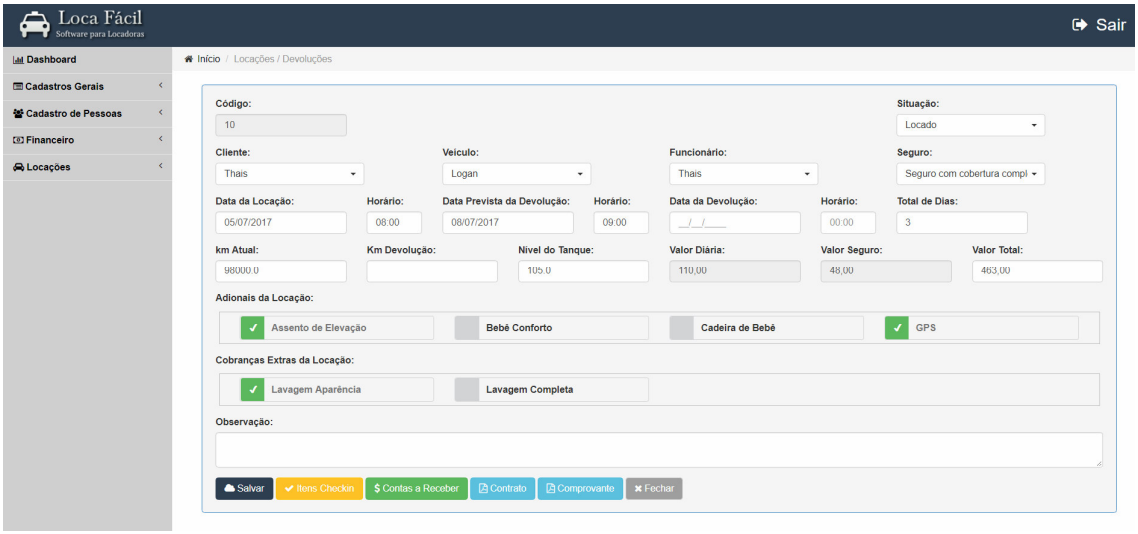

Figura 18 - Tela de Edição da Locação / Devolução

Na tela de edição da locação há um botão de Itens *Checkin* que direciona para uma tela de listagem dos itens pré-cadastrados para a verificação dos itens de checagem do veículo locado, como é possível ver na Figura 19. A lista pode ser utilizada para colocar observações no momento da locação ou para verificação na devolução do veículo.

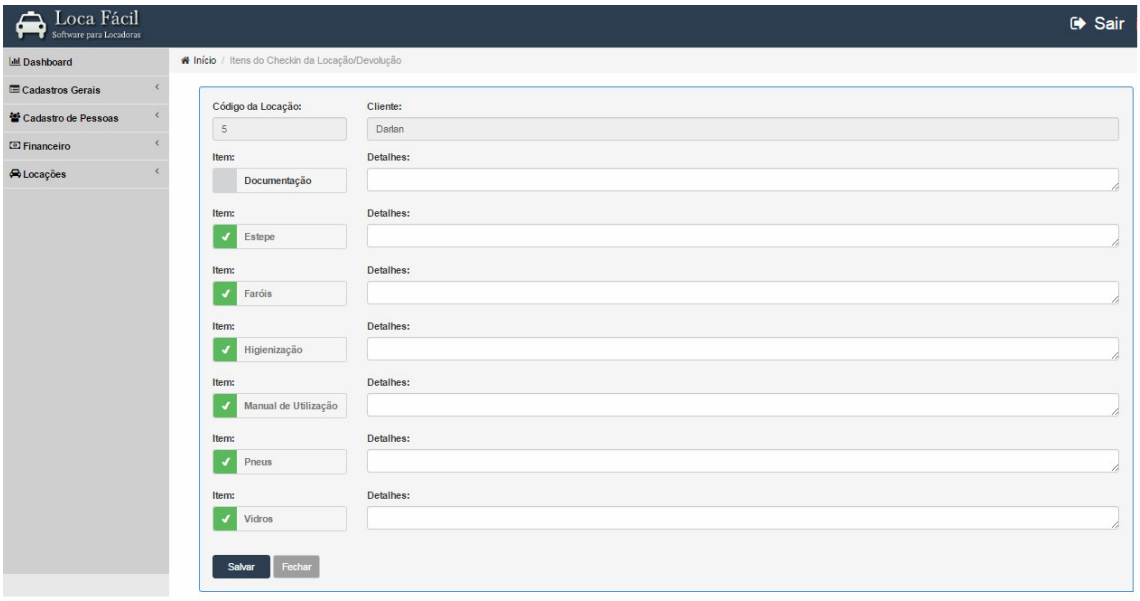

Figura 19 - Tela de Edição da Locação / Devolução

4.3.2 Interface Mobile

O aplicativo *mobile* é destinado aos clientes das locadoras, disponibilizando telas para cadastro de novos clientes, *login*, alteração de dados cadastrais básicos, consultar locações e realizar reservas.

A Figura 20 apresenta a tela de *login* do aplicativo, nela o cliente efetua autenticação, utilizando seu *e-mail* e senha cadastrados no sistema *web,* para ter acesso as funcionalidades do aplicativo. Caso o usuário não possua cadastro, poderá efetuar um novo clicando na opção ‘Cadastre-se’.

.

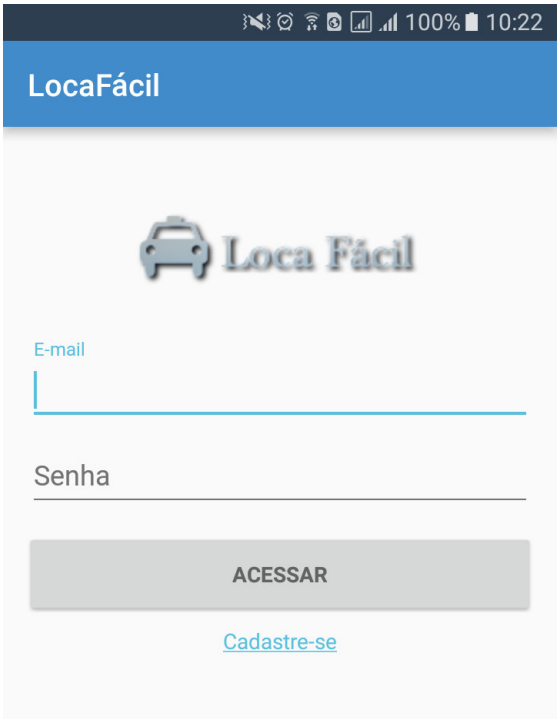

Figura 20 - Tela de *Login Mobile*

A Figura 21 apresenta a tela de cadastro de novo cliente que é acessada pela opção ‘Cadastre-se’ disponível na tela de *login*. Após informar os dados obrigatórios que são: nome, CPF ou CNPJ, *e-mail* e senha, ao clicar em salvar o cadastro é criado e o *login* no aplicativo é efetuado automaticamente.

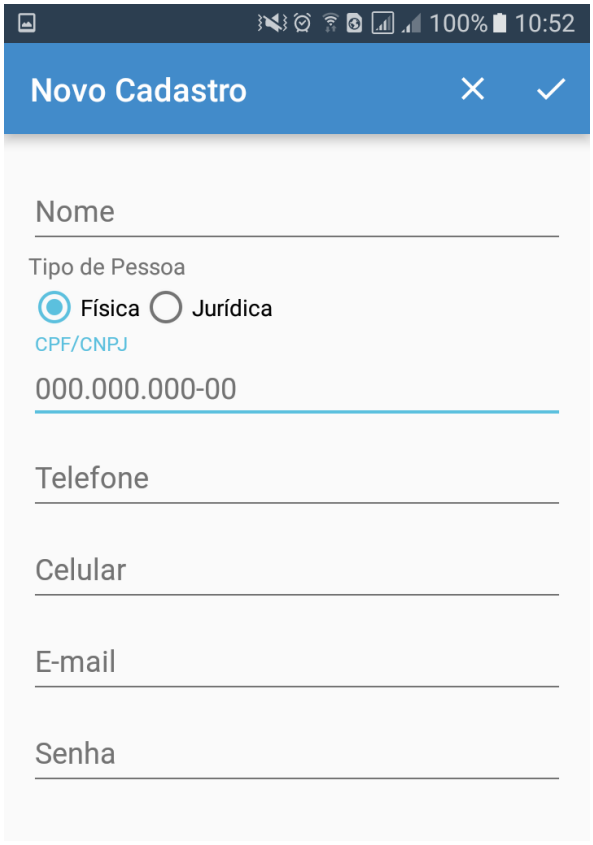

Figura 21 - Tela de Cadastro do Cliente Mobile

A Figura 22 apresenta a tela inicial do aplicativo, que disponibiliza atalhos para acessar suas funcionalidades. Por meio desta tela, o cliente pode visualizar a tela de locações efetuadas, realizar novas reservas e a edição dos seus dados cadastrais básicos.

Figura 22 - Tela Inicial do Aplicativo

A Figura 23 apresenta o menu do aplicativo, nele são exibidas as seguintes opções: *Home* que exibe a tela inicial; Minhas Locações que mostra as locações efetuadas pelo cliente; Meus Dados para realizar a edição dos dados cadastrais básicos; e Sair que efetua o *logoff* do aplicativo retornando para a tela de *login*.

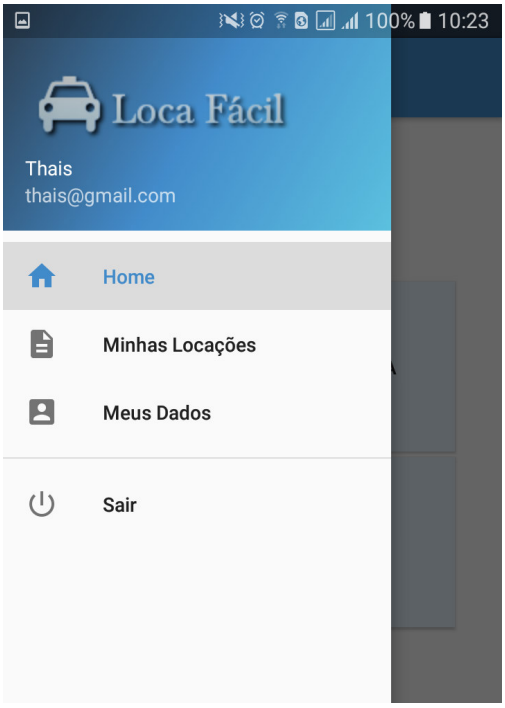

Figura 23 - Tela de Navegação do Aplicativo

A Figura 24 apresenta, em forma de lista, as locações efetuadas pelo cliente autenticado. No título da lista é exibida a data e a situação da locação e ao clicar em um item ela é expandida exibindo mais detalhes da locação, como por exemplo, veículo, seguro e valor da locação. Por meio desta tela também é possível efetuar novas reservas.

Figura 24 - Tela de Consulta de Locações e Devoluções

A Figura 25 apresenta a tela de cadastro de novas reservas, na qual o cliente seleciona o veículo, a opção de seguro, a data, o horário e a quantidade de dias que deseja locar o veículo. Com base nesses dados, é calculado o valor previsto da locação e são exibidos os detalhes da reserva.

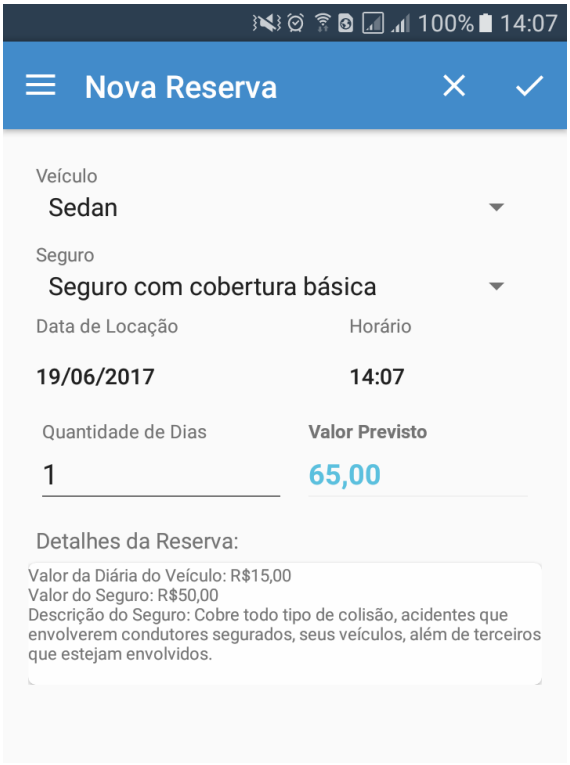

Figura 25 - Tela de Reservar pelo Aplicativo

A Figura 26 apresenta a tela de edição dos dados cadastrais básicos do cliente, não sendo possível editar o nome e CPF/CNPJ, por motivos de segurança.

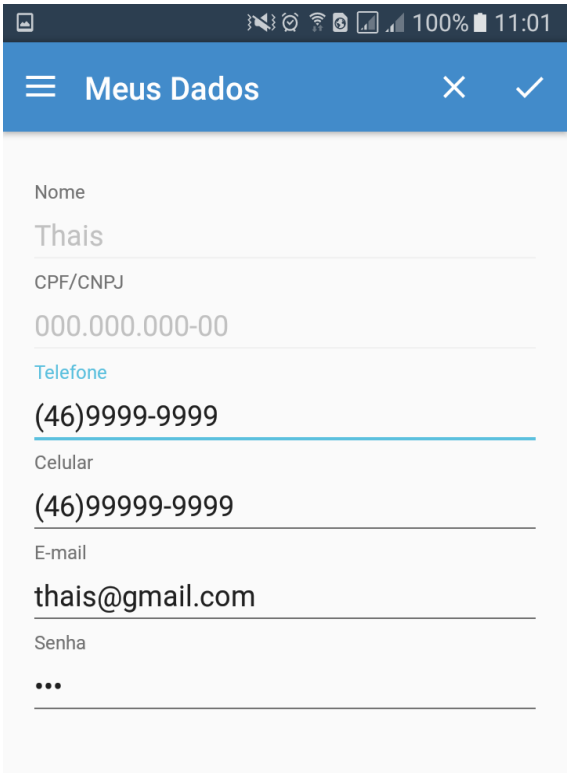

Figura 26 - Tela de Edição do Cadastro de Cliente pelo Aplicativo

4.4 IMPLEMENTAÇÃO DO SISTEMA

A Subseção 4.4.1 apresenta a implementação do sistema *web* e na Subseção 4.4.2 está a implementação da aplicação *mobile*.

4.4.1 Sistema *Web*

O sistema *web* possui os menus de todas as telas de acesso ao sistema, com as liberações e as restrições específicas de cada usuário.

Na Listagem 1 é exibido o código que define as regras de acesso dos usuários às páginas do sistema. Por meio da *tag* antMatchers é definida as páginas permitidas a cada usuário de acordo com sua *role*, ou seja, seu tipo de permissão. Nesta configuração também são definidas as páginas que devem ser exibidas caso o usuário não possua permissão e as páginas caso ocorra sucesso ou falha no *login*.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
      http.csrf().disable()
            .exceptionHandling().accessDeniedPage("/error403").and()
            .formLogin().loginPage("/login").permitAll()
            .defaultSuccessUrl("/privado/index")
            .failureUrl("/login?error=true").permitAll()
            .and().authorizeRequests()
            .antMatchers("/privado/AplicacaoRest/**").permitAll()
            .antMatchers("/privado").hasRole("ADMIN")
            .antMatchers("/privado/index/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
            .antMatchers("/privado/cidade/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
            .antMatchers("/privado/cliente/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
      .antMatchers("/privado/contareceber/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
            .antMatchers("/privado/grupo/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
      .antMatchers("/privado/itemcheckin/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
      .antMatchers("/privado/locacaodevolucao/**").hasAnyRole("FUNCIONARIO
", "ADMIN")
      .antMatchers("/privado/tipoadicional/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
      .antMatchers("/privado/tipocobrancaextra/**").hasAnyRole("FUNCIONARI
O", "ADMIN")
      .antMatchers("/privado/tipocobrancarecebimento/**").hasAnyRole("FUNC
IONARIO", "ADMIN")
      .antMatchers("/privado/seguro/**").hasAnyRole("FUNCIONARIO",
"ADMIN")
      .antMatchers("/privado/**").hasRole("ADMIN");
}
```

Listagem 1 - Tela de WebSecurityCong

O sistema possui vários métodos padrão para as funcionalidades de cadastros, como o método salvar apresentado na Listagem 2. Neste método são passadas por parâmetro informações se os dados são válidos, se há erro e o *model* contendo os dados informados pelo cliente. Caso o *model* recebido contenha algum erro, como campos obrigatórios que não foram informados, é retornada uma mensagem que informa ao usuário que ocorreu uma falha. Caso contrário é verificada a situação em que o registro deve estar, pois se o usuário deixá-lo inativo irá para o *model* como *null* e, dessa forma, deve ser alterada a situação para 2. Antes de salvar, se for uma inclusão é verificado se existe um registro cadastrado com o mesmo nome, caso exista, será exibida uma mensagem alertando ao usuário, senão o registro é salvo utilizando a persistência do *Spring Data JPA*.

```
@RequestMapping(value = "/", method = RequestMethod.POST, produces =
"application/json")
	@ResponseBody
	public String salvar(@Valid Cargo cargo, BindingResult erros, Model
model) {
		JSONObject retorno = new JSONObject();
		try {
			if (erros.hasErrors()) {
				retorno.put("situacao", "ERRO");
				retorno.put("title", "Atenção!");
				retorno.put("type", "error");
				retorno.put("mensagem", "Falha ao salvar
registro!");
			} else {
				Integer tipoAlteracao = 1;

				if (cargo.getSituacao() == null) {
					cargo.setSituacao(2);
				}
				List<Cargo> c =
cargoRepository.findByNomeIgnoreCase(cargo.getNome());
				if (cargo.getId() != null) {
					if (c.size() > 0) {
						if (c.get(0).getId() != cargo.getId())
{
							return
getMensagemNomeIgual().toString();
						}
					}
					tipoAlteracao = 2;
				} else {
					if (c.size() > 0) {
						return
getMensagemNomeIgual().toString();
					}
				}
				cargoRepository.save(cargo);
				salvarLog(cargo.getId(), tipoAlteracao,
criaObsLog(cargo));
				retorno.put("id", cargo.getId());
				retorno.put("situacao", "OK");
				retorno.put("title", "Salvo!");
				retorno.put("type", "success");
				retorno.put("mensagem", "Registro salvo com
sucesso!");
			}
		} catch (Exception ex) {
			retorno.put("situacao", "ERRO");
			retorno.put("title", "Atenção!");
			retorno.put("type", "error");
			retorno.put("mensagem", "Falha ao salvar registro!");
		}
		return retorno.toString();
}
```

Listagem 2 - Salvar Controller

Na Listagem 3 é exibido o método buscar, passando por parâmetro o *id* do registro que deseja realizar a edição, é realizada a pesquisa no banco de dado e o resultado é adicionado ao

model para ser retornado ao usuário. No método *popularView* são adicionados os demais *models* necessários para o preenchimento dos campos redirecionando para o formulário de edição.

```
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public String buscar(@PathVariable Long id, Model model) {
	Cliente cliente = clienteRepository.findOne(id);
	model.addAttribute("cliente", cliente);
	Long idCidade;
	if (cliente.getCidade() != null) {
		idCidade = cliente.getCidade().getId();
	} else {
		idCidade = -1L;
	}
	popularView(model, idCidade);
	return URL_FORM;
}
```

Listagem 3 - Buscar Controller

Na Listagem 4 é exibido o método excluir, passando por parâmetro o *id* do registro que deseja realizar a exclusão. Se o registro possuir vínculos com outras tabelas do sistema o mesmo não será excluído e retornará o resultado do processo ao usuário.

```
@RequestMapping(value = "/remover/{id}", produces = "application/json")
@ResponseBody
public String excluir(@PathVariable Long id) {
	JSONObject retorno = new JSONObject();
	try {
		clienteRepository.delete(id);
		retorno.put("situacao", "OK");
		retorno.put("title", "OK!");
		retorno.put("type", "success");
		retorno.put("mensagem", "Registro removido com sucesso!");
	} catch (Exception ex) {
		if(ex.getMessage().contains("ConstraintViolationException")){
			retorno.put("mensagem", "Registro possui vínculos com o
sistema e não pode ser excluído!");
		}else{
			retorno.put("mensagem", "Falha ao excluir registro!");
		}
		retorno.put("situacao", "ERRO");
		retorno.put("title", "Falha!");
		retorno.put("type", "error");
	}
	return retorno.toString();
}
```

Listagem 4 - Excluir Controller

Na Listagem 5 é apresentado o método listar, nele é gerado um JSON com todos os registros cadastrados para serem exibidos ao usuário, percorrendo a lista e formatando os dados desejados utilizando a estrutura de repetição *forEach*.

```
@RequestMapping(value = "/dados", produces = "application/json")
@ResponseBody
public String listarDados() {
	List<Cliente> lista = clienteRepository.findAll();
	JSONObject retorno = new JSONObject();
	JSONArray array = new JSONArray();
	JSONArray ja;
	for (Cliente cliente : lista) {
		ja = new JSONArray();
		ja.put(cliente.getId());
		ja.put(cliente.getNome());
		if(cliente.getTipoPessoa().equalsIgnoreCase("J")){
			ja.put(cliente.getCnpj());
		}else{
			ja.put(cliente.getCpf());
		}
		ja.put(cliente.getEmail());
		ja.put(cliente.getTelefone());
		ja.put(cliente.getCelular());
		if (cliente.getSituacao() == 1) {
			ja.put("Ativo");
		} else {
			ja.put("Inativo");
		}
		array.put(ja);
	}
	retorno.put("data", array);
	return retorno.toString();
}
```

Listagem 5 - Listar Controller

Na Listagem 6 é apresentado o método quitar, que recebe as informações de pagamento das parcelas via *post*, atualiza as informações de pagamento e atualiza a situação da mesma para 'quitada', retornando ao usuário o resultado do processo.

```
@RequestMapping(value = "/quitar", method=RequestMethod.POST, produces =
"application/json")
@ResponseBody
public String quitar(@Valid ContaReceber contaReceber, BindingResult
erros, Model model) {
	JSONObject retorno = new JSONObject();
	try {
		if (erros.hasErrors()) {
			retorno.put("situacao", "ERRO");
			retorno.put("title", "Falha!");
			retorno.put("type", "error");
			retorno.put("mensagem", "Falha ao quitar o registro!");
		} else {
		ContaReceber r =
contaReceberRepository.findOne(contaReceber.getId());
			r.setSituacao(1);
			r.setDataPagamento(contaReceber.getDataPagamento());
			r.setValorPago(contaReceber.getValorPago());
			r.setDesconto(contaReceber.getDesconto());
			r.setJuro(contaReceber.getJuro());
			if (contaReceber.getTipoCobrancaRecebimento() != null) {

r.setTipoCobrancaRecebimento(contaReceber.getTipoCobrancaRecebimento());
```

```
                  }
                  if(contaReceber.getDataPagamento() == null ||
contaReceber.getValorPago() == null){
                        retorno.put("situacao", "ERRO");
                        retorno.put("title", "Falha!");
                        retorno.put("type", "error");
                        retorno.put("mensagem", "Falha ao quitar o
registro!");
                  }else{
                        contaReceberRepository.save(r);
                        retorno.put("id", contaReceber.getId());
                        retorno.put("situacao", "OK");
                        retorno.put("title", "Salvo!");
                        retorno.put("type", "success");
                        retorno.put("mensagem", "Registro salvo com
sucesso!");
                  }
            }
      } catch (Exception ex) {
            retorno.put("situacao", "ERRO");
            retorno.put("title", "Falha!");
            retorno.put("type", "error");
            retorno.put("mensagem", "Falha ao quitar o registro!");
      }
      return retorno.toString();
}
```

Listagem 6 - Quitar Controller

Na Listagem 7 é apresentado parte do arquivo pom.xml que é responsável pelo gerenciamento das configurações e importação das dependências necessárias utilizando o Maven. Dentro da *tag* "<dependencie>" são informadas os *groups* e *artifactys* das dependências, estes dados podem ser encontrados no site https://mvnrepository.com/.

```
project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <groupId>br.edu.utfpr.pb.locadora</groupId>
      <artifactId>locadora</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <packaging>war</packaging>
      <name>locadora</name>
      <description>locadora</description>
      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>1.4.0.RELEASE</version>
      </parent>
      <properties>
            <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
            <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
            <java.version>1.8</java.version>
      </properties>
      <dependencies>
            <dependency>
```

```
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
```

Listagem 7 - Pom.xml

Na Listagem 8 é apresentado o arquivo application.properties que é responsável por armazenar as informações de configuração da aplicação, como, por exemplo, o prefixo e o sufixo dos arquivos para que não seja necessário informar em cada página, além dos dados de conexão com o banco de dados.

```
spring.mvc.view.prefix: /WEB-INF/view/
spring.mvc.view.suffix: .jsp
spring.jpa.hibernate.ddl-auto: update
spring.jpa.show-sql: true
spring.jpa.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.datasource.url=jdbc:postgresql://localhost:5432/locadora
spring.datasource.username=*******
spring.datasource.password=*******
spring.datasource.driverClassName=org.postgresql.Driver
server.port=8085
```

Listagem 8 - Aplication.properties

Na Listagem 9 é apresentado parte do arquivo que contêm a página base do sistema, que é composta pelo cabeçalho e menus. A *tag* <jsp:invoke> é responsável por invocar os arquivos CSS específicos da página que importará o *layout*. Na *tag* <jsp:doBody /> será exibido o conteúdo das páginas que implementam o *layout*. A *tag* <sec:authorize> é responsável por ocultar os menus dos usuários conforme a *role* definida em seu cadastro.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<jsp:invoke fragment="cssEspecificos"></jsp:invoke>
</head>
<body>
  <div id="wrapper">

<nav class="navbar navbar-default navbar-static-top" role="navigation"
      style="margin-bottom: 0">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse"
        data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span> <span
        class="icon-bar"></span> <span class="icon-bar"></span> <span
        class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="<c:url value="/privado/index" />"
        style="padding: 0px 15px;"> <img src="/static/img/logo.png">
      </a>
    </div>
    <ul class="nav navbar-top-links navbar-right">
      <!--Logout-->
      <li class="navbar-brand">
      <li><a href="/logout" style="font-size: 25px;"><i
        class="fa fa-sign-out fa-fw"></i> Sair</a></li>
```

```
    </ul>
    <div class="navbar-default sidebar" role="navigation">
      <div class="sidebar-nav navbar-collapse">
        <ul class="nav" id="side-menu">
        <sec:authorize access="hasRole('ADMIN')">
          <li><a href="../../../privado/central/dashboard"><i
            class="fa fa-bar-chart CorMenuItem"></i> Dashboard</a></li>
        </sec:authorize>
        <li><a href="#"><i class="fa fa-list-alt"></i> Cadastros
          Gerais<span class="fa arrow"></span></a>
          <ul class="nav nav-second-level">
            <sec:authorize access="hasRole('ADMIN')">
            <li><a href="../../../privado/cargo">Cargos</a></li>
          </sec:authorize>
          <li><a href="../../../privado/cidade">Cidades</a></li>
          <li><a href="../../../privado/grupo">Grupos de Veículos</a>
          </li>
...
     <div id="page-wrapper">
         <jsp:doBody />
    </div>
   </div>
 </body>
</html>
```

Listagem 9 – Template

Na Listagem 10 é apresentado parte de um arquivo JSP que implementa o *layout*, definido pela *tag* <layout:template>, na *tag* <jsp:attribute name="cssEspecificos"> são importados os *css* específicos desta página e na *tag* <jsp:attribute name="scriptsEspecificos"> são importados os arquivos de *scrips* da mesma.

```
<layout:template>
      <jsp:attribute name="cssEspecificos">
            <link rel="stylesheet" href='<c:url
value="/static/css/geral.css" />' />
            <link href="<c:url
value="/static/css/jquery.dataTables.min.css" /> "rel="stylesheet" />
      </jsp:attribute>

      <jsp:attribute name="scriptsEspecificos">
            <script src="../../static/js/bootbox.js"></script>
            <script
src="../../static/js/jquery.dataTables.min.js"></script>
            <script
src="../../static/js/dataTables.bootstrap.js"></script>
        </jsp:attribute>
```

Listagem 10 - JSP

Na Listagem 11 é apresentado o script responsável por gerar a tabela com os dados retornados em formato Json do *controller* por meio de uma requisição AJAX. A tabela é gerada utilizando o *plugin* DataTables sendo informados os dados e as estilizações de exibição.

```
var oTable;
$(document).ready(function(){
      oTable = $('#tabela').dataTable({
            "ajax" : "/privado/contareceber/dados",
            "language" : {
                  "sEmptyTable" : "Nenhum registro encontrado",
                  "sInfo" : "Mostrando de _START_ até _END_ de _TOTAL_
registros",
                  "sInfoEmpty" : "Mostrando 0 até 0 de 0 registros",
                  "sInfoFiltered" : "(Filtrados de _MAX_ registros)",
                  "sInfoPostFix" : "",
                  "sInfoThousands" : ".",
                  "sLengthMenu" : "Mostrar _MENU_ registros por página",
                  "sLoadingRecords" : "Carregando...",
                  "sProcessing" : "Processando...",
            "sZeroRecords" : "Nenhum registro encontrado",
                  "sSearch" : "Pesquisar:",
                  "oPaginate" : {
                        "sNext" : "Próximo",
                        "sPrevious" : "Anterior",
                        "sFirst" : "Primeiro",
                        "sLast" : "Último"
                  },
                  "oAria" : {
                        "sSortAscending" : ": Ordenar colunas de forma
ascendente",
                        "sSortDescending" : ": Ordenar colunas de forma
descendente"
                  }
            },
            "aoColumns": [
             null, null, { "sType": "date-br" }, null, { "sType": "date-
br" }, null, null, null, null, null
            ],
            "order": [[ 2, "asc" ]],
            "columnDefs": [
                  {"className": "dt-center", "targets": "_all"},
                  { "orderable": false, "targets": [9] }
                  ]
      });//Fim dataTable
```

Listagem 11 – DataTable

Na Listagem 12 é apresentado um formulário para inclusão e edição de um cadastro, na *action* do formulário é informado o caminho para ser submetido os dados ao *controller*. O *value* dos campos é recuperado do *model* recebido do *controller* por meio da sintaxe informando o nome do *model* e o campo da tabela desejado, como por exemplo: ${cargo.id}.

```
<div class="container">
      <form id="frm" action="<c:url value="/privado/cargo/"/>"
            method="POST" class="well span6" onSubmit="return
validar(this);">
            <div class="form-group">
                  <div class="row">
                        <div class="col-xs-2">
                              <label for="id">Código:</label>
                              <input type="text" id="id" name="id"
class="form-control" value="${cargo.id}" readonly />
```

```
                        </div>
                        <div class="col-xs-8"></div>
                        <div class="col-xs-2">
                              <label for="situacao">Situação:</label>
                              <div class="check">
                                    <div class="check-success">
                                    <input type="checkbox" name="situacao"
id="situacao"
                                                value="1" ${cargo.situacao
== 2 ? '' : 'checked'}>
                                    <label for="situacao"
id="labelsituacao">${cargo.situacao == 2 ? 'Inativo' : 'Ativo'}</label>
                              </div>
                        </div>
                  </div>
                        </div>                                          </div>
                                                <div class="form-group">
                  <div class="row">
                        <div class="col-xs-4">
                              <label for="nome">Descrição:</label>
                              <input type="text" id="nome" name="nome"
class="form-control" required="required" value="${cargo.nome}" />

                        </div>
                        <div class="col-xs-2">
                              <label for="salarioBase">Salario
Base:</label>
                              <input type="text" id="salarioBase"
name="salarioBase" class="form-control money" value="${cargo.salarioBase}" />
                        </div>
                  </div>
            </div>
            <button type="submit" class="btn btn-primary">Salvar</button>
            <a href="/privado/cargo/" class="btn btn-default"
id="cancelar">Cancelar</a>
      </form>
</div>
```

Listagem 12 - Form JSP

4.4.2 Aplicativo *Mobile*

A comunicação entre o sistema *web* e o aplicativo é realizada por meio da troca de arquivos no formato JSON utilizando os métodos HTTP POST e HTTP GET conforme código apresentado na Listagem 13.

```
public class WebClient {
    private final String url;
    public WebClient(String url) {
        this.url = url;
    }
    public String post(String json) {
        HttpClient httpClient = new DefaultHttpClient();
        HttpPost post = new HttpPost(url);
        try {
            post.setEntity(new StringEntity(json));
            post.setHeader("Content-type", "application/json");
            post.setHeader("Accept", "application/json");
            HttpResponse response = httpClient.execute(post);
            InputStream content = response.getEntity().getContent();
            BufferedReader buffer = new BufferedReader(new
InputStreamReader(content));
            String result = "";
            String retorno = "";
            while ((result = buffer.readLine()) != null) {
                retorno += result;
            }
            return retorno;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
    public String get() {
        String response = "";
        HttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(url);
        try {
            httpGet.setHeader("Accept", "application/json");
            HttpResponse execute = httpClient.execute(httpGet);
            InputStream content = execute.getEntity().getContent();
            BufferedReader buffer = new BufferedReader(new
InputStreamReader(content));
            String s = "";
            while ((s = buffer.readLine()) != null) {
                response += s;
            }
            return response;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

Listagem 13 - Comunicação entre Sistema Web e Aplicativo Mobile

Para que o processo de sincronização ocorra foi criada a classe "SincronizacaoServiceImp" apresentada na Listagem 14. Essa classe estende a classe *AsyncTask* para facilitar o processamento em background. A classe *AsyncTask* encapsula todo o processo de criação de *threads* e *handler* e como ela é genérica possui os quatro métodos a seguir apresentados que foram sobrescritos (ROMANATO, 2017).

a) *onPreExecute* - é executado sempre antes da *thread* ser iniciada. Esse método foi utilizado para iniciar o *ProgressDialog* e exibi-lo ao usuário;

b) *doInBackground* - é o responsável pela maior parte do processamento, pois ele é executado em uma *thread* separada para que a tela do usuário não fique travada durante o processamento. Nesse método foi realizada a chamada à classe "WebClient" exibida na Listagem 13, passando o *Json* para que ele seja enviado ao sistema *web*. O conteúdo retornado pelo sistema *web* é armazenado na variável "jsonResposta" para ser passado por parâmetro para o método *onPostExecute*;

c) *onPostExecute* - é o método que recebe o retorno do *doInBackground*. Este método é responsável por retornar o resultado da sincronização que são os dados retornados pelo sistema *web* no formato *Json* para a *Activity* principal e após isso é finalizado o *ProgressDialog* removendo-o da tela.

d) *onProgressUpdate* - é o responsável por receber as informações do andamento da solicitação para exibir ao usuário

```
public class SincronizacaoServiceImp extends AsyncTask<Object, Object,
String> {
    private Context context;
    private ISincronizar iSincronizar;
    private ProgressDialog progress;
    private String json;
    private String msgProgress;
    private String tipo;
    private final String URL;
    public SincronizacaoServiceImp(Context context, ISincronizar
iSincronizar, String URL, String json, String tipo, String msgProgress) {
        this.URL = URL;
        this.json = json;
        this.iSincronizar = iSincronizar;
        this.context = context;
        this.tipo = tipo;
        this.msgProgress = msgProgress;
    }
    @Override
    protected void onPreExecute() {
        this.progress = new ProgressDialog(this.context);
        this.progress.setIndeterminate(true);
        this.progress.setMessage(msgProgress);
        this.progress.show();
    }
    @Override
    protected String doInBackground(Object... params) {
        String jsonResposta = null;
        if (tipo.equalsIgnoreCase("get")) {
            try {
                if (!json.isEmpty()){
                    jsonResposta = new WebClient(URL +
URLEncoder.encode(json, "UTF-8")).get();
                }else{
                    jsonResposta = new WebClient(URL).get();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
```

```
        } else {
            try {
                if (!(("[]".equals(json)) || (json == null))) {
                    jsonResposta = new WebClient(URL).post(json);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return jsonResposta;
    }
    protected void onPostExecute(String results) {
        this.iSincronizar.depoisDeSincronizar(results);
        this.progress.dismiss();
    }
    @Override
    protected void onProgressUpdate(Object... values) {
        this.progress.setMessage((CharSequence) values[0]);
    }
```

Listagem 14 - Processamento das Requisições

A Listagem 15 exibe o código responsável por gerar a lista expansível exibida na listagem de locações do cliente. O método *getGroupView* é responsável por preencher e retornar os itens principais da lista e o método *getChildView* é responsável por preencher e retornar os dados dos subitens da lista que serão exibidos ao clicar no item principal da lista.

```
public class ExpandableListAdapter extends BaseExpandableListAdapter {
    private List<String> lstGrupos;
    private HashMap<String, List<Locacao>> lstItensGrupos;
    private Context context;

    public ExpandableListAdapter(Context context, List<String> grupos,
HashMap<String, List<Locacao>> itensGrupos) {
        this.context = context;
        lstGrupos = grupos;
        lstItensGrupos = itensGrupos;
    }
    @Override
    public int getGroupCount() {
        return lstGrupos.size();
    }
    @Override
    public int getChildrenCount(int groupPosition) {
        return lstItensGrupos.get(getGroup(groupPosition)).size();
    }
    @Override
    public Object getGroup(int groupPosition) {
        return lstGrupos.get(groupPosition);
    }
    @Override
    public Object getChild(int groupPosition, int childPosition) {
        return
lstItensGrupos.get(getGroup(groupPosition)).get(childPosition);
    }
    @Override
    public long getGroupId(int groupPosition) {
        return groupPosition;
    }
```

```
    @Override
    public long getChildId(int groupPosition, int childPosition) {
        return childPosition;
    }
    @Override
    public boolean hasStableIds() {
        return false;
    }
    @Override
    public View getGroupView(int groupPosition, boolean isExpanded, View
convertView, ViewGroup parent) {
        if (convertView == null) {
            LayoutInflater layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = layoutInflater.inflate(R.layout.group_item,
null);
        }
        TextView tvData = (TextView)
convertView.findViewById(R.id.tvData);
        tvData.setText((String) getGroup(groupPosition));
        return convertView;
    }
    @Override
    public View getChildView(int groupPosition, int childPosition, boolean
isLastChild, View convertView, ViewGroup parent) {
        if (convertView == null) {
            LayoutInflater layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = layoutInflater.inflate(R.layout.child_item,
null);
        }
        TextView tvVeiculo = (TextView)
convertView.findViewById(R.id.tvVeiculo);
        TextView tvSeguro = (TextView)
convertView.findViewById(R.id.tvSeguro);
        TextView tvDataPrevista = (TextView)
convertView.findViewById(R.id.tvDataPrevista);
        TextView tvDataDevolucao = (TextView)
convertView.findViewById(R.id.tvDataDevolucao);
        TextView tvValor = (TextView)
convertView.findViewById(R.id.tvValor);
        Locacao locacao = (Locacao) getChild(groupPosition,
childPosition);
        tvVeiculo.setText(locacao.getVeiculo());
        tvSeguro.setText(locacao.getSeguro());
        tvDataPrevista.setText(locacao.getDataPrevista());
        tvDataDevolucao.setText(locacao.getDatDevolucao());
        tvValor.setText(locacao.getValor());
        return convertView;
    }
    @Override
    public boolean isChildSelectable(int groupPosition, int childPosition)
{
        return true;
    }
}
```

Listagem 15 - Lista Expansível

A Listagem 16 exibe o código responsável pelo cálculo do valor previsto a ser pago pela reserva efetuada pelo cliente. Este método é chamado quando ocorre a alteração das informações de seguros, veículos e quantidades de dias da locação. O cálculo é realizado obtendo os valores desses campos e atualizando o valor total ao cliente.

```
private void calculaTotal() {
    try {
        Veiculo veiculo = (Veiculo) spVeiculo.getSelectedItem();
        Seguro seguro = (Seguro) spSeguro.getSelectedItem();
        NumberFormat currency = NumberFormat.getCurrencyInstance(new
Locale("pt", "BR"));
        Double valorVeiculo = veiculo.getValor();
        Double valorSeguro = seguro.getValor();
        if (etQtdDias.getText().toString().isEmpty() ||
etQtdDias.getText().toString().equalsIgnoreCase("0")) {
            etQtdDias.setText("1");
        }
        Integer qtdDias =
Integer.parseInt(etQtdDias.getText().toString());
        if (valorVeiculo == null) {
            valorVeiculo = 0D;
        }
        if (valorSeguro == null) {
            valorSeguro = 0D;
        }
        totalPagar = (valorVeiculo * qtdDias) + valorSeguro;
        String texto = "Valor da Diária do Veículo: " +
currency.format(valorVeiculo) + " \n";
        texto += "Valor do Seguro: " + currency.format(valorSeguro) + "
\n";
        texto += "Descrição do Seguro: " + seguro.getDescricao() + " \n";
        tvDetalhes.setText(texto);
        DecimalFormat df2 = new DecimalFormat("##,###,###,##0.00",
                new DecimalFormatSymbols(new Locale("pt", "BR")));
        df2.setMinimumFractionDigits(2);
        df2.setParseBigDecimal(true);
        String numeroFormatado = df2.format(totalPagar);
        etValor.setText(numeroFormatado);
    } catch (Exception ex) {
        ex.printStackTrace();
        etValor.setText("0,00");
    }
}
```

Listagem 16 - Cálculo do valor previsto da reserva

A Listagem 17 exibe o código responsável por recuperar o *Json* retornado do sistema *web* com os veículos disponíveis para reserva. Esse *Json* é percorrido por uma estrutura de repetição *for* sendo criado um *ArrayList* com os dados dos veículos para posteriormente serem adicionados no *Spinner* exibido ao cliente.

```
List<Veiculo> lstVeiculos = new ArrayList<>();
JSONArray jaReservas = jsonRetorno.getJSONArray("reservas");
JSONObject jsonVeiculo = new
JSONObject(String.valueOf(jaReservas.getJSONObject(0)));
JSONArray jaVeiculo = jsonVeiculo.getJSONArray("veiculos");
if (jaVeiculo.length() > 0) {
```

```
    for (int i = 0; i < jaVeiculo.length(); i++) {
        JSONObject obj = jaVeiculo.getJSONObject(i);
        Veiculo veiculo = new Veiculo();
        veiculo.setId(obj.optInt("idVeiculo"));
        veiculo.setNome(obj.optString("nomeVeiculo"));
        veiculo.setValor(obj.optDouble("valorVeiculo"));
        lstVeiculos.add(veiculo);
    }
}
if (!lstVeiculos.isEmpty()) {
    ArrayAdapter<Veiculo> adapterVeiculo = new
ArrayAdapter<Veiculo>(getActivity(), android.R.layout.simple_spinner_item,
lstVeiculos);

adapterVeiculo.setDropDownViewResource(android.R.layout.simple_spinner_dro
pdown_item);
    spVeiculo.setAdapter(adapterVeiculo);
}
```

Listagem 17 - Recuperação do arquivo JSON

5 CONCLUSÃO

O desenvolvimento deste trabalho consistiu em projetar e implementar um sistema *web* complementado por um aplicativo *mobile* para o gerenciamento de locadoras de veículos, registrando reservas, locações e devoluções de veículos, além de vários cadastros adicionais. Neste projeto foram inseridas as opções de geração de relatórios e gráficos das informações administrativas e financeiras do sistema.

Diante das diferentes possibilidades que as tecnologias utilizadas no desenvolvimento oferecem foram enfrentados dificuldades e desafios desde a modelagem do sistema, com pesquisa em locadoras e sistemas semelhantes, até a implementação *web* e *mobile*, pesquisando por servidores de aplicação e linguagens de programação que realizassem a comunicação entre essas duas plataformas. Durante o desenvolvimento foram realizados vários testes e alterações para chegar ao resultado esperado.

Este trabalho abordou todo o processo de desenvolvimento: da idealização de um sistema até a sua concretização. O sistema foi desenvolvido visando proporcionar ao usuário uma forma intuitiva e clara para que os usuários possam navegar entre as páginas com facilidade e rapidez.

O desenvolvimento *web* permite aos usuários o acesso ao sistema de qualquer dispositivo com acesso à Internet, sendo os menus responsivos, auxiliando na visualização das telas.

REFERÊNCIAS

AGUSTIN, José Luis Herrero. Model-driven web applications. **Science and Information Conference**, 2015, p. 954-964.

ASSOCIAÇÃO BRASILEIRA DE LOCADORAS DE AUTOMÓVEIS. MINISTÉRIO DO TURISMO. **Locação de automóveis**. Disponível em: <http://www.dadosefatos.turismo.gov.br/dadosefatos/estatisticas_indicadores/locacao_automoveis/>. Acesso em: 17 mar. 2016a.

ASSOCIAÇÃO BRASILEIRA DAS LOCADORAS DE AUTOMÓVEIS. Anuário Associação Brasileiro do Setor de Locação de Automóveis. Disponível em: <http://www.abla.com.br/wp-content/uploads/2016/03/ABLA.pdf>. Acesso em: 31 mai. 2016b.

FRATERNALI, Piero; ROSSI, Gustavo; SÁNCHEZ-FIGUEROA, Fernando. Rich Internet Applications. **IEEE Computer Society**, May/June 2010, p.9-12.

GARRETT, Jesse James. 2005. **Ajax: a new approach to web applications**. Disponível em: < http://adaptivepath.org/ideas/ajax-new-approach-web-applications/>. Acesso em: 22 abr. 2016.

HERTZ. **Aluguel de carros**. Disponível em: <https://www.hertz.com.br/rentacar>. Acesso em: 17 mar. 2016.

LOCALIZA. **Aluguel de carros**. Disponível em: <https://www.localiza.com/brasil >. Acesso em: 17 mar. 2016.

MELIÁ, Santiago, GÓMEZ, Jaime; PÉREZ, Sandy; DÍAZ, Oscar. **Facing Architectural and Technological Variability of Rich Internet Applications.** IEEE Internet Computing, 2010, p. 1-7.

PANG, Zhen; WEN, Fuan; PAN, Xiwei; LUI, Cen. **Migration model for Rich Internet Applications based on PureMVC Framework**. In: International Coriference on Computer Design and Appliations (ICCDA 2010), v. 5, 2010, p. v5-340-v5-343.

PAULSON, Linda Dailey. Building rich web applications with Ajax. **Computer IEEE**, v.38, n.10, p. 14-17, 2005.

RENT A CAR. **Vantagens de alugar um carro para as empresas**. Disponível em: <http://www.rentacarnews.com.br/vantagens-de-alugar-um-carro-para-as-empresas/>. Acesso em: 17 mar. 2016.

ROMANATO, Allan. **Trabalhando com AsyncTask no Android.** 2017 Disponível em: <http://www.devmedia.com.br/trabalhando-com-asynctask-no-android/33481>. Acesso em: 24 mar. 2017.

MVNREPOSITORY. **What's New in Maven.** Disponível em: <https://mvnrepository.com/>. Acesso em: 19 jun. 2017