ETEC IRMÃ AGOSTINA

Centro Paula Souza

Técnico de Informática

Caroline Muniz
Gabriel Magalhães
Hilton Mendes
Jefferson da Silva
Victor Luckesy
Vinicius Superbi

AutoApp: Aplicativo de prevenção de sinistros e controle de gastos com manutenções, despesas e abastecimentos de veículos.

Trabalho de Conclusão de Curso

Volume Único

São Paulo
Novembro de 2016

Caroline Muniz
Gabriel Magalhães
Hilton Mendes
Jefferson da Silva
Victor Luckesy
Vinicius Superbi

AutoApp: Aplicativo de prevenção de sinistros e controle de gastos com manutenções, despesas e abastecimentos de veículos.

Trabalho de Conclusão de Curso apresentado ao Programa de Formação Técnica em Informática da Etec Irmão Agostina, como parte dos requisitos necessários à obtenção do título de Técnico em Informática.

Orientador: Renato Carvalho dos Santos

Volume Único

São Paulo
Novembro de 2016

Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.

Charles Chaplin

RESUMO

A aplicação móvel AutoApp busca auxiliar os motoristas no controle das manutenções e despesas com seus veículos. Esse controle é realizado através do acompanhamento de inserções realizadas no próprio aplicativo, que se condensam em uma grade de histórico. Essa funcionalidade permite que o motorista gerencie de forma simples as manutenções realizadas em seu veículo, bem como as despesas com abastecimentos e impostos. Esse auxilio também contribui para prevenção de futuras falhas mecânicas. Criado para plataforma android, todo seu desenvolvimento foi feito na IDE AndroidStudio. As manutenções disponíveis foram adotadas com base em uma pesquisa prévia. Toda criação, desde a descrição e levantamento de requisitos até codificação e definição de layout foi documentada e sintetizada nesse trabalho.

Palavras-chave: Aplicação; Android; Motoristas; Veículos; Manutenção; Despesas; Abastecimentos.

ABSTRACT

The mobile application AutoApp seeks to support drivers in maintenance control and expenses with their vehicles. This control is carried out through insertions accompaniment held in the application itself, that condences in a historic grid . This function allows the driver to manage, in a easy way, the maintances carried out in his/her vehicle, as well as supplies and taxes expenses. This aid also contributes to future mechanical failures. Created for Android platform, all its development was made in AndroidStudio IDE. The maintenances available were adopted on the basis of a previous research. All the criation, since the description and requirements gathering until codification and layout definition was documented and synthesised in this work.

Keywords: Application; Android; Drivers; Vehicles; Maintenance; Expenses; Supplies.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A informatização em diversos setores de nossas vidas tem se tornado cada vez mais frequente. Além de facilitar e automatizar determinadas tarefas que antes eram vistas como monótonas e enfadonhas, a informática também contribui para o melhoramento e precisão da qualidade das informações anteriormente realizadas somente pela mente humana. É com esse pretexto que o aplicativo móvel para smartphones android *AutoApp* pretende auxiliar os motoristas no controle de gastos e manutenções de seus veículos, contribuindo para prevenção de sinistros e perda de prazos de vencimentos, além de oferecer registros de despesas e gastos e maior controle nas realizações de revisões periódicas.

Há muito tempo já se sabe que a aquisição de um automóvel é motivo para preocupações com gastos e despesas, essa premissa é reforçada pela expressão popular de que se ter um carro é como ter um filho, pois ambos requerem gastos constantes e muita atenção, o que não deixa de ser verdade. E se para se ter um filho é necessário muito planejamento e controle desde o primeiro ano de vida da criança, com um veículo não é diferente. Uma pesquisa veiculada pela revista automotiva Vruum , indica que a soma de todas as despesas no primeiro ano após a aquisição de um veículo, incluindo o valor das prestações, é de aproximadamente R$ 10 mil[1], ou cerca de R$ 840 por mês (GRECO, 2015). Considerando que o salário médio da população brasileira, de acordo com apontamento da IBGE (2015) do segundo semestre de 2015 , é de R$1.113,00 , o gasto total com veículo representa cerca de 48% da renda mensal dos brasileiros. Nota-se, portanto, a necessidade imprescindível de se gerenciar as despesas e gastos com veículos, visto que, devido a proporção que esses valores representam na conjectura salarial, impactam drasticamente no orçamento familiar e individual.

Outro aspecto que reforça a pertinência dessa aplicação está nos acidentes de trânsito. Em março de 2010, a Assembleia-Geral das Nações Unidas editou uma resolução definindo o período de 2011 a 2020 como à "Década de ações para a segurança no trânsito" . O documento, implementado com base em um estudo elaborado pela Organização Mundial da Saúde (OMS), contabiliza, somente em 2009 (ano de elaboração do estudo), cerca de 1,3 milhão de mortes por acidente de trânsito em 178 países (ORGANIZAÇÃO DAS NAÇÕES UNIDAS, 2012). O Brasil aparece em quinto lugar entre os países recordistas em mortes de trânsito. Segundo dados do Ministério da Saúde, em 2014 (último ano disponível) foram cerca de 43 mil mortes, sendo que em 12% dos acidentes foram constatadas falhas mecânicas. Além de causar tristeza

[1] Esse valor refere-se a um carro popular com valor entre R$20 e R$30 mil.

e sofrimento por parte dos familiares e amigos das vítimas, os acidentes de trânsito também provocam um alto custo para o Fundo Nacional de Segurança e Educação de Trânsito – FUNSET relativo ao seguro obrigatório de Danos Pessoais causados por Veículos Automotores de Via Terrestre – DPVAT. Segundo estudo publicado pelo DENATRAN e pelo Instituto de Pesquisa Econômica Aplicada – IPEA em 2006, com dados coletados em 2004 e 2005, o custo social dos acidentes em rodovias foi estimado em cerca de R$ 24,6 bilhões anuais, dos quais R$ 8,1 bilhões correspondiam aos acidentes nas rodovias federais e R$ 16,5 bilhões nas estaduais. A pesquisa constatou que o custo médio do acidente com feridos fica em torno de R$ 90 mil e, com mortes, este valor chega a R$ 421 mil. Em estudo semelhante realizado em 2004 pelo DENATRAN, pelo IPEA e pela Associação Nacional de Transportes Públicos – ANTP para os aglomerados urbanos, a estimativa do custo social de acidentes de trânsito naqueles locais foi de R$ 5,3 bilhões anuais. Tomando-se os dois estudos, o custo social total no Brasil é da ordem de R$ 30 bilhões anuais. Evidentemente que a subtração de 12% desse valor, relativo aos acidentes por falhas mecânicas, não dirimiriam a escabrosidade desses números, no entanto possibilitaria que parte dos recursos financeiros do setor da saúde que é drenada para o atendimento de urgência e traumatologia, para a reabilitação e a inclusão social de acidentados de trânsito fossem destinados para às ações de engenharia, fiscalização e educação de trânsito, portanto, em ações voltadas para a redução de acidentes.

As falhas mecânicas são, de acordo com a Polícia Rodoviária Federal, a quarta maior causa de acidentes declaradas pelos motoristas. Apenas em São Paulo, em média, são registrados 784 atendimentos diariamente por veículos que apresentam falhas mecânicas ou panes elétricas, segundo informações da Agência de Transporte do Estado de São Paulo – Artesp. Manter a revisão do veículo atualizada é a principal recomendação da Polícia Rodoviária Federal para prevenir-se. A verificação periódica de alguns itens do veículo resulta numa condução mais econômica e principalmente mais segura. É o que aponta pesquisa realizada pelo aplicativo espanhol Confortauto. Segundo o estudo, a falta de manutenção aumenta em três vezes a possibilidade do veículo se envolver em acidente. "Para garantir a segurança do condutor ou do piloto e dos passageiros, é necessário manter o veículo em perfeito estado de conservação e funcionamento" ,explica Celso Alves Mariano, especialista e diretor da Tecnodata Educacional, empresa que comercializa sistemas de educação para o trânsito.

Com a aplicação *AutoApp*, os motoristas poderão registrar e gerenciar as realizações de manutenções em seus veículos, o que fará com que os condutores mantenham a manutenção de seu carro em dia, evitando riscos de falhas mecânicas e prevenindo acidentes. Além disso, a aplicação possibilitará o controle de gastos com combustível e outras despesas, como financiamento, seguro, impostos, multas, lavagem e estacionamento, o que garantirá o melhor gerenciamento de recursos, possibilitando economia e

controle. Essa aplicação prima pela pela segurança e conforto dos motoristas, possibitando maiores com cuidados do seu patrimônio, tudo isso em uma interface gráfica amigável, agradável e de fácil interagibilidade.

1.1 Sobre os Desenvolvedores

Composto por seis integrantes, a formação da equipe de desenvolvedores desse projeto se deu antes mesmo do período adequado. Ainda no primeiro módulo do curso, as afabilidades e bom convivimento foram responsáveis pelo aflorescimento de respeito e admiração mutúa entre os integrantes, contribuindo para formação natural do grupo. Caroline Muniz, Gabriel Magalhães, Hilton Mendes, Jefferson da Silva, Victor Luckesy e Vinicius Superbi, fomos responsáveis juntos pela confecção desse trabalho.

2 DESENVOLVIMENTO

2.1 Pesquisa de Campo

Segundo Gil (2002, 17), pesquisa é definida como um

> (...) procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. A pesquisa desenvolve-se por um processo constituído de várias fases, desde a formulação do problema até a apresentação e discussão dos resultados.

Enquanto indagávamo-nos sobre quais seriam as pretensões da aplicativo *AutoApp*, notamos uma dificuldade técnica e empírica em descrever quais seriam as funcionalidades especificas que a aplicação teria. A necessidade de se elaborar uma pesquisa de campo, portanto, deveu-se a essa problemática e abrangência informações, colecionadas através de pesquisas bibliográficas, que poderiam serem agregadas à aplicação. Incertos sobre como contornar esse empecilho, resolvemos consultar o nosso público alvo, os motoristas, e questioná-los sobre quais características uma aplicação que se propunha a auxilia-los no gerenciamento de seus veículos deveria ter. Para isso, elaboramos um plano de pesquisa condizente com a proposição interrogativa, o que serviu como embasamento para todo processo de desenvolvimento da aplicação.

2.1.1 Metodologia de Pesquisa

O método de pesquisa que adotamos é o método indutivo, proposto por empiristas como Bacon, Hobbes, Locke e Hume, para os quais o conhecimento é fundamentado na experiência (GERHARDT; SILVEIRA, 2009) . De acordo com o raciocínio indutivo, na análise de Gil (2002, 10)

> (...)a generalização não deve ser buscada aprioristicamente, mas constatada a partir da observação de casos concretos suficientemente confirmadores dessa realidade.

Assim, detectamos procedimentos realizados frequentemente pelos motoristas, através de uma coleta de dados, e estabelecemos uma generalização que nos permitiu abarcarmos as técnicas mais recorrentes aplicadas na manutenção de veículos, abrangendo a atuação do aplicativo.

2.1.2 O Tipo de Pesquisa

2.1.2.1 Quanto à abordagem

Embora a qualidade nos seja preciosa, precisávamos estabelecer um retrato numérico que definisse a frequência dos tipos de manutenções e despesas que oneram os condutores, portanto recorremos à uma pesquisa quantitativa. Com esse modelo de pesquisa foi possível quantificar as ocorrências de mantença que assolavam com maior grau de incidência os veículos dos motoristas e estabelecer os itens que compreendem a aplicação.

2.1.2.2 Quanto à natureza

Estabelecemos uma pesquisa aplicada que visa estabelecer uma verdade de interesse local, isso é, sem pretensões de gerar conhecimento novos, mas apenas constatar informações para aplicação prática, dirigidos à solução de problemas específicos.

2.1.2.3 Quanto aos objetivos

Nosso plano de pesquisa tem caráter exploratório. Este tipo de pesquisa tem como objetivo proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a construir hipóteses. A grande maioria dessas pesquisas envolve: (a) levantamento bibliográfico; (b) entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; e (c) análise de exemplos que estimulem a compreensão (GIL, 2002).

Como já mencionado, devido à dificuldade técnica e empírica em descrever quais seriam as funcionalidades especificas que a aplicação teria, percebemos a necessidade de nos familiarizarmos mais com a regra de negócio abordado pela nossa aplicação.

2.1.2.4 Quanto aos procedimentos

De acordo com Fonseca (2002) , a pesquisa possibilita uma aproximação e um entendimento da realidade a investigar, e reconhecendo o valor dessa sentença proferida por ele, decidimos que a pesquisa de campo é a que melhor permite compreendermos as necessidades do nosso público alvo. A pesquisa de campo é uma investigação realizada através de um levantamento de dados que permite aos investigadores conhecer o funcionamento de um determinado evento ou fenômeno. Alia-se a isso, a frequência do evento ou fenômeno, item que nos interessa e motiva nessa investigação.

2.1.3 Universo e Amostra

Em uma pesquisa realizada pela Associação Nacional dos Detrans (AND), em parceria com os Detrans Estaduais, com a finalidade de traçar o perfil dos motoristas brasileiros, foi constatado que o número de motoristas habilitados no país já supera a marca de 60,7 milhões. No entanto, para representar esse número em escala micro, aplicamos a pesquisa em forma de formulário (em anexo) a 13 condutores de veículos, o que formou nossa amostragem e possibilitou o delineamento das manutenções mais realizadas pelos motoristas, assim como sugestões arguidas pelos mesmos.

2.1.4 Aplicação e Coletas de Dados da Pesquisa

A aplicação desta pesquisa deu-se através da rede social Facebook. Acompanhado de um texto explicativo, divulgamos o link da pesquisa em grupos especificos de interesses de mecânica automotiva. Esse link redirecionava o entrevistado para um formulário, construido através da ferramenta Google Forms da empresa Google Inc, e neste formulário o entrevistado votava nas manutenções mais recorrentemente realizadas pelos motoristas, previamente elencadas. Após a finalização da pesquisa, a ferramenta Google Forms gera um gráfico com as respostas, conforme a imagem a seguir.

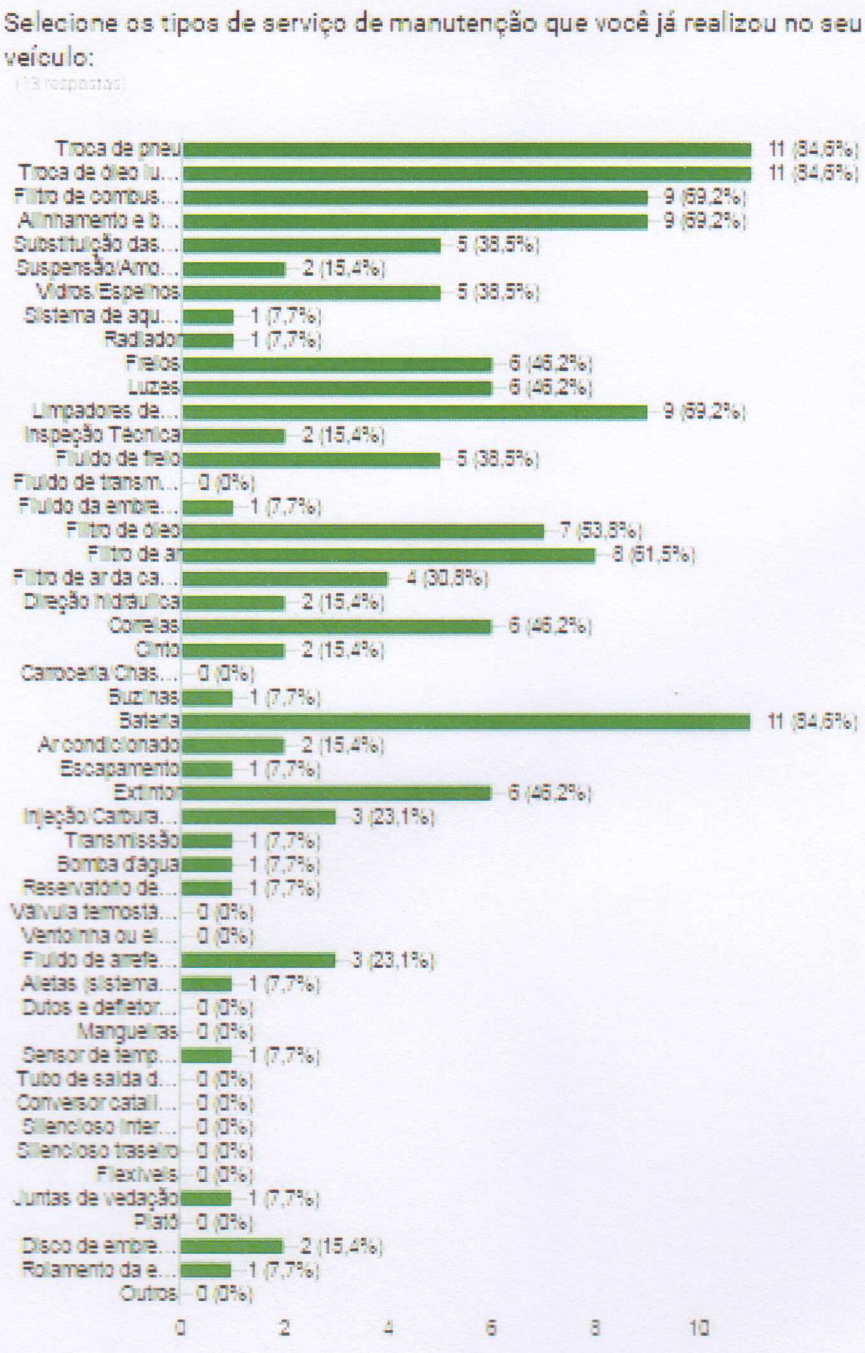

Figura 1 – Gráfico da pesquisa de campo

No final da pesquisa o entrevistado podia, opcionalmente, responder a pergunta "Quais tipos de informações sobre o seu veículo você gostaria de armazenar em uma aplicação de gerenciamento de carros?". Para essa pergunta, obtemos o total de 4 respostas, que estão transcritas a seguir.

- "Datas das manutenções e quais peças foram trocadas e também desempenho como consumo de combustível por exemplo";

- "Datas das manutenções e quais peças foram trocadas e também desempenho como consumo de combustível por exemplo";
- "Funcionamento e mapa do motor. Desempenho do motor e outras parte, como ar condicionado, filtro de óleo, suspensão ...";
- ''Cuidados básicos desde troca de óleo a troca de pneu por exemplo, uma manutenção preventiva de ué cada pessoa deveria fazer no carro todo dia antes de sair com o mesmo".

2.1.5 Análise da Pesquisa

Para analisar os dados, nos foi preciso somente avaliar a maior incidência de votos e a pertinência das respostas abertas. Mesmo verficando que algumas manutenções são mais recorrentes que outras, todas elas foram elencadas na aplicação. Quanto as sugestões arguidas pelas respostas abertas, apenas algumas foram atendidas e implementadas. As que não foram utilizadas foram por implicações técnicas ou impertinência.

2.2 Modelagem de Dados Conceitual

A modelagem de dados conceitual trata-se de uma técnica sistemática de alto nível de abstração, criada em 1970 pelo cientista da computação americano Dr. Peter Pin- Shan Chen, para representar dados ou aspectos de informação de um domínio de negócio ou seus requerimentos de processos. Baseado na percepção do mundo real, o diagrama que deve ser construído aqui é o Diagrama de Entidade e Relacionamento, uma técnica de modelagem que representa graficamente a Modelagem de Entidade –Relacionamento e que reúne uma coleção de conceitos como Entidade, Atributos e Relacionamentos. Entidades são objetos ou partes de um domínio. Atributos são as características da Entidade. Já Relacionamentos são os meios de interação entre as entidades. O estudo dos dados conceituais possibilitam o levantamento de requisitos de um sistema, que pode ser descrito de forma textual, conforme a Descrição Textual Macro da próxima seção.

2.2.1 Descrição Textual Macro

Ao executar a aplicação o usuário deverá inserir seu nome de usuário e senha. Haverá a possibilidade de memorizar esses dados para acessos futuros. Ele também poderá "Entrar com o Facebook", recurso que importará informações sobre o usuário e permitirá acesso instantâneo, exigindo apenas a complementação dos dados fundamentais para o funcionamento do aplicativo. Mas caso o usuário esteja acessando a aplicação pela primeira vez, poderá pressionar a opção "Ainda não tenho

cadastro", e será encaminhado para um formulário de cadastro onde ele deverá inserir o nome completo, data de vencimento da CNH (Carteira Nacional de Habilitação), o nome de usuário e a senha de acesso. No formulário seguinte ele deverá cadastrar o veículo que será gerenciado através da aplicação, devendo inserir dados como o nome do veículo, marca, modelo, placa, ano e estado em que o veículo foi emplacado, sendo ainda opcional o cadastro da seguradora do veículo, valor do seguro, validade do seguro. Todas essas informações são indispensáveis para o correto funcionamento do aplicativo.

Devidamente autenticado, o condutor terá como tela inicial um histórico dos seus últimos registros, que em caso de primeiro acesso estará vazio. Esses registros referem-se aos abastecimentos, despesas e manutenções previamente registradas pelo usuário na aplicação. Para registrar novas ocorrências, ele deverá pressionar o ícone de adicionar ou acessar o menu lateral, e determinar se tratasse de um abastecimento, despesa ou manutenção. Selecionado abastecimento, uma tela solicitando informações sobre o abastecimento será apresentada, com data e hora (que deverá ser preenchido automaticamente, mas terá opção de alteração para registros retroativos), hodômetro, preço, valor, litros, combustível, local e observação. Selecionando "Salvar", a ocorrência abastecimento será adicionada ao histórico de abastecimentos do veículo e poderá ser acessada em qualquer momento. A mesma operação será sujeitada as ocorrências de despesa e manutenção, variando apenas as informações solicitadas no momento do registro, sendo data, hora, tipo de despesa, observação, hodômetro, valor e local para despesa, e data, hora, tipo de serviço, observação, hodômetro, valor e local para manutenção.

O propósito da aplicação é permitir que seja realizado o controle no gerenciamento do veículo, e para isso servirá o histórico.

2.2.2 Ferramenta CASE

Do inglês Computer-Aided Software Enginnering, as ferramentas CASE são utilizadas para produção da modelagem de dados. Neste trabalho, sua utilização foi aplicada para o desenvolvimento do Diagrama de Entidade- Relacionamento, Diagrama UML, Diagrama de Caso de Uso, Diagrama de Banco de Dados e Diagrama de Classes e as ferramentas utilizadas foram DIA, Star UML, Visio e Server Management Studio.

2.2.3 Diagrama de Entidade- Relacionamento

O Diagrama de Entidade- Relacionamento é uma representação gráfica do modelo conceitual. Abaixo, verifica-se duas representações para a aplicação:

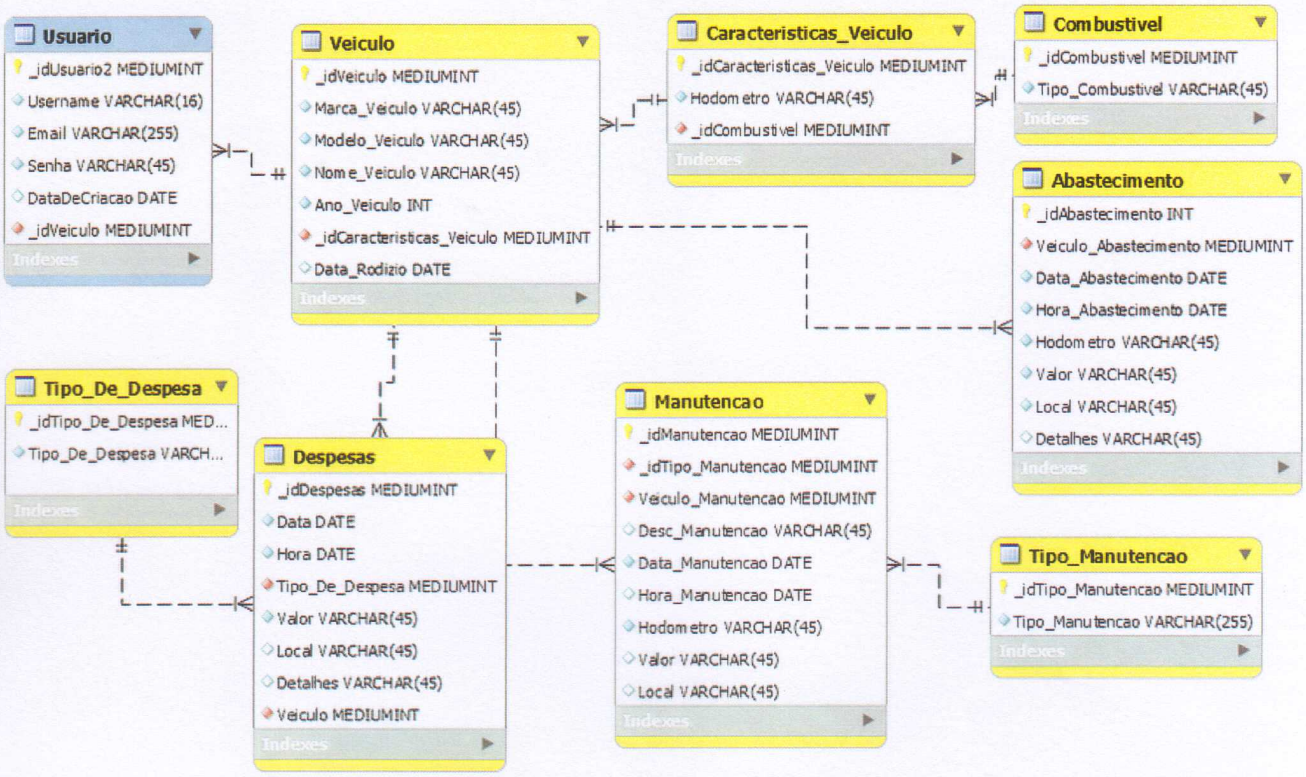

Figura 2 – Diagrama de Entidade- Relacionamento (1)

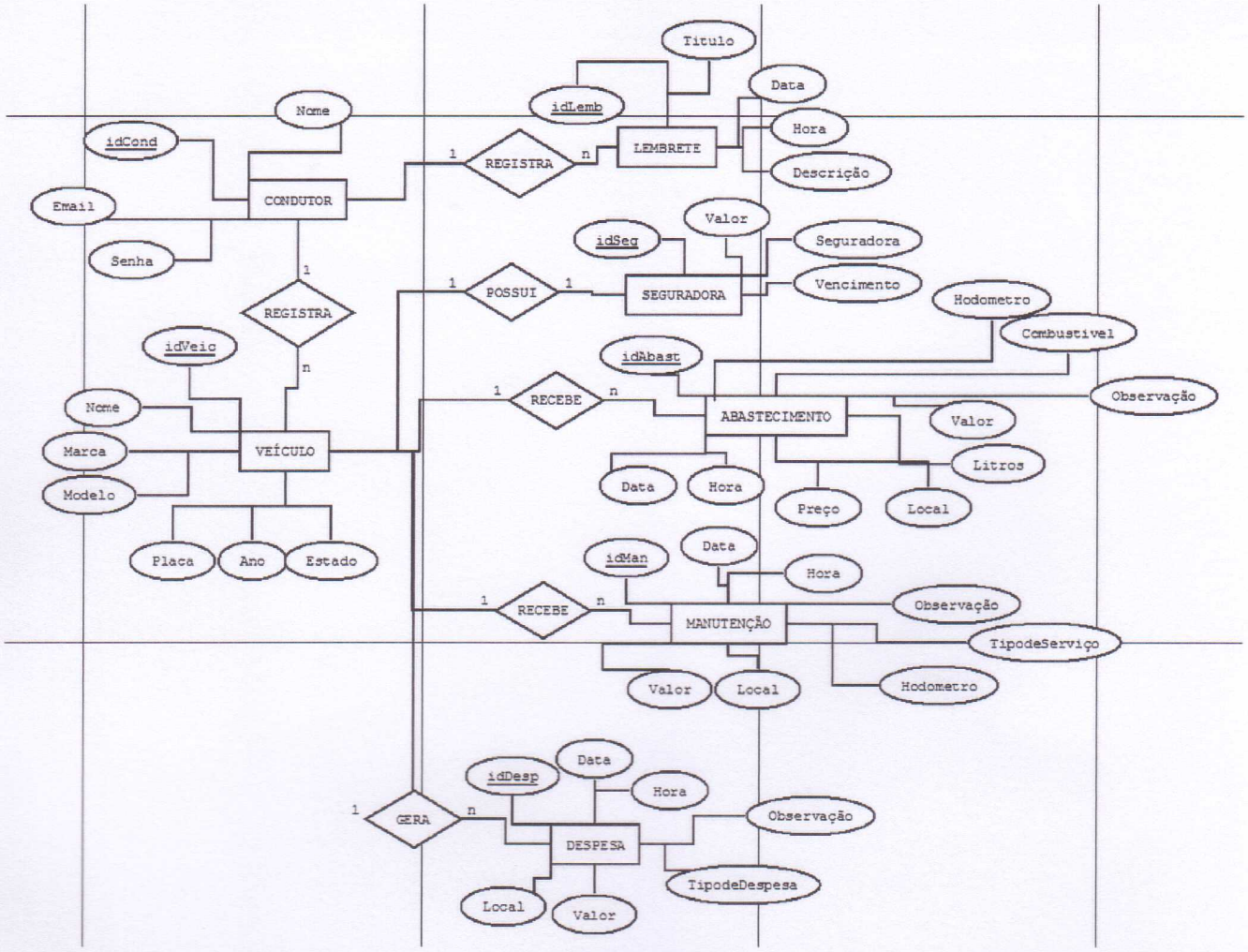

Figura 3 – Diagrama de Entidade- Relacionamento (2)

2.3 Modelagem de Dados Lógica

A modelagem de dados lógica é a fase em que são implementados recursos como adequação de padrão e nomenclatura, definição das chaves primárias e estrangeiras, normalização, reconhecimento dos atores e ações do usuário. Trata-se do Fluxograma, Descrição de Caso de Uso, Diagrama de banco de Dados e Diagrama de Classes apresentados a seguir:

2.3.1 Fluxograma

O fluxograma é uma representação esquemática dos processos do sistema. Os processos indicados nas figuras abaixo indicam a padrão de funcionalidade da aplicação *AutoApp*.

Ao executar a aplicação, o usuário deverá realizar o login com seu nome ou seu e-mail e senha. Caso não possui cadastro, deverá cadastrar-se pela primeira vez inserindo os dados solicitados. O registro do veículo que será gerenciado deverá ser

feito na sequência. Após a inserção dos dados do condutor (usuário) e do veículo, o cadastro das despesas, manutenções e abastecimentos poderão serem feitas.

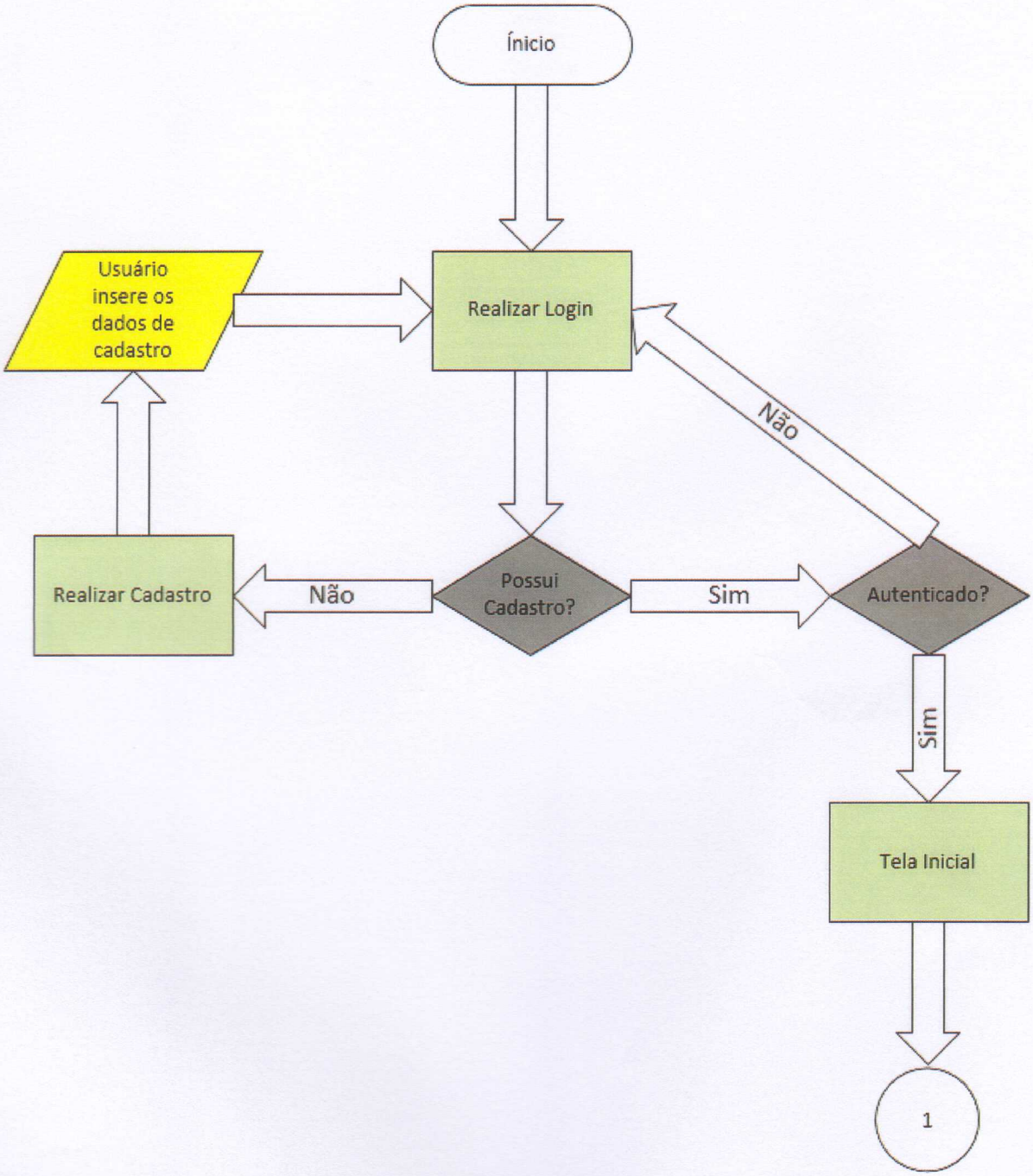

Figura 4 – Fluxograma(1)

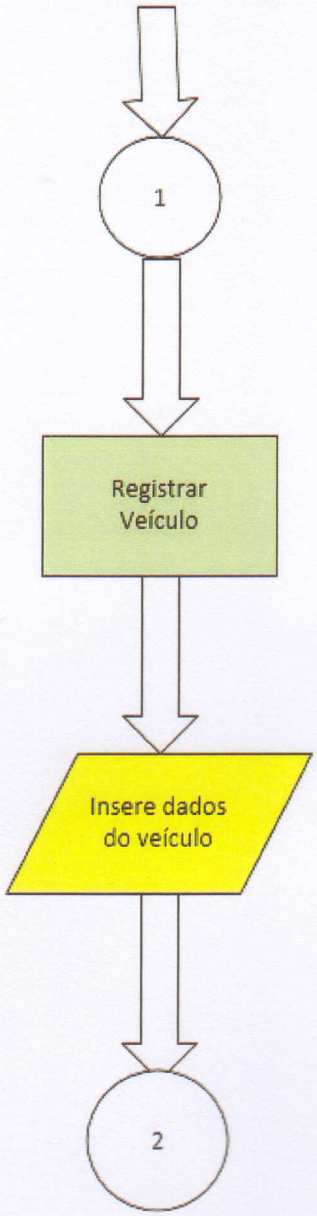

Figura 5 – Fluxograma (2)

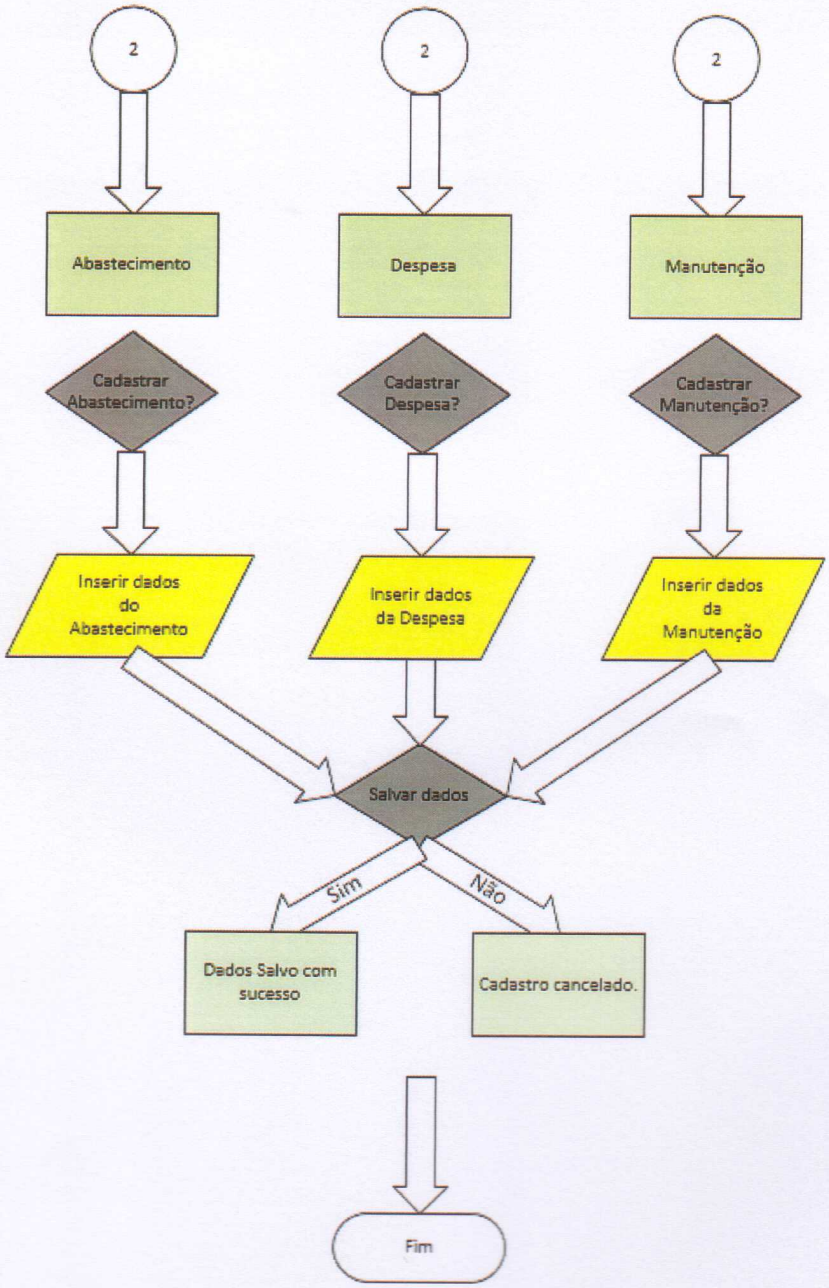

Figura 6 – Fluxograma (3)

2.3.2 Diagrama de Casos de Uso

Esse diagrama representa as possibilidades de interação do usuário com o sistema do ponto de vista do usuário. Na aplicação AutoApp, o ator é o condutor e suas ações são realizar login, realizar cadastro, registrar veículo, salvar registro, cancelar registro, cadastrar despesa, cadastrar abastecimento, cadastrar manutenção, salvar cadastro, consultar histórico, excluir ou editar itens dos históricos. A figura abaixo representa com maiores detalhes as funcionalidades:

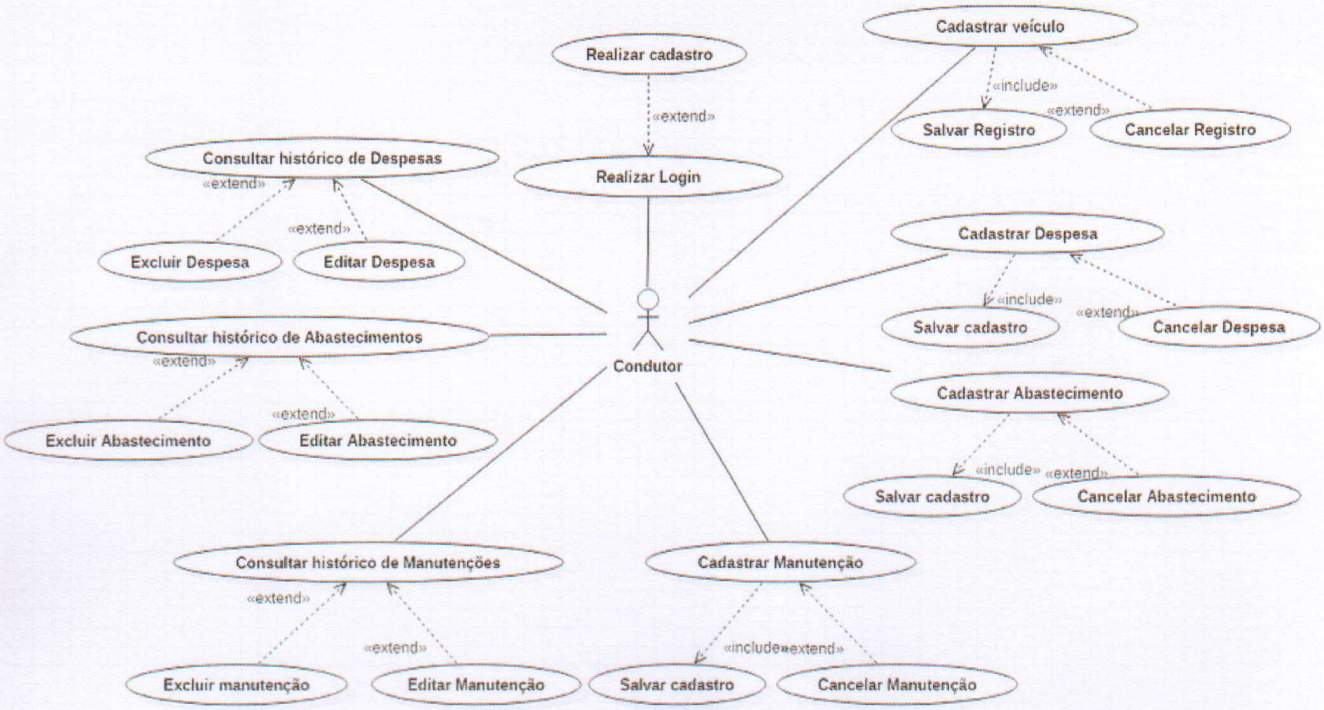

Figura 7 – Diagrama de Casos de Uso

2.3.2.1 Especificação de Caso de Uso

Abaixo estão especificados os casos de usos do Diagrama anterior.

Número Do Caso de Uso	FYC001
Nome do caso de uso	Realizar cadastro
Descrição	Permitirá que o usuário realize o seu cadastro
	permitindo que ele usufrute da aplicação
Pré-Condições	Não há.
Pós-Condições	Não há.
Cenário principal	1. A aplicação oferece ao usuário uma tela de cadastro com:
	1.1. Nome
	1.2. E-mail
	1.3. Senha
	1.4. Entrar com o Facebook
Cenário alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Não há.
Extensão(extends)	Realizar login

Figura 8 – FYC001

Número Do Caso de Uso	FYC002
Nome do caso de uso	Realizar Login
Descrição	Fase de acesso à conta do usuário
Pré-Condições	Possuir cadastro na aplicação
Pós-Condições	Ser autênticado
Cenário Principal	1. A aplicação oferece ao usuário uma tela de login com:
	1.1. Nome ou email
	1.2. Senha
	1.3. Entrar com o facebook
Cenário Alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Não há.
Extensão(extends)	Realizar cadastro

Figura 9 – FYC002

Número Do Caso de Uso	FYC003
Nome do caso de uso	Registrar Veículo
Descrição	Permitirá que o usuário realize o registro do veículo a ser
	gerenciado.
Pré-Condições	Estar logado
Pós-Condições	Dados inseridos serem validados.
Cenário principal	1. A aplicação oferece ao usuário uma tela de registro com:
	1.1. Nome
	1.2. Marca
	1.3. Modelo
	1.4. Placa
	1.5. Ano
	1.6. Estado
Cenário alternativo	Uma tela opcional para cadastro da seguradora será
	apresentada ao usuário.
Exceções	Não há.
Inclusão(includes)	Salva registro
Extensão(extends)	Cancelar Registro

Figura 10 – FYC003

Número Do Caso de Uso	FYC004
Nome do caso de uso	Cadastrar Despesa
Descrição	Permitirá que o usuário cadastre novas despesas do veículo gerenciado.
Pré-Condições	Estar logado
Pós-Condições	Não há.
Cenário principal	1. A aplicação oferece ao usuário uma tela de cadastro com:
	1.1. Data
	1.2. Hora
	1.3. Hodômetro
	1.4. Tipo de Despesa
	1.5. Valor
	1.6. Local
	1.7. Observação
Cenário alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Salvar Cadastro
Extensão(extends)	Cancelar Cadastro

Figura 11 – FYC004

Número Do Caso de Uso	FYC005
Nome do caso de uso	Cadastrar Abastecimento
Descrição	Permitirá que o usuário cadastre novos abastecimentos do veículo gerenciado.
Pré-Condições	Estar logado
Pós-Condições	Não há.
Cenário principal	1. A aplicação oferece ao usuário uma tela de cadastro com:
	1.1. Data
	1.2. Hora
	1.3. Hodômetro
	1.4. Preço
	1.5. Valor
	1.6. Litros
	1.7. Combustivel
	1.8. Local
	1.9. Observação
Cenário alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Salvar Cadastro
Extensão(extends)	Cancelar Cadastro

Figura 12 – FYC005

Número Do Caso de Uso	FYC006
Nome do caso de uso	Cadastrar Manutenção
Descrição	Permitirá que o usuário cadastre novas manutenções do veículo gerenciado.
Pré-Condições	Estar logado
Pós-Condições	Não há.
Cenário principal	1. A aplicação oferece ao usuário uma tela de cadastro com: 1.1. Data 1.2. Hora 1.3. Hodômetro 1.4. Tipo de Serviço 1.5. Valor 1.6. Local 1.7. Observação
Cenário alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Salvar Cadastro
Extensão(extends)	Cancelar Cadastro

Figura 13 – FYC006

Número Do Caso de Uso	FYC007
Nome do caso de uso	Consultar histórico de despesas
Descrição	Permitirá que o usuário consulte o histórico de depesas do veículo gerenciado.
Pré-Condições	Estar logado
Pós-Condições	Não há.
Cenário principal	1. A aplicação oferece ao usuário uma tela de histórico com todas despesas cadastradas previamente.
Cenário alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Não há.
Extensão(extends)	Excluir despesa. Editar despesa.

Figura 14 – FYC007

Número Do Caso de Uso	FYC008
Nome do caso de uso	Consultar histórico de abastecimentos
Descrição	Permitirá que o usuário consulte o histórico de abastecimentos do veículo gerenciado.
Pré-Condições	Estar logado
Pós-Condições	Não há.
Cenário principal	1. A aplicação oferece ao usuário uma tela de histórico com todos os abastecimentos cadastrados previamente.
Cenário alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Não há.
Extensão(extends)	Excluir abastecimento. Editar abastecimento.

Figura 15 – FYC008

Número Do Caso de Uso	FYC009
Nome do caso de uso	Consultar histórico de manutenções
Descrição	Permitirá que o usuário consulte o histórico de manutenções do veículo gerenciado.
Pré-Condições	Estar logado
Pós-Condições	Não há.
Cenário principal	1. A aplicação oferece ao usuário uma tela de histórico com todos as manutenções cadastradas previamente.
Cenário alternativo	Não há.
Exceções	Não há.
Inclusão(includes)	Não há.
Extensão(extends)	Excluir manutenção. Editar manutenção.

Figura 16 – FYC009

2.3.3 Diagrama de Banco de Dados

O Diagrama do Banco de Dados apresenta a relação entre todas as tabelas contidas no Banco de Dados da aplicação. Na figura abaixo, é possível perceber as relações das tabelas:

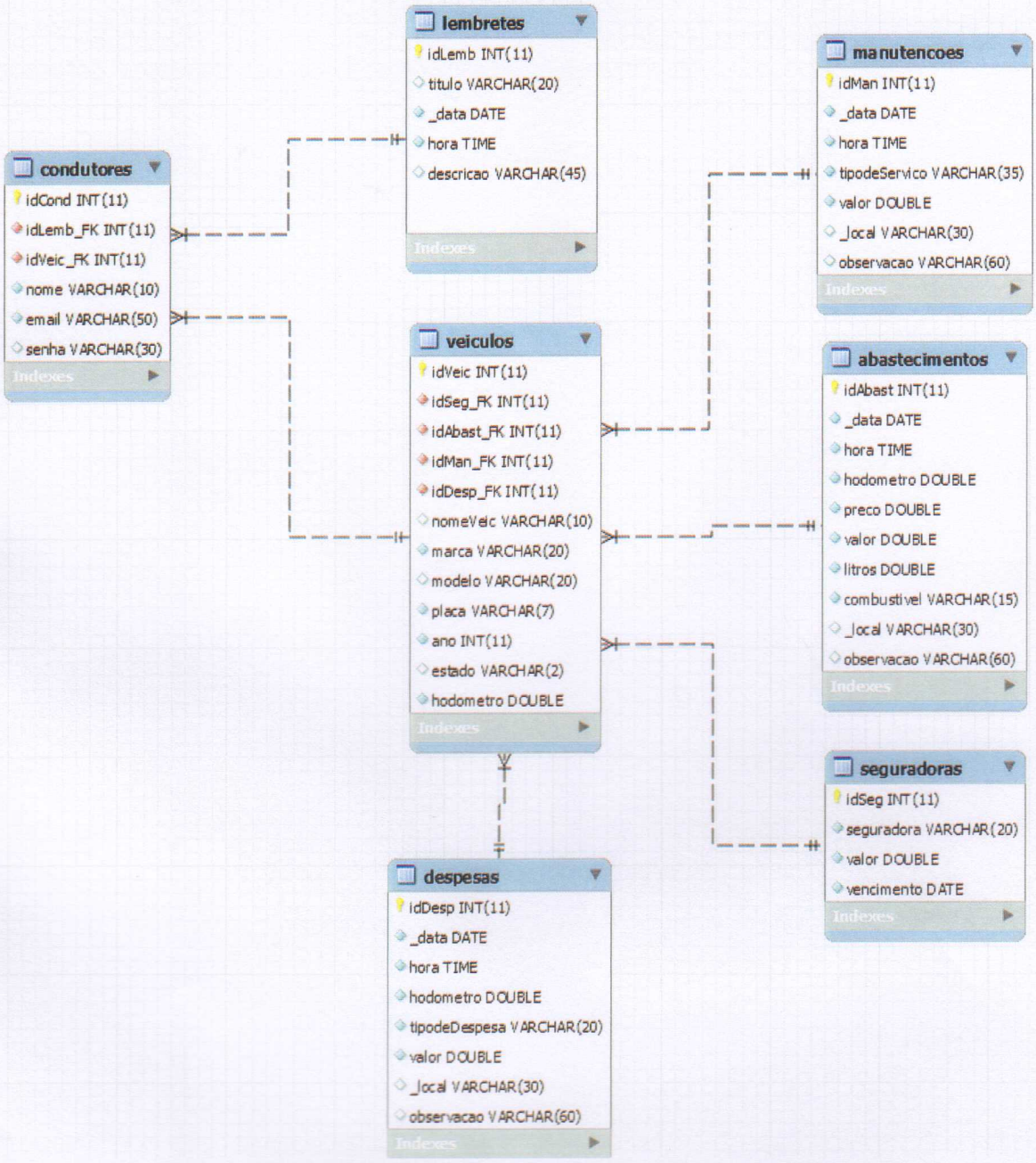

Figura 17 – Diagrama do Banco de Dados

2.3.4 Diagrama de Classes

O diagrama de classes é uma representação da estrutura e relações das classes de um objeto. Nele é indicado todas as classes, atributos e métodos do sistema.

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class UsuarioDAO {

        public boolean inserirUsuario(Usuario usuario) {
                try {
                        Connection conn = new ConnectionFactory
                           ().getConnection();

                        String queryInserir = "INSERT INTO
                           usuario (username, email, senha) "
                                        + "VALUES (?, ?, ?);";

                        PreparedStatement ps = conn.
                           prepareStatement(queryInserir);

                        ps.setString(1, usuario.getUsername());
                        ps.setString(2, usuario.getEmail());
                        ps.setString(3, usuario.getSenha());

                        ps.executeUpdate();

                        conn.close();

                        return true;

                } catch (SQLException | ClassNotFoundException
                   e) {
                        e.printStackTrace();
                        return false;
                }
        }

        public boolean atualizarUsuario(Usuario usuario) {
                try {
                        Connection conn = new ConnectionFactory
                           ().getConnection();

                        String queryAtualizar = "UPDATE usuario
                            SET Username=?, Email=?, Senha=?,
                           DataDeCriacao=? WHERE _id = ?";

                        PreparedStatement ps = conn.
                           prepareStatement(queryAtualizar);

                        ps.setString(1, usuario.getUsername());
                        ps.setString(2, usuario.getEmail());
                        ps.setString(3, usuario.getSenha());
                        ps.setString(4, usuario.
                           getDataDeCriacao());
                        ps.setInt(5, usuario.get_id());
```

```
                ps.executeUpdate();

                conn.close();
            } catch (SQLException | ClassNotFoundException
                e) {
                e.printStackTrace();
                return false;
            }

            return true;
    }

    public boolean excluirUsuario(Usuario usuario) {
            try {
                Connection conn = new ConnectionFactory
                    ().getConnection();

                String queryExcluir = "DELETE FROM
                    usuario where _id = ?";

                PreparedStatement ps = conn.
                    prepareStatement(queryExcluir);

                ps.setInt(1, usuario.get_id());

                ps.executeUpdate();

                conn.close();
            } catch (SQLException | ClassNotFoundException
                e) {
                e.printStackTrace();
                return false;
            }

            return true;
    }

    public ArrayList<Usuario> buscarTodosUsuarios() {
            ArrayList<Usuario> lista = new ArrayList<>();

            try {
                Connection conn = new ConnectionFactory
                    ().getConnection();

                String queryBuscar = "SELECT username,
                    email, senha, _id "
                            + "FROM usuario";

                PreparedStatement ps = conn.
                    prepareStatement(queryBuscar);


                ResultSet rs = ps.executeQuery();
```

```
                while (rs.next()) {
                        Usuario usuario = new Usuario()
                            ;

                        usuario.setUsername(rs.
                            getString(1));
                        usuario.setEmail(rs.getString
                            (2));
                        usuario.setSenha(rs.getString
                            (3));
                        usuario.set_id(rs.getInt(4));

                        lista.add(usuario);
                }

                conn.close();
        } catch (SQLException | ClassNotFoundException
            e) {
                e.printStackTrace();
                return null;
        }

        return lista;
}

public Usuario buscarUsuarioPorUsername(String username
    ){
        Usuario usuario = null;

        try (Connection conn = new ConnectionFactory().
            getConnection();){


                String queryBuscar = "SELECT _id, email
                    "
                                + "FROM usuario WHERE
                                    username = ?";

                PreparedStatement ps = conn.
                    prepareStatement(queryBuscar);

                ps.setString(1, username);

                ResultSet rs = ps.executeQuery();

                if (rs.next()) {
                        usuario = new Usuario();

                        usuario.set_id(Integer.parseInt
                            (rs.getString(1)));
                        usuario.setEmail(rs.getString
                            (2));
```

```
                return usuario;
            } else {
                return null;
            }

        } catch (SQLException | ClassNotFoundException
            e) {
            e.printStackTrace();
            return null;
        }
    }

    /**
     *
     * @param usuario
     * @return 0 caso ocorra erro; -1 se o nome de usu rio
        n o existir; 1 se o nome de usu rio
     * existir mas a senha n o coincidir; 2 caso o nome de
        usu rio existir e a senha coincidir.
     */
    public int checaLogin (String username, String senha) {
        try {
            Connection connUsername = new
                ConnectionFactory().getConnection();

            String queryVerificacaoUsername = "
                SELECT _id FROM usuario WHERE
                username = ?";

            PreparedStatement psUsername =
                connUsername.prepareStatement(
                queryVerificacaoUsername);
            psUsername.setString(1, username);

            ResultSet rsUsername = psUsername.
                executeQuery();

            if (rsUsername.next()) {
                Connection connSenha = new
                    ConnectionFactory().
                    getConnection();

                String queryVerificacaoSenha =
                    "SELECT _id FROM usuario
                    WHERE username = ? AND senha
                     = ?";
                PreparedStatement psSenha =
                    connSenha.prepareStatement(
                    queryVerificacaoSenha);

                psSenha.setString(1, username);
                psSenha.setString(2, senha);
```

```
                                ResultSet rsSenha = psSenha.
                                    executeQuery();

                                if(rsSenha.next()) {
                                        return 2;
                                } else {
                                        return 1;
                                }
                        } else {
                                return -1;
                        }
                } catch (SQLException | ClassNotFoundException
                    e) {
                        e.printStackTrace();
                        return 0;
                }
        }
}
```

- VeiculoDAO

Código 2.2 – VeiculoDAO

```
package com.weirdcode.autoappWS;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class VeiculoDAO {

        public boolean inserirVeiculo(Veiculo veiculo) {

                try (Connection conn = new ConnectionFactory().
                    getConnection();) {

                        String queryInserir = "INSERT INTO
                            veiculo("
                                        + "Placa, Marca_Veiculo
                                            , Modelo_Veiculo,
                                            Nome_Veiculo,
                                            Ano_Veiculo, "
                                        + "Usuario_FK,
                                            HoraCriacao) VALUES
                                            (?, ?, ?, ?, ?, ?,
                                            ?)";

                        PreparedStatement ps = conn.
                            prepareStatement(queryInserir);

                        ps.setString(1, veiculo.getPlaca());
```

```
                ps.setString(2, veiculo.
                    getMarca_veiculo());
                ps.setString(3, veiculo.
                    getModelo_veiculo());
                ps.setString(4, veiculo.getNome_veiculo
                    ());
                ps.setString(5, veiculo.getAno_veiculo
                    ());
                ps.setInt(6, veiculo.getUsuario_fk());
                ps.setString(7, veiculo.getHora_criacao
                    ());

                ps.executeUpdate();

                return true;

            } catch (ClassNotFoundException | SQLException
                e) {

                e.printStackTrace();
                return false;
            }
    }

    public boolean excluirVeiculo(int _id) {

            try (Connection conn = new ConnectionFactory().
                getConnection();) {

                String queryExcluir = "DELETE FROM
                    veiculo WHERE _id = ?";

                PreparedStatement ps = conn.
                    prepareStatement(queryExcluir);

                ps.setInt(1, _id);

                ps.executeUpdate();

                return true;

            } catch (ClassNotFoundException | SQLException
                e) {

                e.printStackTrace();
                return false;
            }
    }

    public boolean atualizarVeiculo(Veiculo veiculo) {

            try (Connection conn = new ConnectionFactory().
                getConnection();) {
```

```
                String queryAtualizar = "UPDATE veiculo
                    SET "
                            + "Placa=?,
                               Marca_Veiculo=?,
                               Modelo_Veiculo=?,
                               Nome_Veiculo=?,
                               Ano_Veiculo=?, "
                            + "Usuario_FK=? ,
                               HoraCriacao=? WHERE
                               _id = ?";

                PreparedStatement ps = conn.
                   prepareStatement(queryAtualizar);

                ps.setString(1, veiculo.getPlaca());
                ps.setString(2, veiculo.
                   getMarca_veiculo());
                ps.setString(3, veiculo.
                   getModelo_veiculo());
                ps.setString(4, veiculo.getNome_veiculo
                   ());
                ps.setString(5, veiculo.getAno_veiculo
                   ());
                ps.setInt(6, veiculo.getUsuario_fk());
                ps.setString(7, veiculo.getHora_criacao
                   ());
                ps.setInt(8, veiculo.get_id());

                ps.executeUpdate();

                return true;

            } catch (ClassNotFoundException | SQLException
               e) {

                e.printStackTrace();
                return false;
            }
    }

    public ArrayList<Veiculo> buscarTodosVeiculos(int _id)
       {
            ArrayList<Veiculo> lista = null;

            try (Connection conn = new ConnectionFactory().
               getConnection();) {

                String queryBuscar = "SELECT _id, Placa
                   , Marca_Veiculo, Modelo_Veiculo,
                   Nome_Veiculo, Ano_Veiculo, "
                            + "Usuario_FK,
                               HoraCriacao FROM
                               veiculo WHERE
                               usuario_fk = ?";
```

```
                PreparedStatement ps = conn.
                    prepareStatement(queryBuscar);
                ps.setInt(1, _id);

                ResultSet rs = ps.executeQuery();

                if (rs != null) {
                        lista = new ArrayList<>();

                        while (rs.next()) {
                                Veiculo veiculo = new
                                    Veiculo();

                                veiculo.set_id(rs.
                                    getInt(1));
                                veiculo.setPlaca(rs.
                                    getString(2));
                                veiculo.
                                    setMarca_veiculo(rs.
                                    getString(3));
                                veiculo.
                                    setModelo_veiculo(rs
                                    .getString(4));
                                veiculo.setNome_veiculo
                                    (rs.getString(5));
                                veiculo.setAno_veiculo(
                                    rs.getString(6));
                                veiculo.setUsuario_fk(
                                    rs.getInt(7));
                                veiculo.setHora_criacao
                                    (rs.getString(8));

                                lista.add(veiculo);
                        }
                } else {
                        return null;
                }

                return lista;

        } catch (SQLException | ClassNotFoundException
            e) {
                e.printStackTrace();
                return null;
        }
    }
}
```

A seguir temos os Beans utilizados

- Usuario

Código 2.3 – UsuarioBean

```
package com.weirdcode.autoappWS;

public class Usuario {
        private int _id;
        private String email;
        private String username;
        private String senha;
        private String dataDeCriacao;

        public Usuario() {
        }

        public Usuario(int _id, String email, String username,
           String senha) {
                super();
                this._id = _id;
                this.email = email;
                this.username = username;
                this.senha = senha;
        }

        public int get_id() {
                return _id;
        }

        public void set_id(int _id) {
                this._id = _id;
        }

        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
                this.email = email;
        }

        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

        public String getSenha() {
                return senha;
        }

        public void setSenha(String senha) {
                this.senha = senha;
        }
```

```
        public String getDataDeCriacao() {
                return dataDeCriacao;
        }

        public void setDataDeCriacao(String dataDeCriacao) {
                this.dataDeCriacao = dataDeCriacao;
        }
}
```

- Veiculo

Código 2.4 – Veiculo

```
package com.weirdcode.autoappWS;

public class Veiculo {
        private int _id;
        private String placa;
        private String marca_veiculo;
        private String modelo_veiculo;
        private String nome_veiculo;
        private String ano_veiculo;
        private int usuario_fk;
        private String hora_criacao;

        public Veiculo() {
                super();
        }

        public Veiculo(int _id, String placa, String
           marca_veiculo, String modelo_veiculo, String
           nome_veiculo,
                        String ano_veiculo, int usuario_fk,
                           String hora_criacao) {
                super();
                this._id = _id;
                this.placa = placa;
                this.marca_veiculo = marca_veiculo;
                this.modelo_veiculo = modelo_veiculo;
                this.nome_veiculo = nome_veiculo;
                this.ano_veiculo = ano_veiculo;
                this.usuario_fk = usuario_fk;
                this.hora_criacao = hora_criacao;
        }

        public int get_id() {
                return _id;
        }

        public void set_id(int _id) {
                this._id = _id;
        }
```

```
public String getPlaca() {
        return placa;
}

public void setPlaca(String placa) {
        this.placa = placa;
}

public String getMarca_veiculo() {
        return marca_veiculo;
}

public void setMarca_veiculo(String marca_veiculo) {
        this.marca_veiculo = marca_veiculo;
}

public String getModelo_veiculo() {
        return modelo_veiculo;
}

public void setModelo_veiculo(String modelo_veiculo) {
        this.modelo_veiculo = modelo_veiculo;
}

public String getNome_veiculo() {
        return nome_veiculo;
}

public void setNome_veiculo(String nome_veiculo) {
        this.nome_veiculo = nome_veiculo;
}

public String getAno_veiculo() {
        return ano_veiculo;
}

public void setAno_veiculo(String ano_veiculo) {
        this.ano_veiculo = ano_veiculo;
}

public int getUsuario_fk() {
        return usuario_fk;
}

public void setUsuario_fk(int usuario_fk) {
        this.usuario_fk = usuario_fk;
}

public String getHora_criacao() {
        return hora_criacao;
}

public void setHora_criacao(String hora_criacao) {
```

```
            this.hora_criacao = hora_criacao;
        }
}
```

2.5 Design

As imagens a seguir compreendem parte da totalidade da aplicação, e permitem notar como o design da aplicação se mostrará.

Figura 19 – Cadastro de usuário

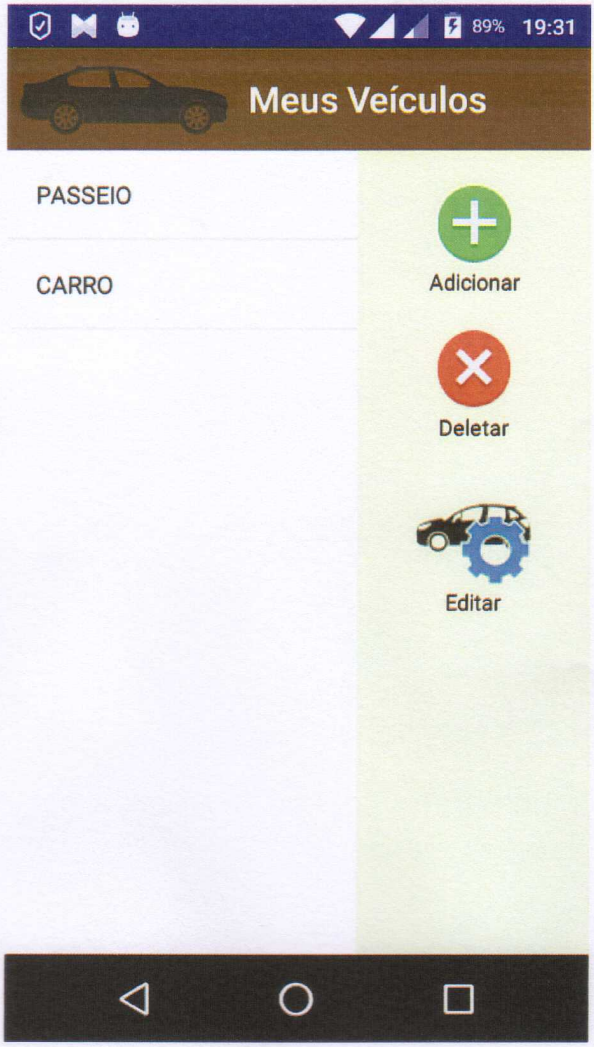

Figura 20 – "Meus Veículos"

3 CONCLUSÃO

A experiência de criar uma aplicação assemelha-se ao trabalho de um artesão. É necessário paciência, planejamento, inspiração e disciplina. A execução desse projeto exigiu um empenho constante do grupo. Foram muitas as vezes que alterações foram necessárias, demandando adaptação e resiliência.

Inicialmente embasamos nossas pretensões em pesquisas bibliográficas, mas logo foi necessário consultar o público alvo da aplicação, visto a necessidade de mais entendimento e profundidade. Para isso serviu-se a pesquisa de campo. Após isso, a modelagem conceitual e física foram facilitadas.

O desenvolvimento se tornou uma aprendizagem contínua. A implementação das telas, bem como o banco de dados, requeriu uma superação constante de obstáculos e desafios, seja pela apresentação de erros ou incapacidade técnica. No entanto, o projeto findou-se exitoso, e adquirimos uma experiência valiosa e memorável que permeará toda nossa vida, tanto no campo profissional quanto profissional.

REFERÊNCIAS

FONSECA, J. J. S. da. *Metodologia da Pesquisa Científica*. Fortaleza, 2002. Citado na página 11.

GERHARDT, T. E.; SILVEIRA, D. T. *Método de Pesquisa*. 1. ed. Porto Alegre: Editora da UFRGS, 2009. Citado na página 10.

GIL, A. C. *Como Elaborar Projetos de Pesquisa*. 4. ed. São Paulo: Atlas, 2002. ISBN 8522431698. Citado 2 vezes nas páginas 10 e 11.

GRECO, E. *Comprou um carro zero quilômetro? Veja os gastos para manter um automóvel*. Minas Gerais: [s.n.], 2015. Disponível em: <http://estadodeminas.vrum.com.br/app/noticia/noticias/2015/10/23/interna_noticias,51556/comprou-um-carro-0km-veja-os-gastos-para-manter-um-automovel.shtml>. Acesso em: 23/09/2016. Citado na página 7.

IBGE. *Pesquisa Nacional por Amostra de Domicílios (Pnad)*. 2015. Citado na página 7.

ORGANIZAÇÃO DAS NAÇÕES UNIDAS. *66/260. Improving global road safety*. 2012. Disponível em: <https://documents-dds-ny.un.org/doc/UNDOC/GEN/N11/474/42/PDF/N1147442.pdf?OpenElement>. Citado na página 7.