

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

**SYSCONTROL – PROTÓTIPO DE APLICATIVO PESSOAL
DE MANUTENÇÃO DE AUTOMÓVEIS PARA DISPOSITIVOS
MÓVEIS**

JULIANO FABIAN

BLUMENAU
2015

2015/2-10

JULIANO FABIAN

**SYSCONTROL – PROTÓTIPO DE APLICATIVO PESSOAL
DE MANUTENÇÃO DE AUTOMÓVEIS PARA DISPOSITIVOS
MÓVEIS**

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Francisco Adell Péricas, Mestre – Orientador

**BLUMENAU
2015**

2015/2-10

**SYSCONTROL – PROTÓTIPO DE APLICATIVO PESSOAL
DE MANUTENÇÃO DE AUTOMÓVEIS PARA DISPOSITIVOS
MÓVEIS**

Por

JULIANO FABIAN

Trabalho de Conclusão de Curso aprovado
para obtenção dos créditos na disciplina de
Trabalho de Conclusão de Curso II pela banca
examinadora formada por:

Presidente: _____
Prof. Francisco Adell Péricas, Mestre - Orientador, FURB

Membro: _____
Prof. Everaldo Artur Grahl, Mestre - FURB

Membro: _____
Prof. Luciana Pereira de Araújo, Mestre – FURB

Blumenau, 10 de dezembro de 2015.

Dedico este trabalho a todos os amigos,
especialmente aqueles que me ajudaram
diretamente na realização deste.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família, em especial meus pais Rui J. Fabian e Marli P. Fabian por sempre me apoiarem.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, professor Francisco Adell Péricas, pelo auxílio e sugestões na elaboração da proposta deste trabalho.

Aos professores do Departamento de Sistemas e Computação da Universidade Regional de Blumenau por suas contribuições durante os semestres letivos.

Se não puder se destacar pelo talento vença
pelo esforço.

Dave Weinbaum

RESUMO

Este trabalho apresenta um aplicativo para plataforma *Android* que permite o usuário ter um controle das manutenções efetuadas no veículo, mantendo um histórico completo do veículo. Com a aplicação é possível acompanhar e registrar a quilometragem, consumo de combustível, manutenções, serviços e despesas do veículo buscando garantir que o veículo esteja em plenas condições de uso com redução de custos e manutenções planejadas. Observada a deficiência em manter as informações de manutenção em dia, verificou-se a necessidade de disponibilizar este controle através de um sistema para dispositivos móveis. Com as informações necessárias para manter o veículo em dia, o aplicativo permite o usuário programar com antecedência as manutenções de acordo com os tempos corretos, garantindo uma maior durabilidade do veículo. A aplicação foi construída na linguagem JAVA utilizando ambiente Android Studio e banco de dados SQLite, sendo compatível para dispositivos móveis com Android 4.0 ou superior. O resultado obtido foi uma aplicação para gerenciar manutenções de automóveis que permite registrar as informações do automóvel e gerar relatórios.

Palavras chaves: Dispositivos móveis. Manutenção de veículos. Android.

ABSTRACT

This paper presents an application for the Android platform that allows the user to have control of the maintenance performed on the vehicle, keeping a complete history of the vehicle. With application can track and record mileage, fuel consumption, maintenance, services and vehicle expenses seeking to ensure that the vehicle is in full working order to reduce costs and planned maintenance. Found deficiency in maintaining the day maintenance information, there is a need to provide this control via a system for mobile devices. With the information necessary to keep the vehicle on time, the application allows the user to program in advance the maintenance according to the correct time, ensuring greater durability of the vehicle. The application was built in Java language using Android Studio environment and SQLite database and is compatible for mobile devices with Android 4.0 or higher. The result was an application to manage car maintenance that allows you to register the car's information and generate reports.

Key-Words: Mobile devices. Support of vehicles. Android.

LISTA DE FIGURAS

Figura 1 - Usuários de smartphone	18
Figura 2 - Usuários de tablets Android e IOS.....	18
Figura 3 - Arquivo Android Manifest .xml	21
Figura 4 - Ciclo de vida activity	23
Figura 5 - Fluxo de chamada de métodos de services.....	25
Figura 6 - Tela de opções menu principal do aplicativo Carrorama	27
Figura 7 - Diagrama de atividades.....	29
Figura 8 - Diagrama de caso de uso	31
Figura 9 - Modelo de entidade e relacionamento	32
Figura 10 - Tela de desenvolvimento Android Studio	34
Figura 11 - Estrutura de diretórios do projeto	35
Figura 12 - Classe que faz a criação das estruturas do banco de dados	36
Figura 13 - Tela inicial emulador Genymotion	37
Figura 14 - Tela <i>login</i>	38
Figura 15 - Tela de edição de usuário.....	39

Figura 16 - Tela principal do aplicativo	40
Figura 17 - Tela cadastro de automóveis.....	41
Figura 18 - Tela de edição de automóveis.....	42
Figura 19 - Tela de edição de automóveis.....	43
Figura 20 - Selecionando automóvel.....	44
Figura 21 - Automóvel padrão selecionado.....	45
Figura 22 - Cadastro de abastecimento.....	46
Figura 23 - Tela de edição de abastecimento.....	47
Figura 24 - Tela de edição de abastecimento.....	48
Figura 25 - Tela de cadastro de estacionamento.....	49
Figura 26 - Tela de edição de estacionamento.....	50
Figura 27 - Tela de manutenções.....	51
Figura 28 - Mensagem de notificação troca de óleo.....	52
Figura 29 - Tela de relatório do aplicativo.....	53
Figura 30 - Tela de abastecimentos.....	54
Figura 31 - Tela de relatório de manutenções.....	55

LISTA DE QUADROS

Quadro 1 - Requisitos Funcionais	30
Quadro 2 - Requisitos não funcionais	30
Quadro 3 - Comparativo entre os aplicativos correlatos.....	56
Quadro 4 - Descrição dos casos de uso.....	61

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface

IDE - Integrated Development Environment

UML - Unified Modeling Language

XML - Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO	14
1.1 OBJETIVOS.....	15
1.2 ESTRUTURA.....	15
2 FUNDAMENTAÇÃO TEÓRICA	16
2.1 DISPOSITIVOS MÓVEIS	16
2.2 ANDROID.....	16
2.3 ANDROID STUDIO	19
2.3.1 PROCESSO DO CICLO DE VIDA.....	20
2.3.2 O ARQUIVO ANDROID MANIFEST.XML	21
2.3.3 COMPONENTE ACTIVITY DA PLATAFORMA ANDROID.....	22
2.4 COMPONENTE SERVICE DA PLATAFORMA ANDROID.....	24
2.4.1 CICLO DE VIDA DE UM SERVIÇO	25
2.5 TRABALHOS CORRELATOS	25
3 DESENVOLVIMENTO DO SOFTWARE SYSCONTROL	28
3.1 LEVANTAMENTO DE INFORMAÇÕES	28
3.1.1 DIAGRAMA DE ATIVIDADES.....	29
3.2 ESPECIFICAÇÃO	29
3.2.2 DIAGRAMA DE CASO DE USO	31
3.2.3 MODELO DE ENTIDADE RELACIONAMENTO	32
3.3 IMPLEMENTAÇÃO	33
3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS	33
3.3.2 ANDROID STUDIO	34
3.3.3 SQLITE DATABASE	36
3.3.4 EMULADOR GENYMOTION	37
3.3.5 OPERACIONALIDADE DA IMPLEMENTAÇÃO	37
3.4 RESULTADOS E DISCUSSÕES.....	55
REFERÊNCIAS	59
APÊNDICE A – DESCRIÇÃO DOS CASOS DE USO	61
APÊNDICE B – DICIONÁRIO DE DADOS.....	64

1. INTRODUÇÃO

Atualmente pesquisas revelam que o mercado de dispositivos móveis redefiniu o comportamento do usuário. Dados do Ibope encomendados pela Qualcomm no Brasil mostram que cerca de 30% dos usuários de *smartphones* costumam verificar seus *gadgets* em um intervalo máximo de cinco minutos (DAL PIAZ, 2013).

Segundo Dal Piaz (2013), a expectativa é de que nos próximos anos mais de 7 bilhões de aparelhos móveis estejam sendo usados no mundo. A tendência é que mais de 24 bilhões de equipamentos estejam conectados em redes 3G e 4G, incluindo periféricos, acessórios, eletrodomésticos, entre outros. Isso tudo deve ajudar a redefinir a relação usuário-máquina, criando uma rede tecnológica.

Tarefas que antes precisavam ser feitas em computadores e *notebooks* agora podem ser realizadas em aparelhos que cabem no bolso. Esses dispositivos foram agregando facilidades que permitem aos usuários inúmeras possibilidades de utilização e com isso estão ocupando cada vez mais espaço na vida das pessoas. O uso de aplicativos para gerenciar atividades relevantes do dia a dia tem se tornado cada vez mais popular.

O mercado corporativo também está crescendo muito e diversas empresas estão buscando incorporar aplicações móveis a seu dia a dia para agilizar seus negócios e integrar as aplicações móveis com seus sistemas de *back-end*. Empresas obviamente visam lucrar espaço em um mundo onde a palavra “mobilidade” está cada vez mais conhecida. (LECHETA, 2009).

Visto as inúmeras possibilidades de aplicações que agilizam o gerenciamento de tarefas, a tendência é que estes recursos ocupem cada vez mais espaço na vida das pessoas. Com a ajuda da tecnologia, um dos gastos que podem ser controlados por meio de aplicação *mobile* é a manutenção de automóveis, alimentando o sistema com informações de combustível, vencimentos e auxiliando o proprietário a ter uma manutenção preventiva e não corretiva do automóvel.

A utilização de uma aplicação *mobile* auxiliará o proprietário a se programar nas manutenções de peças que com o uso provocam desgastes, necessitando de regulagens, melhorando a conservação do automóvel e também evitando o retorno à oficina por causa de quebra e imprevistos. Os resultados a serem alcançados com este tipo de aplicação também se aplicam a gestão de frotas de modo a reduzir os custos e desperdícios, além de aumentar os lucros.

No caso de frotas, a aplicação visa suprir uma necessidade muito comum em empresas, que é a falta de informação dos automóveis. O bom desempenho e a disponibilidade de automóveis de frota dependem da manutenção. O objetivo final é utilizar essas informações para gerir automóveis, garantindo boas condições de manutenção, menor desgaste de peças e maior longevidade de frotas.

1.1 OBJETIVOS

O objetivo deste trabalho é o desenvolvimento de uma aplicação *mobile* de gerenciamento das manutenções veiculares permitindo que o usuário controle as informações de gastos, abastecimento e despesas dos automóveis.

Os objetivos específicos do trabalho são:

- a) permitir controlar o consumo de combustíveis e lubrificantes;
- b) permitir controlar balanceamento e troca de pneus;
- c) cadastrar novas rotinas de manutenção e despesas;
- d) alertar manutenções integrando o aplicativo à agenda do sistema *mobile*.

1.2 ESTRUTURA

No primeiro capítulo tem-se a introdução ao tema principal deste trabalho com a apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica pesquisada sobre gestão de frotas, dispositivos móveis, Android e trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do sistema iniciando-se com o levantamento de informações, tendo na sequência especificação, implementação e por fim resultados e discussão.

No quarto capítulo tem-se as conclusões deste trabalho bem como se apresentam sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 DISPOSITIVOS MÓVEIS

Segundo Johnson (2007), com o passar dos anos, a evolução da computação fez com que os computadores diminuíssem de tamanho e se tornassem fáceis de utilizar em qualquer lugar. Hoje, é possível acessar qualquer dado a partir de dispositivos que estão com as pessoas por todos os lugares: os dispositivos móveis. A computação móvel permite que usuários tenham acesso a serviços independentemente de sua localização. Isso requer suporte à mobilidade e eventualmente à existência de infraestrutura de comunicação sem fio.

A computação móvel cresceu bastante devido a alguns fatores, tais como a necessidade de se obter melhores resultados nos negócios, interagindo em qualquer lugar ou em movimento necessitando de acesso à informação. O preço dos dispositivos está diminuindo enquanto esses estão ficando cada vez mais sofisticados. Além disso, a melhora na infraestrutura de comunicação entre dispositivos, também impulsiona a área (POSSER, 2006).

A proliferação da computação móvel se deve a diversos fatores:

- e) a vantagem e até a necessidade da informação à disposição em qualquer lugar;
- f) a redução dos preços dos dispositivos móveis;
- g) o crescente mercado da telefonia móvel;
- h) conveniência de transporte e utilização.

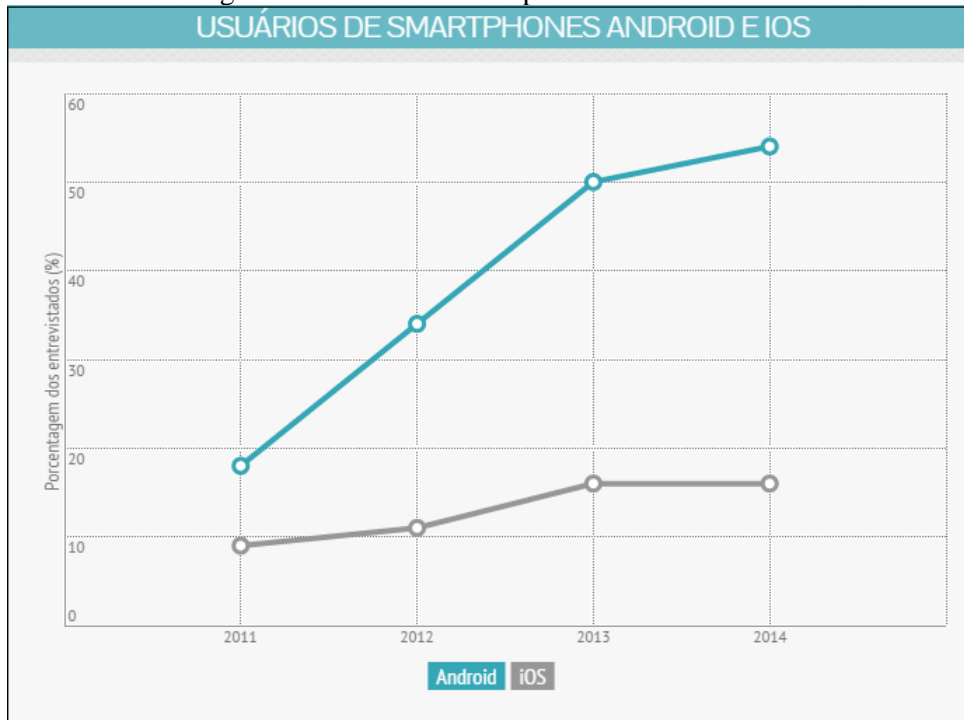
2.2 ANDROID

Android é uma plataforma de software para dispositivos móveis que inclui um sistema operacional, uma camada intermediária *middleware* e aplicativos chave. O Android é um sistema baseado em Linux e faz parte da *Open Handset Alliance* (conjunto de várias empresas que têm como objetivo desenvolver padrões abertos para dispositivos móveis). Existe uma grande comunidade de programadores que faz aplicações para Android, aumentando assim o número de funcionalidades de um dispositivo que contenha este sistema operacional (ANDROID DEVELOPERS, 2011).

Desde 2008 que o Android é *open source*, pois a Google publicou o seu código-fonte no âmbito de Apache License. O Android atualmente é um sistema que pode ser usado em vários dispositivos, desde *smartphones*, *tablet PC's* e *netbooks*, juntamente com toda a biblioteca de desenvolvimento. Ele é considerado uma plataforma móvel completa, livre e aberta, apresentando características incorporadas de outros sistemas, além de inúmeros conceitos inovadores para o segmento móvel como a utilização de tela sensível ao toque para manipulação de objetos virtuais e a utilização de um teclado virtual (ANDROID DEVELOPERS, 2011).

Segundo Amador (2015), a batalha entre os sistemas operacionais *mobile*, apresenta um domínio cada vez maior da plataforma da Google. Em 2014 foi realizada uma pesquisa com 40 mil internautas com idade de 16 a 64 anos de 32 países, os participantes responderam questões relacionadas ao uso dispositivos móveis. Com a pesquisa revelou-se entre os entrevistados, que 54% utilizam dispositivos móveis com sistema Android, enquanto menos de 16% tem aparelhos com iOS e a diferença só cresceu nos últimos anos (Figura 1).

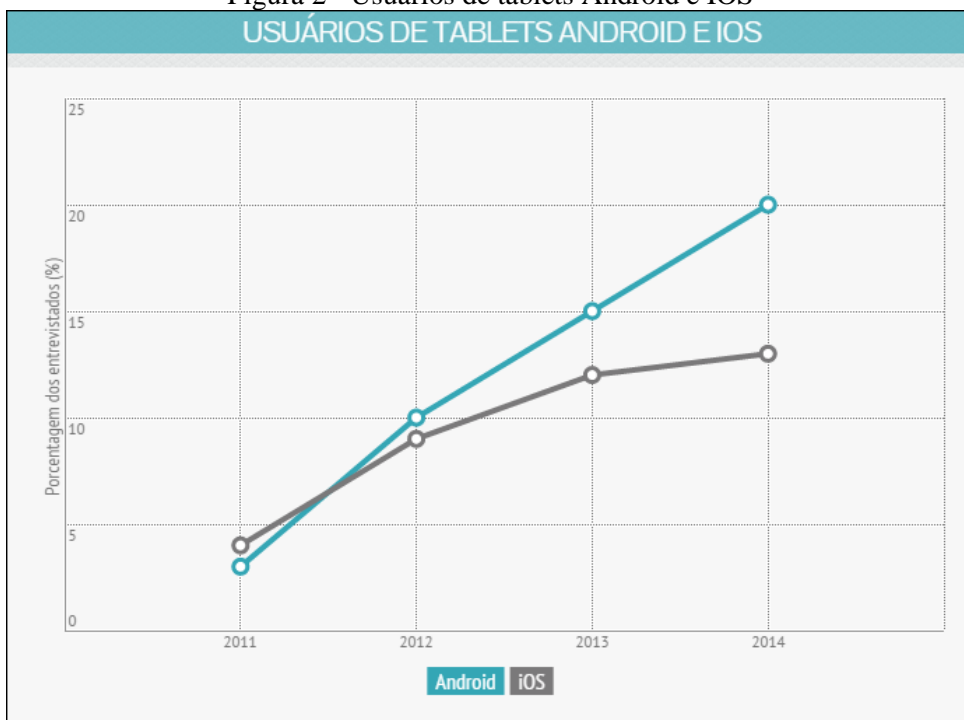
Figura 1- Usuários de smartphones Android e IOS



Fonte: Amador (2015).

O domínio Android também chegou aos *tablets*, onde 20 % dos entrevistados usam o sistema da Google, enquanto 13% são usuários da plataforma da Apple. Os números mostram a virada em relação ao ano de 2011 quando a pesquisa começou a ser feita (Figura 2).

Figura 2 - Usuários de tablets Android e IOS



Fonte: Amador (2015).

A Open Handset Alliance (2007) oferece a uma vasta gama de bibliotecas e ferramentas úteis que podem ser usadas para construir aplicações ricas. Por exemplo, permite aos desenvolvedores Android obter a localização do dispositivo e permite que os dispositivos se comuniquem uns com os outros com aplicações sociais *peer-to-peer*. Além disso, o Android inclui um conjunto completo de ferramentas que foram construídos a partir do zero junto à plataforma de fornecimento de desenvolvedores com alta produtividade e conhecimento sobre suas aplicações.

2.3 ANDROID STUDIO

O Android Studio é a nova *Integrated Development Environment* (IDE) da Google, concebido para o desenvolvimento de aplicativos para celulares, *tablets*, tv, *google wear* e *glass*, fazendo do Android uma plataforma promissora e com alta demanda de mercado. O programa oferece uma ferramenta para lidar com as mais variadas aplicações criadas para o sistema Android. As funções do software incluem a edição de códigos, recursos para *design* de interface de usuário e análise de performance. (ANDROID DEVELOPERS, 2011).

Segundo Avram (2014), em 2103 a Google anunciou a primeira versão do Android Studio baseado no *IntelliJ Community Edition*, feito para facilitar a vida de quem desenvolve aplicativos para plataforma móvel. No entanto, a ferramenta oficial de desenvolvimento para Android ainda apresentava falhas não sendo recomendável na época para o uso profissional.

Após as correções e uma série de melhorias, em dezembro de 2014 a Google disponibilizou a versão 1.0 do Android Studio, considerada estável e passando a ser chamada de IDE oficial da plataforma Android. A seguir são apresentadas as principais funcionalidades da versão 1.0:

- a) um instalador que configura o Android Studio e um ambiente de desenvolvimento com um conjunto de *templates*;
- b) todas as ferramentas de edição de código conforme previsto pelo IntelliJ IDEA;
- c) *dynamic Layout Preview*, que permite os usuários vejam e editem via arrastar e soltar (*drag/drop*) como o aplicativo móvel aparece em vários dispositivos e em todas as versões da *Application Programming Interface* (APIs);
- d) monitor de performance de memória;
- e) sistema de construção com base no Gradle que está integrado com o Android Studio, mas que não é afetado pelas suas atualizações;

- f) integração com *Google Cloud Platform*;
- g) ferramentas para controlar desempenho, usabilidade e compatibilidade de versões;
- h) ProGuard e capacidade de assinar aplicativos.

2.3.1 PROCESSO DO CICLO DE VIDA

Os processos de aplicações que executam no Android são mantidos em memória pelo tempo máximo possível. No entanto, eventualmente é necessário remover processos antigos quando o existe pouca memória disponível. A decisão de qual processo é encerrado depende exclusivamente do estado da interação do processo com o usuário, o sistema vai matar processos menos importantes antes de matar processos mais importantes, na tentativa de liberar memória. A seguir são listados os tipos de processos em ordem de importância. (OPEN HANDSET ALLIANCE, 2007).

- a) `Foreground activity` – a atividade que está sendo rodada naquele instante é considerada a mais importante. Seus processos somente serão mortos como último recurso, se ele usar mais memória do que há disponível no dispositivo. Geralmente nesse momento o dispositivo já chegou ao estado de uso de paginação, então é requerido matar esse processo que será necessário para manter a resposta da interface;
- b) `Visible activity` – uma atividade que está sempre visível mas o usuário não o têm na frente, como uma atividade que fica para atrás de um diálogo em janela flutuante, por exemplo, é considerada extremamente importante e não será morta a não ser que seja requerido para que a atividade de frente possa continuar rodando;
- c) `Background activity` – uma atividade que não é visível ao usuário e que está em modo `onPause()`. Não é crítica e o sistema poderá, de maneira bastante segura, matar seu processo na memória para que possa ser usada por outros processos. Se esse processo precisar ser morto, quando o usuário navegar de volta a ele (fazendo com que ele se torne visível novamente na tela), o método `onCreate(Bundle)` dele será chamado com o `savedInstanceState` suprido pelo `onSaveInstanceState(Bundle)` para que possa ser reiniciado no mesmo estado que foi deixado. Processo vazio é um processo que não tem nenhuma atividade ou outros componentes de aplicação rodando nele (como um `Service` ou `BroadcastReceiver`). Eles são rapidamente mortos pelo sistema assim que é necessário mais memória. Então, quaisquer operações de fundo que precisam ser

executadas fora da atividade devem ser executadas no contexto de uma atividade BroadcastReceiver ou Service para assegurar que o sistema saiba que é necessário mantê-la no ar para que seu processo seja executado.

2.3.2 O ARQUIVO ANDROID MANIFEST.XML

O arquivo `AndroidManifest.xml` é considerado o arquivo principal do projeto, é onde ficam todas as configurações da aplicação. Esse arquivo fica na pasta raiz do projeto e é obrigatório para todas as aplicações Android contendo todas as configurações necessárias para executar a aplicação, como o nome do pacote utilizado, permissões, versão mínima da *Application Programming Interface* (APIs) Android. O `AndroidManifest.xml` também descreve os componentes Activities, Services que fazem parte da aplicação, possibilitando que o sistema Android seja capaz de identifica-los e determinar quando serão executados. Na Figura 3, pode se ver um exemplo de arquivo `AndroidManifest.xml`.

Figura 3 - Arquivo `AndroidManifest.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="br.com.exemplo.oimundo"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".OiMundo"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>

</application>
</manifest>
```

Fonte: Macedo (2011).

Segundo Macedo (2011), dentro da *tag* `<manifest>` é necessário declarar o pacote principal do projeto, utilizando a *tag* `<package>`. É obrigatório que cada *Activity* do projeto esteja declarada no arquivo `AndroidManifest.xml`, caso contrário não é possível utilizá-la. Para declarar a *Activity* é utilizada a *tag* `<activity>`, que recebe o nome da classe e é sempre relativa ao pacote principal.

O projeto pode conter apenas uma, várias ou nenhuma *Activity*, mas o importante é que se for necessário exibir um ícone na tela principal do Android para que a aplicação possa ser iniciada pelo usuário, é necessário pelo menos uma *Activity* e esta deve ser configurada como sendo o ponto de partida da aplicação.

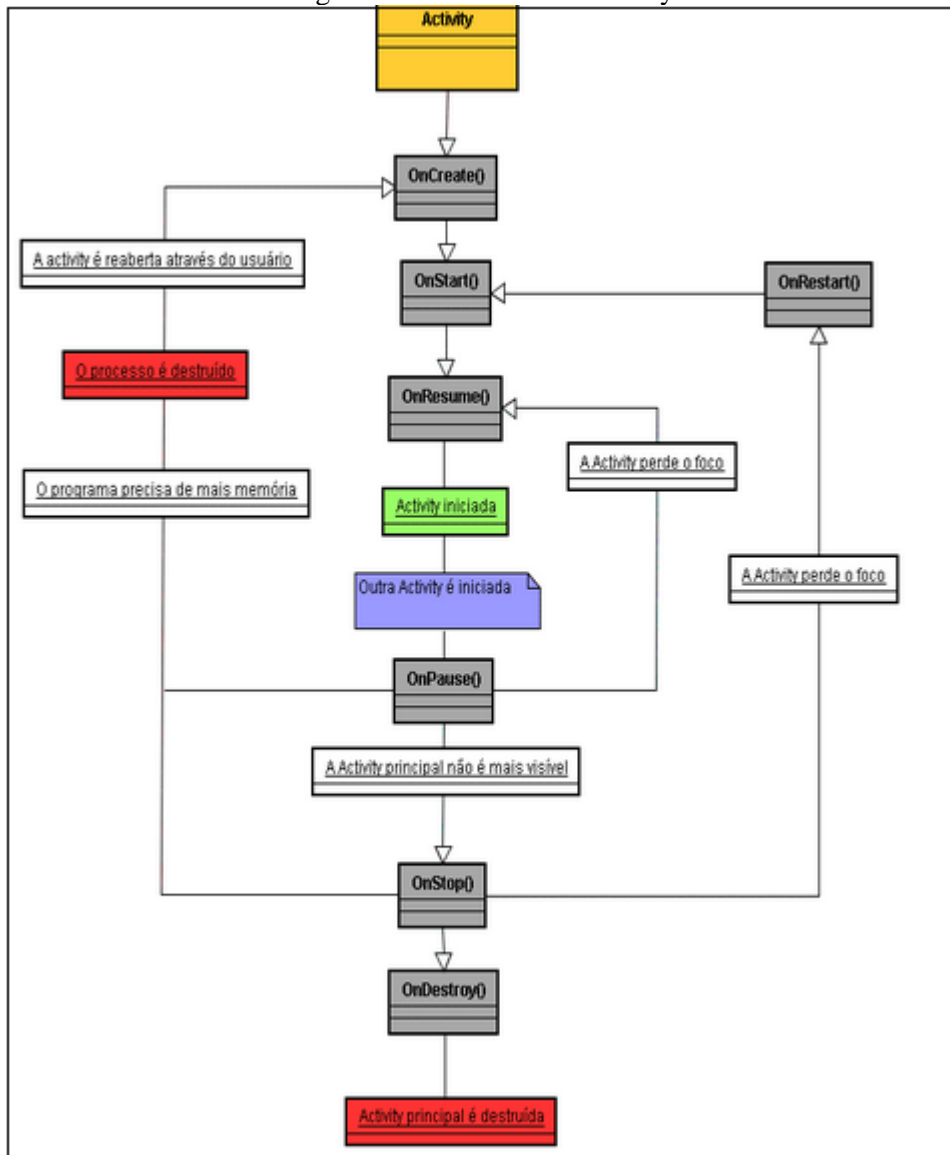
2.3.3 COMPONENTE ACTIVITY DA PLATAFORMA ANDROID

Uma *Activity* é um conceito de interface ao usuário. Ela geralmente representa uma tela da aplicação e contém um ou mais componentes gráficos associados, embora não seja obrigatório. Na maioria das vezes uma *Activity* é apresentada em tela cheia. Ela pode ser também uma janela flutuante ou uma parte integrada de outra *Activity* (HASHIMI E KOMATINENI, 2009, p.24).

Segundo Android Developers (2011), as *Activities* são gerenciadas através de um sistema de pilha de *Activities* (*activity stack*). Cada nova *Activity* disparada será apresentada ao usuário e colocada no topo da pilha, ficando abaixo dela a *Activity* que estava em execução anteriormente. Quando uma *Activity* conclui sua execução a *Activity* abaixo dela na pilha volta para o topo e é apresentada ao usuário.

Sempre que uma *Activity* é inicializada pela primeira vez, o primeiro método a ser executado por ela é o `onCreate()`. Geralmente é neste método em que é definido o *layout* que será apresentado ao usuário, bem como a associação dos componentes gráficos que este *layout* irá conter com variáveis da *Activity*. Na sequência é executado o método `onStart()` apresentando a *Activity* ao usuário. O método `onResume()`, executado logo em sequência, permite a interação do usuário com a *Activity*. Após a execução destes três métodos, uma *Activity* entra no estado de execução. Na Figura 4, segue um diagrama explicando o ciclo de vida de *Activity*.

Figura 4 - Ciclo de vida Activity



Fonte: Silva (2014).

- OnCreate() - É a primeira função a ser executada em uma Activity. Geralmente é a responsável por carregar os layouts XML e outras operações de inicialização. É executada apenas uma vez;
- OnStart() - É chamada imediatamente após a onCreate() e também quando uma Activity que estava em background volta a ter foco;
- OnResume() - Assim como a onStart(), é chamada na inicialização da Activity e também quando uma Activity volta a ter foco;
- OnStart() só é chamada quando a Activity não está mais visível e volta a ter o foco, a onResume() é chamada nas “retomadas de foco”;

- e) `OnPause ()` - É a primeira função a ser invocada quando a *Activity* perde o foco (isso ocorre quando uma nova *Activity* é iniciada);
- f) `OnStop ()` - Só é chamada quando a *Activity* fica completamente encoberta por outra *Activity*;
- g) `OnDestroy ()` - A última função a ser executada. Depois dela a *Activity* é considerada “morta”, ou seja, não pode mais ser relançada, se o usuário voltar a requisitar essa *Activity*, um novo objeto será construído;
- h) `OnRestart ()` - Chamada imediatamente antes da `onStart()`, quando uma *Activity* volta a ter o foco depois de estar em *background*;

2.4 COMPONENTE SERVICE DA PLATAFORMA ANDROID

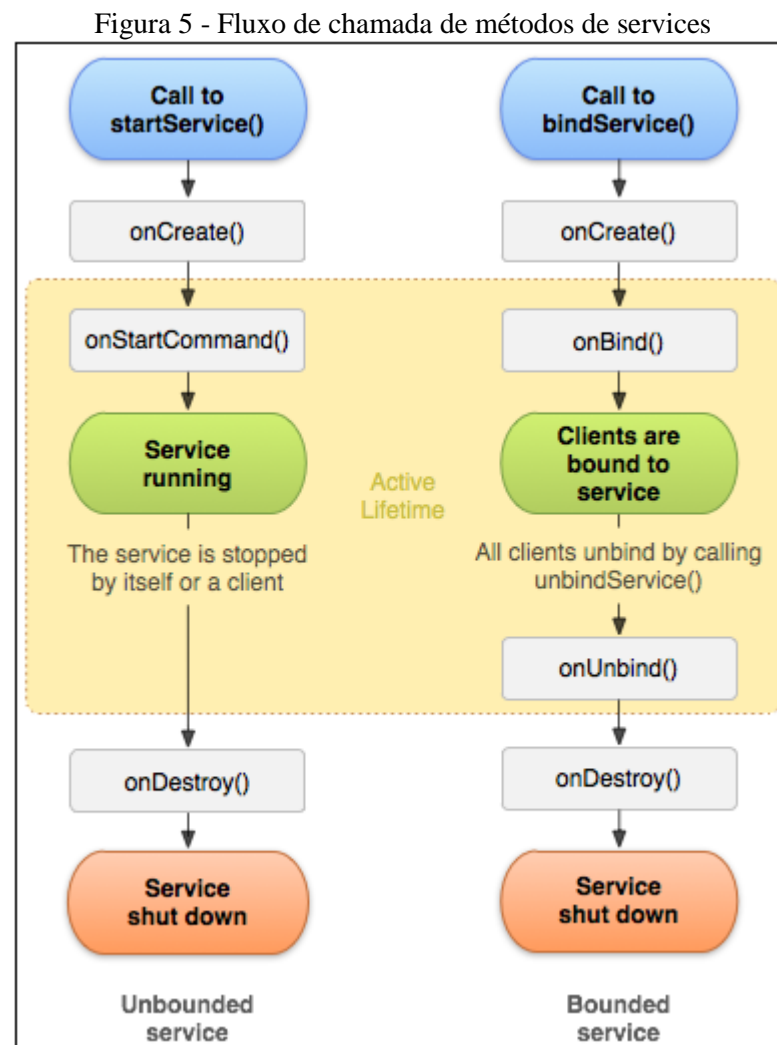
Segundo Android Developers (2011), *services* são componentes fundamentais em uma aplicação, eles podem executar operações de longa execução em segundo plano e não fornecem uma interface de usuário. Outro componente do aplicativo pode iniciar um *service*, que continuará a ser executado em segundo plano, mesmo se o usuário alternar para outro aplicativo. Além disso, um componente pode se ligar a um *service* para interagir com ele e até mesmo realizar a comunicação entre processos.

Segundo Silveira (2015) os *services* são geralmente utilizados para realizar tarefas de sincronização, podendo ter sua execução agendada e não dependendo de nenhuma ação do usuário, por este motivo são conhecidos como componentes de *background*. Existem dois tipos de serviços:

- a) `Started (unbounded)`: são serviços iniciados através de outros componentes com o método `startService()`, após o início, o serviço pode continuar sendo executado indefinidamente.
- b) `Bounded`: são serviços iniciados através do método `bindService()`. Os `Bound Services` interagem com os outros componentes através de uma interface cliente-servidor, que pode ocorrer entre diferentes processos. Um `Bound Service` é executado enquanto possuir requisições a serem tratadas.

2.4.1 CICLO DE VIDA DE UM SERVIÇO

O ciclo de vida de um serviço é análogo ao de uma *Activity*, porém mais simples. Na Figura 5 pode se ver um exemplo de fluo de chamadas de métodos para ambos tipos de serviço:



Fonte: Silveira (2015).

2.5 TRABALHOS CORRELATOS

Pode-se citar como trabalhos correlatos as monografias de Predon (1993) e Hoffmann (2010) para conclusão dos cursos Ciência da Computação e Sistemas de Informação na Universidade Regional de Blumenau.

No trabalho de Pedron (1993) foi proposto o desenvolvimento de uma aplicação *desktop* para plataforma *Windows* de controle de frota de veículos para prefeitura municipal

de Timbó. O sistema trata principalmente a parte de controle de autorização de manutenções e abastecimentos, mas não gera notificação das manutenções previstas.

O trabalho de Hoffmann (2010) consiste na gestão de frotas da empresa Unimed Blumenau, buscando manter a frota em boas condições e controlando os gastos com combustível, que é um fator preocupante de todas as empresas que possuem frota. O sistema é *desktop* para plataforma *Windows* e trata principalmente a reserva de veículos da frota, possibilitando uma melhor visualização da movimentação de veículos, mas não faz a parte de emissão de relatório dos abastecimentos.

Pode-se citar também como sistema correlato o aplicativo desenvolvido pela empresa Going2 Mobile, uma *startup* altamente capacitada a criar, desenvolver e gerenciar aplicativos móveis. O Carrorama propõe controlar tudo que é feito em um automóvel como: abastecimento, manutenções e despesas com financiamento e seguro. O aplicativo emite uma notificação alertando quando algum item de controle estiver para vencer, auxiliando na manutenção e evitando maiores prejuízos financeiros. Tem como diferencial gerar relatórios de desempenho de abastecimentos exibindo a média e a possibilidade de gerar uma planilha do EXCEL para impressão ou consulta (CARRORAMA, 2013). A Figura 6 mostra a tela de opções do menu principal. Através da opção manutenção o aplicativo disponibiliza para o usuário os tipos de controle de manutenção ou então permite adicionar um novo controle.

Figura 6 - Tela de opções menu principal do aplicativo Carrorama



Fonte: Carrorama (2013).

3 DESENVOLVIMENTO DO SOFTWARE SYSCONTROL

Neste capítulo estão descritas as particularidades técnicas do aplicativo tais como a descrição do mesmo, a especificação dos requisitos funcionais e não funcionais, os diagramas de atividades, diagrama de casos de uso e suas descrições. Na sequência apresenta-se a implementação com as técnicas e ferramentas utilizadas e a operacionalidade do sistema, encerrando-se com os resultados e discussões.

3.1 LEVANTAMENTO DE INFORMAÇÕES

O Syscontrol é uma aplicação na plataforma Android para gerenciar as manutenções de automóveis, fazendo com que através da aplicação o usuário tenha uma experiência agradável, intuitiva, facilitando o controle dos gastos e manutenção de seus automóveis.

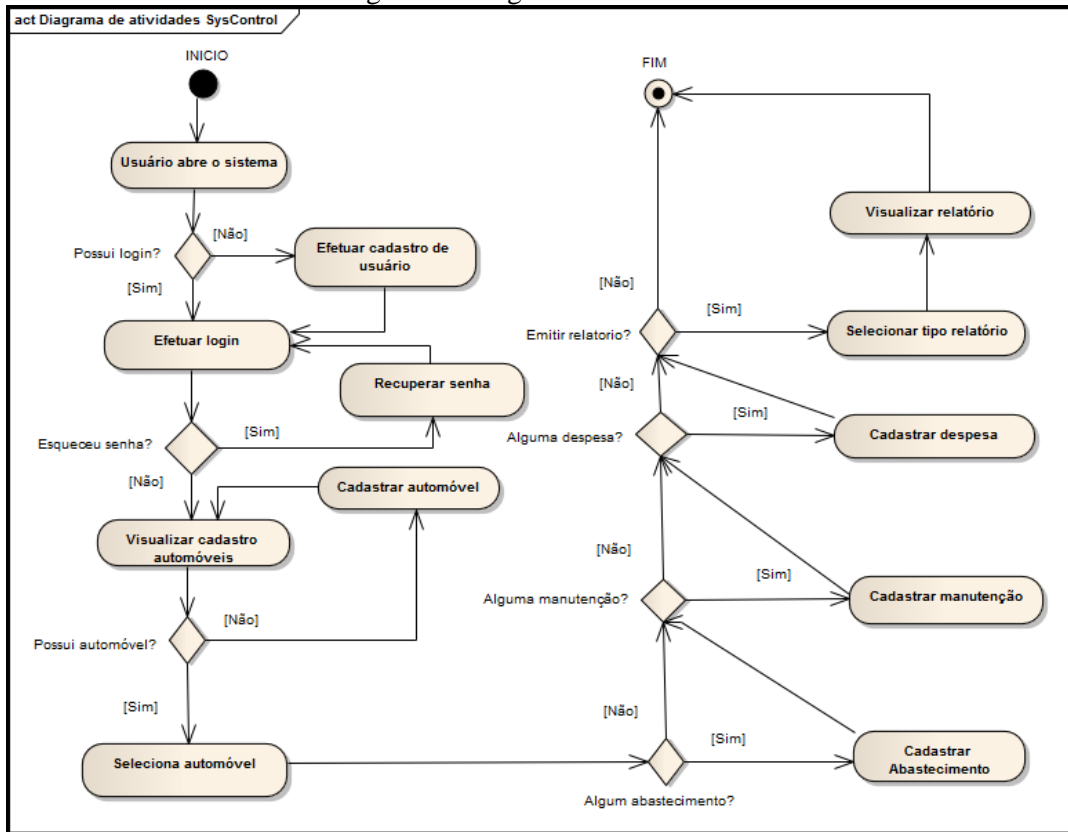
Alguns dos maiores problemas com o controle até então utilizado é ter a integridade das informações, saber o quanto se gastou com determinado automóvel no último mês, ano e quais são as próximas datas de manutenções, troca de óleo, troca de pneus, pagamentos de licenciamento.

De modo geral um dos principais benefícios da aplicação é a mobilidade, o usuário pode instalar a aplicação em qualquer dispositivo móvel com sistema Android que possua versão 4.0 ou superior e registrar as informações dos automóveis que deseja manter o controle. Inicialmente a aplicação exigirá o cadastramento dos dados dos automóveis que deseja-se ter o controle. A inserção destes dados deverá ser feita manualmente para que, com o sistema alimentando, a partir destes dados serão gerados alertas sobre as datas de manutenção, trocas de óleo, pagamento do licenciamento, entre outros.

3.1.1 DIAGRAMA DE ATIVIDADES

Na Figura 7 é apresentado o diagrama de atividades da rotina de manutenção, onde o usuário é responsável por analisar, registrar e atualizar as informações da situação atual do veículo diariamente.

Figura 7 - Diagrama de atividades



3.2 ESPECIFICAÇÃO

Para construção da aplicação será necessária a utilização das seguintes ferramentas:

- Android Studio como plataforma de desenvolvimento;
- SQLite para mecanismo de banco de dados;
- Sparx Systems Enterprise Architect para modelagem de diagramas;
- DB Designer para construção do MER.

O quadro 1 lista os Requisitos Funcionais (RF) para o sistema com os respectivos Casos de Uso (UC) para rastreabilidade.

Quadro 1 - Requisitos Funcionais

Requisitos Funcionais	Caso de Uso
RF01: O sistema deverá permitir manter usuários.	UC01
RF02: O sistema deverá permitir manter automóveis.	UC02
RF03: O sistema deverá permitir visualizar automóveis cadastrados.	UC03
RF04: O sistema deverá emitir notificações alertando manutenções.	UC04
RF05: O sistema deverá consultar histórico de manutenções.	UC05
RF06: O sistema deverá permitir registrar os abastecimentos de combustível.	UC06
RF07: O sistema deverá permitir registrar rotinas de manutenção e despesas.	UC07

O quadro 2 lista os requisitos não funcionais previstos para o sistema

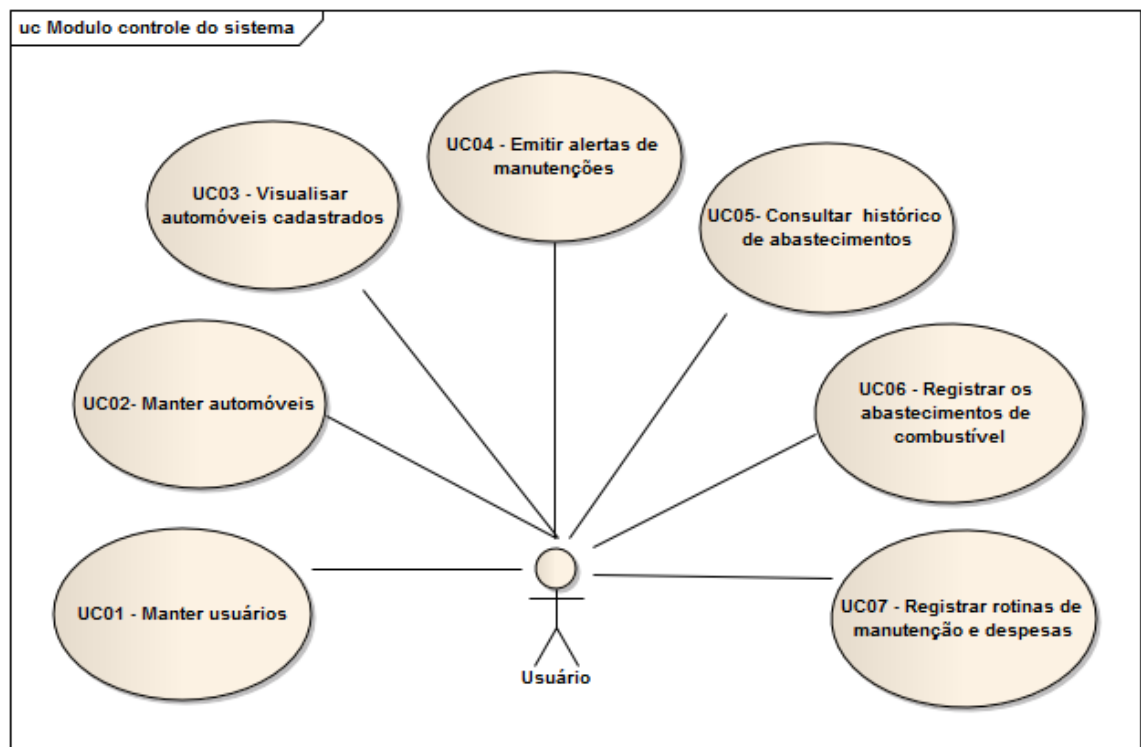
Quadro 2 - Requisitos não funcionais

Requisitos Não Funcionais
RNF01: O sistema deve ser implementado na plataforma Android Studio.
RNF02: O sistema deverá ser compatível com a plataforma Android.
RNF03: O sistema deverá utilizar banco de dados SQLite.
RNF04: O sistema deverá permitir o usuário efetuar login informando usuário e senha.

3.2.2 DIAGRAMA DE CASO DE USO

Esta subseção apresenta o diagrama de caso de uso que será necessário para o entendimento do sistema proposto (Figura 8).

Figura 8 - Diagrama de caso de uso



O Usuário é responsável por controlar e atualizar todas as informações e gerenciar todas as informações relevantes de cadastro no sistema. Os casos de uso deste sistema são:

- a) Manter usuários - o sistema permite cadastrar mais de um usuário que deseja gerenciar as informações de um automóvel;
- b) Manter automóveis - o sistema permite gerenciar a manutenção de um ou mais automóveis sendo vinculado a um único usuário do cadastro de usuário;
- c) Visualizar automóveis cadastrados - ao efetuar *login* o sistema carrega o veículo vinculado àquele usuário, reservando no cadastro de usuário uma consulta com as especificações de “Meu veículo”;
- d) Emitir alertas de manutenções - com base nas informações dos veículos cadastrados, o sistema deve notificar através de alertas quando são as próximas manutenções de acordo com a data limite;

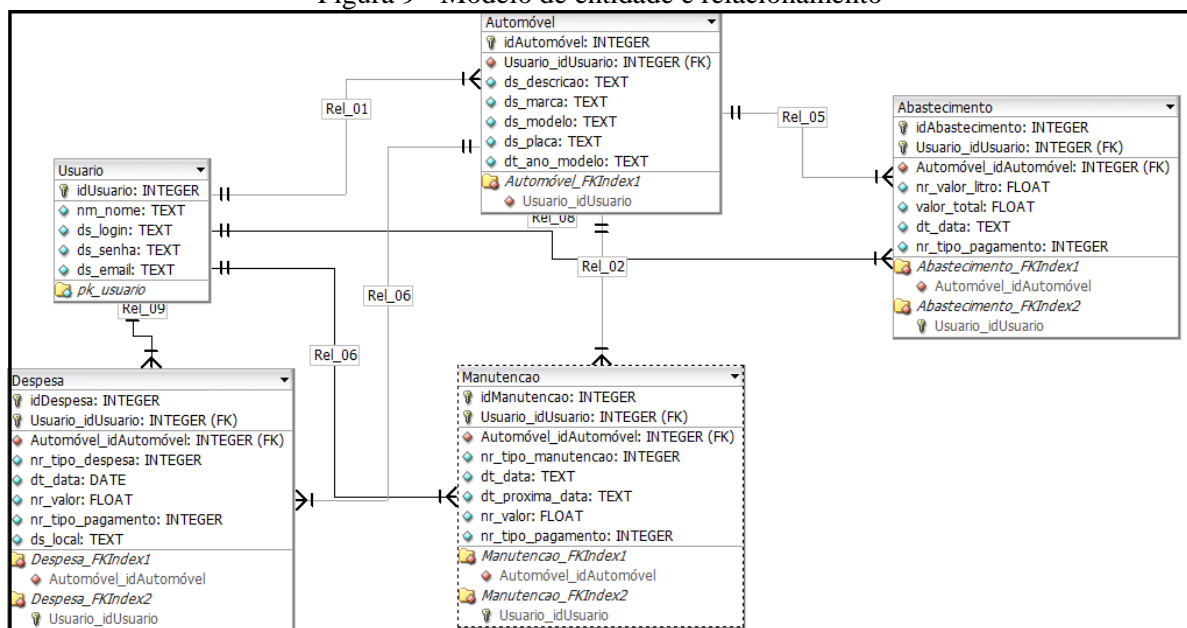
- e) Consultar histórico de abastecimentos - permite o usuário consultar um histórico de todos abastecimentos;
- f) Registrar os abastecimentos de combustível - após cada abastecimento o usuário deve preencher as informações de quantidade de litros e valor pago nos abastecimentos;
- g) Registrar rotinas de manutenção e despesas - o usuário deve registrar as informações de todas atualizações e despesas do veículo.

No Apêndice A encontra-se a descrição completa destes Casos de Uso.

3.2.3 MODELO DE ENTIDADE RELACIONAMENTO

Na Figura 9 é exibido o modelo entidade e relacionamento (MER) do aplicativo desenvolvido. Para construção do modelo foi utilizado a ferramenta DB DesignerFork configurado para notação *cross foot* “Pé-de-galinha”.

Figura 9 - Modelo de entidade e relacionamento



A seguir é apresentada uma breve descrição das entidades criadas para o desenvolvimento do aplicativo:

- a) **Usuário:** entidade que armazena os dados dos usuários cadastrados no aplicativo;
- b) **Automóvel:** entidade que armazena os automóveis cadastrados no aplicativo;
- c) **Abastecimento:** entidade que armazena os dados dos abastecimentos dos automóveis cadastrados no aplicativo;

- d) **Despesa:** entidade que armazena os dados das despesas diárias / mensais dos automóveis cadastrados no aplicativo, esta tabela é alimentada sempre que o usuário cadastra uma nova despesa de um automóvel;
- e) **Manutenção:** entidade que armazena os dados das manutenções dos automóveis, esta tabela é alimentada sempre que o usuário cadastra uma nova manutenção de um automóvel;
- f) **Multa:** entidade que armazena os dados de multas recebidas para automóveis cadastrados, esta tabela é alimentada sempre que o usuário cadastra uma nova multa para um automóvel;
- g) **Imposto:** entidade que armazena os dados de imposto dos automóveis, esta tabela é alimentada sempre que o usuário cadastra um novo imposto de um automóvel;
- h) **Seguro:** entidade que armazena os dados de seguro dos automóveis, esta tabela é alimentada sempre que o usuário cadastra uma nova tarifa de seguro para um automóvel;
- i) **Pneu:** entidade que armazena os dados de troca de 1 ou vários pneus dos automóveis, esta tabela é alimentada sempre que o usuário cadastra uma nova troca de pneus.

No Apêndice B encontra-se o Dicionário de Dados destas tabelas do MER (Figura 8).

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para construção da aplicação foi necessária a utilização das seguintes ferramentas:

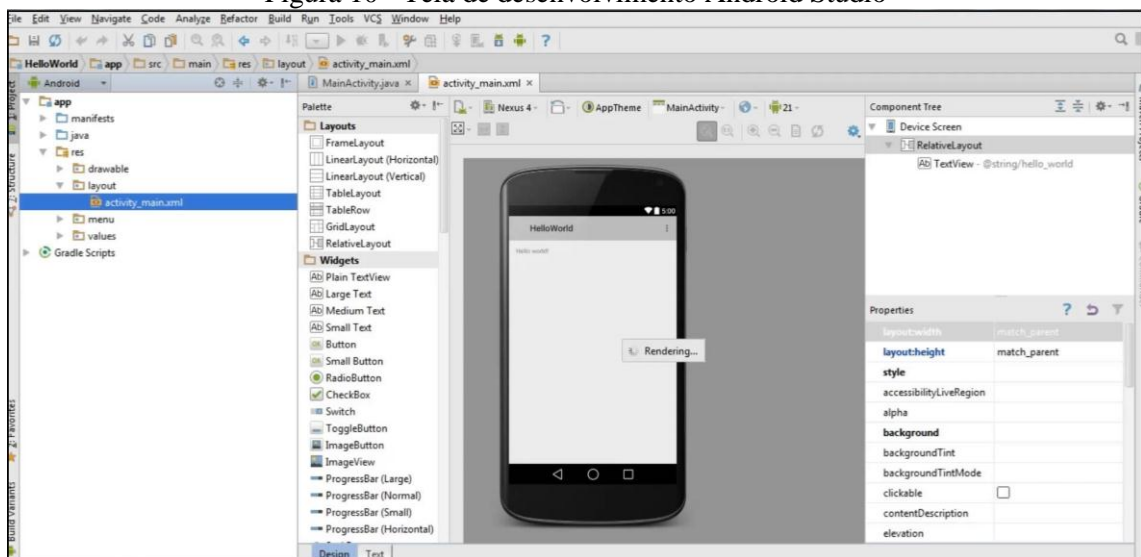
- a) Android Studio como plataforma de desenvolvimento;
- b) SQLite para mecanismo de banco de dados;
- c) Genymotion para testar o aplicativo;
- d) Sparx Systems Enterprise Architect para modelagem de diagramas.

3.3.2 ANDROID STUDIO

Android Studio é a ferramenta de desenvolvimento adotada pela Google como ferramenta oficial de desenvolvimento Android baseada e inspirada na ferramenta IntelliJ IDEA, onde pode-se desenvolver aplicações para Android utilizando a linguagem de programação Java.

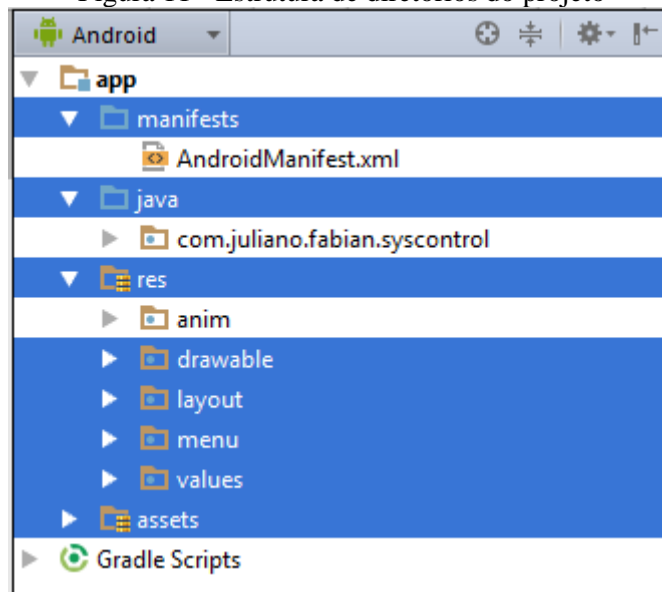
O Android Studio encontra-se disponível para *download* no seu atual site oficial da IDE. Na Figura 10 é apresentada uma imagem com a tela de desenvolvimento do Android Studio.

Figura 10 - Tela de desenvolvimento Android Studio



Para desenvolvimento do aplicativo SysControl foi necessário efetuar o *download* das *Applications Programming Interface (APIs)* versão 8 a 23 através do *Android SDK Manager*. A Figura 11 apresenta a estrutura de diretórios do projeto dividido em modelos conforme a IDE IntelliJ IDEA da JetBrains.

Figura 11 - Estrutura de diretórios do projeto



Os diretórios e arquivos se dividem da seguinte maneira:

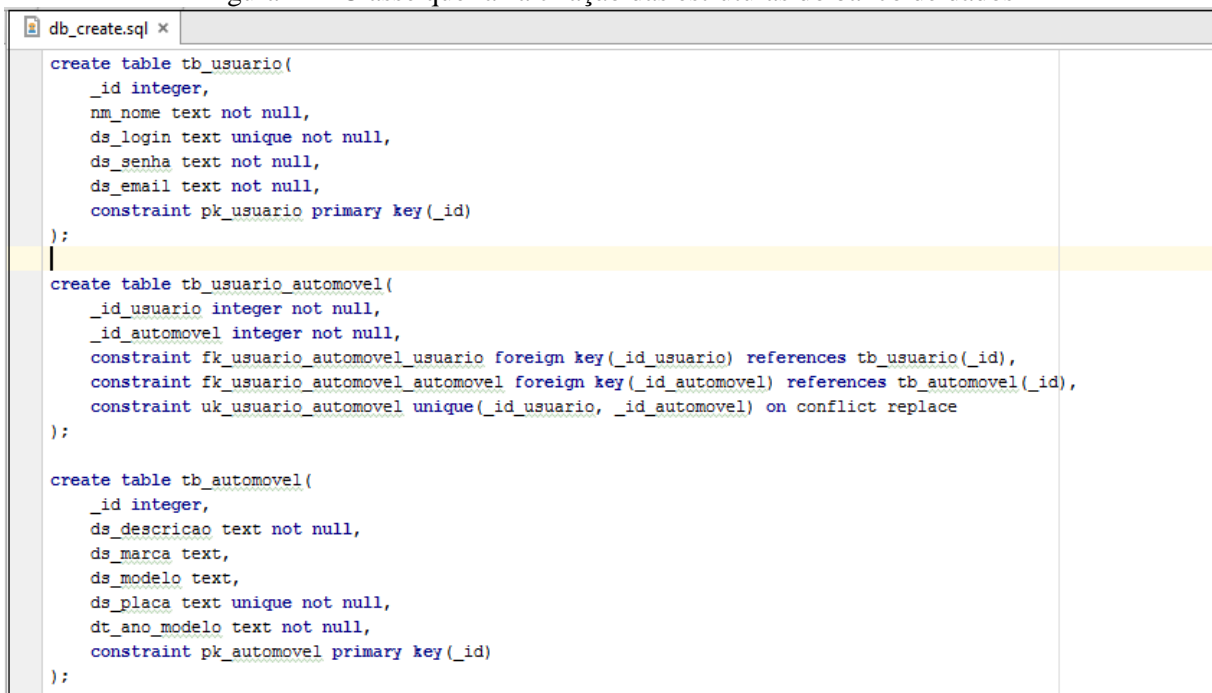
- a) `app` - contém o núcleo do aplicativo, incluindo códigos de classes java, arquivos XML para design da interface gráfica, configurações, ícones.
- b) `manifests` - encontra-se o arquivo `AndroidManifest.xml` que define as propriedades do aplicativo, as activities, as permissões exigidas pelo aplicativo, versão mínima do Android suportada.
- c) `cjava` - diretório que armazena as classes Java que formam o núcleo das funcionalidades do aplicativo.
- d) `res` - contém recursos do projeto como imagens, layout, xmls de configuração listados abaixo:
- e) `drawable`: contém as imagens utilizadas no projeto, cada uma das pastas contém uma versão de uma determinada imagem separadas por definição de tela.
- f) `layout`: contém os arquivos responsáveis pelo design das telas do projeto.
- g) `menu`: contém os arquivos xmls referente aos menus do projeto.
- h) `values`: contém arquivos xmls para parametrização do projeto como cores, mensagens
- i) `assets` - contém o diretório que é responsável por criar as tabelas de banco de dados necessárias na primeira instalação do aplicativo em dispositivos móveis.

3.3.3 SQLITE DATABASE

SQLite é uma biblioteca compacta para gerenciar banco de dados de uso livre que permite salvar os dados de usuários ou de aplicações que estão executando em dispositivos móveis. Por ser desenvolvido para sistemas compactos, o SQLite não é escalável, o que torna seu uso para grandes volumes de informação insustentável. Um banco de dados SQLite é apenas um arquivo único onde as tabelas os dados, *triggers*, chaves primárias e estrangeiras estão todas contidas, as aplicações leem e escrevem no banco de dados pela chamada do SQLite, ou seja, quando é realizado um *select* ou *update* o SQLite lê e escreve para este arquivo.

Na Figura 12, pode-se ver uma imagem de parte da classe que faz a criação das estruturas do banco de dados.

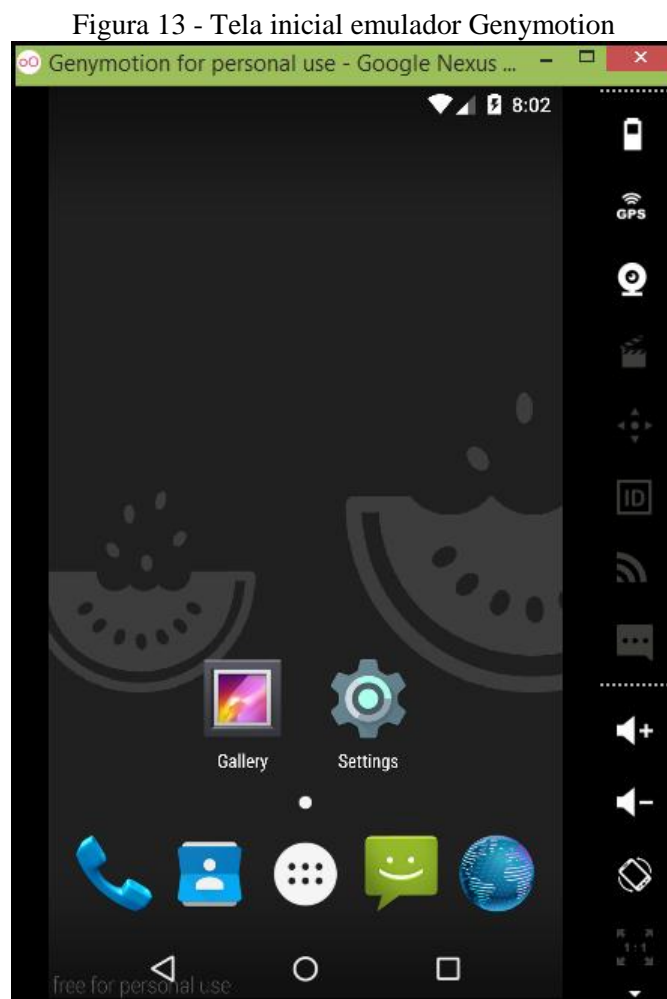
Figura 12 - Classe que faz a criação das estruturas do banco de dados

The image shows a screenshot of a code editor window titled 'db_create.sql'. The code defines three tables: 'tb_usuario', 'tb_usuario_automovel', and 'tb_automovel'. 'tb_usuario' has columns for id, name, login, password, and email, with 'id' as the primary key. 'tb_usuario_automovel' has columns for user id and vehicle id, with foreign key constraints to 'tb_usuario' and 'tb_automovel', and a unique constraint on the combination of both. 'tb_automovel' has columns for id, description, brand, model, license plate, and year, with 'id' as the primary key.

```
create table tb_usuario(  
  _id integer,  
  nm_nome text not null,  
  ds_login text unique not null,  
  ds_senha text not null,  
  ds_email text not null,  
  constraint pk_usuario primary key(_id)  
);  
  
create table tb_usuario_automovel(  
  _id_usuario integer not null,  
  _id_automovel integer not null,  
  constraint fk_usuario_automovel_usuario foreign key(_id_usuario) references tb_usuario(_id),  
  constraint fk_usuario_automovel_automovel foreign key(_id_automovel) references tb_automovel(_id),  
  constraint uk_usuario_automovel unique(_id_usuario, _id_automovel) on conflict replace  
);  
  
create table tb_automovel(  
  _id integer,  
  ds_descricao text not null,  
  ds_marca text,  
  ds_modelo text,  
  ds_placa text unique not null,  
  dt_ano_modelo text not null,  
  constraint pk_automovel primary key(_id)  
);
```

3.3.4 EMULADOR GENYMOTION

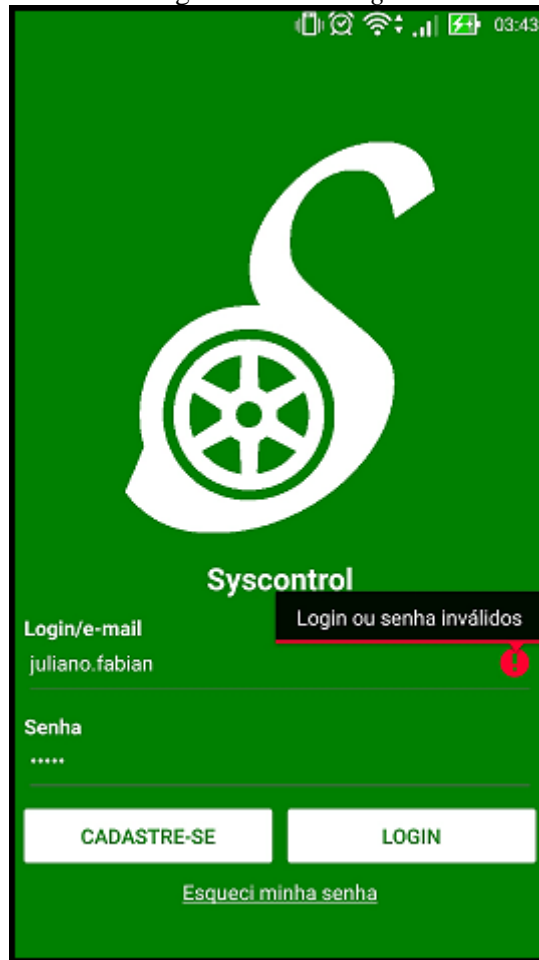
Genymotion é um emulador para Android que vem com um conjunto de máquinas virtuais pré-configuradas e que permite testar funcionalidades de aplicações Android. O Genymotion é possível ser executado diretamente do Android Studio instalando o *plugin* Genymotion. Um dos requisitos é a necessidade de ter o *virtualbox* pré-instalado no sistema. Na Figura 13 pode-se ver a tela inicial do emulador.



3.3.5 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Esta seção apresenta a tela principal do aplicativo, seus componentes, e a visualização de relatórios.

Inicialmente ao abrir o sistema é apresentada a tela para o usuário efetuar o *login* e senha. Ao serem validados os dados o sistema, abrirá a tela principal onde o usuário pode acessar as funções do sistema através do menu. A Figura 14 exhibe a tela de *login* do sistema.

Figura 14 - Tela *login*

Caso o usuário ainda não possua cadastro, ao clicar no botão “Cadastre-se”, abrirá uma nova tela para edição de usuário. Ao acessar a tela de edição de usuário primeiramente o usuário deverá informar o nome, *login*, email, senha e repetir senha. Após preenchidos todos os campos, o usuário deverá clicar no botão “Salvar”. A Figura 15 apresenta a tela de edição de usuário no aplicativo.

Figura 15 - Tela de edição de usuário

Nome
Juliano Fabian

Login
juliano.fabian

Email
juba_blu@hotmail.com

Senha
.....

Repita a senha
.....

1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m Feito

Após realizado o cadastro o usuário poderá efetuar *login* no aplicativo informando o usuário e a senha cadastrados. A Figura 16 apresenta a tela principal do aplicativo com as seguintes opções:

- automóveis - permite que o usuário cadastre um novo automóvel a ser gerenciado no aplicativo;
- abastecimento - permite o usuário cadastre todos os abastecimentos de automóveis, sendo estes guardados em um histórico que poderá ser consultado em relatórios;
- despesas - permite que o usuário possa cadastrar todas as despesas dos automóveis, sendo estas guardadas em um histórico que poderá ser consultado em relatórios na funcionalidade de relatórios do aplicativo.
- manutenções - permite que o usuário cadastre todas as manutenções dos automóveis listadas pelo aplicativo (alinhamentos, balanceamentos, baterias,

escapamentos, faróis / lanternas, freios, outros serviços, pneus novos, pneus rodízio, revisão, suspensões).

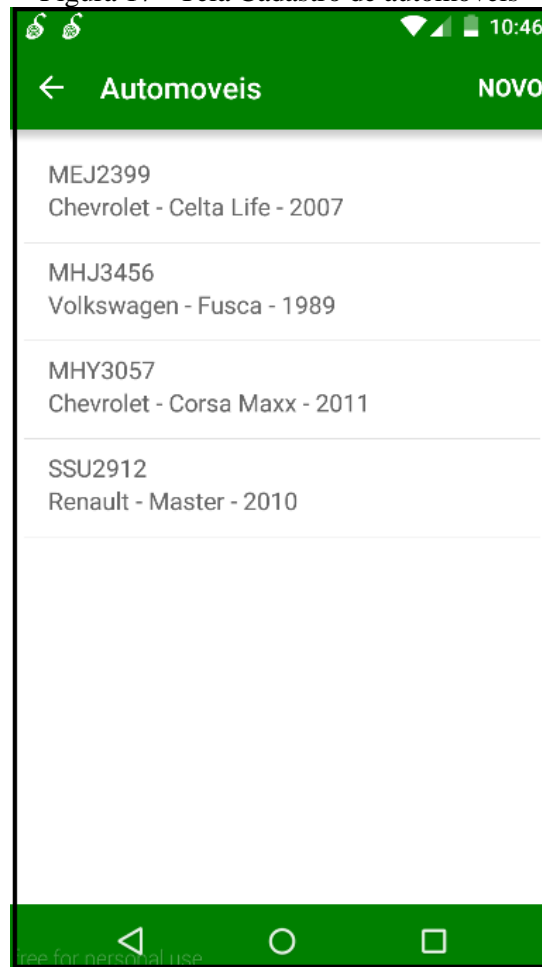
- e) *relatórios* - permite o usuário gerar relatórios e visualiza-los de acordo com os filtros aplicados.
- f) *minha conta* - permite o usuário gerenciar as informações do cadastro do usuário como por exemplo, nome, login, email e senha.

Figura 16 - Tela principal do aplicativo



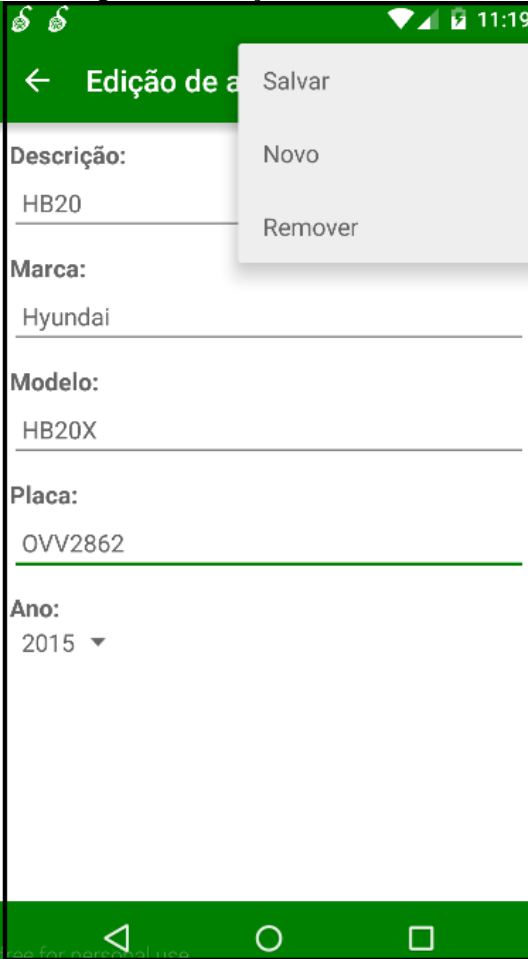
Para consultar o cadastro de um automóvel o usuário deverá selecionar a opção *Automóveis* da tela principal, selecionando esta opção, abrirá uma nova tela listando os automóveis já cadastrados. A Figura 17 apresenta a tela *Automóveis* com a lista de automóveis já cadastrados.

Figura 17 - Tela Cadastro de automóveis



Para cadastrar um novo automóvel o usuário deverá clicar em “Novo” e informar a descrição, marca, modelo, placa e ano. A Figura 18 apresenta a tela de Edição de automóveis.

Figura 18 - Edição de automóveis



The screenshot displays a mobile application interface for editing car information. The screen is titled "Edição de a" (partially visible). The form contains the following fields:

- Descrição:** HB20
- Marca:** Hyundai
- Modelo:** HB20X
- Placa:** OVV2862
- Ano:** 2015 (with a dropdown arrow)

A context menu is open over the "Salvar" button, showing three options: "Salvar", "Novo", and "Remover". The top status bar shows the time as 11:19. The bottom navigation bar is green with standard Android navigation icons.

Com todos os campos preenchidos o usuário deverá clicar no botão “Salvar” e o sistema irá validar as informações, se os todos os campos estiverem preenchidos corretamente o sistema apresentará a mensagem, como mostra a Figura 19.

Figura 19 - Tela de edição de automóveis

← Edição de auto... SELECIONAR ☰

Descrição:
HB20

Marca:
Hyundai

Modelo:
HB20X

Placa:
OVV2862

Ano:
2015 ▾

Registro salvo

Após realizado o cadastro de automóveis, caso o usuário possua mais de um automóvel cadastrado em sua conta, ele deverá selecionar qual automóvel deseja definir como padrão para informar os abastecimentos, manutenções e despesas. Para isso, o usuário deverá clicar sobre um dos automóveis listados na tela *Automóveis* e clicar no botão “Selecionar”.

A Figura 20 apresenta um exemplo de como selecionar um automóvel.

Figura 20 - Selecionando automóvel



Edição de auto... SELECIONAR

Descrição:
HB20

Marca:
Hyundai

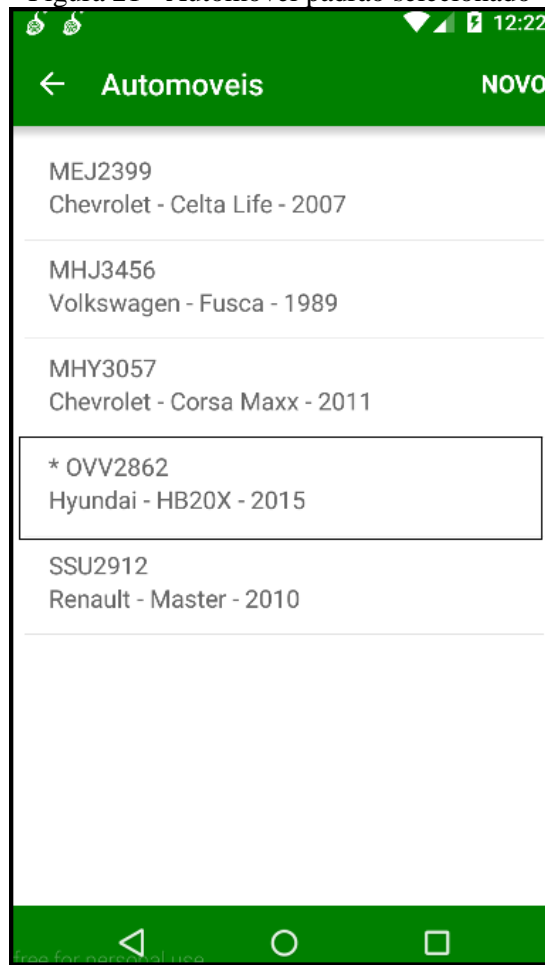
Modelo:
HB20X

Placa:
OVV2862

Ano:
2015 ▼

Após selecionado o automóvel, na tela de Automóveis é destacado com um asterisco o automóvel padrão para aquele usuário registrar as informações que venham a ser registradas. A Figura 21 apresenta um exemplo do automóvel selecionado como padrão.

Figura 21 - Automóvel padrão selecionado



Com o automóvel selecionado, caso recentemente o usuário tenha efetuado um abastecimento, o usuário deverá clicar em *Abastecimento* na tela principal. Selecionando esta opção, abrirá uma nova tela listando todos os abastecimentos já cadastrados para o(s) automóvel(eis) do usuário. A Figura 22 apresenta a tela de cadastro de abastecimentos.

Figura 22 - Cadastro de Abastecimento



Total R\$	Valor por litro R\$	Data
R\$55,00	R\$2,59	10/12/2015
R\$80,00	R\$3,29	09/12/2015
R\$20,00	R\$3,29	08/12/2015
R\$60,00	R\$3,29	08/12/2015
R\$75,00	R\$3,59	08/12/2015
R\$80,00	R\$3,49	06/12/2015
R\$50,00	R\$3,49	01/12/2015
R\$20,00	R\$3,00	10/11/2015

Para cadastrar um novo abastecimento, o usuário deverá clicar no botão “Novo” e informar valor litro, valor total, tipo pagamento e data. A Figura 23 apresenta a tela de edição de Abastecimento.

Figura 23 - Tela de edição de abastecimento

Automovel
HB20

Valor do litro:
R\$3,59

Valor total:
R\$50,00

Tipo de pagamento:
Dinheiro

Data:
10 Dezembro 2015

Com todos os campos preenchidos o usuário deverá clicar no botão “Salvar” e o sistema irá validar as informações, se todos os campos estiverem preenchidos corretamente o sistema apresentará a mensagem, como mostra a Figura 24.

Figura 24 - Tela de edição de abastecimento

Automovel
HB20

Valor do litro:
R\$3,59

Valor total:
R\$50,00

Tipo de pagamento:
Dinheiro

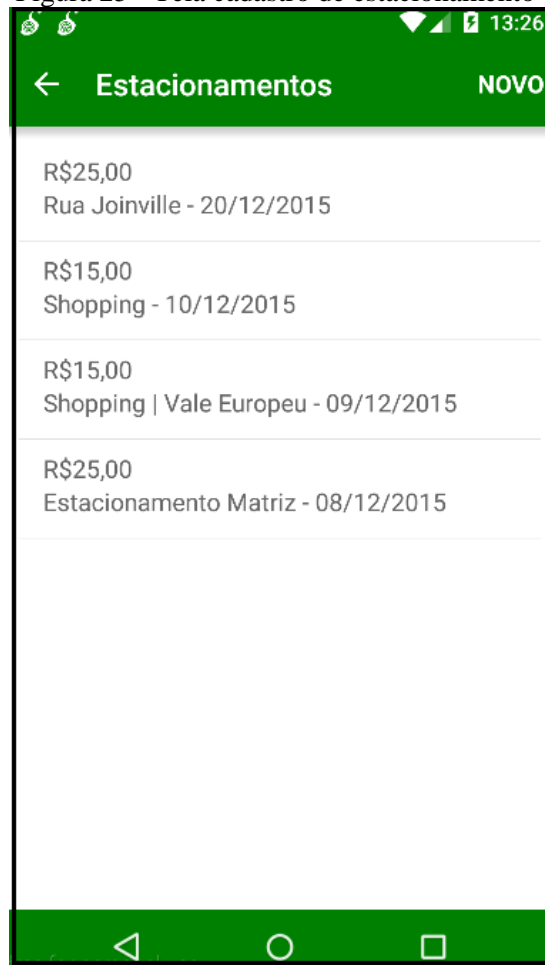
Data:
10 Dezembro 2015

Registro salvo

Após salvo, o novo registro de abastecimento aparecerá na lista de abastecimentos cadastrados para o usuário.

Outra opção da tela principal é a de Despesas, nesta tela são registradas as opções fixas de despesas como Estacionamento, Pedágios e Impostos. Selecionando a opção Estacionamento como exemplo, abrirá uma nova tela onde são listadas todas despesas de estacionamento caso já existam para o usuário. Na Figura 25 é apresentada a tela de cadastro de estacionamentos.

Figura 25 - Tela cadastro de estacionamento



Para cadastrar uma despesa de estacionamento o usuário deverá clicar no botão “Novo” e informar valor, local, tipo de pagamento e data. Na Figura 26 é apresentada a tela de edição da despesa estacionamentos.

Figura 26 - Tela de edição de estacionamentos.



Automovel
HB20

Valor:
R\$25,00

Local:
Torre Estacionamento

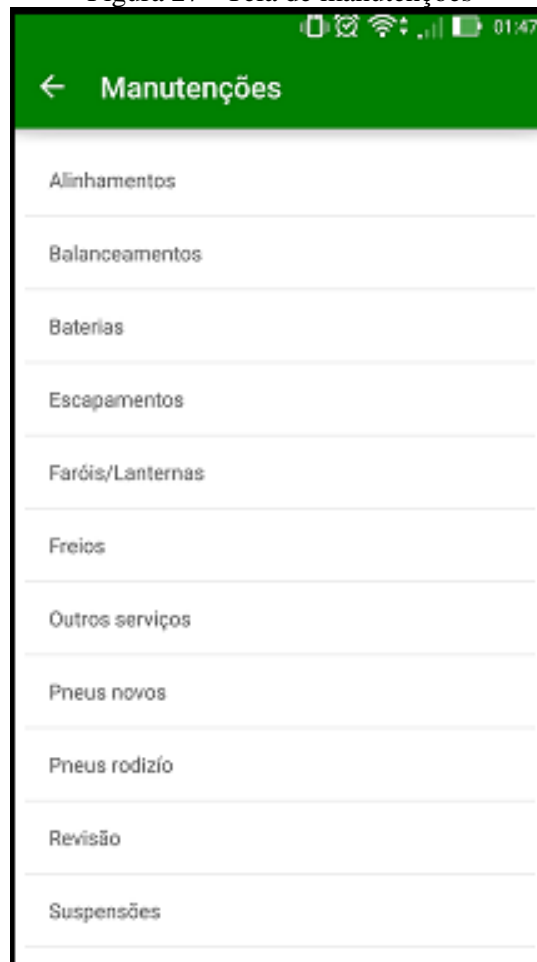
Tipo de pagamento:
Dinheiro

Data:
10 Dezembro 2015

Com todos os campos preenchidos o usuário deverá clicar no botão “Salvar” e o sistema irá validar as informações, se todos os campos estiverem preenchidos corretamente o sistema apresentará a mensagem de “Registro salvo”.

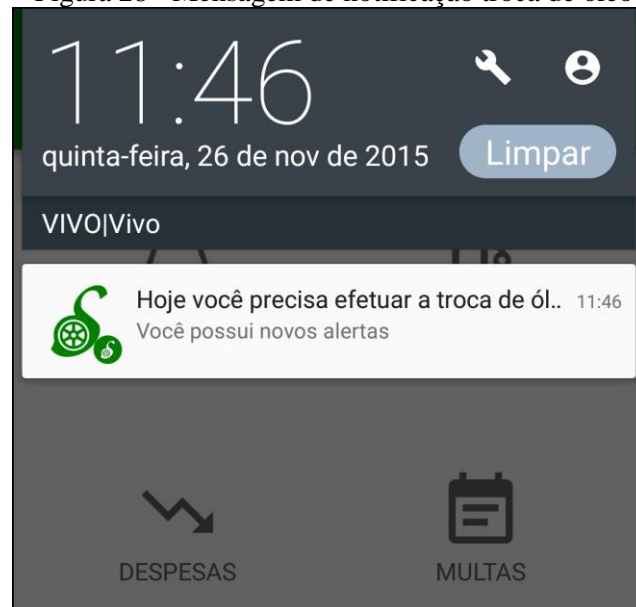
Outra opção da tela principal é a de *Manutenções*, nesta tela o sistema abrirá todas as opções fixas de manutenção de acordo com o que for realizado no automóvel. A Figura 27 apresenta a tela de manutenções, com as opções que o sistema permite registrar.

Figura 27 - Tela de manutenções




Cada uma das opções listadas permite efetuar o cadastro com as informações relevantes da manutenção e registro da data/hora que foi realizado a manutenção. Com a data que foi realizada a manutenção, futuramente o sistema deverá notificar que existem manutenções a serem realizadas de acordo com a data que foi preenchida. Por exemplo: ao efetuar a troca de óleo é cadastrado no sistema a data de quando foi efetuado e quando deverá ser a próxima troca de acordo com os registros existentes para troca de óleo. Na Figura 28 o sistema apresenta um exemplo de notificação de troca de óleo.

Figura 28 - Mensagem de notificação troca de óleo



Outra opção da tela principal do sistema é a visualização de relatórios que permitem visualizar os históricos de gastos com despesas, as manutenções e os abastecimentos dos automóveis. Caso o usuário clique em *Relatórios*, será aberta uma tela com as opções de relatórios que podem ser visualizados. A Figura 29 apresenta a tela de relatórios do aplicativo.

Figura 29 - Tela de relatórios do aplicativo



Relatórios

Tipo de relatório:
Abastecimento ▾

Agrupar por:
 Automóvel Usuário

Período inicial:
01/11/2015

Período final:
30/11/2015

VISUALIZAR

Nesta tela tem se as opções de relatórios: os relatórios podem ser gerados aplicando três filtros, sendo eles combinados ou não. No filtro Tipo de relatório, podem ser escolhidas as opções por: Abastecimentos, Manutenções e Despesas. Escolhido o filtro de relatório, seleciona-se o filtro para agrupar por Automóvel ou Usuário. Por exemplo: Caso o usuário selecione o tipo de relatório Abastecimentos combinado com o filtro para agrupar por Automóvel, Data inicial 01/12/2015 e Data final 14/12/2015, ao clicar no botão “Visualizar” o sistema exibirá um relatório com todos os abastecimentos de acordo com estes filtros. Na Figura 30 é apresentado o relatório de Abastecimentos do exemplo.

Figura 30 - Relatório de abastecimentos

The screenshot shows a mobile application interface for generating a fuel report. The title bar is green with a white back arrow and the text "Relatórios". The status bar at the top shows signal strength, Wi-Fi, battery, and the time 14:10. The main content area is white and contains the following elements:

- Tipo de relatório:** Abastecimentos (with a dropdown arrow)
- Exibir:** Meus automóveis Todos automóveis
- Data inicial:** 01 (dropdown), Dezembro (dropdown), 2015 (dropdown)
- Data final:** 14 (dropdown), Dezembro (dropdown), 2015 (dropdown)
- Report entries:
 - 10/12/2015 - R\$55,00
Juliano Fabian - Fusca
 - 10/12/2015 - R\$50,00
Juliano Fabian - HB20
 - 09/12/2015 - R\$80,00
- Total: R\$470,00**
- A grey button labeled "VISUALIZAR" at the bottom.

The bottom of the screen shows a green navigation bar with standard Android icons (back, home, recent apps) and a small watermark "free for personal use" in the bottom left corner.

Caso o usuário selecione para visualizar relatório com as mesmas datas do período analisado, porém selecionando o tipo de relatório “Manutenções” e filtro para agrupar por Usuário, ao clicar no botão “Visualizar” o sistema exibirá um relatório com todas manutenções realizadas no período de 01/12/2015 a 14/12/2015. Na Figura 31 é apresentado o relatório de manutenções do exemplo.

Figura 31 - Relatório de manutenções

← Relatórios

Tipo de relatório:
Manutenções ▾

Exibir:
 Meus automóveis Todos automóveis

Data inicial:
01 ▾ Dezembro ▾ 2015 ▾

Data final:
14 ▾ Dezembro ▾ 2015 ▾

13/12/2015 - R\$60,00
Juliano Fabian - Fusca

10/12/2015 - R\$110,00
Juliano Fabian - Fusca

09/12/2015 - R\$33,33

Total: R\$1.124,33

VISUALIZAR

3.4 RESULTADOS E DISCUSSÕES

O objetivo geral deste trabalho foi desenvolver um aplicativo para registrar informações de manutenções, despesas e abastecimentos de automóveis para dispositivos móveis, substituindo processos manuais de anotações e planilhas. Com o histórico das informações registradas é possível gerar relatórios específicos de abastecimentos, despesas, manutenções, combinando os filtros de agrupamento e período.

O objetivo específico de alertar manutenções integrando o aplicativo à agenda não foi alcançado devido necessidade de um maior estudo aprofundado da plataforma Android juntamente com os componentes Services. No entanto, os trabalhos correlatos de Pedron (1993) e Hoffmann (2010) não foram desenvolvidos para dispositivos móveis, apenas o aplicativo correlato Carrorama da Going2 possui algumas características em comum conforme podemos analisar no Quadro 3.

Quadro 3 - Comparação entre os aplicativos correlatos

Características	Pedron (1993)	Hoffmann (2010)	Carrorama	Syscontrol
Linguagem de programação	Cobol	Java	Java	Java
Banco de dados	-	MySQL	-	SQLite Database
Sistema	Windows	Windows XP	Android	Android
Foco do trabalho	Controle de frota para prefeitura Municipal de Timbó	Controle de gastos com combustível da frota empresa Unimed	Controle de gastos, despesas de financiamento, seguro abastecimentos	Gerenciar manutenções, abastecimento, despesas, gerar Relatórios
Ambiente de desenvolvimento	Cobol	NetBeans	-	Android Studio

Comparando com o aplicativo desenvolvido por Pedron (1993), pode-se notar que o sistema operacional foi desenvolvido para plataforma *Windows* e consiste no controle de autorização de manutenções, abastecimentos, porém não gera notificação. O objetivo principal do aplicativo de Pedron (1993) é o controle de frotas da prefeitura municipal de Timbó.

Comparando com o aplicativo desenvolvido por Hoffmann (2010), também foi desenvolvido para plataforma *Windows*, trata principalmente a reserva de veículos da frota e controlar os gastos com combustível, porém não faz a emissão de relatórios dos abastecimentos.

O aplicativo Carrorama desenvolvido pela empresa Going2 engloba funcionalidades para registrar informações completas do automóvel desde gastos com abastecimentos e aspectos mais específicos de seguro, financiamento e acessórios do automóvel. O aplicativo é compatível com a plataforma Android e iOS.

4. CONCLUSÕES

Com o avanço do número de dispositivos móveis no mercado, atualmente os usuários tem buscado cada vez mais aplicativos que auxiliam no controle e gerenciamento de atividades do cotidiano. Identificado esta tendência de mercado, foi desenvolvido neste trabalho um aplicativo para gerenciar as manutenções de automóveis, permitindo o proprietário manter um controle e histórico do automóvel.

O aplicativo foi desenvolvido para plataforma Android. Uma das características principais do aplicativo é ter a informação a disposição em qualquer lugar, permitir gerir as informações do automóvel sem a necessidade de um computador ou conexão com a internet.

Um diferencial deste trabalho é a utilização da ferramenta Android Studio que passou a ser uma referência oficial de desenvolvimento para Android, anunciada recentemente pela Google. O Android se demonstrou adequado para o desenvolvimento juntamente com o banco de dados *SQLite Database* que é uma biblioteca compacta para gerenciar banco de dados de uso livre e que permite salvar dados de aplicações que estão executando em dispositivos móveis.

Uma das funcionalidades desenvolvidas que podemos considerar como a principal, é o controle de abastecimentos, onde o usuário pode registrar todos os abastecimentos realizados para os automóveis cadastrados em sua conta e através desses registros visualizar relatórios de quanto está sendo gasto com combustível durante o período especificado.

Outra funcionalidade é o cadastro de manutenções, onde o usuário pode registrar todas as manutenções realizadas nos automóveis e ao preencher as informações é necessário informar a próxima data o qual é recomendando efetuar novamente aquele tipo de manutenção. Com a data próxima informada, por meio de um Service o sistema notificará com 3 dias de antecedência que existe uma nova manutenção a ser realizada.

Diante de todos os resultados obtidos com a aplicação desenvolvida, pode-se considerar que foram satisfatórios, visto que as principais funcionalidades atendem a necessidade do proprietário gerenciar as informações do automóvel através de seu dispositivo móvel.

4.1 EXTENSÕES

O aplicativo foi desenvolvido para que os usuários possam inserir as informações de automóvel e partir destas tomar ações, resultando em redução de custos, manutenções planejadas e melhoraria da vida útil do automóvel.

Diante deste cenário podem ser sugeridas as seguintes extensões:

- a) implementar a aplicação para plataformas Windows Phone e iOS;
- b) desenvolver funcionalidade de envio de notificação por e-mail;
- c) exportar relatórios para pdf;
- d) controlar parcelas de financiamento.
- e) Permitir modificar opções dos campos de manutenções e despesas;
- f) Integrar as manutenções com a agenda do Android.

REFERÊNCIAS

- AMADOR, João Gabriel. **Estudo Mostra Crescimento no Uso de Dispositivos Móveis e domínio Android**. Brasília, 2015. Disponível em: <http://www.correiobraziliense.com.br/app/noticia/tecnologia/2015/01/17/interna_tecnologia,466691/estudo-mostra-crescimento-no-uso-de-dispositivos-moveis-e-dominio-andr.shtml>. Acesso em: 09 set. 2015.
- ANDROID DEVELOPERS. **The developer's guide**. [S.l], 2011. Disponível em: <<http://developer.android.com/guide/>>. Acesso em: 04 ago. 2015.
- AVRAM, Abel. **Desenvolvedores Android são encorajados a migrar do Eclipse para o Android Studio 1.0**. [S.l], 2014. Disponível em: <<http://www.infoq.com/br/news/2014/12/android-studio-1>>
- CARRORAMA, **GooglePlay**. [S.l], 2014. Disponível em: <<https://play.google.com/store/apps/details?id=br.com.going2.carrorama>>. Acesso em: 09 set. 2015.
- DAL PIAZ, Dário **O futuro da tecnologia portáteis estão revolucionando o mercado**. [S.l], 2013. Disponível em: <<http://www.tecmundo.com.br/futuro/43584-o-futuro-da-tecnologia-portateis-estao-revolucionando-o-mercado.htm#ixzz2ciRDUtG4>>. Acesso em: 31 out. 2015.
- HASHIMI, Sayed Y. ; KOMATINENI, Satya. **Pro Android**. São Paulo: Apress, 2009.
- HOFFMANN, Ricardo. **Sistema para reserva de veículos de uma cooperativa de saúde**. [S.l], 2010. 70 f. Trabalho de Conclusão de Curso (Bacharel em Sistemas de Informação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- JOHNSON, Thienne M. **JAVA para dispositivos móveis: desenvolvendo aplicações com J2ME**. São Paulo: Novatec, 2007.
- LECHETA R. Ricardo. **Android Aprenda a criar aplicações para dispositivos móveis com o Android SDK 3ª Edição**. São Paulo, 2009. Disponível em: <<https://www.novatec.com.br/livros/googleand3/capitulo9788575223444.pdf>>. Acesso em: 05 ago. 2015.
- MACEDO, Rodrigo. **Android na Veia** [S.l], 2011. Disponível em: <<http://ww.androidnaveia.com.br/2011/02/desenvolvimento.html>>. Acesso em 26/11/2015.
- OPEN HANDSET ALLIANCE. **Industry Leaders Announce Open Platform for Mobile Devices**. [S.l], 2007. Disponível em: <http://www.openhandsetalliance.com/press_110507.html> Acesso em 26/11/2015.
- PEDRON, Edson J. **Projeto de um sistema de frota de veículos**. 1993. 48 f. Trabalho de Conclusão de Curso (Bacharel em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- POSSER, Lucas Nascimento. **Computação Móvel MLearning: Estudo e construção de um protótipo para smartphone. Porto Alegre, 2006. Disponível em:** <http://www.uniritter.edu.br/graduacao/informatica/sistemas/downloads/Computacao_Movel_e_M-Learning.pdf>. Acesso em: 31 out. 2015.
- SILVA, Francisco Mendes Citeli. **Entendendo o ciclo de vida de uma aplicação**. [S.l], 2014. Disponível em: <<http://www.devmedia.com.br/entendendo-o-ciclo-de-vida-de-uma-aplicacao-android/22922>> Acesso em: 31 out. 2015.

SILVEIRA, Felipe. **Criando um Service em Android**, [S.l], 2015. Disponível em <<http://www.felipesilveira.com.br/2015/03/service-em-android/>> Acesso em: 05/12/2015.

APÊNDICE A – Descrição dos Casos de Uso

Este Apêndice apresenta a descrição dos principais casos de uso descritos na seção de especificação deste trabalho, conforme o Quadro 4.

Quadro 4 - Descrição dos casos de uso

<p>UC01 Manter Usuário Permite o usuário cadastrar as informações da conta do usuário, os dados de um novo usuário do sistema, bem como alterar ou excluir informações do usuário.</p> <p>Cenários Manter usuário {Principal}</p> <ol style="list-style-type: none"> 1. Usuário clica no botão “Cadastrar-se da tela de login”. 2. Sistema exibe tela de edição do usuário. 3. Usuário informa nome, login, email, senha repetir senha. 4. Sistema valida os dados informados e salva o novo registro. <p>Usuário já cadastrado {Exceção} No passo 3 caso já exista um usuário cadastrado, ao preencher o campo “nome” ou “login”, será exibida a mensagem “usuário já cadastrado”.</p> <p>UC02 Manter automóveis Permite o usuário cadastrar um novo automóvel no sistema.</p> <p>Cenários Manter automóveis {Principal}</p> <ol style="list-style-type: none"> 1. Usuário clica no botão “Automóveis” da tela principal. 2. Sistema exibe automóveis cadastrados caso já existe. 3. Usuário clica no botão novo informa “descrição”, “marca”, “modelo”, “placa”, “ano” e clica em Salvar”. 4. Sistema valida os dados informados e salva o novo automóvel. <p>Automóvel já cadastrado {Exceção} No passo 3, caso já exista uma automóvel cadastrado, ao preencher o campo “placa”, será exibida a mensagem “placa já cadastrada”.</p> <p>UC03 Visualizar automóveis cadastrados Permite visualizar os automóveis já cadastrados no sistema</p> <p>Cenários Visualizar automóveis cadastrados {Principal}</p> <ol style="list-style-type: none"> 1. Usuário clica no botão “Automóveis”. 2. Sistema apresenta automóveis cadastrados, <p>Alterar cadastros de automóveis {Alternativo}</p> <ol style="list-style-type: none"> 1. Usuário seleciona o automóvel e a operação que deseja fazer, editar, novo ou deletar. 2. Sistema valida os dados informados e salva.
--

UC04 Cadastrar manutenção / Emitir alertas de manutenções

Permite o usuário consultar as manutenções que foram cadastradas e que estão pendentes a serem realizadas.

Cenários**Visualizar manutenção {Principal}**

1. Usuário acessa tela de Manutenções
2. Sistema exibe as manutenções realizadas com a “data próxima”.
3. Usuário seleciona a opção desejada e caso a data de manutenção já tenha expirado, com 4 dias de antecedência o sistema notificará.
4. Sistema retorna a tela inicial.

Manutenção não cadastrada {Exceção}

No passo 2, caso ainda não tenha manutenção cadastrada, o usuário escolhe o tipo de manutenção clica no botão “Novo” e informa o “valor”, “tipo pagamento”, “data”, e “próxima data”.

UC05 Consultar histórico de abastecimentos

Permite consultar todos os abastecimentos realizados.

Cenários**Consultar abastecimentos de combustível {Principal}**

1. Usuário clica no botão “Abastecimentos”.
2. Sistema exibe todos os abastecimentos realizados para os automóveis do usuário.

No passo 2, caso não tenha abastecimentos cadastrados,

UC06 Registrar os abastecimentos de combustível {Principal}

Permite registrar todas as despesas de combustível

Cenários**Registrar os abastecimentos de combustível {Principal}**

1. Usuário clica no botão “Abastecimento”.
2. O usuário clica no botão novo e informa o “valor”, “tipo pagamento”, “data” e “próxima data”.
3. Sistema valida os dados informados e salva o novo abastecimento.

UC07 Registrar rotinas de despesas

Permite registrar as despesas.

Cenários**Registrar rotinas de despesas {Principal}**

1. Usuário clica no botão “Despesas”.
2. Sistema exibe os tipos de despesas, clicando no tipo é exibido detalhes da despesa.
3. Caso a despesa ainda não tenha a despesa cadastrada, o usuário informa o “valor”, “local”, “tipo de pagamento” e “data”.
4. Sistema valida os dados informados e salva o novo registro de despesa.

Registro não cadastrado {Exceção}

No passo 2 ao inserir as informações do novo registro, caso o usuário não informe os dados obrigatórios, o sistema apresenta uma mensagem “Dados obrigatórios não preenchidos”.

APÊNDICE B – Dicionário de Dados

Neste apêndice, estão listadas as estruturas básicas de dados que são utilizadas no banco de dados do sistema. Os tipos de dados primários usados estão descritos a seguir:

- a) `Text`: valores textuais / descritivos;
- b) `Integer`: valores numéricos inteiros;
- c) `Float`: valores numéricos de ponto flutuante;
- d) `DateTime`: valores que representam data;

Automóvel				
Armazena os automóveis cadastrados no sistema				
Campo	Descrição	Tipo	Chave primária	Chave estrangeira
idAutomóvel	Número	<i>Integer</i>	Sim	Sim
ds_descricao	Descrição do automóvel	<i>Varchar</i>	Não	Não
ds_marca	Nome da marca	<i>Varchar</i>	Não	Não
ds_modelo	Descrição modelo do automóvel	<i>Varchar</i>	Não	Não
ds_placa	Placa do automóvel	<i>Varchar</i>	Não	Não
dt_ano_modelo	Data modelo automóvel	<i>Date</i>	Não	Não

Usuário				
Armazena os usuários cadastrados no sistema				
Campo	Descrição	Tipo	Chave primária	Chave estrangeira
idUsuário	Número sequencial do usuário	<i>Integer</i>	Sim	Sim
nm_usuario	Nome do usuário	<i>Varchar</i>	Não	Não
ds_login	Nome do usuário no aplicativo	<i>Varchar</i>	Não	Não
ds_senha	Senha para acesso do aplicativo	<i>Varchar</i>	Não	Não
ds_email	Email do usuário	<i>Varchar</i>	Não	Não

Abastecimento				
Armazena os dados dos abastecimentos dos automóveis				
Campo	Descrição	Tipo	Chave primária	Chave estrangeira
id_Abastecimento	Número sequencial do abastecimento	<i>Integer</i>	Sim	Não
nr_valor_litro	Valor do litro do combustível	<i>Float</i>	Não	Não
nr_valor_total	Valor litro total abastecido	<i>Float</i>	Não	Não
dt_data	Data do abastecimento	<i>Date</i>	Não	Não
nr_tipo_pagamento	Número tipo pagamento	<i>Integer</i>	Não	Não

Despesa				
Armazena as despesas cadastradas no sistema				
Campo	Descrição	Tipo	Chave primária	Chave estrangeira
idDespesa	Número sequencial da despesa	<i>Integer</i>	Sim	Não
nr_tipo_despesa	Número tipo despesa	<i>Integer</i>	Não	Não
dt_data	Data da despesa	<i>Date</i>	Não	Não
nr_valor	Valor despesa	<i>Float</i>	Não	Não
nr_tipo_pagamento	Tipo pagamento	<i>Integer</i>	Não	Não
ds_local	Descrição local	<i>Varchar</i>	Não	Não

Manutenção				
Armazena as manutenções cadastradas para os automóveis				
Campo	Descrição	Tipo	Chave primária	Chave estrangeira
idManutencao	Número sequencial da manutenção	<i>Integer</i>	Sim	Não
nr_tipo_manutencao	Número tipo manutenção	<i>Integer</i>	Não	Não
dt_data	Data da manutenção	<i>Datetime</i>	Não	Não
nr_valor	Valor manutenção	<i>Float</i>	Não	Não
nr_tipo_pagamento	Tipo pagamento	<i>Integer</i>	Não	Não