UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ERICK CRISTIANO SZULHA

RENAN FERNANDO SIQUEIRA

SOLUÇÃO WEB E MOBILE PARA GESTÃO DE FROTA

TRABALHO DE DIPLOMAÇÃO

PONTA GROSSA

2015

ERICK CRISTIANO SZULHA

RENAN FERNANDO SIQUEIRA

SOLUÇÃO WEB E MOBILE PARA GESTÃO DE FROTA

Trabalho de Diplomação apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná.

Orientadora: Profª Drª Simone de Almeida

PONTA GROSSA

2015

TERMO DE APROVAÇÃO
SOLUÇÃO WEB E MOBILE PARA GESTÃO DE FROTA

por

ERICK CRISTIANO SZULHA
RENAN FERNANDO SIQUEIRA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 09 de novembro de 2015 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Simone de Almeida
Prof.(a) Orientador(a)

_____
Luiz Rafael Schmitke
Membro titular

_____
Richard Duarte Ribeiro
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

AGRADECIMENTOS

A Deus, pela saúde, perseverança e força dada para superarmos os desafios e dificuldades.

Aos nossos pais, pelo amor, incentivo e apoio incondicional.

A Universidade Tecnológica Federal do Paraná, representado por sua diretoria e corpo docente, por nos proporcionar um ensino de referência nacional.

A nossa orientadora e coordenadora Simone de Almeida, que sempre esteve disposta a nos orientar e aconselhar com tanta dedicação.

E a todos os demais, familiares e amigos, que fizeram parte de nossa formação, o nosso muito obrigado.

RESUMO

SIQUEIRA, Renan F.; SZULHA, Erick C. **Solução Web e Mobile para Gestão de Frota:** 2015. 62 páginas. Trabalho de Conclusão de Curso Tecnologia em Análise e Desenvolvimento de Sistemas - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2015.

Devido à grande malha rodoviária existente em nosso país e buscando oferecer maior segurança aos motoristas e gerenciamento da frota, desenvolveu-se um sistema que visa obter maior controle de frota por meio da utilização de ferramentas de localização. O sistema tem o intuito de facilitar as tomadas de decisões de forma clara utilizando recursos visuais com dados provenientes das aplicações *mobile* presentes em cada elemento da frota a ser gerenciada. Os dados de localização e eventos registrados podem ser coletados automaticamente e/ou manualmente pelo motorista operante no veículo e enviadas via *Web-services* para a aplicação Web. Tais módulos foram aplicados em uma equipe de logística já existente para acompanhamento de resultados e ganhos na visualização de indicadores de economia. Devido ao acompanhamento detalhado do percurso, foi possível identificar redução de quilometragem rodada, gerando economia no consumo de combustível e redução do tempo de entregas, oferecendo mais qualidade no processo logístico da empresa.

Palavras-chave: Gerenciamento de Frota. Geolocalização. Mobile. Malha Rodoviária.

ABSTRACT

SIQUEIRA, Renan F.; SZULHA, Erick C. **Web and Mobile Solution for Fleet Management:** 2015. 62 pages. TD – System Analyses and Development Technology, Federal Technological University of Paraná. Ponta Grossa, 2015.

Due to the large existing road network in our country and seeking to provide greater security for drivers and fleet management, a system was developed that seeks greater control of fleet through the use of location tools. The system aims to facilitate the decision taking in a more clearly way using visuals resources with data from mobile applications present on each element of the fleet to be managed. The location data and recorded events are collected automatically and manually by the driver operating the vehicle and sent by Web-services to a Web application. These modules have been applied in a logistics team that already exists for monitoring results and gains in economy indicators display. Due to the detailed tracking of the route it was possible to identify reduction the mileage driven, reducing fuel consumption and decreasing delivery time by offering more quality in the logistics process of the company.

Keywords: *Fleet management. Geolocation. Mobile. Highway Network.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Interface de Programação de Aplicações
CMD	Terminal do Windows
CPF	Cadastro de Pessoa Física
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
GPS	Sistema de posicionamento global
HDPI	Alta densidade
HTML	Linguagem de Marcação de Hipertexto
HTTP	Protocolo de Transferência de Hipertexto
IDE	*Integrated Development Environment*
JDK	*Java Development Kit*
LDPI	Baixa densidade
MDPI	Média densidade
MVC	*Model-View-Controller*
PHP	Pré-processador de hipertexto
SHA-1	Algoritmo de Segurança HASH 1
SQL	*Structured Query Language*
TKU	Toneladas por quilometro útil
XML	*Extensible Markup Language*
WWW	*World Wide Web*

LISTA DE ACRONIMOS

BOOT	Inicialização do dispositivo
IMEI	*International Mobile Equipment Identity*
JSON	*JavaScript Object Notation*
MAC	*Media Access Control*
TAC	*Type Allocation Code*
GEO-IP	*Geographical Internet Protocol*

SUMÁRIO

1 INTRODUÇÃO

Devido a importância do meio rodoviário no país, cada vez mais buscam-se melhorias para o controle da frota de caminhões e também a segurança de motoristas envolvidos nesse ramo. No Brasil, o transporte de produtos é realizado em sua maioria pela malha rodoviária, responsável por quase 63% do TKU (toneladas por quilometro útil) movimentado no país (ILOS, 2008).

Porém, com uma malha tão extensa, o governo tem dificuldades na manutenção periódica dessas estradas. Um estudo recente mostra que o estado geral das rodovias do país é deficiente. Quase 60% do trecho avaliado[1] foi considerado em mau estado, com problemas principalmente na geometria da via e na sinalização, além da má conservação da pavimentação (CNT, 2010).

Este trabalho destina-se a aplicar um monitoramento, por meio de um aplicativo para celular, através de uma interface agradável ao motorista que trará paralelamente ao monitoramento, segurança ao seu percurso em casos de sinistros ou problemas diversos.

1.1 OBJETIVOS

Os objetivos do trabalho estão divididos em geral e específicos que serão descritos respectivamente nas subseções 1.1.1 e 1.1.2.

1.1.1 OBJETIVO GERAL

Desenvolver uma solução móvel para monitoramento e controle de frota de caminhões, que visa obter, por parte das transportadoras, um gerenciamento do percurso do motorista.

---

[1] A avaliação da CNT levou em consideração quase 91 mil quilômetros de estradas, que correspondem à extensão de toda a rede federal pavimentada e às principais rodovias estaduais.

1.1.2 OBJETIVOS ESPECÍFICOS

- Identificar os requisitos para o desenvolvimento do sistema;
- Modelar o banco de dados dos sistemas Web e *mobile;*
- Desenvolver a aplicação móvel;
- Desenvolver a interface Web de controle e monitoramento.

1.2 JUSTIFICATIVA DO TRABALHO

Atualmente existem produtos semelhantes no mercado, porém com foco diferenciado[2]. Em sua maioria, as ferramentas utilizam um dispositivo de Sistema de Posicionamento Global (GPS) acoplado ao veículo com o qual coletam seu posicionamento na malha rodoviária e por meio de uma interface Web mostram o deslocamento percorrido à central que possui também, mecanismos de bloqueio veicular. Essa aplicação demanda maior investimento nos veículos e uma quantidade maior de pessoas para o constante monitoramento da frota.

Como diferencial, buscou-se minimizar o custo de implantação e aumentar a agilidade na operação do sistema, tanto gerencial quanto operacional pelos condutores.

Com a aplicação presente no *smartphone* do motorista condutor é possível identificar a localização das cargas, monitorar as viagens, auditar paradas, velocidades e percurso realizado, além de notificar o motorista para o cumprimento da lei nº 12.619, de 30 de abril de 2012 que define intervalos obrigatórios de descanso e pernoite durante a realização das viagens, evitando o desgaste e zelando pela saúde dos motoristas.

---

[2] Truckpad: Aplicativo que através do GPS do *smartphone* informa a rota percorrida pelo motorista para identificar motoristas para a sua carga e vice-versa.

1.3 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está dividido em cinco capítulos, considerando este já apresentado. O Capítulo dois apresenta a revisão bibliográfica do trabalho e apresentação das ferramentas utilizadas no desenvolvimento. No Capítulo três mostra-se o desenvolvimento e suas interfaces, no Capítulo quatro são apresentados os resultados obtidos baseados em testes reais e no quinto Capítulo são apresentadas as considerações finais do trabalho e possíveis trabalhos futuros associados ao tema.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são abordados os tópicos relevantes para o embasamento necessário dos conteúdos deste trabalho. A seção 2.1 discorre sobre o sistema operacional Android; a seção 2.2 retrata *International Mobile Equipment Identity* (IMEI); a seção 2.3 aborda os conceitos utilizados de Geomapeamento, a seção 2.3.1 explica Geomapeamento via GPS; a seção 2.3.2 explica Geomapeamento via Geo-IP.

A seção 2.4 se refere a PHP; a seção 2.5 aborda os Banco de Dados utilizados no desenvolvimento do sistema; a seção 2.5.1 discorre sobre POSTGRESQL; a seção 2.5.2 explica o SQLITE; a seção 2.6 comenta sobre o uso do *Framework* Codeigniter; a seção 2.7 relata o embasamento jurídico; do qual foram extraídas algumas regras de negócio e por fim são apresentadas as considerações finais do capítulo na seção 2.8.

2.1 ANDROID

A plataforma de desenvolvimento móvel escolhida para a realização deste trabalho foi o Android. Uma plataforma criada para dispositivos móveis que é baseado em Linux. Desenvolvido inicialmente pela *Open Handset Alliance* e mais tarde adquirido pelo Google (ED BURNETTE, 2009).

Com aproximadamente 81% do mercado mundial de *smartphones* em 2014, de acordo com IDC (2015), o Android se mantém nos últimos anos como plataforma móvel mais utilizada no mundo, conforme demonstra a Tabela 1.

Tabela 1 – Comparativo de mercado de plataformas móveis

Plataforma	Unidades 2014	Mercado 2014	Unidades 2013	Mercado 2013
Android	1.059.3	81,5%	802,2	78,7%
iOS	192,7	14,8	153,4	15,1%
Windows Phone	34,9	2,7%	33,5	3,3%
Black Berry	5,8	0,4%	19,2	1,9%
Outros	7,7	0,6%	2,3	0,2%
Total	**1.300.4**	**100,0%**	**1.018.7**	**100,0%**

FONTE: Adaptado de IDC (2015)

Isso motivou o desenvolvimento nessa plataforma, já que sua abrangência e facilidades na busca de conteúdos para criação são consideravelmente maiores e mais acessíveis, além de ser inteiramente gratuito.

A licença de desenvolvedor Android é necessária apenas para a publicação do aplicativo na *PlayStore* e possui uma taxa única de 25 dólares por desenvolvedor (ANDROID DEVELOPER, 2015). Além disso, não há limites de aplicações publicadas na conta.

2.2 IMEI

Para identificar unicamente os dispositivos, foi utilizado o código *International Mobile Equipment Identity* (IMEI) que é um número de identificação global e único para cada telefone celular. O número consiste em quatro grupos contendo 15 dígitos (14 dígitos + 1 dígito de verificação), que segue o padrão: AAAAAAAA-BBBBBB-C, onde:

- A: Código Tipo de Alocação (TAC) fornecida pela Autoridade de Certificação. Ele é codificado em oito dígitos, os dois primeiros dígitos do código de país onde o dispositivo móvel foi registado.
- B: Número de série de fabricação do produto, codificado ao longo de seis números.
- C: Dígito de verificação.

Para identificar este código, basta digitar *#06# no telefone e simular uma chamada ou verificar uma possível etiqueta contida no interior do aparelho e até mesmo na embalagem do produto.

2.3 SERVIÇO DE LOCALIZAÇÃO

Apesar de ter sido desenvolvido há quatro décadas, o sistema de posicionamento global tornou-se popular apenas com a chegada dos *smartphones* (FERRARI, 2014). As funcionalidades que a obtenção de localização do usuário pode oferecer são incrivelmente úteis, entre elas, podem-se destacar atividades do dia a dia que já se tornaram muito mais fáceis com essa tecnologia.

A obtenção dessas localizações pode ser realizada de duas maneiras distintas, via GPS ou via Geo-IP.

2.3.1 GPS

É um sistema de posicionamento por satélite que fornece ao dispositivo receptor a sua posição geográfica, altitude, velocidade e fuso horário. É a maneira mais precisa e também mais utilizada pelos dispositivos de rastreamento. Seu uso é livre e gratuito para os usuários, por isso, sua funcionalidade é facilmente encontrada em *smartphones* e dispositivos veiculares, porém, é um recurso que demanda uma quantidade significativa dos níveis de bateria do dispositivo.

> Sistema de rádio navegação baseado no espaço que transmite pulsos de navegação altamente precisos aos usuários na Terra. [...] Um receptor GPS operado por um usuário na Terra mede o tempo que leva sinais de rádio para viajar de quatro ou mais satélites à sua localização, calcula a distância de cada satélite e, a partir deste cálculo determina longitude, latitude e altitude do usuário. (ENCICLOPÉDIA BRITÂNICA, 2012).

Cada um dos 24 satélites componentes do sistema transmite continuamente um sinal de rádio que contém informações sobre a sua posição orbital, vinculado à um referencial geodésico, e o tempo marcado por seu relógio atômico interno. (ZANOTTA; CAPPELLETTO; MATSUOKA, 2011).

2.3.2 Geo-IP

Geographical Internet Protocol (GeoIP) refere-se ao método de localização de um terminal através da identificação do endereço de IP. Embora GeoIP possa identificar a localização de um terminal para uma cidade, requer o uso de um banco de dados para uma identificação mais precisa.

Baseada em *IP-Geolocation* é o mapeamento de um endereço IP ou endereço *Media Access Control* (MAC) para a localização geográfica de um dispositivo de computação ou dispositivo móvel à Internet (IPLOCATION, 2015).

2.4 PHP

A linguagem de programação *Hypertext Preprocessor* (PHP) é uma das linguagens mais utilizadas no mundo de acordo com o índice Tiobe (2015), o que facilitou a escolha dessa linguagem para o desenvolvimento do módulo Web. Seu funcionamento consiste em processar o algoritmo no servidor e gerar um código HTML que é então enviado ao cliente, que recebe os resultados da execução desse *script.*

Além de ser uma linguagem de fácil aprendizado e vasto material de suporte, o PHP também é facilmente integrado ao banco de dados Postgres, que foi utilizado em conjunto na aplicação.

2.5 BANCO DE DADOS

O sistema gerenciador de banco de dados está presente no cotidiano da sociedade. Estes são utilizados de formas simples e complexas, desde uma agenda que armazena nomes, endereços e telefones de diversas pessoas até sistemas bancários ou de caixas eletrônicos (ALECRIM, 2005).

Para o desenvolvimento dessa aplicação, foi necessária a criação de duas bases de dados distintas, utilizando o Postgres para o ambiente Web e o SQLITE para cada dispositivo móvel que possui a aplicação. O SQLITE é necessário na aplicação *mobile* porque o aplicativo pode não estar constantemente conectado a uma rede de dados para a comunicação imediata com os servidores, por isso, as ações e eventos gerados são armazenados localmente até que o acesso a uma rede de Internet seja reestabelecido para que a sincronização possa ser realizada.

2.5.1 POSTGRES

Postgres é um sistema de banco de dados *open source* objeto-relacional. Possui mais de 15 anos de desenvolvimento e uma arquitetura comprovada que ganhou forte reputação de confiabilidade, integridade de dados e correção (POSTGRES, 2015).

É o sucessor do sistema de banco de dados relacional INGRES. Os principais objetivos do projeto do sistema são a proporcionar um melhor apoio para objetos complexos, fornecer extensibilidade para tipos de dados, operadores e métodos de acesso, e também fornecer facilidades para bancos de dados ativos. (MICHAEL STONEBRAKER, 1986).

2.5.2 SQLITE

Uma poderosa biblioteca de banco de dados baseado em SQL (*Structured Query Language*) que atua como um “mini SGBD”, capaz de controlar diversos bancos de dados que podem conter diversas tabelas. O desenvolvedor pode criar o banco de dados e as tabelas, assim como manipular seus dados através dos comandos DDL (*Data Definition Language*) e DML (*Data Manipulation Language*) do SQL padrão (DEVMEDIA, 2010).

SQLITE é uma biblioteca de *software*, sem servidor, de simples configuração, que é um motor de banco de dados SQL transacional autossuficiente. Diferentemente da maioria dos outros bancos de dados SQL, o SQLite não tem um processo servidor separado, ele lê e escreve diretamente em arquivos de disco comuns. Um banco de dados SQL completo com várias tabelas, índices, *triggers* e *views*, está contido em um único arquivo em disco. O formato do arquivo de banco de dados é multiplataforma (SQLITE, 2015).

2.6 FRAMEWORK CODEIGNITER

O *CodeIgniter* é um *framework*[3] para desenvolvimento de aplicações PHP. Considerado um *toolkit*[4], ou seja, uma caixa de ferramentas cujo objetivo é permitir o desenvolvimento de aplicações muito mais rápido do que poderíamos fazer sem a utilização de um *framework*. Ele contém um conjunto de bibliotecas para tarefas

[3] Framework: abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.
[4] Toolkit: conjunto de ferramentas de auxílio no desenvolvimento.

comuns, interfaces simples e uma estrutura lógica para acesso a estas bibliotecas (DEVMEDIA, 2013).

Ele é um *framework* PHP poderoso, com um baixo custo de armazenamento, construído para os desenvolvedores que precisam de um *kit* de ferramentas para criar aplicações Web com recursos completos (CODEIGNITER, 2015).

2.7 EMBASAMENTO JURÍDICO

O aplicativo móvel auxilia o condutor ao notificar-lhe nos momentos corretos os períodos de descanso que deverão ser realizados no decorrer do percurso.

O Art. 235-D da lei nº 12.619, de 30 de abril de 2012, regula as viagens de longa distância, assim consideradas aquelas em que o motorista profissional permanece fora da base da empresa, matriz ou filial e de sua residência por mais de 24 (vinte e quatro) horas. O mesmo determina:

I - intervalo mínimo de 30 (trinta) minutos para descanso a cada 4 (quatro) horas de tempo ininterrupto de direção, podendo ser fracionados o tempo de direção e o de intervalo de descanso, desde que não completadas as 4 (quatro) horas ininterruptas de direção;

II - intervalo mínimo de 1 (uma) hora para refeição, podendo coincidir ou não com o intervalo de descanso do inciso I.

2.8 CONSIDERAÇÕES DO CAPÍTULO

Unindo tecnologias de fácil acesso é possível criar uma aplicação útil e eficiente. Juntamente ao *smartphone*, os motoristas e transportadoras possuem uma importante ferramenta de trabalho. No próximo capítulo, é demonstrado como todas essas ferramentas foram utilizadas no experimento prático desta proposta.

3 DESENVOLVIMENTO DO *SOFTWARE*

Este capítulo está dividido em cinco seções principais, devido a necessidade do desenvolvimento em duas linguagens distintas de programação. A seção 3.1 apresenta a versão Web do desenvolvimento, descrevendo a API do Google *Maps* e abordando o *Web-service*. A seção 3.2 discorre sobre o desenvolvimento *mobile*, com o uso da API do Google Maps, *Web-service* e *StartBoot*.

A seção 3.3 apresenta a interface Web, discorrendo sobre o seu acesso, as opções de menu existentes e demonstra a parametrização do sistema e a forma de gerenciamento da frota. A seção 3.4 demonstra a interface *mobile*, discorrendo sobre o seu acesso, as opções de menu, a geração de eventos, os relatórios disponíveis, além de demonstrar o processo de sincronização das aplicações Web e *mobile*. Por fim são apresentadas as considerações finais do capítulo na seção 3.5.

3.1 DESENVOLVIMENTO WEB

Na Figura 1 pode-se visualizar o fluxo da aplicação de forma geral: cadastro, sincronização com a aplicação *mobile*, geração de *logs* e sincronização com a aplicação Web.

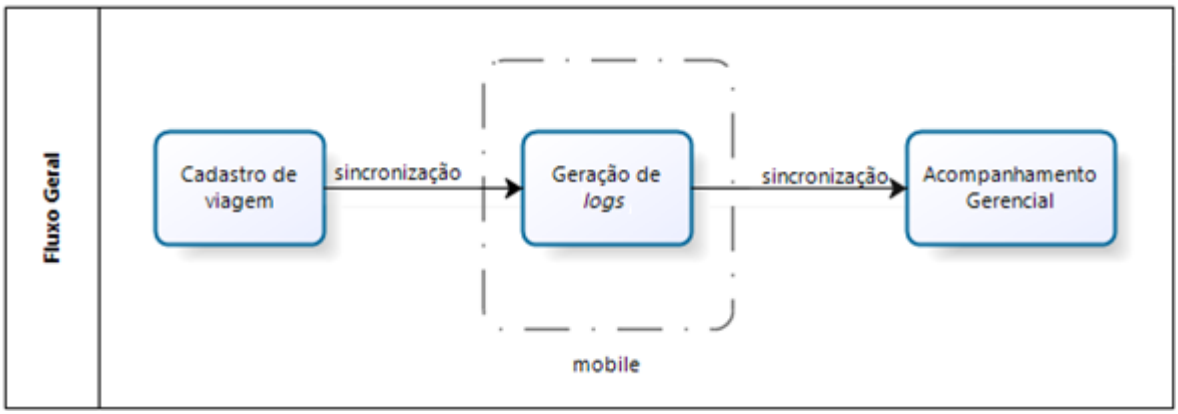


Figura 1 – Fluxograma do fluxo geral do sistema
FONTE: Autoria própria

Armazenada em um servidor locado na nuvem, a aplicação Web foi desenvolvida em PHP e utiliza *framework* CodeIgniter e o padrão *Model-View-Controller* (MVC).

O *framework* CodeIgniter requer algumas definições iniciais para o funcionamento padrão. As definições aplicadas nesse projeto foram:

- Arquivo “autoload.php” (application/config/autoload.php):
  $autoload['libraries'] = array('database','session','form_validation');
  $autoload['helper'] = array('form','html','url');
  As definições inseridas no arquivo autoload.php são necessárias para que bibliotecas sejam iniciadas automaticamente ao iniciar com a aplicação, não requerendo uma chamada isolada em cada módulo do sistema. No caso deste trabalho, necessita-se que as bibliotecas de banco de dados, controle de sessão e validações de formulário sejam iniciadas dessa maneira.

- Arquivo “database.php” (application/config/database.php):
  $db['default']['hostname'] = 'pgsql.pmob.com.br;
  $db['default']['username'] = 'renansiqueira';
  $db['default']['password'] = 'xxxxxxxxx';
  $db['default']['database'] = 'renansiqueira';
  $db['default']['dbdriver'] = 'postgre';
  Definições necessárias para que a conexão com o banco de dados seja realizada. Além do *hostname*, usuário e senha, também é necessário informar o nome do *database* da aplicação e o *driver* da conexão, que neste caso é o “*postgre”* – *driver* do banco de dados PostgresSQL.

- Arquivo “routes.php” (application/config/routes.php):
  $route['default_controller'] = ''login/login'';
  Definido o *controller* que será iniciado por padrão ao acessar a página, nesse caso, o *login*.

O fluxo da aplicação Web é bastante simples, a interface do usuário, conecta-se localmente ao banco de dados Postgres e aos *Web-services* de entrada e saída, que se comunicam com as aplicações *mobile*, conforme ilustrado na Figura 2.

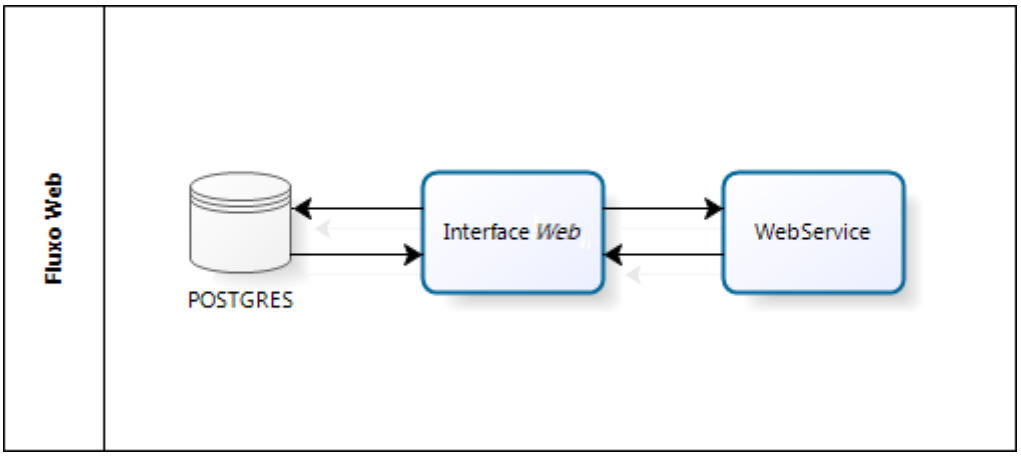


Figura 2 – Fluxo geral da aplicação Web
FONTE: Autoria própria

De maneira simplificada, a interface Web possui os cadastros básicos da aplicação que são armazenados no banco de dados. Os *Web-services* são as portas de entrada e saída dos dados comunicados entre Web e *mobile*. Esses dados são tratados pela interface e gravados na base de dados.

3.1.1 Estrutura do Banco de Dados Web

Composta por 14 tabelas relacionadas, a estrutura foi criada no gerenciador de banco de dados Postgres, que além de gratuito e *open-source* possui grandes vantagens de desempenho e suporte em fóruns de tecnologia. Todos os dados do sistema (Web e *mobile*) são centralizados no banco de dados Web, dessa forma, as aplicações *mobiles* atuam como coletores de dados até que a sincronia com o sistema central seja realizada. A estrutura pode ser visualizada na Figura 3.

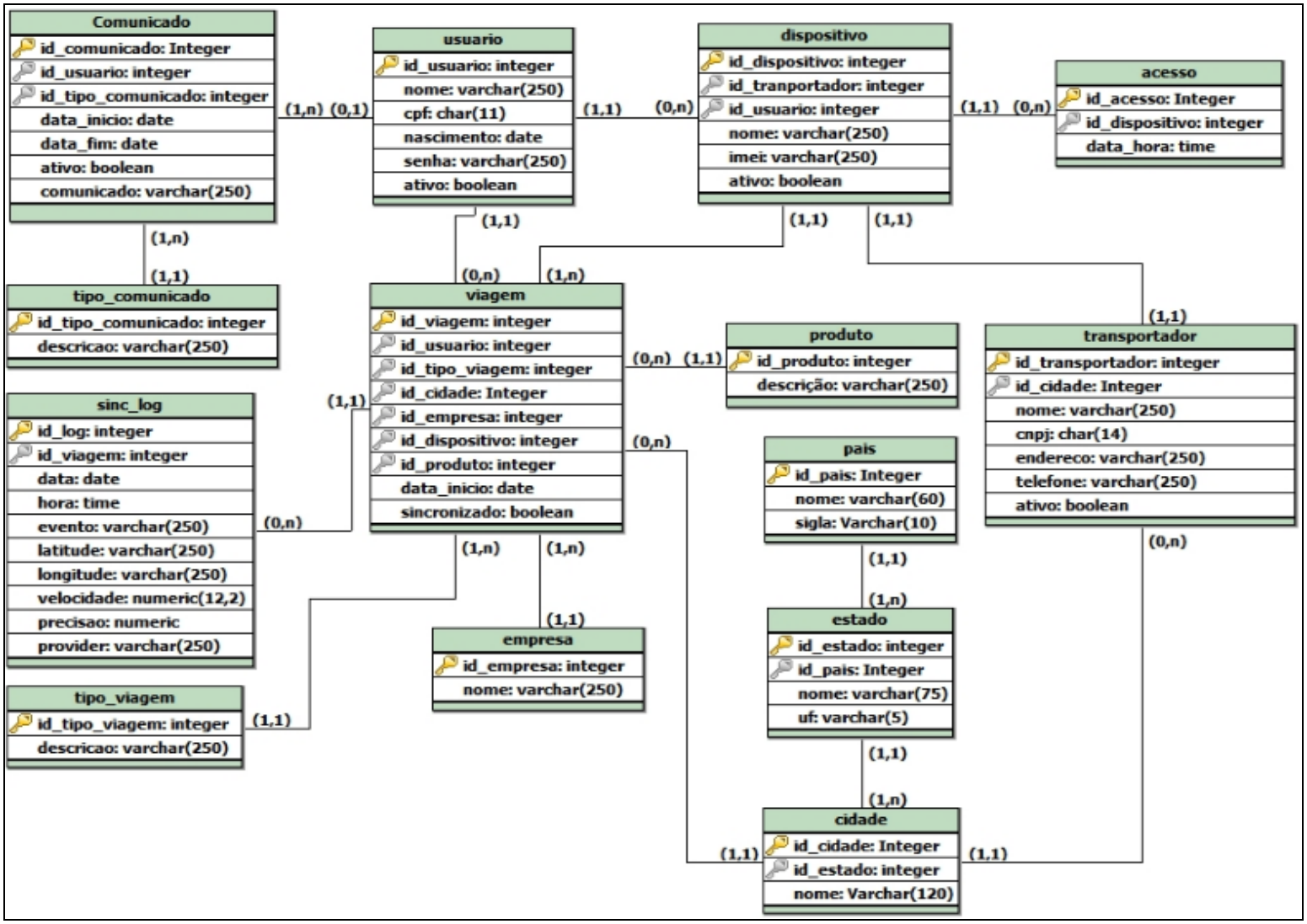


Figura 3 – Estrutura do banco de dados da aplicação Web
FONTE: Autoria própria

No centro da estrutura, a tabela “viagem” agrupa os dados que compõe a viagem. São eles: tipo de viagem, cidade e estado, empresa de destino, dispositivo (caminhão a realizar a viagem), produto transportado e a data de início da viagem. Todos esses dados são provenientes de suas próprias tabelas para que possa ser possível criar relatórios precisos com tais índices.

A tabela “sinc_log” armazena os eventos registrados (manuais e automáticos) durante o percurso da viagem. Para cada evento, é armazenada a sua data e hora, descrição do evento, latitude, longitude, velocidade e precisão (em metros) do GPS. A coluna “*provider*” informa qual foi o método utilizado pela aplicação em obter a localização GPS no momento do registro do evento, que podem ser “GPS” ou “*Network”*.

A tabela “usuario” armazena dados dos usuários do módulo Web, onde CPF também é utilizado como identificador para se *logar* no sistema.

A tabela “dispositivo”, por sua vez, guarda dados dos dispositivos que representam os caminhões da frota.

A tabela “acesso” registra todas as tentativas de sincronização realizadas pelos dispositivos *mobiles* a fim de auditar tal informação no horário de recebimento e envio dos dados de percurso.

3.1.2 API do Google Maps

A utilização da API do Google *Maps* para a demonstração geográfica da localização dos caminhões é apenas permitida para ambientes não corporativos que possuam no máximo 2500 requisições de geocodificação diárias. O uso da API para empresas requer um licenciamento e consequentemente uma cobrança pelos recursos adicionais do serviço.

Na aplicação Web, essa ferramenta foi utilizada para demonstrar no mapa a localização e a rota obtida pelo rastreamento dos dispositivos. Para o funcionamento, basta incorporar ao código a biblioteca necessária, conforme ilustrado na Figura 4:

```
<script src="http://maps.googleapis.com/maps/api/js"></script>
```

Figura 4 – Biblioteca para funcionamento da API do Google Maps
FONTE: Autoria própria

Com isso, as funcionalidades de inserção de marcadores e traços, já podem ser inseridas via *Javascript* na página, bastando inicializar a geração do mapa em uma *div*[5] previamente criada com o comando apresentado na Figura 5.

```
<script>
  function initialize() {
    var mapCanvas = document.getElementById('map-canvas');
    var mapOptions = {
      center: new google.maps.LatLng(44.5403, -78.5463),
      zoom: 8,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    }
    var map = new google.maps.Map(mapCanvas, mapOptions);
  }
</script>
```

Figura 5 – Script de inicialização do mapa
FONTE: Google (2015)

Onde:

- *map-canvas* é o nome da *div* que receberá o mapa
- *center* é o parâmetro para informar a localização de onde o mapa será centralizado inicialmente
- *zoom* é o parâmetro para informar em qual nível de zoom o mapa será iniciado (quanto maior, maior o zoom)
- *mapTypeId* é o parâmetro para definir que tipo de mapa será carregado, que pode ser:
    - google.maps.MapTypeId.ROADMAP: Mapas rodoviários padrões;

[5] div ou tag define uma divisão ou uma seção em HTML. É usada para agrupar elementos para formata-los com CSS (W3Schools, 2015).

- google.maps.MapTypeId.SATELLITE: Imagens de satélite do *Google Earth*;
- google.maps.MapTypeId.HYBRID: Mistura de visualização dos mapas rodoviários e das imagens de satélite;
- google.maps.MapTypeId.TERRAIN: Mapa físico com base nas informações do térreo.

A inserção de “marcadores” no mapa é feita por meio do comando em *Javascript* demonstrado na Figura 6:

```
var location = new google.maps.LatLng(lati,longi)
$('#map_canvas').gmap('addMarker', {
    'position': location,
    icon: new google.maps.MarkerImage('../../public/imagens/icon_map.png')
}).click(function() {
    $('#map_canvas').gmap('openInfoWindow', {'content': msg}, this);
});
```

Figura 6 – Inserção de marcador em mapa
FONTE: Autoria própria

Por sua vez, a inserção de linhas que definem o trajeto percorrido durante a viagem é realizada a partir do comando da Figura 7.

```
$('#map_geral').gmap('addShape', 'Polyline', {
        'path': cordenadas,
        'strokeColor': "#4265fb",
        'strokeOpacity': 0.8,
        'strokeWeight': 4,
        'fillColor': "#001261",
        'fillOpacity': 0.55
    }
);
```

Figura 7 – Inserção de linhas em mapa
FONTE: Autoria própria

Onde a variável “coordenadas” é um *array* de pontos de localização. Dessa forma, cria-se uma linha entre a – b, b – c, c – d, e assim por diante até o último

ponto de localização gravado. Entre os parâmetros da função, pode-se definir a cor da linha e sombra, opacidade e tamanho.

3.1.3 Web-services

Foram criados dois métodos para a comunicação de dados entre a aplicação web e *mobile*. São arquivos em PHP que recebem uma requisição *POST* enviada pelo aplicativo e retornam uma *string* em formato *JavaScript Object Notation* (JSON).

Os métodos para a comunicação são:

- Método "*webservice*": Responsável por localizar e enviar as viagens pendentes por motorista. São recebidos os parâmetros de "IMEI" e "chave_acesso" (chave de segurança definido em ambas as aplicações). Com o IMEI, consulta-se o banco de dados em busca de viagens pendentes para o motorista, como mostra a Figura 8.

```
$imei = $_POST['IMEI'];
$con = pg_connect("host=localhost port=5432 dbname=renansiqueira user=renansiqueira password=xxxxxx");
$sql = "SELECT *
        FROM frota.view_viagem where imei = ".$imei."
        AND id_viagem not in
        (SELECT id_viagem
         FROM frota.sinc_log l
         WHERE l.evento = 'VIAGEM FINALIZADA' )";
```

Figura 8 – Consulta de viagens por IMEI
FONTE: Autoria própria

Com o tratamento dos registros e a verificação de comunicados gerais cadastrados, o resultado é retornado em formato JSON, como apresentado na Figura 9:

```php
$sql = "select * from frota.view_status";
$status = pg_query($con, $sql);
$status = pg_fetch_assoc($status);
$geral[0]['comunicado_geral'] = ($status['comunicado']);
$geral[0]['tipo_aviso'] = $status['id_tipo_aviso'];

if(is_array($geral) && is_array($viagens))
    $arrayRetorno[] = array_merge($geral, $viagens);
else if (is_array($geral))
    $arrayRetorno[] = $geral;

echo json_encode($arrayRetorno);
```

Figura 9 – Retorno da consulta por JSON
FONTE: Autoria própria

- Método “webserviceExport”: este *Web-service* por sua vez tem a tarefa de receber as viagens do dispositivo e atualizar os logs gerados no banco de dados da aplicação Web para acompanhamento gerencial. Na Figura 10 é demonstrado o recebimento e validação da viagem recebida pelo aplicativo.

```php
$chave = $_POST['chave_acesso'];
$con = pg_connect("host=localhost
                   port=5432
                   dbname=renansiqueira
                   user=renansiqueira
                   password=xxxxxx");
if($chave == '07961315977'){
    $imei = $_POST['IMEI'];
    $data = $_POST;
    pg_query($con, "BEGIN;");
    foreach($data as $viagem){
        $res = explode(";", $viagem);
        $codViagem = $res[0];
        $sql = "SELECT id_viagem
                FROM frota.view_viagem
                WHERE imei = '{$imei}' and id_viagem = {$codViagem}";
        $result = pg_query($con, $sql);
        //...
```

Figura 10 – Recebimento e validação da viagem recebida
FONTE: Autoria própria

Como a comunicação é realizada por *strings* de caracteres, além dos parâmetros de “chave_acesso” e “IMEI”, também são enviadas as viagens realizadas separadas por “;” (ponto e vírgula). Na Figura 11 é demonstrado o tratamento dos dados recebidos e a gravação no banco de dados.

```
    //...
    if(pg_num_rows($result) > 0){
        $viagemSave['INICIO'] = $res[1]; $viagemSave['FIM'] = $res[2];
        $logs = explode("|", $res[3]);
        foreach($logs as $log){
            $logIn = explode('&', $log);
            $dataTemp = explode("-", $logIn[0]);
            $data = date('Y-m-d', mktime(0, 0, 0, $dataTemp[1], $dataTemp[0], $dataTemp[2]));
            $logsSave['DATA']       = $data;
            $logsSave['HORA']       = $logIn[1];
            $logsSave['EVENTO']     = $logIn[2];
            $logsSave['LATITUDE']   = ($logIn[3]!="")?$logIn[3]:"null";
            $logsSave['LONGITUDE']  = ($logIn[4]!="")?$logIn[4]:"null";
            $logsSave['VELOCIDADE'] = ($logIn[5]!="")?$logIn[5]:"null";
            $logsSave['PRECISAO']   = ($logIn[6]!="")?$logIn[6]:"null";
            $logsSave['PROVIDER']   = ($logIn[7]!="")?$logIn[7]:"null";

            $sql = "INSERT INTO frota.sinc_log (id_viagem, data, hora, evento, latitude,
                        longitude, velocidade, precisao, provider)
                    VALUES (".$codViagem.", '".$logsSave['DATA']."',
                            '".$logsSave['HORA']."', '".$logsSave['EVENTO']."',
                            '".$logsSave['LATITUDE']."', '".$logsSave['LONGITUDE']."',
                            ".$logsSave['VELOCIDADE'].", ".$logsSave['PRECISAO'].",
                            '".$logsSave['PROVIDER']."');";
            pg_query($con, $sql);
        }
    }
}
pg_query($con, "COMMIT;");
echo count($data);
```

Figura 11 – Tratamento dos dados e gravação no banco de dados
FONTE: Autoria própria

Em um laço de repetição as viagens são armazenadas na variável “res”, esta que é separada pelo caractere “&” a fim de identificar os dados de cada viagem. O retorno da chamada é a quantidade de viagens comunicadas para que a aplicação *mobile* valide se a requisição foi realizada com sucesso.

3.2 DESENVOLVIMENTO *MOBILE*

A representação da estrutura de fluxo da aplicação *mobile* é demonstrada na Figura 12.

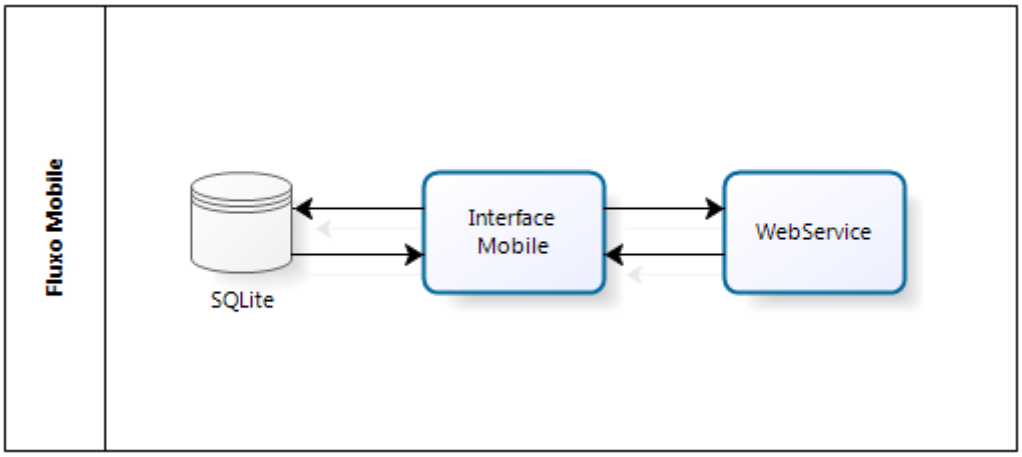


Figura 12 – Fluxo geral da aplicação *mobile*
FONTE: Autoria própria

A interface da aplicação *mobile* se comunica com os *Web-services* e armazena os dados localmente para geração de *logs* dos eventos e suas respectivas localizações. Posteriormente, os dados são recuperados do banco de dados e enviados ao *Web-service* de recebimento da interface Web para gerenciamento e demonstrativo de relatórios.

Utilizando o programa Eclipse[6] para o desenvolvimento do aplicativo, foi necessário abordar o relacionamento entre W*eb-services*, a utilização banco de dados SQLite e integração com serviços de localização do Google.

A estrutura para o desenvolvimento é dividida em dois principais diretórios: **src** e **res**, como na Figura 13. O diretório **src** armazena as classes e *Activities.* Cada *Activity* representa o controle de uma tela no aplicativo, *MainActivity*, *PainelActivity* e *RelatorioActivity* neste caso. No diretório **res**, são inseridos todos os recursos gráficos da aplicação como *views, que são as* telas em *Extensible Markup Language* (XML), imagens e modelos de *layout*.

[6] Eclipse: Ferramenta *Integrated Development Environment* (IDE*)* para desenvolvimento nas linguagens de programação Java, PHP e C/C++.(ECLIPSE, 2015)

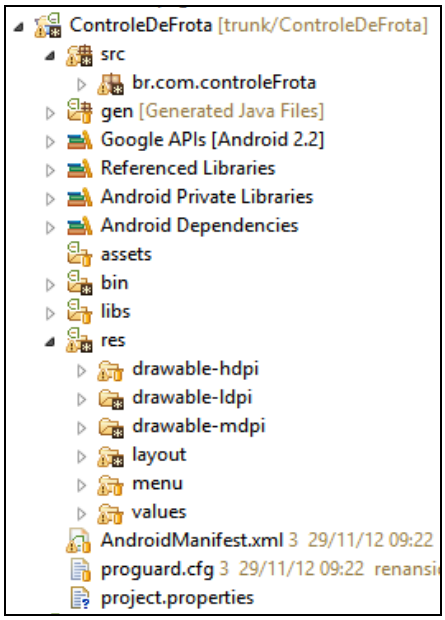


Figura 13 – Diretórios da aplicação mobile
FONTE: Autoria própria

No arquivo AndroidManifest.xml são definidas todas as permissões que o aplicativo requisitará ao dispositivo, tais como acesso à Internet, compartilhamento de localização, acesso de leitura e escrita no cartão de memória, entre outros. Também neste arquivo são identificadas todas as *Activities* da aplicação e a hierarquia entre elas.

É importante ressaltar a organização em que as imagens devem ser incorporadas na estrutura. Elas devem ser replicadas e distribuídas nos diretórios específicos, com o tamanho correspondente para cada nível de densidade que os dispositivos podem possuir. Na Tabela 2, pode-se observar como o Android identifica tais imagens e define qual é a mais apropriada para cada dispositivo.

Tabela 2 – Escala de imagens

Resolução da tela	DPI	Pixel ratio	Tamanho da imagem (pixels)
HDPI	240	1,5	72 x 72
MDPI	160	1.0	48 x 48
LDPI	120	0,75	36 x 36

FONTE: Google (2015)

Nessa aplicação foram utilizados os três níveis de densidade mais comuns: Alta densidade (HDPI), média densidade (MDPI) e baixa densidade (LDPI).

3.2.1 Estrutura do Banco de Dados *Mobile*

Para a possiblidade de trabalhar de maneira *off-line* foi necessária a criação do banco de dados na aplicação *mobile* para armazenar os dados para a posterior sincronização quando a conexão com a Internet for novamente estabelecida. Na Figura 14 é demonstrado, por meio do modelo relacional, a estrutura que é criada isoladamente em cada aplicativo.

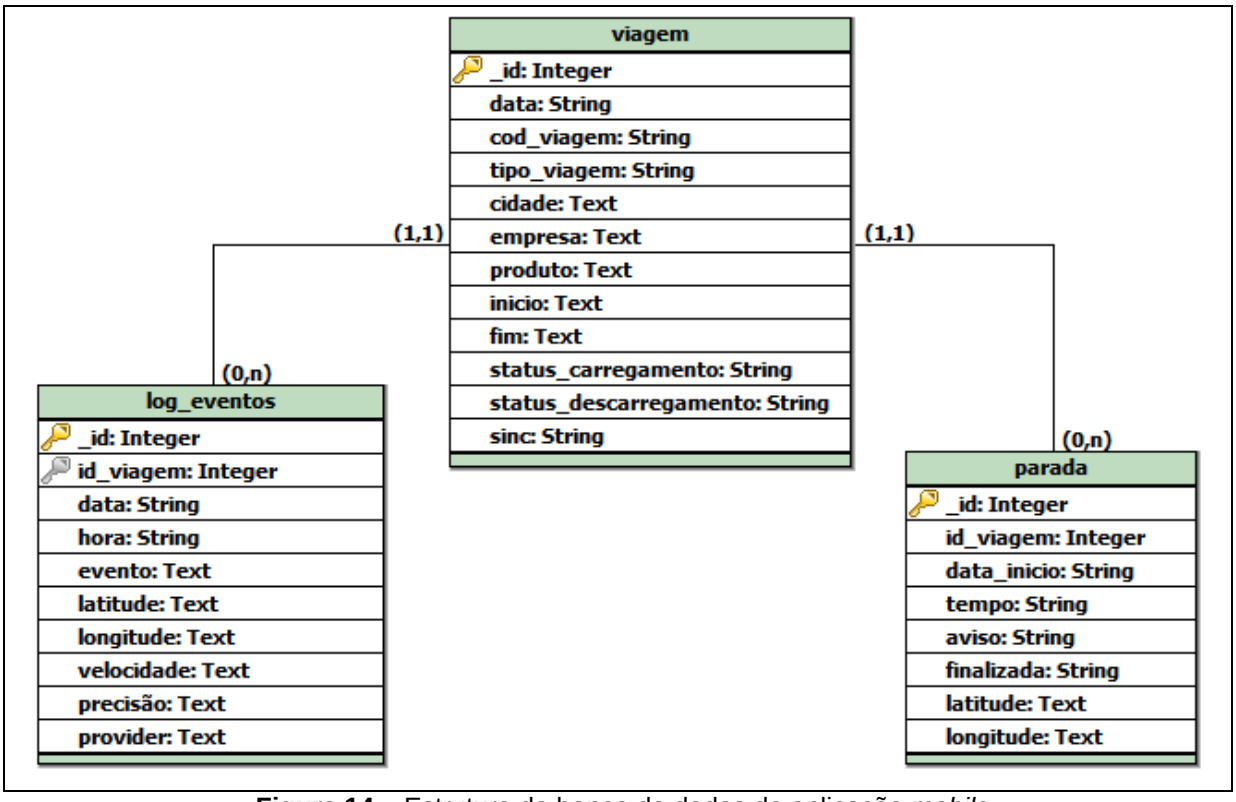


Figura 14 – Estrutura do banco de dados da aplicação *mobile*
FONTE: Autoria própria

Como pode-se observar, os tipos de dados disponíveis são extremamente limitados se comparado aos bancos de dados de alto desempenho, sendo disponíveis apenas os tipos *Integer*, *Real*, *String*, *Text* e *Blob*.

Na tabela “viagem”, são armazenados os dados que são utilizados para a visualização das informações pertinentes a viagem no painel de controle da aplicação. Já na tabela “log_eventos”, são armazenados todos os eventos registrados, como localizações e os registros de chegada e término da viagem. Na tabela “parada” armazenam-se os detalhes das paradas informadas manualmente pelo condutor.

Uma tabela também foi criada para armazenar apenas os comunicados enviados pela aplicação Web, conforme a Figura 15.

Figura 15 – Tabela *mobile* "comunicado"
FONTE: Autoria própria

Como pode ser visto, a tabela não possui nenhum relacionamento com outras tabelas da aplicação, motivo pelo qual os comunicados são inteiramente independentes das viagens e usuários.

3.2.2 API do Google Maps

Para a utilização das bibliotecas necessárias para o funcionamento do serviço de localização, é necessário que se obtenha uma chave de API, disponibilizada pela própria Google de forma gratuita. Para a obtenção dessa chave, deve-se seguir os seguintes passos:

- Localizar o arquivo de certificado de liberação e armazenamento de chaves. Neste caso, o arquivo está alocado em: C:\Users\user_name\.android\debug.keystore
- Usando o Terminal do Windows (CMD) é preciso navegar até a pasta *bin* do *Java Development Kit* (JDK) para ter acesso aos comandos *Keytool*.
- Executar a seguinte linha de comando: keytool -list -v –keystore "C:\Users\user_name\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
- Na saída informada (ilustrada na Figura 16), é possível visualizar o Algoritmo de Segurança HASH 1 (*SHA-1) fingerprint*[7]

[7] SHA-1 fingerprint: É uma pequena sequência de bytes usados para identificar uma longa chave pública.

```
Alias name: androiddebugkey
Creation date: Jan 01, 2013
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 4aa9b300
Valid from: Mon Jan 01 08:04:04 UTC 2013 until: Mon Jan 01 18:04:04 PST 2033
Certificate fingerprints:
MD5:  AE:9F:95:D0:A6:86:89:BC:A8:70:BA:34:FF:6A:AC:F9
SHA1: BB:0D:AC:74:D3:21:E1:43:07:71:9B:62:90:AF:A1:66:6E:44:5D:75
Signature algorithm name: SHA1withRSA
Version: 3
```

Figura 16 – Obtenção da assinatura *SHA-1 fingerprint*
FONTE: Autoria própria

- Acessar Google APIs Console pelo *link*: http://code.google.com/apis/console/
- Será solicitada a criação projeto que será usada para controlar o uso do Google *Maps* API Android.
- Uma listagem de APIs e serviços serão mostrados na janela principal. Entre eles, deve-se selecionar o Google *Maps* API v2 Android. Basta clicar no botão de ativação.
- Na barra de navegação à esquerda, deve-se clicar em API *Access* e posteriormente no botão *Create New Android Key*.
- Na caixa de diálogo, deve ser inserido o *SHA-1 fingerprint* obtido anteriormente concatenado de um ponto e vírgula e em seguida o nome do pacote do aplicativo, como mostra a Figura 17.

```
BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75;br.com.controleFrota
```

Figura 17 – Agrupamento da assinatura SHA1 com o pacote da aplicação
FONTE: Autoria própria

- Feito isso, é exibida a chave para utilização da API (Figura 18).

```
AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0
```

Figura 18 – Chave para utilização da API
FONTE: Autoria própria

- Deve-se então adicionar esta chave no arquivo AndroidManifest.xml como demonstrado na Figura 19.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0"
/>
```

Figura 19 – Definição da chave da API na aplicação
FONTE: Autoria própria

Com essas definições, a aplicação *mobile* obtém permissão e acesso aos recursos de geomapeamento do Google para Android.

3.2.3 Web-service

Como na aplicação Web, também foram criados dois métodos para a comunicação. Ambos enviam uma requisição *POST*, recebem a resposta dessa requisição em um formato de *string* JSON e são tratados para que possam ser interpretados e armazenados no banco de dados SQLITE. O padrão de comunicação via *POST* em Android é demonstrado na Figura 20.

Pode-se observar a instanciação da classe DefaultHttpClient() que trata a comunicação pelo Protocolo de Transferência de Hipertexto (HTTP) até o servidor. É possível observar também os valores a serem enviados em um *ArrayList* chamado *nameValuePairs* e a execução da requisição de comunicação pelo comando *execute*. Posteriormente, o tratamento do retorno da requisição e o armazenamento das informações na variável *result são* destacadas na Figura 20.

```
InputStream is = null;
String result = "";
try{
    HttpClient httpclient = new DefaultHttpClient();
    HttpPost httppost = new HttpPost(strURLExp);

    List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(2);
    nameValuePairs.add(new BasicNameValuePair("IMEI", imei));
    nameValuePairs.add(new BasicNameValuePair("chave_acesso", "07961315977"));

    httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
    HttpResponse response = httpclient.execute(httppost);
    HttpEntity entity = response.getEntity();
    is = entity.getContent();
}catch(Exception e){
    Log.e("log_tag", "Erro no HTTP " + e.toString());
}
try{
    BufferedReader reader = new BufferedReader(new InputStreamReader(is,"iso-8859-1"),8);
    StringBuilder sb = new StringBuilder();
    String line = null;
    while ((line = reader.readLine()) != null) {
        sb.append(line + "\n");
    }
    is.close();
    result=sb.toString();
}catch(Exception e){
    Log.e("TAG", "Erro ao converter o resultado: " + e.toString());
}
```

Figura 20 – Padrão de comunicação via HTTP
FONTE: Autoria própria

Os métodos de comunicação implementados no projeto foram:

- getServerData() (Figura 21): Realiza o tratamento dos dados recebidos pelo servidor e insere no banco de dados as viagens importadas.

```
private void getServerData() {

    JSONArray jArray = new JSONArray(result);
    JSONArray jArrayInd = jArray.getJSONArray(0);
    JSONObject json_geral = jArrayInd.getJSONObject(0);
    comunicado_geral = json_geral.getString("comunicado_geral");
    tipo_aviso = json_geral.getString("tipo_aviso");

    this.dh.deleteAllComunicado();
    if(!comunicado_geral.equals("null") && !tipo_aviso.equals("null"))
        this.dh.insertComunicado(comunicado_geral, tipo_aviso);

    for(int i=1;i<jArrayInd.length();i++){
        JSONObject json_data = jArrayInd.getJSONObject(i);
        Cursor viagens = dh.selectAllViagens("cod_viagem = "
                                +json_data.getString("cod_viagem"));
        if (viagens.getCount() == 0) {
            this.dh.insertViagem(
                    json_data.getString("data_inicio"),
                    json_data.getString("cod_viagem"),
                    json_data.getString("tipo_carregamento"),
                    json_data.getString("destino"),
                    json_data.getString("empresa"),
                    json_data.getString("produto"),
                    "0", "0", "0", "0", "0");
            qtde++;
        }
    }
}
```

Figura 21 – Tratamento de dados recebidos
FONTE: Autoria própria

É possível verificar a obtenção dos dados de “comunicado_geral” e “tipo_aviso” enviados pelo servidor Web de forma isolada exclusão de todos os comunicados armazenados localmente e o salvamento de um novo, caso exista, e também o registro dos dados de novas viagens com os dados recebidos.

- SincExportData() (Figura 22): Seleciona as viagens e organiza os dados em um formato específico que possa ser enviado ao *Web-service* via *POST*. Também altera o *status* da viagem para “sincronizado = 1” (sincronizado).

```
private void sincExportData() {

    Cursor viagens = this.dh.selectAllViagens();
    Integer i = 0;
    if (viagens.moveToFirst() && viagens.getCount() > 0) {
        do {
            Cursor logs = this.dh.selectAllLog("id_viagem = "+viagens.getString(0));
            String textLog = "";
            if (logs.moveToFirst() && logs.getCount() > 0) {
                do {
                    textLog += "|"+logs.getString(2)+"&"+logs.getString(3)+"&"+logs.getString(4)
                            +"&"+logs.getString(5)+"&"+logs.getString(6)+"&"+logs.getString(7)
                            +"&"+logs.getString(8)+"&"+logs.getString(9);
                } while (logs.moveToNext());
            }
            String textViagem = viagens.getString(2) +";"+ viagens.getString(7)
                                +";"+  viagens.getString(10) + ";";
            String textPost = textViagem+textLog;
            nameValuePairs.add(new BasicNameValuePair("VIAGEM_"+i.toString(), textPost));
            dh.updateViagem(Integer.parseInt(viagens.getString(0)), null, null, null, null, "1");
            i++;
        } while (viagens.moveToNext());
    }else{
        Log.d("TAG", "Sem viagens para exportar");
    }

    httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
    HttpResponse response = httpclient.execute(httppost);
    HttpEntity entity = response.getEntity();
    is = entity.getContent();
}
```

Figura 22 – Método para envio das viagens finalizadas
FONTE: Autoria própria

Neste caso, foi necessário utilizar um cursor para que fosse possível percorrer linha a linha as viagens do banco de dados e monta-las no formato que será lido no servidor Web. O formato utilizado para o envio foi a separação de viagens pelo caractere “|” (pipe) e a separação de dados da viagem pelo caractere “;” (ponto e vírgula).

3.2.4 StartBoot

Assim como outros sistemas operacionais, o Android permite que aplicações sejam inicializadas automaticamente ao concluir a inicialização do dispositivo (*Boot*). Neste caso, a funcionalidade foi necessária para iniciar automaticamente a aplicação caso uma viagem já esteja em andamento, para que no caso de um reinício inesperado ou esgotamento de bateria interrompa a atividade do dispositivo. Para defini-la deve-se criar uma classe chamada *StartBoot* como mostrado na Figura 23.

```
public class StartBoot extends BroadcastReceiver {


    Integer idViagem = null;
    BaseDados dh;

    @Override
    public void onReceive(Context context, Intent intent) {
        if ("android.intent.action.BOOT_COMPLETED".equals(intent.getAction())) {

            this.dh = new BaseDados(context);
            Cursor viagensIniciadas = dh.selectViagemEmAndamento();
            if (viagensIniciadas.moveToFirst() && viagensIniciadas.getCount() > 0) {

                Integer idViagem = Integer.parseInt(viagensIniciadas.getString(0));
                if(idViagem > 0){
                    Intent i = new Intent(context, Painel.class);
                    i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    context.startActivity(i);
                }
            }
        }
    }
}
```

Figura 23 – Método para iniciar a aplicação após o boot do dispositivo
FONTE: Autoria própria

O método identifica ao sistema operacional (Android) que esse aplicativo solicita a chamada para si ao concluir-se a inicialização do aparelho. Dessa forma, é verificado se existe alguma viagem em andamento e abre-se a tela de painel de controle da viagem. Caso nenhuma viagem esteja iniciada, a aplicação ignora a chamada e não realiza a abertura do painel de controle.

3.3 INTERFACE WEB

Nesta seção, são detalhadas as interfaces Web e suas respectivas funções no sistema.

3.3.1 Acesso ao Sistema

O acesso ao sistema se dá pela identificação do Cadastro de Pessoa Física (CPF) e uma senha internamente criptografada. A escolha do CPF como identificação para a realização do *login* se deu pelo fato de já ser uma chave única por pessoa.

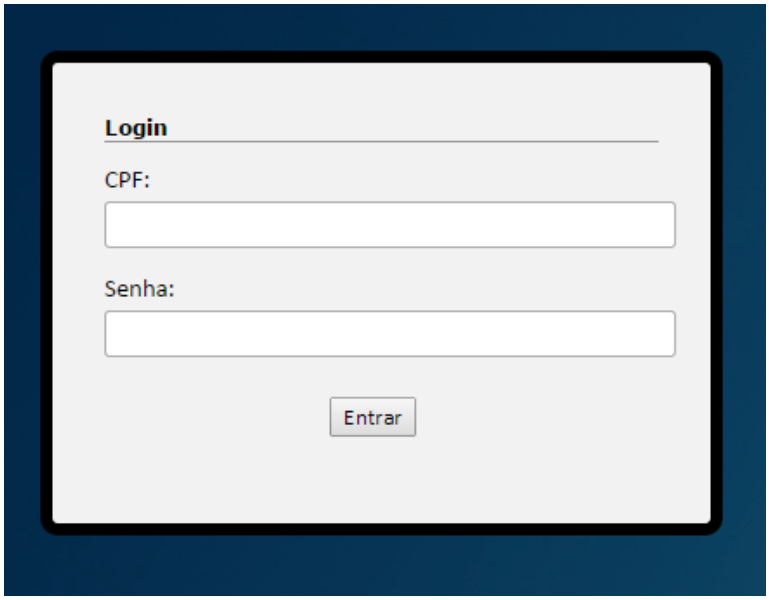


Figura 24 – Login no sistema Web
FONTE: Autoria própria

O usuário deve informar o CPF (que será formatado automaticamente), a senha de acesso único e clicar no botão “Entrar”.

3.3.2 Opções de menu de acesso

Para a navegação entre as telas da aplicação Web, se fez necessária a criação de um menu hierárquico no topo do *layout*, como demonstrado na Figura 25.

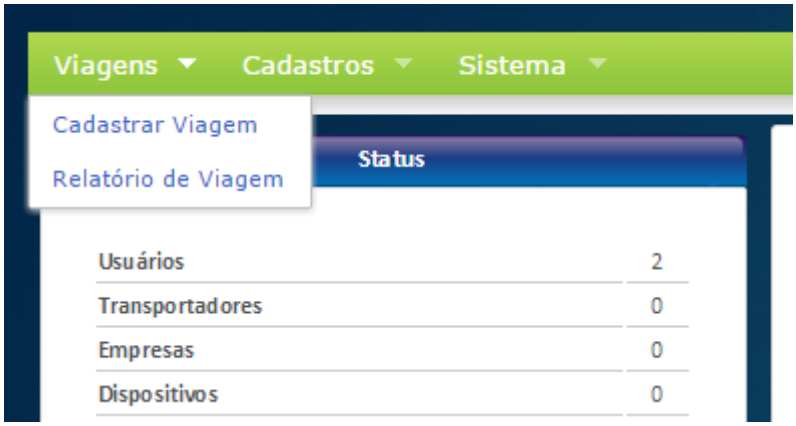


Figura 25 – Opções de menu de acesso
FONTE: Autoria própria

O menu de acesso foi criado a partir do *script* JQUERY “*Stormfish*”, que possui uma implementação extremamente fácil, dada da seguinte forma:

Primeiramente é necessário organizar a estrutura de menu que deverá utilizar as tags “*ul”* e “*li”* de HTML, conforme a Figura 26.

```
<ul class="sf-menu">
    <li>
        <a>Viagens</a>
        <ul>
            <li>
                <a href="http://pmob.com.br/frota/index.php/viagens/viagens">Cadastrar Viagem</a>
            </li>
            <li>
                <a href="http://pmob.com.br/frota/index.php/viagens_relatorio/relatorio">Relatório de Viagem</a>
            </li>
        </ul>
    </li>
    <li>
        <a>Cadastros</a>
        <ul>
            <li>
                <a href="http://pmob.com.br/frota/index.php/empresas/empresas">Empresas</a>
            </li>
            <li>
                <a href="http://pmob.com.br/frota/index.php/produtos/produtos">Produtos</a>
            </li>
        </ul>
    </li>
    <li>
</ul>
```

Figura 26 – Bloco de código da estrutura do menu
FONTE: Autoria própria

Após isso, basta chamar a função do *script Stormfish* como para a montagem do menu, conforme a Figura 27. É importante inserir essa chamada após o carregamento total da página para que se garanta que os elementos que serão aplicados nela já tenham sido criados.

```
$(document).ready(function() {
    $("ul.sf-menu").superfish();
});
```

Figura 27 – Bloco de código da chamada do script Stormfish
Fonte: Autoria própria

3.3.3 Parametrização do sistema

Algumas configurações iniciais são necessárias para que os dispositivos se comuniquem corretamente com a aplicação. Todos os dispositivos deverão ser cadastrados pelo IMEI. Cada dispositivo representa um entregador ou caminhão no qual pode realizar as viagens, e devem ser nomeados de forma amigável para posteriores consultas e cadastros, conforme ilustrado na Figura 28.

Figura 28 – Cadastro de dispositivos
FONTE: Autoria própria

O usuário deverá informar a transportadora o nome e o IMEI do dispositivo e clicar em “Cadastrar”. Uma listagem de todos os dispositivos cadastrados é mostrada abaixo do formulário.

Também deverão ser cadastrados os produtos (Figura 29) que serão transportados pela transportadora.

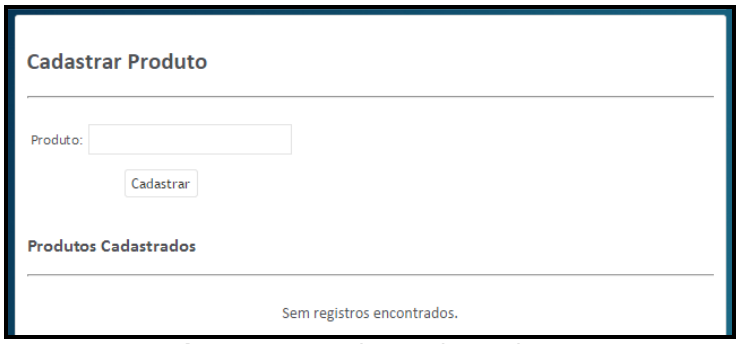


Figura 29 – Cadastro de produtos
FONTE: Autoria própria

O usuário deverá informar o nome dos produtos que serão transportados, e clicar em “Cadastrar”. Uma listagem de todos os produtos cadastradas é mostrado abaixo do formulário.

Também é necessário cadastrar as empresas (Figura 30) onde ocorrerão as cargas e descargas das viagens.

Figura 30 – Cadastro de empresas
FONTE: Autoria própria

O usuário deverá informar o nome da empresa, e clicar em “Cadastrar”. Uma listagem de todas as empresas cadastradas é mostrada abaixo do formulário.

3.3.4 Gerenciamento da Frota

Após o cadastramento dos dispositivos, produtos e empresas, as viagens já podem ser cadastradas a fim de serem sincronizadas com seus respectivos caminhões. Na Figura 31 pode-se observar o formulário de cadastro de viagens.

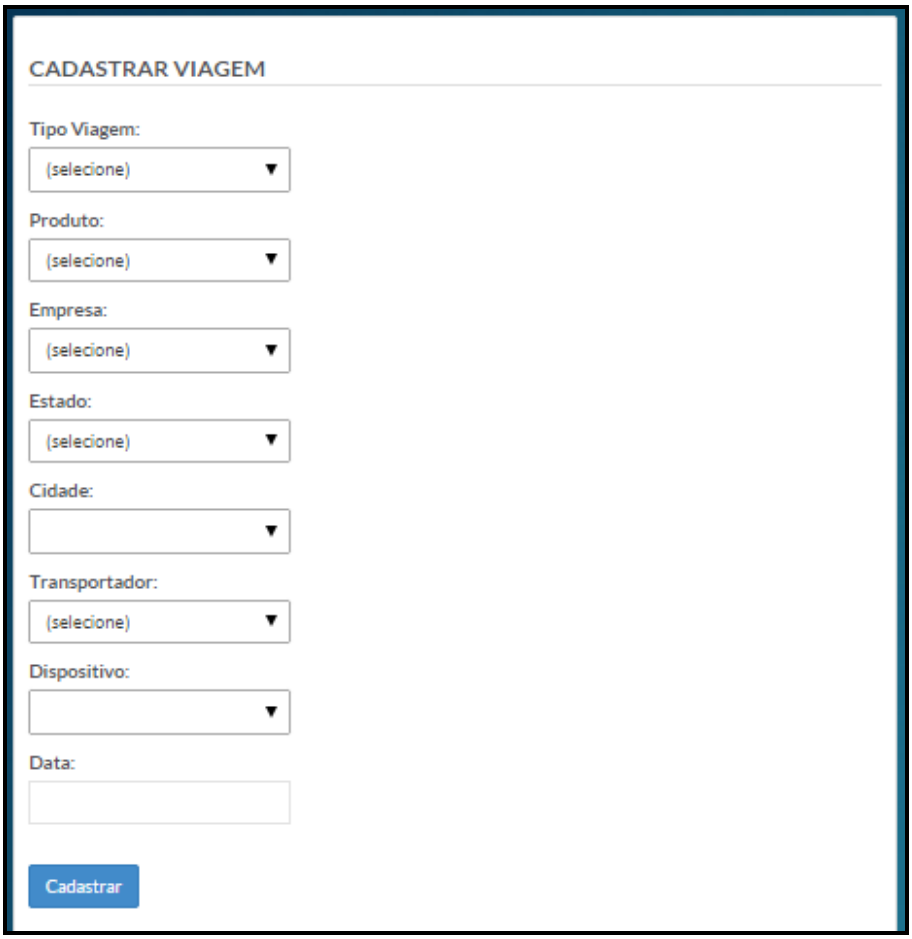


Figura 31 – Cadastro de viagens
FONTE: Autoria própria

Com o cadastro realizado, a viagem já se encontra disponível para o sincronismo com o dispositivo relacionado, podendo ser visualizada na Figura 32.

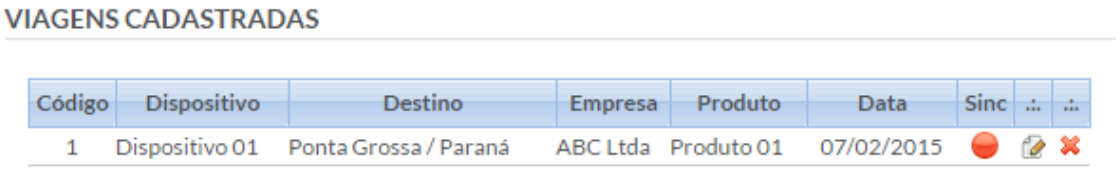


Figura 32 – Viagens cadastradas
FONTE: Autoria própria

A coluna “Sinc” informa se a viagem foi corretamente sincronizada pelo aplicativo (círculo de cor verde) ou ainda está em situação pendente para sincronia (círculo de cor vermelha).

3.3.5 Relatório de acompanhamento

Para o acompanhamento e gerenciamento das viagens realizadas, o usuário da aplicação Web possui acesso aos dados registrados durante a realização das viagens. Na Figura 33 pode-se observar o resumo geral.

Detalhes Viagem: 5	
Tranportadora	Transportadora LTDA
Dispositivo	Dispositivo 01
Destino	Ponta Grossa / Paraná
Empresa	ABC Ltda
Produto	Produto 01
Tipo	Carregamento
Data	02/10/2015

Figura 33 – Resumo da viagem
FONTE: Autoria própria

Também na tela de relatório é mostrado o percurso realizado pelo condutor (traço azul), bem como onde foram realizadas as paradas e eventos registrados durante a viagem (marcadores em vermelho), como mostrado na Figura 34.

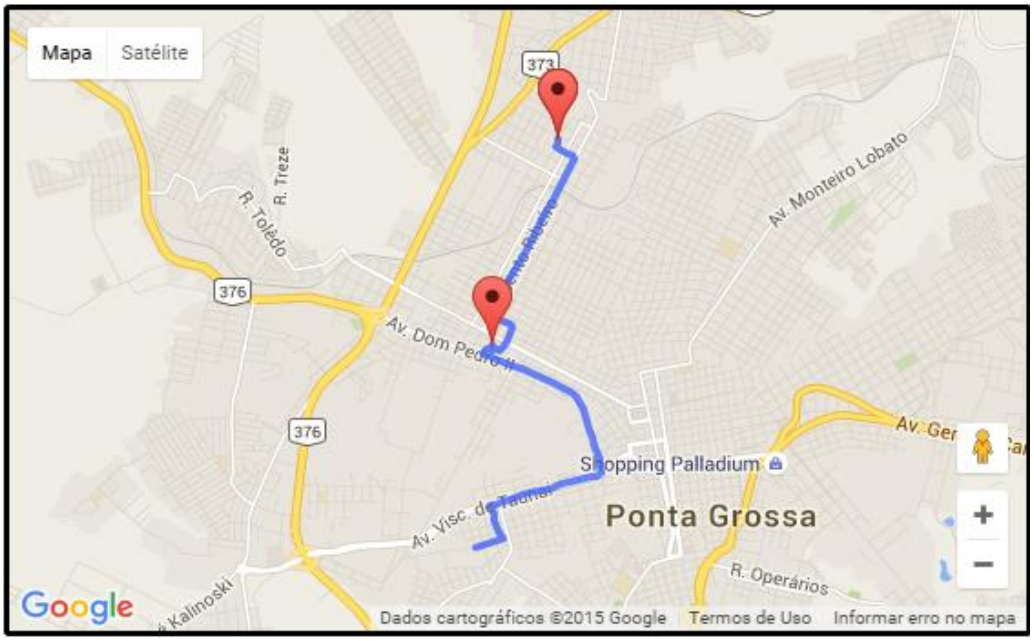


Figura 34 – Percurso da viagem
FONTE: Autoria própria

Por fim, o detalhamento dos eventos gerados com a descrição da parada, data e hora, e a velocidade no momento do registro do evento, como demonstrado na Figura 35.

Data Hora	Evento	Velocidade	.:.
03/10/2015 12:18:44	INICIO VIAGEM	-	
03/10/2015 12:28:17	PARADA PARA DESCANSO	1.00 km/h	
03/10/2015 12:41:29	REINICIO DE VIAGEM	-	
03/10/2015 12:47:21	CHEGADA DESCARREGAMENTO	0.00 km/h	
03/10/2015 12:47:22	INICIO DESCARREGAMENTO	0.00 km/h	
03/10/2015 12:47:25	TERMINO DESCARREGAMENTO	0.00 km/h	
03/10/2015 12:47:25	VIAGEM FINALIZADA	0.00 km/h	

Figura 35 – Eventos da viagem
FONTE: Autoria própria

Ao clicar no ícone de mapa em cada evento, o detalhamento sobre ele é exibido para que seja possível identificar a localização de cada atividade, como demonstrado na Figura 36.

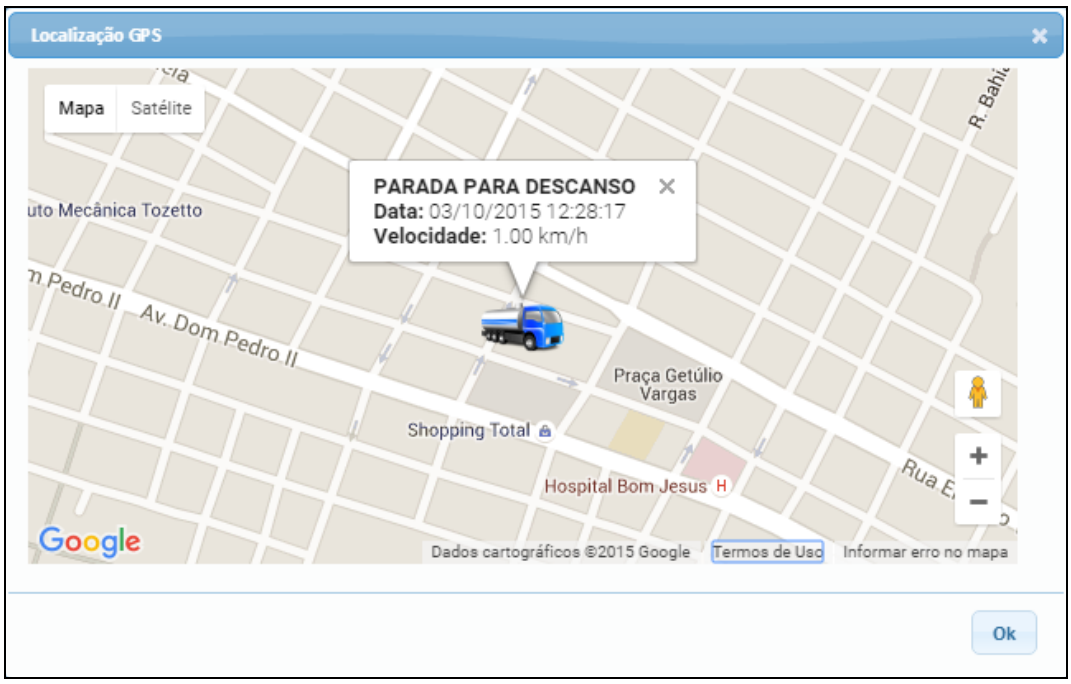


Figura 36 – Detalhamento de evento
FONTE: Autoria própria

Com tais dados, é possível monitorar e auditar todo o histórico do percurso realizado durante as viagens.

3.4 INTERFACE *MOBILE*

Nesta seção, serão detalhadas as interfaces da aplicação *mobile* e suas respectivas funções no sistema.

3.4.1 Acesso ao Sistema

O acesso ao aplicativo não requer uma tela de *login*, pois o cadastramento realizado na interface Web por meio do IMEI que identifica o aparelho automaticamente ao se abrir a aplicação. Se um dispositivo não identificado acessar a aplicação, nenhuma viagem será importada para realização.

3.4.2 Opções de menu de acesso

As opções do menu de acesso na aplicação *mobile*, são: Painel de Controle, Relatório e Sincronização, conforme pode ser visto na Figura 37:

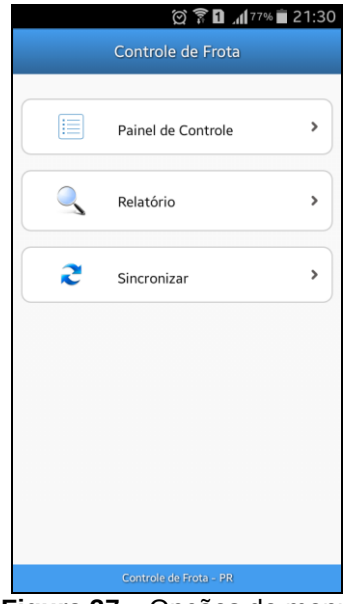


Figura 37 – Opções de menu
FONTE: Autoria própria

A opção de Painel de Controle levará o usuário as suas viagens pendentes, a opção de relatório dará ao usuário a possibilidade de pesquisar entre as viagens já realizadas ou em andamento para acompanhamento de *logs* gerados e a opção

“Sincronizar” para que o sincronismo de viagens entre as aplicações *mobile* e Web possam ser geradas.

3.4.3 Geração de Eventos

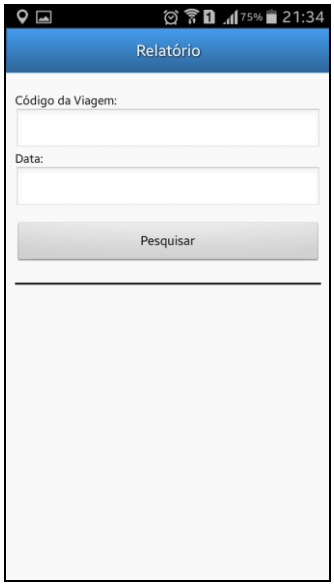
A tela do Painel de Controle mostrada na Figura 38 obtém as informações básicas da viagem, como destino, propósito e produto a ser transportado. Nessa tela, também são marcados os eventos da viagem e a localização automática do dispositivo.

Figura 38 – Geração de eventos
FONTE: Autoria própria

O usuário pode iniciar a viagem, informar as paradas realizadas e as etapas de carregamento e descarregamento, conforme o tipo de viagem. Um contador de tempo é iniciado e também é demonstrado o status de localização GPS, que pode ser: “GPS desativado”, “Localizando GPS”, “Obtendo uma melhor precisão” e “GPS”.

3.4.4 Relatório

O sistema também disponibiliza ao usuário uma tela de relatório para consulta básica de viagens realizadas pelo dispositivo. Para realizar a busca pode ser inserido o código da viagem, ou a data inserida para o início da mesma, conforme a Figura 39.



a) Pesquisa

b) Relatório de viagem

Figura 39 – Relatório de viagem
FONTE: Autoria própria

Na imagem “a” pode-se observar o formulário de pesquisa de viagens, que podem ser filtradas por código ou data. Na imagem “b” é possível visualizar os detalhes da viagem selecionada, como data de início e término, destino, carga e se é uma viagem de carregamento ou descarregamento.

3.4.5 Sincronização

Na interface de sincronização com a aplicação Web, conforme a Figura 40 são informadas as viagens recebidas e enviadas pelas sincronizações entre os *Web-services*.

Figura 40 – Sincronização
FONTE: Autoria própria

Basta acessar a tela de sincronização que a comunicação entre a aplicação *mobile* e Web é automaticamente iniciada. Nessa comunicação são enviados, em um formato específico, todos os percursos realizados pelo dispositivo e também o recebimento de novas viagens cadastradas na aplicação Web. Por se tratar de uma comunicação básica de *strings*, seu custo de banda de dados é extremamente baixo.

3.5 CONSIDERAÇÕES DO CAPÍTULO

De modo geral, o desenvolvimento do trabalho se baseou em ferramentas gratuitas para implementar suas funcionalidades. No ambiente Web foram utilizadas ferramentas para auxiliar no desenvolvimento, como o *framework* e *plug-ins* de *interface* e também para melhorar a experiência do usuário no manuseio da ferramenta, como *plug-ins* de formulários e mapeamento.

No módulo *mobile*, por sua vez, o desenvolvimento foi inteiramente baseado em pesquisas e tutoriais de criação para cada necessidade que se fazia no projeto, como a criação de *interfaces*, comunicações via *Web-services* e suas particularidades específicas no desenvolvimento de uma aplicação móvel.

4 RESULTADOS

Neste capítulo são apresentados os resultados obtidos com o desenvolvimento do trabalho, estando organizado em cinco seções. A seção 4.1 apresenta a análise das cargas retornadas ao depósito, a seção 4.2 discorre sobre o consumo de combustível, a seção 4.3 mostra a adaptabilidade do aplicativo, a seção 4.4 apresenta o aprendizado adquirido pelos acadêmicos com a pesquisa realizada e por fim, a seção 4.5 discorre sobre as considerações do capítulo.

4.1 CARGAS RETORNADAS AO DEPÓSITO

Observou-se uma diminuição de entregas não efetuadas por motivos de ausência do cliente e não localização do endereço informado. Na Figura 41, pode-se observar o comparativo mensal de cargas entregues na data prevista por entregadores que utilizaram o aplicativo e por entregadores que não utilizaram.

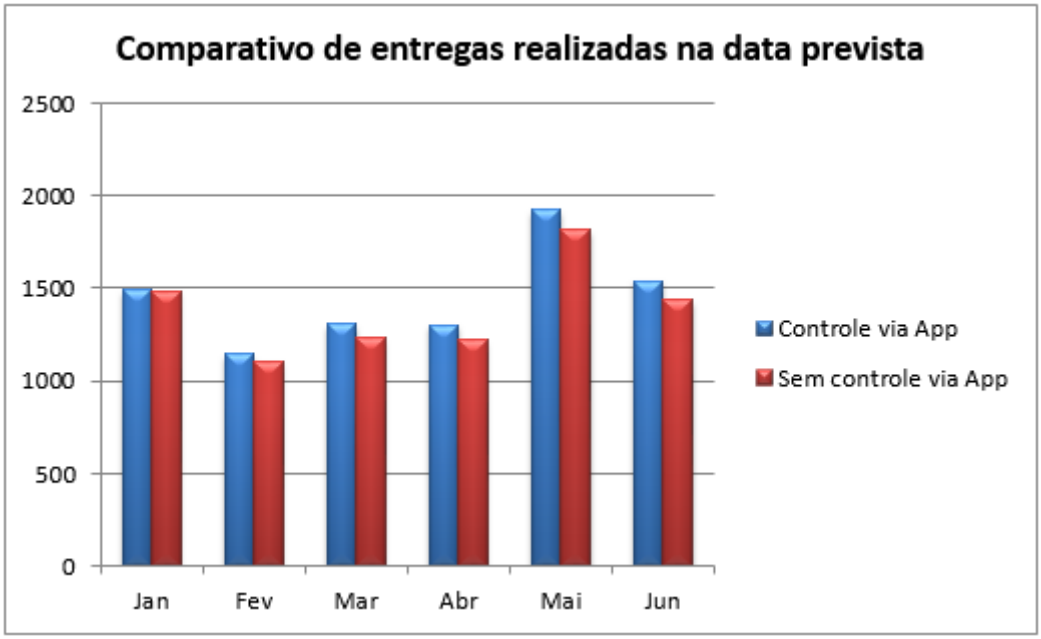


Figura 41 – Comparativo de entregas realizadas na data prevista
FONTE: Autoria própria

Para a elaboração do resultado, foram selecionados cinco condutores utilizando o aplicativo e outros cinco que não utilizaram separados em equipes da empresa ABC Ltda. Em ambas as equipes, foram selecionados ambientes similares

para que o resultado fosse corretamente mensurável. Para cada equipe, três entregadores para atendimento em uma cidade de 300.000 a 350.000 habitantes e dois para cidades de 50.000 a 100.000 habitantes.

4.2 CONSUMO DE COMBUSTÍVEL

Com a diminuição de entregas não efetuadas, houve economia média mensal de 2,5% dos caminhões das mesmas equipes selecionadas no teste anterior, como demonstrado na Figura 42.

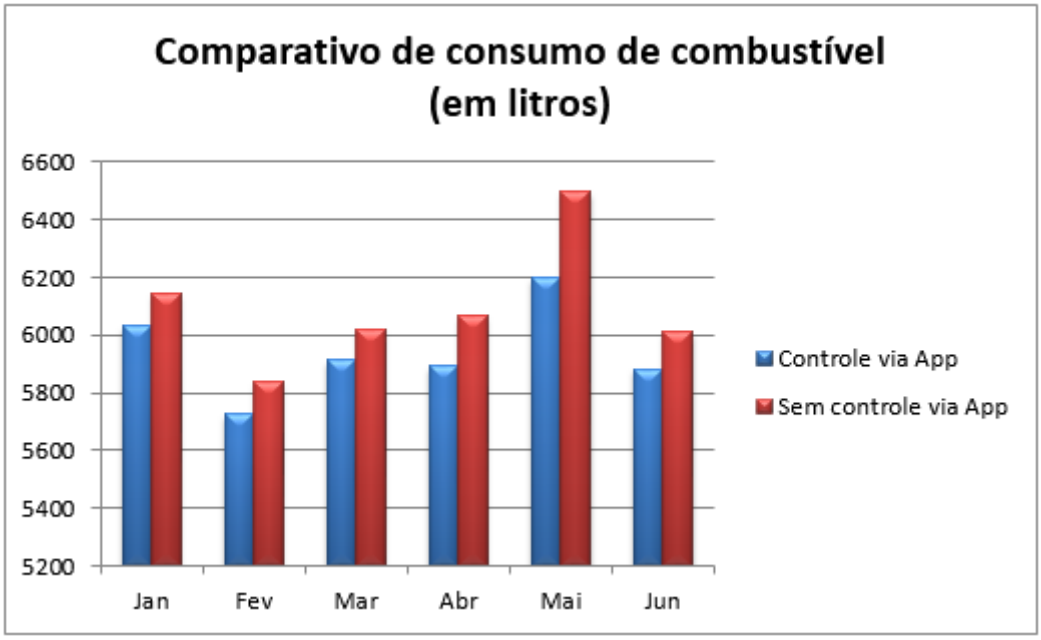


Figura 42 – Comparativo de consumo de combustível (em litros)
FONTE: Autoria própria

4.3 ADAPTABILIDADE AO APLICATIVO

Para analisar a adaptabilidade dos entregadores ao uso do aplicativo, foram selecionados dez entregadores para responderem as perguntas:

- Qual a facilidade no uso da aplicação (nota de 0 a 5)
- Qual o auxílio na realização das entregas (nota de 0 a 5)
- Qual o controle e organização do trabalho (nota de 0 a 5)

Tabela 3 – Questionário de adaptabilidade ao aplicativo

	Facilidade	**Auxílio**	**Controle**
Entregador 1	5	2	4
Entregador 2	4	3	5
Entregador 3	5	4	3
Entregador 4	5	4	2
Entregador 5	3	3	4
Entregador 6	4	4	4
Entregador 7	5	5	5
Entregador 8	3	5	3
Entregador 9	5	5	4
Entregador 10	5	3	5
TOTAL	**44**	**38**	**39**

Fonte: Autoria própria

Analisando a os dados da Tabela 3 que os valores variam de 0 a 5 como nota, observa-se que a facilidade de uso da aplicação é bem alta, fato que se dá pela simplicidade e objetividade que a aplicação proporciona. Ao analisar o quesito auxílio é possível inferir que o resultado que a ferramenta traz com a facilidade de uso e controle proporcionado, pois a aplicação, de maneira fácil e rápida auxilia na realização das viagens no percurso de entrega.

O mais destacado pelos entregadores foi a facilidade em avisar e receber da central as situações de viagem. E por fim, o controle e organização das entregas também teve destaque pela facilidade em reunir as entregas em uma única tela, dessa forma proporcionando uma visão rápida de toda a situação das cargas.

4.4 APRENDIZADO ACADÊMICO

Pode-se destacar também o aprendizado dos acadêmicos autores no desenvolvimento do presente trabalho, principalmente no desenvolvimento da aplicação Android que não foi abordado no decorrer do curso. Noções de geomapeamento e banco de dados foram amplamente aperfeiçoados como também a assertividade em realização de buscas por periódicos e materiais de pesquisa disponíveis na Web.

4.5 CONSIDERAÇÕES DO CAPÍTULO

Analisando os resultados observa-se que a aplicação traz resultados positivos no consumo de combustível e melhora nas entregas principalmente em que os clientes estavam ausentes e posteriormente avisam a central a qual ativa novamente a entrega para que o condutor realize uma nova tentativa de entrega. Dessa maneira a mercadoria não retorna para base, poupando tempo e combustível.

É notável também que, de acordo com os resultados, a aplicação traz agilidade e organização com a maneira que os dados são demonstrados. Considera-se que quando aplicada em novos motoristas ou em locais desconhecidos pelos condutores a aplicação traga resultados ainda mais eficazes.

5 CONSIDERAÇÕES FINAIS

Neste capítulo serão apresentadas as considerações finais do trabalho, demonstrando no Capítulo 5.1 a Conclusão e 5.2 os Trabalhos Futuros que podem ser acrescidos neste trabalho.

5.1 CONCLUSÃO

O desenvolvimento desse projeto possibilitou alcançar os objetivos determinados em sua criação, não se limitando apenas ao gerenciamento eficaz das transportadoras, mas também as facilidades para a realização do trabalho de entregas de cargas por parte dos motoristas.

A versão Web traz além da informatização e organização do controle de viagens, um grande diferencial na facilidade de inserir, auditar, gerenciar e acompanhar a frota de maneira rápida e eficiente por meio de recursos visuais.

A versão *mobile*, por sua vez, atuando como coletor de dados das viagens traz a mobilidade como grande trunfo da solução, além da facilidade do manuseio da ferramenta com uma interface acessível, já que seu uso se dará muitas vezes durante a realização das viagens.

Apesar da incapacidade de trabalhar totalmente *off-line*, necessitando de uma conexão ativa com a Internet para sincronização dos dados, o módulo *mobile* consegue contornar, de maneira aceitável, as falhas de cobertura de rede móvel existentes nas operadoras do país. Com os recursos e funcionalidades disponíveis na aplicação, o sistema tem grandes possibilidades de se enquadrar no mercado de gerenciamento de frota e suas soluções apresentadas podem ser de grande valia para o ramo rodoviário do país, que atualmente é tão carente de novas tecnologias de segurança e auditoria.

O controle, gerenciamento e acompanhamento da frota tornam o trabalho de entrega de cargas mais seguro, além de facilitar as tomadas de decisões por parte da administração da frota.

5.2 TRABALHOS FUTUROS

Por se tratar de um projeto piloto, novas implementações e melhorias podem ser incorporadas com o intuito de agregar mais agilidade e eficiência. No decorrer do desenvolvimento foram identificadas novas funcionalidades e também possíveis novas tecnologias para a continuação deste projeto, são eles:

- Remodelar o layout do aplicativo *mobile* para as novas tendências;
- Implementar para sincronizar automaticamente em períodos automáticos;
- Adequar a aplicação Web para padrões de layout responsivo que se adapta as várias resoluções de tamanhos de tela existentes nos dispositivos atuais a fim de possibilitar uma melhor experiência para usuários que utilizarão o sistema pelo *smartphone*;
- Desenvolver na aplicação *mobile* funcionalidades emergenciais, a fim de notificar a central no caso de problemas durante o percurso das viagens como por exemplo pneu furado ou assalto.
- Criar versões *mobile* para as plataformas iOS e Windows Phone.

REFERÊNCIAS

ALECRIM, Elmasri. Banco de dados MySql e PostgreSQL. 2008. Disponível em: <http://www.infowester.com/postgremysql.php> Acesso em: 16 de Jul. 2014

ANDROID, Get Started with Publishing. 2015. Disponível em: <http://developer.android.com/distribute/googleplay/start.html> Acesso em: 21 de Out. 2015

ANDROID. LocationManager. 2015. Disponível em: <http://developer.android.com/reference/android/location/LocationManager.html> Acesso em: 20 de Ago. 2015.

BRASIL. Decreto-lei nº 12.619, de 30 de abril de 2012.

CODEIGNITER. About CodeIgniter. 2015. Disponível em: <https://www.codeigniter.com/help/about> Acesso em: 27 de Set. 2015.

DALL'OGLIO, Pablo. PHP Programando com Orientação a Objetos. 2. ed. São Paulo: Novatec, 2009. 573 p. (*Model View Controller) p. 477 – 478.*

DEVMEDIA. Introdução ao framework PHP CodeIgniter. Disponível em <http://www.devmedia.com.br/introducao-ao-framework-php-codeigniter/27346> Acesso em: 27 de Set. 2015.

DEVMEDIA. SQLite no Android Trabalhando com persistência de dados no Android. Disponível em <http://www.devmedia.com.br/post-19201-SQLite-noAndroid.html> Acesso em: 27 de Set. 2015.

ECLIPSE. About. Disponível em: <https://www.eclipse.org/ide/> Acesso em: 12 de Nov. 2015.

ENCYCLOPAEDIA BRITANNICA. Britannica Academic. Encyclopædia Britannica Inc., 2015. GPS. Disponível em: <http://academic.eb.com/EBchecked/topic/235395/GPS> Acesso em: 27 de Ago. 2015.

FERRARI, Bruno. "Todo o poder ao freguês: smartphones, geolocalizacao e softwares de big data estão transformando a experiência de comprar--e aniquilando a fronteira entre o mundo real e o digital." Exame 19 Feb. 2014: 30+. Academic OneFile. Acesso em: 20 de Ago. 2015.

FRAMINGHAM, Mass. Top Four Smartphone Operating Systems. 2015. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS25450615> Acesso em: 21 de Set. 2015

GOOGLE, Adding a Google Map to your website. 2015. Disponível em: <https://developers.google.com/maps/tutorials/fundamentals/adding-a-google-map> Acesso em: 20 de Ago. de 2015.

HIJJAR, Maria F.; LOBO Alexandre. Cenário da infraestrutura rodoviária no Brasil. Disponível em: <http://www.ilos.com.br/web/index.php?option=com_content&task=view&id=1807&Itemid=74&lang=br> Acesso em: 08 de Set. 2015.

HTML Tag. Disponível em: <http://www.w3schools.com/tags/tag_div.asp> Acesso em: 31 de Ago. 2015.

IDC International Data Corporation, Smartphone OS Market Share. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp> Acesso em: 12 de Nov. 2015.

IPLOCATION, What is IP-based Geolocation? 2015. Disponível em: <https://www.iplocation.net/> Acesso em: 13 de Nov. 2015.

PHP, Hypertext Preprocessor. Disponível em: <https://php.net/> Acesso em: 14 de Fev. 2014.

POSTGRESQL, About. Disponível em: <http://www.postgresql.org/about/ > Acesso em: 20 de Ago. 2015.

SQLITE, About SQLite. Disponível em: <https://www.sqlite.org/about.html> Acesso em: 20 de Ago. 2015.

TIOBE Index for August 2015. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html> Acesso em: 31 de Ago. 2015.

TRUCKPAD, Disponível em:<http:// http://truckpad.com.br/> Acesso em 12 de nov. 2015

Zanotta, Daniel C.; Cappelletto, Eliane; Matsuoka, Marcelo T. O GPS: unindo ciência e tecnologia em aulas de física. 2011. Disponível em: <http://www.scielo.br/scielo.php?pid=S1806-11172011000200014&script=sci_arttext> Acesso em: 13 de Nov. 2015.