Revista Engenho, vol.8 – Setembro de 2013

SISTEMA WEB PARA CONTROLE DE FROTAS

Felipe Ricardo Freitas,

lipofreitas@yahoo.com.br.

Carlos Eduardo Câmara, Carlos Eduardo.

Centro Universitário Padre Anchieta

dinhocamara@gmail.com.

RESUMO

Este artigo tem como objetivo o desenvolvimento de um sistema web para o controle de frotas de veículos, que auxilie as empresas disponibilizando as informações referentes à localização de seus veículos, possibilitando assim a tomada de decisão baseada na posição atual de cada veiculo. Os estudos realizados neste projeto demonstram que a cada dia torna-se imprescindível monitorar os veículos de uma frota. Utilizando a tecnologia *GPS*, aliada da plataforma *WEB*, as informações serão vistas em tempo real pelos administradores das frotas em qualquer lugar que eles estejam fisicamente.

Palavras chave: *GPS, PostgreSql, Java, PHP*.

ABSTRACT

This article aims to develop a web-based system for tracking vehicle fleets, which assists companies providing the information regarding the location of their vehicles, thus enabling decision making based on the current position of each vehicle. Studies show that this project every day it becomes essential to monitor a fleet of vehicles. Using GPS technology, combined with the web platform, the information will be viewed in real time by the managers of fleets anywhere they are physically.

Keywords: GPS, PostgreSql, Java, PHP.

INTRODUAÇÃO

A tecnologia *Global Positioning System (GPS)* está em crescente evolução, esta tecnologia foi criada em 1973 pelo departamento de defesa dos EUA (*DoD* - Estados Unidos da America), mas foi declarado totalmente operacional apenas em 1995.

"Sem sombra de dúvidas, o sistema GPS é o maior avanço tecnológico das últimas décadas, na navegação e no posicionamento desde o advento da bússola" (SEGANTINE, 2005).

Hoje a tecnologia *GPS* é o principal auxílio aos motoristas em suas rotas, neste caso o *GPS* tem a função de mostrar ao condutor qual o caminho a ser seguido para chegar ao destino desejado. Outro objetivo desta tecnologia é auxiliar o rastreamento de veículos, principalmente de caminhões. Em 1994 a Autotrac Comércio e Telecomunicações S/A, empresa que tem como um de seus sócios o tricampeão mundial de Fórmula 1 Nelson Piquet, deu início ao segmento de comunicação móvel de dados, monitoramento e rastreamento de frotas no Brasil (AUTOTRAC, 2012).

Existem alguns sistemas de controle de frotas que auxiliam as empresas, mas esses sistemas apenas possibilitam que se possa saber onde está determinado veículo e se naquele exato momento está dentro de sua rota. A ideia deste trabalho consiste em mostrar um histórico de cada veículo, assim será possível verificar e monitorar de forma automática a localização exata e analisar se o mesmo está percorrendo ou percorreu realmente a sua rota pré-determinada e se efetuou algum desvio.

Por exemplo, uma empresa de taxi poderá controlar todas as suas viagens, como escolher qual veículo atenderá a um chamado, verificar o tempo gasto em cada viagem ou se esteve parado por muito tempo sem nenhuma notificação e assim tomar alguma decisão, como acionar a segurança por exemplo.

OBJETIVO

Neste trabalho é apresentado o desenvolvimento de um sistema de controle de frotas, com objetivo principal de exibição visual da posição de veículos sobre um mapa através do posicionamento por GPS. Exibindo desta forma ao gestor de logística (ou ao usuário do

sistema), a posição atual de cada veículo previamente cadastrado e as informações necessárias ao seu monitoramento.

Consequentemente outros recursos serão alcançados, como poder analisar o tempo gasto por um determinado veículo em seu trajeto, controle da distância percorrida, prever o tempo necessário para um trajeto ser completado, verificar o tempo restante para um veículo chegar ao final de sua rota e até mesmo avisar ao gestor quando um veículo saiu de sua rota pré-determinada. Informações que podem ser retiradas do serviço de mapas ou do programa de logística que determine rotas.

GPS

O sistema *GPS* é um sistema de radio-navegação, desenvolvido pelo departamento de defesa dos Estados Unidos da América. Este sistema utiliza informações obtidas de uma constelação de satélites conhecida como *Navigation Satellite with Time and Ranging (NAVSTAR).*

A constelação conta com 24 satélites, sendo 21 em operação e mais 3 reservas. Estes satélites estão orbitando em 6 planos inclinados em 55º em relação ao plano do Equador, a uma altitude média aproximada de 20.200 km. Estas órbitas são no formato de elipse com a distância do maior semieixo aproximado de 26.600 km, como mostra a figura 1. O tempo de cada ciclo destes satélites é de 11h 57’ 58,3’’ no Tempo Universal Coordenado (*UTC*).

Esta constelação garante que cada satélite repita sua posição todo dia com defasagem aproximadamente de 4 minutos, também garante que a qualquer instante haja pelo menos 4 satélites acima da linha do horizonte de uma antena receptora (SEGANTINE, 2005).

Figura 8- Ilustração da constelação de satélites (GPSCENTER, 2012).

Na superfície terrestre existem 5 estações de controle, sendo que uma delas recebe a posição de Estação *Master* de Controle. Essas estações têm a função de manter os satélites dentro de suas respectivas órbitas e manter o bom funcionamento de cada satélite, verificando as placas solares, nível de energia das baterias, o nível de combustível necessário para que o satélite possa voltar a sua posição caso necessário (SEGANTINE, 2005).

Essas estações de controle têm a responsabilidade de controlar os relógios de cada satélite, ou seja, sincronizá-los, rastreá-los e fornecer suas posições periodicamente.

Os satélites emitem ondas de rádio em duas frequências a L1 que opera a 1575,42 MHz e a L2 a 127,60 MHz. Estas ondas são emitidas para a superfície, com a identificação do satélite, e com informações do horário de seu relógio que possui precisão de nanosegundos. O receptor captura essas informações determina a sua distância em relação aquele satélite fazendo cálculo pelo tempo em que a informação levou para chegar até ele. Tendo essa informação o receptor utiliza o método conhecido como Trilateração (triangulação) para saber sua posição. Este método de localização utiliza a informação da distância entre o receptor e o satélite e traça um circulo em torno do satélite, para ter a localização exata do receptor este método utiliza a informação de pelo menos 4 satélites, assim a localização do receptor está no único ponto em que todos os círculos se encontram (LANGENDOLFF, 2008).

Cálculo da distância:

$R_i = C \times Dt_i$

Onde: R_i – Distancia entre o receptor e o satélite emissor.

C – Velocidade da Luz.

Dt_i – tempo transmitido pelo satélite.

Observação: o índice i refere ao número do satélite (i=1, 2, 3).

FERRAMENTAS DE DESENVOLVIMENTO

Ferramentas de desenvolvimento tratam-se das linguagens de programação, sistemas de gerenciamento de banco de dados e tecnologias utilizadas para o desenvolvimento de um sistema ou aplicação.

As escolhas das ferramentas de desenvolvimento deste projeto foram feitas levando em consideração as linguagens de programação disponíveis no mercado para o desenvolvimento de aplicações *WEB*, linguagens para desenvolvimento em plataforma *Android*, banco de dados robusto e seguro e ferramentas gratuitas.

Banco de Dados

O Banco de dados escolhido foi o *Postgresql* que é um sistema de gerenciamento de banco de dados objeto-relacional de código aberto e de uso gratuito. Com mais de 15 anos de desenvolvimento ativo, é resultado de uma grande evolução de um projeto que foi desenvolvido na Universidade de Berkeley na Califórnia. Tinha como líder do projeto Michael Stonebraker, um dos pioneiros em banco de dados relacionais.

O *Postgresql* é comprovadamente um sistema de gerenciamento de forte reputação em confiabilidade, segurança e integridade de dados. Por esses motivos ele vem sendo escolhido como melhor alternativa por muitas empresas (POSTGRESQL, 2012).

PHP

PHP (um acrônimo recursivo para "*PHP: Hypertext Preprocessor*", originalmente *Personal Home Page*), trata-se de uma linguagem interpretada, que originalmente é usada apenas para o desenvolvimento de aplicações que atuavam no lado do servidor, com capacidade de gerar conteúdo dinâmico.

Criado em 1995 por Rasmus Lerdof, o *PHP* foi uma das primeiras linguagens passiveis a inserção em documentos *HTML*. O objetivo principal do *PHP* é o desenvolvimento *WEB*. Suas principais características são:

- Velocidade e robustez;
- Estruturado e orientação a objetos;
- Portabilidade;
- Tipagem dinâmica;
- Sintaxe muito parecida com C/C++;
- Código aberto;
- O código *PHP* é processado no lado do servidor, e é enviado para o cliente através do *browser*. No lado do cliente, é gerada apenas a visualização no *browser* através de *HTML*. Uma das vantagens desta

linguagem é que se trata de uma linguagem de *scripts*, ou seja, o código não precisa ser compilado antes da execução (PHP, 2012).

Google Maps API

Trata-se de um serviço gratuito, disponível para qualquer site que o público possa usar gratuitamente. O *Google Maps* é um serviço de geolocalização disponibilizado pela *Google*, que permite efetuar pesquisa e visualização de mapas e imagens de satélite da Terra.

API é o acrônimo de *Application Programming Interface* ou, em português, *Interface* de Programação de Aplicativos. Esta *interface* é o conjunto de padrões de programação que permite a construção de aplicativos utilizando o *Google Maps* através de classes desenvolvidas pela *Google* e disponibilizadas para desenvolvedores (MAPS, 2012).

Java

Desenvolvido pela *Sun Microsystems* na década de 1990, e chefiado por James Gosling, trata-se de uma linguagem orientada a objeto. Tem como principal característica a portabilidade, porque não depende de plataforma, uma aplicação em *Java* é carregada e executada pela máquina virtual *Java Virtual Machine (JVM)* que é um *middleware* entre o programa-código e o código executável para o determinado sistema operacional. Deste modo uma aplicação *Java* pode ser executada em qualquer plataforma, *hardware/SO*, desde que tenha uma *JVM* instalada. Outras características são a versatilidade, a eficiência e a segurança. Deste modo o *Java* é utilizado desde em *laptops* à *datacenters*, de *games* à supercomputadores científicos, de telefones celulares à *Internet*.

Até os dias de hoje, a plataforma *Java* atraiu mais 9 milhões de desenvolvedores de *software*.

Alguns números:

- 1,1 bilhão de *desktops* executam *Java;*
- 930 milhões de *downloads* do *Java Runtime Environment* a cada ano;
- 3 bilhões de telefones celulares executam *Java*;
- 100% de *Blu-ray players* executam *Java*;
- 1,4 bilhões de Placas *Java* são fabricadas a cada ano;
- Decodificadores *Java*, impressoras, câmeras *WEB*, *games*, sistemas de navegação automotiva, casas lotéricas, dispositivos médicos, estações de pagamento de estacionamento e mais (JAVA.com, 2012).

PLATAFORMAS DE SUPORTE DO SISTEMA PROPOSTO

Servidor

No desenvolvimento deste projeto foi utilizado um servidor, onde está hospedado o banco de dados cujas informações serão gravadas, como o cadastro da empresa, seus usuários, veículos, dispositivos e os históricos de posições, neste servidor também estão configuradas as plataformas de desenvolvimento. Neste servidor também está hospedado o sistema *web*, onde são exibidas todas as informações e utilizado para cadastrar e alterar informações.

Este ambiente está configurado sobre um sistema operacional *Ubuntu 12.04 Precise Pangolin LTS*, que se trata de um sistema sobre o *Kernel Linux*, baseado em *Debian*. O *Ubuntu* é patrocinado pela Canonical, e desenvolvido por uma comunidade formada por voluntários (UBUNTU, 2012).

Mobile

Para a captura da posição de determinado veículo ou usuário, foi desenvolvido um aplicativo para capturar a localização e transmitir para um banco de dados, este aplicativo foi desenvolvido para aparelhos que utilizam o sistema *Android* e possuem tecnologias *GPS* e acesso a *internet* por sinal 3G e/ou *Edge*.

Android é um sistema operacional para dispositivos móveis, como celulares e *tablets*, baseado no *Kernel Linux*. É desenvolvido pela *Open Handset Alliance*, que é uma aliança de várias empresas entre elas a *Google Corp*., que lidera esse grupo, *Motorola*, *Samsung*, *Nvidia*, *Dell*, *Intel* entre outras (ANDROID, 2012).

DOCUMENTAÇÃO DE DESENVOLVIMENTO DO PROTÓTIPO

A *Unified Modeling Language (UML)* que é uma linguagem visual para especificação, construção e documentação de *softwares*. A *UML* é uma linguagem diagramática que padroniza as metodologias de desenvolvimento de sistemas. A *UML* baseia-se em diagramas modelados e classificados de acordo com a abstração da visualização dos processos de um sistema ou aplicação (SILVA, 2001).

Através da *UML* utilizamos os diagramas de casa de uso, de classe e de entidade-relacionamento para demonstrar as funcionalidades e comportamento do sistema proposto neste artigo.

Diagrama de caso de uso

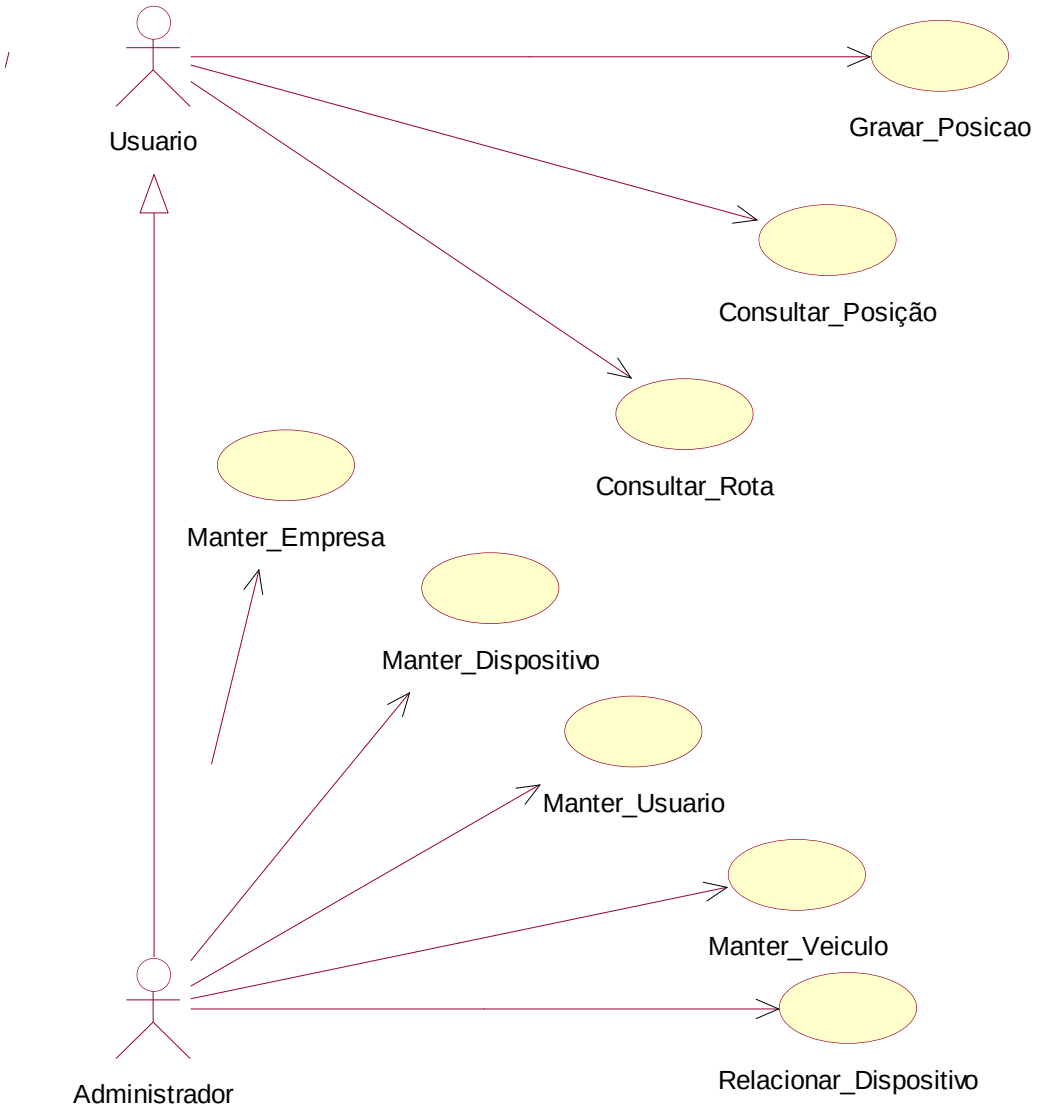

Figura 9 – Diagrama de Caso de Uso

A figura 2 apresenta as funcionalidades do sistema, a partir da interação com os atores externos. Onde o ator Administrador (Gestor de Logística) é o responsável por cadastrar e alterar os cadastros de empresas, dispositivos, usuários, veículos e o relacionamento entre empresa, dispositivo, usuário e veiculo, através dos casos de uso, Manter_Empresa, Manter_Dispositivo, Manter_Usuario, Manter_Veiculo e Relacionar_Dispositivo.

Já o ator Usuário exerce as atividades de inserir sua posição, consultar sua posição e suas rotas através dos casos de uso Gravar_Posição, Consultar_Posição e Consultar_Rota.

Cada passo corresponde a uma ação de um ator como mostra a tabela 1.

Administrador cadastra a empresa. Administrador cadastra o dispositivo. Administrador cadastra o usuário. Administrador cadastra o veiculo. Para cada dispositivo cadastrado o Administrador cadastra o relacionamento dele com o Usuário e veiculo. Usuário grava sua posição. Usuário consulta sua posição. Usuário consulta sua rota.

Tabela 1 - Fluxo Principal

1.1 Diagrama de Classes

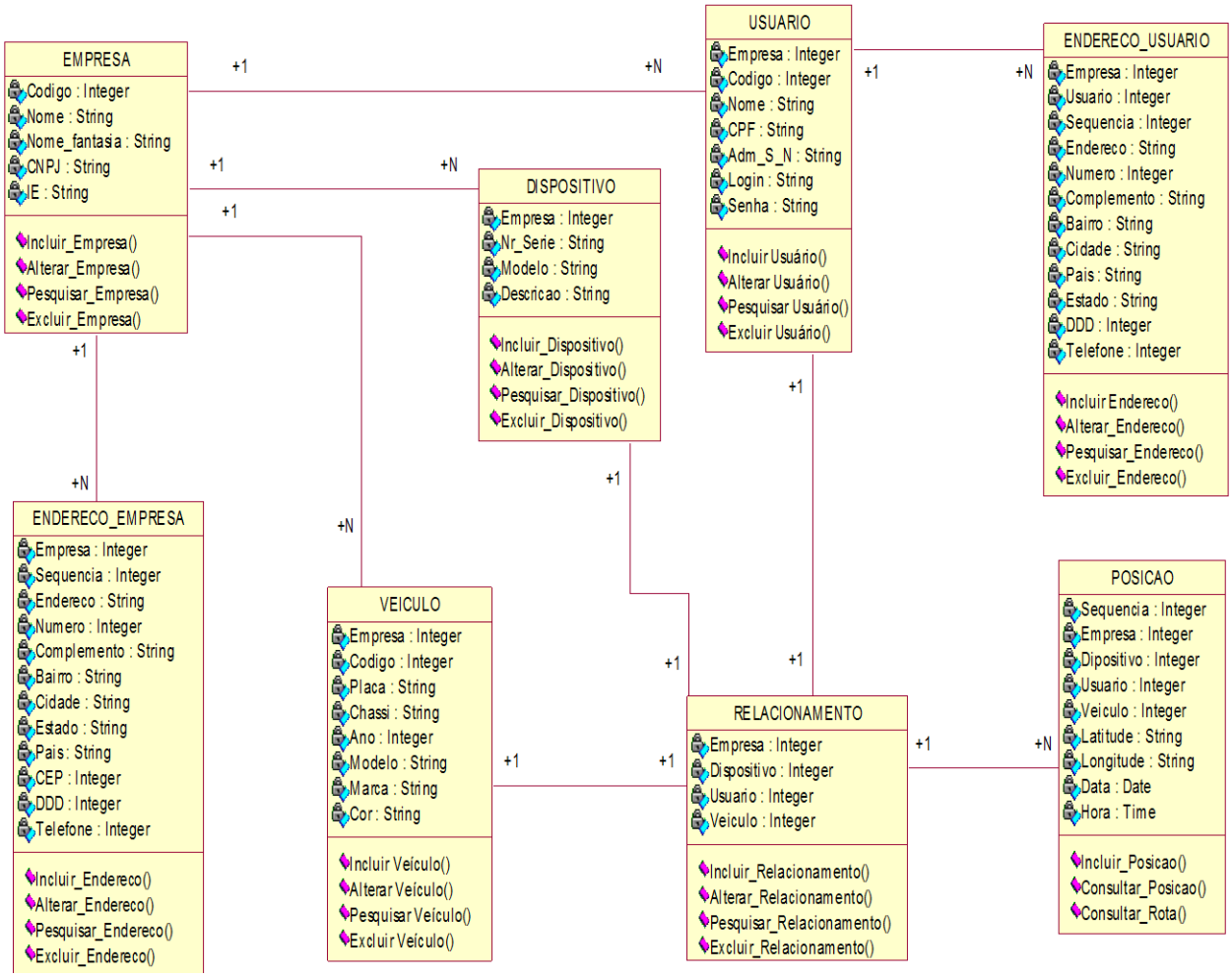

Figura 10 - Diagrama de Classe

A figura 3 mostra o diagrama de classes deste projeto. Onde para cada empresa pode-se ter vários usuários, vários veículos e vários dispositivos. Na classe relacionamento liga-se um dispositivo a um usuário e a um veiculo. A classe posição está ligada à classe RELACIONAMENTO.

Diagrama Entidade-Relacionamento

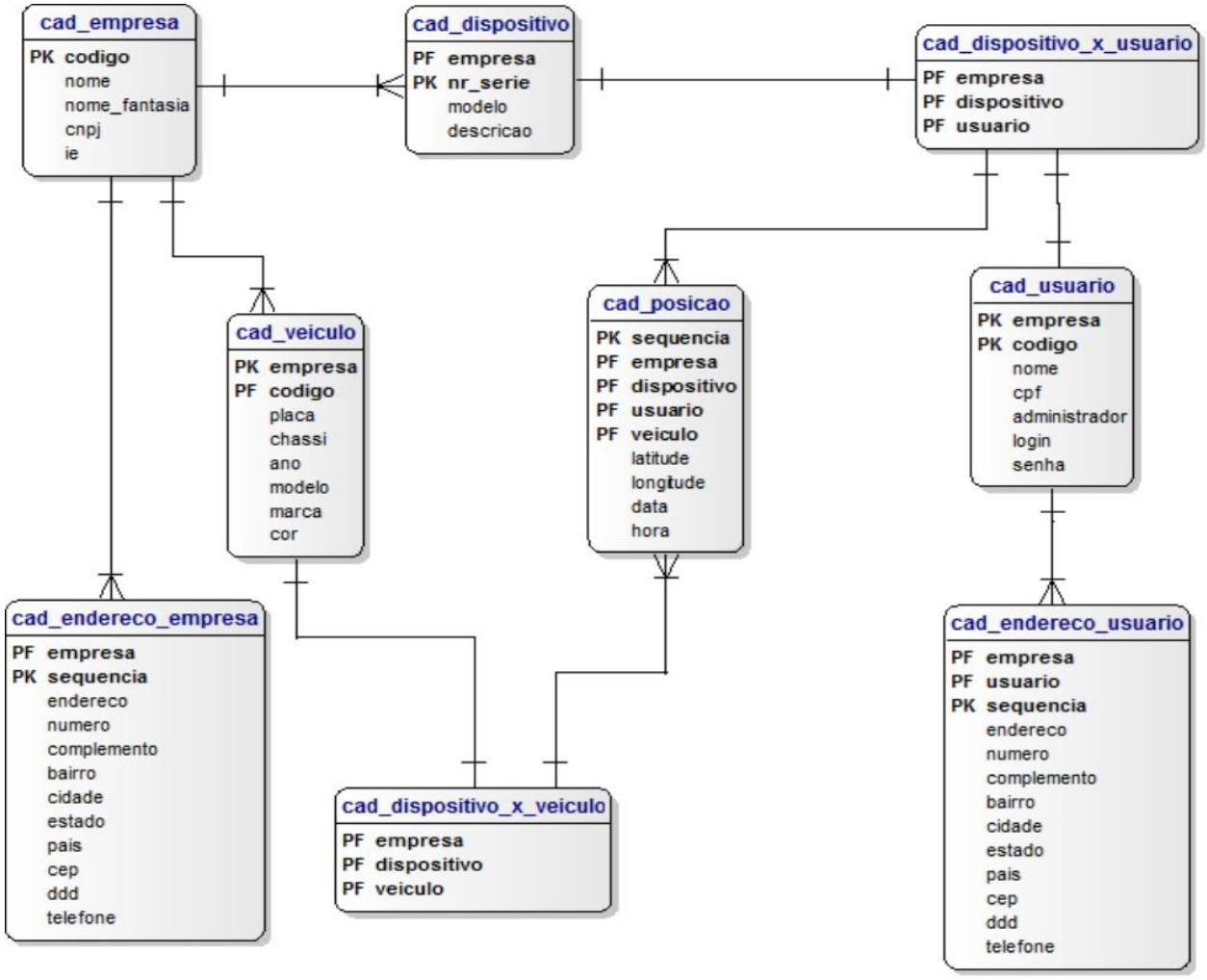

Figura 11 - Diagrama Entidade Relacionamento

A figura 4 apresenta o Diagrama de Entidade-Relacionamento deste projeto. A tabela empresa possui a chave primária código, onde todas as outras tabelas possuem a chave estrangeira empresa, que referencia esta tabela. Portanto pode-se ter várias empresas utilizando o mesmo sistema e a mesma base de dados.

Para cada empresa cadastrada na base de dados, podem-se cadastrar vários dispositivos, vários usuários, e vários veículos. No diagrama entidade relacionamento (figura 4), existem duas tabelas, cad_dispositivo_x_veiculo e cad_dispositivo_x_usuario, onde essas tabelas são derivadas da classe relacionamento na figura 3. Assim é possível relacionar um dispositivo com um usuário e com um veiculo.

A tabela posição é utilizada para guardar a informação de localização dos dispositivos, e através da classe relacionamento é inserido também o usuário e veículo.

SISTEMA CHAKAL

Neste projeto, o sistema está dividido em três módulos, a aplicação *mobile*, o banco de dados e a aplicação *WEB*. Estes módulos estão divididos em duas plataformas. Sendo essas plataformas um dispositivo com *SO Android* e um servidor com *SO Ubuntu* como mostra a figura 5.

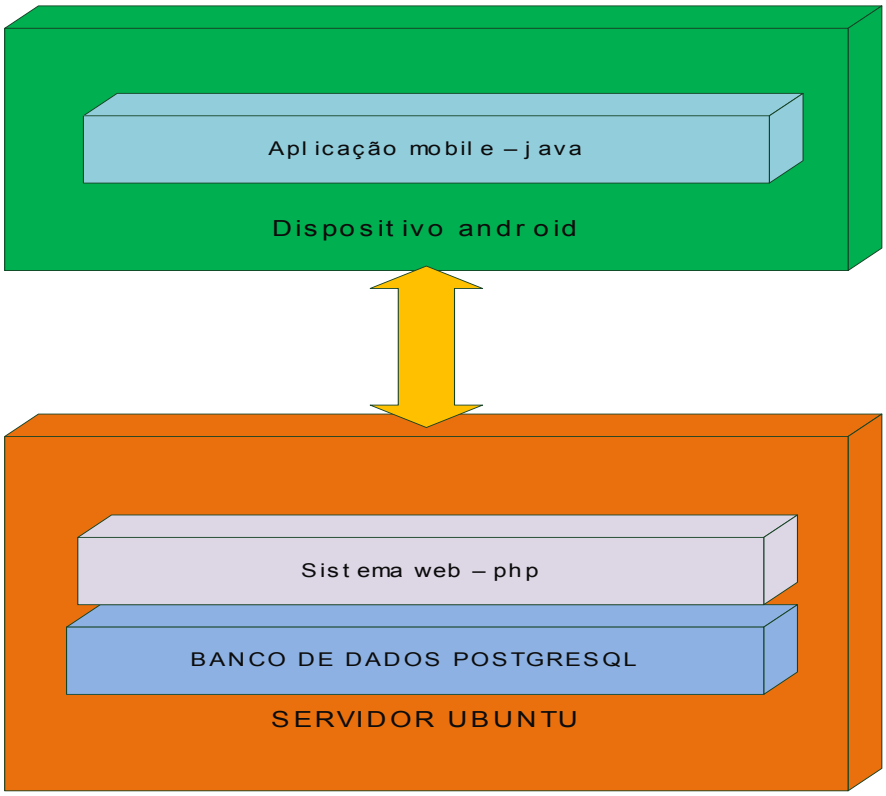

Figura 12 - Módulos do Sistema

Descrição do desenvolvimento funcional aplicação mobile

O módulo *mobile* é composto por uma única aplicação que ao iniciar, captura o *International Mobile Equipment Identity (IMEI)*, que nada mais é do que um número de serie com 15 algarismos para identificação de equipamentos móveis. Por se tratar de aplicação *Android* é necessário solicitar as permissões para utilizar recursos do dispositivo, neste caso para capturar o *IMEI*, é utilizada a seguinte permissão:

READ_PHONE_STATE

Esta permissão possibilita a leitura do estado do telefone, podendo assim capturar algumas informações, entre elas o *IMEI*, a localização do dispositivo (LACHETA, 2012).

Após capturar o *IMEI* do dispositivo a aplicação solicita ao *driver GPS* do dispositivo a localização do mesmo, para a utilização do *GPS* é necessário solicitar a seguinte permissão:

ACCESS_FINE_LOCATION

Esta permissão habilita o programa escrito a capturar a posição do aparelho utilizando a tecnologia *GPS* (LACHETA, 2012).

Após a captura de sua posição, a aplicação abre uma conexão com o banco de dados no servidor, faz a inserção da posição e fecha a conexão, esta conexão é feita utilizando a tecnologia 3G e/ou *Edge*. Esta conexão com a *Internet* é liberada pela permissão:

INTERNET

Esta permissão possibilita aos aplicativos abrir *sockets* de rede, e assim usufruírem da *Internet* (LACHETA, 2012).

A figura 6 mostra o método IniciarServico que utiliza a classe LocationManager, essa classe fornece acesso aos serviços de localização do sistema, a localização é dada na forma de latitude e longitude e para que possamos capturar as notificações de mudança de posição, utiliza-se a classe LocationListener. Toda vez que a localização é alterada o método Atualizar é chamado.

```
public void IniciarServico()
{
    LocationManager locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);


    LocationListener locationListener = new LocationListener() {
        public void onLocationChanged(Location location) {
          Atualizar(location);
        }

        public void onStatusChanged(String provider, int status, Bundle extras) {}

        public void onProviderEnabled(String provider) {}

        public void onProviderDisabled(String provider) {}
      };

      locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
}
```

Figura 13 - Trecho de Código para Localização do Dispositivo

A figura 7 mostra método Atualizar, toda vez que este método é chamado ele abre uma conexão com o banco de dados utilizando a classe DriverManager, com a conexão aberta, a variável sql recebe uma *string* contendo o comando a ser executado que é a chamada da função fnc_insere_posicao, o comando executeUpdate executa essa comando, e depois a classe close fecha a conexão com o banco de dados.

```
public void Atualizar(Location location)
{
    Double latPoint = location.getLatitude();
    Double lngPoint = location.getLongitude();
    TelephonyManager telephonyManager = (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
    resultArea.setText("'"+ latPoint.toString() +"', '"+ lngPoint.toString()+"'");

    try {
        Class.forName("org.postgresql.Driver");
    } catch (ClassNotFoundException e) {
        e.printStackTrace();

    }
    String url = "jdbc:postgresql://189.103.224.251:5432/dbchacal?user=postgres&password=FELIPE)%)*()";
    Connection conn;
    try {
         DriverManager.setLoginTimeout(5);
         conn = DriverManager.getConnection(url);
         Statement st = conn.createStatement();
         String sql;

         sql = "SELECT fnc_insere_posicao(" + telephonyManager.getDeviceId() + ", '" + latPoint.toString() + "', '" +lngPoint.toString() +"')";
         st.executeUpdate(sql);
         st.close();
         conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Figura 14 - Classe Responsável pela Conexão com o Banco de Dados

A figura 8 mostra a função fnc_insere_posicao, ela recebe os parâmetros número de *IMEI*, latitude e longitude. Após receber esses parâmetros ela seleciona uma nova sequencia, localiza qual empresa pertence esse dispositivo, qual o veículo e qual usuário e insere um registro com essas informações na tabela cad_posicao, com data e hora atual.

```
-- Function: fnc_insere_posicao(numeric, text, text)

-- DROP FUNCTION fnc_insere_posicao(numeric, text, text);

CREATE OR REPLACE FUNCTION fnc_insere_posicao(var_imei numeric, var_latitude text, var_longitude text)
  RETURNS text AS
$BODY$
  DECLARE
        var_sequencia integer;
        var_empresa integer;
        var_usuario integer;
        var_veiculo integer;
  BEGIN
     var_sequencia := (SELECT nextval('seq_posicao'));
     var_empresa   := (SELECT empresa FROM cad_dispositivo WHERE nr_serie = var_imei);
     var_usuario   := (SELECT usuario FROM cad_dispositivo_x_usuario WHERE empresa = var_empresa AND dispositivo = var_imei);
     var_veiculo   := (SELECT veiculo FROM cad_dispositivo_x_veiculo WHERE empresa = var_empresa AND dispositivo = var_imei);

     INSERT INTO cad_posicao(sequencia, empresa, dispositivo, usuario, veiculo, latitude,
            longitude, data, hora)
    VALUES (var_sequencia , var_empresa, var_imei, var_usuario, var_veiculo, var_latitude, var_longitude, CURRENT_DATE, CURRENT_TIME);

                RETURN 'OK';
        END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION fnc_insere_posicao(numeric, text, text)
  OWNER TO postgres;
```

Figura 15 - Função do Banco de Dados para inserção na tabela cad_posição

Descrição do desenvolvimento funcional aplicações *web*

A aplicação *WEB* é o responsável por inserir no banco de dados informações da empresa, dos dispositivos, dos veículos e usuários. Nesse módulo são feitas as consultas da posição ou localização dos dispositivos naquele momento, ou também, onde eles estiveram localizados em determinada data, dia e hora. É possível também, verificar a rota que estes dispositivos percorreram.

A aplicação *web* foi desenvolvida em linguagem *PHP* e, para a conexão com o banco de dados *PostgreSQL*, foi desenvolvida a classe cPostgreDB, que é a responsável por abrir uma conexão com o banco de dados, executar um comando, retornar uma resposta e encerrar esta conexão. A figura 9 exibe uma parte deste código.

```php
function cPostgreDB($DB="dbchacal", $Host="189.103.224.251", $PgPort=5432, $User="postgres",
                    $pass="FELIPE)%)*()", $persist=0)
{
    $this->host=$Host;
    $this->dbname=$DB;
    $this->username=$User;
    $this->password=$pass;
    $this->port=$PgPort;
    $this->persistent=$persist;
    $this->Connect();
}

function Connect()
{
    $connect="host=".$this->host." port=".$this->port." dbname=".$this->dbname." user=".$this->username;
    if (!empty($this->password))
        $connect.=" password=".$this->password;
    if ($this->persistent)
        $this->dbconnect= pg_pconnect($connect);
    else
        $this->dbconnect= pg_pconnect($connect);
    if (!$this->dbconnect)
        $this->error="cannot connect to database ".$this->dbname;
}

function fechaConn()
{
    pg_close($this->dbconnect);
}
```

Figura 16 - Classe responsável pela conexão com PostgreSQL

Para iniciar a utilização do sistema *Chakal*, é necessário que o responsável pela empresa faça o cadastro no sistema. Feito isso ele também se cadastrou como usuário administrador desta empresa. Depois de feito este cadastro, este usuário pode começar a utilizar o sistema. Para acessar o sistema ele deve se autenticar na tela de *login* do sistema, como mostra a figurar 10.

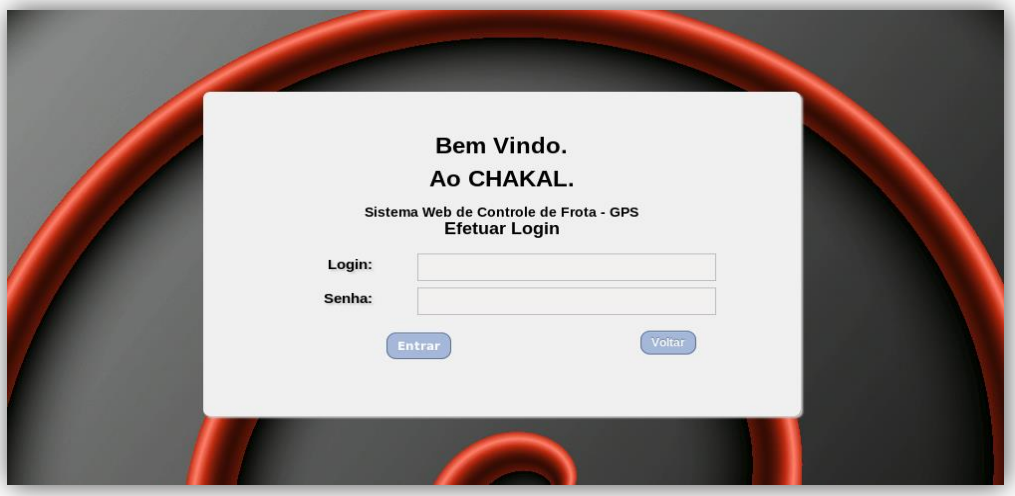

Figura 17- Tela de Login do Sistema

O sistema conta com o cadastro de dispositivos, usuário que no caso será um funcionário da empresa, ou seja, este usuário é um funcionário da empresa, conta também com o cadastro de veículos, cujos veículos pertencem à empresa.

Como todo o monitoramento é feito a partir de um dispositivo com o sistema *Android* instalado, é necessário relacionar o dispositivo com um usuário e/ou com um veículo como mostra a figura 11.

Figura 18 - Relacionamento de Dispositivos

A partir do momento em que o dispositivo começar a gravar sua localização, será possível localizar o dispositivo e lendo este relacionamento sabemos qual o veículo e qual o usuário que está utilizando este dispositivo e, a partir da latitude e longitude, o sistema exibe sua posição através de um mapa, neste caso, utilizando a *API* do *Google Maps*. Utilizando os seguintes comandos:

GOOGLE.MAPS.LATLNG

Esta classe recebe os valores de latitude e longitude que serão utilizados no mapa.

MYOPTIONS

Classe que tem a função de criar o controle sobre este mapa como "zoom" (nivel de proximidade da superfície), "center"(localização do centro do mapa exibido) e "mapTypeId" (tipo de mapa exibido) entre outros parâmetros.

Os tipos de mapas que podem ser escolhidos são:

- ROADMAP exibe as blocos 2D normais, padrão, do *Google Maps*.
- SATELLITE exibe blocos fotográficos.
- HYBRID exibe uma mistura entre blocos fotográficos e uma camada de blocos com recursos importantes (estradas, nomes de cidade).
- TERRAIN exibe blocos de relevo físico para exibição de recursos de elevação e água (montanhas, rios etc.).

GOOGLE.MAPS.MAP

Esta é a classe responsável por criar o mapa, utilizando os parâmetros informados na opção “mapTypeId” na classe “myOptions”.

GOOGLE.MAPS.MARKER

Classe responsável por criar um marcador sobre o mapa exibido.

GOOGLE.MAPS.POLYLINE

Classe utilizada para criar polilinhas a partir de latitudes e longitudes, recebidas a partir da classe “google.maps.LatLng”, formando assim uma linha sobre o mapa (API, 2012).

A figura 12 mostra o trecho que cria o mapa e exibe a posição do dispositivo utilizando os comandos citados, criando um marcador no mapa. Nesta figura vemos que para capturar a posição do equipamento, é feito uma consulta no banco de dados através do objeto “retornaLocalização”, que retorna a latitude e longitude, passando os valores para a classe

“google.maps.LatLng”.

```
function initialize() {
    var latlng = new google.maps.LatLng(
    <?
        include_once('conexao.php');
        $dados = new cPostgreDB();

        $retDados = $dados->retornaLocalizacao($var_empresa, $var_dispositivo);

        while ($retorno = pg_fetch_array($retDados))
        {
            $var_posicao = $retorno['local'];
        }
        echo $var_posicao;
    ?>);
    var myOptions = {
                    zoom: 15,
                    center: latlng,
                    mapTypeId: google.maps.MapTypeId.TERRAIN};

                    var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);


    var marker = new google.maps.Marker({
                                        position: latlng,
                                        title:""});
    marker.setMap(map);
}
```

Figura 19- Código PHP exibindo a posição do dispositivo

TESTE FUNCIONAL

Resultado esperado

O objetivo deste trabalho é possibilitar de maneira eficaz o controle de uma frota de veículos de uma empresa, por tanto é esperado que o sistema proporcione aos responsáveis pela empresa um total controle sobre seus veículos.

A proposta deste projeto é o monitoramento em tempo real de uma frota de veículos, sobre a visualização na forma de mapa.

É proposto também por este projeto disponibilizar para os administradores da empresa, a possibilidade de consultar rotas realizadas por um veiculo, e consultar onde o mesmo esteve em determinado dia e hora.

Cenário de teste

Os testes devem ser realizados, tendo um fluxo de dados de entrada, e seus resultados devem ser avaliados. Deste modo os resultados obtidos nos testes são comparados com os resultados esperados (PRESSMAN, 2005).

A entrada de dados deve obedecer duas primícias essenciais:

Minimize o número de ações de entrada exigidas do usuário, facilitando os processos de entrada de dados.

Mantenha a consistência entre a exibição das informações e entrada de dados, garantindo que o dado informado pelo usuário seja cadastrado no banco de dados (PRESSMAN, 2005).

O cenário proposto para o teste deste projeto é a criação de uma empresa fictícia dentro do sistema, após este processo o próximo passo é o cadastro de alguns usuários e em seguida cadastrar alguns veículos e os dispositivos. Com os cadastros realizados deve-se cadastrar o relacionamento dos dispositivos com os usuários e com os veículos.

Neste ponto deve ser feita uma verificação nas informações cadastradas no banco de dados, validando e verificando se realmente são as informações inseridas pelo usuário para garantir a integridade dos dados e a confiabilidade do sistema (GUERRA, 2009).

Neste momento é preciso que a aplicação mobile seja executada, e que os veículos se movimentem pelas ruas, a fim de verificarmos se o dispositivo está transmitindo sua localização para o sistema e se esta informação é realmente sua localização.

Os sistemas que trabalham em tempo real geram certa ação em resposta a eventos externos. Para cumprir essa função, o sistema deve realizar o controle e aquisição de dados em alta velocidade sob severas restrições de tempo e confiabilidade (PRESSMAN, 2005).

É necessário verificar se a posição exibida pelo sistema é a posição em tempo real de cada dispositivo e se as rotas traçadas pelo sistema são as rotas percorridas pelos dispositivos.

As informações apresentadas ao usuário devem ser completas, legíveis, satisfazendo as necessidades do usuário. As informações exibidas devem ser relevantes ao contexto atual e de fácil interpretação (PRESSMAN, 2005).

Para avaliação do sistema desenvolvido neste projeto, o teste contará com o cadastro de 5 empresas, onde para cada uma dessas empresas será cadastrado 1 administrador, 4 usuários, 4 veículos e 4 dispositivos. Com esses cadastros realizados será necessários relacionar cada dispositivo com 1 usuário e 1 veículo.

Neste momento será avaliado como o sistema trata a divisão de empresas, onde um administrador de uma empresa não pode ter acesso a informações de outra empresa.

Para esta avaliação 4 veículos de uma empresa deverão sair ao mesmo tempo e se movimentar pela cidade, a cidade onde os testes foram realizados é a cidade de Jundiaí no estado de São Paulo.

Resultados obtidos

No teste realizado foi possível cadastrar as empresas no sistema, com seus administradores, usuários, dispositivos e veículos. Também foi efetuado o relacionamento entre esses cadastros.

O sistema tratou corretamente a divisão entre empresas, ou seja, cada administrador só conseguiu ver informações referentes à sua empresa, e não conseguiu ver informações de outras empresas.

O *software* desenvolvido e instalado no dispositivo com *Android* realizou sua tarefa corretamente, incluindo as informações de latitude e longitude corretamente no banco de dados, sendo transmitido a localização na média de 3 registros por segundo.

A aplicação que exibe em tempo real a localização de cada dispositivo exibiu corretamente a posição, como mostra a figura 13.

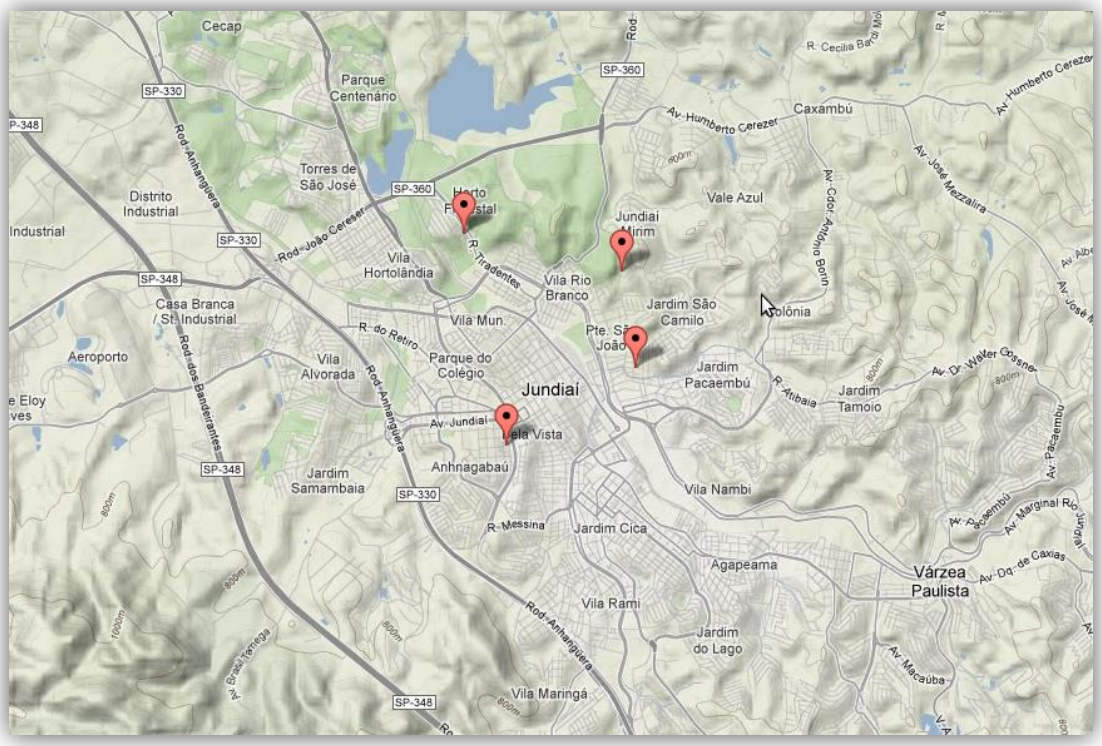

Figura 20 Localização atual dos dispositivos.

O sistema também exibiu de maneira correta as rotas percorridas por cada dispositivo. Foi possível ver a rota de cada dispositivo em um determinado período, ou a rota percorrida por vários dispositivos em um determinado período, como exibido na figura 13.

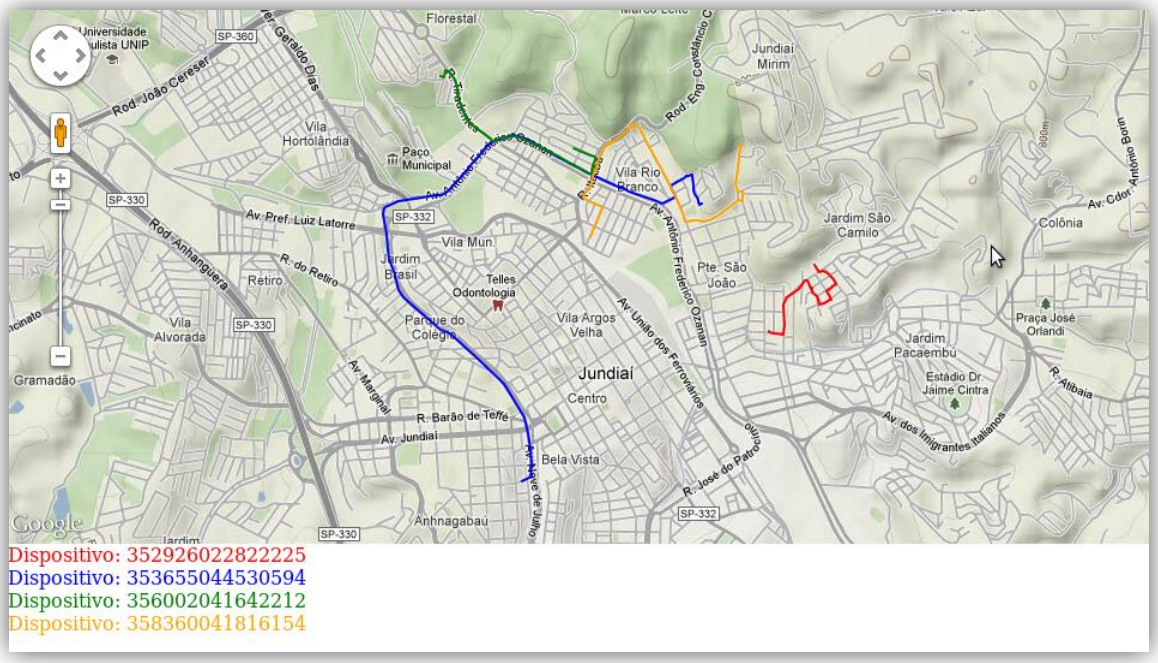

Figura 21- Rota realizada pelos dispositivos

Ao realizar os testes foi constatado que é possível monitorar não somente os veículos como também os funcionários, isso devido ao rastreamento ser feito a partir do dispositivo e o mesmo pode ser relacionado com um veículo e com um funcionário, sendo obrigatório ser relacionado com pelo menos um dos dois e não com os dois ao mesmo tempo.

CONCLUSÃO

Com base nos estudos feitos para o desenvolvimento deste projeto, concluí-se que nos dias de hoje está cada vez mais difícil manter o controle sobre os veículos de uma empresa, tornando assim necessário o auxílio de ferramentas para controlar os veículos pertencentes a uma empresa.

Com base nas informações levantadas no decorrer deste projeto, conclui-se que o sistema *web* desenvolvido neste projeto, utilizando as ferramentas e tecnologias entre elas o *GPS* e dispositivos com sistema *Android*, possibilita ao administrador de uma empresa controlar sua frota de veículo em tempo real, obtendo a informação de onde estão seus veículos, proporcionando a tomada de decisão do administrador mais rápida e baseada em informações reais. Além de possibilitar a consulta em tempo real da localização de um veículo, é possível

consultar as rotas percorridas pelos mesmos, sabendo em quais locais cada veiculo passou, e quando foi à chegada dele em determinado ponto.

O protótipo desenvolvido neste projeto destaca-se pela inserção direta de informação no banco de dados pelo dispositivo *Android,* deste modo à inserção e visualização da informação torna-se mais rápida e precisa quanto à visualização em tempo real. Outro ganho deste projeto é o desenvolvimento do sistema sobre a plataforma *PHP*, por se tratar de uma linguagem *WEB*, possibilita ao administrador visualizar as informações de sua frota de qualquer local onde ele estiver. E como o dispositivo utilizado neste projeto utiliza o sistema *Android*, também é possível que o administrador visualize as informações pelo próprio dispositivo.

Após os testes realizados concluímos que o sistema proposto neste trabalho atende as necessidades das empresas nos dias de hoje. Os testes mostraram que se pode ter várias empresas utilizando a mesma base de dados, pois o sistema consegue separar as informações por meio do código de cada empresa, sem exibir informações erradas. Foi verificado que a aplicação *mobile* fez seu papel corretamente, inserindo sua localização corretamente na base de dados. E enquanto o dispositivo se movia pela cidade o sistema exibia corretamente sua posição atual. Outro ponto interessante que foi constatado com base nos teste é que é possível fazer o rastreamento não somente de um veículo, mas também de um usuário, pois o relacionamento entre dispositivo, veículo e usuário, não é obrigatório o relacionamento entre os três, e sim apenas relacionar o dispositivo com pelo menos um entre os dois, veículo ou usuário.

Com o término deste projeto pretende-se buscar parceria com empresas para a implantação deste sistema e continuar com o desenvolvimento do mesmo, agregando funcionalidades como possibilitar ao administrador a inclusão e alteração dos cadastros via *smartphones*. Após o termino pretende-se também desenvolver um módulo do sistema para *tablets*, tendo a funcionalidade de enviar a posição assim como é feito hoje nos celulares (dispositivos), mas agregando a ele a funcionalidade de navegador *GPS*, com opções de informação ao administrador de algumas situações que o mesmo possa estar passando como um assalto, problema com os veículos, e outras situações.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID. Disponível em < http://www.cesar.org.br/site/files/file/WM18_Android.pdf/>, acesso em 25/08/2012.

API. Disponível em < https://developers.google.com/maps/?hl=pt-br/>, acesso em 03/11/2012.

AUTOTRAC. Disponível em <http://pt.wikipedia.org/wiki/Autotrac/>, acesso em 28/03/2012.

GUERRA Ana Cervigni; COLOMBO Regina Maria Thienne S. *Tecnologia da Informação: Qualidade de Produto de Software.* Brasília: PBQP Software, 2009. 429 p.

JAVA. Disponível em < http://www.java.com/pt_BR/about/>, acesso em 25/08/2012.

LACHETA, Ricardo R. *Google Android*. 2ª Edição. São Paulo: Novatec, 2010. 608p.

LANGENDOLFF, A.; PELLEGRINI, G. Fundamentos de cartografia e o sistema de posicionamento global - GPS, Santa Maria, p. 30-34, 2008.

MAPS. Disponível em < https://developers.google.com/maps/?hl=pt-br/>, acesso em 28/10/2012.

PHP. Disponível em <http://pt.wikipedia.org/wiki/PHP#Visibilid/>, acesso em 25/08/2012.

POSTGRESQL. Disponível em < http://www.postgresql.org.br/sobre/>, acesso em 25/08/2012.

PRESSMAN Roger S. *Software Engineering: A Practitioner's Approach*. 6ª Edição. New York: Mc Graw Hill, 2005. 880 p.

SEGANTINE, Paulo Cesar Lima. *GPS – Sistema de Posicionamento Global*. São Carlos: EESC/USP, 2005. 364p.

SILVA Alberto Manuel Rodrigues; VIDEIRA Carlos Alberto Escaleira. *UML Metodologias e Ferramentas CASE*. Porto: Centro Atlântico, 2001. 552 p.

UBUNTU. Disponível em < http://www.ubuntu-br.org/ubuntu/>, acesso em 02/09/2012.