
**CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB**

**CURSO DE ENGENHARIA DE COMPUTAÇÃO**

**LEANDRO NOGUEIRA DE SOUSA**

# RASTREAMENTO DE VEÍCULO POR GPS

**Orientador: Prof. M.C. Maria Marony Sousa Farias**

Brasília

Dezembro, 2013

**LEANDRO NOGUEIRA DE SOUSA**

# RASTREAMENTO DE VEÍCULO POR GPS

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof. M.C. Maria Marony Sousa Farias

Brasília

Dezembro, 2013

**LEANDRO NOGUEIRA DE SOUSA**

**RASTREAMENTO DE VEÍCULO POR GPS**

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof. M.C. Maria Marony Sousa Farias

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas -FATECS.

_____

Prof. Abiezer Amarilia Fernandes

Coordenador do Curso

**Banca Examinadora:**

_____

Prof. Maria Marony Sousa Farias, Mestre em Engenharia Elétrica – UFPB - PB.

Orientadora

_____

Prof. Cléber da Silva Pinheiro, Doutor.

_____

Prof. Flávio Klein, Mestre.

_____

Prof. Fernando Chagas, Mestre.

Dedico este trabalho a minha família,

especialmente a minha mãe Rensi Nogueira Porpino,

por proporcionar toda a minha educação.

## AGRADECIMENTOS

Agradeço a todos que de alguma forma colaboraram agregando conhecimento para que se pudesse realizar este trabalho.

Agradeço minha família, especialmente minha mãe Rensi Nogueira Porpino e minha companheira Sabrina de Jesus Santana, por terem me apoiado durante meus estudos.

Por fim agradeço aos amigos da empresa Trix Tecnologia Inteligente por todos os conhecimentos que adquiri e que foram muito importantes na elaboração deste trabalho. Um agradecimento especial a Evandro Moura e Marcos Soares, que foram duas pessoas com quem aprendi muito.

**Resumo**

O projeto desenvolvido traz uma solução para rastreamento de qualquer tipo de veículo, desde que tenha um aparelho rastreador GPS com conexão à Internet. No projeto foi utilizado um modelo especifico de aparelho GPS, mas pode-se usar outros modelos sem que haja modificação no código do sistema desenvolvido desde que a forma de comunicação do aparelho seja similar a do que foi utilizado no projeto. Foi desenvolvido um sistema em Web com as linguagens de programação Java e JavaScript. Esse sistema recebe as coordenadas enviadas pelo rastreador GPS através da Internet, utilizando a rede de telefonia móvel. O sistema interpreta essas coordenadas, as armazena em um banco de dados. Ao acessar o sistema o usuário pode ver as coordenadas de onde determinado veículo esteve, sua velocidade e a data e hora em que o veículo esteve em determinado local.

Palavras Chave: GPS, Java, JavaScript, coordenadas.

**Abstract**

The project developed brings a solution for tracking any type of vehicle, providing you have a GPS tracker device with an Internet connection. In designing a specific model GPS device was used, but you can use other models without any modification to the code system developed since the form of the communication device is similar to what was used in the project. A system has been developed with Web languages Java and JavaScript programming. This system receives the coordinates sent by GPS tracker via the Internet using the mobile phone network. The system interprets these coordinates, stores them in a database. By accessing the system the user can see the coordinates of where the vehicle has been determined, its speed and the date and time that the vehicle has been in a certain place.

Keywords: GPS, Java, JavaScript, coordinates.

**Sumário**

## Lista de Figuras

## Lista de Siglas

API: Application Programming Interface

ASCII: American Standard Code for Information Interchange

Baud-rate: É o número de vezes em que um sinal muda ou varia seu estado em um canal de comunicação. (http://www.tech-faq.com/difference-between-bit-rate-and-baud-rate.html, acessado em 15/04/2013).

BPM: Business Process Management

EJB: Enterprise JavaBeans

GPS: Global Positioning System

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

IDE: Integrated Development Environment

IMEI: International Mobile Equipment Identity

JPA: Java Persistence API

JSF: JavaServer Faces

JSON: JavaScript Object Notation

JSP: JavaServer Pages

JSR: Java Specification Requests

JVM: Java Virtual Machine

ORM: Object-relational Mapping

POJO: Plain Old Java Objects

SIM: Subscriber Identity Module

SQL: Structured Query Language

SMS: Short Message Service

TCP/IP: Transmission Control Protocol/Internet Protocol

XHTML: Extensible HyperText Markup Language

XML: Extensible Markup Language

# Capítulo 1 - Introdução

## 1.1 - Apresentação do Problema

A tecnologia vem sendo cada vez mais adotada no dia-a-dia das pessoas. Com o passar do tempo são desenvolvidos sistemas de segurança cada vez mais sofisticados. Muitas empresas investem em sistemas de monitoramento como câmeras de vigilância e rastreamento por GPS.

Com um sistema de rastreamento por GPS pode-se saber onde está o objeto rastreado. Normalmente o rastreador GPS fica em um veículo, já que é um bem de valor relativamente alto, dependendo de marca e modelo.

Com o número de veículos aumentando a cada dia, também cresce o número de roubos e furtos a veículos. Uma das soluções para esse problema é a contração de um seguro para o veículo. Para muitos essa solução não se aplica devido ao valor do seguro. Esse valor pode variar de acordo com vários fatores como idade, local onde vive entre outras coisas. Mas geralmente não é um valor baixo. Em uma pesquisa realizada para cotação de seguros em várias seguradoras para um homem de 45 anos, casado em São Paulo para um veículo de até 30 mil reais o menor valor cotado foi de R$ 1.167,52 e o maior valor foi de R$ 2.136,54. (Wiltgen, Júlia, 2013)

Apenas de janeiro a junho de 2010 foram registrados 191.347 roubos ou furtos de carros no Brasil de acordo com uma pesquisa da CNSeg (Confederação nacional da Empresas de Seguros Gerais) com bases nos dados do Denatran. (Departamento Nacional de Trânsito). Em alguns estados como Rio Grande do Sul, Santa Catarina a porcentagem de recuperação desses veículos é superior a 60%. Mas em outros estados a porcentagem de recuperação desses veículos é de apenas 30%. (DENATRAN, 2013).

O sistema requer um servidor que esteja funcionando e conectado a internet durante todo o tempo. Para isso seria necessário pelo menos duas conexões com a internet, caso uma parasse de funcionar a outra entraria em funcionamento sem que o usuário percebesse o problema.

A capacidade de armazenamento do sistema também é outro ponto crítico. Além de aumentar gradativamente com o tempo, também deve aumentar de acordo com a quantidade de usuários no sistema. Apesar do tamanho da informação gravada no banco de dados ser

pequena, o número de gravações realizas no banco é contínuo. A cada 20 segundos um novo registro é gravado no banco de dados, esse tempo pode ser configurado. Caso o sistema tenha 1000 usuários tem-se 3000 registros novos gravados a cada minuto no banco de dados.

Outro problema refere-se à cobertura GSM da operadora. Caso o veículo vá para uma localização em que o rastreador GPS não consiga enviar as informações para o servidor essas informações poderão ser pedidas. Para se tentar evitar esse problema alguns rastreadores GPS possuem um *slot* para cartão SD. Assim, quando o rastreador perde a conexão com o servidor, ele começa a armazenar no cartão as localizações. Quando o conexão é restabelecida o rastreador envia os dados armazenados para o servidor.

## 1.2 - Objetivos do Trabalho

O objetivo geral do trabalho é a elaboração de um sistema web em que qualquer pessoa com acesso a Internet e com um aparelho de rastreamento GPS cadastrado no sistema possa saber a localização do seu veículo.

Os objetivos específicos do trabalho são:

- Primeiro desenvolver um sistema web em que o usuário possa acessar através de um navegador web e fazer o seu cadastro.
- O segundo objetivo específico é integrar o rastreador GPS ao sistema, assim o sistema está apto a receber e gravar as informações enviadas pelo rastreador.

## 1.3 - Justificativa e Importância do trabalho

Com o número de veículos nas ruas aumentando a cada dia também aumenta o número de furtos e roubos a veículos, por isso a procura por este tipo de sistema aumenta cada vez mais.

O rastreamento de veículo por GPS também vem sendo adotado cada vez mais por empresas que possuem uma frota de veículos como empresas de transporte de carga, empresas de viagem, taxistas entre outros.

Com o rastreamento por GPS, é possível monitorar o caminho de determinado veículo, saber todo o trajeto percorrido durante um intervalor de tempo, monitorar em tempo real a localização do veículo. Saber em quanto tempo um trajeto foi percorrido.

Outro fator importante é o valor que se paga para ter seu veículo monitorado que é relativamente baixo comparado com o preço de um seguro. Essas são algumas vantagens de se utilizar um sistema que monitora veículos.

## 1.4 – Escopo do Trabalho

O trabalho tem como escopo um sistema capaz de cadastrar proprietários e seus veículos com rastreadores GPS e tem como principal função a visualização das localizações em um mapa no navegador do usuário. Esse mapa mostra a localização do veículo com precisão que pode variar de acordo com o rastreador utilizado.

## 1.5 – Resultados Esperados

Os resultados esperados são que o sistema web esteja eficiente. Atenda o seu propósito de rastrear veículos e atenda as expectativas dos usuários.

Também se espera que o sistema esteja preparado para que possa ter vários usuários usando-o simultaneamente em um ambiente com alta concorrência de recursos. Quanto maior o número de usuários no sistema maior deve ser o investimento na infraestrutura do sistema.

## 1.6 – Estrutura de Trabalho

A elaboração deste projeto foi dividida em três capítulos, conforme descrição:

Capítulo 1 – Introdução: Neste capítulo será apresentado o problema que motivou a elaboração desse trabalho. Descrevendo alguns fatores que levam ao problema apresentado, métodos para resolução do problema, além de descrever com clareza como o projeto desenvolvido ajuda na solução do problema.

Capítulo 2 – Referencial Teórico: Neste capítulo será descrita toda a teoria utilizada para elaboração do projeto. Descrevendo quais ferramentas foram utilizadas e porque tais ferramentas foram utilizadas. Os métodos utilizados para resolução do problema apresentado.

Capítulo 3 – Modelo proposto: Neste capitulo será descrito como foi aplicada a teoria apresentada, como foram utilizadas as ferramentas apresentadas. Detalhando cada etapa do processo da elaboração do projeto.

Capítulo 4 – Implementação e Aplicação: Neste capítulo será descrito como o projeto foi implementado. Mostrando como foram realizados os testes. Detalhando as dificuldades e as soluções para cada tarefa do projeto.

Capítulo 5 – Conclusão: Neste capítulo serão descritas as conclusões do projeto. Quais foram os resultados obtidos do projeto, incluindo o que foi feito com sucesso. Os problemas enfrentados durante a elaboração do projeto, descrevendo como evitar os problemas encontrados.

# Capítulo 2 – Referencial Teórico

## 2.1 – Rastreamento por GPS

O Sistema de Posicionamento Global foi desenvolvido pelo Departamento de Defesa dos Estados Unidos, para ser usado como sistema de navegação pelo exército americano. O GPS foi adota amplamente para uso civil em diversas aplicações.

O sistema GPS opera com 24 satélites a aproximadamente 20.200 km da superfície da terra. Os satélites são distribuídos em seis órbitas diferentes de forma que sempre haja pelo menos quatro visíveis em qualquer ponto do planeta.

Com apenas três satélites é possível saber a localização do usuário, calculando a distância da antena do usuário a dos satélites, a quarta medida é necessária devido a não sincronização dos relógios do satélite com o usuário. Na figura 2.1 é mostrado um exemplo de como o sistema funciona. Cada satélite está sincronizado com três aparelhos GPS, cada um em seu respectivo veículo.

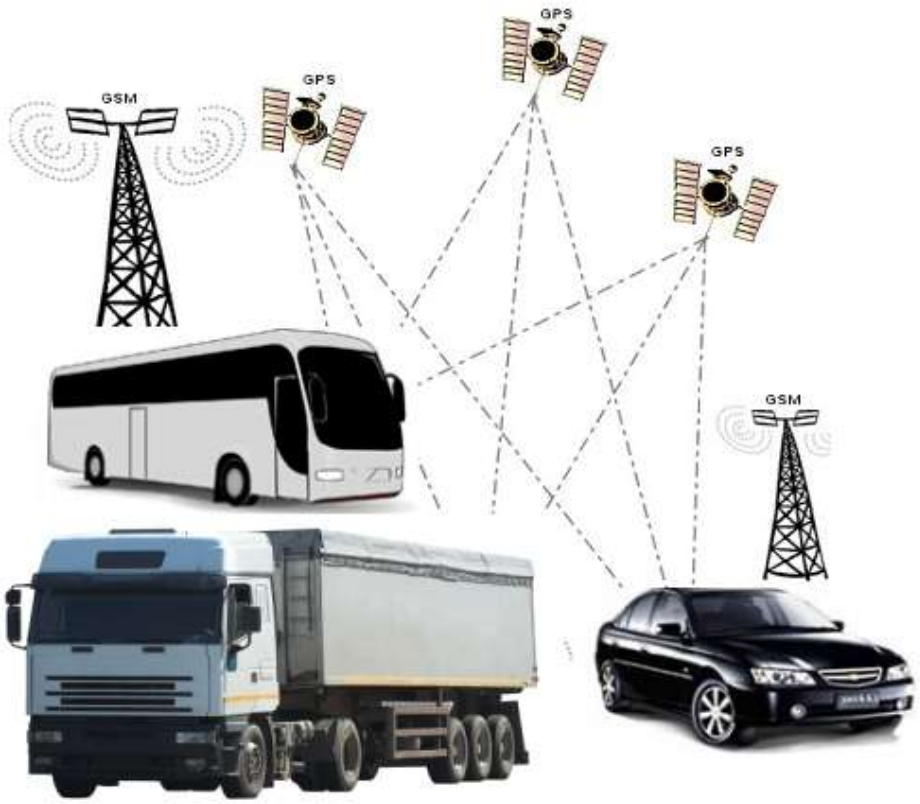


**Figura 2.1 – Rastreamento GPS**

Com o uso crescente de dispositivos eletrônicos de navegação e com várias empresas oferecendo seus produtos no mercado foi necessária a padronização da comunicação entre esses equipamento. Essa padronização foi realizada pelo NMEA (National Marine Eletronics Association). Foi criada a especificação NMEA 0183.

A NMEA 0183 especifica que a comunicação utiliza os caracteres ASCII, com comunicação serial com baud-rate de 4800, 8 bits por símbolo, um ou mais bits de parada, sem paridade e sem *handshake.* (UFRJ, 2013).

As informações devem ter a seguinte forma:

$ttsss,d1,d2,d3,....<CR><LF>

$GPRMC,211232,A,2230.9860,S,04405.9376,W,2.3,24.3,190408,21.2,W,A*0C<CR><LF>

tt: Abreviatura do fabricante do equipamento.

sss: Tipo de informação a ser transmitida.

d1,d2,d3: São dados da informação.

Exemplo: RMC – Informação mínima de navegação recomendada – tipo C.

2111232: Hora global.

A: Informação válida. Caso o GPS não esteja reconhecendo nenhum satélite é transmitido V.

2230.9860 = 22° 30.986’ de latitude.

S: Hemisfério Sul.

04405.9376 = 44° 5.9376 de longitude.

W: Lado Oeste.

**2.2 – Linguagem a Orientação a objetos**

Objeto é uma entidade do mundo real que tem uma identidade. Objetos podem representar entidades concretas ou conceituais. Na linguagem de programação, cada objeto precisa de um mecanismo de identificação que deve ser única, uniforme e independente do conteúdo do objeto.

Um objeto possui atributos e comportamentos que formam a estrutura do objeto, objetos com a mesma estrutura são agrupados em classes. Uma classe é a abstração de um objeto que será importante dentro da aplicação. Cada objeto é uma instância de uma classe, assim cada instância pode ter valores de atributos diferentes. (Unicamp, 2013)

Linguagens orientadas a objeto vieram com o objetivo de tornar o desenvolvimento de software mais rápido e econômico. A linguagem orientada a objetos mais utilizada atualmente é o Java com cerca de 17,3% de utilização no mercado. (TIOBE Software BV, 2013).

**2.3 – Java**

A linguagem de programação utilizada no desenvolvimento do projeto é o Java, por ser umas das linguagens mais adotadas atualmente em desenvolvimento de aplicações web com várias opções de framework que agilizam o desenvolvimento além de possuir uma comunidade bastante ativa.

O Java surgiu em 1991 quando a Sun Microsystems financiou um projeto chamado Green que resultou na linguagem de programação Java. Com a explosão da World Wide Web em 1993 a Sun viu o potencial do Java para trazer conteúdo dinâmico as páginas da Web. (Deitel, Java como programar).

Em 1996, a Sun lançou a primeira versão do Java, o Java 1.0, que era bastante limitado. Em 1998, a Sun lançou o Java2 *Standard Edition* e duas outras versões: a *Micro Edition* para dispositivos embutidos e a *Enterprise Edition* para processamento no lado do servidor.

A versão 5.0 lançada em 2004 foi a primeira versão desde a 1.1 em que a linguagem sofreu uma grande atualização. No fim de 2006 foi lançada a versão 6 do Java com melhorias adicionais de desempenho e aprimoramentos de biblioteca. (Horstmann, Core Java, Volume 1: Fundamentos).

A linguagem Java é utilizada principalmente para desenvolvimento de aplicações Web com as especificações Java EE (Java *Enterprise Edition*). Java EE é uma especificação bem determinada que padroniza a implementação de software. Há implementações de vários fabricantes e desde que ela siga o padrão Java EE poderá ser substituída por qualquer outro software que também siga a implementação. (Caelum, 2013).

Em um ambiente de desenvolvimento típico do Java, a primeira coisa a se fazer e desenvolver as classes que possuem extensão *.java*. Então o compilador cria bytecodes e os armazena em arquivos com extensão *.class*. O carregador de classe lê esses arquivos colocando-os na memória. O verificar de arquivos confirma que todos os bytecodes são válidos e não violam restrições de segurança do Java. Por último, a JVM lê os bytecodes e os converte para linguagem de máquina. Conforme mostrado na figura 2.2.

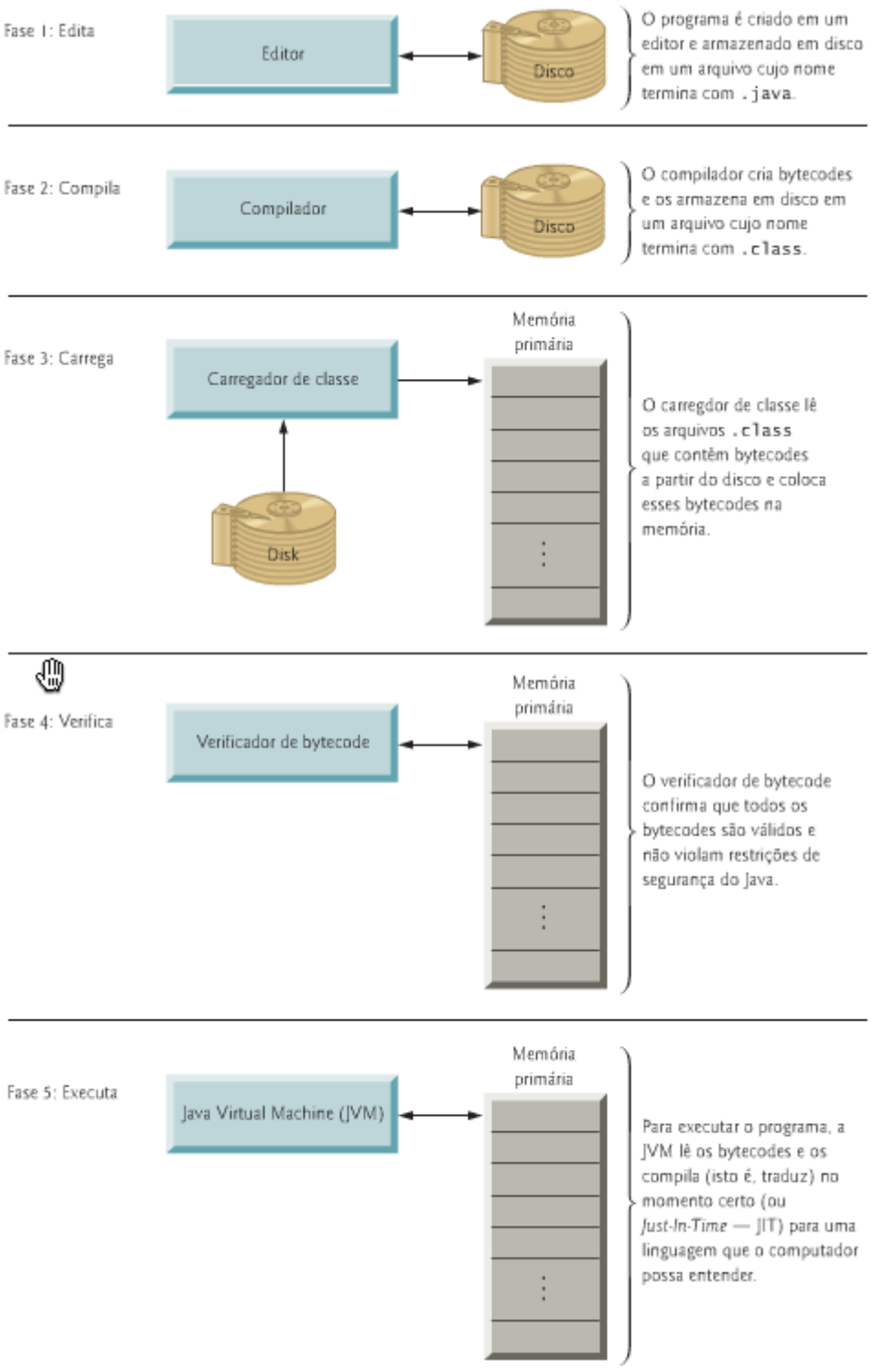


**Figura 2.2 - Ambiente de desenvolvimento Java. Fonte: Deitel, Java como programar**

As instruções *bytecode* geradas pelo compilador são independentes de arquitetura, elas podem ser interpretadas em qualquer máquina e instantaneamente convertidos em código de máquina nativo.

A interpretação de *bytecode* é naturalmente mais lenta do que executar instruções de máquina já compiladas. Devido a isso, as máquinas virtuais tem a opção de converter as sequências de *bytecode* no momento em que as executa, um processo chamado de compilação *just-in-time.* Com esse processo, a qualidade dos compiladores com esse recurso é tão boa quanto à dos compiladores tradicionais. [Horstmann, Core Java, Volume 1: Fundamentos]

Em alguns casos chega a ser melhor porque possui mais informações disponíveis, com isso pode identificar qual código é executado frequentemente e otimizar apenas a velocidade desse código.

Existem várias IDEs para se desenvolver em Java. A que foi utilizada para desenvolver o software em Java foi o Eclipse que é uma das mais utilizadas por programadores além de ser gratuita.

**2.4 – JavaScript**

Além da linguagem de programação Java, também foi utilizada a o JavaScript devido a utilização da API Google Maps.

O JavaScript nasceu para permitir que se escrevesse script diretamente no código HTML, assim serviria a dois propósitos: 1) Uma linguagem de scripting em que os administradores de servidor Web pudessem conectar suas páginas a outros serviços 2) No lado do cliente adicionar interatividade à página melhorando a experiência do usuário ao navegar em uma página Web.

Em dezembro de 1995, a Netscape em parceria com a Sun lançaram o primeira versão conhecida como JavaScript 1.0. Em 1996, o JavaScript 1.0 foi implementado no navegador Netscape Navigator 2.0, que dominava o mercado.

Como o JavaScript deve rodar do lado, ou seja, no navegador do usuário, é incorporado ao navegador um interpretador JavaScript. Com ele pode-se manipular o conteúdo e apresentação das página web, manipular o navegador, interagir com formulários e interagir com outras linguagens dinâmicas.

O JavaScript é muito utilizado para realizar validações de campos e no uso do Ajax na página web. Com as constantes melhorias em tecnologias com JSF e GWT, muitas vezes não é necessário utilizar JavaScript em páginas web. Mas ainda assim continua sendo muito utilizada.

### 2.4.1 - JSON

JSON é um formato de troca de dados baseado em um subconjunto do JavaScript. Utiliza um formato de texto completamente independente de linguagem. Por ser de fácil geração e análise por máquinas e de fácil escrita e leitura por programadores, além de utilizar convenções das linguagens mais utilizadas, o JSON é um dos melhores e mais utilizados formatos de troca de dados.

JSON é construído em duas estruturas:

- Uma coleção de pares de nome/valor.

Um objeto é um conjunto de não ordenado de pares nome/valor, conforme mostrado na figura 2.3.

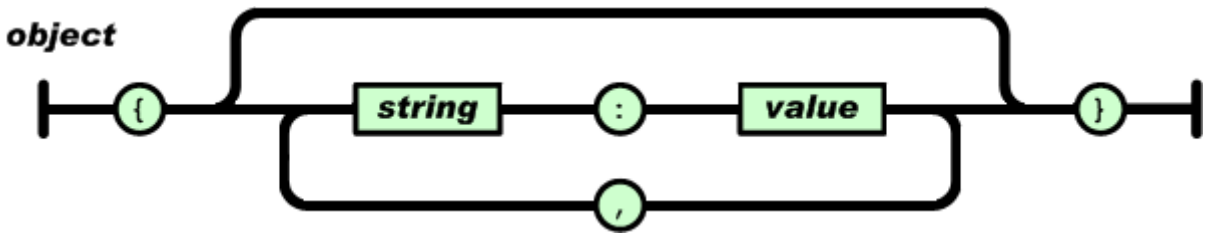


**Figura 2.3 - Objeto JSON. Fonte: http://www.json.org/**

- Lista ordenada de valores

Array é uma lista ordernada de valores, como mostrado na figura 2.4.

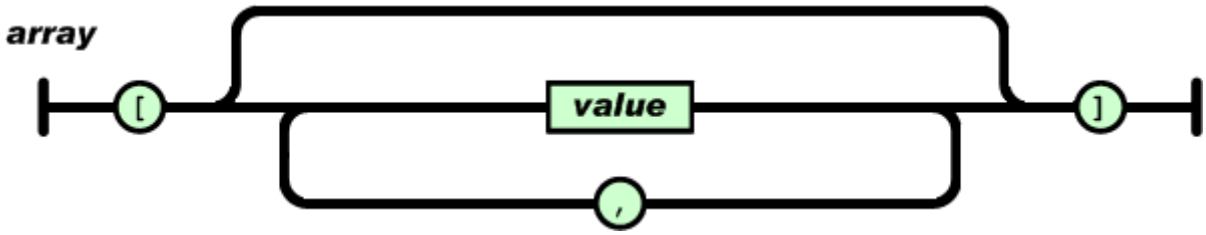


**Figura 2.4 - Array JSON. Fonte: http://www.json.org/**

Todas as linguagens de programação modernas possuem essas estruturas, o que facilita a troca de dados, além de possuir bibliotecas específicas para cada linguagem que possuem recursos avançados para tratamento do JSON.

## 2.5 – Sistema Gerenciador de Banco de Dados

É constituído por um conjunto de dados, associados a um conjunto de programas para acesso a esses dados. O conjunto de dados é chamado de banco de dados. O gerenciamento de informações implica na definição das estruturas de armazenamento e na manipulação dessas informações.

Dentro da estrutura do banco de dados está o modelo de dados que é um conjunto de ferramentas conceituais usadas para descrição dos dados, relacionamentos entre dados, semântica de dados e regras de consistência. Os modelos são classificados em três diferentes grupos: modelos lógicos com base em objetos, modelos lógicos com base em registros e modelos físicos. (Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Sistema de Banco de Dados).

Os modelos lógicos com base em objetos são usados para descrever o nível lógico e de visões, sendo flexíveis e permitindo visualizar as restrições dos dados. O modelo relacional mais conhecido é o modelo entidade-relacionamento.

O modelo entidade-relacionamento é formado por entidades que possuem relacionamentos entre si. Um relacionamento é uma associação entre entidades. Cada entidade possui seus atributos assim como um objeto do mundo real.

Há vários SGBDs disponíveis como Oracle, SQLServer, MySQL. O que foi utilizado no desenvolvimento do projeto foi o PostgreSQL que assim como o MySQL é gratuito. Além de ser um dos mais robustos disponíveis atualmente.

## 2.6 – PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional de código aberto. Derivado do projeto POSTGRES da universidade de Berkeley que foi originalmente patrocinado pela DARPA (Agência de pesquisas avançadas para defesa), ARO (Departamento de pesquisa militar), NFS (Fundação de pesquisa nacional) e ESL Inc.

O início da implementação do projeto foi em 1986, em apenas um ano tornou-se operacional. A primeira versão para lançada para o público foi em 1989. Em 1996, o nome do *software* foi alterado para PostgreSQL. Daí em diante foi implementada novos recursos e melhorias dos já existentes. (Wiki PostgreSQL, 2013).

## 2.7 – Java Persistence API - JPA

JPA é uma especificação utilizada na camada de persistência para facilitar o mapeamento dos objetos para o banco de dados.

Para se utilizar a especificação JPA, deve-se utilizar alguma implementação, que são *frameworks* ORM (*Object relational mapping*). As implementações mais conhecidas são: Hibernate, EclipseLink, OpenJPA e TopLink. Apesar do Hibernate ter originado a especificação JPA o EclipseLink é a implementação de referencia. (Caelum, 2013).

Com o JPA e possível mapear um POJO [1] para um banco de dados, quando um POJO é mapeado ele é chamado de *Bean* de entidade que é uma classe Java mapeada usando *Java Persistence Metadata*.

O mapeamento é automático e eleva consideravelmente a produtividade no desenvolvimento de aplicações e torna a migração entre bancos de dados mais rápida e fácil já que o código SQL é gerado pelo *framework* em tempo de execução.

ORM é uma técnica de mapeamento objeto relacional que cria uma camada de mapeamento entre os objetos e o modelo relacional (Banco de dados), de forma a abstrair o acesso ao mesmo. (http://www.devmedia.com.br/orm-object-relational-mapping-revista-easy-net-magazine-28/27158, acesso em 12/05/2013).

## 2.8 - Jboss Seam

JBoss Seam é um framework para desenvolvimento de aplicações Java EE, integrando tecnologia como Ajax, JSF, JPA, EJB 3.0 e BPM.

Foi desenvolvido para eliminar a complexidade em níveis de arquitetura e API. Oferece aos desenvolvedores total controle sobre a implementação da lógica de negócios, diminuição da configuração de arquivos XML dispondo de anotações para classes Java e

[1] POJO(*Plain Old Java Object*) – Objeto Java plano, simples, sem complexidade.

componentes bem definidos para a camada de apresentação, com vários componentes de interface.

O JBoss Sean é liderado por Gavin King, que também criou o Hibernate, e sua especificação é a JSR 299: Web Beans, que visa padronizar a forma de desenvolvimento sugerida pelo Seam.

A principal funcionalidade o JBoss Seam é integrar JSF e EJB3, através de componentes gerenciados (managed components), declaração de componentes JSF (backing beans) ou controlar a passagem de objetos entre as camadas de apresentação e negócio. Na figura 2.5 é mostrado como o JBoss Seam é integrado na arquitetura Java EE.

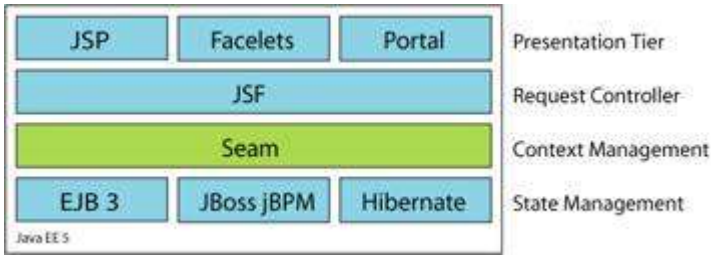


**Figura 2.5 – Integração do framework JBoss Seam em uma arquitetura Java EE. Fonte: http://www.devmedia.com.br/artigo-java-magazine-58-jboss-seam/9488 acessado em 25/09/2013**

### 2.8.1 – Seam Security

Seam Security possui inúmeras funcionalidades para segurança de aplicações Java EE. Dentre as quais pode-se destacar:

#### 2.8.1.1 – Autenticação

Autenticação é o ato de determinar ou confirmar a identidade de um usuário. Geralmente a autenticação é feita por uma login e senha, que são as credenciais do usuário para acessar a aplicação.

Com o uso do seam security, é possível utilizar serviços externos de autenticação como contas do Google, Facebook, servidor de diretórios LDAP ou Active Directory. Também é possível implementar um autenticador customizado para a aplicação, tudo isso graças a sua flexível API.

#### 2.8.1.2 - Autorização

Autorização controla quais ações o usuário pode realizar dentro da aplicação. Autorização pode ser grosseiramente dividida em duas categorias: uma em que as permissões de acesso são definidas por grupos ou responsabilidades e outras em que as permissões de acesso são definidas individualmente por usuário em determinado objeto da aplicação.

Há também permissões baseadas em regras que podem ser determinadas por uma lógica de negócio na aplicação.

#### 2.8.1.3 - Conceitos básicos.

A maioria das APIs de segurança são baseadas no bean Identity, que representa a identidade do usuário corrente e tem o escopo de sessão, ou seja, sua identidade não é perdida dentro da aplicação enquanto a sessão estiver mantida. Quando o usuário efetua o acesso a aplicação sua identidade é definida no escopo do ciclo de vida de sua sessão corrente.
Outro bean importante é o Credentials. Sua função é manter as credenciais do usuário na sessão. (JBoss Seam, 2013).

### 2.8.2 – Seam International

O Seam International é um dos módulos do Seam Framework. Este módulo é responsável pela configuração de localização, Idioma e fuso horários na aplicação. Permite ao desenvolvedor criar uma aplicação em vários idiomas de forma rápida e simples. Além de possuir recursos de fácil implementação para fusos horários e localização.

## 2.9 – Google Maps

Google Maps possui várias APIs que podem ser incorporadas a aplicação conforme a necessidade da mesma.

| API | Descrição |
|---|---|
| Google Maps JavaScript API | Incorpore um mapa do Google em sua página da web usando JavaScript. Manipule o mapa e adicione conteúdo com a ajuda de vários serviços. |
| Google Maps API for Flash | Use essa API ActionScript para incorporar um mapa do Google na sua página da web ou aplicativo baseado em Flash. Manipule o mapa em três dimensões e adicione conteúdo com a ajuda de vários serviços. |
| Google Earth API | Incorpore um verdadeiro globo digital em 3D à sua página da web. Leve os seus visitantes a qualquer lugar da Terra (até mesmo nas profundezas dos oceanos) sem tirá-los de sua página da web. |
| Google Static Maps API | Incorpore uma imagem simples e rápida do Google Maps em sua página da web ou site para celular sem precisar de códigos JavaScript ou qualquer carregamento dinâmico de página. |
| Serviços da web | Use solicitações de URL para acessar informações de geocodificação, rotas, elevação e lugares dos aplicativos cliente e manipule os resultados em JSON ou XML. |
| Google Maps Data API | Visualize, armazene e atualize dados de mapa por meio de feeds da Google Data API, usado um modelo de elementos (marcadores, linhas e formas) e coleções de elementos. |

**Figura 2.6 - APIs Google Maps. Fonte: DevMedia, disponível em http://www.devmedia.com.br/introducao-a-google-maps-api/26967 acessado em 05/05/2013**

As APIs utilizadas neste projeto são:

Google Maps JavaScript API v3 -Consiste em um conjunto de classes JavasScript que fornece ao usuário as ferramentas necessárias para construir aplicações que exibem mapas e permitem realizar consultas de endereços (*geoconding*).

Google Geocoding API v3 – É um serviço que fornece uma forma direta de acessar um geocodificador por meio de uma solicitação HTTP. Também fornece o serviço de geocodificação reversa.

A geocodificação é o processo de conversão de endereços em que se passa um endereço por extenso e é retornada a posição do endereço. Geocodificação reversa que é a transformação de coordenadas em endereços. (Google Developers, 2013)

Para que se possa utilizar a API deve-se ter uma chave de autenticação, sem essa chave não se pode incorporar a biblioteca a aplicação. A chave pode ser adquirida de forma gratuita mas possui algumas limitações. Com a chave pode-se realizar até 25.000 requisições por dia, essa limitação só é habilitada quando o site ultrapassa o limite por mais de 90 dias consecutivos.

**2.10 - Primefaces**

No projeto foi muito utilizado o framework Primefaces que torna o desenvolvimeno de aplicações web mais rápido e simples devido aos seus diversos componentes. Ao utilizar o Primefaces, muitas vezes não será necessário que o desenvolvedor programe em JavaScript no desenvolvimento das páginas web.

Primefaces é uma suíte de componentes de código aberto para JSF 2.0. O Primefaces possui mais de 100 componentes que podem ser incorporados a aplicação. Assim como o Primefaces, existem várias outras bibliotecas como: RichFaces, IceFaces entre outras. Não há um consenso de qual é a melhor, cada uma possui seus pontos positivos e negativos.

O número de desenvolvedores que vem adotando o Primefaces vem crescendo cada vez mais conforme mostrado na figura 2.7.

Um dos fatores que fazem o Primefaces crescer perante outras bibliotecas é a facilidade de inclusão da biblioteca na aplicação além de outras vantagens como uma vasta gama de componentes ricos, componentes adicionais para desenvolvimento móvel e boa documentação.

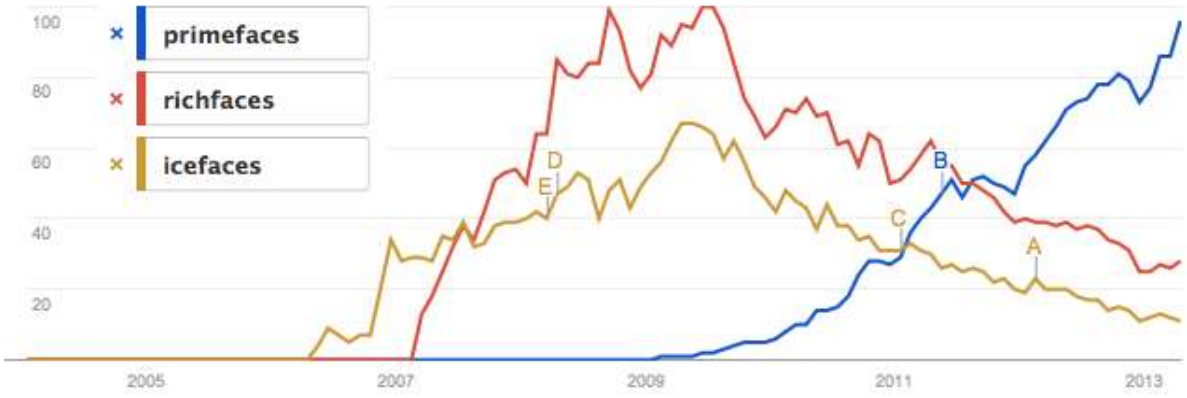


**Figura 2.7 - Uso de bibliotecas JSF - Fonte: http://www.primefaces.org/whyprimefaces.html - acessado em 08/05/2013**

# Capítulo 3 – Modelo Proposto

## 3.1 – Modelo Geral Proposto

O modelo consiste no desenvolvimento de um sistema em que será cadastrado um ou mais rastreadores GPS associados a determinado veículo e usuários. A partir desse cadastro, todos os usuários cadastrados de forma vinculada ao proprietário do veículo poderão ter acesso ao sistema e consultar a localização dos veículos, conforme mostrado na figura 3.1.

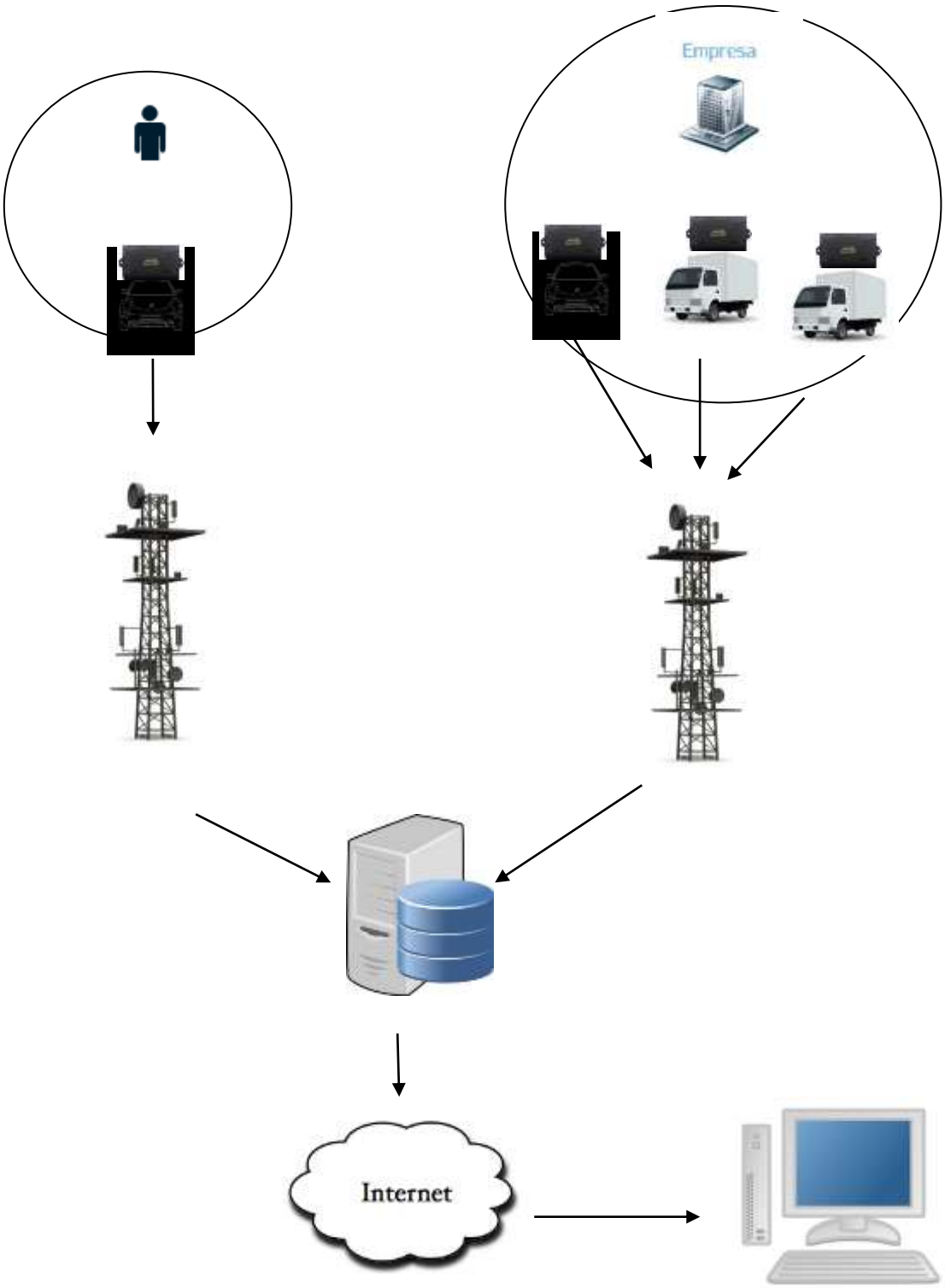

**Figura 3.1 – Modelo Geral do Projeto**

## 3.2 – Descrição das Etapas do Modelo

Neste capítulo serão descritos todos os fluxos disponíveis aos usuários do sistema.

### 3.2.1 – Cadastro de Proprietário

Na figura 3.2 é mostrado o diagrama de atividade para cadastro de proprietário.

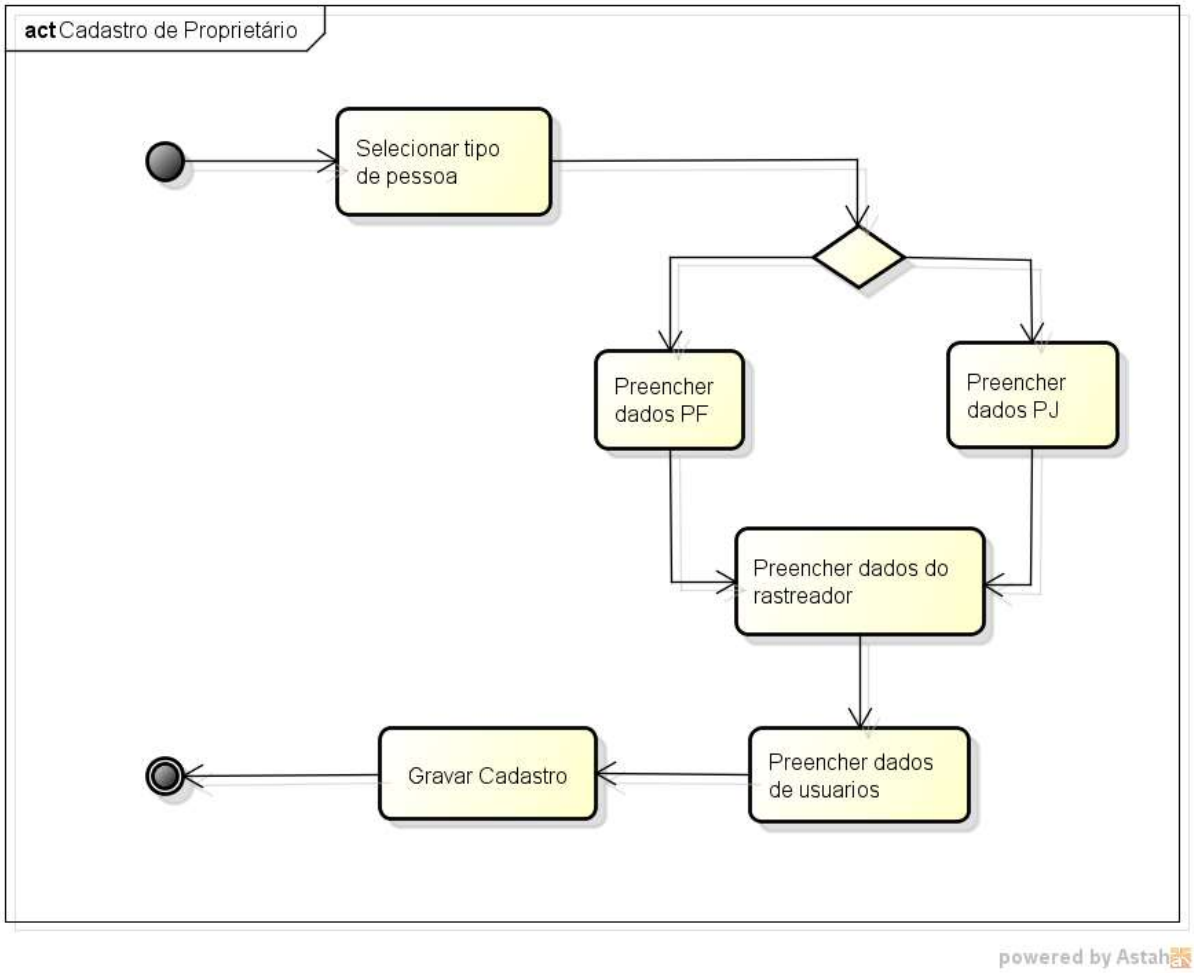


**Figura 3.2 - Diagrama de atividade do cadastro de proprietário. Fonte: Autor**

O cadastro de proprietário é realizado na primeira vez em que o usuário acessa o sistema. Nesse cadastro são definidos os rastreadores GPS e usuários que podem ter acesso às localizações dos rastreadores cadastrados.

Na figura 3.3 é mostrado o cadastro de proprietário pessoa física que possui um veículo com rastreador GPS e apenas um usuário de acesso ao sistema. Nesse caso como há apenas um usuário vinculado ao proprietário ele deve ser definido como Usuário Master.

Nada impede que o proprietário pessoa física possa ter vários usuários e carros vinculados a ele.

**Figura 3.3 – Modelo de cadastro de usuário pessoa física. Fonte: Autor**

Na figura 3.4 é mostrado o cadastro de um proprietário pessoa jurídica que possui vários veículos e vários usuários. No momento do cadastro do proprietário são definidos quais usuários são Usuários Master, que possuem permissão para editar os dados do proprietário e quais são Usuários Comuns, que podem apenas consultar as localizações dos veículos.

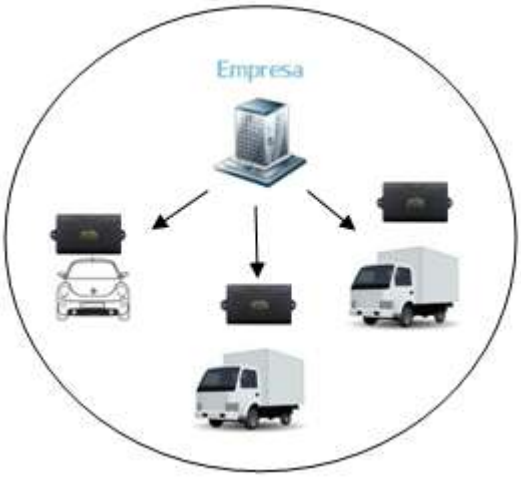


**Figura 3.4 – Modelo de cadastro de usuário pessoa jurídica. Fonte: Autor**

### 3.2.2 - Comunicação entre servidor e Rastreador

Como já dito anteriormente o rastreador GPS possui um SIM *card* como um número de telefone ativo e com conexão a Internet. A conexão utiliza a rede GSM da operadora, por esse meio é feita a comunicação entre o rastreador e o servidor como mostrado na figura 3.5. Também pode ser configurado o protocolo de comunicação entre rastreador e servidor. O rastreador aceita os protocolos TCP e UDP.

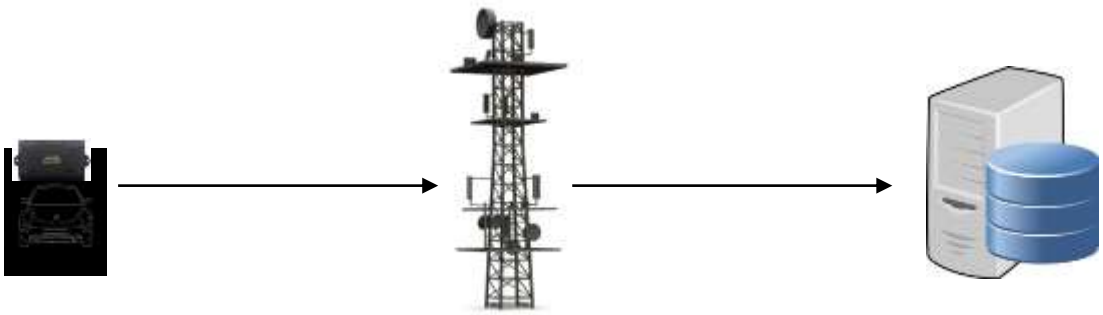

**Figura 3.5 – Comunicação entre rastreador e servidor. Fonte: Autor**

### 3.2.2.1 - Servidor

Para o sistema ser desenvolvido o servidor dever ter instalado uma JVM com Java 6 ou superior e um servidor de aplicação. No desenvolvimento do sistema foi utilizado o servidor de aplicação JBoss Application Server 6.

### 3.2.2.2 - Rastreador GPS

O rastreador utilizado foi o modelo TK-104 de fabricação chinesa, mostrado na figura 3.6.

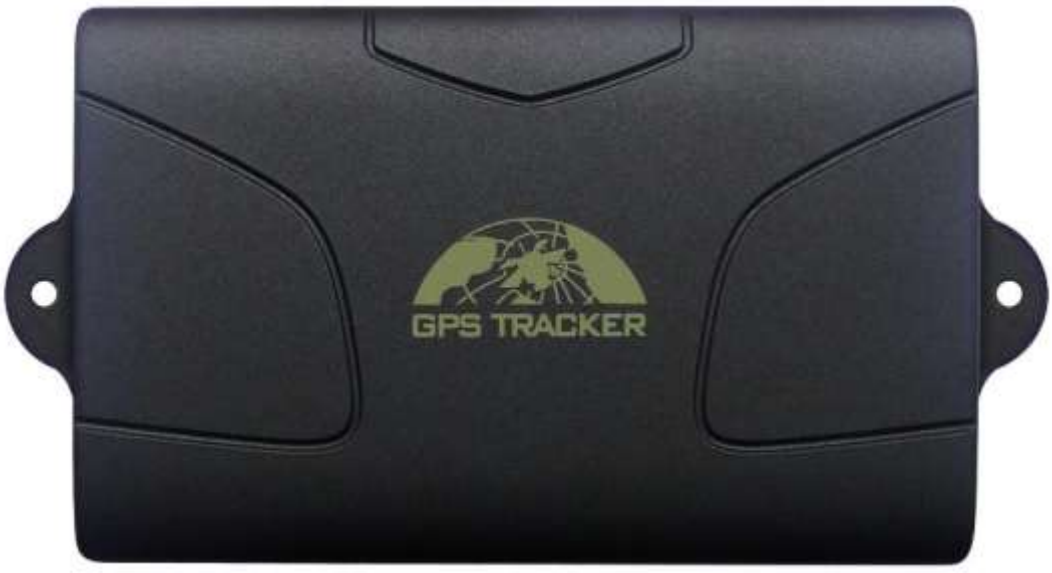


**Figura 3.6 - Visão frontal do Rastreador GPS TK-104. Fonte: Autor**

Este modelo de rastreador possui entradas para uma antena de GPS externa, uma antena GSM externa, um cartão de memória e um SIM *card*. Também possui entrada para carregador e microfone.

### 3.2.3 - Consulta de Localizações

No processo de consulta de localizações pelo usuário são exibidas as localizações em seu navegador. As localizações podem ser vistas em tempo real ou em um período de pesquisa pelo usuário. Na figura 3.7 é mostrado o modelo em que as localizações armazenadas em um servidor chegam até o usuário através da internet. Como o sistema é web ele pode ser acessado de computadores, tablets, smartphones ou qualquer dispositivo que possua um browser capaz de executar JavaScript.


**Figura 3.7 – Consulta de localizações pelo usuário do sistema. Fonte: Autor**

## 3.4 – Descrição da Implementação

O primeiro passo para implementação foi à compra do rastreador GPS utilizado no projeto. Com o rastreador pode-se iniciar a pesquisa para integração entre rastreador e sistema.

Para desenvolver o sistema foi utilizado o Eclipse por ser um dos melhores e mais utilizados ambientes para se desenvolver em Java. Eclipse possui *plugins* como Maven2 e JBoss Tools que foram utilizados para o desenvolvimento do projeto.

A modelagem de dados foi feita utilizando a ferramenta pgModeler, seguindo a Terceira Forma Normal. A partir da modelagem foi possível gerar o script para criação do banco de dados. Para gerar e editar o banco de dados foi utilizada a ferramenta PgAdmin.

O banco de dados utilizado foi o PostgreSQL por ser um dos mais robustos atualmente além de ser *open source*.

# Capítulo 4 – Implementação e Aplicação

Neste capítulo será descrita toda a implementação realizada para funcionamento do projeto.

## 4.1 – Modelagem de Dados

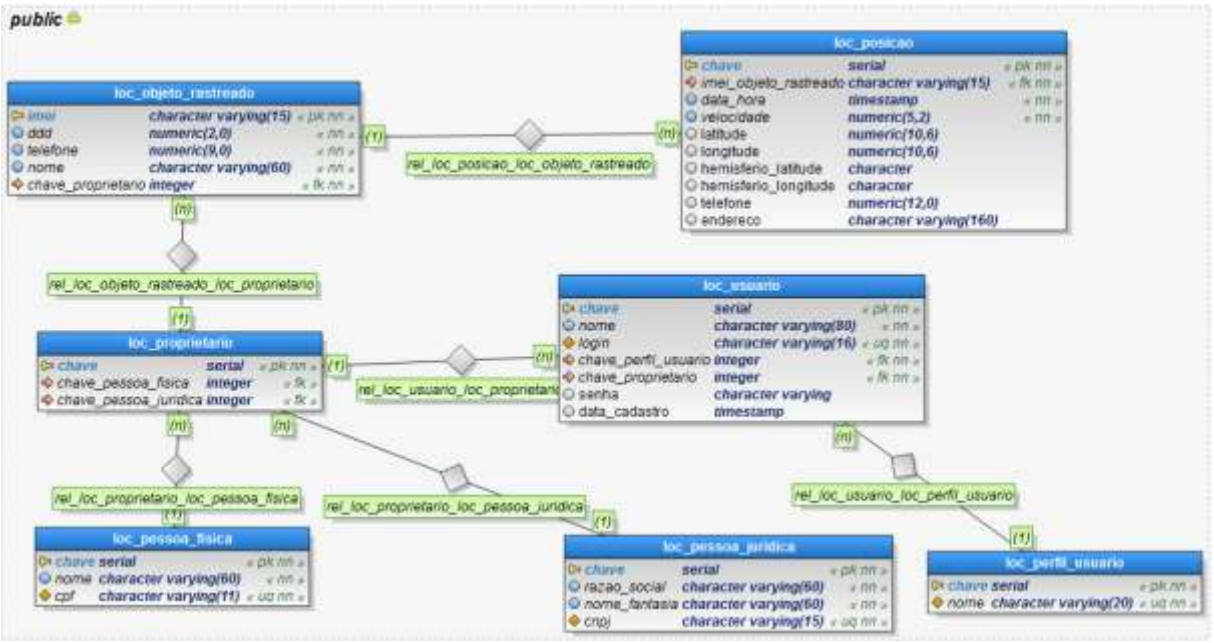


**Figura 4.1 – Modelagem de Dados utilizando o programa pgModeler – Fonte: Autor**

Feita a modelagem como mostrado na figura 4.1. Foi gerado o script para criação do banco de dados.

Com o script gerado, foi criado o banco de dados com a ferramenta PgAdmin além de serem feitos ajustes no banco e preenchida as tabela de perfis dos usuários.

Com o banco de dados criado pode-se iniciar a implementação do sistema. Todos os *beans* do sistema foram gerados a partir da perspectiva JPA do Eclipse. Nessa perspectiva é possível configurar o banco de dados e fazer o que o Eclipse gere todos os objetos a partir das tabelas criadas no banco de dados.

## 4.2 - Cadastro de Proprietário

Neste capítulo é descrito foi implementado o cadastro de proprietário no sistema desenvolvido.

### 4.2.1 – Implementação do cadastro de proprietário

Os primeiros dados que devem ser preenchidos são referentes ao proprietário, que pode ser pessoa física ou jurídica.

Quando o proprietário for pessoa física devem ser preenchidos o nome e o CPF, conforme a figura 4.2.

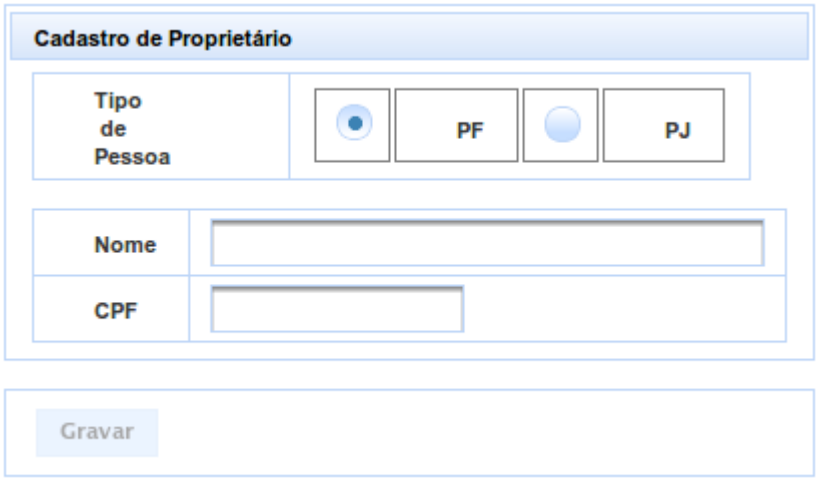


**Figura 4.2 – Cadastro de proprietário pessoa física. Fonte: Autor**

Quando o proprietário for pessoa Jurídica devem ser preenchidos a razão social, nome fantasia e CNPJ como mostrado na figura 4.3.

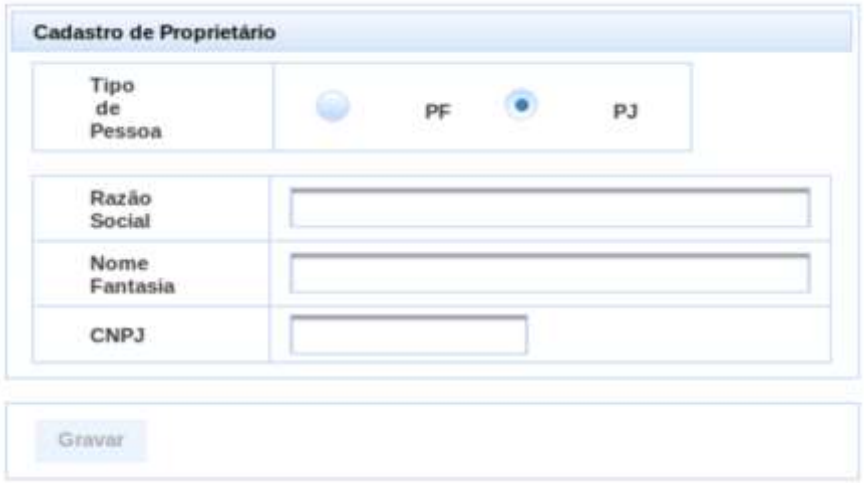


**Figura 4.3 – Cadastro de proprietário pessoa jurídica. Fonte: Autor**

Cada aparelho GPS é identificado pelo seu IMEI. É a partir do IMEI que é enviado pelo aparelho GPS que é possível saber qual aparelho está enviando informações para o servidor. Cada aparelho GPS também deve ter um SIM *card*, é através dele que o aparelho consegue se conectar ao servidor através da Internet. O número telefônico associado ao SIM *card* também deve ser cadastrado.

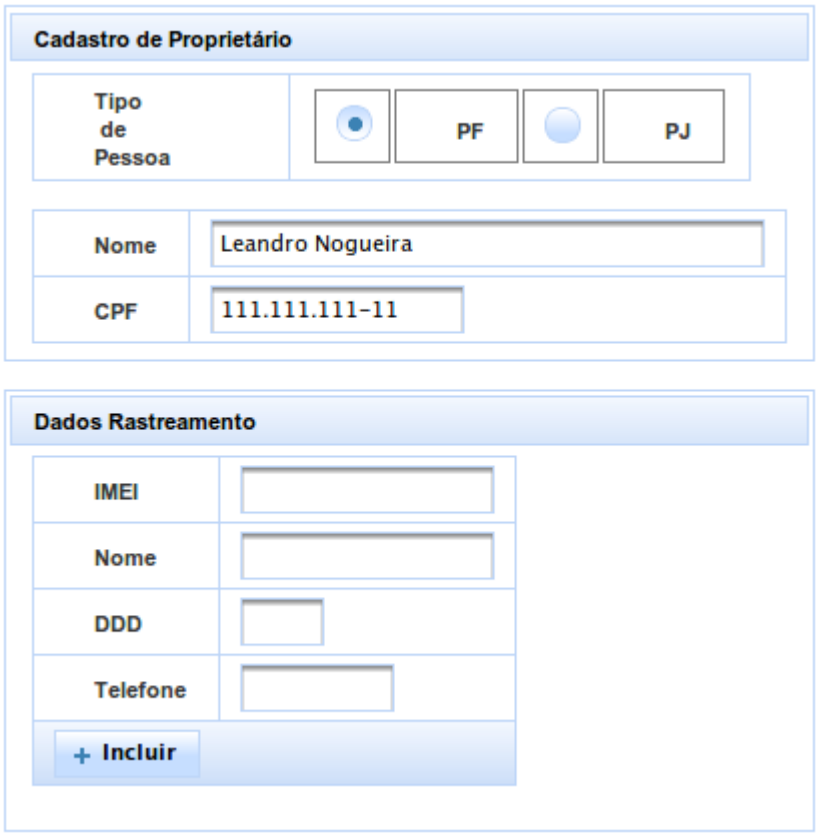


**Figura 4.4 – Dados de cadastro do rastreador e SIM *card*. Fonte: Autor**

Conforme mostrado na figura 4.4, os dados referentes aos aparelhos GPS que devem ser preenchidos são IMEI, um nome que servirá para identificá-lo o DDD e o número do SIM *card* vinculado ao aparelho GPS.

Após cadastrar os objetos rastreados é necessário cadastrar ao menos um usuário vinculado ao proprietário. Os usuários cadastrados poderão acessar o sistema com seu *login* e senha cadastrados.


**Figura 4.5 – Cadastro de usuários vinculados ao proprietário. Fonte: Autor**

Conforme mostrado na figura 4.5, para se cadastrar um usuário deve-se preencher nome, perfil, *login* e senha.

Os perfis disponíveis para cadastro de usuários são:

Usuário Master: Usuário com permissão de edição dos dados do proprietário cadastrado no sistema. Esse perfil pode cadastrar e excluir veículos rastreados e usuários do proprietário

Usuário: Esse perfil tem permissão apenas de visualização das localizações no sistema.

No sistema também há o perfil Administrador, que não pode ser cadastrado no sistema. Esse perfil pode ver todos os proprietários, veículos e usuários do sistema. Só é possível cadastrar usuário com esse perfil alterando o banco de dados.

### 4.3 - Configuração do Aparelho GPS

Para que o rastreador GPS se comunique com o servidor de aplicação é necessário configurá-lo. Todas as configurações podem ser feitas ou por SMS enviada para o rastreador o conectando-o a um computador usando a porta USB utilizando o programa apropriado.

Todas as configurações realizadas no rastreador utilizado no projeto foram feita por SMS. O primeiro passo é inicializar o rastreador GPS executando o comando “begin+senha”, caso seja iniciado com sucesso será retornada a mensagem “begin ok”. Após inicializar o rastreador, pode-se alterar sua senha com o comando “password+senha antiga+nova senha”.

O próximo passo é configurar o rastreador GPS para rastreamento continuo com o comando “fix030s***n+senha”, o valor 030 é o intervalo em segundos em que é enviada a localização para o servidor. Esse valor deve ter três dígitos e estar entre 020 e 255 seguido por (s: segundo; m: minuto; h: hora). O valor *** pode ser substituído por um número de três dígitos e define quantas localizações devem ser enviadas ao servidor no intervalo estipulado, com o valor *** não há limite de localizações a serem enviadas ao servidor, ou seja, é enviado ilimitadamente.

A configuração de rastreamento também pode ser feita por distância. Assim sempre que é percorrida uma distância pré-determinada será enviada a localização para o servidor. Para essa configuração deve-se executar o comando “distance+senha+distância”, o valor distância deve conter quatro dígitos.

As configurações do rastreador também permitem que seja enviada a localização para o servidor toda vez que o veículo mude altere sua trajetória para habilitar essa função deve-se enviar o comando “angle+senha ângulo”, o valor padrão do ângulo é de 30 graus, caso o comando seja executado com sucesso o rastreador responderá com a mensagem “angle ok”.

Outra função que pode ser bastante útil é o não envio de localização caso o veículo para de se mover, essa função pode ser habilitada com o comando “suppress+senha”.

Para checar o estado do rastreador pode-se enviar o comando “check+senha”, com esse comando, são retornadas informações do aparelho como: Power: que define se o rastreador está sendo recarregado. Battery com a porcentagem de carregamento da bateria. GPS retornado OK ou NO GPS caso não haja sinal. GSM Signal  com valor entre 1 e 32 que representa a força do sinal GSM.

Por último deve-se configurar o IP e porta para qual o rastreador envia as localizações. O comando responsável por essa configuração é o seguinte: “adminip+senha IP porta”. O IP pode variar de acordo com a máquina em que o sistema esteja sendo executado. A porta definida no desenvolvimento do projeto é a 7000.

### 4.4 - Informações do Rastreador GPS

Com o cadastro realizado e o rastreador configurado o sistema está apto a receber os dados do aparelho GPS.

O servidor recebe do rastreador uma *string* com as informações de localização. Abaixo está um exemplo das informações recebidas pelo servidor.

imei:863070010729159,tracker,1310142132,,F,003158.000,A,1601.0747,S,04803.4379,W,17.32,198.94,;

A primeira identificação é o IMEI do rastreador, é a partir dele que é possível identificar o aparelho que está enviando as informações.

Outras informações relevantes enviadas pelo rastreador são data e hora que no exemplo é 1310142132, onde 131014 é data no formato ano, mês e dia e 2132 é o horário em que a informação é enviada pelo rastreador.

A latitude enviada no exemplo é 1601.0747 e vem seguida do hemisfério que no exemplo ‘S’ de *south* (sul). A longitude é representada pelo valor 04803.4379 seguida pela representação do hemisfério W de *west* (oeste).

A latitude e a longitude enviados pelo rastreador GPS não são compatíveis com a API Google Maps, por isso é necessário converter essas informações para o padrão correto antes de gravar no banco de dados.

Tanto a latitude quanto a longitude vêm no formato xxmm.dddd, onde xx é o grau, mm são os minutos e dddd é a parte decimal dos minutos. A API Google Maps aceita três formatos de latitude e longitude: Graus decimais; Graus minutos e segundos; Graus e minutos decimais.

Para converter para o formato Graus decimais aplica-se o seguinte cálculo para as coordenadas recebidas no sistema.

Para $l$: latitude recebida pelo sistema

$$l_a = l \,/\, 100$$

Não devem ser consideradas as casas decimais em $l_a$.

$$l_c = l_a + ((\big(l - (l_a * 100)\big)/\, 60)$$

Em $l_c$ é a latitude convertida para o padrão aceito pelo Google Maps e pronta para ser gravada no sistema. Esse cálculo também deve ser aplicado para longitude

Em seguida, é apresentada a velocidade em que o veículo está no momento em que é enviada a informação para o servidor. Essa velocidade está milha por hora. Antes de gravar a velocidade, ela é convertida para km/h.

**4.5 - Recebimento das Informações**

Através da rede GSM é estabelecida uma conexão entre o servidor web e o rastreador GPS. Essa conexão é realizada por um Socket como mostrado na figura 4.6.

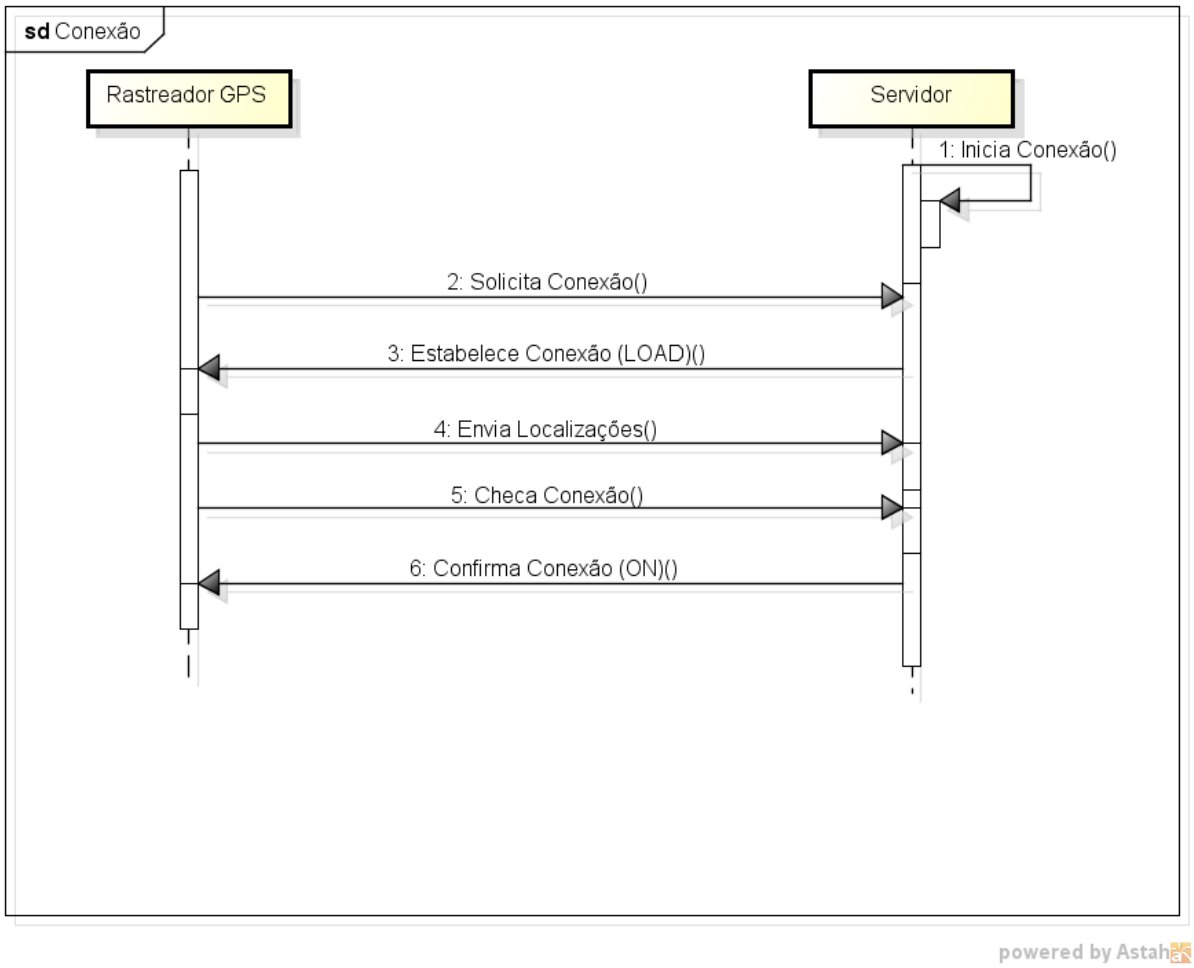


**Figura 4.6 – Diagrama de sequência de conexão entre rastreador e servidor. Fonte: Autor**

O método iniciarSocket(), mostrado na figura 4.7, é executado no momento em que o servidor é iniciado. Neste momento é criado um Server Socket que espera alguma conexão na porta 7000. Em seguida é criado um fluxo de entrada de dados (*InputStream*) e saída de dados (*OutputStream*).

Quando o rastreador GPS se conecta ao servidor ele envia uma mensagem parecida como com a mostrada anteriormente com IMEI, coordenadas e etc. Então o servidor deve responder com "LOAD". Nesse momento a conexão é estabelecida e o rastreador passa a enviar as informações com sua localização.

A cada trinta segundos o rastreador envia uma mensagem para o servidor para continuar com a conexão ativa e aguarda a resposta "ON", se o servidor não enviar a resposta a conexão é perdida, então se deve iniciar todo o processo novamente para se estabelecer uma nova conexão.

```
private void iniciarSocket() {
    try {
        this.serverSocket = new ServerSocket(7000);
        while (true) {
            this.socket = this.serverSocket.accept();

            this.inputStream = this.socket.getInputStream();
            this.outputStream = this.socket.getOutputStream();
            byte[] buffer = new byte[this.socket.getReceiveBufferSize()];
            this.inputStream.read(buffer, 0, buffer.length);
            String dados = new String(buffer, "ASCII");
            System.out.println("dados: "  + dados);
            this.outputStream.write(new String("LOAD").getBytes("ASCII"));
            int bufferSize = 0;
            do {
                bufferSize = inputStream.read(buffer, 0, buffer.length);
                dados = new String(buffer, "ASCII");
                System.out.println(dados);
                if (bufferSize > 0) {
                    if (dados.startsWith("imei:")) {
                        this.gravarPosicao(dados);
                    } else {
                        this.outputStream.write(new String("ON").getBytes("ASCII"));
                    }
                }
            } while (bufferSize > 0);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

**Figura 4.7 – Método que inicia Socket e recebe localizações do rastreador. Fonte: Autor**

## 4.6 - Consulta das Localizações

Após o cadastro do usuário e do aparelho GPS e de sua configuração o sistema passa a gravar as localizações enviadas. Ao acessar o sistema o usuário poderá visualizar as localizações gravadas.

Para carregar as localizações foi, utilizado a API Google Maps. Essa API como já descrita anteriormente utilizando a linguagem JavaScript para carregar o mapa no navegador do usuário. Na figura 4.8 e 4.9 é mostrado como foi utilizada a API no projeto.

```
<script type="text/javascript">

    var poly;
    var map;
    var markersArray = [];
    var seguir = true;

    function initialize() {
        var myLatlng = new google.maps.LatLng(-16.015333,-48.046108);
        var mapOptions = {
            zoom: 16,
            center: myLatlng,
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };

        map = new google.maps.Map(document.getElementById('map_canvas'), mapOptions);

        var polyOptions = {
            strokeColor: '#000000',
            strokeOpacity: 1.0,
            strokeWeight: 3
        }
        poly = new google.maps.Polyline(polyOptions);
        poly.setMap(map);
        converterJsonParaObjetos();
    }

    function converterJsonParaObjetos() {
        var jsonObjetos = document.getElementById('formMapa:hiddenPosicoes').value;
        var obj = eval("(" + jsonObjetos + ")");
        if (obj.listaPosicao.length > 0) {
```

**Figura 4.8 – Função que carrega o mapa no navegador. Fonte: Autor**

Na função initialize() é criada a variável myLatLng que armazena a latitude e longitude iniciais do mapa. Também é definido o *zoom* e a camada em que o mapa é exibido.

Em seguida é criado o objeto mapa com parâmetros definidos anteriormente. Logo abaixo define-se os parâmetros da poli linhas que traçam o caminho percorrido pelo veículo no mapa.

localizador/src/main/we
templates/defaultmapa.

```
                markersArray.pop();
            } else {
                path.push(latLng);
                map.setCenter(latLng);
            }
        } else {
            path.push(latLng);
            map.setCenter(latLng);
        }
        var descricao = "Nome: " + nome + "\n";
        descricao += "Velocidade: " + velocidade + " km/h \n";
        descricao += "Data e Hora: " + dataHora + "\n";
        descricao += "Endereço: " + endereco;
        var p = markersArray.length;
        var marker = new google.maps.Marker({
            position: latLng,
            title: p.toString(),
            map: map
        });
        markersArray.push(marker);
        attachSecretMessage(marker, descricao);
    }
} else {
    if (!seguir) {
        poly.getPath().clear();
        for (j = 0; j != markersArray.length; j++) {
            markersArray[j].setMap(null);
        }
        markersArray.length = 0;
    }
}
```

**Figura 4.9 - Função que carrega as localizações no navegador. Fonte: Autor**

Na função carregarJsonParaObjetos() são capturadas todas as localizações do veículo armazenadas em um campo *hidden* no formulário da página web. As localizações são carregas em uma *string* no formato JSON. Então é utilizada a função eval() para converter a *string* para uma *array*.

Após a conversão, são extraídas as informações como velocidade, data, hora e endereço para serem apresentadas ao usuário.

Com todas as informações capturadas e convertidas para visualização para o usuário é possível visualizar as informações como mostrado na figura 4.10.

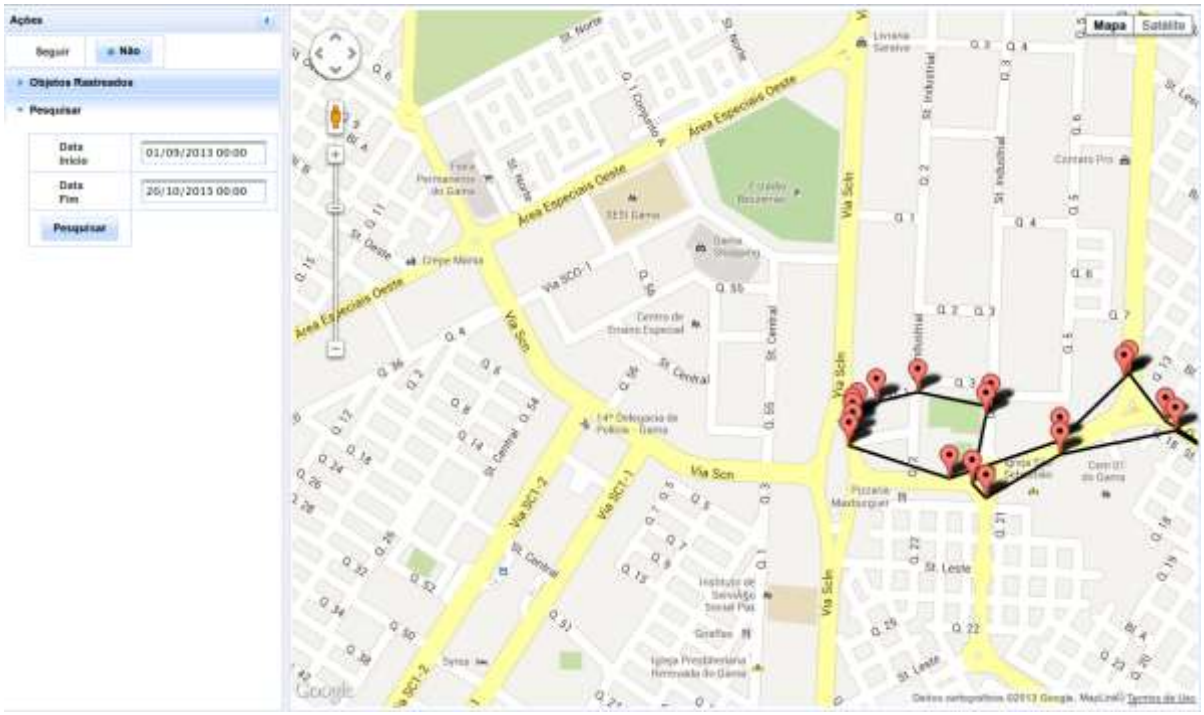


**Figura 4.10 - Mapa com localizações carregadas. Fonte: Autor**

# Capítulo 5 – Conclusão

## 5.1 – Conclusões

Após a conclusão do desenvolvimento do projeto pode-se concluir que o objetivo geral de implementação de um software de rastreamento de veículo por GPS foi concluído com sucesso. Os objetivos específicos também foram concluídos. O sistema web desenvolvido realiza todas as operações necessárias para que o usuário possa se cadastrar no sistema e rastrear seus veículos.

A integração do sistema com o rastreador GPS funciona corretamente desde que o meio de comunicação entre ambos também esteja funcionando. Todos os testes realizados de cadastro, rastreamento em tempo real e rastreamento por período obtiveram sucesso. Os dados apresentados ao usuário foram idênticos aos recebidos pelo sistema pelo rastreador.

Os testes realizados no sistema possuem uma massa de dados pequena o que não permite determinar com certeza absoluta como o sistema se comportaria em um ambiente de acesso por vários usuários simultaneamente. Mas pela arquitetura utilizada supõe-se que não deve haver problema com o desempenho do sistema até uma determinada quantidade de usuários. Obviamente quanto mais usuários acessarem o sistema mais recursos devem ser investidos na infraestrutura do sistema.

Um dos principais focos do trabalho foi à utilização de ferramentas gratuitas para manter o custo do projeto baixo, o que foi conseguido com sucesso. No total foi gasto apenas R$ 350,00 na compra do rastreador GPS.

Dentre os principais desafios e conhecimentos adquiridos pode-se citar a utilização do *framework* Primefaces juntamente com o Google Maps. A princípio a utilização dessas duas ferramentas trouxe alguns problemas de compatibilidade, mas no final acabou facilitando o desenvolvimento do projeto.

Outro problema encontrado foi com relação ao aparelho rastreador GPS utilizado. Devido ao aparelho ser de fabricação chinesa e não ser muito utilizado não foi encontrado um manual detalhando como o rastreador se comunica com o servidor. Por isso foi utilizado o manual de outro aparelho o TK-102. Esse aparelho é bastante comum e possui semelhanças de comandos e no padrão de comunicação com o sistema de rastreamento.

## 5.2 – Sugestões para Trabalhos Futuros

Para sugestões de trabalhos futuros, sugere-se a implementação de um sistema que utilize todos os recursos dos rastreadores GPS. Há diversos aparelhos capazes de desligar o veículo remotamente, ouvir o som ambiente do ambiente em que o aparelho está. Alguns possuem câmera de vídeo, sensor de choque, sensor de temperatura. Enfim, é possível desenvolver um sistema que utilize todos esses recursos, desde que o aparelho seja instalado no veículo.

Também é possível realizar melhorias no software utilizado no rastreamento. Seria interessante utilizar outros ferramentas para visualizar mapas como Bing Maps e Open StreetMap. Melhorias como traçar a melhor rota entre dois pontos no mapa, determinar uma área permitida onde o veículo pode trafegar, caso ele saia dessa área envia um alerta para um contato pré-determinado. Também poderia integrar o sistema a outros modelos de rastreadores GPS.

**Referências Bibliográficas**

ABRAHAM, S.; KORTH, H. F.; SUDARSHAN, S. Sistema de Banco de Dados, Editora Pearson

Caelum - Introdução prática ao JPA com Hibernate. Disponível em: <http://www.caelum.com.br/apostila-java-web/uma-introducao-pratica-ao-jpa-com-hibernate/#14-2-java-persistence-api-e-frameworks-orm>. Acessado em 12 maio de 2013

Caelum - O que é Java EE. Disponível em: <http://www.caelum.com.br/apostila-java-web/o-que-e-java-ee/#3-1-como-o-java-ee-pode-te-ajudar-a-enfrentar-problemas>. Acessado em 12 de maio de 2013

CAMPBELL, I. V. Sistema de Posicionamento Globa – GPS. Disponível em: http://www.gta.ufrj.br/grad/08_1/gps/>. Acessado em 14 de abril de 2013

Core J2EE Patterns – View Helper. Disponível em: <http://www.oracle.com/technetwork/java/viewhelper-139885.html>. Acessado em 12 maio 2013

DEITEL, Java: como programar. 8 ª Edição, 2010

DENATRAN. Disponível em: <http://www.roubadosbr.com.br/mais_roubados.php> Acessado em 15 de abril de 2013

GOLDMAN, D. JavaScript a Bíblia, Editora Campus

Google Developers - A Google Geocoding API. Disponível em: <(https://developers.google.com/maps/documentation/geocoding/?hl=pt-br>. Acessado em 5 de maio de 2013

HORSTMANN. Core Java , Volume 1: Fundamentos

Instituto CEUB de Pesquisa e Desenvolvimento. Curso de GPS e Cartografia Básica - Fundamentos GPS. Disponível em: <http://www.ufscar.br/~debe/geo/paginas/tutoriais/pdf/gps/Fundamento%20GPS.pdf>. Acessado em: 14 de abril de 2013

JBoss Seam. Disponível em: <http://docs.jboss.org/seam/3/security/latest/reference/en-US/html_single/, acessado em 22/10/2013>. Acessado em 12 de maio de 2013

Oracle – ViewHelper. Disponível em: <http://www.oracle.com/technetwork/java/viewhelper-139885.html>. Acessado em 12 de maio de 2013

ORM Object Relational Mapping. Disponível em: <http://www.devmedia.com.br/orm-object-relational-mapping-revista-easy-net-magazine-28/27158>. Acessado em 12 maio 2013

PostgreSQL – Introdução e Histórico. Disponível em: <http://wiki.postgresql.org/wiki/Introdução_e_Histórico>. Acessado em 25 de maio 2013

Primefaces. Diponível em: <http://www.primefaces.org>. Acessado em 08 maio 2013

Serviços da Web da API do Google Maps – A Google Geocoding API. Disponível em: <https://developers.google.com/maps/documentation/geocoding/?hl=pt-br>. Acessado em 05 de maio de 2013

Serviços da Web da API do Google Maps – Usage Limits and Billing. Disponível em: <https://developers.google.com/maps/documentation/javascript/usage?hl=pt-br>, Acessado 05 de maio de 2013

The Java EE 6 Tutorial. Disponível em: <http://docs.oracle.com/javaee/5/tutorial/doc/bnbpz.html>. Acessado em 12 maio 2013

TIOBE Software BV. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acessado em 06 de dezembro de 2013

UFRJ. Disponível em: <http://www.gta.ufrj.br/grad/08_1/gps/>. Acessado em 15 de abril de 2013

Unicamp - Introdução a Orientação Objetos. Disponível em: <http://www.dca.fee.unicamp.br/cursos/POOCPP/node4.html>. Acessado em 15 de abril de 2013

Wiki PostgreSQL - Introdução e Histórico. Disponível em: <http://wiki.postgresql.org/wiki/Introdução_e_Histórico)>. Acessado em 12 de maio de 2013

Wiltgen, Júlia - Saiba quais carros têm seguro mais caro e por quê. Disponível em: <http://exame.abril.com.br/seu-dinheiro/carros/noticias/saiba-quais-carros-tem-os-seguros-mais-caros-e-por-que>. Acessado em 11 de novembro de 2013

## Apêndice A – Código fonte do sistema

### I. Socket

Classe responsável por estabelecer a conexão com o rastreador GPS e receber as informações para serem gravadas no banco de dados.

```
package br.com.localizador.infra;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

import br.com.localizador.service.ThreadObjetoRastreadoService;

public class ThreadServidorSocket implements Runnable {

	private InputStream inputStream;
	private OutputStream outputStream;
	private Socket socket;
	private ServerSocket serverSocket;

	private void iniciarSocket() {
		try {
			this.serverSocket = new ServerSocket(7000);
			while (true) {
				this.socket = this.serverSocket.accept();

				this.inputStream =
this.socket.getInputStream();
				this.outputStream =
this.socket.getOutputStream();
				byte[] buffer = new
byte[this.socket.getReceiveBufferSize()];
				this.inputStream.read(buffer, 0,
buffer.length);
			String dados = new String(buffer, "ASCII");
				this.outputStream.write(new
String("LOAD").getBytes("ASCII"));
			int bufferSize = 0;
			 do {
				bufferSize = inputStream.read(buffer, 0,
buffer.length);
```

```
                        dados = new String(buffer, "ASCII");
                        System.out.println(dados);
                        if (bufferSize > 0) {
                              if (dados.startsWith("imei:")) {
                                    this.gravarPosicao(dados);
                              } else {
                                    this.outputStream.write(new
String("ON").getBytes("ASCII"));
                              }
                        }
                  } while (bufferSize > 0);
                }
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

      private void gravarPosicao(final String dados) {
            ThreadObjetoRastreadoService
threadObjetoRastreadoService;
            threadObjetoRastreadoService = new
ThreadObjetoRastreadoService(dados);
            Thread thread = new Thread(threadObjetoRastreadoService);
            thread.start();
      }

      public void finalizarSocket() {
            try {
                  this.inputStream.close();
                  this.outputStream.close();
                  this.socket.close();
                  this.serverSocket.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

      @Override
      public void run() {
            this.iniciarSocket();
      }

}
```

## II. Mapa

Neste arquivo XHTML está escrito o código que renderiza o mapa na tela para visualização das localizações pelo usuário do sistema.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:c="http://java.sun.com/jsp/jstl/core"
      xmlns:p="http://primefaces.org/ui">

<f:view contentType="text/html">
      <h:head>
            <title>#{resources['label.geral.tituloProjeto']}</title>
            <meta name="viewport" http-equiv="Content-Type"
content="text/html; charset=utf-8; initial-scale=1.0, user-
scalable=no" />
            <h:outputStylesheet name="css/screen.css" />

            <style type="text/css">
                  html { height: 100% }
                  body { height: 100%; margin: 0px; padding: 0px }
                  #map_canvas { height: 100% }
            </style>

            <script type="text/javascript"
src="http://maps.google.com/maps/api/js?v=3.11&amp;sensor=false&amp;
region=pt-br&amp;key=AIzaSyB6Dmh8pF-Os0T4kEJm58SWI9ojc4jab70"/>

            <script type="text/javascript">

                  var poly;
                  var map;
                  var markersArray = [];
                  var seguir = true;

                  function initialize() {
                        var myLatlng = new google.maps.LatLng(-
16.015333,-48.046108);
                        var mapOptions = {
                            zoom: 16,
                            center: myLatlng,
                            mapTypeId: google.maps.MapTypeId.ROADMAP
                        };
```

```
                    map = new
google.maps.Map(document.getElementById('map_canvas'), mapOptions);

                    var polyOptions = {
                    strokeColor: '#000000',
                    strokeOpacity: 1.0,
                    strokeWeight: 3
                    }
                    poly = new google.maps.Polyline(polyOptions);
                    poly.setMap(map);
                    converterJsonParaObjetos();
               }

               function converterJsonParaObjetos() {
                    var jsonObjetos =
document.getElementById('formMapa:hiddenPosicoes').value;
                    var obj = eval("(" + jsonObjetos + ")");
                    if (obj.listaPosicao.length > 0) {
                         for (j = 0; j != obj.listaPosicao.length;
j++) {
                              var nome = obj.nome;
                              var chave =
obj.listaPosicao[j].chave;
                              var dataHora =
obj.listaPosicao[j].dataHora;
                              var latitude =
obj.listaPosicao[j].latitude;
                              var longitude =
obj.listaPosicao[j].longitude;
                              var endereco =
obj.listaPosicao[j].endereco;
                              var velocidade =
obj.listaPosicao[j].velocidade;
                              var latLng = new
google.maps.LatLng(latitude, longitude);
                              var path = poly.getPath();
                              if (path.getLength() > 1) {
                                   if
(path.getAt(path.getLength()-1).equals(latLng)) {

     path.setAt(path.getLength()-1, latLng);

     markersArray[markersArray.length-1].setMap(null);
                                        markersArray.pop();
                                   } else {
                                        path.push(latLng);
                                        map.setCenter(latLng);
                                   }
```

```
                                    } else {
                                          path.push(latLng);
                                          map.setCenter(latLng);
                                    }
                                    var descricao = "Nome: " + nome +
"\n";
                                    descricao += "Velocidade: " +
velocidade + " km/h \n";
                                    descricao += "Data e Hora: " +
dataHora + "\n";
                                    descricao += "EndereÃ§o: " +
endereco;
                                    var p = markersArray.length;
                                    var marker = new
google.maps.Marker({
                                    position: latLng,
                                    title: p.toString(),
                                    map: map
                                    });
                                    markersArray.push(marker);
                                    attachSecretMessage(marker,
descricao);
                              }
                        } else {
                              if (!seguir) {
                                    poly.getPath().clear();
                                    for (j = 0; j !=
markersArray.length; j++) {
                                          markersArray[j].setMap(null);
                                    }
                                    markersArray.length = 0;
                              }
                        }
                  }

                  function attachSecretMessage(marker, descricao) {
                        var infowindow = new google.maps.InfoWindow(
                                    { content: descricao,
                                  size: new google.maps.Size(40,80)
                        });
                        google.maps.event.addListener(marker, 'click',
function() {
                              infowindow.open(map,marker);
                        });
                  }

                  function definirSeguir(bool) {
                        seguir = bool;
```

```
                }

                function loadScript() {
                     var script = document.createElement("script");
                     script.type = "text/javascript";
                     var url =
"http://maps.google.com/maps/api/js?v=3.11&amp;sensor=false";
                     url += "&amp;region=pt-br";
                     url += "&amp;key=AIzaSyB6Dmh8pF-
Os0T4kEJm58SWI9ojc4jab70";
                     url += "&amp;callback=initialize";
                     script.src = url;
                     document.body.appendChild(script);
                }

                window.onload = loadScript;

          </script>
     </h:head>

     <h:body>
          <h:form id="formMapa">
                <p:outputPanel id="idPanelCid">
                     <input type="hidden" id="cid" name="cid"
value="#{mapaViewHelper.conversation.id}" />
                </p:outputPanel>

                <p:growl id="messages" showDetail="false"
autoUpdate="true"/>

                <p:layout fullPage="true" >

                     <p:layoutUnit id="layoutUnitLeft"
position="west" size="285" maxSize="285"
                                   collapsible="true" collapsed="true"
resizable="true"

     header="#{resources['label.geral.acoes']}" >
                               <p:panelGrid columns="2"
id="panelOpcoes">
                                   <h:outputLabel
value="#{resources['label.geral.seguir']}" />
                                   <p:selectBooleanButton label=""
value="#{mapaViewHelper.seguir}"
onLabel="#{resources['label.geral.sim']}"

     offLabel="#{resources['label.geral.nao']}" onIcon="ui-icon-
check" offIcon="ui-icon-close">
```

```
                                        <p:ajax  process="@this" />
                                    </p:selectBooleanButton>
                                </p:panelGrid>
                                <p:accordionPanel id="accordionMenu">
                                    <p:tab
title="#{resources['label.geral.objetosRastreados']}">
                                        <p:orderList
value="#{mapaViewHelper.mapaTO.listaObjetoRastreado}"

     var="objetoRastreado" itemValue="#{objetoRastreado}"
converter="objetoRastreado">
                                            <p:column>
                                                <p:commandLink
actionListener="#{mapaViewHelper.selecionarObjetoRastreado}"

     value="#{objetoRastreado.nome}"
oncomplete="converterJsonParaObjetos();"

     update=":formMapa:hiddenPosicoes">

     <f:setPropertyActionListener
target="#{mapaViewHelper.mapaTO.objetoRastreado.imei}"

     value="#{objetoRastreado.imei}" />
                                                </p:commandLink>
                                            </p:column>
                                        </p:orderList>
                                    </p:tab>
                                    <p:tab
title="#{resources['label.geral.pesquisar']}">
                                        <p:panelGrid columns="2"
id="panelData">
                                            <h:outputLabel
value="#{resources['label.geral.dataInicio']}" />
                                            <p:calendar
value="#{mapaViewHelper.mapaTO.objetoRastreadoFiltro.dataInicial}"

     pattern="dd/MM/yyyy HH:mm" locale="pt"
mindate="#{customIdentity.usuario.dataCadastro}"

     maxdate="#{mapaViewHelper.dataAtual}"/>
                                            <h:outputLabel
value="#{resources['label.geral.dataFim']}" />
                                            <p:calendar
value="#{mapaViewHelper.mapaTO.objetoRastreadoFiltro.dataFinal}"

     pattern="dd/MM/yyyy HH:mm" locale="pt"
mindate="#{customIdentity.usuario.dataCadastro}"
```

```
      maxdate="#{mapaViewHelper.dataAtual}" />
                                                <p:column colspan="2">
                                                     <p:commandButton
action="#{mapaViewHelper.filtrar()}"

      update=":formMapa:hiddenPosicoes, :formMapa:panelOpcoes"

      value="#{resources['label.geral.pesquisar']}"

      oncomplete="definirSeguir(false);
converterJsonParaObjetos();"/>
                                                </p:column>
                                           </p:panelGrid>

                                      </p:tab>
                                 </p:accordionPanel>
                            </p:layoutUnit>

                            <p:layoutUnit id="layoutUnitCenter"
position="center" >
                                 <p:poll interval="20"
listener="#{mapaViewHelper.atualizar()}"
update="formMapa:hiddenPosicoes"

      oncomplete="converterJsonParaObjetos();" process="@this"/>
                                 <div id="map_canvas" style="width:100%;
height:100%"/>
                 </p:layoutUnit>
                 </p:layout>
                 <h:inputHidden
value="#{mapaViewHelper.mapaTO.posicoes}" id="hiddenPosicoes"/>
            </h:form>
      </h:body>
</f:view>
</html>
```

### III. Cadastro de Proprietário

Código fonte responsável por gerar a página de cadastro de proprietário no sistema.

```
<?xml version="1.0" encoding="UTF-8"?>
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:p="http://primefaces.org/ui"
```

```
   template="/WEB-INF/templates/default.xhtml">
   <ui:define name="content">

     <h:form id="formCadastroProprietario">

          <p:outputPanel id="idPanelCid">
               <input type="hidden" id="cid" name="cid"
value="#{administracaoViewHelper.conversation.id}" />
          </p:outputPanel>

          <p:growl id="messages" showDetail="false"
autoUpdate="true"/>

        <p:panel
header="#{resources['label.geral.cadastroProprietario']}"
               id="panelCadastroProprietario">
          <p:panelGrid columns="2">
               <h:outputLabel for="radioTipoPessoa"
value="#{resources['label.geral.tipoPessoa']}"/>
               <p:selectOneRadio id="radioTipoPessoa"
required="true" immediate="true"

     valueChangeListener="${administracaoViewHelper.selecionarTipoPe
ssoa}"

     value="#{administracaoViewHelper.administracaoTO.proprietario.t
ipoPessoa}">
                    <f:selectItem
itemValue="#{resources['label.geral.pessoaFisica']}"

     itemLabel="#{resources['label.geral.pessoaFisica']}"/>
                    <f:selectItem
itemValue="#{resources['label.geral.pessoaJuridica']}"

     itemLabel="#{resources['label.geral.pessoaJuridica']}"/>
                         <p:ajax immediate="true" event="change"
update="formCadastroProprietario"/>
               </p:selectOneRadio>
             </p:panelGrid>
               <p:spacer height="2px"/>
             <p:panelGrid columns="2" id="gridPessoaFisica"

     rendered="#{administracaoViewHelper.administracaoTO.proprietari
o.tipoPessoa eq 'PF'}">
               <h:outputLabel for="inputNomePessoaFisica"
value="#{resources['label.geral.nome']}"/>
```

```
                <p:inputText
value="#{administracaoViewHelper.administracaoTO.proprietario.pessoa
Fisica.nome}"
                        required="true" maxlength="60"
id="inputNomePessoaFisica" size="50"/>
                <h:outputLabel for="inputCpf"
value="#{resources['label.geral.cpf']}"/>
                <p:inputMask mask="999.999.999-99" required="true"
id="inputCpf"

     valueChangeListener="#{administracaoViewHelper.validarCpfPropri
etario}"

     value="#{administracaoViewHelper.administracaoTO.proprietario.p
essoaFisica.cpf}">
                            <p:ajax immediate="true" event="blur"
update="formCadastroProprietario"

     process="formCadastroProprietario:panelCadastroProprietario"/>
                        </p:inputMask>
             </p:panelGrid>
             <p:panelGrid columns="2" id="gridPessoaJuridica"

     rendered="#{administracaoViewHelper.administracaoTO.proprietari
o.tipoPessoa eq 'PJ'}">
                <h:outputLabel for="inputRazaoSocial"
value="#{resources['label.geral.razaoSocial']}"/>
                <p:inputText
value="#{administracaoViewHelper.administracaoTO.proprietario.pessoa
Juridica.razaoSocial}"
                            required="true" maxlength="60"
id="inputRazaoSocial" size="50"/>
                <h:outputLabel for="inputNomeFantasia"
value="#{resources['label.geral.nomeFantasia']}"/>
                <p:inputText
value="#{administracaoViewHelper.administracaoTO.proprietario.pessoa
Juridica.nomeFantasia}"
                            required="true" maxlength="60"
id="inputNomeFantasia" size="50"/>
                <h:outputLabel for="inputCnpj"
value="#{resources['label.geral.cnpj']}"/>
                <p:inputMask mask="99.999.999/9999-99"
required="true" id="inputCnpj"

     valueChangeListener="#{administracaoViewHelper.validarCnpjPropr
ietario}"
```

```
     value="#{administracaoViewHelper.administracaoTO.proprietario.p
essoaJuridica.cnpj}">
                      <p:ajax immediate="true" event="blur"
update="formCadastroProprietario"

     process="formCadastroProprietario:panelCadastroProprietario"/>
                </p:inputMask>
              </p:panelGrid>
           </p:panel>
           <p:spacer height="2px"/>
           <p:panel id="panelDadosRastreamento"
header="#{resources['label.geral.dadosRastreamento']}"
                 rendered="#{not empty
administracaoViewHelper.administracaoTO.proprietario.pessoaFisica.cp
f
                            or not empty
administracaoViewHelper.administracaoTO.proprietario.pessoaJuridica.
cnpj}">
            <p:panelGrid columns="2" id="gridDadosRastreamento">
                      <h:outputLabel for="inputImei"
value="#{resources['label.geral.imei']}"/>
                      <p:inputText
value="#{administracaoViewHelper.administracaoTO.objetoRastreado.ime
i}"
                                   maxlength="15" id="inputImei"/>
                      <h:outputLabel for="inputNomeObjetoRastreado"
value="#{resources['label.geral.nome']}"/>
                      <p:inputText
value="#{administracaoViewHelper.administracaoTO.objetoRastreado.nom
e}"
                                   maxlength="60"
id="inputNomeObjetoRastreado"/>
                      <h:outputLabel for="inputDDD"
value="#{resources['label.geral.ddd']}"/>
                      <p:inputText
value="#{administracaoViewHelper.administracaoTO.objetoRastreado.ddd
}"
                                   maxlength="2" id="inputDDD"
size="3"/>
                      <h:outputLabel for="inputTelefone"
value="#{resources['label.geral.telefone']}"/>
                      <p:inputText
value="#{administracaoViewHelper.administracaoTO.objetoRastreado.tel
efone}"
                                   maxlength="9" id="inputTelefone"
size="10"/>
                      <f:facet name="footer">
```

```
                    <p:commandButton
value="#{resources['label.geral.incluir']}"
                                update="formCadastroProprietario"
partialSubmit="true" ajax="true"

     process="formCadastroProprietario:panelDadosRastreamento"
                                icon="ui-icon-plus" style="margin:0"
action="#{administracaoViewHelper.incluirObjetoRastreado}"/>
               </f:facet>
          </p:panelGrid>
          <p:spacer height="2px"/>
          <p:dataTable id="tableObjetosCadastrados"
var="objetoRastreado" editable="true"
                    rendered="#{not empty
administracaoViewHelper.administracaoTO.proprietario.listaObjetoRast
reado}"

     value="#{administracaoViewHelper.administracaoTO.proprietario.l
istaObjetoRastreado}">
               <f:facet name="header">
                    <h:outputText
value="#{resources['label.geral.objetosRastreados']}"/>
               </f:facet>
               <p:ajax event="rowEdit"
listener="#{administracaoViewHelper.editarObjetoRastreado}"

     update=":formCadastroProprietario:messages"/>
               <p:ajax event="rowEditCancel"
listener="#{administracaoViewHelper.cancelarEdicaoObjetoRastreado}"

     update=":formCadastroProprietario:messages"/>
               <p:column
headerText="#{resources['label.geral.imei']}">
                    <p:cellEditor>
                         <f:facet name="output">
                              <h:outputText
value="#{objetoRastreado.imei}"/>
                         </f:facet>
                         <f:facet name="input">
                              <p:inputText
value="#{objetoRastreado.imei}" maxlength="15" required="true"/>
                         </f:facet>
                    </p:cellEditor>
               </p:column>
               <p:column
headerText="#{resources['label.geral.nome']}">
                    <p:cellEditor>
                         <f:facet name="output">
```

```
                                    <h:outputText
value="#{objetoRastreado.nome}"/>
                              </f:facet>
                              <f:facet name="input">
                                    <p:inputText
value="#{objetoRastreado.nome}" maxlength="60" required="true"/>
                              </f:facet>
                        </p:cellEditor>
                  </p:column>
                  <p:column
headerText="#{resources['label.geral.ddd']}">
                              <p:cellEditor>
                                    <f:facet name="output">
                                    <h:outputText
value="#{objetoRastreado.ddd}"/>
                              </f:facet>
                              <f:facet name="input">
                                    <p:inputText
value="#{objetoRastreado.ddd}" maxlength="2" required="true"/>
                              </f:facet>
                        </p:cellEditor>
                  </p:column>
                  <p:column
headerText="#{resources['label.geral.telefone']}">
                        <p:cellEditor>
                              <f:facet name="output">
                                    <h:outputText
value="#{objetoRastreado.telefone}"/>
                              </f:facet>
                              <f:facet name="input">
                                    <p:inputText
value="#{objetoRastreado.telefone}" maxlength="9" required="true"/>
                              </f:facet>
                        </p:cellEditor>
                  </p:column>
                  <p:column>
                        <p:rowEditor/>
                  </p:column>
            </p:dataTable>
            </p:panel>
            <p:spacer height="2px"/>
            <p:panel id="panelUsuarios"
header="#{resources['label.geral.usuarios']}"
                  rendered="#{not empty
administracaoViewHelper.administracaoTO.proprietario.listaObjetoRast
reado}">
            <p:panelGrid columns="2" id="gridDadosUsuarios">
```

```
                <h:outputLabel for="inputNomeUsuario"
value="#{resources['label.geral.nome']}"/>
                <p:inputText
value="#{administracaoViewHelper.administracaoTO.usuario.nome}"
                        maxlength="60" id="inputNomeUsuario"
size="50"/>
                <h:outputLabel for="inputPerfilUsuario"
value="#{resources['label.geral.perfil']}"/>
                <p:selectOneMenu id="inputPerfilUsuario"
label="#{resources['label.geral.perfil']}"

     value="#{administracaoViewHelper.administracaoTO.usuario.perfil
Usuario.chave}">
                    <f:selectItem itemValue=""
itemLabel="#{resources['label.geral.selecione']}" />
                    <f:selectItems
value="#{combosBean.listaPerfilUsuario}" var="perfilUsuario"
                                itemValue="#{perfilUsuario.chave}"
itemLabel="#{perfilUsuario.nome}"/>
                </p:selectOneMenu>
                <h:outputLabel for="inputLoginUsuario"
value="#{resources['label.geral.login']}"/>
                <p:inputText
value="#{administracaoViewHelper.administracaoTO.usuario.login}"
                        maxlength="20" id="inputLoginUsuario"
size="20"

     valueChangeListener="#{administracaoViewHelper.validarLogin}">
                    <p:ajax immediate="true" process="@this"
event="blur" update="@this" />
                </p:inputText>
                <h:outputLabel for="senhaUsuario"
value="#{resources['label.geral.senha']}"/>
                <p:password
value="#{administracaoViewHelper.administracaoTO.usuario.senha}"
size="15" id="senhaUsuario"
                        feedback="true" maxlength="12"
promptLabel="#{resources['label.geral.senha']}"

     strongLabel="#{resources['label.geral.senhaForte']}"

     goodLabel="#{resources['label.geral.senhaBoa']}"
weakLabel="#{resources['label.geral.senhaFraca']}"/>
                <f:facet name="footer">
                    <p:commandButton
value="#{resources['label.geral.incluir']}" ajax="true"
partialSubmit="true"
```

```
                                    update="formCadastroProprietario"
process="formCadastroProprietario:panelUsuarios"
                                    icon="ui-icon-plus" style="margin:0"
action="#{administracaoViewHelper.incluirUsuario}"/>
                </f:facet>
           </p:panelGrid>
           <p:spacer height="2px"/>
           <p:dataTable id="tableUsuarios" var="usuario"
                      rendered="#{not empty
administracaoViewHelper.administracaoTO.proprietario.listaUsuario}"

     value="#{administracaoViewHelper.administracaoTO.proprietario.l
istaUsuario}">
                <f:facet name="header">
                      <h:outputText
value="#{resources['label.geral.usuarios']}"/>
                </f:facet>
                <p:column
headerText="#{resources['label.geral.nome']}">
                      <h:outputText value="#{usuario.nome}"/>
                </p:column>
                <p:column
headerText="#{resources['label.geral.perfil']}">
                      <h:outputText
value="#{usuario.perfilUsuario.nome}"/>
                </p:column>
                <p:column
headerText="#{resources['label.geral.login']}">
                      <h:outputText value="#{usuario.login}"/>
                </p:column>
           </p:dataTable>
           </p:panel>
           <p:panel id="panelButtons">
           <p:commandButton
value="#{resources['label.geral.gravar']}"
action="#{administracaoViewHelper.gravar}"
                      ajax="false" disabled="#{not
administracaoViewHelper.cadastroValido}"/>
           </p:panel>
           </h:form>
   </ui:define>
</ui:composition>
```

# Apêndice B – Rastreador GPS

## I. Especificações do Rastreador GPS

| Content | specifications | |
|---|---|---|
| DIM. | 94mm*60mm*38mm | |
| Weight | 300g | |
| Network | GSM/GPRS | |
| Band | 850/900/1800/1900Mhz | |
| GSM/GPRS Module | SIM900B | |
| GPS Module | SIRF3 chip | |
| GPS sensitivity | -159dBm | |
| GPS Accuracy | 5m | |
| GPS Start time | Cold status | 45s |
| | Hot status | 1s |
| EXT. power | DC 12 V -24V | |
| Operating Current | Sleep Current | 4mA±1mA |
| | GSM operating Current | 54mA±3mA |
| | GPRS Current(data transfer) | 60mA±5mA |
| Battery | 3.7V 6000mA/h Polymer battery | |
| Charge current | 1.2A±0.2A | |
| Charge time | 5h | |
| Standby time | GSM sleep mode | 1500h |
| Storage Temp. | -40°C to +85°C | |
| Operation Temp. | -20°C to +65°C | |
| Humidity | 5%--95% non-condensing | |

## II. Imagens do rastreador GPS