SISTEMA DE RASTREAMENTO EM TEMPO REAL

Aluno: Thiago Feitosa de Abreu – RA: 2021872/2

Orientador: M.C. Claudio Penedo

Brasília – DF, dezembro de 2008

SISTEMA DE RASTREAMENTO EM TEMPO REAL

por

Thiago Feitosa de Abreu

Trabalho apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito parcial para obtenção do título de Engenheiro da Computação

Brasília – DF, dezembro de 2008

Banca Examinadora

_____

Prof. M.C. Claudio Penedo
Orientador

_____

Prof. Ms.C. Francisco Javier De Obaldia Díaz
Examinador

_____

Prof. M.C. Maria Marony Sousa Farias Nascimento
Examinador

Agradecimentos

Em primeiro lugar a Deus pela saúde, pelas oportunidades e pelos desafios que me colocou na vida, até hoje, para que conseguisse com muita dedicação e sabedoria, alcançar mais um objetivo, mais uma vitória.

Aos meus Pais, Manoel Abreu e Janilde Feitosa, pelo carinho e preocupação que sempre tiveram. Pela educação que me deram, e, claro, pela vontade e dedicação incansáveis de me guiar e orientar a seguir, sempre, o melhor caminho.

A todos os colegas e principalmente, os amigos, que me deram todo o apoio e incentivo para a realização deste projeto. Em especial, aos que se envolveram em momentos críticos para me ajudar no desenvolvimento da Interface para o celular, Vinicius Ferraz (Harry Potter), João e Fabio Oliveira (Seninha).

Aos professores do curso de Engenharia da Computação pelo longo trabalho que desempenharam de compartilhar seus conhecimentos, durante toda a minha formação acadêmica. E que sem dúvida, são grandes responsáveis pelo sucesso profissional de todos nós alunos.

E, por fim, para todos aqueles que direta ou indiretamente, estiveram presentes e dispostos a ajudar de alguma forma na conclusão deste projeto.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

ÍNDICE DE SIGLAS E ABREVIATURAS

GPS	Global Positioning System
GPRS	General Packet Radio Service
NMEA	Especificação de dados para comunicação de dispositivos eletrônicos de navegação
GSM	Global System for Mobile Communications
J2ME	Java 2, Micro Edition
ASP.NET	Linguagem baseada no Framework .NET voltada para Web
TDMA	Time Division Multiple Access
CDMA	Code Division Multiple Access
MIDP	Mobile Information Device Profile
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
SCO	Synchronous Connection-Oriented
ACL	Asynchronous Connection-Less
SGML	Standard Generalized Markup Language
HTML	HyperText Markup Language
RADAR	Radio Detection and Ranging
SGBD	Sistema de Gerenciamento de Banco de Dados
IIS	Internet Information Service
API	Application Programming Interface
SPP	Serial Port Profile

UUID	Universally Unique Identifier

RESUMO

Este trabalho apresenta o desenvolvimento de uma solução de rastreamento com o auxílio de um GPS integrado a um telefone celular, que envia informações de localização para um banco de dados na internet que é responsável pelo armazenamento do histórico e do percurso realizado.

A solução desenvolvida foi implementada utilizando um módulo GPS e um celular com tecnologia Bluetooth para que houvesse a conexão entre os dois dispositivos. Através do celular, o envio das informações para a internet é realizado por GPRS (serviço disponível na rede de telefonia móvel não baseado em voz).

Com o envio das informações para a internet e o armazenamento em um banco de dados, foi utilizada a API do Google Maps, disponível gratuitamente, para comprovar na internet o percurso realizado e a localização do celular rastreado.

Nesta solução foi utilizada a ferramenta NetBeans para desenvolver a aplicação do dispositivo móvel, que obtém do módulo GPS os dados de localização (latitude e longitude). A aplicação Web foi desenvolvida utilizando a ferramenta Microsoft Visual Studio 2005 e o banco de dados utilizado é o SQL Server 2005. A linguagem de desenvolvimento da aplicação do celular foi Java J2ME e para a aplicação Web foi utilizado ASP.NET/C#.

Palavras-chave: GPS, bluetooth, J2ME, ASP.NET, C#, Banco de Dados SQL Server.

CAPÍTULO 1 – INTRODUÇÃO

Neste capítulo é feita a apresentação deste trabalho de projeto final descrevendo tópicos como motivação, objetivos e a estrutura da Monografia.

1.1 MOTIVAÇÃO

A internet tem revolucionado o mundo computacional e das comunicações de uma forma não vista anteriormente. A invenção do telégrafo, telefone, rádio e computador prepararam um ambiente para uma integração nunca imaginada. A internet tornou-se um mecanismo de disseminação da informação e interação entre indivíduos e seus computadores, independente de sua localização geográfica. Além de representar um dos mais bem sucedidos exemplos dos benefícios da manutenção do investimento e do compromisso com a pesquisa e o desenvolvimento de uma infra-estrutura para a informação. [INTERNET SOCIETY, A brief of the Internet – 15/10/2008]

Com isso, a internet tornou-se um dos mais importantes meios de comunicação no mundo, possibilitando a obtenção e a troca de informações dos mais diversos tipos. Com a mesma velocidade de crescimento, a rede de telefonia móvel tornou-se também outro meio de comunicação que se moderniza a cada dia que passa.

Diante de tanta diversidade tecnológica disponível, que está em constante e crescente modernização, tornou-se viável desenvolver um projeto como este, utilizando a rede mundial de computadores e a disponibilidade da rede de telefonia móvel.

Entretanto, por maior que seja a modernização e a disponibilidade do sinal das antenas que formam a rede de telefonia celular, ainda esbarramos em alguns problemas, como: falta de sinal em locais muito abaixo do nível do solo, perda de comunicação do GPS em locais onde existem barreiras sólidas, como túneis, garagens subterrâneas, prédios e etc.

Mesmo com a existência dessas barreiras, pelas quais teria que deparar na criação deste projeto utilizando GPS como uma parte da solução, foi na aviação que notou-se a necessidade de se conseguir rastrear praticamente em tempo real, dependendo única e exclusivamente de um bom sinal de celular e a internet como meios de comunicação para o envio e a visualização dos dados, respectivamente.

Na aviação, quando se voa de planador, um avião que não possui motor, é comum acontecer de chegar a uma determinada altitude, ainda segura, porém baixa, que é preciso decidir o melhor local de pouso, tendo em vista a impossibilidade de ganhar altitude para prosseguir no vôo normalmente devido a uma série de fatores externos que envolvem o vôo de planador. A parte ruim é que, na maioria dos casos, esses pousos ocorrem longe do aeródromo local, por exemplo, em alguma plantação, fazendas e locais descampados.

Nessas eventualidades, o piloto, que sempre voa com um GPS, entra em contato por celular com sua equipe para dar as coordenadas do local onde pousou e ser resgatado. Logo, como são duas coisas que o piloto sempre vai carregar durante um vôo de planador, surgiu a idéia de integrá-los e, assim, disponibilizar uma solução em que se pudesse acompanhar por meio da internet o trajeto que está sendo percorrido pelo planador.

Contudo, para facilitar a demonstração e comprovação dos dados e trajetos percorridos, todos os testes serão feitos em automóveis. O caso do planador foi apenas um exemplo de aplicação onde houve a motivação da realização do projeto.

1.2 OBJETIVOS

O projeto tem como finalidade, o desenvolvimento e a implementação de uma solução de rastreamento que utilize um celular GSM e um módulo GPS com tecnologia Bluetooth para se comunicar com o aparelho celular. E, a partir desta comunicação, desenvolver um sistema para ser instalado no celular, que obtenha os dados de posicionamento geográfico, para que, através da rede de

telefonia celular (GPRS), possibilita o envio de dados para a internet e gravar em Banco de dados o histórico de posições ocupadas pelo dispositivo rastreado.

Os objetivos específicos do trabalho são:

a) Conectar o celular no GPS via Bluetooth e integrar através de software desenvolvido

b) Obter dados de Posicionamento Geográfico, como:
    a. Latitude
    b. Longitude
    c. Data e Hora

c) Enviar dados coletados, para um Banco de Dados disponível na internet

d) Armazenar dados recebidos do celular e disponibilizar uma interface Web que trace o caminho percorrido pelo objeto rastreado no Google Maps®

1.3 ESTRUTURA DA MONOGRAFIA

Este projeto constitui-se de quatro capítulos, incluindo a introdução. Segue uma breve descrição:

- Capítulo 2 – Referencial Tecnológico – aborda os principais conceitos e definições envolvidas neste projeto final, como J2ME, ASP.NET/C#, Telefonia Móvel, GPRS, dentre outros.
- Capítulo 3 – Desenvolvimento – descrição detalhada do desenvolvimento propriamente dito, deste projeto. Citando-se todas as ferramentas utilizadas, descrições detalhadas de código desenvolvido nas aplicações, detalhes da implementação, dos testes e dos resultados obtidos.

- Capítulo 4 – Conclusões – descreve as principais conclusões obtidas neste projeto.

CAPÍTULO 2 – REFERENCIAL TECNOLÓGICO

Neste capítulo, faz-se referência teórica de todos os assuntos abordados no desenvolvimento deste projeto.

2.1. Hardware

2.1.1. Telefonia Celular

A rede de telefonia celular é uma rede de telecomunicações que foi projetada para prover serviços de telefonia móvel, ou seja, permitir a comunicação entre uma ou mais estações móveis, que podemos chamar de telefone celular.

O telefone celular nada mais é do que um rádio extremamente sofisticado. O telefone foi inventado por Alexander Graham Bell, em 1876. Por volta de 1880, Nikolai Tesla inventou a comunicação sem fio através do rádio, que mais tarde, foi apresentado oficialmente por um italiano chamado Guglielmo Marconi.

Com o passar dos anos, essas duas tecnologia se complementaram e, assim, deu-se início ao surgimento dos telefones celulares. Porém, antes de sua criação, as pessoas que precisavam ter a capacidade de se comunicar de forma móvel, instalavam rádio-telefone nos carros.

A chamada radiotelefonia funcionava por meio de uma antena central por cidade e pouco mais de 20 canais em cada antena. Para se comunicar com essa antena, era necessário que o carro tivesse um transmissor muito potente para que estabelecer a conexão com um dos canais disponíveis. Isso significava que, o número de pessoas que poderiam usar seus rádios-telefones simultaneamente, limitava-se ao número de canais disponíveis na Antena, ou seja, um número muito baixo. [Oficina da NET, Construindo conhecimento – Como funcionam os telefones celulares – 10/10/2008]

O que ocorre na atual rede de telefonia celular é a possibilidade da divisão de uma cidade em pequenas células, permitindo a reutilização de freqüência em uma mesma cidade. Logo, diversas pessoas podem usar seus telefones celulares de forma simultânea.

Para entender melhor o conceito e o avanço da tecnologia na telefonia celular, podemos exemplificar alguns conceitos:

- **Half-duplex**: Quando dois dispositivos se comunicam, utilizam a mesma freqüência, sendo que apenas um pode falar, enquanto o outro dispositivo apenas escuta como é o caso ilustrado na Figura 2.1.

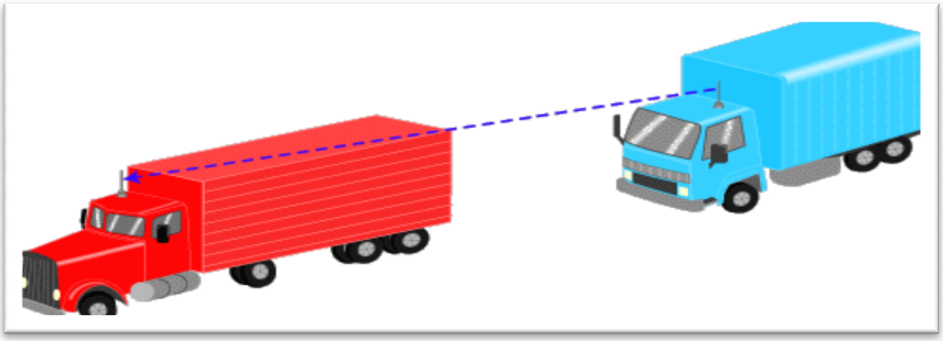


Figura 2.1 – Comunicação Half-Duplex

- **Full-duplex**: Durante a comunicação entre dois dispositivos, uma freqüência é usada para falar e a outra freqüência, independente, apenas para escuta. Ilustra-se este caso na Figura 2.2. Ambos os dispositivos podem se comunicar ao mesmo tempo.

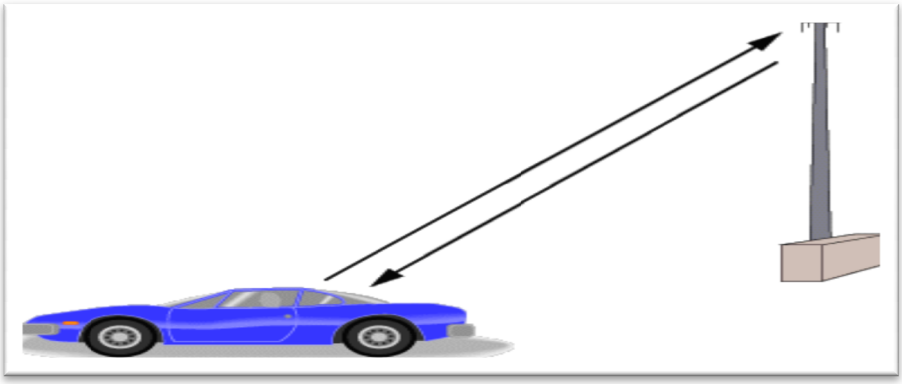


Figura 2.2 – Comunicação Full-Duplex

Com isso podemos definir um telefone celular como sendo uma estação móvel que funciona através de um sistema de comunicação sem fio, que possui características específicas, normalmente vinculadas a alguma geração de telefonia celular, como TDMA, CDMA, GSM, GPRS, EDGE, entre outras.

2.1.2. GPRS (General Packet Radio Service – Serviço de Rádio de Pacote Geral)

A tecnologia GPRS é uma tecnologia que aumenta as taxas de transferência de dados nas redes GSM. Utiliza a comutação de pacotes para o transporte dos dados. E, diferente das tecnologias anteriores, que utilizavam comutação por circuito, o GPRS funciona como um serviço por demanda. Estará sempre ativo, porém, o recurso necessário somente será atribuído quando houver necessidade de transferir dados, ou seja, utiliza comutação por pacotes.

O acesso ao serviço GPRS pode ser limitado a cada operadora de telefonia celular ou limitado pelo próprio aparelho celular caso não possua essa tecnologia disponível. Nos telefones celulares que possuem assinatura e tecnologia disponível, é preciso, normalmente, conhecer algumas configurações de software e hardware para o funcionamento deste serviço.

2.1.2.1. Comutação de Circuitos

Comutação de circuitos funciona de forma dedicada, ou seja, quando é necessário efetuar um transporte de dados, uma conexão é estabelecida entre a origem e o destino e esta, permanece dedicada às duas entidades, durante toda a chamada, até que a conexão seja interrompida.

2.1.2.2. Comutação de Pacotes

A comutação de pacotes é um paradigma de comunicação de dados em que pacotes são individualmente encaminhados entre nós de uma rede através

de ligações de dados tipicamente partilhadas por outros nós. [TANEBAUM 2003, 4ª edição].

A técnica da utilização de comutação de pacotes na tecnologia GPRS permite que várias entidades possam compartilhar os mesmos recursos, o que permitirá uma gerência mais eficiente dos recursos de rede e também sua capacidade.

2.1.3. Coordenadas Geográficas

As localizações no planeta Terra podem ser especificadas usando um sistema de coordenadas esféricas. O sistema de coordenadas geográficas funciona de acordo com os eixos da Terra. São definidos dois traçados de medida dos quais podemos chamar um de Latitude e outro de Longitude. A Latitude é o ângulo medido de um ponto qualquer na Terra, em relação à linha do Equador. O outro ângulo chamado de Longitude é a medida do Meridiano de Greenwich em relação a um ponto qualquer na Terra. A combinação destes dois pontos forma uma coordenada geográfica.

As linhas constantes da Latitude são chamadas de paralelos. Elas traçam círculos na superfície da Terra, mas o único paralelo que forma um círculo perfeito é a linha do Equador que possui Latitude de 0º. As linhas constantes de Longitude são chamadas de Meridianos, elas passam pelo Meridiano principal que é chamado de Greenwich. Ao contrário dos paralelos, os meridianos são círculos perfeitos e não são paralelos uns aos outros.

2.1.3.1. Margem de Erro

Na portaria número 006 de 20 de janeiro de 2003 da ANATEL (Agência Nacional de Telecomunicações), têm-se o seguinte artigo:

“Art. 3º O valor numérico da coordenada indicada no sistema geodésico WGS-84, considerado aqui, como uma grandeza mensurável, deve ser definido de modo a que o desvio máximo deste ao padrão de referência nacional, conforme

estabelecido pelos órgãos competentes na esfera da Administração Pública Federal, seja inferior a 1'' (um segundo) para latitude e longitude geodésicas e de 100m (cem metros) para altitude elipsoidal, já considerados quaisquer erros sistemáticos ou aleatórios nos processos de medição e/ou conversão utilizados." [www.anatel.gov.br – 10/10/2008]

2.1.4. GPS (Global Positioning System – Sistema de Posicionamento Global)

O uso de sinais de rádio para determinar a posição foi um avanço significante na navegação. O equipamento de rádio-navegação funcionou na época da II Guerra Mundial quando foi desenvolvido o RADAR – Radio Detection and Ranging – e a capacidade de medir lapsos de tempo entre emissão/recepção de ondas de rádio.

O GPS surgiu a partir de interesses militares e foi muito usado pelos Estados Unidos, na guerra do Golfo, para orientar os mísseis para o alvo. O GPS é um sofisticado sistema eletrônico de navegação, baseado em uma rede de satélites que permite localização imediata em qualquer ponto da Terra com uma precisão muito alta. Consiste basicamente em três partes: um sofisticado sistema de satélites orbitando ao redor da Terra, estações rastreadoras localizadas em diferentes pontos do globo e os receptores GPS.

Com 26 satélites disponíveis orbitando na Terra a 20000 km de altitude, a qualquer momento, pelo menos 5 (cinco) satélites estarão sobre o céu do receptor de um usuário em qualquer ponto do mundo. O sistema GPS se baseia no principio de triangulação, segundo o observador conhece a posição de um conjunto de satélites em relação a um referencial inercial e a sua posição em relação a este conjunto, e obtém sua própria posição no sistema de referencia.[Wikipedia – Sistema de Posicionamento Global – 18/10/2008]

O receptor tem que conhecer as localizações dos satélites. Uma lista de posições conhecidas como almanaque, é transmitida de cada satélite para os

receptores. Controles em Terra ficam rastreando os satélites e os mantendo atualizados.

2.2. Software

2.2.1. Plataforma Java e Biblioteca J2ME

A plataforma Java é uma linguagem de programação orientada a objetos, desenvolvida pela Sun Microsystems. É uma linguagem de alto nível, com sintaxe extremamente similar ao C++, bem estruturada, extensível e segura. [Sun Microsystems]

Dentro da plataforma Java, utilizamos a biblioteca J2ME (Java 2 Micro Edition) que disponibiliza um ambiente flexível e robusto para o desenvolvimento de aplicações para dispositivos móveis. Os principais componentes do J2ME são o CDC (Connected Device Configuration), o CLDC (Connected Limited Device Configuration) e o MIDP (Mobile Interface Device Profiles).

2.2.2. World Wide Web

Inicialmente, no inicio da década de 70, a internet estava limitada às universidades e instituições de pesquisa. Posteriormente, os militares adotaram a tecnologia e, finalmente, o governo dos Estados Unidos da América, decidiu permitir o acesso à Internet para propósitos comerciais. Quando essa decisão foi tomada, havia um ressentimento entre as comunidades de pesquisa e militar, pois se achava que os tempos de resposta ficariam péssimos, devido à "rede" poder ficar saturada com tantos usuários. [H.M. Deitel, 2003]

E, na verdade, ocorreu o oposto. As empresas rapidamente perceberam que, fazendo uso eficaz da Internet, elas poderiam refinar suas operações e oferecer novos e melhores serviços para seus clientes. As empresas

começaram a gastar enormes quantidades de dinheiro par desenvolver e aprimorar sua presença na Internet. Isso gerou uma feroz concorrência entre as operadoras de comunicação e os fornecedores de hardware e software, para atender à crescente demanda por infra-estrutura. O resultado é que a largura de banda (isto é, a capacidade de transporte de informações das linhas de comunicação) na Internet aumentou tremendamente, enquanto os custos do hardware caíram verticalmente. É amplamente aceito que a Internet desempenhou um papel significativo no crescimento econômico que muitas nações industrializadas experimentaram na ultima década. [H.M. Deitel, 2003]

A World Wide Web permite que os usuários de computador localizem e vejam documentos baseados em multimídia (isto é, documentos com textos, elementos gráficos, animações, áudios e/ou vídeos) sobre praticamente qualquer assunto. Embora a Internet tenha se desenvolvido há mais de 30 anos, a introdução da World Wide Web (WWW) foi um evento relativamente recente. Em 1989, Tim Berners-Lee, da CERN (a organização européia de pesquisa nuclear), começou a desenvolver uma tecnologia para compartilhar informações por meio de documentos de texto com links. Baseando a nova linguagem no bem estabelecido SGML (Standard Generalized Markup Language – linguagem de marcação generalizada padrão) – um padrão para o intercambio de dados comerciais -, Berners-Lee chamou sua invenção de HTML (HyperText Markup Language – linguagem de marcação de hipertexto). Ele também escreveu protocolos de comunicação para formar a espinha dorsal de seu novo sistema de informações de hipertexto, que chamou de World Wide Web. [H.M. Deitel, 2003]

2.2.3. Banco de Dados

Podemos entender por Banco de Dados, qualquer sistema que reúna e mantenha bem organizada uma massa de informações. Onde nessa massa de informações, possuímos alguns conceitos como: tabelas, registros, colunas, relacionamentos, campos e etc.

Um SGBD (Sistema Gerenciador de Banco de Dados) é usado para armazenar as informações de uma forma que permita a pesquisa e consulta de diversas maneiras. Esse SGBD adota um modelo de dados, que, atualmente, o mais adotado é o modelo relacional, onde as estruturas têm forma de tabelas, compostas por linhas e colunas.

Os Bancos de Dados são o método preferencial de armazenamento de informações em aplicações multiusuário. Entretanto, podem ser comumente utilizadas para muitos programas de correio eletrônico e organizadores pessoais, por exemplo, que baseiam suas tecnologias padronizadas em Banco de Dados.

2.2.3.1. Microsoft SQL Server

Um aplicativo de Banco de Dados é um software exclusivo para gerenciamento de um Banco de Dados. Estes softwares, geralmente, abrangem uma variedade de necessidades e objetivos específicos, fornecendo uma interface para gerenciamento e manipulação de um Banco de Dados.

Um exemplo de aplicativo de Banco de Dados é o Microsoft SQL Server, um software desenvolvido pela Microsoft para o gerenciamento do Banco de Dados relacionais. Neste software, a Microsoft fornece uma plataforma de dados produtiva, robusta, confiável e que permite que você execute alguns aplicativos de missão crítica, reduzindo o tempo e o custo com o desenvolvimento e gerenciamento de aplicações.

2.2.4. ASP.NET

O ASP.NET é uma linguagem de programação voltada para a Web e baseada no .NET Framework. Assim, herda todas as características deste Framework, podendo ser escrita em várias linguagens, como C# e Visual Basic .NET.

É através de um componente do IIS (Internet Information Service) que o .NET Framework é interpretado, possibilitando a exibição de paginas Web dinâmicas.

O .NET Framework é uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código escrito e gerado pelo .NET poderá ser executado em qualquer dispositivo que possua o framework instalado.

2.2.5. Google Maps API

A API (Application Programming Interface) do Google Maps, disponível gratuitamente na internet, possibilita traçar rotas e definir a localização de um determinado dispositivo no mapa.

É uma interface que roda pela Web e disponibiliza um número muito grande de utilidades e ferramentas, para manipulação de mapas. E, ao adicionarmos conteúdo e variáveis a esses mapas, pode-se consumir uma variedade de serviços gratuitamente, como distância entre dois pontos, melhor rota para um determinado destino, entre outros.

CAPÍTULO 3 – DESENVOLVIMENTO

Este capítulo trata das especificações técnicas dos dispositivos envolvidos neste projeto e da implementação.

Primeiramente fala-se à respeito da descrição técnica de cada dispositivo, em seguida, será apresentada a topologia do projeto. Com esta breve descrição técnica, será explicado o desenvolvimento do software utilizado no celular para efetuar a conexão com o GPS, da interface WEB que foi desenvolvida para receber os dados que o celular enviará, para ser gravado no Banco de Dados.

E, por fim, a interface WEB desenvolvida para o suposto usuário que contrataria o serviço de rastreamento disponível, a fim de obter a rota do seu carro traçada em um determinado período de tempo, através do Google Maps.

3.1. Especificações Técnicas

Nesta seção são apresentados todos os dispositivos utilizados e suas especificações técnicas.

3.1.1. Telefone Celular Nokia N73

O modelo utilizado possui um sistema operacional Symbian OS v9.1 e uma plataforma de desenvolvimento S60 3rd Edition que possibilita desenvolver um software na linguagem Java para ser instalado no celular e obter os dados do GPS. Ilustra-se o celular na Figura 3.1.

Figura 3.1 – Celular Nokia N73

Possui suporte de dados às tecnologias WCDMA, HSCSD, CSD, EGPRS e GPRS. Esta última, a tecnologia utilizada para enviar os dados obtidos para a internet.

Além dessas tecnologias, suporta conexões por Infravermelho, USB 2.0 (através de cabo Nokia CA-53) e Bluetooth, que se utiliza para conectar o celular ao GPS.

3.1.2. Receptor GPS Bluetooth (BT-572F)

Conforme a ilustração da Figura 3.2 e 3.3, o dispositivo Receptor GPS foi escolhido por seu tamanho compacto, baixo custo e pela disponibilidade da tecnologia Bluetooth, não sendo necessário o uso de cabos para a comunicação entre o dispositivo e o telefone celular.

Figura 3.2 – Receptor GPS BT-572F

É um GPS desenvolvido para obter um ótimo desempenho a baixo custo. Possui 44 canais disponíveis para recepção de dados via satélite, um curto tempo de inicialização e uma rápida aquisição de sinal.

Figura 3.3 – Receptor GPS BT-572F na palma da mão

O Bluetooth versão 2.0 possibilita uma conexão com outro dispositivo numa distância de até 10 metros. Possui três led`s indicadores nas cores Azul (Status do Bluetooth), Vermelho (Carga da bateria baixa) e Verde (Carregando bateria).

3.1.2.1. Bluetooth

O Bluetooth, nada mais é do que uma tecnologia que permite uma comunicação simples, rápida e segura entre dispositivos como: computadores, smartphones, telefones celulares, mouses, fones de ouvido, entre outros.

Bluetooth é um padrão global de comunicação sem fio e de baixo consumo de energia e que possibilita a comunicação e a transmissão de dados, em modo full-duplex, entre dois dispositivos compatíveis com a tecnologia. A transmissão de dados é feita através de radiofreqüência, possibilitando que um dispositivo encontre o outro, independente de sua localização, limitando-se apenas, a estar dentro do limite de abrangência.

Além disso, como o Bluetooth foi criado para funcionar em qualquer parte do mundo, fez-se necessária a adoção de uma freqüência de rádio aberta, que fosse padrão em qualquer lugar do planeta. O Bluetooth utiliza basicamente dois padrões para efetuar a ligação entre emissor e receptor. O SCO (Synchronous Connection-Oriented) e ACL (Asynchronous Connection-Less).

O SCO é responsável por estabelecer uma conexão sincronizada entre o dispositivo master e o dispositivo escravo. Desta forma, normalmente o SCO acaba sendo utilizado para o envio contínuo de dados, como voz. Por funcionar dessa forma, o SCO não permite o reenvio de pacotes de dados perdido.

O ACL por sua vez, estabelece uma conexão assíncrona, entre emissor e receptor. Desta forma, permite o reenvio de pacotes de dados perdidos, garantindo a integridade das informações trocadas entre os dispositivos.

3.2. Topologia do Projeto

A topologia a ser utilizada no projeto está descrita na Figura 3.4.

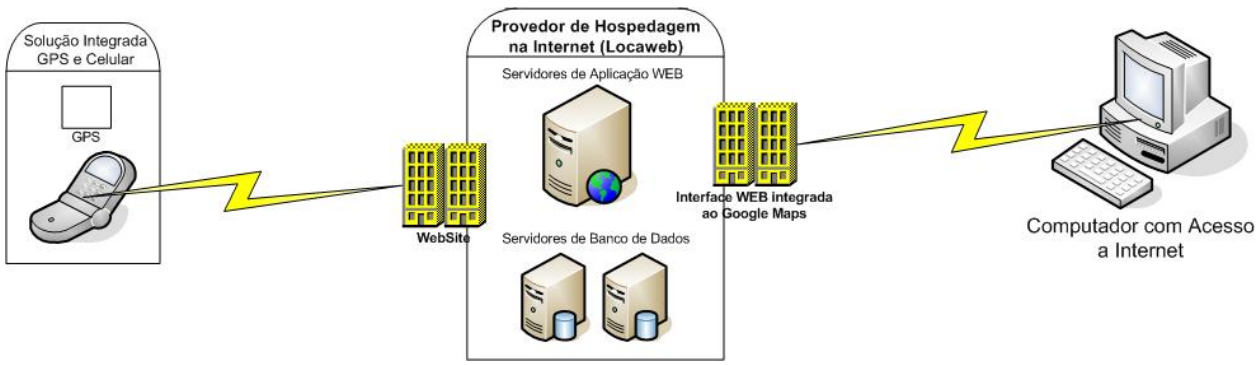


Figura 3.4 – Topologia do Projeto

Visualizam-se duas vertentes principais na topologia deste projeto. A primeira delas sendo a solução de integração entre Celular e GPS, que envolve desenvolvimento de software para efetuar essa integração entre os dois dispositivos. A segunda, trata-se de um provedor de hospedagem em que

disponibilizaremos a Interface Web responsável por comunicar o usuário a respeito da localização dos dispositivos GPS.

3.3. Aplicativo Java - Interface do Celular

Esta, sem dúvida, foi a parte mais desafiadora de todo o projeto, pois envolvia uma linguagem de programação que não possuía tanta afinidade e não havia outra alternativa a não ser desenvolver um software que se conectasse ao GPS através desta linguagem.

Aprimorando-se o conhecimento sobre tecnologia Bluetooth e também o conhecimento básico sobre a linguagem de programação Java, tornou-se possível o desenvolvimento deste aplicativo.

Durante as pesquisas na internet, foram encontradas classes para realizar a conexão de dispositivos que possuíssem Bluetooth. Tal classe foi desenvolvida e publicada em um WebSite da internet por Sergio Estevão. [http://sergioestevao.com/blogs/midp – 2008]

Das classes encontradas, foram utilizadas especificamente as classes que efetuam a pesquisa de dispositivos GPS e que gerenciam a conexão que pode ser realizada entre os dispositivos.

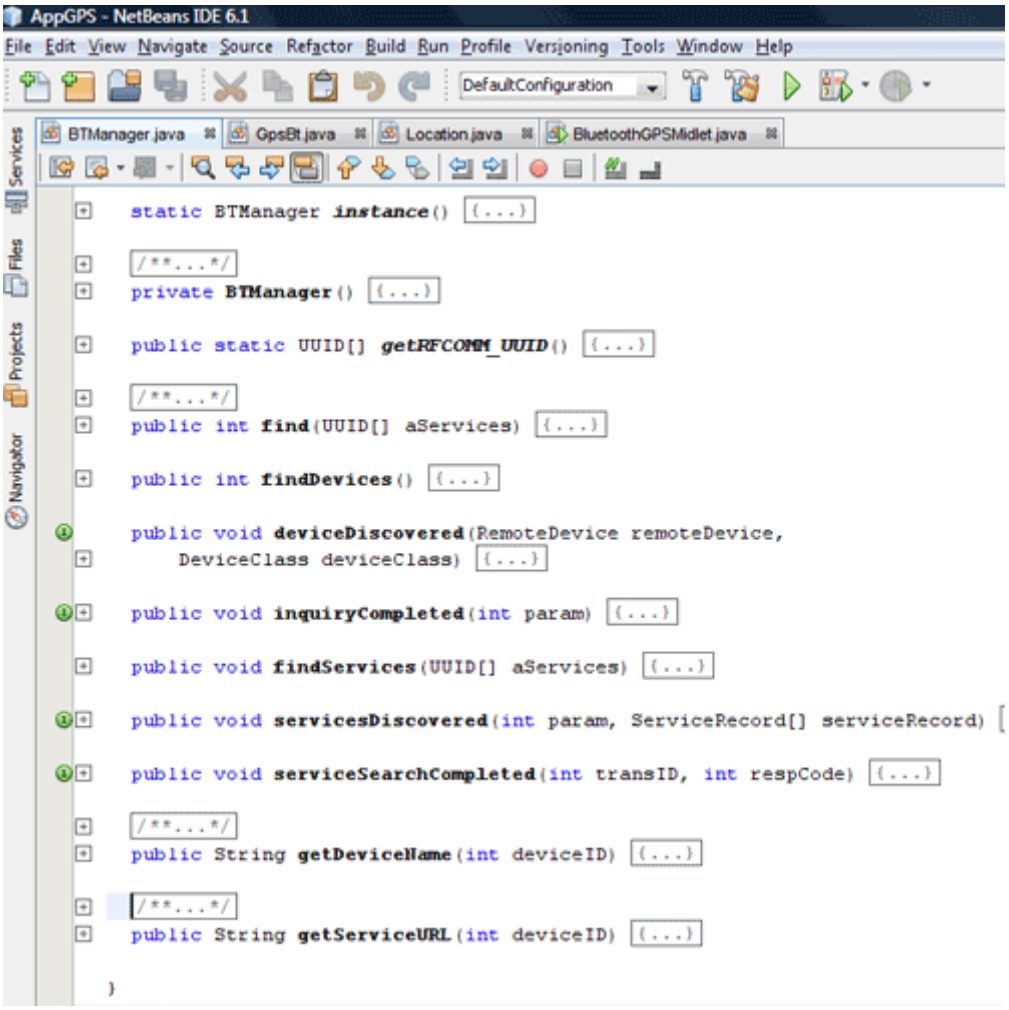


Figura 3.5 – Classe BTManager e métodos disponíveis

Na Figura 3.5, ilustra-se a classe responsável por gerenciar a conexão Bluetooth do celular. Existem vários métodos dentro da classe, porém, serão apresentados alguns exemplos, de acordo com sua relevância para o entendimento:

- *getRFCOMM_UUID()* – Este método nos retornar um array de objetos que utilizam um SPP (Serial Port Profile – *Perfil de porta Serial*) para realizar conexão com outros dispositivos, isso, devido à especificação feita no método, que o retorno deveria conter apenas dispositivos desta natureza, que no caso é especificado pelo UUID (Universally Unique Identifier – *Identificador Único Universal*) 0x1101;
- *findDevices*() – Encontra todos os dispositivos que estão contidos no array de objetos encontrados no método anterior.

- *getDeviceName(int deviceID)* – Depois de encontrado os dispositivos Bluetooth disponíveis para conexão, utilizamos este método para mostrar seus respectivos nomes.

Outra classe importante é a GpsBt que fica responsável por estabelecer a conexão entre os dispositivos depois de encontrados. A Figura 3.6 mostra os métodos encontrados nesta classe.

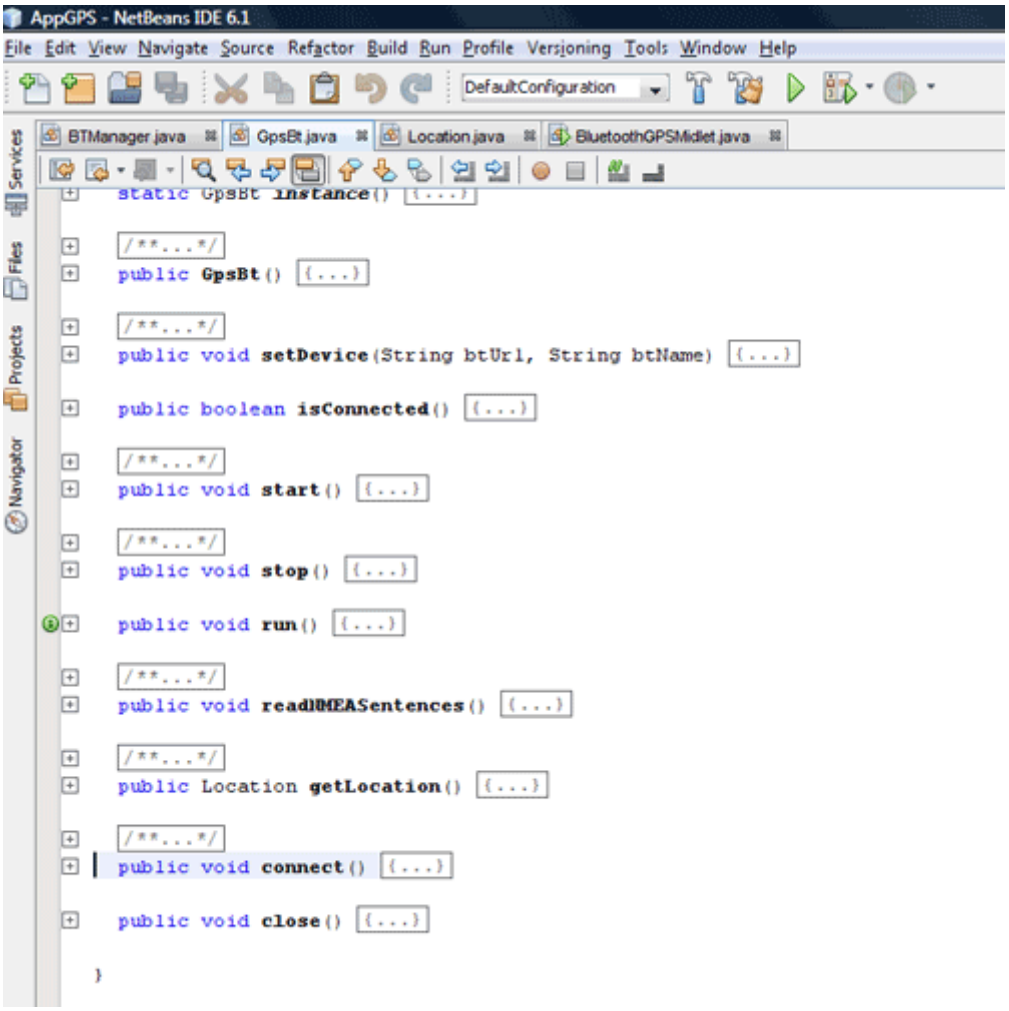


Figura 3.6 – Classe GpsBt e seus métodos

Para estabelecer a conexão, esta classe é instanciada na tela principal do aplicativo e através do método *start()*, é inicializada uma Thread que faz a chamada do método *run()*. Com isso, torna-se possível efetuar a conexão via Bluetooth com o dispositivo encontrado e selecionado pelo usuário através da tela principal.

A partir deste instante já é possível obter os dados geográficos de localização através do dispositivo GPS conectado ao celular. Desta maneira, torna-se necessária a utilização de outra classe que faz a leitura correta dos dados que são obtidos no GPS, ilustrada na Figura 3.7. Da forma como o GPS obtém e transmite os dados recebidos do satélite, quando se efetua a leitura sem o uso de um software geográfico, jamais poderia ser compreendido por se tratar de uma seqüência de caracteres que formam uma sentença. Esta sentença precisa ser traduzida para o bom entendimento e a posterior aplicação em um software qualquer de posicionamento geográfico.

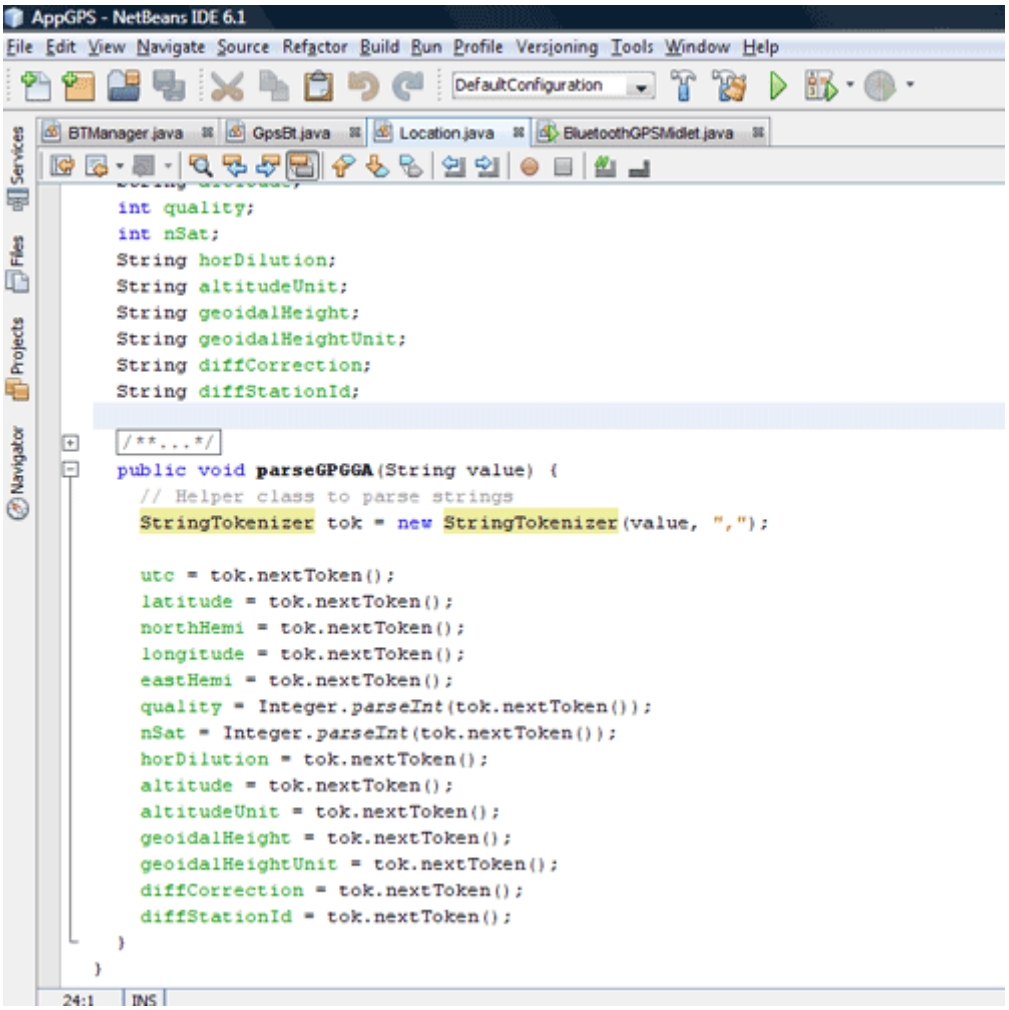


Figura 3.7 – Classe Location

Além de declarar diversos atributos (variáveis da classe), esta classe possui apenas um método responsável por separar os dados contidos na sentença que é obtida do GPS e atribuir seus devidos valores a cada atributo.

Cada atributo será utilizado para preencher a tela principal com o valor que vai ser mostrado para o usuário, pois uma vez que a classe seja instanciada e o método tenha sido executado, todos os atributos possuirão valores para serem exibidos na tela.

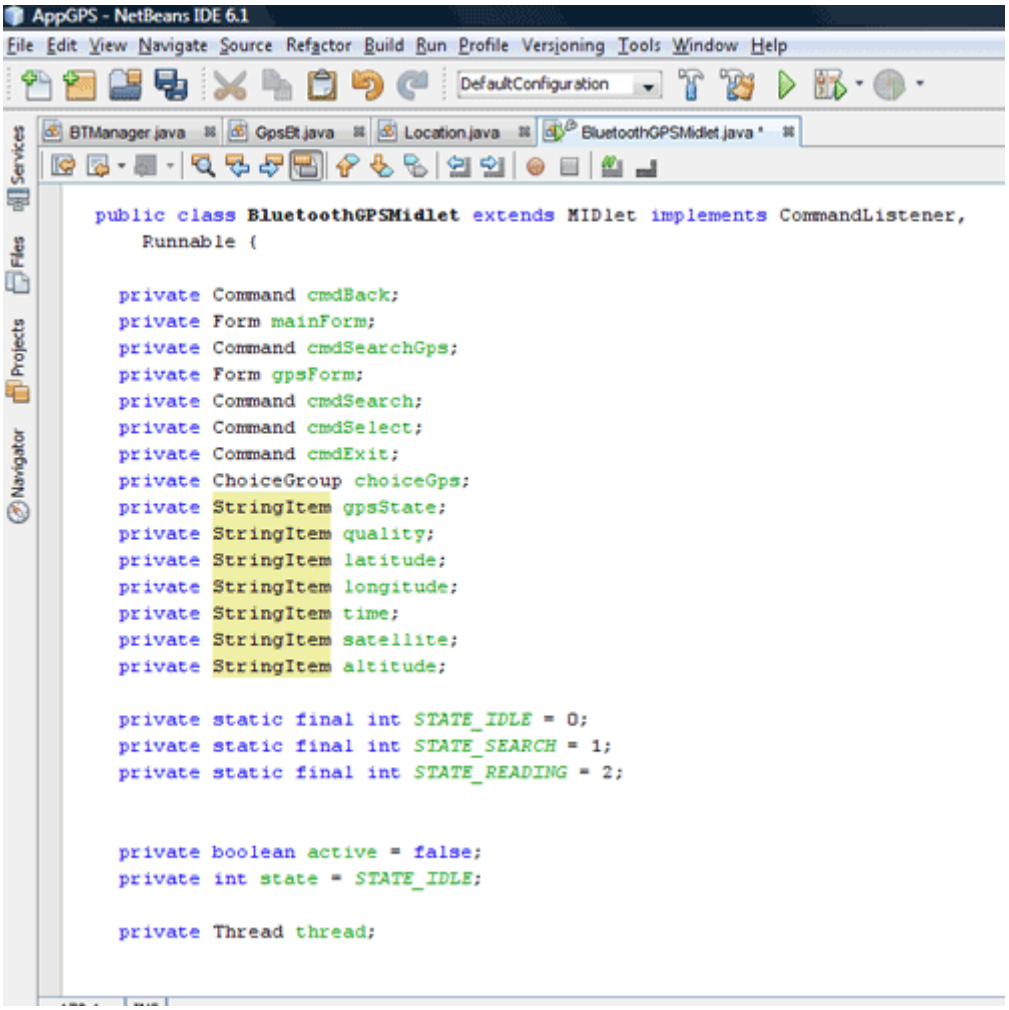


Figura 3.8 – Classe BluetoothGPSMidlet

Na Figura 3.8 e 3.9 ilustra-se o código fonte do formulário que exibe todos os valores para o usuário. Essa é a tela principal do aplicativo desenvolvido.

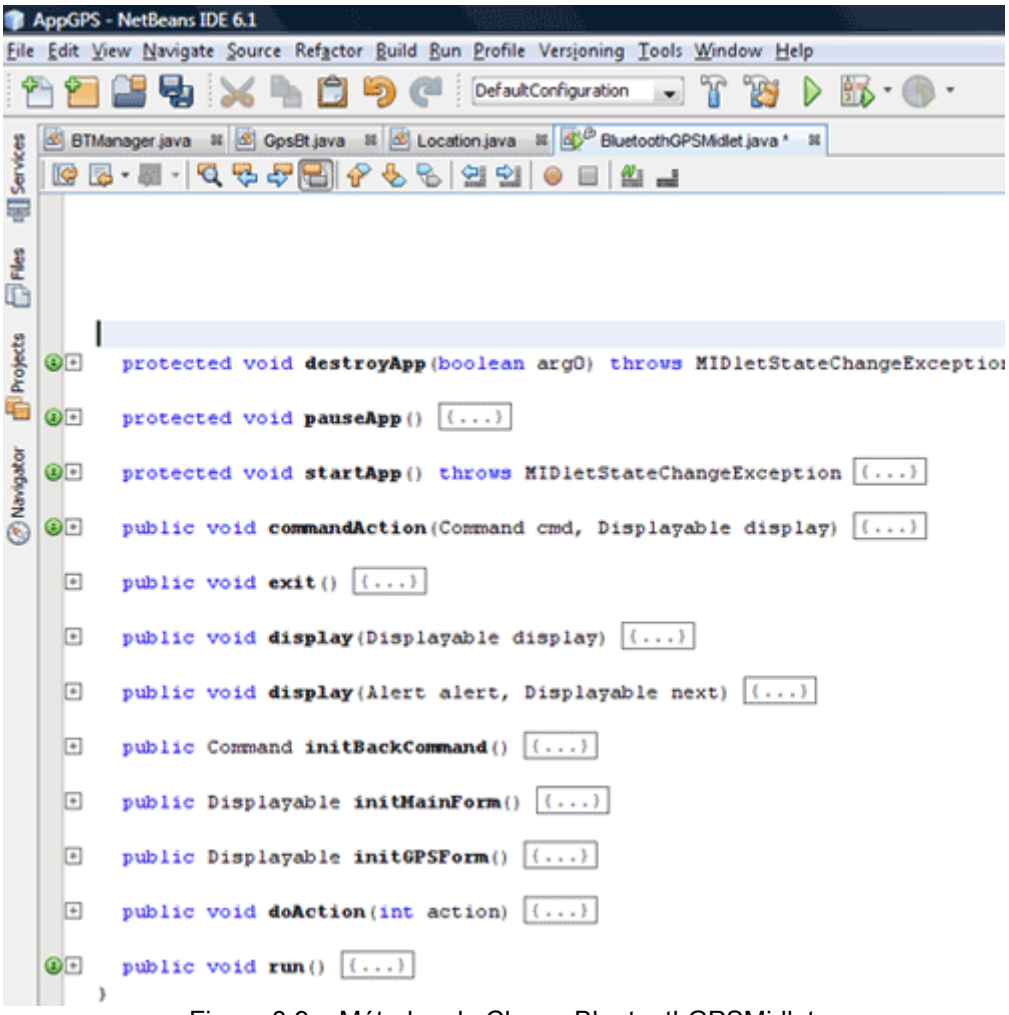


Figura 3.9 – Métodos da Classe BluetoothGPSMidlet

Ao desenvolver um software ou aplicação na linguagem Java utilizando J2ME, MidLet é um termo usado para designar o formulário que será visualizado pelo usuário. Por analogia, podemos comparar a uma página web que visualizamos através de um navegador qualquer, como Internet Explorer, FireFox, entre outros.

Quando se cria um MidLet alguns métodos são criados automaticamente, pela própria ferramenta de desenvolvimento, como: *destroyApp()*, *pauseApp()*, *startApp()*.

Os outros métodos foram utilizados para executar ações de controle e interação com o usuário.

- *initMainForm()* – é responsável por mostrar para o usuário no momento em que o aplicativo começa a ser executado no celular,

os dados do formulário inicial que segue basicamente um padrão e pode ser customizado, que não foi o caso neste projeto.

- *initGPSForm()* – Depois que o aplicativo é inicializado e o formulário inicial é carregado, este método é executado para mostrar os dados do GPS que serão visualizados pelo usuário com os campos, Status, Qualidade do sinal GPS, Satélites, Latitude, Longitude e etc.
- *CommandAction()* – principal função detectar qual a ação que o usuário realizou ao clicar em alguns dos botões do celular que correspondem ao menu disponível no formulário da tela principal deste aplicativo.

Figura 3.10 – Método run() usado no MidLet para mostrar dados do GPS

- *run()* – este método é responsável por instanciar a classe *GpsBt* que conseqüentemente irá percorrer internamente todas as classes que foram mostradas anteriormente, através da herança

entre cada uma delas, mostrando assim os dados obtidos através do GPS na tela principal do aplicativo.

3.3.1. Demonstração prática

Na Figura 3.11 ilustra-se a utilização prática, depois de instalado o aplicativo no celular.

Figura 3.11 – Abrindo Aplicativo instalado no celular, BluetoothGPS

Abrindo a listagem de aplicativos instalados no celular, escolhe-se a opção que se refere ao aplicativo desenvolvido que foi instalado, *BluetoothGPS*.

Na Figura 3.12 pode-se observar a tela principal onde serão visualizados os dados obtidos do GPS.

Figura 3.12 – Tela inicial do Aplicativo

Como na Figura 3.12 a conexão ainda não estava estabelecida, clica-se no botão *Procurar*, em seguida, abre-se o formulário de pesquisa mostrado na Figura 3.13.

Figura 3.13 – Tela de Procura de dispositivos GPS

Logo em seguida, escolhe-se a opção de *Pesquisar* conforme a Figura 3.14.

Figura 3.14 – Executando comando para iniciar pesquisa de dispositivos

Após selecionar a opção pesquisar, o aplicativo *BluetoothGPS* irá iniciar a procura por dispositivos com Bluetooth ativado e disponível para conexão. Para que o aplicativo não fique parado sem que o usuário seja informado do que está acontecendo, uma mensagem fica passando na parte superior do formulário informando que a pesquisa por dispositivos está sendo realizada, conforme podemos visualizar na Figura 3.15.

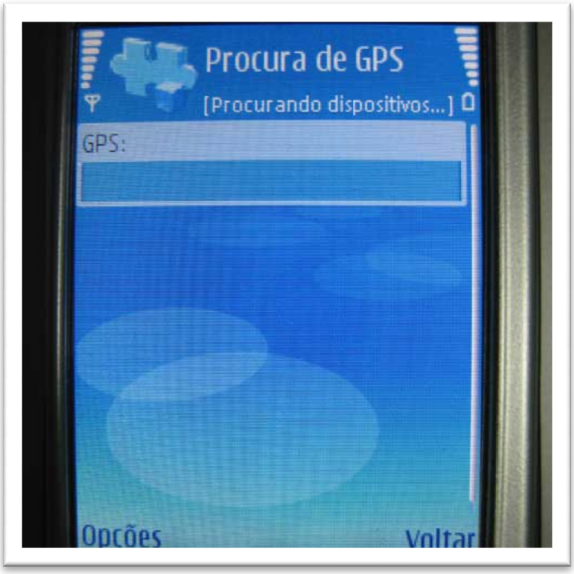


Figura 3.15 – Aplicativo durante a pesquisa de dispositivos

Após cerca de 5 segundos o dispositivo GPS é encontrado e aparece na tela para ser selecionado dentre quaisquer outros dispositivos que pudessem vir a serem encontrados, ilustrado na Figura 3.16.

Figura 3.16 – Resultado da pesquisa. Listando o(s) dispositivo(s) encontrado(s)

Com a lista de dispositivos preenchida após a pesquisa, escolhe-se o menu Opções e seleciona a opção Selecionar, em seguida, o aplicativo volta à tela principal já com a conexão entre o celular e o dispositivo GPS realizada, conforme ilustrado na Figura 3.17.

Figura 3.17 – Celular conectado ao GPS

Porém, o GPS estava conectado, mas nenhum sinal de latitude e longitude havia sido obtido, motivo que pode ser justificado pela localização do GPS, na janela e dentro de casa, conforme a Figura 3.18.

Figura 3.18 – GPS Piscando após conexão estabelecida

Pelo fato de não ter sido possível obter os dados de latitude e longitude, o GPS foi reposicionado. E, conforme as Figuras 3.19 e 3.20, o GPS encontrava-se sobre o telhado da casa vizinha.

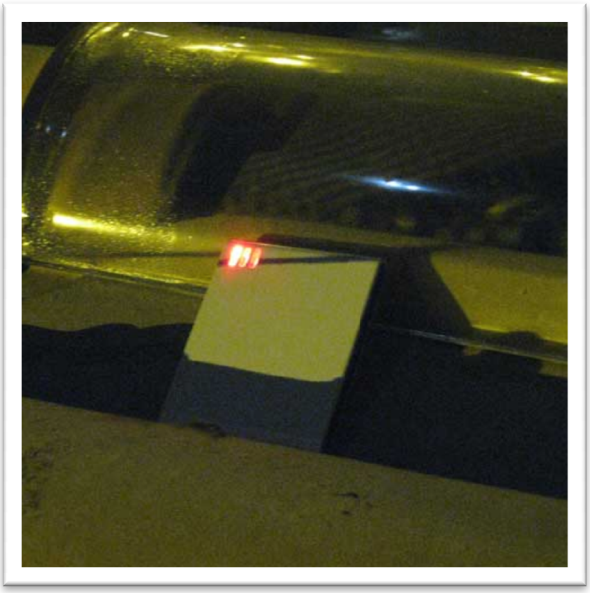


Figura 3.19 – GPS posicionado no telhado do vizinho

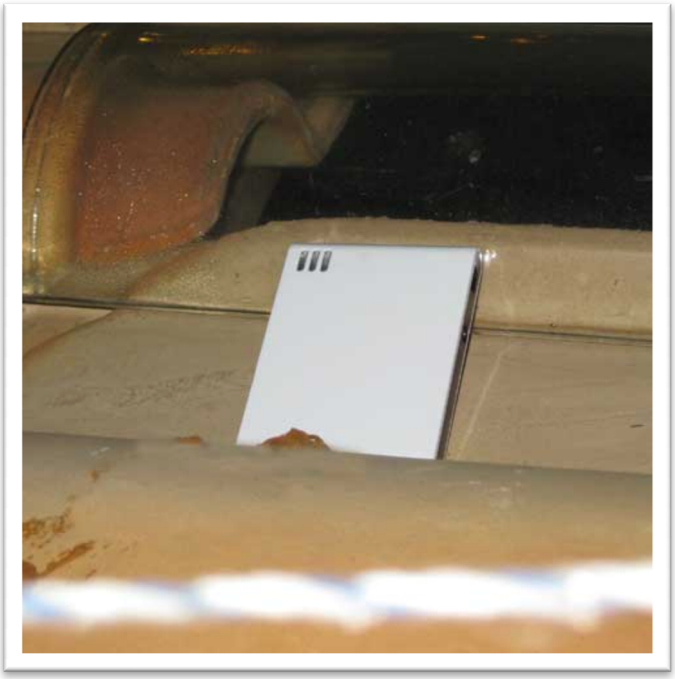


Figura 3.20 – GPS no telhado do vizinho

Na Figura 3.21, verifica-se a comprovação dos resultados após a mudança de posição do GPS, para um local aberto sem obstáculos de construção como o teto de uma casa, paredes e etc.

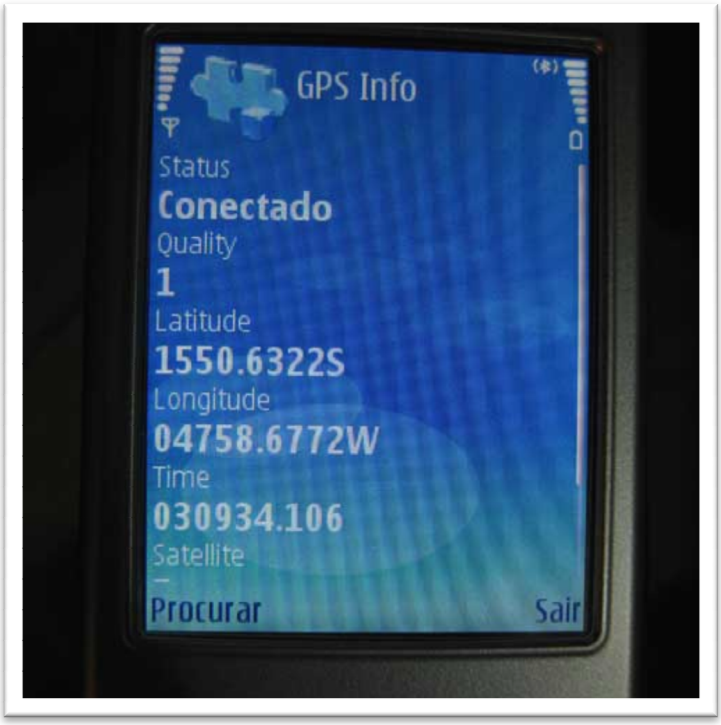


Figura 3.21 – Tela principal do aplicativo após reposicionamento do GPS

Conforme pode-se observar na figura 3.21, a coordenada geográfica de latitude era de 1550.6322S e de Longitude de 04758.6772W. Com essa localização realizou-se a demonstração através do Google Maps da localização deste ponto mostrado no celular. Para isso, precisamos definir algumas conversões.

3.3.2. Conversão de valores (NMEA para Decimal)

Para utilizar os dados obtidos através da conexão entre o celular e o GPS, em aplicações que permitem a visualização de uma determinada localização geográfica, é necessário converter do formato NMEA, interpretado por dispositivos GPS para o formato decimal, o qual, torna-se entendível na linguagem cartográfica.

A regra para conversão é simples. Devem-se separar os dois primeiros dígitos da sentença referente à coordenada de Latitude e separá-los do resto da sentença de modo que restem dois algarismos antes do sinal de ponto. Os algarismos separados representarão os graus, como na Equação 3.1.

Equação 3.1 – Conversão Latitude (1º Passo)

Em seguida, o restante dos algarismos da sentença referente à Latitude, deve ser dividido por 60, de acordo com a Equação 3.2.

$$50.6322 / 60 = 0.84387$$

Equação 3.2 – Conversão Latitude (2º Passo)

A Equação 3.3 mostra como é obtido o resultado final. Somando os dois primeiros algarismos que foram separados no primeiro passo desta conversão com o resultado obtido da divisão por 60.

$$0.84387 + 15 = 15.84387$$

Equação 3.3 – Conversão Latitude (3º Passo)

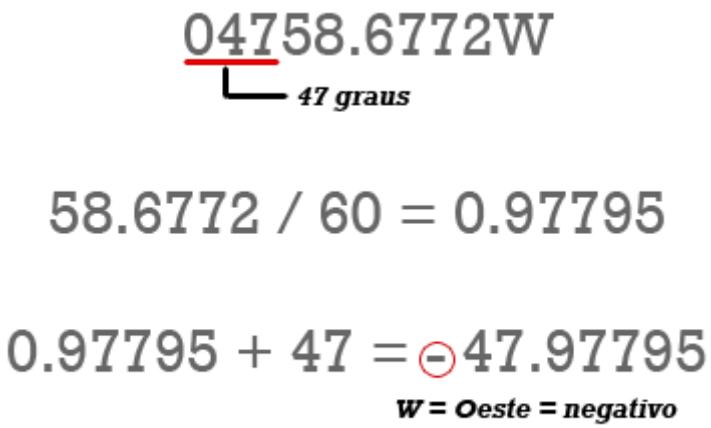
Por fim, como a letra informada na sentença do padrão NMEA é a letra S (Sul), por definição, representa a parte debaixo do Globo, em relação à linha do Equador, que é conhecido como Hemisfério Sul e representado numericamente com o sinal matemático negativo (-). E, também por definição, o lado esquerdo do Globo em relação ao meridiano de Greenwich, é chamado de Oeste e também é convencionado com o sinal negativo. Norte e Leste, conseqüentemente possuem o sinal matemático positivo (+) em sua representação numérica.

$$0.84387 + 15 = -15.84387$$

S = Sul = sinal negativo

Equação 3.4 – Conversão Latitude (4º Passo)

Realizando a mesma operação com a sentença que representa a Longitude do ponto informado pelo GPS, obtêm-se o resultado conforme a Equação 3.4.



Equação 3.5 – Conversão Longitude (Todos os Passos)

Com a representação no formato decimal de -15.84387 graus de Latitude e -47.97795 graus de Longitude, pode-se colocar no Google Maps, que será a ferramenta utilizada para visualização das coordenadas geográficas obtida pelo GPS. E assim, será visualizada a representação do nosso ponto de localização que no caso dos testes realizados, foi no Guará II. Essa ilustração pode ser visualizada nas Figuras 3.22 e 3.23.

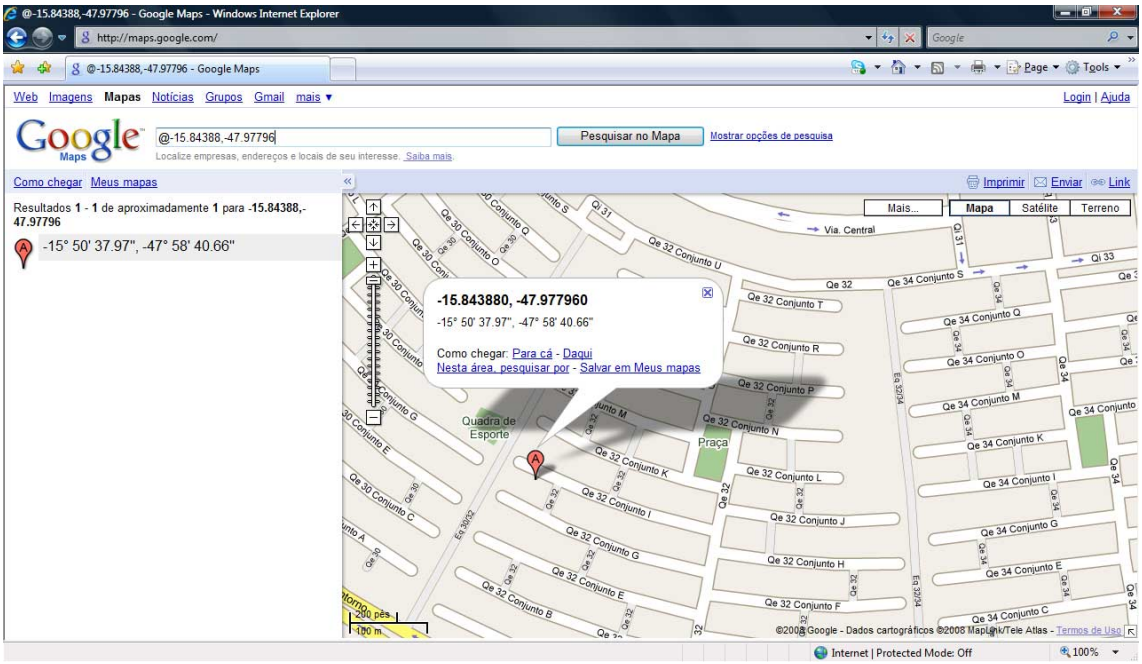


Figura 3.22 – Print Screen da tela do Computador

Demonstração com pouco Zoom na Figura 3.22. Aproximando-se do ponto, verifica-se o endereço da rua e o conjunto onde estava localizado o GPS.

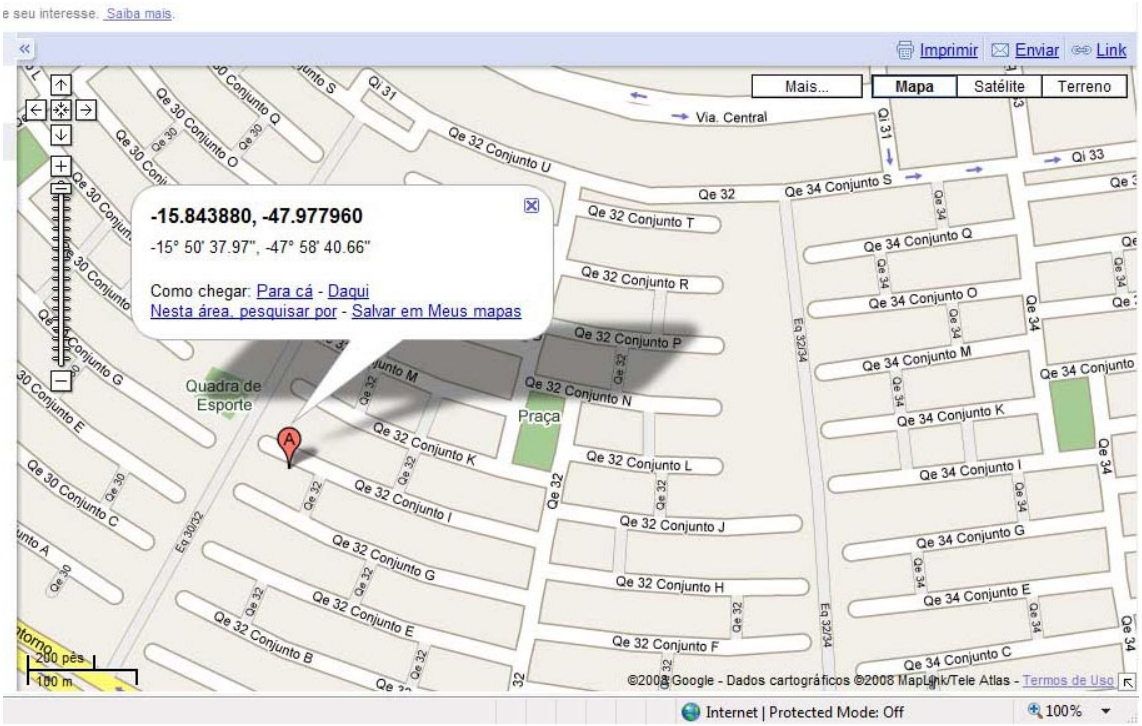


Figura 3.23 – Zoom no Print Screen para visualizar endereço das ruas

Desta forma, pode-se observar que a obtenção das coordenadas geográficas através do celular integrado com o GPS, funciona.

3.4. Aplicativo ASP.NET - Interface WEB

A interface WEB também chamada de WebSite, ficará disponível 24 (vinte e quatro) horas por dia, 7 (sete) dias por semana para ser acessado por qualquer pessoa que possua conexão com a internet.

A interface Web é responsável pela recepção dos dados que serão enviados pelo celular, pelo tratamento que envolve a conversão dos dados, conforme mostrado no item anterior deste capítulo, e, ainda, pelo armazenamento destas informações no Banco de Dados.

3.4.1. Banco de Dados

O Banco de Dados utilizado para o armazenamento das informações é o SQL Server 2005. Neste banco de dados teremos apenas uma tabela para o armazenamento das informações, conforme a Figura 3.24 que descreve as colunas desta tabela.

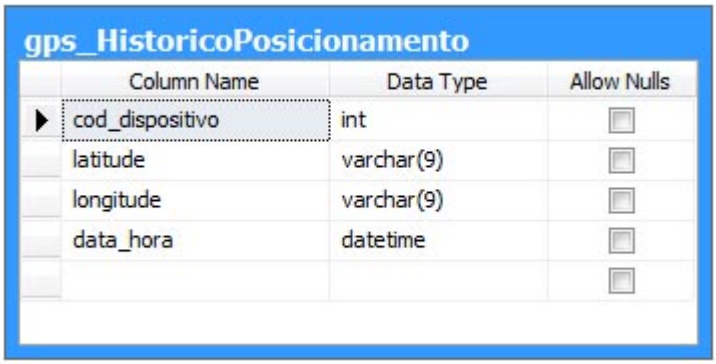


Figura 3.24 – Estrutura da tabela do Banco de Dados

A primeira coluna chamada *cod_dispositivo* representa um código enviado pelo celular do tipo numérico que na tabela corresponde a um número inteiro. Existem ainda, duas colunas chamadas *latitude* e *longitude*, que irão armazenar as coordenadas geográficas. Ambos são do tipo varchar com tamanho de 9 (nove) posições. E, por último, o campo *data_hora*, que registra a data e horário, com precisão de milissegundos, em que aquele dado foi inserido no Banco de Dados.

3.4.2. Recepção dos Dados

Uma vez que a conexão entre celular e GPS é estabelecida, os dados passam a ser enviados para nossa Interface WEB a cada 20 segundos ou qualquer outro intervalo de tempo que seja definido na aplicação Java, desenvolvida para o celular. Para tanto, é necessário existir uma página Web seja acessada pelo aplicativo Java, e os parâmetros da coordenada geográfica sejam enviados através da URL (endereço do WebSite) desta página, que chama-se *RecebeDados.aspx*.

Visualmente a página não mostra absolutamente nada. Caso efetue-se um teste abrindo a página em qualquer navegador, ela ficará em branco, pois a única função que possui, é de receber os dados que serão passados pela URL e retornar para o celular o horário que o registro foi inserido no Banco de Dados. A seguir demonstra-se como isso ocorre, visualizando o código-fonte da página.

Nesta ocasião já não é mais utilizada a linguagem Java para o desenvolvimento da aplicação. Utilizamos a linguagem C#, nativa do ASP.NET.

Na Figura 3.25 e 3.26 podem-se visualizar os métodos existentes na página que receberá os dados da localização do GPS no momento em que estiver sendo enviado pela internet através da conexão GPRS disponível na rede de telefonia celular.

```
14  public partial class RecebeDados : System.Web.UI.Page
15  {
16      protected void Page_Load(object sender, EventArgs e)
17      {
18          string latitude = Request.QueryString["latitude"];
19          string longitude = Request.QueryString["longitude"];
20
21          InsereCoordenadaGeografica(1,latitude, longitude);
22      }
23
24      private void InsereCoordenadaGeografica(int codDispositivo, string latitude, string longitude)
25      {
26          Base metodo = new Base();
27          string latitudeDecimal = defineSinal(latitude) + ConverteNMEA_Decimal(latitude);
28          string longitudeDecimal = defineSinal(longitude) + ConverteNMEA_Decimal(longitude);
29
30          metodo.ExecutaSQL("INSERT INTO gps_HistoricoPosicionamento (cod_dispositivo, latitude, longitude) VALUES " +
31              "(" + codDispositivo + ",'" + latitudeDecimal + "','" + longitudeDecimal + "')");
32      }
33
34      private string defineSinal(string valor)
35      {
36          string retorno = "";
37          string direcao = valor.Substring(valor.Length - 1, 1);
38          if (direcao == "S" || direcao == "W")
39          {
40              retorno = "-";
41          }
42
43          return retorno;
```

Figura 3.25 – Página RecebeDados.aspx (Parte 1)

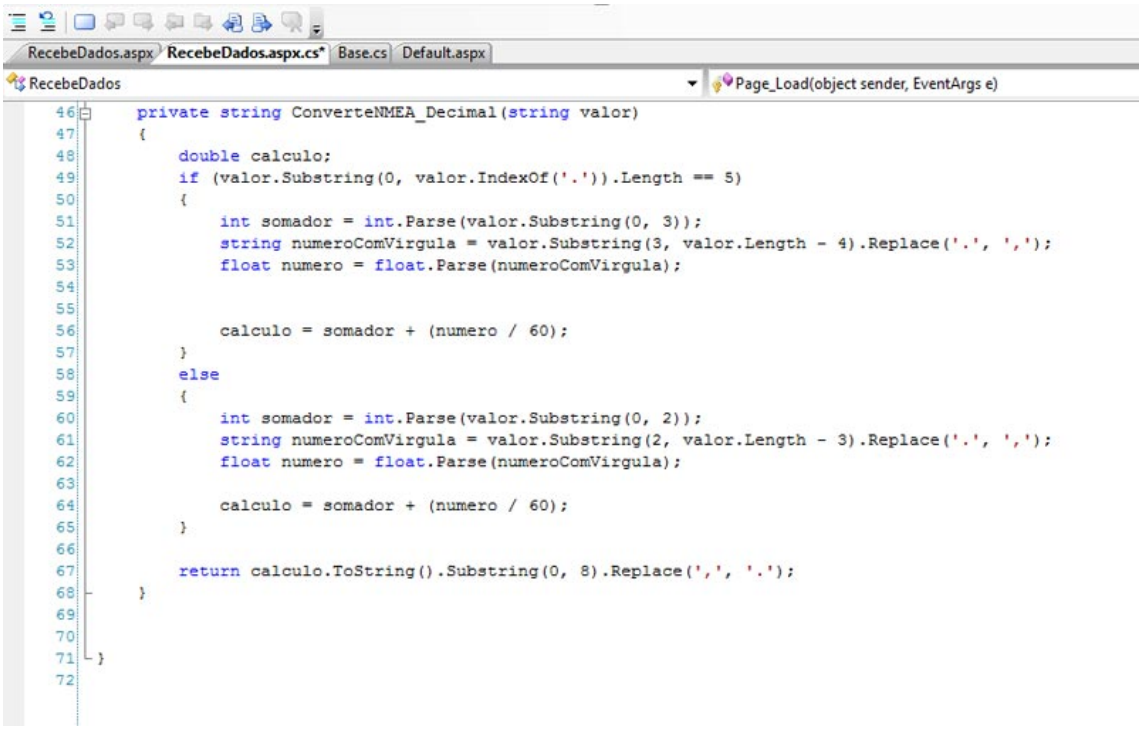


Figura 3.26 – Página RecebeDados.aspx (Parte 2)

O primeiro método é executado quando a página é carregada e é responsável por recuperar os dados que o celular enviar. Após a recuperação dos dados, chama-se o método *InsereCoordenadaGeografica()*. Com a utilização deste método a conversão dos dados do padrão NMEA para decimal e a definição do sinal (positivo ou negativo) para a coordenada geográfica, é feita.

Com todos os parâmetros definidos, de latitude, longitude, horário, número de satélites detectados entre outros, executa-se outro método responsável por inserir os dados no Banco de Dados, é o *ExecutaSQL()*. Para executar este método criou-se uma classe, denominada *Base.cs*, onde todos os métodos de acesso a Banco de Dados que seriam necessários, foram criados. Esta classe pode ser vista nas Figuras 3.27 e 3.28.

```
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace Util
{
    public class Base
    {
        private SqlConnection cn;

        public SqlConnection GeraConexao()
        {
            string strConexao = "SERVER=THIAGOLAPTOP;DATABASE=projeto_final;UID=sa;PWD=123;APP=PF
            try
            {
                if (cn.State == ConnectionState.Open)
                {
                    return cn;
                }
                else
                {
                    cn = new SqlConnection(strConexao);
                    cn.Open();
                    return cn;
                }
            }
            catch
            {
                cn = new SqlConnection(strConexao);
```

Figura 3.27 – Classe Base.cs (Parte 1)

```
                    cn = new SqlConnection(strConexao);
                    cn.Open();
                    return cn;
                }
            }
            catch
            {
                cn = new SqlConnection(strConexao);
                cn.Open();
                return cn;
            }
        }

        public SqlDataAdapter GeraAdapter(string strSQL)
        {
            cn = GeraConexao();
            SqlDataAdapter da = new SqlDataAdapter(strSQL, cn);

            return da;
        }

        public void ExecutaSQL(string strSQL)
        {
            cn = GeraConexao();
            SqlCommand cmd = new SqlCommand(strSQL, cn);

            cmd.ExecuteNonQuery();

            cn.Close();
            cn.Dispose();
```

Figura 3.28 – Classe Base.cs (Parte 2)

À medida que o celular envia os dados para o Banco de Dados, a interface Web os recebe e faz a inserção de cada um dos dados enviados na tabela existente.

Na Figura 3.29 pode-se visualizar o trecho de código, onde se utiliza um método da linguagem Java no aplicativo instalado no celular, que é responsável por enviar os dados via GPRS para Internet, na interface Web.

```
if (gpsBt.isConnected()){
  gpsState.setText("Conectado");
  Location location = gpsBt.getLocation();
  quality.setText(location.quality+"");
  latitude.setText(location.latitude + location.northHemi);
  longitude.setText(location.longitude + location.eastHemi);
  time.setText(location.utc);
  satellite.setText(location.nSat + "");
  altitude.setText(location.altitude + location.altitudeUnit);
      try {
          Connector.open("http://www.infodev.com.br/ProjetoFinal/RecebeDados.aspx?latitude="+
                  latitude.getText() +"&longitude="+ longitude.getText());
      } catch (IOException ex) {
          ex.printStackTrace();
      }
```

Figura 3.29 – Trecho de código Java onde é acessada a Interface Web

3.4.3. Demonstração Prática

Após a inserção de todos os dados obtidos através do celular conectado ao GPS, foi criada uma interface de visualização dos dados que foram registrados no Banco de Dados. Essa interface fica disponível na internet, através de um Website hospedado em um provedor de Hospedagem. Pressupõe-se que este serviço de hospedagem funcione ininterruptamente possibilitando ao usuário a visualização dos dados em qualquer horário e em qualquer local do mundo que possuir um meio de acesso à Internet.

Descreve-se abaixo as funcionalidades disponibilizadas na Interface Web para este projeto, bem como a página inicial desta Interface que podemos visualizar na Figura 3.30.

Figura 3.30 – Página inicial da Interface Web

3.4.3.1. Última Localização

Através da Interface Web, uma das funcionalidades disponibilizadas é a de obter a última localização do dispositivo em questão.

Independente do funcionamento do GPS, esta funcionalidade irá mostrar a última coordenada geográfica, definida por um ponto no Mapa, obtida pelo envio das informações capturadas pelo celular e enviadas para o Banco de Dados.

No caso do GPS estar em funcionamento, conectado ao celular e enviando regularmente informações para o Banco de Dados, este ponto será alterado a cada 30 segundos, de acordo com a movimentação do GPS.

Na Figura 3.31 é possível visualizar a forma com que será mostrado o último local, onde o celular enviou as coordenadas geográficas.

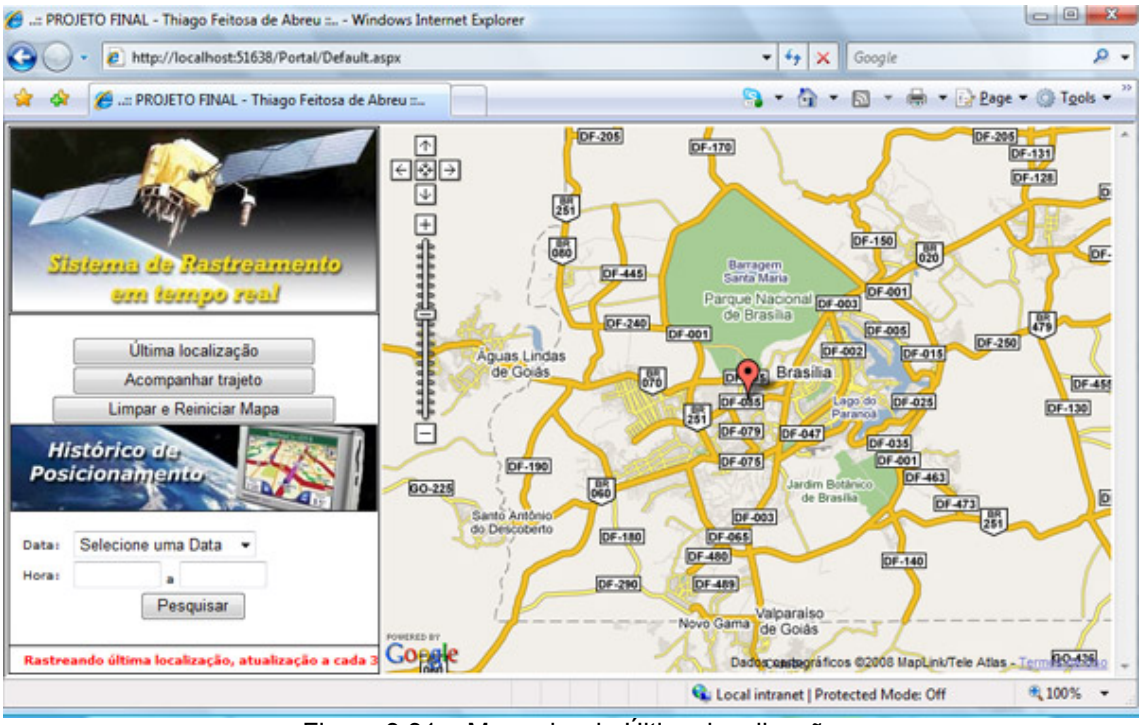


Figura 3.31 – Marcador da Última localização

Aumentando o zoom, visualiza-se com mais detalhes o último local onde o GPS foi localizado, conforme a Figura 3.32.

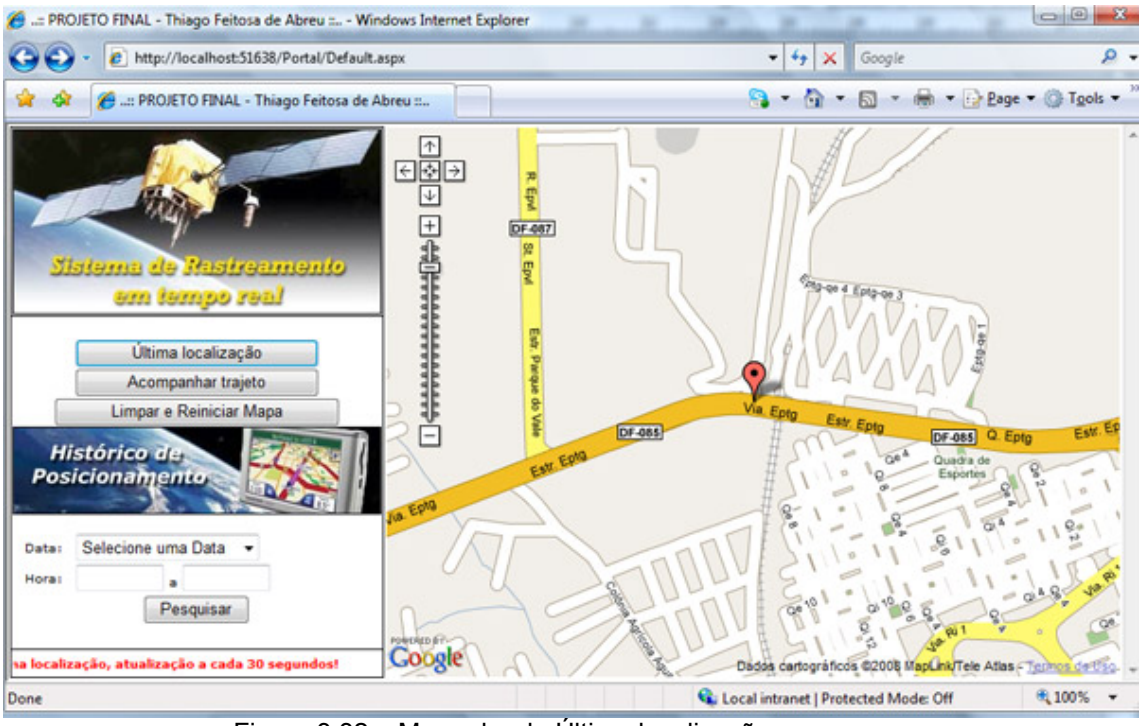


Figura 3.32 – Marcador da Última localização com zoom

Tecnicamente, a obtenção destes dados ocorre de forma bem simples. Efetua-se uma pesquisa, por meio de um comando SQL no Banco de Dados que irá trazer o último registro encontrado na tabela.

E, com o registro do Banco de Dados, usa-se o método GMarker() que recebe como parâmetros duas coordenadas, uma de latitude e outra de longitude, para assim, marcar no Mapa o ponto exato.

3.4.3.2. Rastreamento em Tempo Real

Esta é a funcionalidade mais dinâmica deste projeto, pois é possível realizar um rastreamento em tempo real do GPS, obviamente, enquanto estiver em funcionamento, conectado com o celular e efetuando o envio das informações para o Banco de Dados a cada 20 segundos.

Ao clicar no botão Acompanhar trajeto, o mapa é carregado com um marcador na última localização recebida no Banco de dados, e, a partir deste momento, sofrerá atualizações a cada 30 segundos para traçar a rota que estiver sendo realizada naquele momento pelo GPS, conforme as Figuras 3.33, 3.34, 3.35 e 3.36, que ilustram o trajeto sendo montado.

Figura 3.33 – Rastreamento em tempo real (Parte 1)

Na Figura 3.34 visualizam-se outros pontos durante o rastreamento.

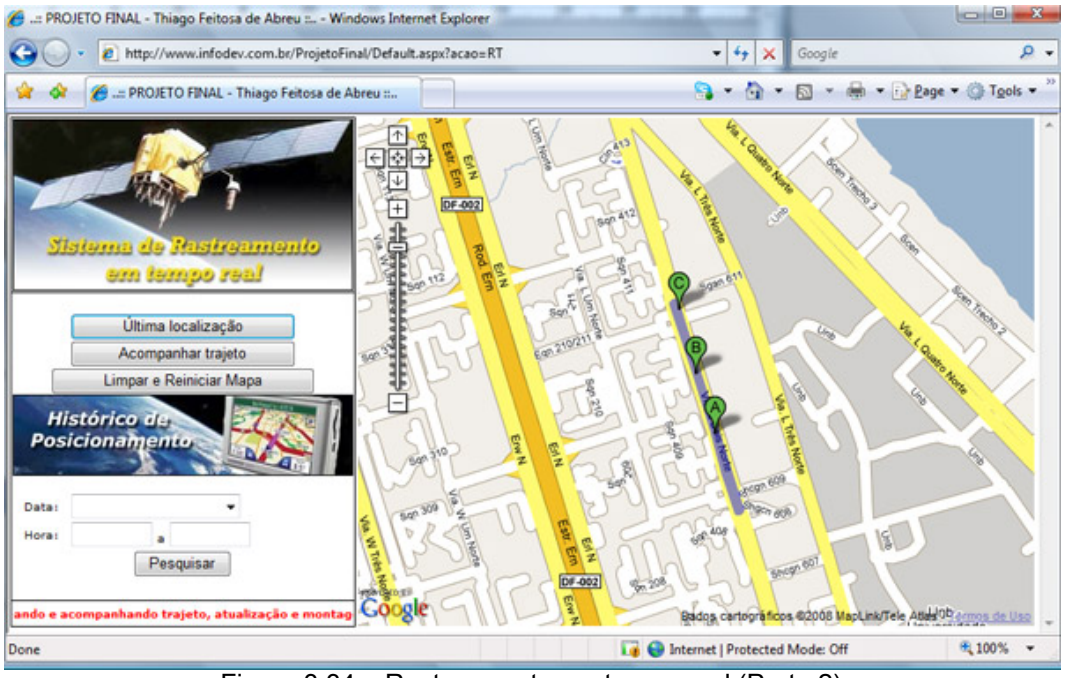


Figura 3.34 – Rastreamento em tempo real (Parte 2)

É notável na Figura 3.35 o desvio da rota realizado pelo próprio Google Maps. Fazendo-se uma análise cuidadosa nos pontos do mapa, será possível notar que os marcadores representados por letras do alfabeto seguem por uma via e parte do percurso está grafado em azul fora de qualquer uma das marcações com letras do alfabeto.

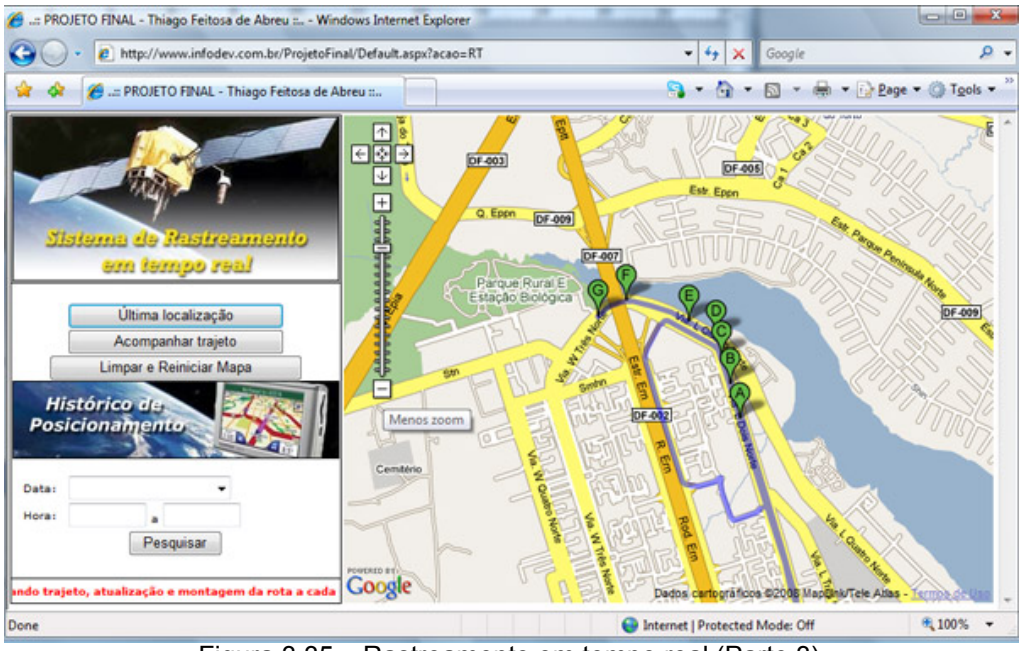


Figura 3.35 – Rastreamento em tempo real (Parte 3)

É um problema que ocorre durante o acompanhamento do GPS em tempo real e é descrito na seção dos problemas encontrados durante o desenvolvimento do projeto.

Após o rastreamento de um número maior de pontos de localização, a rota foi corrigida sem intervenção alguma da Interface Web, conforme pode-se notar na Figura 3.36.

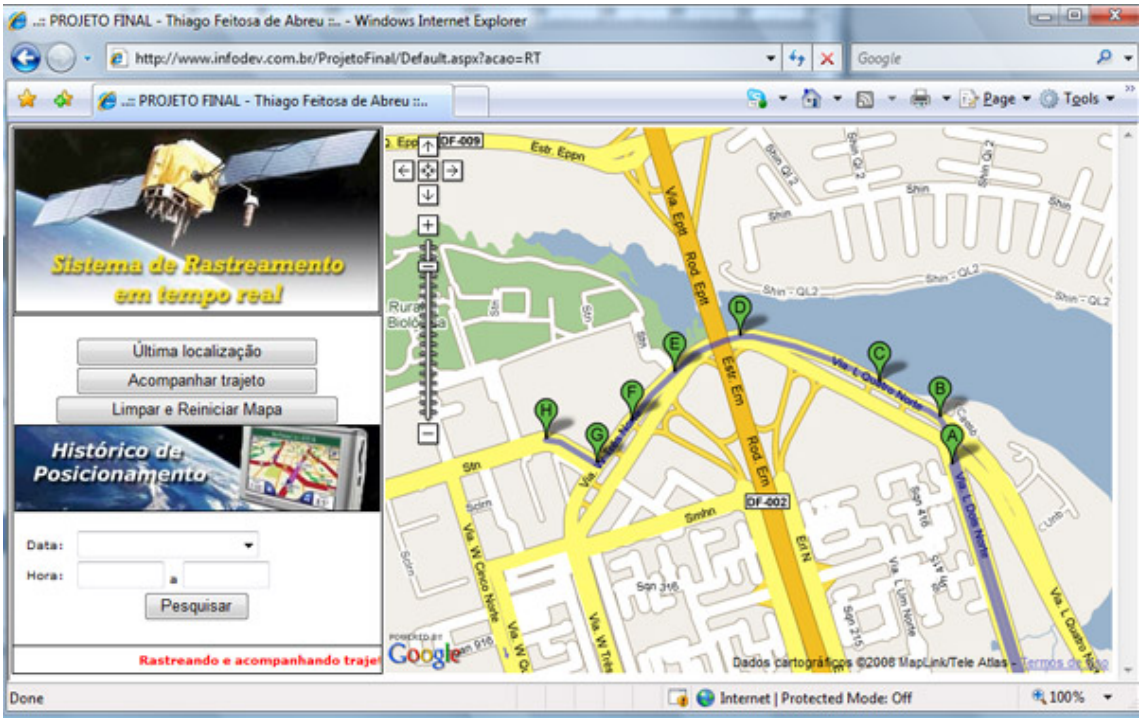


Figura 3.36 – Rastreamento em tempo real (Parte 4)

3.4.3.3. Histórico de Posicionamento

Esta opção permite consultar um histórico de posicionamento do GPS em uma determinada data, num espaço de tempo.

Conforme ilustrado nas Figuras 3.37 e 3.38, onde existem exemplos do preenchimento de Datas distintas, foi preenchido o formulário com os dados necessários, Data e intervalo de Hora, obtendo a visualização no Mapa, do trajeto que foi realizado.

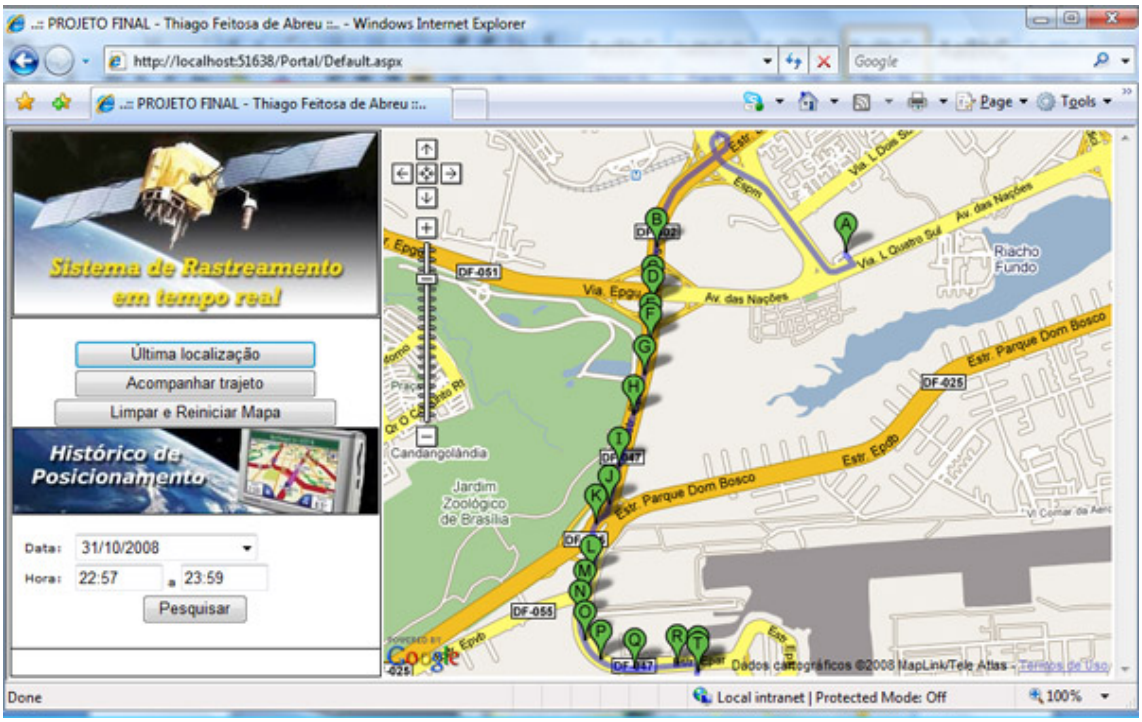


Figura 3.37 – Histórico de Posicionamento (Percurso 01)

Na Figura 3.38, mostra-se a pesquisa Histórica em datas e horários diferentes.

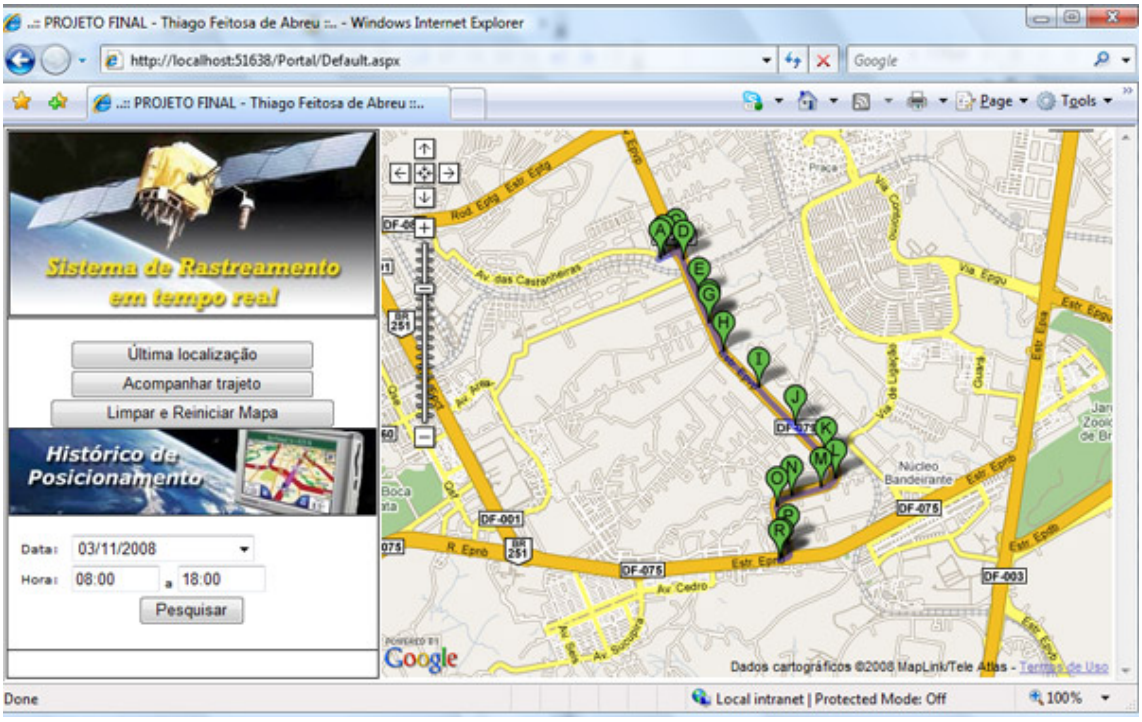


Figura 3.38 – Histórico de Posicionamento (Percurso 02)

Caso não sejam preenchidos os dados necessários, o sistema informa ao usuário que, para realizar a pesquisa é necessário o preenchimento de todos os campos do formulário, conforme a Figura 3.39.

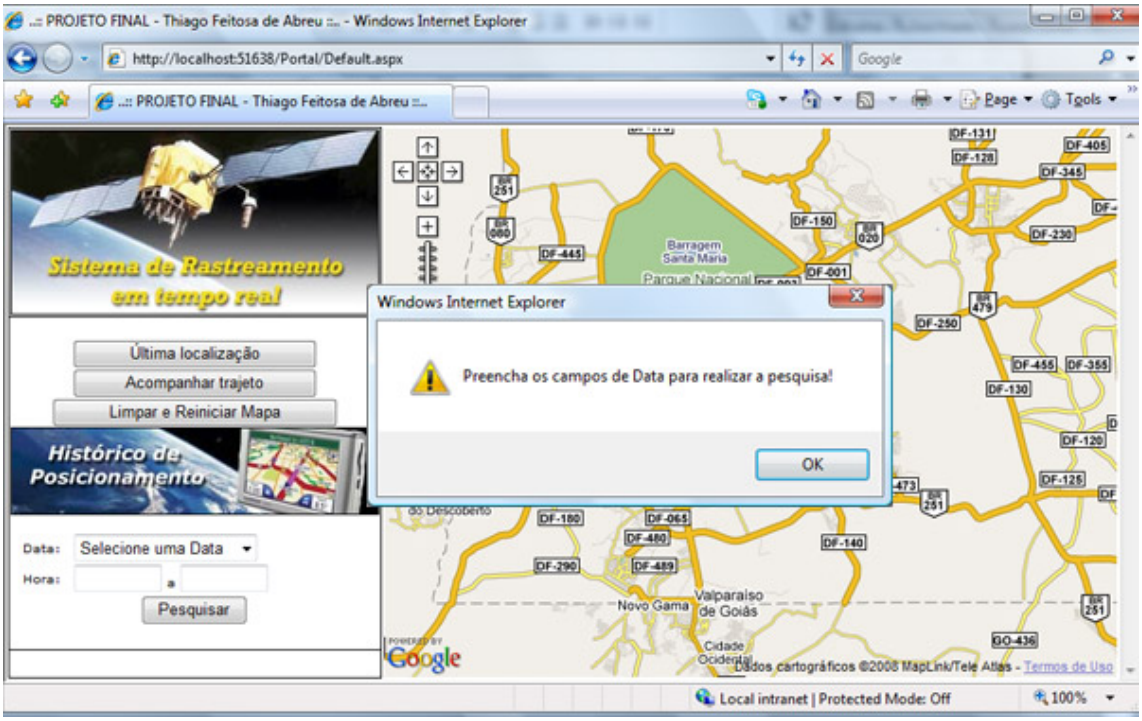


Figura 3.39 – Dados obrigatórios na pesquisa de Histórico

3.5. Google Maps

A funcionalidade do Google Maps foi essencial para a demonstração de todos os resultados obtidos no projeto, pois nos permite gratuitamente, apenas para fins acadêmicos e não comerciais, sua utilização.

Além disso, possibilita localizar alguma coordenada geográfica, distância entre dois endereços, duas cidades ou estados, melhor rota entre dois pontos, entre outras possibilidades.

3.5.1. API Google

Application Programming Interface (API) é o pacote de classes, métodos, funções e propriedades que o Google disponibiliza gratuitamente para o uso nos seus Mapas.

Através desta API é possível manipular no Mapa uma rota, uma marca em um determinado endereço, uma coordenada geográfica entre outras ações. A API é uma poderosa ferramenta para ser utilizada em páginas Web, pois agrega conteúdo informativo de uma forma muito simples nos Mapas, permitindo assim, o desenvolvimento de aplicações robustas e interativas.

Para utilizar essa API, são necessários dois pré-requisitos básicos. O primeiro é que se obtenha uma chave para ser utilizada nessa API, conforme a Figura 3.40 ilustra.

Figura 3.40 – Chave (Key) para uso do Google Maps

O segundo pré-requisito é o conhecimento necessário na linguagem Javascript, que nada mais é do que uma linguagem de scripts que é executada do lado do cliente, ou seja, no seu próprio navegador, sem precisar ir até o servidor do WebSite, efetuar algum tipo de validação.

Com o domínio desta linguagem de programação, o desenvolvimento de aplicativos utilizando a API do Google Maps se torna muito mais fácil, pois, a API é totalmente baseada em Javascript, linguagem versátil e robusta. .

3.6. Problemas encontrados

Descrevem-se abaixo os problemas encontrados e a solução para resolvê-los durante o projeto.

3.6.1. Interface do Celular

Durante a implementação, o maior problema encontrado, que com o passar dos meses de desenvolvimento deste projeto começou a se tornar um desafio, era o de enviar os dados para Internet. Sempre que a aplicação começava a rodar no celular e tentava enviar os dados, uma exceção era gerada e não conseguia progredir.

O problema era com o Sistema Operacional do celular utilizado neste projeto. O Sistema Operacional não permitia a execução de algumas classes do Framework do Java (J2ME) devido a aplicação desenvolvida não ter sido assinada digitalmente.

O custo de uma validação em sites como VeriSign por exemplo, é alto, tendo em vista que a utilização seria somente para fins acadêmicos. Contudo, diante de tal dificuldade fez-se necessário encontrar alternativa para contornar esse problema, seguindo o que estava previsto na proposta de projeto final.

3.6.1.1. Solução do Problema

Foi descoberta uma forma de contornar a situação usando a mesma classe do Framework J2ME que havia sido utilizado anteriormente, porém, mudando a passagem de parâmetros. Com essa mudança no código da Interface Java, foi possível conectar o celular à Internet para efetuar o envio das informações referentes às coordenadas geográficas.

Como as informações são buscada no GPS e mostradas na tela do celular, com um intervalo de tempo inferior a 1(um) segundo, da mesma

maneira, as informações eram enviadas para o servidor de Banco de Dados. Deveria achar uma maneira de determinar que o envio fosse realizado a cada 20 (vinte) segundos.

Muitas foram as tentativas utilizando as classes TimerTask, Timer, Thread e Date no framework J2ME, porém, com nenhuma delas obteve-se muito sucesso, por ter que implementar novas classes que herdavam métodos e atributos dessas classes nativas do Framework. E para que funcionasse, foi necessário efetuar a chamada do método, diretamente dessas classes, que por uma questão de segurança da arquitetura do Java, não era possível fazer uma classe acessar o método de um formulário do tipo MidLet que estende a classe *CommandListener*.

Desta forma, a solução encontrada para contornar este problema foi criar duas instâncias de Data e Hora, que no caso do J2ME chama-se *Calendar*, para efetuar a comparação. Quando a diferença de segundos entre a última hora registrada e a hora atual fosse maior ou igual a 20 (vinte) segundos, o método responsável pelo envio das informações era executado.

3.6.2. GPS

O GPS é um dispositivo que não pode sofrer nenhum tipo de intervenção no seu sinal, como por exemplo, submeter o uso de um GPS dentro de um local fechado onde haja telhado, teto de concreto ou lugares subterrâneos que impeçam o bom funcionamento do GPS na recepção de suas coordenadas geográficas pelos satélites.

Diante disso, qualquer que fosse a edificação ou local coberto que passasse com o GPS em funcionamento, havia a quebra da recepção do sinal. Com isso, a obtenção das coordenadas ficava comprometida e deixavam de ser vistas na tela do celular, que apenas mostrava uma coordenada geográfica fixa mesmo que estivéssemos em movimento.

Acredita-se que este fato possa ser justificado pelo fato de tratar-se de um módulo GPS portátil.

3.6.2.1. Solução sugerida do problema

A solução que poderia resolver esta questão de perda de sinal e acabar evitando a não obtenção dos dados pelo celular, seria acoplar uma antena externa ao GPS que possui esta possibilidade, possibilitando o aumento do sinal na recepção e minimizando sua perda ao passar por viadutos ou perto de edificações que viessem a prejudicar o sinal recebido pelo satélite.

3.6.3. Interface WEB

Um dos problemas encontrados foi durante a integração com a API do Google Maps.

Durante a fase de testes, quando utilizada a funcionalidade *Acompanhar Trajeto*, a inteligência presente na API do Google Maps criava um marcador no mapa e nomeando-o com uma letra do alfabeto. Porém, a partir do momento em que as letras do alfabeto eram todas utilizadas, a Interface parava de funcionar. Ou seja, quando o número de registros obtidos, a partir do momento que se iniciou o rastreamento em tempo real, ultrapassava o total de letras do alfabeto, não era possível montar a rota no Mapa, pois o Google Maps não interpretava corretamente aquela quantidade de registros.

Além disso, um segundo problema enfrentado foi durante o rastreamento em tempo real, também, em que o próprio Google Maps traçava uma rota fora dos marcadores inseridos no mapa, tornando o trajeto completamente confuso.

3.6.3.1. Solução do Problema

A fim de evitar que o erro do número de letras do alfabeto acontecesse e o usuário deixasse de visualizar a rota que estava sendo percorrida, foi decidido iniciar novamente a contagem dos registros. Sendo assim, quando

faltasse uma letra para completar o alfabeto, o desenho da rota era iniciado a partir do último ponto que havia sido obtido.

Com esta solução foi possível evitar que o Mapa deixasse de ser visualizado para o usuário que estivesse utilizando os serviços da Interface Web, entretanto, deixava-se de traçar a rota completa do percurso que estava sendo acompanhado, por ter ficado limitado ao número de letras do alfabeto.

Já para o problema da rota sendo traçada de forma confusa, fora dos marcadores no mapa, a solução foi buscar os registros do Banco de Dados ordenados de forma crescente, ordenados pela hora em que foi inserida. Acontecia exatamente o inverso, e por isso, o Google Maps acabava traçando a rota incorretamente.

CAPÍTULO 4 – CONCLUSÕES

Este projeto teve como finalidade o desenvolvimento de uma Interface (Aplicativo/Software) para um celular (utilizando linguagem de programação Java) e através desta Interface, estabelecer a conexão entre um módulo GPS portátil e um celular. Ambos possuindo tecnologia Bluetooth para se comunicarem. Com essa comunicação estabelecida, o celular seria o dispositivo responsável pelo envio das coordenadas geográficas para um servidor de banco de dados disponível na Internet.

Paralelamente ao desenvolvimento desta Interface para o celular, desenvolveu-se a interface Web, que seria responsável pelo armazenamento dos dados que o celular enviasse. Portanto, foi desenvolvida uma solução completa de envio, recepção e visualização das coordenadas geográficas. Para esta última ação foi usado o Google Maps em nossa Interface Web.

De acordo com o estudo e os testes realizados neste projeto, constatou-se que se pode fornecer uma solução de rastreamento em tempo real de baixo custo, utilizando redes e transmissões já existentes, que tornam a solução deste projeto, uma solução de alta disponibilidade. Demonstra-se que com sua utilização, a possibilidade de se rastrear um objeto (Carro, aeronave, moto, etc.) em tempo real, é completamente viável. Depende-se única e exclusivamente, da aquisição de um módulo GPS, como o utilizado neste projeto e a disponibilidade de um pacote de Dados no seu plano de telefonia móvel junto à operadora responsável pelo seu número de celular.

Nenhum tipo de problema ou indisponibilidade dos serviços de telefonia móvel durante toda a fase de implementação, desenvolvimento e testes foram detectados.

Também não se detectou nenhuma falha ou indisponibilidade nos serviços de hospedagem da Interface Web, que guardam os dados de localização configurados para serem enviados a cada 20 (vinte) segundos e possibilitam a visualização dos mesmos em forma de rota e marcadores no Google Maps.

Podemos assim, concluir que o objetivo deste projeto foi atingido com sucesso.

Como proposta de trabalhos futuros, sugere-se a melhoria da interface Web para que seja possível calcular velocidade do objeto sendo rastreado e distância do percurso realizado.

Além disso, seria interessante possibilitar uma forma de comunicação, por exemplo, via SMS, da interface Web para o celular e vice-versa.

REFERÊNCIAS BIBLIOGRÁFICAS

[1]. DEITEL H. M. 2003 – DEITEL, C# Como programar. 1. Ed. Pearson Education, 2003.

[2]. COMER Douglas E. 2006 – COMER, Interligação em Rede com TCP/IP. 5. Ed. Campus, 2006.

[3]. http://aprendendofisica.pro.br/alunos/index.php/cp202/2007/05/16/principio_de_funcionamento_do_gps acessado no dia 06/08/2008.

[4]. http://code.google.com/apis/maps/documentation/examples/index.html acessado no dia 1º/11/2008.

[5]. http://code.google.com/apis/maps/documentation/controls.html acessado em 02/09/2008.

[6]. http://conversationswithmyself.com/maps/tracker/gmapTracker.html acessado em 18/08/2008.

[7]. https://developer.sonyericsson.com/site/global/techsupport/tipstrickscode/java/p_standby_midlet_jp7phones.jsp acessado em 21/09/2008.

[8]. https://developer.sonyericsson.com/site/global/techsupport/tipstrickscode/java/p_java.jsp acessado em 25/09/2008.

[9]. http://developers.sun.com/mobility/midp/articles/network/ acessado em 07/11/2008.

[10]. http://econym.googlepages.com/index.htm acessado no dia 30/08/2008.

[11]. http://j2me.datamazon.com/index.php?option=com_content&task=category&sectionid=6&id=22&Itemid=42 acessado em 20/10/2008.

[12]. http://java.sun.com/products/javacomm/ acessado em 23/09/2008.

[13]. http://java.sun.com/products/sjwtoolkit/ acessado em 03/08/2008.

[14]. http://meiobit.pop.com.br/meio-bit/miscelaneas/a-historia-da-internet acessado em 26/10/2008.

[15]. http://pt.wikipedia.org/wiki/Comuta%C3%A7%C3%A3o_de_pacotes acessado em 27/07/2008.

[16]. http://pt.wikipedia.org/wiki/GPS acessado em 18/10/2008.

[17]. http://www.forum.nokia.com/devices/N73 acessado no dia 15/10/2008.

[18]. http://www.guj.com.br/posts/list/40085.java acessado em 22/10/2008.

[19]. http://www.guj.com.br/posts/list/54570.java#286521 acessado dia 15/08/2008.

[20]. http://www.gpsglobal.com.br acessado no dia 08/10/2008.

[21]. http://www.gpsglobal.com.br/Artigos/ITA12a.html acessado no dia 06/10/2008.

[22]. http://www.infowester.com/bluetooth.php acessado em 02/08/2008.

[23]. http://www.isoc.org/internet/history/brief.shtml acessado em 15/10/2008.

[24]. http://www.oficinadanet.com.br/artigo/676/como_funcionam_os_telefones_celulares acessado em 10/10/2008.

[25]. http://www.pocketgpsworld.com/globalsat-bt-359-gps-review.php acessado em 15/08/2008.

[26]. MRIDULA PARIHAR & PAUL LASALLE & ROB CRIMGER & ET AL – TCP/IP a Bíblia. 1. Ed. Campus, 2002.

[27]. Site da ANATEL - Sistema Geodésico, Portaria 006 de 20 de janeiro de 2003, http://www.anatel.gov.br/Tools/frame.asp?link=/biblioteca/portaria/2003/portaria_006_2003.pdf acessado no dia 10/10/2008.

[28]. TANENBAUM 2003 - TANENBAUM, A. S. *Redes de Computadores*. 4. ed. Rio de Janeiro, Brasil: Ed. Campus, 2003.

[29]. VALENTINO LEE & HEATHER SCHNEIDER & ROBBIE SCHELL 2005 – Aplicações Móveis: Arquitetura, Projeto e Desenvolvimento. 1. Ed. Makron Books.

ANEXO I

Transcreve-se abaixo as classes em Java utilizadas para o desenvolvimento do programa que foi instalado no celular, que são de autoria de Sérgio Estevão.

A classe *Location.java* é responsável por atribuir os valores corretos a todos os atributos públicos referente aos dados que são recebidos do GPS.

Location.java

```
/**
 * Sérgio Estevão
 * MIDP Adventures
 */
public class Location {
  // NMEA Elements
  String utc;
  String latitude;
  String northHemi;
  String longitude;
  String eastHemi;
  String altitude;
  int quality;
  int nSat;
  String horDilution;
  String altitudeUnit;
  String geoidalHeight;
  String geoidalHeightUnit;
```

```
  String diffCorrection;
  String diffStationId;

  /**
   * Method that parses a NMEA string and returns Location. For more
info check
   * this page: http://www.gpsinformation.org/dale/nmea.htm#GGA
   *
   * @param value -
   *          string that represent NMEA GGA string
   */
  public void parseGPGGA(String value) {
    // Helper class to parse strings
    StringTokenizer tok = new StringTokenizer(value, ",");
    utc = tok.nextToken();
    latitude = tok.nextToken();
    northHemi = tok.nextToken();
    longitude = tok.nextToken();
    eastHemi = tok.nextToken();
    quality = Integer.parseInt(tok.nextToken());
    nSat = Integer.parseInt(tok.nextToken());
    horDilution = tok.nextToken();
    altitude = tok.nextToken();
    altitudeUnit = tok.nextToken();
    geoidalHeight = tok.nextToken();
    geoidalHeightUnit = tok.nextToken();
```

```
    diffCorrection = tok.nextToken();
    diffStationId = tok.nextToken();
  }
}
```

A classe *GPSBt.java* é utilizada somente quando o usuário seleciona o GPS, depois de ter realizado a pesquisa de dispositivos. Com isso, é iniciada a leitura dos dados.

GPSBt.java

```
/**
 * Sérgio Estevão
 * MIDP Adventures
 */

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
public class GpsBt implements Runnable {
  public int mode = 0;
  public boolean isActive = false;
  public boolean isConnected = false;
  // current bluetooth device
  public String btUrl = "";
  public String btName = "";
  public String error = "No Error";
```

```
// current connection
StreamConnection conn = null;
DataInputStream in = null;
DataOutputStream out = null;
StringBuffer sb;
String currentInfo;
// state
public final static byte STATE_SEARCH_SENTENCE_BEGIN = 0;
public final static byte STATE_READ_DATA_TYPE = 1;
public final static byte STATE_READ_SENTENCE = 2;
static GpsBt gpsReader = null;
static GpsBt instance() {
  if (gpsReader == null) {
    gpsReader = new GpsBt();
  }
  return gpsReader;
}
/** Creates a new instance of GPSReader */
public GpsBt() {
  currentInfo = null;
  isConnected = false;
  isActive = false;
  sb = new StringBuffer(100);
}
/**
 * Sets the bluetooth gps device to connect and read information
```

```
 *
 * @param btUrl -
 *          bluetooth url address
 * @param btName -
 *          bluetooth friendly name
 */
public void setDevice(String btUrl, String btName) {
  this.btUrl = btUrl;
  this.btName = btName;
}
public boolean isConnected() {
  return isConnected;
}
/**
 * Start connection to device, and main thread
 */
public void start() {
  if (isActive) {
    stop();
  }
  connect();
  if (isConnected) {
    isActive = true;
    Thread t = new Thread(this);
    t.start();
  }
```

```
}

/**
 * Stop connection to the device and ends thread
 */
public void stop() {
  if (isActive) {
    isActive = false;
    try {
      while (isConnected) {
        close();
        Thread.sleep(100);
      }
    } catch (Throwable t) {
      error = "stop:" + t.toString();
      close();
    }
  }
}
public void run() {
  isActive = true;
  while (isActive) {
    try {
      // check if connection is still open
      if (!isConnected && isActive) {
        // connect to gps device
        connect();
```

```
      } else {
        // read NMEA Strings
        readNMEASentences();
      }
    } catch (Throwable t) {
      error = "run:" + t.toString();
      close();
    }
  }
  close();
  isActive = false;
}
/**
 * Read characteres from device until finds a GPGAA NMEA sentence
 */
public void readNMEASentences() {
  try {
    if (!isConnected) {
      return;
    }
    int size = in.available();
    if (size <= 0) {
      return;
    }
    for (int j = 0; j < size; j++) {
      int i = in.read();
```

```
if (i != -1) {
  char l = (char) i;
  switch (mode) {
  case (STATE_SEARCH_SENTENCE_BEGIN): {
    // search for the sentence begin
    if (l == '$') {
      // found begin of sentence
      mode = 1;
      sb.setLength(0);
    }
  }
    break;
  case (STATE_READ_DATA_TYPE): {
    // check what kind of sentence we have
    sb.append(l);
    if (sb.length() == 6) {
      if (sb.toString().startsWith("GPGGA")) {
        mode = STATE_READ_SENTENCE;
        sb.setLength(0);
      } else {
        mode = STATE_SEARCH_SENTENCE_BEGIN;
        sb.setLength(0);
      }
    }
  }
    break;
```

```
        case (STATE_READ_SENTENCE): {
          // read data from sentence
          sb.append(l);
          if ((l == 13) || (l == 10) || (l == '$')) {
            mode = STATE_SEARCH_SENTENCE_BEGIN;
            synchronized (this) {
              currentInfo = new String(sb.toString());
            }
            Thread.sleep(1000);
          }
        }
          break;
        }

      } else {
        close();
      }
    }

  } catch (Exception e) {
    error = "fetch" + e.toString();
    close();
  }
}
/**
 * Parses data from current sentence and return a Location.
```

```
   *
   * @return
   */
  public Location getLocation() {
    Location location = new Location();
    try {
      if (isConnected && isActive && currentInfo != null) {
          location.parseGPGGA(currentInfo);
      }
    } catch (Throwable t) {
      error = "get:" + t.toString();
      close();
    }
    return location;
  }
  /**
   * Connect to the bluetooth device
   */
  public void connect() {
    if (btUrl == null || (btUrl.trim().compareTo("") == 0)) {
      isConnected = false;
      return;
    }
    try {
      conn = (StreamConnection) Connector.open(btUrl,
Connector.READ_WRITE);
```

```
    in = new DataInputStream(conn.openInputStream());
    isConnected = true;
    mode = 0;
  } catch (IOException e) {
    close();
  }
}
public void close() {
  try {
    if (in != null) {
      in.close();
    }
    if (out != null) {
      out.close();
    }
    if (conn != null) {
      conn.close();
    }
    in = null;
    out = null;
    conn = null;
  } catch (Throwable t) {
    error = "close" + t.toString();
  } finally {
    in = null;
    out = null;
```

```
      conn = null;
    }
    isConnected = false;
  }
}
```

Classe *BTManager.java* faz o gerenciamento dos dispositivos e também é responsável por localizar e conectá-los quando solicitado pelo usuário.

BTManager.java

```
/**
 * Sérgio Estevão
 * MIDP Adventures
 */

import java.util.Vector;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.UUID;

public class BTManager implements DiscoveryListener {
  public static final int BLUETOOTH_TIMEOUT = 30000;
  public Vector btDevicesFound;
```

```
public Vector btServicesFound;
private boolean isBTSearchComplete;
static BTManager btManager = null;
static BTManager instance() {
  if (btManager == null) {
    btManager = new BTManager();
  }
  return btManager;
}
/** Creates a new instance of BTManager */
private BTManager() {
  btDevicesFound = new Vector();
  btServicesFound = new Vector();
}
public static UUID[] getRFCOMM_UUID() {
  UUID[] uuidSet;
  UUID RFCOMM_UUID = new UUID(0x1101); // RFCOMM service
  uuidSet = new UUID[1];
  uuidSet[0] = RFCOMM_UUID;
  return uuidSet;
}
/**
 * Finds bluetooth devices
 *
 * @param aServices,
```

```
   *           an array with the service UUID identifiers you want to
search
   * @returns the number of devices found
   */
  public int find(UUID[] aServices) {
    findDevices();
    findServices(aServices);
    return btDevicesFound.size();
  }
  public int findDevices() {
    try {
      // cleans previous elements
      btDevicesFound.removeAllElements();
      isBTSearchComplete = false;
      LocalDevice local = LocalDevice.getLocalDevice();
      DiscoveryAgent discoveryAgent = local.getDiscoveryAgent();
      // discover new devices
      discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
      while ((!isBTSearchComplete)) {
        // waits for a fixed time, to avoid long search
        synchronized (this) {
          this.wait(BTManager.BLUETOOTH_TIMEOUT);
        }
        // check if search is completed
        if (!isBTSearchComplete) {
          // search no yet completed so let's cancel it
```

```
        discoveryAgent.cancelInquiry(this);
      }
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
  return btDevicesFound.size();
}
public void deviceDiscovered(RemoteDevice remoteDevice,
    DeviceClass deviceClass) {
  btDevicesFound.addElement(remoteDevice);
}
public void inquiryCompleted(int param) {
  isBTSearchComplete = true;
  // notifies and wake main thread that device search is completed
  synchronized (this) {
    this.notify();
  }
}
public void findServices(UUID[] aServices) {
  // cleans previous elements
  btServicesFound.removeAllElements();
  try {
    LocalDevice local = LocalDevice.getLocalDevice();
    DiscoveryAgent discoveryAgent = local.getDiscoveryAgent();
    // discover services
```

```
      if (btDevicesFound.size() > 0) {
        for (int i = 0; i < btDevicesFound.size(); i++) {
          isBTSearchComplete = false;
          // adds a null element in case we don't found service
          btServicesFound.addElement(null);
          int transID = discoveryAgent.searchServices(null, aServices,
              (RemoteDevice) (btDevicesFound.elementAt(i)), this);
          // wait for service discovery ends
          synchronized (this) {
            this.wait(BTManager.BLUETOOTH_TIMEOUT);
          }
          if (!isBTSearchComplete) {
            discoveryAgent.cancelServiceSearch(transID);
          }
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void servicesDiscovered(int param, ServiceRecord[]
serviceRecord) {
    int index = btServicesFound.size() - 1;
    for (int i = 0; i < serviceRecord.length; i++) {
      btServicesFound.setElementAt(serviceRecord[i], index);
    }
```

```
}
public void serviceSearchCompleted(int transID, int respCode) {
  isBTSearchComplete = true;
  // notifies and wake mains thread that service search is completed
  synchronized (this) {
    this.notify();
  }
}
/**
 * Get a human readable name of the BT device.
 *
 * @param deviceID
 * @return the friendly name for the BT device
 */
public String getDeviceName(int deviceID) {
  try {
    return ((RemoteDevice) btDevicesFound.elementAt(deviceID))
        .getFriendlyName(false);
  } catch (Exception e) {
    e.printStackTrace();
  }
  return "Error";
}
/**
 * Gets the URL address of the the service you want to connect
 *
```

```
   * @param deviceID
   * @return the Url address for the device and service found
   */
  public String getServiceURL(int deviceID) {
    try {
      return ((ServiceRecord) btServicesFound.elementAt(deviceID))
          .getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT,
false);
    } catch (Exception e) {
      e.printStackTrace();
    }
    return "Error";
  }
}
```

APÊNDICE I

A classe *BlueToothGPSMidlet.java* é a classe principal do aplicativo JAVA e é responsável por comunicar-se com todas as outras classes existentes na aplicação e descritas no Anexo deste projeto. Além disso, é a classe que permite a visualização das informações na tela do celular.

BlueToothGPSMidlet.java

```
/**
 * Sérgio Estevão
 * MIDP Adventures
 */
import javax.microedition.io.*;
import java.io.*;
import java.util.*;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.lcdui.Ticker;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
```

```
public class BluetoothGPSMidlet extends MIDlet implements
CommandListener,
        Runnable {

    private Command cmdBack;
    private Form mainForm;
    private Command cmdSearchGps;
    private Form gpsForm;
    private Command cmdSearch;
    private Command cmdSelect;
    private Command cmdExit;
    private ChoiceGroup choiceGps;
    private StringItem gpsState;
    private StringItem quality;
    private StringItem latitude;
    private StringItem longitude;
    private StringItem time;
    private StringItem satellite;
    private static final int STATE_IDLE = 0;
    private static final int STATE_SEARCH = 1;
    private static final int STATE_READING = 2;
    private boolean active = false;
    private int state = STATE_IDLE;
    private Thread thread;
    /* Alteracoes Thiago */
```

```
    private HttpConnection httpCon;

    private InputStream input;

    private StringItem texto;

    private Location location;

    private int contador = 0;

    private int contadorAtual = 0;

    private java.util.Calendar horaInicial = java.util.Calendar.getInstance();


    protected void destroyApp(boolean arg0) throws MIDletStateChangeException {

        exit();

    }


    protected void pauseApp() {

    }


    protected void startApp() throws MIDletStateChangeException {

        display(initMainForm());

    }


    public void commandAction(Command cmd, Displayable display) {

        if (display == mainForm) {

            if (cmd == cmdSearchGps) {

                display(initGPSForm());

            } else if (cmd == cmdExit) {

                exit();
```

```
            }
        } else if (display == gpsForm) {
            if (cmd == cmdSearch) {
                doAction(STATE_SEARCH);
            }
            if (cmd == cmdSelect) {
                int option = choiceGps.getSelectedIndex();
                // any device selected?
                if (option != -1) {
                    // set gps reader to selected device
                    GpsBt.instance().setDevice(
BTManager.instance().getServiceURL(option),
BTManager.instance().getDeviceName(option));
                    doAction(STATE_READING);
                    // start reading value;
                    GpsBt.instance().start();
                    display(initMainForm());
                }
            } else if (cmd == cmdBack) {
                display(initMainForm());
            }
        }
    }

    public void exit() {
```

```
    notifyDestroyed();
}

public void display(Displayable display) {
    // shows the display.
    Display.getDisplay(this).setCurrent(display);
}

public void display(Alert alert, Displayable next) {
    // shows the alert screen.
    Display.getDisplay(this).setCurrent(alert, next);
}

public Command initBackCommand() {
    if (cmdBack == null) {
        cmdBack = new Command("Voltar", Command.BACK, 1);
    }
    return cmdBack;
}

public Displayable initMainForm() {
    if (mainForm == null) {
        mainForm = new Form("Inform. GPS");
        mainForm.setCommandListener(this);
        cmdSearchGps = new Command("Pesquisar", Command.ITEM, 1);
        cmdExit = new Command("Sair", Command.EXIT, 1);
```

```
        mainForm.addCommand(cmdSearchGps);
        mainForm.addCommand(cmdExit);

        gpsState = new StringItem("Status", "-");
        quality = new StringItem("Qualidade", "-");
        latitude = new StringItem("Latitude", "-");
        longitude = new StringItem("Longitude", "-");
        time = new StringItem("Hora UTC", "-");
        satellite = new StringItem("Satelite", "-");
        texto = new StringItem("Retorno", "-");

        mainForm.append(gpsState);
        mainForm.append(quality);
        mainForm.append(latitude);
        mainForm.append(longitude);
        mainForm.append(time);
        mainForm.append(satellite);
        mainForm.append(texto);

    }
    return mainForm;
}

public Displayable initGPSForm() {
    if (gpsForm == null) {
        gpsForm = new Form("Pesquisa de GPS");
```

```
            gpsForm.setCommandListener(this);
            cmdSearch = new Command("Pesquisa", Command.ITEM, 1);
            gpsForm.addCommand(cmdSearch);
            cmdSelect = new Command("Seleciona", Command.ITEM, 1);
            gpsForm.addCommand(cmdSelect);
            gpsForm.addCommand(initBackCommand());

            choiceGps = new ChoiceGroup("GPS:",
ChoiceGroup.EXCLUSIVE);
            gpsForm.append(choiceGps);
        }
        return gpsForm;
    }

    public void doAction(int action) {
        if (thread == null) {
            thread = new Thread(this);

            thread.start();
        }
        state = action;
    }

    public void run() {

        active = true;
```

```
        while (active) {
            switch (state) {
                case (STATE_IDLE):
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    break;
                case (STATE_SEARCH):
                    choiceGps.deleteAll();
                    gpsForm.setTicker(new Ticker("Procurando
dispositivos..."));

BTManager.instance().find(BTManager.getRFCOMM_UUID());
                    int size =
BTManager.instance().btDevicesFound.size();
                    for (int i = 0; i < size; i++) {

choiceGps.append(BTManager.instance().getDeviceName(i), null);
                    }
                    gpsForm.setTicker(null);
                    if (size == 0) {
                        display(new Alert("Nenhum dispositivo
encontrado"));
                    }
                    doAction(STATE_IDLE);
                    break;
```

```
                case (STATE_READING):
                    GpsBt gpsBt = GpsBt.instance();
                    if (gpsBt.isConnected()) {
                        gpsState.setText("Conectado");
                        location = gpsBt.getLocation();
                        quality.setText(location.quality + "");
                        latitude.setText(location.latitude            +
location.northHemi);
                        longitude.setText(location.longitude          +
location.eastHemi);
                        time.setText(location.utc);
                        satellite.setText(location.nSat + "");
                        java.util.Calendar          horaAtual         =
Calendar.getInstance();

                        if      ((horaAtual.get(Calendar.SECOND)      -
horaInicial.get(Calendar.SECOND)) == 20 ||
                                (horaAtual.get(Calendar.SECOND)       -
horaInicial.get(Calendar.SECOND)) == -20 ||
                                (horaAtual.get(Calendar.SECOND)       -
horaInicial.get(Calendar.SECOND)) == 40 ||
                                (horaAtual.get(Calendar.SECOND)       -
horaInicial.get(Calendar.SECOND)) == -40) {
                            horaInicial = horaAtual;
                            EnviaInformacoes();
                        }
                    } else {
                        gpsState.setText("Desconectado");
                    }
```

```
                    break;
                }
            }
        }

    public void EnviaInformacoes() {
        try {
            String url = "http://www.infodev.com.br/ProjetoFinal/" +
                    "RecebeDados.aspx?latitude=" + location.latitude +
location.northHemi +
                    "&longitude="      +      location.longitude      +
location.eastHemi;
            httpCon = (HttpConnection) Connector.open(url);
            httpCon.setRequestMethod(HttpConnection.GET);
            String message = (String) httpCon.getResponseMessage();
            //System.out.println("Message:" + message);
            //texto.setText("Conectado");
            input = httpCon.openInputStream();

            byte[] buffer = new byte[500];
            int readSize = input.read(buffer);
            String textContent = new String(buffer, 0, readSize);
            texto.setText(textContent.substring(11));
            //texto.setText(String.valueOf(contadorAtual));
            input.close();
            httpCon.close();
        } catch (Exception e) {
```

```
            //System.out.println("Exception: \n" + e.toString() +
            //"\n\n");
            texto.setText("Erro:");
        }
    }
}
```

APÊNDICE II

Código-fonte do Aplicativo desenvolvido para a Interface Web. Essa é a página responsável pelo recebimento das informações que o celular captura e envia para a internet. Possui métodos responsáveis por recuperar os dados através da URL e os converte para o formato decimal, para inserir no Banco de Dados.

RecebeDados.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using Util;


public partial class RecebeDados : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string latitude = Request.QueryString["latitude"];
        string longitude = Request.QueryString["longitude"];

        InsereCoordenadaGeografica(1,latitude, longitude);
        Response.Write("Enviado ok! " +
DateTime.Now.ToString("HH:mm:ss"));
    }

    private void InsereCoordenadaGeografica(int codDispositivo, string
latitude, string longitude)
    {
        Base metodo = new Base();
        string latitudeDecimal = defineSinal(latitude) +
ConverteNMEA_Decimal(latitude);
        string longitudeDecimal = defineSinal(longitude) +
ConverteNMEA_Decimal(longitude);

        metodo.ExecutaSQL("INSERT INTO gps_HistoricoPosicionamento
(cod_dispositivo, latitude, longitude) VALUES " +
            "(" + codDispositivo + ",'" + latitudeDecimal + "','" +
longitudeDecimal + "')");
    }

    private string defineSinal(string valor)
    {
        string retorno = "";
```

```
        string direcao = valor.Substring(valor.Length - 1, 1);
        if (direcao == "S" || direcao == "W")
        {
            retorno = "-";
        }

        return retorno;
    }

    private string ConverteNMEA_Decimal(string valor)
    {
        double calculo;
        if (valor.Substring(0, valor.IndexOf('.')).Length == 5)
        {
            int somador = int.Parse(valor.Substring(0, 3));
            string numeroComVirgula = valor.Substring(3, valor.Length
- 4).Replace('.', ',');
            float numero = float.Parse(numeroComVirgula);


            calculo = somador + (numero / 60);
        }
        else
        {
            int somador = int.Parse(valor.Substring(0, 2));
            string numeroComVirgula = valor.Substring(2, valor.Length
- 3).Replace('.', ',');
            float numero = float.Parse(numeroComVirgula);

            calculo = somador + (numero / 60);
        }

        return calculo.ToString().Substring(0, 8).Replace(',', '.');
    }


}
```

Página principal da Interface Web. Responsável por mostrar as funcionalidades existentes na aplicação e o mapa que o usuário poderá visualizar as rotas realizadas.

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>..:: PROJETO FINAL - Thiago Feitosa de Abreu ::..</title>
    <link href="css/global.css" rel="stylesheet" type="text/css" />
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAblXI
pQoOq-mK6R15mpsmbhQsT3GG7EUizTmXWxn7oiOu6mHiOhRvvZVK5tK0bqa-
PcTMVxIdinzDFw"
```

```
        type="text/javascript"></script>
</head>
<body onload="initialize()" onunload="GUnload()">
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <div>
            <table width="900px" cellpadding="0" cellspacing="2">
                <tr>
                    <td style="width: 300px" valign="top">
                        <table width="100%" height="450px"
cellpadding="1" cellspacing="1" style="background-color: Black;">
                            <tr style="background-color: White;">
                                <td align="center">
                                    <img src="img/topo_form.jpg"
border="1" /></td>
                            </tr>
                            <tr style="background-color:
White;height:270px;">
                                <td align="center">
                                    <asp:Button
ID="btnUltimaLocalizacao" Width="200px" runat="server" Text="Última
localização" OnClick="btnUltimaLocalizacao_Click" />
                                    <br />
                                    <asp:Button
ID="btnAcompanharTrajeto" Width="200px" runat="server"
Text="Acompanhar trajeto" OnClick="btnAcompanharTrajeto_Click" />
                                    <br />
                                    <asp:Button
ID="btnLimparReiniciar" runat="server" Text="  Limpar e Reiniciar Mapa
" OnClick="btnLimparReiniciar_Click" />
                                    <br />
                                    <img
src="img/topo_pesquisaHist.jpg" border="1" />
                                    <br />
                                    <table width="100%"
cellpadding="0" cellspacing="2">
                                        <tr>
                                            <td>
                                                Data:</td>
                                            <td align="left">
                                                <asp:DropDownList
ID="ddlData" runat="server" Width="150px">

</asp:DropDownList></td>
                                        </tr>
                                        <tr>
                                            <td>
                                                Hora:</td>
                                            <td align="left">
                                                <asp:TextBox
ID="txtHoraI" MaxLength="5" runat="server" Width="66px"></asp:TextBox>
                                                a
                                                <asp:TextBox
ID="txtHoraF" MaxLength="5" runat="server"
Width="66px"></asp:TextBox></td>
                                        </tr>
                                        <tr>
                                            <td align="center"
colspan="2">
```

```
                                                    <asp:Button
ID="btnPesquisar" runat="server" Text="Pesquisar"
OnClick="btnPesquisar_Click" />
                                                </td>
                                            </tr>
                                        </table>
                                    </td>
                                </tr>
                                <tr style="background-color: White;">
                                    <td align="center"
style="height:20px;">
                                        <marquee>
                                             <asp:Label
ID="lblMensagem" runat="server" Font-Bold="True"
ForeColor="Red"></asp:Label>
                                        </marquee>
                                    </td>
                                </tr>
                            </table>
                        </td>
                        <td>
                            <div id="mapc" style="width: 600px; height:
450px">
                            </div>
                        </td>
                    </tr>
                </table>
            </div>
        </form>
</body>
</html>
```

Default.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using Util;
using System.Text;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (Request.QueryString["acao"] != null)
            {
                switch (Request.QueryString["acao"].ToString())
                {
                    case "UL": // Última localização
                        UltimaLocalizacao();
```

```
                        break;
                    case "RT": // Rastreando trajeto em tempo real
                        RastreandoTrajeto();
                        break;
                }
            }
            else
            {
                CarregaMapa();
            }
        }
    }

    #region " Métodos Privados "

    private bool ValidaCampos()
    {
        if (ddlData.SelectedValue != "")
        {
            try
            {
                DateTime dataInicial =
Convert.ToDateTime(ddlData.SelectedValue);
                DateTime dataFinal =
Convert.ToDateTime(ddlData.SelectedValue);
                return true;
            }
            catch
            {
                CarregaMapa();
                ClientScript.RegisterClientScriptBlock(this.GetType(),
"validacao", "alert('As Datas informadas não estão no formato
correto.');", true);
                return false;
            }
        }
        else
        {
            CarregaMapa();
            ClientScript.RegisterClientScriptBlock(this.GetType(),
"erro", "alert('Preencha os campos de Data para realizar a
pesquisa!');", true);
            return false;
        }
    }

    private void CarregaDatasDisponiveis()
    {
        Base metodo = new Base();
        DataTable dt = metodo.GerarDataTable("SELECT DISTINCT
CONVERT(varchar,data_hora,103) as data FROM
gps_HistoricoPosicionamento");
        ddlData.DataSource = dt.DefaultView;
        ddlData.DataTextField = "data";
        ddlData.DataValueField = "data";
        ddlData.DataBind();
        ddlData.Items.Insert(0, new ListItem("Selecione uma Data",
""));
    }

    private void CarregaMapa()
```

```
    {
        CarregaDatasDisponiveis();

        StringBuilder funcao = new StringBuilder();
        funcao.Append("function initialize() { ");
        funcao.Append("var map = new
GMap2(document.getElementById(\"mapc\")); ");
        funcao.Append("map.setCenter(new GLatLng(-15.83434,-47.92142),
10); ");
        funcao.Append("map.addControl(new GLargeMapControl()); ");
        funcao.Append("map.setMapType(G_NORMAL_MAP); ");
        funcao.Append(" }");
        ClientScript.RegisterStartupScript(this.GetType(), "Rotas",
funcao.ToString(), true);
    }

    private void UltimaLocalizacao()
    {
        lblMensagem.Text = "Rastreando última localização, atualização
a cada 30 segundos!";
        Base metodo = new Base();

        DataTable dt = metodo.GerarDataTable("SELECT TOP 1 latitude,
longitude FROM gps_HistoricoPosicionamento " +
            "ORDER BY data_hora DESC");

        string primeiraCoordenada = dt.Rows[0]["latitude"].ToString()
+ "," + dt.Rows[0]["longitude"].ToString();

        StringBuilder funcao = new StringBuilder();
        funcao.Append("function initialize() { ");
        funcao.Append("var map = new
GMap2(document.getElementById(\"mapc\")); ");
        funcao.Append("map.setCenter(new GLatLng(" +
primeiraCoordenada + "), 10); ");
        funcao.Append("map.addControl(new GLargeMapControl()); ");
        funcao.Append("map.setMapType(G_NORMAL_MAP); ");
        funcao.Append("map.addOverlay(new GMarker(new GLatLng(" +
primeiraCoordenada + "))); ");
        funcao.Append(" }");
        ClientScript.RegisterStartupScript(this.GetType(), "Rotas",
funcao.ToString(), true);
        Response.AppendHeader("Refresh", "30;
URL=http://www.infodev.com.br/ProjetoFinal/Default.aspx?acao=UL");
        //Response.AppendHeader("Refresh", "30;
URL=http://localhost:51638/Portal/Default.aspx?acao=UL");
    }

    private void RastreandoTrajeto()
    {
        if (Session["contador"] == null)
            Session["contador"] = 1;
        else
            Session["contador"] = Convert.ToInt32(Session["contador"])
+ 1;

        lblMensagem.Text = "Rastreando e acompanhando trajeto,
atualização e montagem da rota a cada 30 segundos!";
        Base metodo = new Base();
        StringBuilder funcao = new StringBuilder();
        string sql = String.Empty;
```

```
        if (Convert.ToInt32(Session["contador"].ToString()) > 24)
        {
            Session["contador"] = 1;
        }

        sql = "SELECT TOP " + Session["contador"].ToString() + "
latitude, longitude, data_hora " +
              "FROM gps_HistoricoPosicionamento ORDER BY data_hora
DESC";

        DataTable dt = metodo.GerarDataTable(sql);

        string primeiraCoordenada = dt.Rows[0]["latitude"].ToString()
+ "," + dt.Rows[0]["longitude"].ToString();
        int totalLinhas = dt.Rows.Count == 1 ? 1 : dt.Rows.Count - 1;
        int contador = 0;

        if (Convert.ToInt32(Session["contador"].ToString()) == 1)
        {
            funcao.Append("function initialize() { ");
            funcao.Append("var map = new
GMap2(document.getElementById(\"mapc\")); ");
            funcao.Append("map.setCenter(new GLatLng(" +
primeiraCoordenada + "), 10); ");
            funcao.Append("map.addControl(new GLargeMapControl()); ");
            funcao.Append("map.setMapType(G_NORMAL_MAP); ");
            funcao.Append("map.addOverlay(new GMarker(new GLatLng(" +
primeiraCoordenada + "))); ");
            funcao.Append(" }");
        }
        else
        {
            funcao.Append("function initialize() { ");
            funcao.Append("var map = new
GMap2(document.getElementById(\"mapc\")); ");
            funcao.Append("map.addControl(new GLargeMapControl()); ");
            funcao.Append("map.setMapType(G_NORMAL_MAP); ");
            funcao.Append("var directions = new GDirections(map,  ");

//funcao.Append("document.getElementById(\"informacoes\"));  ");
            funcao.Append("document.getElementById(null));  ");

            funcao.Append("var arrLocation = new Array(" +
totalLinhas.ToString() + "); ");

            dt.DefaultView.Sort = "data_hora ASC";
            foreach (DataRowView drv in dt.DefaultView)
            {
                funcao.Append("arrLocation[" + contador.ToString() +
"] = new GLatLng(" + drv[0].ToString() + "," + drv[1].ToString() + ");
");
                contador++;
            }

            funcao.Append("directions.clear(); ");
            funcao.Append("directions.loadFromWaypoints(arrLocation);
");
            funcao.Append(" }");
        }
```

```
        ClientScript.RegisterStartupScript(this.GetType(), "Rotas",
funcao.ToString(), true);
        Response.AppendHeader("Refresh", "30;
URL=http://www.infodev.com.br/ProjetoFinal/Default.aspx?acao=RT");
        //Response.AppendHeader("Refresh", "30;
URL=http://localhost:51638/Portal/Default.aspx?acao=RT");
    }

    private void PesquisaHistorico()
    {
        Base metodo = new Base();

        string dataInicio = Convert.ToDateTime(ddlData.SelectedValue +
" " + (txtHoraI.Text == "" ? "00:00" : txtHoraI.Text)).ToString("yyyy-
MM-dd HH:mm:ss");
        string dataFinal = Convert.ToDateTime(ddlData.SelectedValue +
" " + (txtHoraF.Text == "" ? "23:59" : txtHoraF.Text)).ToString("yyyy-
MM-dd HH:mm:ss");
        DataTable dt = metodo.GerarDataTable("SELECT latitude,
longitude FROM gps_HistoricoPosicionamento " +
                        "where data_hora >= CONVERT(DATETIME,'" +
dataInicio + "',101) " +
                        "and data_hora <= CONVERT(DATETIME,'" +
dataFinal + "',101)");

        string primeiraCoordenada = dt.Rows[0]["latitude"].ToString()
+ "," + dt.Rows[0]["longitude"].ToString();
        int totalLinhas = dt.Rows.Count == 1 ? 1 : dt.Rows.Count - 1;
        int contador = 0;

        StringBuilder funcao = new StringBuilder();
        funcao.Append("function initialize() { ");
        funcao.Append("var map = new
GMap2(document.getElementById(\"mapc\")); ");
        //funcao.Append("map.setCenter(new GLatLng(" +
primeiraCoordenada + "), 10); ");
        funcao.Append("map.addControl(new GLargeMapControl()); ");
        funcao.Append("map.setMapType(G_NORMAL_MAP); ");
        funcao.Append("var directions = new GDirections(map,  ");
        //funcao.Append("document.getElementById(\"informacoes\"));
");
        funcao.Append("document.getElementById(null));  ");

        funcao.Append("var arrLocation = new Array(" +
totalLinhas.ToString() + "); ");
        foreach (DataRow dr in dt.Rows)
        {
            funcao.Append("arrLocation[" + contador.ToString() + "] =
new GLatLng(" + dr["latitude"].ToString() + "," +
dr["longitude"].ToString() + "); ");
            contador++;
        }

        funcao.Append("directions.clear(); ");
        funcao.Append("directions.loadFromWaypoints(arrLocation); ");
        funcao.Append(" }");

        ClientScript.RegisterStartupScript(this.GetType(), "Rotas",
funcao.ToString(), true);
    }
```

```
    private void LimpaCamposPesquisa()
    {
        ddlData.SelectedValue = "";
        txtHoraI.Text = "";
        txtHoraF.Text = "";
    }

    #endregion

    #region " Eventos "

    protected void btnUltimaLocalizacao_Click(object sender, EventArgs
e)
    {
        LimpaCamposPesquisa();
        UltimaLocalizacao();
    }

    protected void btnAcompanharTrajeto_Click(object sender, EventArgs
e)
    {
        LimpaCamposPesquisa();
        RastreandoTrajeto();
    }

    protected void btnPesquisar_Click(object sender, EventArgs e)
    {
        if (ValidaCampos())
            PesquisaHistorico();
    }

    protected void btnLimparReiniciar_Click(object sender, EventArgs
e)
    {
        LimpaCamposPesquisa();
        Session["contador"] = null;
        lblMensagem.Text = "";
        Response.Redirect("Default.aspx");
    }

    #endregion


}
```