FERNANDO TAHARA TAKAGI
RAFAEL MEDEIROS TEIXEIRA
THIAGO KOJI MASUKI

SISTEMA DE MONITORAMENTO E PREDIÇÃO DE TEMPO DE CHEGADA DE ÔNIBUS DE LINHA

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia da Computação.

São Paulo
2010

FERNANDO TAHARA TAKAGI
RAFAEL MEDEIROS TEIXEIRA
THIAGO KOJI MASUKI

SISTEMA DE MONITORAMENTO E PREDIÇÃO DE TEMPO DE CHEGADA DE ÔNIBUS DE LINHA

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a Graduação em Engenharia da Computação.

Orientador:
Prof. Dr. Carlos Cugnasca

São Paulo
2010

Texto da decidatória

AGRADECIMENTOS

Agradecemos ao Prof. Dr. Carlos Cugnasca, que nos ofereceu incentivo contínuo nesse projeto e foi de grande auxílio para conseguirmos finalizá-lo.

RESUMO

Uma problema que todas as grandes cidades do mundo têm em comum, sejam elas de qualquer continente ou índice de desenvolvimento social, é o trânsito caótico. É consenso entre os especialistas que a única solução viável para esse problema é a adoção do sistema de transporte público por uma maior parcela da população. Porém, é difícil convencer as pessoas a deixarem seus carros em casa quando a alternativa é um meio de transporte superlotado, com constantes atrasos e horários imprevisíveis.

Esse projeto de formatura visa amenizar o problema do transporte público, propondo um sistema capaz de fornecer à população uma estimativa do tempo de chegada do próximo ônibus de uma determinada linha a um ponto escolhido. Essa informação é acessível de qualquer lugar via internet, e permite ao usuário do transporte público planejar melhor suas viagens, evitando perda de tempo no ponto aguardando a chegada do ônibus.

Palavras-chave: Sistema de Posicionamento Global, Ônibus, Rastreamento, Transporte público, Estimativa de tempo.

ABSTRACT

One problem that is common for all major cities of the world, whatever continent they are in or human development index they have, is the chaotic traffic. It is a consensus among specialists that the only solution for this problem is a greater adoption of public transportation by the population. It is hard, however, to convince people to leave their cars home and struggle in a packed transportation system, with constant delays and unpredictable schedules.

This project's goal is to alleviate the public transportation problem by proposing a system which is capable of providing an estimate of the time that will take for the next bus to arrive to a particular bus stop. This information is accessible from everywhere through internet and allows the public transportation passengers to plan their trips better, avoiding waste of time on the bus stop waiting for the next vehicle.

Keywords: Global Positioning System, Bus, Tracking, Public transportation, Time estimate.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GPS Global Positioning System

DGPS Differential GPS

WAAS Wide Area Augmentation System

NMEA National Marine Electronics Association

GPS General Packet Radio Service

WAP Wireless Application Protocol

PCMCIA Personal Computer Memory Card International Association) ou celulares

GSM Global System for Mobile Communications

ISP Internet Service Provider

JVM Java Virtual Machine

JDBC Java Database Connectivity

EJB Enterprise JavaBeans

JSP JavaServer Pages

SUMÁRIO

1 INTRODUÇÃO

1.1 Objetivo

O objetivo deste trabalho é desenvolver um sistema capaz de fornecer, através de um aplicativo web, a previsão do tempo restante para um ônibus chegar a um determinado ponto, exibindo também a sua posição em tempo real. Para tanto, será utilizado um módulo embarcado no ônibus a ser monitorado, equipado com um receptor de sinal GPS e um modem GPRS, que envia periodicamente para um servidor dados acerca de sua posição e velocidade. A predição é realizada utilizando-se algoritmos que leva em conta informações históricas do trecho percorrido pelo ônibus, armazenadas em uma base de dados.

1.2 Motivação

1.2.1 Aplicabilidade prática

A principal motivação do trabalho é criar uma solução que auxilie na diminuição do tempo gasto no ponto de ônibus pelos usuários do sistema de transporte público, de modo que estes usuários consigam otimizar seu tempo e usufruir de uma melhor qualidade de vida. Uma pesquisa realizada pelo Ibope para o Observatório Cidadão Nossa São Paulo[1], que ouviu 1512 moradores de todas as regiões da cidade entre os dias 2 e 16 de dezembro de 2009, mostrou que a área de transporte/mobilidade é uma das que apresenta o maior índice de insatisfação. Com relação ao tempo de espera nos pontos de ônibus, a nota média dada pela população foi 4,0, numa faixa de 1 a 10.

Esta pesquisa realizada e nossas observações e experiências pessoais como usuários deixam evidente que existe um problema sério no transporte público das cidades brasileiras. Não é pretendido por este sistema resolver por completo o problema do transporte público, mas sim amenizá-lo, oferecendo maior previsibilidade do tempo das viagens para o usuário, para que

[1]Disponível em http://nossasaopaulo.org.br/observatorio/biblioteca/Pesquisa_IRBEM_Ibope_2010.pdf

este seja capaz de realizar um maior planejamento de sua viagem.

1.3 Organização

O conteúdo do trabalho está organizado da seguinte forma:

Capítulo 1 - Introdução São apresentados nesse capítulo os objetivos e a motivação do trabalho desenvolvido nas seções seguintes.

Capítulo 2 - Aspectos Conceituais Apresenta conceitos teóricos e tecnológicos básicos para a compreensão do trabalho aqui apresentado, bem como trabalhos que abordaram problemas semelhantes ao aqui proposto.

Capítulo 3 - Especificação do Projeto de Formatura Esse capítulo contém a especificação do projeto que foi implementado. São descritos os requisitos funcionais e não funcionais que o projeto preenche, bem como sua arquitetura, casos de uso e diagramas de classe.

Capítulo 4 - Metodologia Apresenta a metodologia de projeto que foi aplicada para planejar e acompanhar o desenvolvimento do trabalho.

Capítulo 5 - Projeto e Implementação Nesse capítulo são apresentados detalhes da implementação do que foi especificado no capítulo 3.

Capítulo 6 - Testes Aqui são apresentados os testes que foram utilizados para validar o projeto, bem como os seus respectivos resultados.

Capítulo 7 - Considerações Finais Esse capítulo apresenta as conclusões finais do grupo acerca do trabalho produzido, bem como trabalhos futuros que podem ser desenvolvidos a partir desse.

2 ASPECTOS CONCEITUAIS

2.1 Tecnologias e conceitos

2.1.1 Métodos de estimativa de tempo de chegada de ônibus

Muitos pesquisadores adotaram uma série de abordagens diferentes na tentativa de modelar o trânsito de veículos em grandes cidades e prever o tempo de chegada de ônibus aos pontos de parada. Dentre as mais recorrentes, podem ser citados métodos com dados históricos, abordagens baseadas em modelos e métodos estatísticos.

Um método bem conhecido que utiliza uma abordagem histórica é apresentado em (LIN; ZENG, 1999). Nele, além de dados de viagens anteriores, são apresentados algoritmos que usam a posição atual do ônibus, os horários agendados para cada ponto e o atraso do ônibus nos pontos já alcançados. Há também métodos que utilizam dados históricos específicos por horário e dia da semana, como o apresentado em (WEIGANG et al., 2005).

O Filtro de Kalman (KALMAN et al., 1960) é utilizado para determinar sistemas lineares capazes de prever e separar um sinal randômico (ruído) na modelagem de um determinado sistema. O algoritmo apresentado em (PADMANABAN; VANAJAKSHI; SUBRAMANIAN, 2009) utiliza essa ferramenta para fornecer melhores estimativas. A principal característica desse método é analisar o tempo de viagem do ônibus separadamente em dois componentes: o tempo de deslocamento e o tempo em que o ônibus fica parado no ponto aguardando o embarque dos usuários. A estimativa do tempo de chegada de um ônibus é feita separadamente para cada um desses componentes, usando um método específico para cada uma delas.

2.1.2 GPS

Global Positioning System, ou sistema de posicionamento global, é um sistema de informação eletrônico que fornece, via rádio, a um aparelho receptor móvel, a posição do mesmo com referência às coordenadas terrestres. É a tecnologia em torno da qual gira o projeto aqui

apresentado.

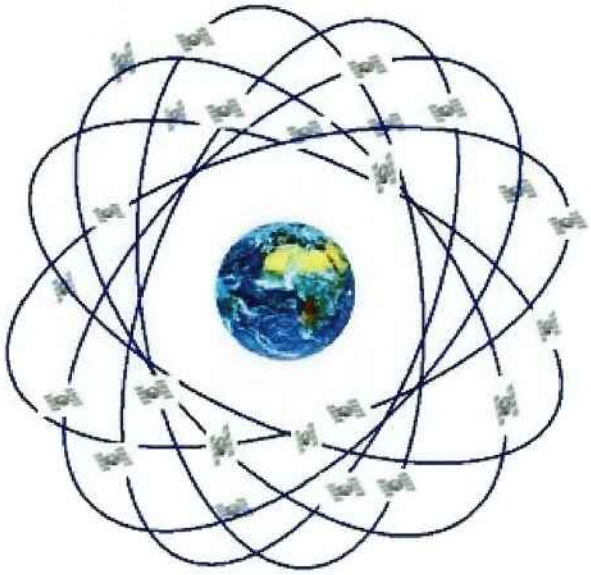

Figura 2.1: *Satélites do GPS.*

O GPS é um sistema de rádio-navegação baseado em satélites, desenvolvido e operado pelo DoD (Departament of Defense, EUA). É composto por uma constelação de 27 satélites operacionais (24 em operação e 3 extras caso haja falha nos outros), orbitando a uma altura de 20.200 km em 6 órbitas e com uma inclinação, em relação ao equador, de 55°, conforme a figura 2.1. Cada um deles efetua uma volta em torno da Terra a aproximadamente cada 12 horas, e possui uma vida útil de praticamente 10 anos.

O sistema possibilita a condição mínima para navegação: em qualquer lugar do mundo e a qualquer momento, existem pelo menos 4 satélites acima do plano do horizonte do observador. Além disso, o sinal do GPS é gratuito.

2.1.2.1 Histórico do GPS

O sistema de posicionamento global (GPS), desenvolvido pelos Estados Unidos, surgiu como necessidade da corrida armamentista entre os EUA e a União Soviética, com o objetivo de obter, em tempo real, a posição exata de uma determinada entidade. Os soviéticos desenvolveram um sistema equivalente, chamado Glonass (GLObal NAvigation Satellite System).

Do ponto de vista da navegação, o GPS surgiu como uma expansão do sistema de navegação NNSS/TRANSIT, da Marinha Americana. O GPS também é conhecido pela sigla NAVSTAR (NAVigation Satellite with Time And Ranging). Seu primeiro satélite foi lançado em 22 fevereiro de 1978 (ROCHA, 2003).

2.1.2.2 Princípio de funcionamento

Como descrito em (ROCHA, 2003) Os receptores de GPS de uso civil processam continuamente o cálculo da posição atual a partir dos dados de código. Dados de código são computados a partir de um código gerado pelo satélite GPS e depois transmitidos para o usuário, através de um sinal de rádio. O código de acesso livre usado é o C/A (*Course/Acquisition*) que fornece o SPS (*Standard Positioning Service*). Esses receptores GPS usam a frequência L1 de 1,575 GHz na banda UHF do espectro eletromagnético. A desvantagem dessa frequência é ela ser do tipo "line-of-sight", ou seja, não deve existir obstáculos entre a antena do receptor GPS e o céu.

Os satélites enviam dados modulados sobre uma onda que se propaga com uma frequência predefinida (L1: 1,575 GHz e L2: 1,227 GHz). Eles contêm sua posição, horário da transmissão, os meios para o cálculo da distância até o satélite (dados de efemérides, que são tabelas que fornecem, em intervalos de tempo regularmente espaçados, as coordenadas que definem a posição de um astro. Com base nas efemérides, o astronômo sabe quando é possível observar um determinado astro no firmamento da sua localidade), parâmetros de correção das influências atmosféricas, e o almanaque, constituído de dados genéricos da localização e estado de todos os satélites da constelação GPS.

O receptor GPS calcula a distância até os satélites GPS, cronometrando o tempo de viagem de um sinal do satélite ao receptor. Em determinado momento, o satélite inicia a transmissão de um longo padrão digital, chamado código pseudo-aleatório. O receptor produz o mesmo padrão digital no mesmo momento, uma vez que seus relógios estão sincronizados. O sinal do satélite chega ao receptor com um atraso em relação ao padrão por ele produzido, e a extensão desse atraso é igual ao tempo de viagem do sinal. O receptor multiplica esse tempo pela velocidade da luz para determinar qual distância o sinal viajou. Supondo que o sinal tenha viajado em linha reta, essa é a distância do receptor até o satélite.

Para realizar essa medição, tanto o receptor quanto o satélite necessitam de relógios que possam ser sincronizados no nível do nanossegundo ($10^{-9}s$). Para criar um sistema de posicionamento via satélite utilizando somente relógios sincronizados, seriam necessários relógios atômicos não somente em todos os satélites, mas também no próprio receptor. Isso seria um problema devido ao alto custo de um relógio atômico (entre US$50 mil e US$100 mil) para um receptor.

O sistema de posicionamento global possui uma solução eficaz para esse problema. Cada satélite contém um relógio atômico, mas o receptor utiliza um relógio de quartzo comum, reini-

ciado constantemente. O receptor observa os sinais provenientes de quatro ou mais satélites e ajusta a sua própria imprecisão. O valor correto de hora fará com que todos os sinais que o receptor está recebendo alinhem-se em um único ponto no espaço. Esse valor de hora é o mesmo dos relógios atômicos em todos os satélites. Dessa forma, o receptor ajusta seu relógio de acordo com esse valor de hora e passa a ter a mesma hora que todos os relógios atômicos têm em todos os satélites.

Os receptores devem enxergar ao menos 3 satélites para o cálculo da sua posição, através do método chamado trilateração. As distâncias entre o receptor e o satélite são consideradas como raios de esferas, cada uma delas tendo um satélite como centro. A posição do receptor será, então, o ponto comum de interseção entre as 3 esferas (existirão 2 pontos comuns de interseção, porém um deles não estará na superfície terrestre), como mostra a figura 2.2. Recebendo os sinais de 4 satélites ou mais, o receptor GPS poderá aumentar a precisão da posição e calcular a elevação ou altitude atual. A determinação da distância e da posição do satélite é calculada com base nos dados do almanaque, que é uma tabela dos números dos satélites com seus parâmetros orbitais.

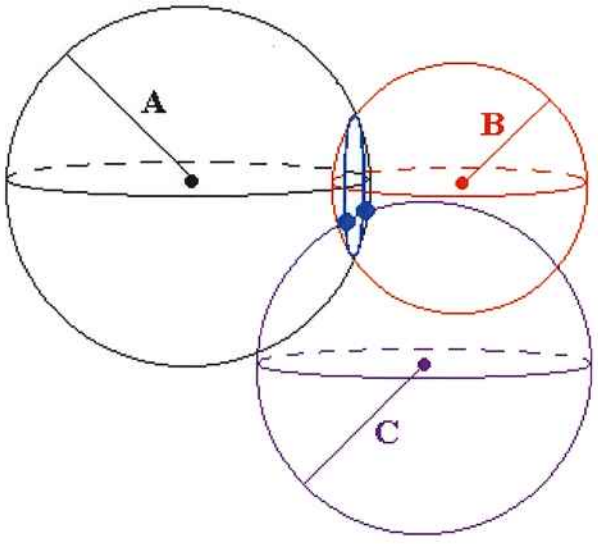

Figura 2.2: *Trilateração GPS.*

O receptor continuamente irá selecionar os melhores satélites em vista para o cálculo das posições a uma taxa de, na maioria dos receptores, uma posição por segundo.

O receptor GPS fornece coordenadas, altitude, velocidade, azimute e hora, sob condições favoráveis. Também pode extrair informações derivadas do tempo e posição, como distâncias, fotoperíodo (visibilidade solar, ou a exposiçao luminosa que um espaço físico recebe, medido em tempo e quantidade), e visibilidade lunar.

2.1.2.3 Precisão

Segundo (ROCHA, 2003), os usuários civis sofriam no passado a influência da S/A (*Selective Availability*), um fator que degradava propositadamente a precisão do GPS para um valor em torno de 100 metros na horizontal, 156 metros na vertical, e 340 nanosegundos no tempo. Esta influência foi eliminada no ano 2000, aproximando a precisão do código SPS (8-60 metros) ao militar, PPS (6-20 metros). Apesar disso, o código militar apresenta algumas diferenças; por exemplo, seu código é criptografado e apresenta uma melhor performance no que se refere à recepçao do sinal.

Diluição da precisão A diluição da Precisão (DOP) quantifica a influência da geometria da constelação de satélites na acurácia das coordenadas obtidas. Também é conhecido como coeficiente de rigidez.
É uma grandeza adimensional, variando de 1 a 10 (sendo 1 o melhor valor), e sua utilidade é indicar o melhor ou pior momento para se obter uma posição.

DGPS DGPS (*Differential GPS*) é um método utilizado, em tempo real ou pós-processamento, para remover a maioria dos erros da utilização do GPS.

A técnica consiste de um receptor GPS estacionário, sobre um ponto de coordenadas conhecidas (chamada de estação base), e alguns receptores itinerantes. Como estes receptores estão relativamente próximos, irão experimentar erros similares que podem ser corrigidos por cálculos diferenciais, permitindo uma precisão prática entre 3 e 5 metros. Os receptores GPS podem ser conectados a receptores sinalizadores (módulo diferencial de correção de posição) para receberem correções DGPS.

WAAS O WAAS - *Wide Area Augmentation System* - foi desenvolvido recentemente, e consiste de um sistema que corrige ainda mais o sinal civil do GPS. Tem a finalidade principal de fornecer uma maior precisão para os procedimentos de aproximações de aviões durante suas aterrissagens.

O sistema WAAS, composto de 25 estações terrestres no território norte-americano, e 2 satélites geoestacionários sobre o Equador, possibilita uma precisão de 3 metros ou menos.

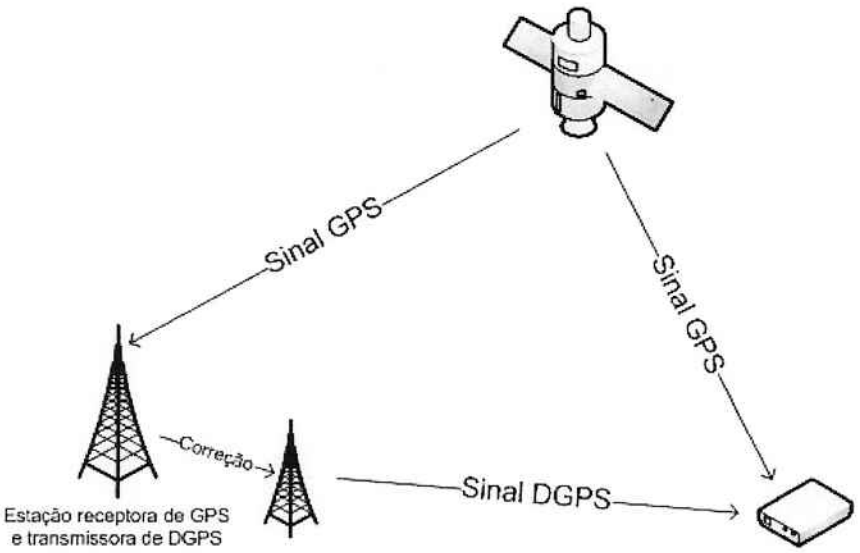

Figura 2.3: *Correção diferencial.*

2.1.2.4 Protocolo NMEA

NMEA 0183 é um conjunto de especificações de dados e elétricas para a comunicação de dispositivos eletrônicos de navegação (EL-RABBANY, 2002). No caso do presente projeto, a importância deste protocolo é ressaltada, pois é usado pelos receptores GPS. Este padrão é usado pela maioria dos dispositivos receptores de sinal GPS da atualidade, apesar de já existir um sucessor deste padrão, o NMEA 2000. Foi definido e é controlado pela associação norte-americana *National Marine Electronics Association.*

O padrão NMEA 0183 utiliza um padrão simples de comunicação serial ASCII, que define como os dados são transmitidos por meio de uma "sentença"ou "frase". As principais características desse padrão de comunicação estão descritos na tabela 2.1. Na camada de aplicação, o padrão define o conteúdo de cada tipo de sentença (mensagem).

Tabela 2.1: Configuração serial do padrão NMEA 0183 (camada de enlace de dados)

Característica	Valor
Bit rate	4800
Bits de dados	8
Paridade	Nenhuma
Bits de parada	1
Handshake	Ausente

- Cada caractere inicial de uma mensagem é um sinal de dólar ($).
- Os próximos 5 caracteres identificam o comunicador (2 caracteres) e o tipo da mensagem (3 caracteres).

- Todos os campos de dados que seguem são delimitados por vírgula
- Quando os dados estão indisponíveis, o campo correspondente contém bytes NUL (nulos). No caso "123,,456", por exemplo , os dados do segundo campo estão indisponíveis.
- O caractere que segue o último campo de dados é um asterisco.
- O asterisco é seguido por um checksum de 2 dígitos representando um número hexadecimal. O checksum representa o OU exclusivo de todos os caracteres entre $ e *. Este checksum é opcional para a maioria das sentenças, mas é compulsório para RMB e RMC (entre outros).
- <CR><LF> termina a mensagem.

Como exemplo, o alarme para a chegada em um ponto possui a seguinte forma:

*$GPAAM,A,A,0.10,N,WPTNME*32*

onde:

Tabela 2.2: Significado dos campos de uma mensagem NMEA GPAAM

Característica	Valor
GP	ID do comunicador (GP para uma unidade GPS, GL para GLONASS)
AAM	Alarme de chegada (Arrival alarm)
A	Entrou no círculo de chegada (Arrival circle entered)
A	Passou da perpendicular (Perpendicular passed)
0.10	Raio do círculo
N	Milhas Náuticas
WPTNME	Nome do waypoint (ponto de referência)
32	Dados do checksum

As principais mensagens de dados NMEA geradas pelos receptores GPS de navegação são as seguintes:

BOD - Origem, Destino e Azimutes (verdadeiro e magnético) da posição para o destino.

BWC - Rumo (Azimute) e distância (Great Circle - Linha Ortodrômica) para o destino.

GLL - Posição Geográfica (Latitude e Longitude).

GSA - Valores do DOP e satélites em uso.

GSV - Parâmetros orbitais dos satélites captados no momento.

RMB - Informações genéricas para navegação (XTE, BRG, Origem, Destino, SPD, etc.)

RMC - Latitude, Longitude, SOG, COG, Data, Declinação Magnética.

RTE - Rota Ativa: número da sentença (1 a 4), número da rota, nomes dos pontos.

WPL - Informação dos pontos (waypoints) do GPS: latitude, longitude, nome, etc.

ZDA - Data, hora e fuso horário.

2.1.3 GPRS

General Packet Radio Service, ou Serviço de Rádio de Pacote Geral, é um serviço que permite o envio e recepção de dados através de uma rede telefônica móvel. Esta tecnologia é utilizada no projeto aqui apresentado para envio dos dados recolhidos pelo GPS ao servidor do sistema.

Segundo descrito em (SEURRE; SAVELLI; PIETRI, 2003), o GPRS é uma evolução das redes GSM ou TDMA (IS-95). Por causa disso, o GPRS é normalmente definido como uma tecnologia 2.5G. O GPRS utiliza a infra-estrutura de rede celular existente com um upgrade de software nas estações e a adição de um *gateway* GPRS que conecta a rede GPRS à Internet.

Assim como no GSM, os pacotes de dados também são enviados através de múltiplos slots de tempo, mas não existe reserva. Os slots são alocados conforme a demanda dos pacotes enviados ou recebidos. Consegue-se, desta forma, um serviço de dados com conexão permanente, sem a necessidade de reservar permanentemente slots de tempo para o transporte de dados.

As principais características do GPRS são:

- Taxa de transporte de dados máxima de 26 a 40 kbit/s, podendo chegar (na teoria) a 171,2kbit/s.
- Conexão de dados sem a necessidade de se estabelecer um circuito telefônico, o que permite a cobrança por utilização e não por tempo de conexão, e faz com que o serviço esteja sempre disponível para o usuário.
- Padronizado para transporte de dados definidos pelos protocolos IP e X.25.

É possível acessar a Internet por GPRS usando um celular, PDA, notebook ou PC. Um celular com WAP (*Wireless Application Protocol*) ou microbrowser XHTML pode ser usado para navegar pelos websites WAP, checar e-mails e realizar outras atividades. UM PDA, notebook ou PC precisa estar equipado com um modem GPRS para acessar a Internet por meio de uma rede GPRS. Modem GPRS estão disponíveis na forma de cartões CF (*Compact Flash*), cartões PCMCIA (*Personal Computer Memory Card International Association*) ou celulares.

Para usar um celular como um modem GPRS, é preciso possuir os drivers do modem do celular do fabricante e um tipo de conexão ao seu computador (*Bluetooth*, IrDA ou cabo serial (USB)).

O acesso à internet por GPRS é oferecido por operadoras de rede celular utilizando o padrão GSM (*Global System for Mobile Communications*), como mostra a figura 2.4. As operadoras geralmente agem como ISPs (Internet Service Providers) e possuem *gateways* à Internet. Uma provedora GPRS pode oferecer vários planos de serviço, como taxa mensal fixa para tempo ilimitado de conexão, cobrança baseada no número de bytes transferidos e pacotes pré-pagos.

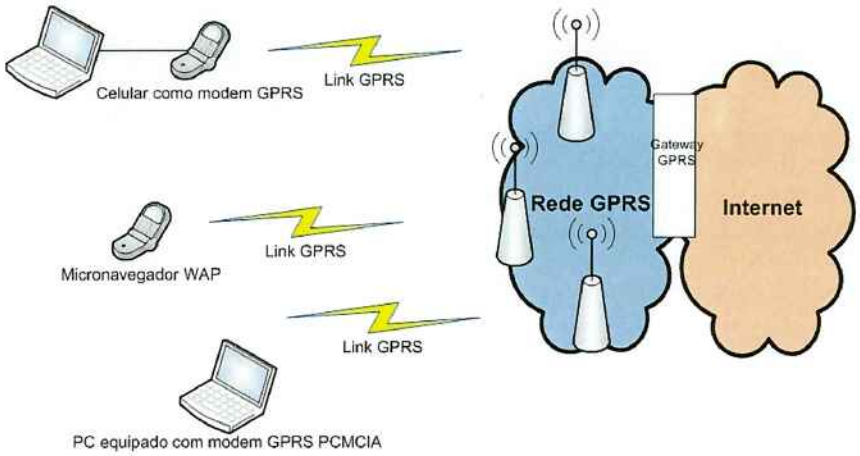

Figura 2.4: *Acesso à Internet através do GPRS.*

2.1.4 Comunicação Serial

2.1.5 Java

Java foi a linguagem de programação escolhida para o desenvolvimento do projeto. As principais razões para tal escolha foram: a familiaridade dos integrantes do grupo com a linguagem, e a sua grande popularidade, contando com muitas bibliotecas já estabelecidas prontas para uso (principalmente as envolvendo recursos de rede), que tornam o desenvolvimento mais rápido, e, dessa forma, nos possibilita focar nos aspectos mais relevantes do projeto em si.

O código de um software escrito na linguagem Java é compilado para uma forma intermediária de código, denominada *bytecode*, a qual por sua vez é interpretada pelas Máquinas Virtuais Java (JVMs).

Algumas das principais características da linguagem Java:

- Orientação a objetos.
- Portabilidade (Independência de plataforma) através da Máquina Virtual Java (*Java Virtual Machine* - JVM), um programa que carrega e executa os aplicativos Java, convertendo os bytecodes em código executável de máquina. A JVM é responsável pelo gerenciamento dos aplicativos, à medida que são executados. Graças à máquina virtual Java, os programas escritos em Java podem funcionar em qualquer plataforma de hardware e software que possua uma versão da JVM, tornando assim essas aplicações independentes da plataforma onde funcionam.
- Recursos de Rede - Possui extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP;
- Facilidade - a linguagem é derivada das linguagem C e C++, sendo assim familiar. Além disso, o ambiente retira do programador a responsabilidade de gerenciar a memória e os ponteiros;
- Simplicidade na especificação, tanto da linguagem como do ambiente de execução (JVM);
- É distribuída com um vasto conjunto de bibliotecas (ou APIs);
- Possui facilidades para criação de programas distribuídos e multitarefa;
- Desalocação de memória automática por processo de coleta de lixo (garbage collector);
- Carga Dinâmica de Código - Programas em Java são formados por uma coleção de classes armazenadas independentemente e que podem ser carregadas no momento de utilização.
- Desempenho - a linguagem Java suporta vários recursos de alto desempenho, como multithreading, compilação just-in-time e utilização de código nativo. O byte-code do Java possui um desempenho aproximadamente 20 vezes inferior que o de um código nativo do processador (ou linguagem de máquina), mas que em comparação com linguagens Script, como o HTML, o JavaScript ou o VBScript, ou até mesmo o próprio byte-code do VB (Visual Basic), possui uma performance superior.

O Java possui ótima integração com bancos de dados relacionais através do *Java Database Connectivity*, ou JDBC, um conjunto de classes e interfaces (API) escritas em Java que fazem o envio de instruções SQL para qualquer banco de dados relacional.

Para cada banco de dados, há um driver JDBC (instalado na máquina cliente), que converte requisições de programas Java para um protocolo que seja conhecido do Sistema Gerenciador de Banco de Dados (SGBD) em questão. O driver pode ser classificado em quatro categorias:

Tipo 1: Ponte JDBC-ODBC É o tipo mais simples, porém restrito à plataforma Windows. Utiliza ODBC para conectar-se com o banco de dados, convertendo métodos JDBC em chamadas às funções do ODBC. Esta ponte é normalmente usada quando não há um driver puro-Java (tipo 4) para determinado SGBD.

Tipo 2: Driver API-Nativo O driver API-Nativo traduz as chamadas JDBC para as chamadas da API cliente do banco de dados usado. Como a Ponte JDBC-ODBC, pode ser necessário software extra instalado na máquina cliente.

Tipo 3: Driver de Protocolo de Rede Traduz a chamada JDBC para um protocolo de rede independente do banco de dados utilizado, que é traduzido para o protocolo do banco de dados por um servidor. Por utilizar um protocolo independente, pode conectar as aplicações clientes Java a vários SGBDs diferentes. É o modelo mais flexível e pode ser visto como um driver intermediário.

Tipo 4: Driver nativo Converte as chamadas JDBC diretamente para o protocolo do banco de dados. Implementado em Java, normalmente é independente de plataforma e escrito pelos próprios desenvolvedores. É o tipo mais recomendado para ser usado.

2.1.5.1 PostgreSQL

O PostgreSQL é um projeto open source coordenado pelo PostgreSQL Global Development Group. Ele é um Banco de Dados Objeto-Relacional, ou seja, suporta diretamente classes, objetos e instâncias em seu esquema (*schema*) e na sua linguagem de consulta (*query language*), mantendo a sinergia com o sistema desenvolvido em Java.

Alguns dos principais recursos do PostgreSQL são:

- Consultas complexas

- Chaves estrangeiras
- Visões
- Estrutura para guardar dados Georeferenciados PostGIS
- Linguagem Procedural em várias linguagens (PL/pgSQL, PL/Python, PL/Java, PL/Perl) para Procedimentos armazenados (Stored procedures);
- Índices: é incluso suporte para índices do tipo B+-tree, hash, GiST e GiN.
- Triggers: são eventos "disparados"pela execução de comandos SQL DML.
- Controle de concorrência multi-versão por MVCC: o PostgreSQL gerencia concorrência através do sistema chamado Multi-Version Concurrency Control (MVCC), eliminando a necessidade de travas para leitura, e assegura que o banco mantém o princípio ACID (Atomicidade, Consistência, Isolamento e Durabilidade) de uma maneira eficiente.
- Herança: tabelas podem herdar características de uma tabela-pai.

2.1.5.2 Apache Tomcat

O Tomcat é um container de *servlets* de código aberto criado pela *Apache Software Foundation*. Um *servlet* é uma classe Java que segue a *Java Servlet API*, interface pela qual uma classe Java recebe requisições HTTP.

Não pode ser considerado um servidor de aplicação por não implementar toda a especificação para tal, não possuindo por exemplo suporte a EJB. O Tomcat possui três componentes: Catalina, Coyote e Jasper.

Catalina É o container de *servlets*. Implementa as especificações de servlet e de *JavaServer Pages* (JSP).

Coyote É o conector HTTP do Tomcat. É responsável por escutar as requisições em uma porta TCP e redirecioná-las para a *engine* do Tomcat, que as processa e retorna respostas para clientes também através do Coyote.

Jasper É o *Engine* JSP do Tomcat. Ele analisa arquivos JSP e os compila como servlets, que serão posteriormente manipulados pelo Catalina.

O Apache Tomcat está atualmente na versão 6.

2.1.5.3 Google Maps

Serviço de busca e visualização de mapas gratuito fornecido pelo Google. Além de mapas, é possível visualizar imagens de satélite, visões panorâmicas de ruas (Google Street View) e imagens da Lua e Marte.

O Google Maps possui uma API para *Javascript* que permite o uso de seus recursos e sua incorporação em sites. O serviço também é gratuito e pode ser utilizado até mesmo em sites comerciais, com restrições apenas para sites onde os usuários pagam pelo serviço ou para uso por meio de intranet. O Google fornece uma seção para desenvolvedores em seu site, contendo ampla documentação e exemplos.

O serviço da API do Google Maps que foi utilizado no desenvolvimento do projeto foi o da solicitação da pesquisa de rotas entre dois ou mais lugares.

A API faz uma requisição passando alguns parâmetros como: endereço de origem, endereço de destino, uma lista de endereços de pontos intermediários entre outros. O retorno da requisição é uma rota com uma lista de trechos. Cada trecho tem os dados de origem e destino (com dados de latitude e longitude de cada), e a distância em metros desse trecho.

Os objetos da API utilizados no projeto são descritos nas tabelas 2.3, 2.4, 2.5 e 2.6.

O serviço do Google Maps se tornou um dos principais componentes para o funcionamento do sistema, já que possibilitou a obtenção de dados geográficos de rotas configuradas na aplicação. Além disso, tanto o cadastro das linhas como a visualização dos dados se tornou muito mais fácil e intuitiva, e o resultado mais satisfatório aos usuários do sistema.

2.2 Estado da Arte

A seguir, são apresentados sistemas que apresentam funcionalidades e propósitos semelhantes ao aqui proposto.

SPTrans - Olho Vivo Existe hoje em São Paulo, implementado pela companhia SPTrans, um sistema que também visa fornecer estimativas de tempo de chegada de ônibus para a população. O serviço só atinge ônibus que circulam em corredores dedicados. O sistema utiliza GPS para aquisição de dados sobre a posição dos ônibus. Os dados são enviados pelo ônibus utilizando GPRS.

Segundo relatório técnico da SPTrans[1] , o algoritmo de estimativa de tempo utiliza dados históricos para calcular o tempo. O tempo necessário para percorrer o trecho entre um ponto e outro é armazenado em um banco de dados. O sistema determina quantos trechos estão entre a posição atual do ônibus e o ponto de parada para o qual se deseja realizar a estimativa e soma as médias dos tempos armazenados no banco de dados para aqueles trechos.

A estimativa de tempo é exibida somente no próprio ponto de parada, através de painéis de mensagens variáveis (PMVs). Existe também um sistema chamado Olho Vivo, que utiliza os dados dos ônibus para gerar um mapa de fluidez nos corredores de ônibus, fornecendo velocidades e tempos médios de viagem para cada corredor.

NextBus A empresa WebTech Wireless, que atua na área de rastreamento de frotas por GPS, possui um sistema com funcionalidades semelhantes às do sistema aqui proposto, o NextBus. A empresa implementou essa solução com sucesso em cidades dos Estados Unidos como Chicago e São Fransisco, porém não há notícia de alguma cidade brasileira que possua um sistema parecido.

O NextBus é capaz de apresentar as estimativas de tempo de chegada pela internet, por PDAs e *smartphones*, por SMS e painéis eletrônicos nos pontos. O sistema também foi implementado com sucesso em linhas de trem. O fabricante não fornece informações sobre o método utilizado para chegar na estimativa de tempo.

[1]Sistemas Informatizados Para a Gestão do Transporte Coletivo do Município de São Paulo - Maio/2009

Tabela 2.3: Especificação do objeto DirectionsRequest

Propriedades	Tipo	Descrição
avoidHighways	boolean	Se esta propriedade for definida como "true", ela instrui o serviço de rotas a evitar as rodovias onde for possível. Opcional.
avoidTolls	boolean	Se esta propriedade for definida como "true", ela instrui o serviço de rotas a evitar as estradas com pedágio onde for possível. Opcional.
destination	LatLng | string	Local de destino. Pode ser especificada como uma string a ser geocodificada ou um objeto LatLng. Obrigatório.
optimizeWaypoints	boolean	Se esta propriedade for definida como "true", o DirectionService tentará reordenar os pontos de referência intermediários fornecidos para minimizar o custo geral do trajeto. Se os pontos de referência forem otimizados, inspecione DirectionsRoute waypoint order na resposta para determinar a nova ordem.
origin	LatLng | string	Local de origem. Pode ser especificada como uma string a ser geocodificada ou um objeto LatLng. Obrigatório.
provideRouteAlternatives	boolean	Indica se trajetos alternativos devem ser fornecidos ou não. Opcional.
region	string	Código de região utilizado como polarização para solicitações de geocodificação. Opcional.
travelMode	BICYCLING | DRIVING | WALKING	Tipo de rota solicitada. Obrigatório.
unitSystem	IMPERIAL | METRIC	Sistema de medidas escolhido para ser usado ao exibir a distância. Por padrão, o sistema de medidas usado no país de origem é definido.
waypoints	Array<DirectionsWaypoint>	Matriz de pontos de referência intermediários. As rotas serão calculadas da origem ao destino com base em cada ponto de referência nesta matriz. Opcional.

Tabela 2.4: Especificação do objeto DirectionsResult

Propriedades	Tipo	Descrição
routes	Array<DirectionsRoute>	Uma matriz de DirectionsRoutes, cada um contendo informações sobre os trechos e etapas dos quais é composto. Haverá apenas um trajeto, a menos que DirectionsRequest seja formado de provideRouteAlternatives definidas como true. Anteriormente, esta propriedade era conhecida como "viagens".

Tabela 2.5: Especificação do objeto DirectionsRoute

Propriedades	Tipo	Descrição
bounds	LatLngBounds	Os limites para este trajeto.
copyrights	string	O texto de direitos autorais a ser exibido para este trajeto.
legs	Array<DirectionsLeg>	Uma matriz de DirectionsLegs, cada um contendo informações sobre as etapas das quais é composto. Haverá um trecho para cada ponto de referência ou destino especificado. Assim, um trajeto sem pontos de referência conterá um DirectionsLeg e um trajeto com um ponto de referência conterá dois. Anteriormente, esta propriedade era conhecida como "trajetos".
overview_path	Array<LatLng>	Uma matriz de LatLngs que representam o percurso inteiro deste trajeto. O caminho é simplificado para se adequar a contextos onde é necessário um pequeno número de vértices (como os URLs da Google Static Maps API)
warnings	Array<string>	Avisos a serem exibidos quando estas rotas estiverem sendo mostradas.
waypoint_order	Array<number>	Se optimizeWaypoints for definida como true, este campo conterá a nova ordem dos pontos de referência da entrada. Por exemplo, se a entrada for: Origem: Los Angeles Pontos de referência: Dallas, Bangor, Phoenix Destino: Nova York e a saída for ordenada da seguinte maneira: Origem: Los Angeles Pontos de referência: Phoenix, Dallas, Bangor Destino: Nova York este campo será uma Array contendo os valores [2, 0, 1]. A numeração dos pontos de referência é baseada em zero. Se algum ponto de referência da entrada tiver um campo stopover definido como false, este campo será vazio, pois a otimização de trajeto não está disponível para esse tipo de consulta.

Tabela 2.6: Especificação do objeto DirectionsLeg

Propriedades	Tipo	Descrição
distance	DirectionsDistance	A distância total coberta por este trecho. Esta propriedade pode ser indefinida quando a distância não for conhecida.
duration	DirectionsDuration	A duração total deste trecho. Esta propriedade pode ser indefinida quando a duração não for conhecida.
end_address	string	O endereço do destino deste trecho.
end_location	LatLng	O DirectionsService calcula rotas entre os locais usando a opção de transporte mais próxima (geralmente uma estrada) nos locais de início e término. end_location indica o destino geocodificado real, que pode ser diferente do end_location da última etapa se, por exemplo, a estrada não for próxima do destino deste trecho.
start_address	string	O endereço da origem deste trecho.
start_location	LatLng	O DirectionsService calcula rotas entre os locais usando a opção de transporte mais próxima (geralmente uma estrada) nos locais de início e término. start_location indica a origem geocodificada real, que pode ser diferente do start_location da primeira etapa se, por exemplo, a estrada não for próxima da origem deste trecho.

3 ESPECIFICAÇÃO DO PROJETO DE FORMATURA

3.1 Requisitos

3.1.1 Requisitos funcionais

Nessa seção são levantados e descritos todos os requisitos funcionais e não-funcionais do sistema a ser implementado.

3.1.1.1 Estimativa do tempo de chegada

É a principal funcionalidade do sistema. O sistema deve ser capaz de prever o tempo necessário para que o ônibus mais próximo de uma determinada linha chegue em um de seus pontos de parada. O sistema deve apresentar ao usuário as linhas de ônibus existentes (cadastradas no sistema). Ao selecionar a linha desejada, o usuário deverá escolher um ponto de ônibus, dentre uma lista de todos os pontos da linha selecionada. A partir destas informações, o sistema deverá estimar o tempo de chegada do próximo ônibus da linha escolhida pelo usuário à parada de ônibus desejada. Esse cálculo será realizado a partir de um algoritmo específico e dados históricos armazenados em uma base de dados.

3.1.1.2 Exibição da posição atual do ônibus

O sistema deve ser capaz de exibir em um mapa a posição atual do ônibus mais próximo a um determinado ponto.

3.1.1.3 Cadastro e manutenção de linhas e pontos de ônibus

O sistema deve possibilitar que o administrador do sistema cadastre novas linhas de ônibus e configure suas rotas e paradas, ou alterar linhas já existentes. Deve existir uma interface dedicada a esse fim.

3.1.2 Requisitos não-funcionais

3.1.2.1 Usabilidade

Dado que o objetivo de fornecer a estimativa de tempo de chegada de um ônibus a um determinado ponto é fazer com que o usuário não perca tempo esperando no ponto, imagina-se que o usuário típico do sistema esteja de saída de sua casa ou local de trabalho. Por isso, o site deve apresentar uma interface simples, que forneça todas as informações que o usuário busca com o mínimo de cliques necessários. Uma interação simples com o sistema deve levar em torno de um minuto.

3.1.2.2 Desempenho

Pelo mesmo motivo explicitado na seção 3.1.2.1, o sistema deve apresentar razoável desempenho. Foi estipulado um tempo máximo de 5 segundos para a obtenção da previsão do tempo de chegada.

3.1.2.3 Compatibilidade

O sistema deve ser compatível com a grande maioria dos *browsers* disponíveis no mercado, bem como para celulares e PDAs. Ficou estabelecido que o sistema deverá ser comprovadamente funcional com pelo menos 80% dos *browsers* e com os três principais *Smartphones* do mercado: *iPhone*, *Blackberry* e plataforma *Android*. A tabela 3.1 mostra estatísticas do uso de navegadores na internet em Setembro de 2010, com os navegadores a serem suportados em destaque.[1].

3.1.2.4 Precisão

Se por um lado a estimativa do tempo de chegada do ônibus é a principal funcionalidade do sistema e é fundamental que ela forneça dados consistentes, é sabido que o trânsito contém elementos aleatórios e portanto muito difíceis de serem modelados a contento. Fatores como chuva, acidentes ou simplesmente excesso de veículos ocorrem com frequência em cidades como São Paulo e podem interferir na estimativa.

[1]Disponível em http://www.w3counter.com/globalstats.php?year=2010&month=9

Tabela 3.1: Estatísticas do uso de navegadores - Setembro de 2010

Posição	Browser	Porcentagem
1	**Internet Explorer 8**	27,5%
2	**Firefox 3.6**	23,6%
3	**Internet Explorer 7**	10,0 %
4	**Chrome 6**	8,3%
5	**Internet Explorer 6**	5,8%
6	**Safari 5**	4,2%
7	Firefox 3.5	4,2%
8	Chrome 5	2,6%
9	Firefox 3	2,1%
10	Safari 4	1,0 %
-	Outros	10,3 %

3.2 Arquitetura

3.2.1 Definição do Sistema

O sistema realiza o cálculo da estimativa do tempo de chegada de um ônibus de linha a um ponto de ônibus qualquer escolhido, por meio do recebimento de dados de posicionamento e velocidade do ônibus. O sistema deve possuir uma interface Web, em que um usuário comum seja capaz de selecionar uma linha e um ponto de ônibus. Como resposta, o usuário deve receber o tempo que o próximo ônibus da linha levará para chegar ao ponto escolhido.

3.2.2 Arquitetura do Sistema

Da forma com que o problema se apresenta, é natural dividir o sistema em dois módulos: módulo embarcado, que recebe e analisa dados providos por um receptor de sinal GPS, e os envia ao módulo central por rede de telefonia celular (em um sistema real funcionando com várias linhas, haverá vários módulos embarcados, um para cada onibus) ; e outro central, um servidor que consolida as informações recebidas dos diversos módulos embarcados do sistema, mostra a movimentação dos ônibus da linha escolhida em um mapa, calcula estimativas de tempo de chegada aos pontos, e as fornece ao usuário quando requisitadas.

Módulo central: Servidor (Website e Aplicação) Disponibiliza um portal disponível na internet, acessível tanto por computadores quanto por dispositivos móveis (celulares, PDAs, smartphones), onde os usuários podem realizar consultas, escolhendo um ponto de ônibus e uma linha de ônibus, e obter como resposta uma estimativa otimizada do tempo em que

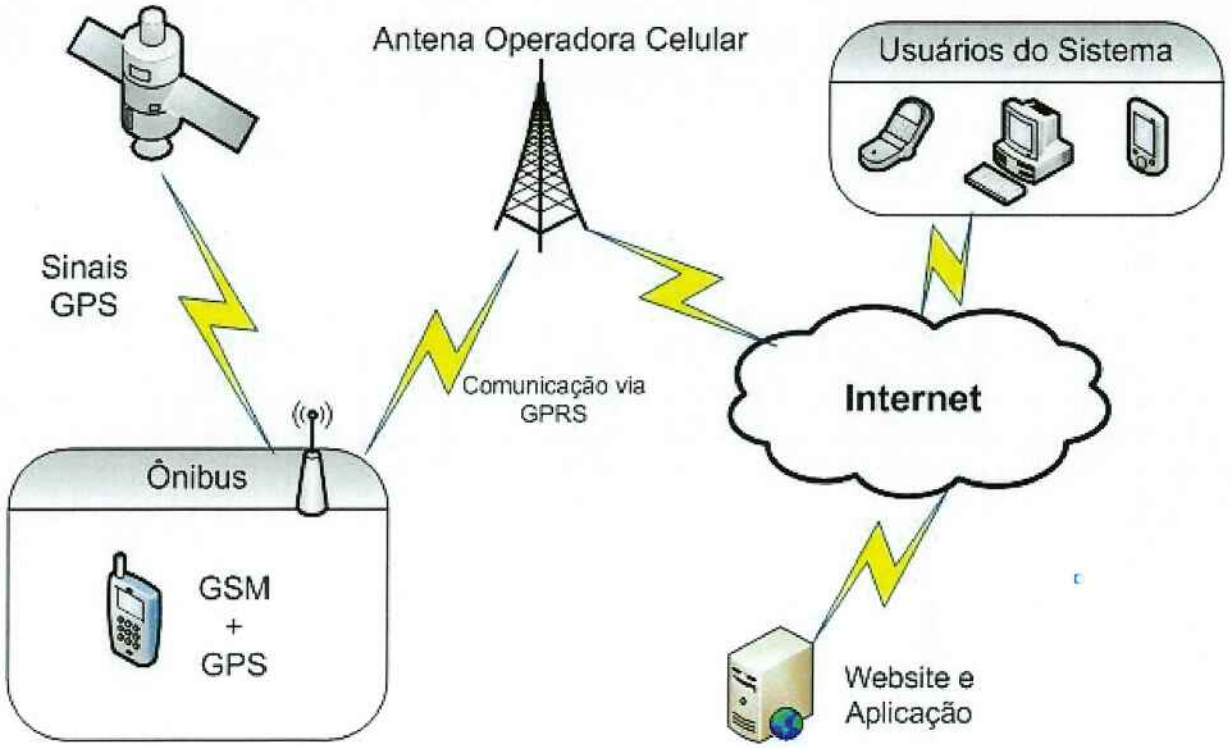

Figura 3.1: *Arquitetura do sistema.*

o próximo ônibus da linha desejada chegará ao ponto escolhido. Além disso, ao escolher a linha, o sistema mostrará em um mapa a movimentação dos ônibus da linha selecionada. Para realizar o cálculo da estimativa de tempo, esse módulo deve receber como entrada os dados de posicionamento e velocidade providos pelo módulo embarcado, adquiridos por meio do sistema GPS e enviados através do sistema de comunicação de redes celular GSM/GPRS, e utilizar um algoritmo específico de cálculo de estimativa de tempo, retornando os resultados para o website. A cada trecho percorrido pelo ônibus, a velocidade média apresentada pelo veículo neste percurso é incorporada ao banco de dados, com o objetivo de se manter uma base histórica de viagens, que deve ser utilizada no cálculo da estimativa de tempo de viagens futuras.

Módulo embarcado: Componente do ônibus Componente embarcado do sistema. Deve incluir um dispositivo com GPS, um mini-controlador ou dispositivo capaz de processamento de dados, e um modem GSM/GPRS para o envio de dados de velocidade e posicionamento do ônibus para o módulo central. A maioria dos receptores GPS recebem sinais dos satélites a cada segundo, e foi decidido que cada módulo embarcado deve enviar dados ao módulo central a cada 10 segundos; esse tempo pode ser diminuído caso haja problemas, como quanto à identificação de que um ônibus tenha passado por um ponto.

O componente do ônibus pode ser substituído por um simulador. Um software simples,

que deve simular as trajetórias dos ônibus de uma linha e servir como substituto ao módulo do ônibus, sendo seu objetivo principal o de testes do sistema; devem ser definidas variáveis aleatórias que permitam representar de modo aproximado as operações de uma linha de ônibus real. Seus dados serão fornecidos como entrada para o componente Servidor.

Um servidor de banco de dados é responsável pela interface entre o módulo central e os módulos embarcados. Ambos se comunicam com o servidor de banco de dados através de requisições HTTP do tipo POST. Isso confere ao sistema um alto desacoplamento entre os módulos principais, o que traz diversas vantagens no desenvolvimento e na manutenção do sistema.

3.2.3 Arquitetura do Módulo Web

O módulo Web foi projetado utilizando uma arquitetura em três camadas. Segundo (LIU, 2009), essa arquitetura é uma evolução natural da arquitetura cliente/servidor. Ela visa compartimentar os dados, a lógica e a apresentação de uma aplicação para que cada uma dessas camadas possa ser implantada em um servidor diferente, permitindo uma melhor distribuição da carga e do uso de *hardware* especializado para cada tipo de requisição. Essa arquitetura também torna o desenvolvimento mais rápido, possibilitando o trabalho em paralelo em cada uma das camadas, além de facilitar a substituição de uma das camadas quando necessário. O principal uso dessa arquitetura é em sistemas web.

Na figura 3.2 são mostradas as três camadas, cujas principais características são descritas a seguir:

Camada de apresentação É na camada de apresentação que a interação com o usuário ocorre. Por ela, dados são coletados e exibidos. Podem haver várias implementações dessa camada. No caso de *websites*, por exemplo, as implementações podem ser específicas para cada plataforma (computadores, PDAs e *tablets*, por exemplo).

Camada de negócio Nessa camada, os dados recebidos pela camada de apresentação ou retornados pela camada de dados são processados. Todos os algoritmos e manipulações dos dados devem estar aqui. No caso da aplicação aqui apresentada, a camada de dados representa o servidor Tomcat.

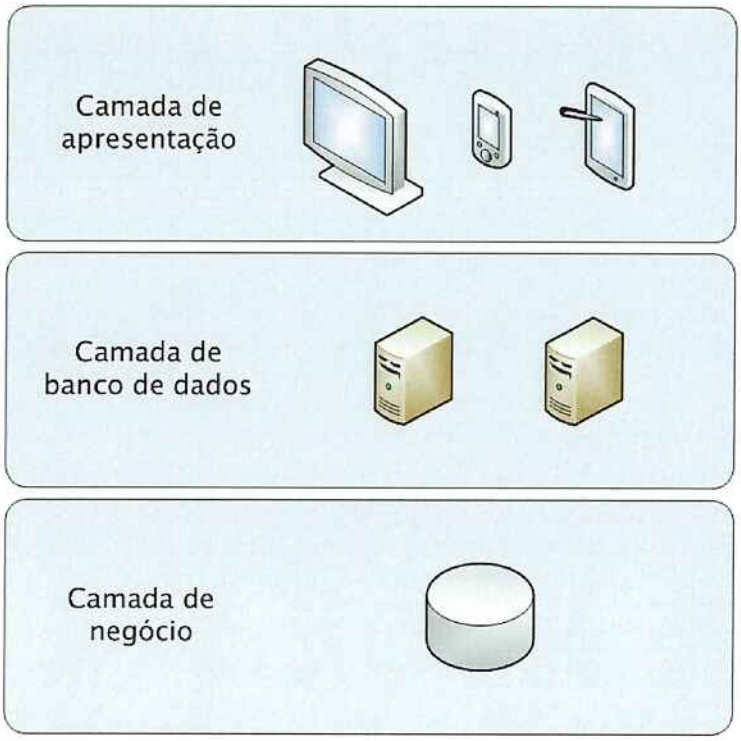

Figura 3.2: *Arquitetura em três camadas do módulo Web.*

Camada de banco de dados Consiste dos servidores de banco de dados. É nessa camada onde os dados são armazenados e recuperados. É importante que essa camada mantenha os dados independentes das demais camadas.

3.3 Casos de uso

São apresentados e descritos a seguir os principais casos de uso levantados para o sistema, de acordo com o diagrama de casos de uso representado na figura 3.3.

3.3.1 Atores

Os atores identificados no sistema são:

Administrador do Sistema: indivíduo pelo cadastro de linhas de ônibus e pontos de ônibus;

Usuário comum: interage com a interface Web, enviando dados ao sistema Web para o cálculo da estimativa de tempo de chegada.

Módulo Simulador ou Módulo do Ônibus: interage com o sistema Web, fornecendo dados de velocidade e posicionamento dos ônibus da linha.

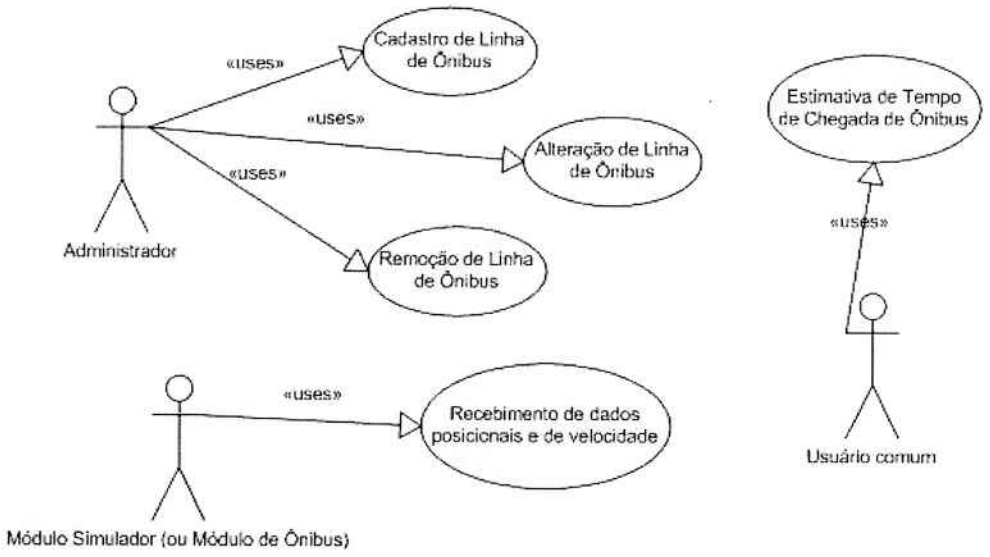

Figura 3.3: *Diagrama de casos de uso.*

3.3.2 Descrição dos casos de uso

3.3.2.1 Cadastro de linha de ônibus

Atores: Administrador do Sistema

Pré-condição: Administrador autenticado no sistema.

Descrição:

1. O administrador seleciona a opção "Cadastro de Linha de Ônibus".
2. O sistema apresenta um formulário com os seguintes campos: nome da linha, nome de sentidos (ida/volta) da linha (se necessário) e número de ônibus naquela linha.
3. O administrador digita os dados solicitados e confirma.
4. O sistema apresenta um formulário dinâmico de cadastro de pontos de ônibus, com os seguintes campos: nome do ponto (opcional), sentido da linha (ida/volta) e se o ponto é de parada. A posição geográfica do ponto será definida utilizando um mapa incorporado ao formulário, onde será possível marcar o ponto com um clique.
5. O administrador insere os dados e confirma.
6. O sistema armazena essas informações

Pós-condição: Cadastro de uma linha de ônibus e possíveis pontos de parada.

Exceção: Não se Aplica.

3.3.2.2 Alteração de linha de ônibus

Atores: Administrador do Sistema

Pré-condição: Administrador autenticado no sistema.

Descrição:

1. O administrador seleciona a opção "Alteração de Linha de Ônibus".
2. O sistema apresenta todas as linhas de ônibus cadastradas.
3. O administrador seleciona a linha desejada.
4. O sistema apresenta um formulário com os seguintes campos cadastrados: nome da linha, nome de sentidos (ida/volta) da linha.
5. O administrador altera os dados desejados e confirma.
6. O sistema apresenta os pontos de ônibus cadastrados, com os seguintes campos: nome do ponto (opcional), sentido da linha (ida/volta), coordenadas geográficas do ponto.
7. O administrador altera os dados desejados, exclui ou inclui novos pontos, e confirma.
8. O sistema armazena essas informações

Pós-condição: Alteração de uma linha de ônibus e seus pontos de parada.

Exceção: Não se Aplica.

3.3.2.3 Remoção de linha de ônibus

Atores: Administrador do Sistema

Pré-condição: Administrador autenticado no sistema.

Descrição:

1. O administrador seleciona a opção "Remoção de Linha de Ônibus".
2. O sistema apresenta todas as linhas de ônibus cadastradas.
3. O administrador seleciona a(s) linha(s) desejada(s) para remoção e confirma.
4. O sistema armazena essas informações

Pós-condição: Remoção de uma linha de ônibus e seus pontos de parada.

Exceção: Não se Aplica.

3.3.2.4 Consulta de Estimativa de Tempo de Chegada de Ônibus

Atores: Usuário comum

Pré-condição: Usuário acessa a seção de Estimativa de Tempo de Chegada de Ônibus do website

Descrição:

1. Sistema mostra ao usuário as linhas de ônibus cadastradas;
2. Usuário seleciona a linha desejada;
3. Sistema mostra ao usuário duas opções de sentido da linha (ida/volta)
4. Usuário seleciona a opção desejada;
5. Sistema mostra ao usuário uma lista dos pontos de ônibus cadastrados para o sentido escolhido;
6. Usuário seleciona o(s) ponto(s) de ônibus desejados;
7. O sistema, ao receber essa informação pela interface web, realiza o cálculo da estimativa de tempo de chegada do ônibus ao(s) ponto(s) desejados e informa na mesma página ao usuário.

Pós-condição: tempo de chegada estimado e mostrado ao usuário

Exceção: não se aplica

3.3.2.5 Recebimento de dados posicionais e de velocidade

Atores: Módulo simulador ou módulo do ônibus

Pré-condição: modulo ativado para enviar dados ao sistema Web

Descrição:

1. Após cada intervalo de tempo especificado, o ator envia as informações de posição e velocidade ao sistema Web

2. O sistema, ao receber essas informações, as identifica (sendo de qual linha/sentido/ônibus?), e armazena em uma base de dados, relacionadas a variáveis do algoritmo de cálculo de estimativa, como horário, data, tempo, etc.

Pós-condição: Dados posicionais e de velocidade gravados em uma base de dados, devidamente relacionados às variáveis do algoritmo do cálculo de estimativa, a cada intervalo de tempo.

Exceção: não se aplica.

3.4 Diagramas de classe

3.4.1 Módulo embarcado

O software projetado para o módulo embarcado pode ser representado pelo diagrama de classes da figura 3.4. A seguir são descritas as classes e suas interfaces.

3.4.1.1 FrontEnd

É a classe que implementa a interface do módulo embarcado, utilizando a biblioteca *Java Swing*. É responsável por invocar o laço principal do módulo quando o usuário pressiona o botão "começar". Pega também o número da porta COM onde está o receptor do GPS e utiliza-o para chamar a classe SerialReader corretamente.

3.4.1.2 GUIWriter

A classe GUIWriter interage com a interface do módulo embarcado, escrevendo os dados das mensagens lidas, já previamente analisados e verificados pela classe NMEAParser.

3.4.1.3 NMEAParser

A classe NMEAParser é responsável por retirar as informações de horário, latitude, longitude e velocidade contidas nas mensagens enviadas pelo aparelho GPS. Foram escolhidas as mensagens do tipo GPRMC para obter as informações necessárias. Foi utilizada também a variável Satellite Fix Status para determinar se a mensagem é válida ou não. Caso não seja, os dados são ignorados.

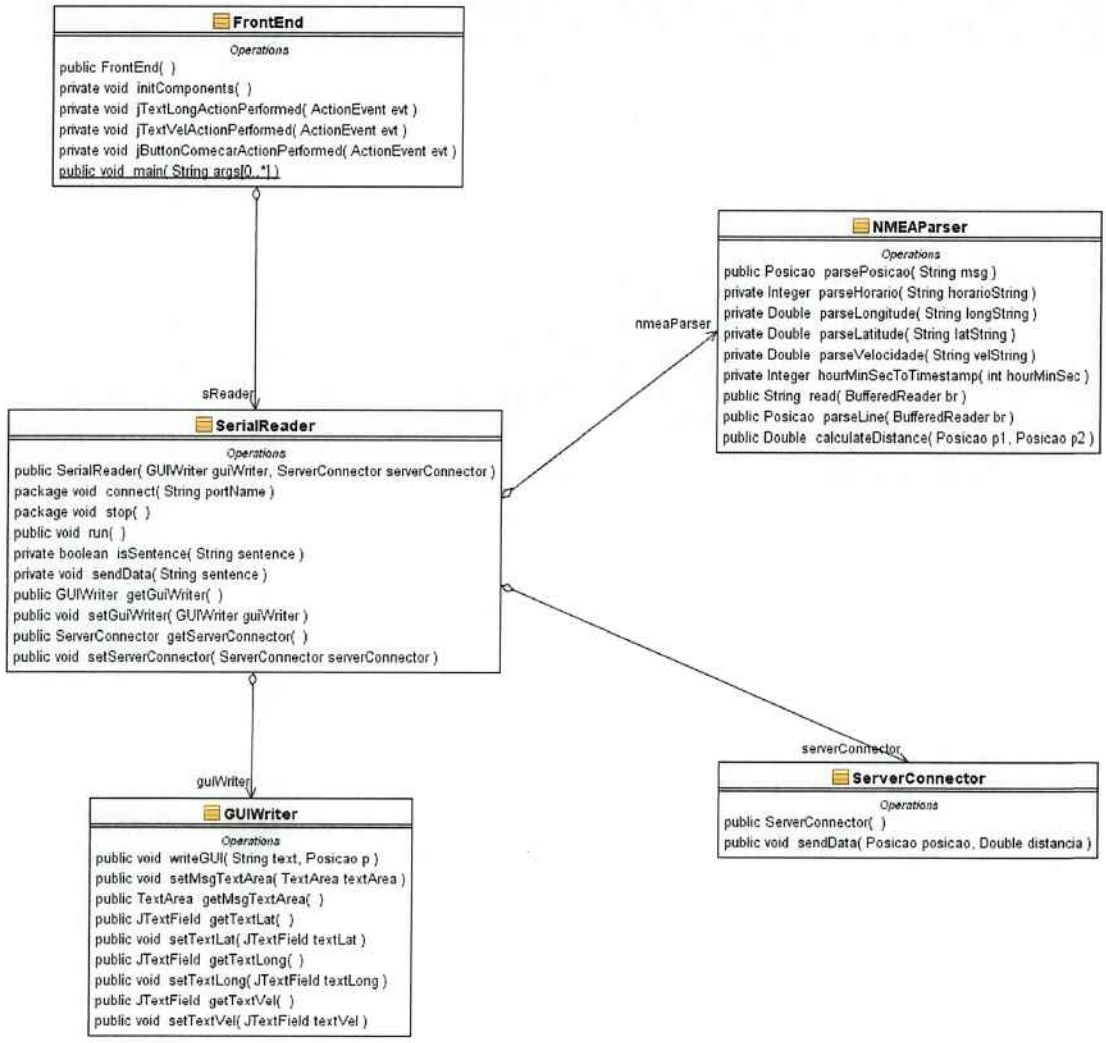

Figura 3.4: *Diagrama de classes do software do módulo embarcado.*

É necessário tratar as informações para obtê-las no formato esperado pela aplicação: a velocidade deve ser convertida de nós para km/h, enquanto o instante da medida deve ser convertido de ano/mês/dia/hora/minuto/segundo para UNIX timestamp.

3.4.1.4 SerialReader

A classe SerialReader faz a interface do sistema com o receptor GPS. Ela utiliza a biblioteca nativa RXTX para acessar a porta serial na qual está ligado o GPS e fazer a leitura.

Os dados vindos do GPS são recebidos de forma contínua, ou seja, são enviadas sequências de caracteres que não respeitam as divisões por mensagens. Sendo assim, cada leitura pode retornar uma parte de uma mensagem, uma mensagem completa ou partes de mais de uma

mensagem. Para analisar os dados de mensagens de forma separada, esses caracteres são acumulados e, quando o caractere de início de mensagem $ é recebido, os caracteres presentes no *buffer* (referentes à mensagem anterior) são enviados para o NMEAParser para análise. O *buffer* é então limpo e os dados da mensagem seguinte começam a ser acumulados.

Outro dado necessário para o funcionamento do algoritmo é a distância percorrida pelo ônibus desde o último ponto de parada (isto é, do começo do trecho atual). Como a API do Google Maps não faz esse cálculo, deve-se obtê-lo a partir dos dados do GPS, utilizando o método descrito em 5.1.0.1.

A aproximação apresentou resultados suficientemente precisos para a nossa aplicação, conforme é apresentado na seção 7. Essas mensagens são lidas através da classe SerialReader, que chama o NMEAParser e envia as informações para o servidor utilizando GPRS.

O envio dos dados para o servidor é delegado para a classe ServerConnector. A inserção dos dados lidos na GUI é feito pela classe GUIWriter.

3.4.1.5 ServerConnector

Classe responsável por enviar de fato as mensagens para o servidor. O envio é feito através de uma requisição HTTP do tipo POST para o servidor. Como resposta, é recebido um booleano que determina se o ônibus já ultrapassou o ponto de parada seguinte. Em caso positivo, deve-se zerar a distância até então somada.

3.4.2 Módulo web

O Módulo Web é dividido em cinco partes: Entidades, Gerenciadores, Servlets, Simulador e páginas web (jsps, css, javascript).

3.4.2.1 Entidades

O pacote de entidades faz o mapeamento das tabelas em classes java. Cada instância de entidade representada uma entrada na tabela associada.

3.4.2.2 Gerenciadores

A sessão de gerenciadores junta todas as classes java que fazem o processamento dos dados. Neles que estão toda a lógica do sistema como o cálculo da estimativa, o processamento

dos dados do embarcado, o gerenciamento de entidades entre outros.

3.4.2.3 Servlets

Servlet é um componente do lado servidor que gera dados HTML e XML para a camada de apresentação de um aplicativo Web. É basicamente uma classe na linguagem de programação Java que dinamicamente processa requisições e respostas, proporcionando dessa maneira novos recursos aos servidores. As Servlets criadas tratam todas as requisições do sistema, entre elas o cadastro de linhas e as consultas de estimativas de tempo.

3.4.2.4 Simulador

Foi desenvolvido também um simulador para testes do aplicativo e dos algoritmos de cálculo. Esse simulador representa um módulo embarcado, enviando dados ao servidor.

3.4.2.5 Páginas Web

As páginas web foram desenvolvidas utilizando a tecnologia das JavaServer Pages (JSP).

JavaServer Pages (JSP) é uma tecnologia utilizada no desenvolvimento de aplicações para Web, similar às tecnologias Active Server Pages (ASP) da Microsoft ou PHP. Por ser baseada na linguagem de programação Java, tem a vantagem da portabilidade de plataforma, que permite a sua execução em diversos sistemas operacionais, como o Windows da Microsoft, Unix e Linux.

Além disso foram utilizadas também tecnologia Javascript e Cascading Style Sheets (CSS), para tornar as páginas do aplicativo mais interativas e visuais.

3.5 Especificação do banco de dados

O diagrama na figura 3.5 mostra o modelo do banco de dados final do projeto.

3.6 Hardware do módulo embarcado

Para a escolha do hardware que seria utilizado no módulo embarcado, o grupo pesquisou diversas alternativas e as julgou usando como critérios viabilidade, custo e valor acadêmico. A

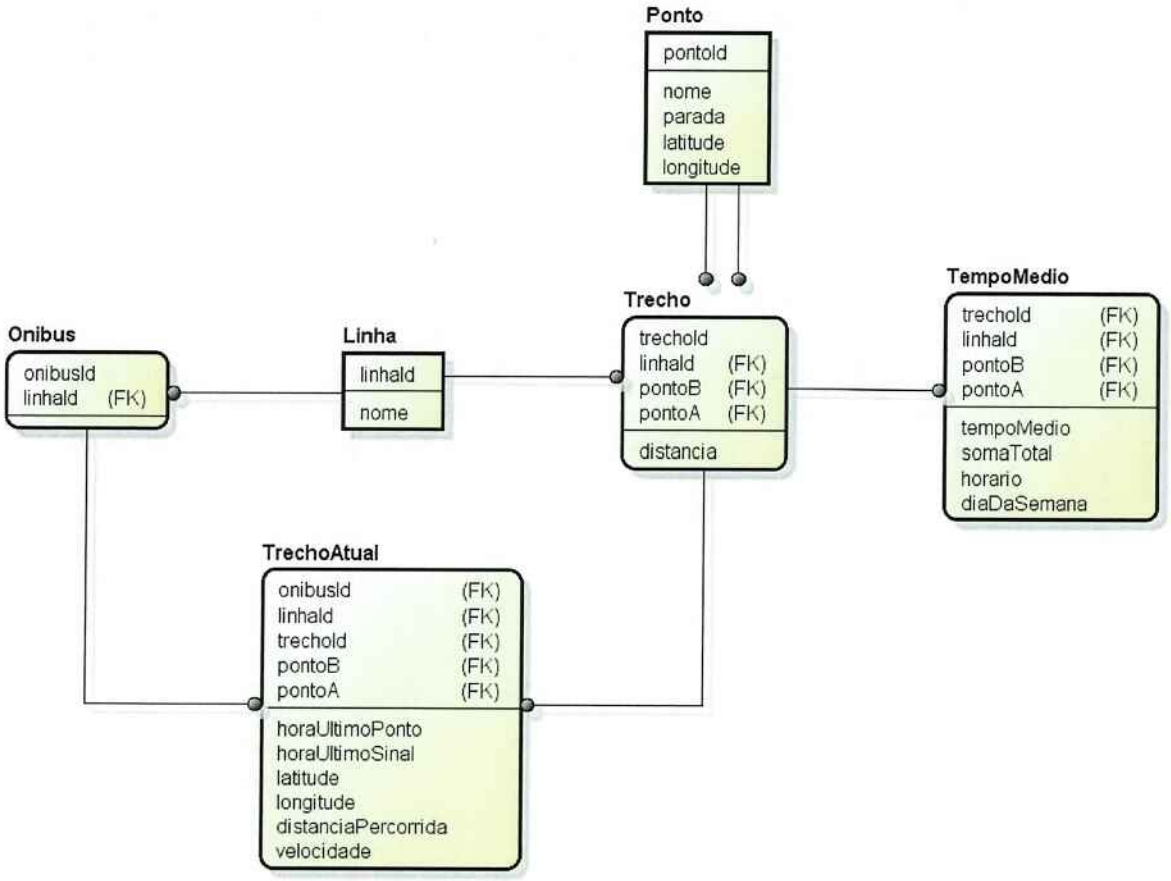

Figura 3.5: *Diagrama de Banco de Dados.*

seguir é apresentada uma breve descrição dessas alternativas, com seus prós e contras, além da solução que foi de fato adotada.

3.6.1 Alternativas consideradas

3.6.1.1 Módulo de Rastreamento de veículos

Componentes: Módulo rastreador de veículos

Descrição: Existem módulos de rastreamento à venda no mercado que fornecem todos os serviços necessários pelo projeto. Trata-se de um receptor de GPS que envia a cada segundo a posição e a velocidade do veículo para um servidor a escolher através de um modem GPRS.

Prós: É a solução mais simples. O único trabalho exigido é o de configuração do módulo para enviar os dados a um servidor.

Contras: Uma vez que o módulo é uma caixa-preta, é também a solução de menor valor acadêmico, já que seriam eliminados do escopo o projeto e a implementação do equipamento embarcado no ônibus. Também não haveria acesso à documentação e à arquitetura do módulo. Caso essa solução fosse adotada, seria necessário expandir as funcionalidades do aplicativo web para contrabalancear a simplicidade dessa parte do projeto.

3.6.1.2 G24-Java + GPS Module

Componentes: Módulo Wireless G24-Java e GPS Module Motorola

Descrição: O G24-Java é um módulo wireless stand-alone da Motorola que possui recursos muito semelhantes ao de um celular, tais como CPU, memória, I/O e transmissão de dados usando redes GSM e GPRS. Além disso, como o nome sugere, esse módulo pode programado usando a linguagem J2ME.

O GPS Module é um receptor de GPS não controlado que pode ser acoplado a um computador ou ao G24 para fornecer dados sobre posicionamento. Os módulos podem ser integrados como no diagrama de blocos apresentado na figura 3.6

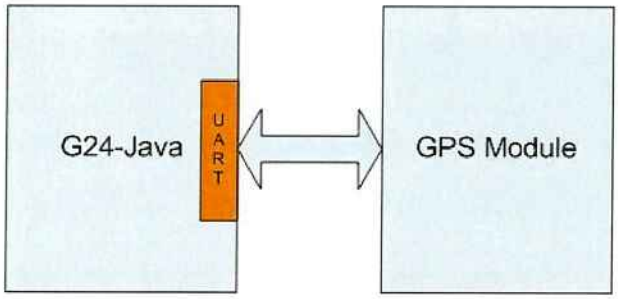

Figura 3.6: *Diagrama de blocos de uma possivel solução usando G24-Java.*

Essa solução nos fornece algo muito próximo do equipamento embarcado a ser utilizado em um sistema real de rastreamento de veículos. Os módulos são pequenos, operam com baixo consumo de energia e seu encapsulamento permite o uso em ambientes hostis.

Prós: O módulo G24 é bem versátil e permite a programação de todas as funcionalidades que o projeto exige. Além disso, o fato dele ser programado em J2ME e suportar a especificação JSR-183, que fornece classes para facilitar o acesso aos dados do GPS, torna o desenvolvimento muito simples. A Motorola fornece em seu site vasta documentação sobre ambos os módulos.

Contras: O preço dos componentes é alto e não foi encontrado nenhum fornecedor no Brasil que os vendesse em pequenas quantidades.

3.6.1.3 Receptor de GPS USB + Notebook + Celular

Componentes: Notebook, dispositivo receptor de GPS USB, celular com modem GPRS

Descrição: Essa solução utiliza um notebook como controlador do receptor GPS. Os dados são enviados para o servidor utilizando um modem GPRS. Para isso, pode ser utilizado um celular comum, já que a grande maioria dos aparelhos atuais possui essa funcionalidade. A arquitetura básica dessa solução está representada no diagrama da figura 3.7.

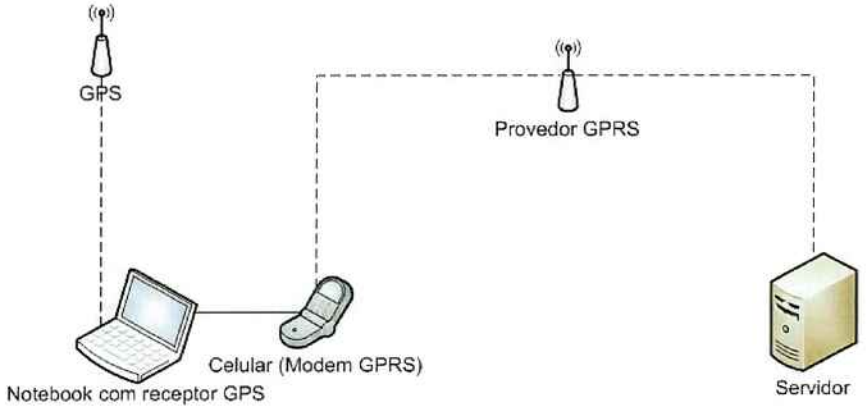

Figura 3.7: *Acesso à Internet atraves do GPRS.*

Prós: O receptor GPS é barato (em torno de 50 reais) e de fácil aquisição. Para os demais componentes, podem ser utilizados os notebooks e celulares de uso pessoal dos integrantes do grupo para realizar o teste. Essa solução também apresenta um razoável desafio técnico, já que o software embarcado terá de fazer o trabalho de interpretação das entradas do GPS, tratamento de imprecisões e envio de dados para o servidor.

Contras: Essa solução resulta em uma prova de conceito apenas. Uma vez que o notebook não é o equipamento mais indicado para esse fim por questões de custo, tamanho e resistência, é inviável colocá-los em diversos ônibus. Caso queira-se colocar o sistema em produção, é necessário substituir o notebook e o celular por componentes apropriados para esse fim.

3.6.1.4 Receptor de GPS + microcontrolador + circuito

Componentes: CI receptor de GPS e microcontrolador ARM7.

Descrição: Ao invés de utilizar o computador ou o módulo G24 para controlar um receptor de GPS, é possível usar um microcontrolador e desenvolver um circuito para tal. No caso do ARM7, a programação pode ser feita em Java, diminuindo o tempo de desenvolvimento do software se comparado ao desenvolvimento em C ou Assembly necessário para outros microcontroladores.

Prós: Dessa forma seria obtido um produto pequeno e totalmente otimizado para as funções necessárias, pronto para implantação.

Contras: O desenvolvimento de um circuito para integrar o microcontrolador e o receptor de GPS demandaria muito mais tempo que as outras soluções, além de fugir do escopo do projeto. Talvez o projeto do circuito por si só fosse suficiente para desenvolver um trabalho de conclusão de curso.

3.6.2 Hardware escolhido

O grupo julgou a terceira solução (Receptor GPS USB + Notebook + Celular) como sendo a mais adequada aos nossos propósitos. Ela apresenta um nível de complexidade compatível com o escopo do nosso trabalho, e ao mesmo tempo agrega mais valor do que uma solução pronta do tipo "caixa-preta". Além disso, o receptor GPS USB é barato e de fácil obtenção, diferentemente dos componentes das outras soluções. Pesou contra essa proposta o fato de ela não produzir um produto apropriado para ser instalado em um ônibus e ser utilizado de fato. Porém, uma vez que o objetivo desse trabalho é provar o conceito proposto, e não projetar um produto viável comercialmente, esse problema não é impeditivo.

O receptor comprado é o modelo ND-100 da Globalsat. A seguir são mostradas as suas especificações. [2]

- Frequência L1: 1575,42 MHz
- Dimensões: 65.5 mm x 23 mm x 11 mm)
- Protocolo GPS: I NMEA 0183v3 GGA(1seg), GSA(1seg), GSV(5seg),RMC(1seg)
- Protocolo USB: USB 2.0

[2]Retirado do site do fabricante: http://www.globalsat.com.tw/products-page.php?menu=2&gs_en_product_id=2&gs_en_product_cnt_id=30&img_id=405 &product_cnt_folder=5

Figura 3.8: *Foto do receptor ND-100 da GlobalSat.*

- Taxa de transferência: 38400
- Taxa de aquisição:
  - Hot start: 1.5 segundos em média
  - Warm start: 32 segundos em média
  - Cold start: 34 segundos em média
  - Reaquisição: <1 segundos em média
- Sensibilidade: -161dBm
- Canais: 48

Os valores de hot, warm e cold start referem-se às seguintes situações:

- Hot start: Quando o receptor é inicializado, a última posição é conhecida, assim como o horário atual. Os dados de efemérides e o almanaque são válidos.
- Warm start: A úlima posição e o horário atual são conhecidas, os dados de efemérides são válidos, porém o almanaque não está disponível ou desatualizado (mais de 4 horas).
- Cold start: Não é conhecida a última posição, nem o horário atual.

4 METODOLOGIA

Foi utilizado o método linear sequencial clássico (também conhecido modelo em cascata) como metodologia para guiar a especificação e implementação desse projeto. Como é descrito em (PRESSMAN, 2005), esse método sugere a adoção de uma abordagem sistemática e linear para a concepção do Software, passando por etapas de análise, projeto, codificação, testes e suporte. A figura 4.1 representa essas fases e os principais documentos produzidos em cada uma delas.

Serão descritas a seguir as atividades típicas de um projeto utilizando o método linear e de que forma elas foram aplicadas no nosso projeto.

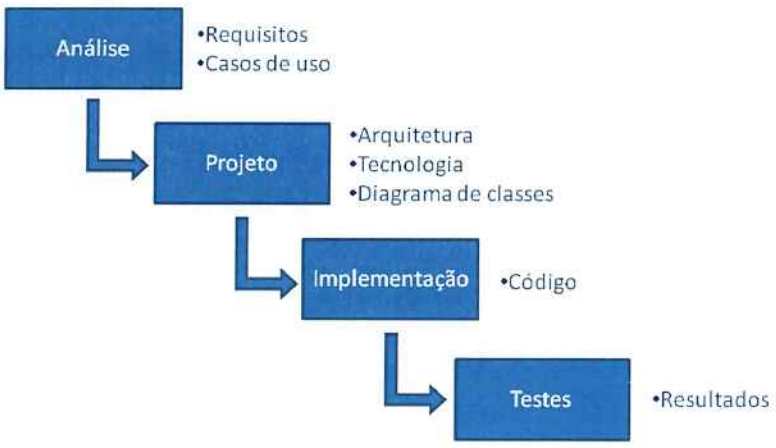

Figura 4.1: *Fases típicas do método linear sequencial e seus principais produtos.*

Análise de Requisitos Essa atividade envolve levantar e formalizar todas as características funcionais e não funcionais desejáveis do sistema a ser implementado. Itens como funções,

comportamento, performance e interface devem ser definidos de maneira objetiva, utilizando métodos objetivos para sua posterior avaliação.

Essa fase consiste em detectar um problema passível de ser resolvido utilizando a tecnologia da informação e explorar como um sistema pode fazê-lo. Como essa fase requer bom conhecimento acerca do negócio ao qual o sistema se destina, a pesquisa foi largamente focada nesse negócio, detectando soluções e métodos já existentes para problemas semelhantes.

Projeto Aqui são definidos aspectos de mais baixo nível do sistema. Nessa etapa o software deve ser definido em quatro perspectivas: estrutura de banco de dados, arquitetura de software, interfaces e algoritmos.

Codificação Trata-se da implementação propriamente dita do sistema. Quanto mais minuciosa tiver sido a fase de projeto, mais natural e mecânica será a codificação. É natural, porém, que alguns aspectos do sistema definidos nas atividades anteriores sejam alterados nessa fase, já que a codificação pode evidenciar aspectos antes não observados.

Testes Nessa fase são realizados testes para verificar se todos os requisitos estão preenchidos, bem como para encontrar defeitos de implementação. O teste deve focar tanto na lógica interna do software como em aspectos funcionais e sistêmicos.

No nosso projeto, a abordagem de testes foi ligeiramente modificada em relação ao modelo em cascata típico. Os testes unitários de cada classe foram feitos concomitantemente à codificação. Dessa forma, pode-se garantir a correta execução de funções internas das classes assim que elas são implementadas. Foi utilizado o *framework* JUnit para teste de classes Java, que permite a fácil implementação e execução automática de todos os testes unitários do projeto.

Permaneceram nessa fase testes que não são possíveis de serem executados durante a implementação, como testes sistêmicos e de performance.

Suporte Uma vez entregue e em uso, é natural que o software passe por alterações, seja por terem sido encontrados defeitos, seja para readequação das funcionalidades e interfaces do sistema. Como o nosso objetivo ao desenvolver o sistema não é colocá-lo em produção, essa etapa não foi considerada.

Essas atividades serão realizadas de acordo com o cronograma apresentado na tabela 4.1.

Tabela 4.1: Cronograma do Projeto

Atividade	Duraçao	Data Inicial	Data Final
Sistema de Monitoramento e Predição de Tempo de Chegada de Ônibus de Linha	285 dias	01/03/2010	10/12/2010
1. Análise e Projeto	126 dias	01/03/2010	05/07/2010
Análise de Requisitos do sistema	31 dias	01/03/2010	01/04/2010
Análise de Hardware	30 dias	01/04/2010	01/05/2010
Definição de Arquitetura do Sistema	65 dias	01/05/2010	05/07/2010
2. Desenvolvimento	153 dias	06/07/2010	06/12/2010
Desenvolvimento do Aplicativo Web e Simulador	148 dias	06/07/2010	01/12/2010
Desenvolvimento do software embarcado	87 dias	05/09/2010	01/12/2010
Integração	57 dias	10/10/2010	06/12/2010
3. Testes	30 dias	10/11/2010	10/12/2010
Testes de Validação Funcional	30 dias	10/11/2010	10/12/2010
Testes de Integração	20 dias	20/11/2010	10/12/2010
4. Aceitação	117 dias	10/08/2010	05/12/2010
Monografia	117 dias	10/08/2010	05/12/2010

5 PROJETO E IMPLEMENTAÇÃO

5.1 Módulo embarcado

5.1.0.1 Cálculo da distância percorrida

No método de estimativa de tempo escolhido, é necessário possuir a distância percorrida pelo ônibus em um dado intervalo de tempo. Há vários métodos para obter essa medida. Serão descritas a seguir as alternativas estudadas e testadas. Na seção 7 são apresentados os testes que justificam a escolha feita.

Cálculo pela posição Tendo as informações de latitude e longitude enviadas pelo receptor de GPS a cada segundo, pode-se calcular a distância percorrida aproximando cada um desses trechos do percurso para uma reta. Um bom método para efetuar esse cálculo é utilizar a função Haversine.

Segundo descrito em (RIDER; HUTCHINSON, 1943), a função haversine é uma função trigonométrica muito utilizada em navegações náuticas. Haversine é a abreviação de "*half versed sine*", ou metade do inverso do cosseno (Equação 5.1).

$$hav(A) = \frac{1}{2}(1 - cos(A)) \tag{5.1}$$

Essa função é usada para relacionar a posição de dois pontos com latitudes ϕ_1 e ϕ_2 pertencentes ao círculo maior de uma esfera de raio R e a distância d entre ambos através da fórmula 5.2 (NAVY, 1987).

$$hav(\frac{d}{R}) = hav(\Delta\phi) + cos(\phi_1)cos(\phi_2)hav(\delta\lambda) \tag{5.2}$$

A distância é obtida utilizando a função arcsen (fórmula 5.3). No caso de cálculo de distância entre pontos na terra, R é o raio médio do planeta (6371 Km).

$$d = 2R \arcsin \sqrt{hav(\frac{d}{R})} \tag{5.3}$$

Cálculo pela velocidade Sabendo que o receptor GPS é capaz de obter a velocidade com a qual o ônibus se desloca com razoável precisão, é possível utilizar esse dado para obter a distância percorrida. Dessa forma, assume-se que a variação da velocidade é constante entre uma medida e outra e obtém-se a fórmula 5.4 para o cálculo da distância, onde *d* é a distância, v_1 é a velocidade na medida anterior, v_2 é a velocidade na medida atual e t é o intervalo de tempo entre as duas medidas. Como o intervalo de tempo é 1 segundo, t = 1.

$$d = \frac{(v_2 + v_1)}{2t} \tag{5.4}$$

Perda de sinal Ambos os métodos devem ser capazes de manter o cálculo da distância percorrida mesmo que ocorra perda de sinal ou que o sistema receba sinais espúrios.

Para o método utilizando a fórmula de Haversine, considera-se que qualquer deslocamento que represente uma velocidade superior a 120 Km/h é proveniente de dados de posição incorretos. Nesse caso, o deslocamento obtido é ignorado e o deslocamento da medida anterior é utilizado.

O método pela velocidade comporta-se de maneira semelhante. Velocidades superiores a 120 Km/h são consideradas erradas e a velocidade da medida anterior é utilizada no cálculo.

Método utilizado Com base nos resultados obtidos nos testes, o segundo método (cálculo pela velocidade instantânea de cada medida) foi escolhido para calcular a distância já percorrida. Além de sua ligeiramente superior precisão, esse método exige um processamento muito menor.

5.1.0.2 Interface

Foi desenvolvida uma interface para o módulo embarcado para que se possa verificar o seu funcionamento. Ela apresenta na tela dados sobre a posição e a velocidade do veículo, bem como mostra as mensagens NMEA que estão sendo recebidas. Na figura 5.1 encontra-se uma imagem dessa interface.

Antes de começar a monitorar o veículo, deve-se informar em qual porta COM o receptor GPS está. Isso feito, o botão "Começar" faz com que as informações sejam exibidas na tela a

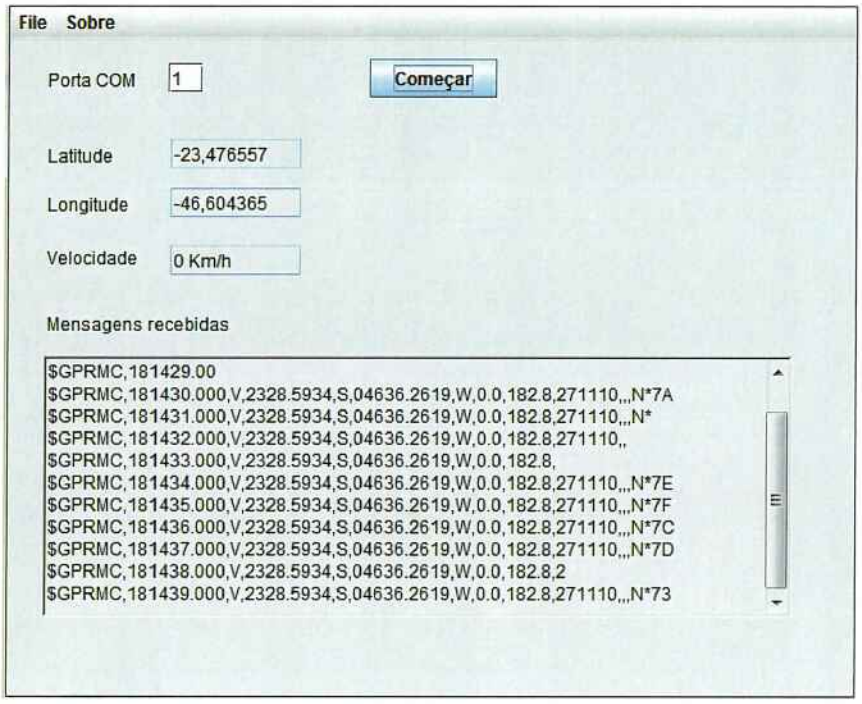

Figura 5.1: *Interface do módulo embarcado.*

cada segundo.

A interface foi implementada utilizando o pacote *Java Swing*. Para a montagem dos elementos gráficos da interface foi utilizado o *NetBeans GUI Builder*, que permite posicionar e redimensionar visualmente botões, campos de texto, *labels* etc.

5.2 Simulador de Ônibus

Para o desenvolvimento do sistema, o uso de um módulo embarcado em um veículo de verdade em movimento o tempo todo seria totalmente inviável.

Pensando nisso, foi definido o protocolo de comunicação entre o módulo embarcado e o servidor, a fim de criar um simulador.

O simulador possui uma interface para o usuário poder controlar a velocidade que o ônibus percorre o trecho. Além disso a interface também mostra o trecho que está sendo percorrido, bem como a porcentagem que ja foi percorrida.

A figura 5.2 mostra a interface do simulador.

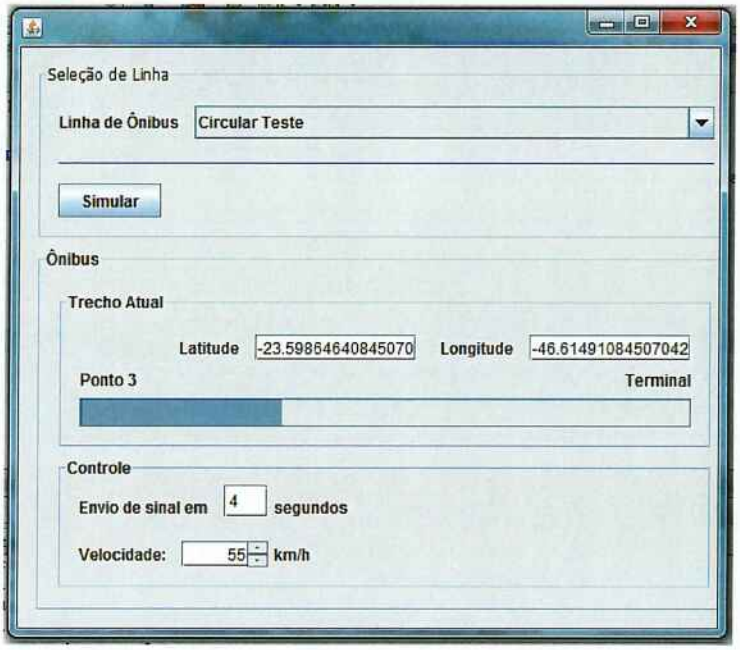

Figura 5.2: *Interface do Simulador de Ônibus.*

5.3 Módulo Web

5.3.1 Método de estimativa de tempo

Uma das características mais importantes do projeto é a estimativa do tempo de chegada do ônibus. Essa informação é a que realmente importa para o usuário do sistema de transporte, e a sua imprecisão pode frustrar o usuário. Por isso, é fundamental escolher um algoritmo capaz de fornecer dados minimamente confiáveis. Essa é uma tarefa complexa, já que o trânsito é um sistema extremamente difícil de ser modelado, em especial em cidades grandes com problemas de engarrafamento, como São Paulo.

Dos métodos pesquisados e apresentados brevemente na seção 2.1.1, foi implementado um algoritmo adaptado a partir do apresentado em (WEIGANG et al., 2005). Ele utiliza dados de viagens anteriores e parte do princípio que o GPS enviará dados sobre a velocidade com a qual o ônibus está transitando.

O princípio usado para estimar o tempo é o de que quanto menor for a distância entre o ônibus e a próxima parada, maior será a importância da velocidade instantânea do ônibus no cálculo da estimativa de tempo em relação à velocidade média, e vice-versa. Estima-se o tempo de chegada do ônibus dividindo o trajeto do mesmo em várias partes e calculando uma média ponderada entre as duas velocidades (instantânea e média), sendo que seus pesos são determinados pela distância até o ponto para o qual se quer a estimativa. A velocidade média utilizada é específica para cada intervalo e período do dia.

Sendo assim, para estimar o tempo de chegada de um ônibus a n pontos de distância, usa-se a fórmula 5.5.

$$Vc = \frac{Dp}{Dt} * Va + \frac{(Dt - Dp)}{Dt} * Vm \tag{5.5}$$

Nessa fórmula, a velocidade calculada (Vc) é a soma dos pesos da velocidade atual (Va) com a velocidade média (Vm). Os pesos são: as frações da distância percorrida (Dp) pela distância total (Dt) para a velocidade calculada e a distância que falta (Dt-Dp) divida sobre a distância total (Dt) para a velocidade media.

5.3.2 Processamento dos Dados do Módulo Embarcado

Ao receber os dados do módulo embarcado, o aplicativo realiza o seguinte procedimento:

- Se o ônibus chegou a um ponto:
  - Adiciona o tempo gasto no percurso deste último trecho ao histórico da média desse mesmo trecho.
- Atualiza a posição atual do ônibus para possível visualização de sua localização através da interface web.

5.3.3 Interface

O aplicativo web possui telas de cadastros para serem usadas por administradores do sistema, bem como telas de interação com usuários.

5.3.3.1 Cadastro de Linhas

O cadastro de linhas é feito utilizando a ferramenta do Google Maps. Em primeiro lugar são definidos todos os endereços dos pontos da linha, inclusive pontos sem ser de parada, para mapeamento correto da trajetória da linha. Cada ponto recebe também um nome.

O segundo passo é a geração da rota. Nesse passo a rota definida é exibida no mapa, para confirmação. A visualização da linha diretamente no mapa facilita o mapeamento e também fornece dados importantes como a distância de cada trecho e a latitude e longitude de cada ponto.

Em seguida são especificados o nome da linha e o número de ônibus que ela possui. Após este passo a linha pode ser criada corretamente.

Por se tratar de uma tela de uso interno, apenas para administradores do sistema, não foi gasto tempo do projeto para melhorar seu visual.

A tela de cadastro é exibida na figura 5.3

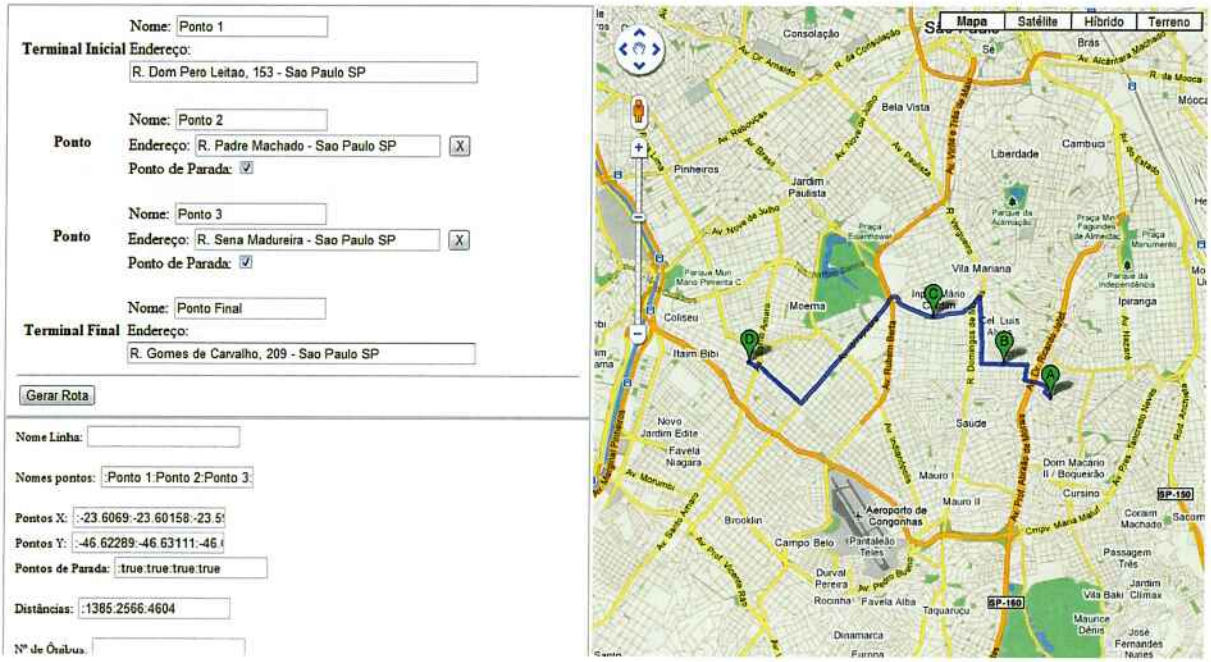

Figura 5.3: *Tela de Cadastro de Linha do Sistema.*

5.3.3.2 Portal Web

A segunda parte do aplicativo consiste do portal web para consulta das informações. É através desse site que os usuários interagem com o sistema, fazendo buscas relevantes a cada um.

O portal possui um menu para quatro telas:

Home A página inicial, onde são explicadas as funcionalidades do sistema.

Monitorar Linha A tela mais importante do site. É nela que as consultas são feitas, e o usuário pode usufruir dos dados obtidos. Nessa tela a linha desejada é exibida no mapa, e o usuário pode consultar o tempo estimado de espera em qualquer ponto dessa linha. A tela pode ser observada na figura 5.4.

Como Funciona Página feita para explicar aos interessados o funcionamento do sistema.

Sobre Um breve resumo sobre o projeto, os seus responsáveis e seus objetivos.

Figura 5.4: *Tela de consulta do Sistema.*

6 TESTES E AVALIAÇÃO

6.1 Testes e avaliação

Nessa seção serão descritos os testes realizados para validar o funcionamento do sistema, juntamente com seus resultados. Os testes estão divididos em três partes: testes do módulo embarcado, testes do módulo web e testes de integração.

6.1.1 Testes do módulo embarcado

6.1.1.1 Teste do cálculo da distância

Como descrito na seção 5.1.0.1, consideramos a utilização de dois métodos para o cálculo da distância percorrida, um utilizando a velocidade instantânea do veículo a cada medida e outro utilizando a diferença entre as posições de duas medidas consecutivas. São mostrados nas tabelas 6.1 e 6.2 os resultados de testes com ambos os métodos.

Os testes foram realizados da seguinte maneira: inicialmente foram determinados dois percursos contendo curvas, semáforos e pontos de tráfego pesado, a fim de simular um percurso típico de um ônibus em São Paulo. Esse trajeto foi percorrido em período de tráfego normal e a distância percorrida foi medida utilizando o hodômetro parcial do carro, cuja menor medida é a centena de metros.

Tabela 6.1: Resultado dos testes de cálculo de distância percorrida (Percurso 1)

Amostra	Hodômetro	Cálculo p/ velocidade	Cálculo p/ posição
1	4,1	3,8	3,8
2	4,1	3,9	4,5
3	4,1	4,0	4,2

Na tabela 6.3 verificamos os erros médios obtidos em cada percurso e a média geral.

Tabela 6.2: Resultado dos testes de cálculo de distância percorrida (Percurso 2)

Amostra	Hodômetro	Cálculo p/ velocidade	Cálculo p/ posição
1	5,3	4,7	5,5
2	5,3	4,9	4,4
3	5,3	5	5,6

Tabela 6.3: Erros médios no cálculo das distâncias percorridas

Método	Erro médio: percurso 1	Erro médio: percurso 2	Erro médio
Cálculo p/ velocidade	4,9%	8,2%	6,6%
Cálculo p/ posição	6,5%	8,9%	7,7%

6.1.2 Testes do algoritmo de estimativa de tempo

Foram realizados diversos testes para análise da precisão da estimativa de tempo.

Para a realização dos testes foi utilizado o simulador de ônibus, que gerou valores de velocidade aleatoriamente em uma distribuição normal com média no valor estipulado pelo usuário.

Os testes levaram em consideração o valor estimado, informado pelo sistema, de um ponto ate o próximo, em comparação ao valor real em segundos que o ônibus de fato demorou para chegar ao ponto. Foram realizadas 100 iterações, e calculados todos os desvios.

Os resultados são apresentados na tabela 6.4.

Tabela 6.4: Resultados do Cálculo de Estimativa

Iterações	Média do Desvio	Desvio Padrão
100	36,83 segundos	41,43 segundos

Os resultados foram bastante aceitáveis, considerando que as estimativas possuem um erro de 41,43 segundos para mais ou para menos.

6.1.3 Testes de integração

Tendo garantido o correto funcionamento dos dois módulos, passamos para a verificação da integração de ambos e do comportamento do sistema como um todo.

6.1.3.1 Teste simples de funcionamento

O primeiro teste de integração feito consistiu em verificar a comunicação entre os dois módulos. Usando um PC com o GPS e o celular (utilizado como modem GPRS) conectados, tentamos enviar os dados para um outro PC no qual rodava aplicação web. Para testar o comportamento do sistema na ocorrência de problemas na rede GSM que atrasassem o envio da mensagem, simulamos essa situação forçando na aplicação um tempo de resposta de cinco segundos.

Resultados Foi possível enviar as mensagens sem qualquer dificuldade. A aplicação web recebeu os dados e conseguiu marcar o ponto onde o outro PC estava no mapa do *Google Maps*. O tempo médio entre o envio da mensagem e o recebimento da resposta foi de 0,8 segundos.

O sistema também se comportou corretamente nas situações onde o tempo de resposta foi maior. Não houve perda de dados do receptor GPS enquanto o sistema aguardava a resposta do servidor, já que os dados enviados pelo mesmo são acumulados em um *buffer*.

O envio de cinco mensagens exigiu 6 Kb de *upload* e 4 Kb de *download*, valores suficientemente baixos para manter bom desempenho e baixo custo na transferência de dados.

7 CONSIDERAÇÕES FINAIS

7.1 Conclusão

Os resultados obtidos foram positivos. A escolha da arquitetura e da interface entre os dois grandes módulos do projeto se mostrou muito adequada ao propósito do projeto. O desenvolvimento de ambos foi completamente independente e não foi encontrada qualquer dificuldade na integração dos mesmos. As tecnologias utilizadas no projeto também apresentaram níveis adequados de confiabilidade e exatidão. Como resultado, conseguimos alcançar utilizando o simulador previsões com erros na casa de minuto, erro considerado baixo se for levada em conta a imprevisibilidade do trânsito. Utilizando o módulo embarcado para aquisição de dados de posição foram alcançados resultados semelhantes, porém com eventuais erros maiores em decorrência do fato do sistema não conseguir determinar em algumas situações que o ônibus passou por um ponto.

Apesar do erro ter sido suficientemente baixo, um aspecto que foi negligenciado durante o desenvolvimento do trabalho é o fato do erro da previsão do tempo para mais é muito mais nocivo do que o erro para menos. No primeiro caso, o usuário perderia o ônibus; no segundo, esperaria por um pouco mais de tempo no ponto. Esse aspecto deveria ter sido levado em conta, mesmo que isso resultasse em erros maiores para menos.

7.2 Considerações Pessoais

Após onze meses de pesquisa e trabalho, o saldo do projeto é definitivamente positivo. Foi uma forma de colocarmos à prova todos os conceitos aprendidos nos cinco anos de faculdade, desde os fundamentos mais básicos de programação à decisões de arquitetura de software e gerenciamento de projetos.

No âmbito técnico, o desenvolvimento do projeto nos exigiu o contato com tecnologias que até então eram desconhecidas, tais como o GPS e o GPRS. O uso de tecnologias por

nós já dominadas, como o Java, serviu como forma de consolidação e amadurecimento da técnica e do conhecimento. Especificar o projeto desde a definição de requisitos também foi um exercício de grande valia para a nossa formação, já que essa talvez seja uma das habilidades mais importantes do engenheiro da computação hoje. Em momento nenhum nos faltou conhecimento ou fundamento teórico, o que demonstra a solidez do conhecimento absorvido na escola.

No aspecto gerencial, o trabalho de conclusão foi provavelmente o maior projeto do qual os integrantes do grupo participaram de todas as fases do processo até agora. Pode-se observar como é importante planejar e documentar cada passo para evitar atrasos, retrabalhos e surpresas desagradáveis. Para isso, o embasamento teórico obtido durante o curso foi fundamental. O grupo teve plena capacidade de avaliar o projeto e definir qual seria a melhor forma de conduzi-lo, utilizando metodologias bem estabelecidas.

7.3 Trabalhos Futuros

Como foi apontado na seção 3.6, o sistema desenvolvido para o módulo embarcado não é viável comercialmente. Portanto, um próximo passo na direção de um produto acabado e comercializável seria projetar um hardware que pudesse ser colocado em uma caixa e que fosse capaz de suportar as condições físicas às quais seria submetido dentro de um ônibus em movimento (vibração, calor, choques etc).

O algoritmo de estimativa de tempo pode passar a ter mais dados para realizar a sua análise. Ele poderia receber informações de *webservices* sobre as condições climáticas no momento ou dados sobre alagamentos da Central de Gerenciamento de Emergências da prefeitura de São Paulo.

Apesar de termos testado o sistema em situações reais, não é possível prever como o sistema se comportaria quando submetido ao uso. Esse uso o colocaria em situações de trânsito não previstas no projeto e implementação do sistema, nas quais não se sabe qual seria o desempenho do algoritmo de estimativa do tempo. A implementação de uma prova de conceito em um ônibus prestando serviço evidenciaria essas situações, que poderiam resultar em novas versões do algoritmo.

Há também a possibilidade de adicionar novas funcionalidades ao sistema. O sistema poderia, por exemplo, disparar o envio de uma mensagem de texto SMS para o usuário quando o ônibus estivesse a um certo tempo de chegar ao ponto previamente cadastrado pelo usuário. Isso aumentaria o número de possíveis usuários do sistema, uma vez que não

seria mais necessário possuir um *smartphone* para receber notificações sem estar na frente do computador.

REFERÊNCIAS BIBLIOGRÁFICAS

EL-RABBANY, A. *Introduction to GPS: the Global Positioning System*. Artech House, 2002. (Artech House mobile communications series). ISBN 9781580531832. Disponível em: <http://books.google.com/books?id=U2JmghrrB8cC>.

KALMAN, R. et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, Citeseer, v. 82, n. 1, p. 35–45, 1960.

LIN, W.; ZENG, J. Experimental study of real-time bus arrival time prediction with GPS data. *Transportation Research Record: Journal of the Transportation Research Board*, Trans Res Board, v. 1666, n. -1, p. 101–109, 1999. ISSN 0361-1981.

LIU, H. *Software performance and scalability: a quantitative approach*. [S.l.]: John Wiley & Sons, 2009. (Quantitative software engineering series). ISBN 9780470462539.

NAVY, R. *Admiralty manual of navigation*. [S.l.]: Stationery Office, 1987. (BR Series). ISBN 9780117728806.

PADMANABAN, R.; VANAJAKSHI, L.; SUBRAMANIAN, S. Estimation of bus travel time incorporating dwell time for APTS applications. **Intelligent Vehicles Symposium, 2009 IEEE**. [S.l.], 2009. p. 955–959. ISSN 1931-0587.

PRESSMAN, R. *Software engineering: a practitioner's approach*. [S.l.]: McGraw-Hill, 2005. (Mcgraw Hill: higher Education). ISBN 9780073019338.

RIDER, P.; HUTCHINSON, C. *Navigational trigonometry*. [S.l.]: The Macmillan company, 1943.

ROCHA, J. *GPS - Uma Abordagem Prática - Aplicações Náuticas, Terrestres e de Geoprocessamento*. [S.l.]: Jose Antonio Manso, 2003. ISBN 9788574095547.

SEURRE, E.; SAVELLI, P.; PIETRI, P. *GPRS for mobile Internet*. Artech House, 2003. (Artech House mobile communications series). ISBN 9781580536004. Disponível em: <http://books.google.com/books?id=rkx1gF-3KRgC>.

WEIGANG, L.; KOENDJBIHARIE, W.; YAMASHITA, Y.; MACIVER, A. Algorithms for estimating bus arrival times using GPS data. **Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on**. [S.l.], 2005. p. 868–873. ISBN 0780373898.