

**LIGA DE ENSINO DO RIO GRANDE DO NORTE  
FACULDADE NATALENSE PARA O DESENVOLVIMENTO  
DO RIO GRANDE DO NORTE**

**Helton Jorge Dias Cordeiro**

**SISTEMA DE ESTIMATIVA DE TEMPO DE ESPERA**

**SISTEMA DE ESTIMATIVA DE TEMPO DE ESPERA**

Curso de Pós-Graduação em Desenvolvimento de Sistemas Corporativos da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte - como requisito obrigatório para obtenção do grau de Especialista em Desenvolvimento de Sistemas Corporativos.

Orientador:

Ulisses Teodoro Neto, MSc.

**NATAL  
2008**

HELTON JORGE DIAS CORDEIRO



## SISTEMA DE ESTIMATIVA DE TEMPO DE ESPERA

## SISTEMA DE ESTIMATIVA DE TEMPO DE ESPERA

Monografia apresentada ao Curso de Pós Graduação em Desenvolvimento de Sistemas Corporativos da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte como requisito obrigatório para obtenção do grau de Especialista em Desenvolvimento de Sistemas Corporativos.

Orientador:  
Ulisses Telemaco Neto, MSc.

Natal  
2008



M  
004.43  
C 819.5  
2008  
LX. 01

Catálogo na Publicação – Biblioteca da FARN  
Setor de Processos Técnicos

Cordeiro, Helton Jorge Dias

Sistema de estimativa de tempo de espera / Helton Jorge Dias  
Cordeiro. – Natal: FARN, 2008.  
42f.

Orientador: Ulisses Telemaco Neto (Msc.)

Monografia (Especialização em Desenvolvimento de Sistemas  
Corporativos) – Faculdade Natalense para o Desenvolvimento do Rio  
Grande do Norte.

1. Desenvolvimento de Sistema – Monografia. 2. Plataforma Java –  
Monografia. 3. Sistema de Estimativa de Tempo de Espera – Monografia.  
4. Transporte Coletivo – Monografia. I. Título.

RN/FARN/BC

CDU 004.43

HELTON JORGE DIAS CORDEIRO

O presente trabalho é uma proposta de desenvolvimento e aplicação de um sistema de estimativa de tempo de espera que auxilia na tomada de decisão do usuário de transporte coletivo. É apresentado um breve histórico do surgimento do transporte coletivo no Brasil e a problemática surgida aos dias atuais de manter-se o fluxo de ir e vir das cidades brasileiras no trânsito caótico das grandes cidades. São descritas as principais tecnologias utilizadas na criação deste sistema (GPS, Webservice, JME) e, por fim, a implementação do SETS (Sistema de Estimativa de Tempo de Espera) apresentando os resultados obtidos.

## SISTEMA DE ESTIMATIVA DE TEMPO DE ESPERA

Palavras-Chave: Transporte Coletivo, WebService, GPS, JME.

Monografia apresentada ao Curso de Especialização em Desenvolvimento de Sistemas Corporativos da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte, aprovada em \_\_\_\_ de \_\_\_\_ de \_\_\_\_\_, pela banca examinadora composta pelos seguintes membros:

---

Ulisses Telemaco Neto, MSc.  
Orientador

---

Gleydson Lima, MSc.  
Membro Examinador

## RESUMO

O presente trabalho é uma proposta de desenvolvimento e aplicação de um sistema de estimativa de tempo de espera que auxilia na tomada de decisão do usuário de transporte coletivo. É apresentado um breve histórico do surgimento do transporte coletivo no Brasil e a problemática surgida nos dias atuais de manter-se o direito de ir e vir dos cidadãos brasileiros no trânsito caótico das grandes cidades. São descritas as principais tecnologias utilizadas na criação deste sistema (GPS, Webservice, JME) e, por fim, a implementação do SETE (Sistema de Estimativa de Tempo de Espera) apresentando os seus diagramas, subsistemas integrados o seu funcionamento e os resultados obtidos com as ferramentas e tecnologias empregadas.

*Key words: Public Transportation, Web Service, GPS, JME.*

**Palavras-Chave:** Transporte Coletivo. Web Service. GPS. JME.

## ABSTRACT

This work is a proposal for developing and implementing a system for estimating the waiting time that helps in decision-making by the user of public transportation. It presented a brief history of the emergence of public transportation in Brazil and the issue has emerged today to maintain the right to come and go for Brazilian citizens in the chaotic traffic of large cities. They described the main technologies used in creating this system (GPS, Web Service, JME) and, finally, the implementation of SETE (Estimated Time to Wait System) by presenting their diagrams, subsystems integrated its operation and the results obtained with the tools and technologies employed.

**Key words:** Public Transportation. Web Service. GPS. JME.

## SUMÁRIO

|                                                         |           |
|---------------------------------------------------------|-----------|
| <b>INTRODUÇÃO.....</b>                                  | <b>1</b>  |
| <b>1. GPS.....</b>                                      | <b>2</b>  |
| 1.1 SEGMENTO ESPACIAL.....                              | 3         |
| 1.2 SEGMENTO TERRESTRE.....                             | 3         |
| 1.2.1 Sub-Segmento de Controle.....                     | 3         |
| 1.2.2 Sub-Segmento de Usuário.....                      | 4         |
| <b>2. JAVA ME.....</b>                                  | <b>5</b>  |
| <b>3. WEB SERVICE.....</b>                              | <b>6</b>  |
| <b>4. O TRANSPORTE COLETIVO.....</b>                    | <b>7</b>  |
| <b>5. SISTEMA DE ESTIMATIVA DE TEMPO DE ESPERA.....</b> | <b>9</b>  |
| 5.1 SUBSISTEMAS.....                                    | 10        |
| 5.2 FUNCIONAMENTO DO SETE.....                          | 10        |
| 5.3 DIAGRAMA DE CASO DE USO.....                        | 11        |
| 5.4 MODELO CONCEITUAL.....                              | 12        |
| 5.5 DIAGRAMA DE CLASSES.....                            | 13        |
| 5.6 DIAGRAMA DE PACOTES.....                            | 14        |
| 5.7 IMPLEMENTAÇÃO.....                                  | 14        |
| 5.7.1 A classe “Gestor”.....                            | 14        |
| 5.7.2 A classe “SeteWebService”.....                    | 18        |
| 5.7.3 Implementação do Cliente JME.....                 | 19        |
| <b>6. CONCLUSÕES.....</b>                               | <b>21</b> |
| <b>REFERÊNCIAS.....</b>                                 | <b>22</b> |
| <b>ANEXO A – CLASSES DO PACOTE “ENTIDADE”.....</b>      | <b>25</b> |
| <b>ANEXO B – CLASSES DO PACOTE “NEGÓCIO”.....</b>       | <b>29</b> |
| <b>ANEXO C – WEB SERVICE.....</b>                       | <b>31</b> |



## INTRODUÇÃO

Atualmente tem-se verificado um aumento do número de automóveis particulares nas grandes cidades causando grandes congestionamentos e atrasos na vida da população. Como proposto por Bicalho (1998), uma das soluções seria conscientizar a população a utilizar outras formas de transporte que não automóvel particular. Existe atualmente uma grande preocupação por parte dos especialistas da área em entender a problemática dos congestionamentos nas grandes cidades e buscar soluções para a redução deste problema. Uma das grandes armas para o combate ao desenfreado crescimento da frota de carros particulares nas grandes cidades está na conscientização da população em buscar cada vez mais o uso do transporte coletivo. Através de artifícios como o uso de GPS, ônibus de porta baixa, faixa exclusiva para ônibus nas vias, as empresas e órgãos do setor de transporte e trânsito vêm buscando atrair a população ao uso do transporte coletivo. Em vista do grande avanço tecnológico no setor e da urgente necessidade de soluções que busquem desafogar o trânsito nas grandes cidades, este trabalho vem fornecer às empresas do setor de trânsito, à comunidade de desenvolvedores e aos órgãos do setor de transporte e trânsito uma iniciativa de tentar atrair a população das grandes cidades ao uso deste tipo de transporte, buscando através de um sistema de informação de baixo custo e alta acessibilidade, facilitar a tomada de decisão do usuário de ônibus. O referido sistema tem como propósito fornecer o tempo estimado de espera do ônibus que o usuário deseja consultar a partir do uso do seu dispositivo móvel (Celular, Smartphone, PDA, etc.) e faz uso de tecnologias emergentes no mercado atual como GPS, Webservice e JME.

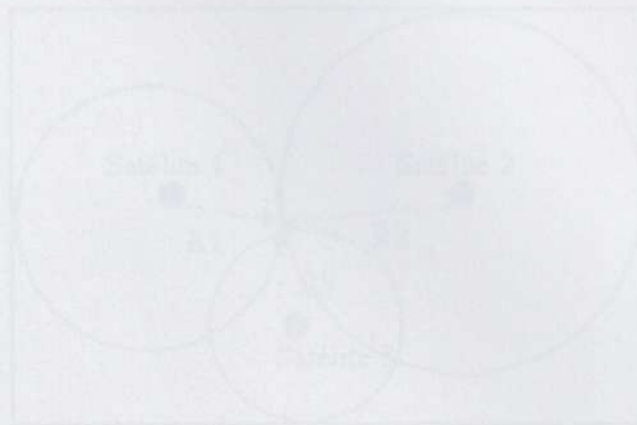


Figura 1: O Princípio Básico do GPS  
Fonte: Farnes M. Johnson (2000)

O GPS é formado por dois segmentos. O segmento espacial inclui 24 satélites e o segmento terrestre pode ser subdividido em dois sub-segmentos, de Controle e de Usuário (Johnson, 1996).

### SEGMENTO ESPACIAL

Formado pelos satélites que orbitam em órbita, consiste de 24 satélites, em 6 órbitas, com 4 satélites em cada órbita. Os satélites possuem a órbita em torno da Terra a uma altitude de aproximadamente 11.000 milhas aéreas. Cada satélite transmite

## 1. GPS

*Latitude é o oeste quando agente está no leste e vice-versa.*

*Millôr Fernandes*

O GPS (*Global Positioning System*, GPS) trata-se de um instrumento de navegação por satélite, criado para fins militares que tem como principal propósito prover sinais que permitam que qualquer usuário no planeta, equipado com um receptor GPS, possa calcular corretamente e em tempo real sua posição geográfica (longitude, latitude e altitude). O padrão de uso para os usuários comuns (não militares) é o Sistema de Posicionamento Padrão (*Standard Positioning System*, SPS). Está disponível para qualquer usuário no mundo, em todas as condições climáticas e de forma contínua.

A tecnologia é recente: a primeira vez que o GPS foi usado efetivamente foi na Guerra do Golfo, em 90 e 91. Alguns anos depois, o aparelho foi liberado para o uso de civis e passou a ser comercializado.

O princípio básico de funcionamento do GPS se baseia no princípio da triangulação, segundo o qual 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 referência. O sistema de referência utilizado pelo sistema GPS é o WGS (WGS-72 até 1986 e WGS-84 a partir de 1987).

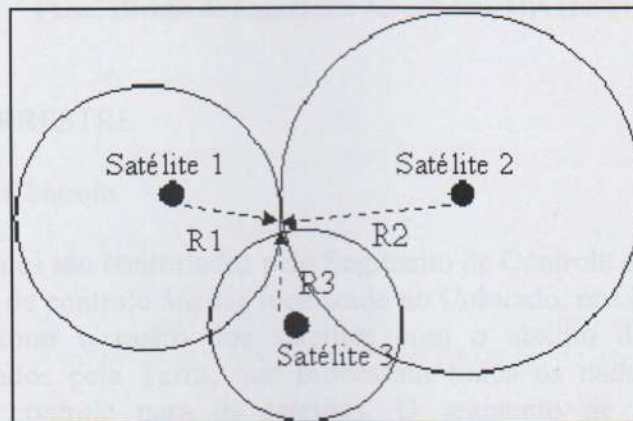


Figura 1: O Princípio Básico do GPS  
Fonte: Thienne M. Johnson (2008)

O GPS é formado por dois segmentos. O segmento espacial inclui 24 satélites e o segmento terrestre pode ser subdividido em dois sub-segmentos, de Controle e de Usuário (Mohamed-Slim, 1996).

### 1.1 SEGMENTO ESPACIAL

Formado pelos satélites que estão em órbita, consiste de 24 satélites, em 6 órbitas diferentes com 4 satélites em cada órbita. Os satélites percorrem a órbita em torno da Terra a cada 12 horas, a uma altitude de aproximadamente 11.000 milhas náuticas. Cada satélite tem

28 graus de visualização sobre a Terra e estão inclinados 55 graus em relação à linha do Equador. Por alguns momentos, vários pontos da Terra são visualizados ao mesmo tempo por 6 a 10 satélites que focalizam a mesma área. Isto fornece redundância, pois apenas 4 satélites são necessários para uma determinação tridimensional de posição. Na Figura 2, é mostrado a distribuição de cada órbita dos satélites em relação à Terra.

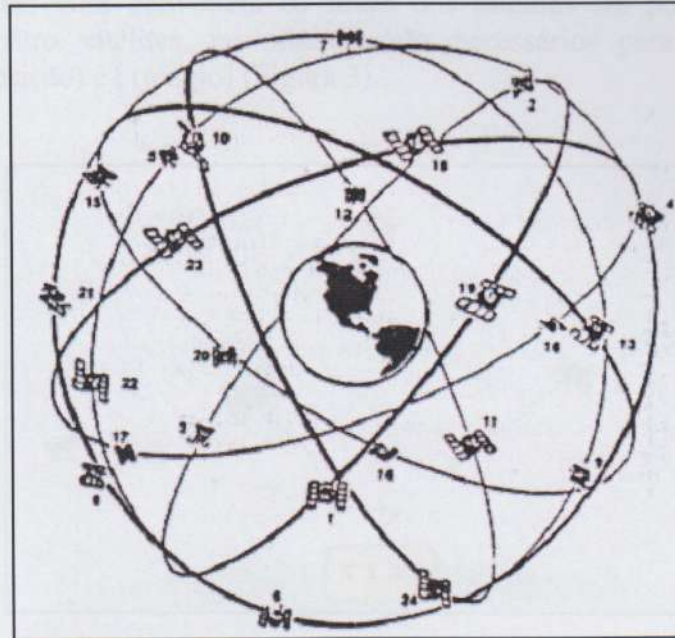


Figura 2: Distribuição de órbita  
Fonte: Divisão de Engenharia Aeronáutica, ITA (2008)

## 1.2 SEGMENTO TERRESTRE

### 1.2.1 Sub-segmento de Controle

Todos os 24 satélites são controlados pelo Segmento de Controle em terra. Este controle é feito por uma estação de controle *Master* localizado no Colorado, nos Estados Unidos. Ela é responsável por monitorar o rastro dos satélites com o auxílio de cinco estações de monitoramento espalhadas pela Terra, que processam todos os dados e então enviam a correção e sinais de controle para os satélites. O segmento de controle monitora o desempenho total do sistema, corrige posições do satélite e reprograma o sistema com o padrão necessário. A localização de cada estação de monitoramento oferece um monitoramento constante de cada satélite. Todas as estações de monitoramento trilham os satélites, determinam seu desempenho operacional, checam os parâmetros e passam estas informações para a estação *Master*. A estação *Master* pode então determinar os parâmetros de órbita de um satélite e transferir dados de correção para o mesmo satélite. A determinação desse número de satélites circulando o globo, mais os planos de órbita dos satélites, junto com a estrutura de comando e controle, faz com que o GPS assegure que um número mínimo de quatro satélites sempre estará disponível para oferecer, em qualquer lugar da superfície da Terra, uma posição precisa de determinado objeto (através de um receptor - um *GPS Receiver*). O mesmo aspecto de precisão espacial com o mínimo de recurso necessário, oferecendo uma informação segura, é assegurado na localização de cada estação de monitoramento. Também é assegurada a posição exata de cada satélite, supervisionando-o a

todo o momento. Estes dois fatores são necessários para assegurar uma precisão tridimensional na determinação da posição – localização geográfica.

### 1.2.2 Sub-segmento de Usuário

O Segmento do Usuário do GPS consiste dos receptores GPS e a comunidade de usuários. Os receptores GPS convertem os sinais dos satélites em posição, velocidade, e tempo estimado. Quatro satélites, no mínimo, são necessários para calcular as quatro dimensões: x, y, z (posição) e t (tempo) (Figura 3).

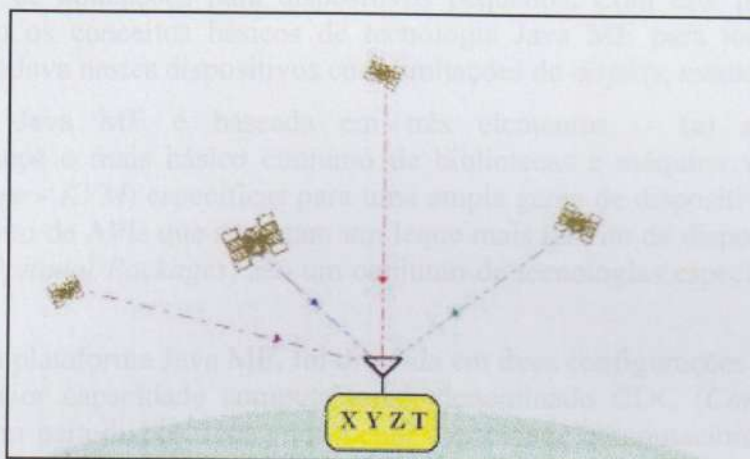


Figura 3: Recepção do Sinal  
Fonte: DANA (2000)

Os Receptores GPS são usados para orientação à navegação, posicionamento, disseminação do tempo, e outros recursos. A navegação é uma das funções do GPS, e é usada por usuários de aeronaves, navios, veículos, e por indivíduos que usam o receptor portátil. Também é usado por observatórios astronômicos, por empresas de telecomunicações e laboratórios científicos; podendo configurar sinais controlados por frequências específicas para um determinado propósito, estudo ou experiência.

O GPS permite a qualquer usuário saber a sua localização, velocidade e tempo, 24 horas por dia, sob quaisquer condições atmosféricas e em qualquer ponto do globo terrestre. Como o nome sugere, o GPS é um sistema de abrangência global. Esse sistema tem facilitado todas as atividades que necessitam de posicionamento, fazendo que algumas concepções antigas pudessem ser postas em prática. Um exemplo claro disso é o que vem ocorrendo com o desenvolvimento da agricultura de precisão, um conceito estabelecido por volta de 1929, que só agora tem sido posto em prática, graças à integração de várias geotecnologias, dentre elas o GPS.

## 2. JAVA ME

Java Platform, Micro Edition (Java ME) é uma coleção de tecnologias e especificações desenvolvidas com o intuito de criar uma plataforma que se ajuste às exigências dos dispositivos móveis, tais como celulares, PDAs, dispositivos embarcados, e avançados dispositivos móveis. É uma coleção de tecnologias e especificações que podem ser combinadas para criar um completo ambiente de execução Java especificamente para se adequar às exigências de um determinado mercado ou dispositivo.

A tecnologia Java ME foi originalmente criada para lidar com as exigências associadas ao desenvolvimento de aplicações para dispositivos pequenos. Com este propósito a *Sun Microsystems* definiu os conceitos básicos de tecnologia Java ME para tornar possível a criação de aplicações Java nestes dispositivos com limitações de *display*, memória e energia.

A tecnologia Java ME é baseada em três elementos: - (a) a configuração (*Configuration*) fornece o mais básico conjunto de bibliotecas e máquina virtual (*Kernel-based Virtual Machine - KVM*) específicas para uma ampla gama de dispositivos; (b) o perfil (*Profile*) é um conjunto de APIs que suportam um leque mais restrito de dispositivos, e (c) os pacotes opcionais (*Optional Packages*) são um conjunto de tecnologias específicas para cada API.

Com o tempo a plataforma Java ME, foi dividida em duas configurações base, uma para dispositivos com maior capacidade computacional, denominado CDC (*Connected Device Configuration*) e outra para dispositivos com menor capacidade computacional, denominada CLDC (*Connected Limited Device Configuration*).

O MIDP é o mais conhecido perfil do Java ME baseado na CLDC e KVM, é amplamente utilizado por todo o mundo, principalmente no desenvolvimento de aplicativos para celulares, PDAs e outros dispositivos móveis. A figura abaixo representa uma visão geral dos componentes da tecnologia Java ME e como se relaciona com as outras tecnologias Java (Figura 4).

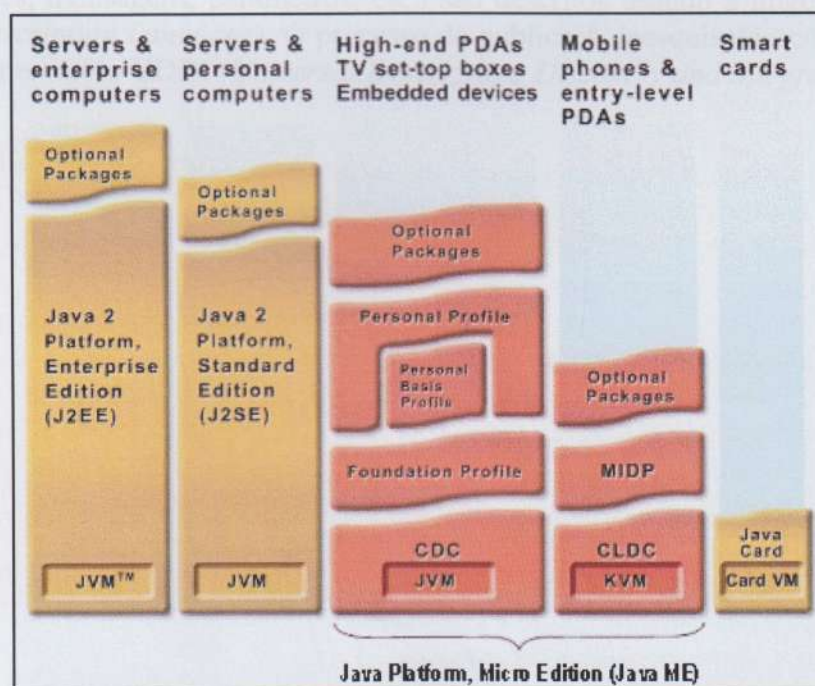


Figura 4 – Tecnologias Java  
Fonte: Sun Microsystems (2008)

### 3. WEB SERVICE

Com a disseminação da computação distribuída, os *Web Services* surgiram com o propósito de conectar diferentes sistemas computacionais, estejam os dados em estações de trabalho, servidores ou até mesmo em mainframes. Em resumo, a tecnologia permite que dispositivos conectados à Internet troquem mensagens entre si, sem a intervenção direta dos usuários.

Um *WebService* é um componente que possui suas funcionalidades acessíveis pela rede através de troca de mensagens baseadas em XML (*eXtensible Markup Language*). A disponibilização das operações e a descrição do serviço também ocorrem através do padrão XML. O arquivo descritor do serviço possui todas as informações necessárias para que outros componentes possam interagir com o serviço, incluindo o formato das mensagens para as chamadas aos métodos do serviço, protocolos de comunicação e as formas de localização do serviço. Um dos maiores benefícios dessa interface é a abstração dos detalhes de implementação do serviço, permitindo que seja acessado independente da plataforma de hardware ou software na qual foi implementado. Como as mensagens trocadas para a comunicação são baseadas no padrão XML, também tem-se a flexibilidade com relação à linguagem de programação tanto na implementação do serviço quanto no componente que acessará o *WebService*. Estas características permitem e motivam a implementação de aplicações Web baseadas em *WebService* por torná-las fracamente acopladas com as outras partes do código da aplicação. Com isso, as aplicações adquirem uma arquitetura componentizada e tornam-se flexíveis com relação às várias plataformas disponíveis no mercado.

Um *Web Service* geralmente é implementado para disponibilizar uma determinada funcionalidade autocontida visando a reusabilidade do *WebService* e a interoperabilidade com outros sistemas. As chamadas às operações, incluindo os parâmetros de entrada/saída, são codificadas no protocolo SOAP (*Simple Object Access Protocol*, baseado em XML). Os serviços (operações, mensagens, parâmetros, etc.) são descritos usando a linguagem WSDL (*Web Services Description Language*). O processo de publicação/pesquisa/descoberta de *Web Services* utiliza o protocolo UDDI (*Universal Description, Discovery and Integration*).

#### 4. O TRANSPORTE COLETIVO

A implantação do transporte urbano começou com a chegada da Família Real Portuguesa ao País, em 1808, há exatos 200 anos. Foi D. João VI quem promoveu o primeiro serviço de transporte coletivo brasileiro. Aconteceu por ocasião das constantes viagens da Corte para a fazenda de Santa Cruz, que ficava distante uns 50 quilômetros da cidade do Rio de Janeiro. Um empregado da corte, chamado Sebastião Fábregas Surigué, conseguiu a concessão junto ao Príncipe Regente para explorar um serviço de carruagens que transportaria pessoas até a localidade para que participassem da cerimônia do "beija-mão", solenidade em que súditos se dirigiam ao monarca para beijar sua mão, mostrando submissão e apreço pelo rei. A concessão perduraria por cinco anos, firmada em decreto de 18 de agosto de 1827, data à qual se pode atribuir o nascimento do transporte coletivo no Brasil. Os deslocamentos tinham um caráter praticamente intermunicipal, mas foi a primeira iniciativa de se organizar o transporte de pessoas seguindo itinerários e horários definidos, a partir de uma área urbana, mediante cobrança de passagem. A partir daí, devido à abertura dos portos, surgem as primeiras empresas de ônibus, também chamados de "gôndolas", na cidade do Rio de Janeiro, entre 1837 e 1860.

A situação atual do transporte urbano no Brasil vem sendo muito discutida por diversos autores, entre eles: Associação Nacional dos Transportes Públicos - ANTP (1998), Cedano (1999), Junqueira & Pinto (1999) e Pereira et al. (1999). Aspectos relacionados à má qualidade, baixa confiabilidade, pouca flexibilidade, ineficiência do transporte urbano, tarifas elevadas, comparativamente à baixa qualidade do serviço oferecido, a capacidade e/ou disposição de pagamento da população, e a queda da produtividade das empresas operadoras de transporte regulamentado por ônibus, entre outros, são alguns dos pontos que têm recebido a atenção dos pesquisadores e vêm sendo objeto de análises, especialmente quanto às tendências que se configuram para o futuro do transporte coletivo urbano, visto que o setor vem perdendo prestígio junto à opinião pública, à classe política, às entidades civis e aos próprios cidadãos que dele dependem.

Atualmente a população está trocando o transporte coletivo por outras alternativas (a pé ou outro veículo), de acordo com o perfil do transporte apresentado pelo Ministério das Cidades e pela Associação Nacional de Transportes Urbanos. A pesquisa realizada pelos dois órgãos aponta para um "índice de passageiros por quilômetro", correspondente ao índice de produtividade, considerado baixo (em torno de 2). O relatório traz, no entanto, os dados da produtividade em anos anteriores. A explicação, segundo o diretor do Departamento de Cidadania e Inclusão Social do Ministério das Cidades, Luiz Carlos Bertotto, está não só nos preços e qualidade do transporte coletivo, que levaria os passageiros a andarem a pé. Disse ele que, por outro lado, parte dos usuários do transporte público também vem migrando para o uso de automóveis. Bertotto informou que a migração é uma tendência mundial, e não uma característica exclusivamente brasileira. Segundo ele, além dos outros meios de locomoção, também concorrem com o transporte coletivo mudanças de hábito como o uso de telefones celulares, internet, informática, entre outras facilidades que fazem com que a população se desloque menos. Destacou também, que o transporte público tem sido prejudicado pela lentidão do trânsito, já que a velocidade média do transporte por ônibus é inferior a 20 km/h em 49% dos municípios.

*A cidade cresce em um ritmo e o crescimento do volume da frota é bem maior. Tanto que, quando se fala sobre bons índices da economia, compara-se com o aumento de carros novos vendidos. Há 10 anos, se colocássemos*

*todos os automóveis de São Paulo um atrás do outro,  
daria pra ir do Oiapoque ao Chuí.*

**Maria da Penha**

*Coordenadora da divisão de trânsito do Instituto de  
Engenharia.*

Nos anos 70 e 80 o sistema de ônibus, reconhecidamente deficiente, era alvo predileto de protestos e pancadarias e outras manifestações empreendidas por estudantes e trabalhadores, em geral em repúdio aos reajustes acelerados das tarifas para acompanhar a estratosférica inflação que caracterizava aquelas décadas. Nos anos 90, sobretudo na segunda metade, com a inflação contida, as manifestações passaram a ter como alvo as deficiências operacionais dos ônibus que se aprofundaram com a invasão dos “perueiros”. Nos primeiros anos do século 21 o que se vê é o sistema de ônibus sendo colocado na berlinda como uma das soluções para enfrentar a falta de mobilidade nas cidades.

O setor de transporte público de passageiros exerce papel fundamental de integração do tecido urbano, afetando diretamente a produtividade das demais atividades econômicas. Segundo Bicalho (1998), o transporte tem importância social decisiva porque permite que as pessoas acessem locais de trabalho, equipamentos sociais e de lazer, e oportunidades de consumo, principalmente nas cidades de porte médio e grande, onde é maior a dependência da população dos meios de transporte coletivo. Ainda, o transporte coletivo não é menos importante sob o ponto de vista econômico, pois viabiliza o crescimento das cidades, organizando e condicionando a ocupação dos espaços urbanos e o uso do solo através da articulação entre as atividades desenvolvidas em diferentes locais. Trata-se, portanto, de serviço público essencial, que deve atender às necessidades sociais e dar suporte às atividades econômicas.

No Dia Mundial Sem Carro, moradores de cidades brasileiras param para refletir sobre o estilo de vida urbano. Ruas para a circulação de veículos motorizados são fechadas. O objetivo é incentivar a população a procurar alternativas de transporte. O problema, segundo engenheiros especialistas em trânsito, é que as cidades não estão preparadas para o crescimento da frota. Não há espaço nas ruas, nem estacionamento suficiente para tantos veículos. Os resultados são o aumento da poluição e o stress cada vez maior dos motoristas.

Bilhetagem eletrônica, ônibus de piso baixo, GPS, corredores segregados ou exclusivos para ônibus são alguns dos itens que compõem o arsenal tecnológico que está sendo adotado para manter e atrair o passageiro ao sistema de transporte coletivo. O sistema de ônibus urbano, que quase sempre esteve na defensiva, a reboque dos avanços, vem buscando como aliados a tecnologia de produtos e sistemas para, a um só tempo, satisfazer o usuário com um serviço que possa garantir o ir-e-vir de maneira rápida, segura e confortável.

Enfim, partindo-se do pressuposto que o setor de transportes vem passando por inúmeras modificações, que a qualidade do serviço prestado deve ser melhorada, já que existe uma ligação intrínseca entre o transporte coletivo urbano e a qualidade de vida de seus usuários (FERNANDES; BODMER, 1995), justifica-se a elaboração deste trabalho como uma nova ferramenta no auxílio à tomada de decisão dos usuários de transporte coletivo.

## 5. SISTEMA DE ESTIMATIVA DE TEMPO DE ESPERA

O sistema de estimativa de tempo de espera (SETE) consiste em fornecer ao usuário do transporte coletivo o tempo de espera estimado do ônibus que deseja apanhar a partir do uso do seu aparelho celular. De posse dessa informação o usuário poderá se beneficiar de várias maneiras como, por exemplo, decidir qual a melhor linha usar de acordo com o tempo de cada uma delas; programar-se para apanhar um ônibus com maior confiabilidade no seu horário, etc.

Para receber uma estimativa de tempo o usuário deve selecionar o ponto de ônibus e a linha que deseja consultar o horário de chegada do ônibus (Figura 6). Estas informações serão fornecidas a partir do aplicativo instalado no dispositivo móvel do usuário (Celular, PDA, etc.) conforme figura abaixo. A aplicação foi projetada com o objetivo de atingir o maior número de usuários com diferentes tipos de dispositivos, por isso a escolha da plataforma JME.

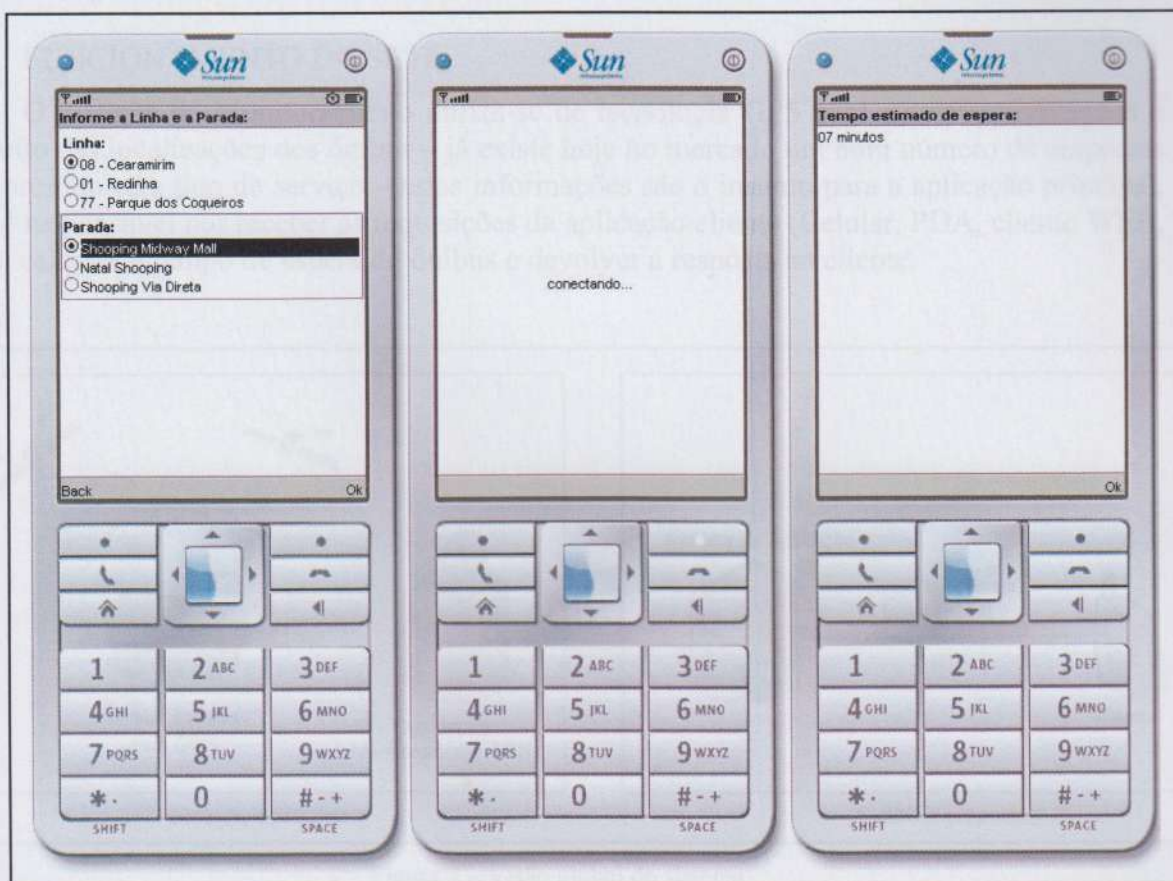


Figura 6 – Aplicação Cliente  
Fonte: Dados da pesquisa

## 5.1 SUBSISTEMAS

O sistema está subdividido em três módulos:

- 1) Sistema de Monitoramento GPS - Responsável por fornecer a localização atual dos ônibus assim como manter uma base de dados com informações das linhas, rotas, trechos, carros e pontos.
- 2) Servidor de Aplicação - Onde se encontra instalado a aplicação principal e o Webservice. Responsável pela consulta à base de dados do Sistema de Monitoramento, cálculo do tempo de espera e publicação do serviço.
- 3) Aplicação Cliente - Aplicativo instalado no aparelho celular, responsável pelo consumo do Webservice. Pode também ser qualquer tipo de cliente que possua a funcionalidade de consumir um Webservice como, por exemplo, um cliente WEB.

## 5.2 FUNCIONAMENTO DO SETE

O Sistema de Monitoramento utiliza-se de tecnologia GPS para coletar informações a respeito das localizações dos ônibus – já existe hoje no mercado um bom número de empresas que prestam este tipo de serviço – estas informações são o insumo para a aplicação principal, que é responsável por receber as requisições da aplicação cliente (Celular, PDA, cliente WEB, etc.), calcular o tempo de espera do ônibus e devolver a resposta ao cliente.

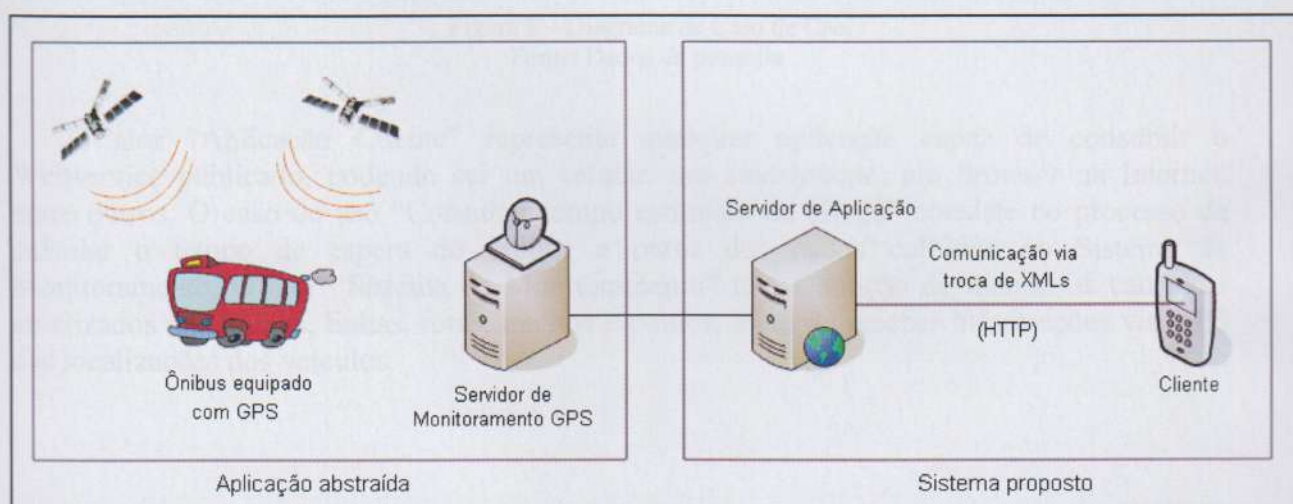


Figura 7 – Visão global do sistema  
Fonte: Dados da pesquisa

### 5.3 DIAGRAMA DE CASO DE USO

O Diagrama de Caso de Uso (DCU) é um dos diagramas da UML e corresponde a uma visão externa de alto nível do sistema. Esse diagrama representa graficamente os atores, casos de uso e relacionamentos entre esses elementos. O DCU tem o objetivo de ilustrar em um nível alto de abstração quais elementos externos interagem com que funcionalidades do sistema. Nesse sentido, a finalidade de um DCU é apresentar um tipo de “diagrama de contexto” que apresenta os elementos externos de um sistema e as maneiras segundo as quais eles as utilizam. Abaixo está representado o DCU do SETE (Figura 8).

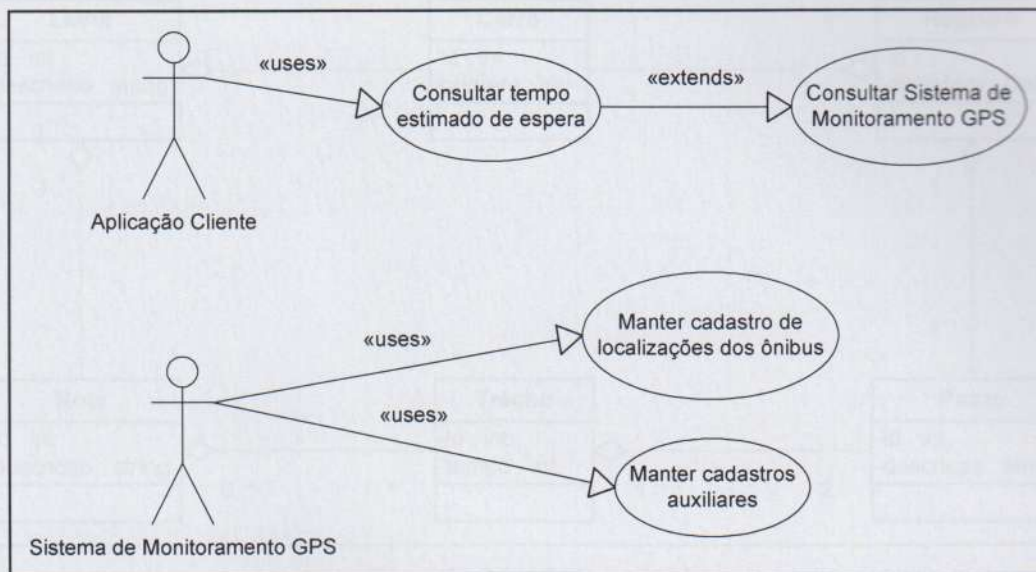


Figura 8 – Diagrama de Caso de Uso  
Fonte: Dados da pesquisa

O ator “Aplicação Cliente” representa qualquer aplicação capaz de consumir o Webservice publicado, podendo ser um celular, um *smartphone*, um *browser* na Internet, entre outros. O caso de uso “Consultar tempo estimado de espera” consiste no processo de calcular o tempo de espera do ônibus a partir dos dados colhidos no Sistema de Monitoramento. O ator “Sistema de Monitoramento” tem a função de manter os cadastros atualizados dos ônibus, linhas, rotas, trechos e pontos, além de receber informações via GPS das localizações dos veículos.

#### 5.4 MODELO CONCEITUAL

O Modelo Conceitual ou Modelo de Classes de Análise representa as classes de análise, os seja, as que se tornam evidentes na medida em que focamos a atenção sobre “o que” o sistema deve fazer. Por definição, um modelo de classes de análise não leva em consideração restrições inerentes à tecnologia a ser utilizada na solução de um problema. Representa os conceitos do domínio em estudo, tem sua perspectiva destinada ao cliente. Abaixo o modelo conceitual do SETE.

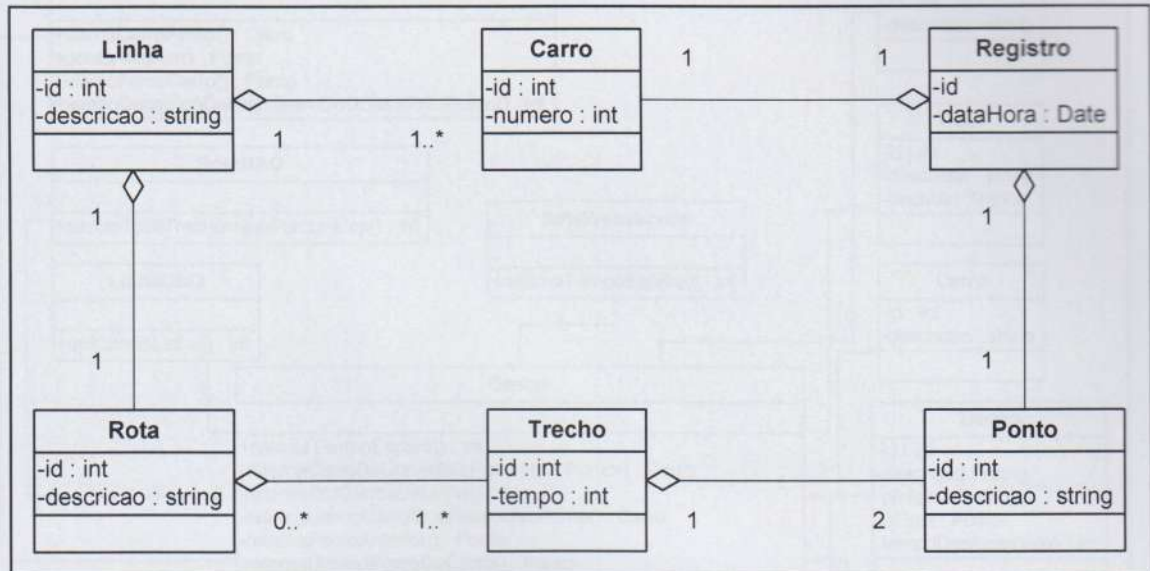


Figura 9 – Modelo Conceitual  
Fonte: Dados da pesquisa

Neste diagrama pode-se observar os relacionamentos por “agregação” entre as classes, pois entende-se que existe entre elas uma relação todo parte onde podemos dizer que um trecho “contém” dois pontos, ou um registro “contém” um carro e um ponto e assim sucessivamente. Para esta visão do sistema definiu-se o conceito de “Linha” como sendo um objeto composto de um ou vários “Carros” e uma única “Rota”. Por sua vez, uma “Rota” é constituída de um ou vários “Trechos”. Cada “Trecho” possui dois “Pontos”. Um “Registro” é a representação da informação que chega sistematicamente no momento em que um “Carro” passa por um “Ponto” e é composto, além desses dois objetos, pela hora em que isto ocorreu.

## 5.5 DIAGRAMA DE CLASSES

O modelo de implementação tem sua perspectiva destinada ao time de desenvolvimento, aborda alguns detalhes de implementação como as operações (métodos), tipo dos atributos, etc. Não foram considerados para este diagrama os métodos modificadores (*get*, *set*) das classes do pacote “entidade” para melhor visualização.

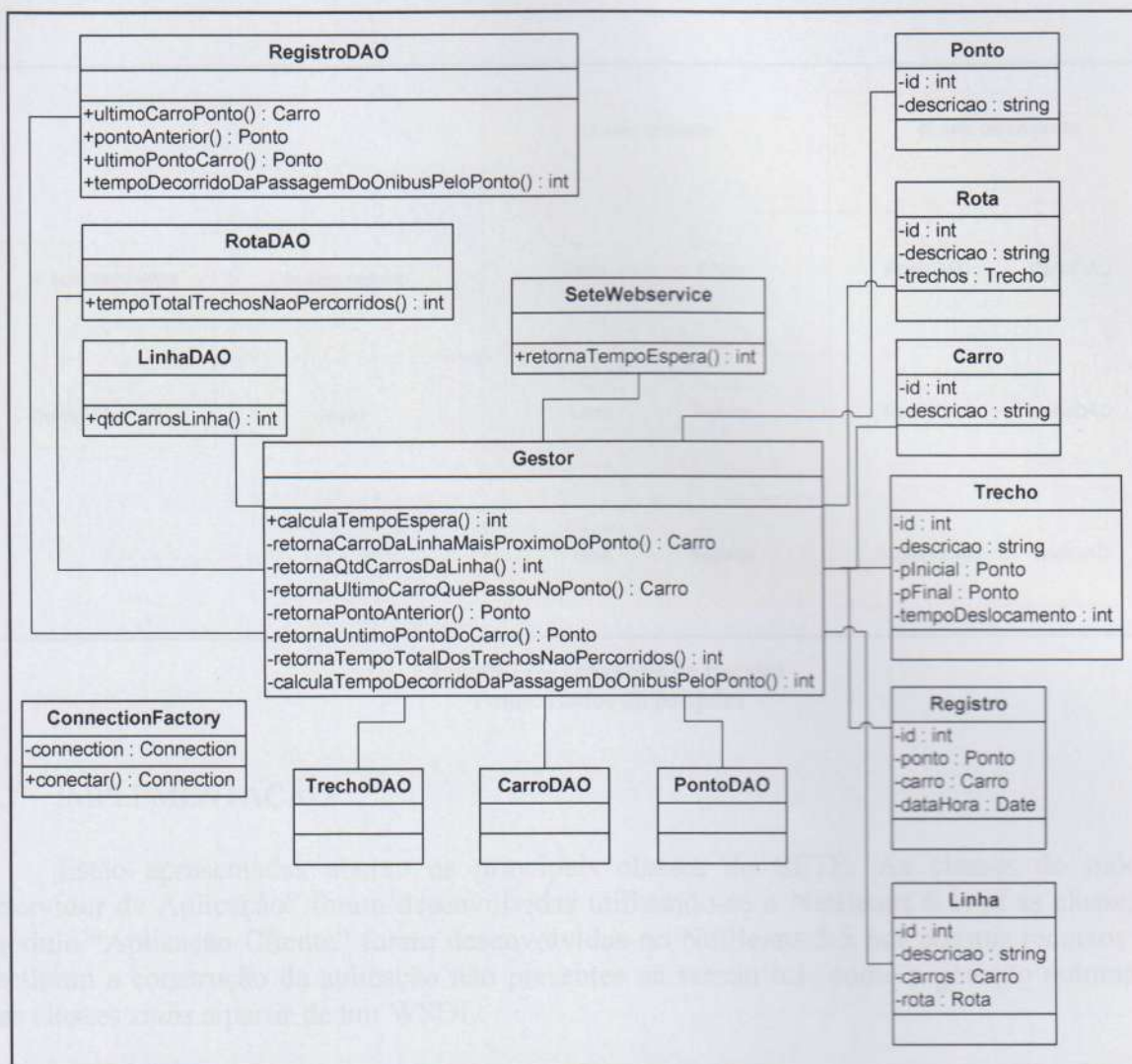


Figura 10 – Diagrama de Classes

Fonte: Dados da pesquisa

## 5.6 DIAGRAMA DE PACOTES

Neste diagrama as classes estão com seus estereótipos definidos como *boundary*, *control*, *entity* e *table* (Figura 11) de acordo com suas respectivas funcionalidades no sistema. Para efeito de simplificação não foram colocados neste diagrama as classes referentes ao cliente da aplicação, no nosso caso o *midlet*, pois existem outras implementações para esta aplicação, como por exemplo, um cliente WEB.

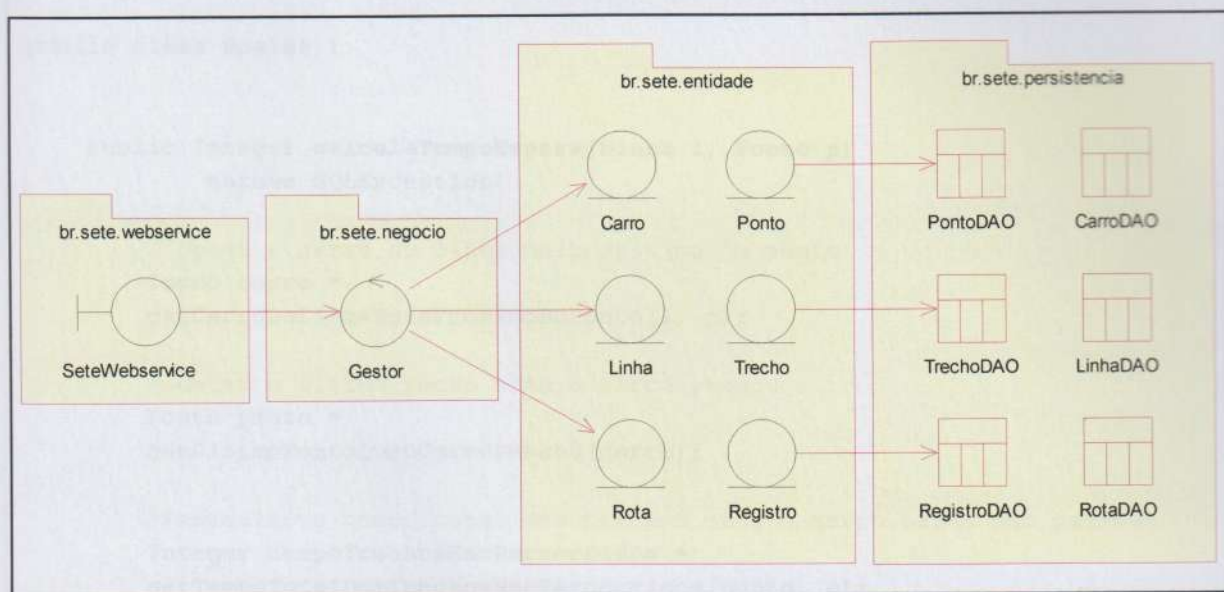


Figura 11 – Diagrama de Pacotes  
Fonte: Dados da pesquisa

## 5.7 IMPLEMENTAÇÃO

Estão apresentadas abaixo as principais classes do SETE. As classes do módulo “Servidor de Aplicação” foram desenvolvidas utilizando-se o NetBeans 6.1, já as classes do módulo “Aplicação Cliente” foram desenvolvidas no NetBeans 5.5 por possuir recursos que facilitam a construção da aplicação não presentes na versão 6.1, como a geração automática das classes *stubs* a partir de um WSDL.

### 5.7.1 A classe “Gestor”

A classe “Gestor” contém a principal lógica de negócio do sistema. É nela onde se pode observar de forma direta como é obtido o tempo de espera estimado do ônibus. O cálculo do tempo de espera estimado é dado através dos seguintes procedimentos:

- ✓ Obter o carro (da linha selecionada pelo usuário) mais próximo do ponto (informado pelo usuário) e que ainda não passou pelo ponto informado;
- ✓ Obter o último ponto onde o ônibus passou;
- ✓ Calcular o tempo total dos trechos onde o ônibus ainda não passou;
- ✓ Obter o tempo decorrido da passagem do ônibus pelo ultimo ponto até agora;

- ✓ Subtrair o tempo decorrido do tempo dos trechos não percorridos.

Após o cálculo do tempo estimado de espera é retornado o valor em minutos ou uma exceção é lançada (Figura 11).

```
public class Gestor {

    public Integer calculaTempoEspera(Linha l, Ponto p)
        throws SQLException{

        //obter o carro da linha mais próximo do ponto
        Carro carro =
            getCarroDaLinhaMaisProximoDoPonto(l, p);

        //obter o último ponto onde o carro passou
        Ponto ponto =
            getUltimoPontoQueOCarroPassou(carro);

        //calcular o tempo total dos trechos onde o carro ainda não passou
        Integer tempoTrechosNaoPercorridos =
            getTempoTotalDosTrechosNaoPercorridos(ponto, p);

        //obter o tempo decorrido do momento da passagem do ônibus no
        //último ponto até o momento presente
        Integer tempoDecorrido =
            getTempoDecorridoDaPassagemDoOnibusPeloPonto(ponto, carro);

        //subtrair o tempo decorrido do tempo dos trechos não percorridos
        return tempoTrechosNaoPercorridos - tempoDecorrido;

    }
}
```

Figura 12: Método “calculaTempoEspera” da classe “Gestor”

Fonte: Dados da pesquisa

Aprofundando-se na funcionalidade desta classe temos descrito abaixo o método “getCarroDaLinhaMaisProximoDoPonto” (Figura 12) onde é recuperado o ônibus (pertencente a linha selecionada pelo usuário) que está mais próximo do ponto (selecionado pelo usuário) para que possa ser calculado posteriormente o tempo de deslocamento do ônibus até o ponto indicado.

```

//retorna o carro mais próximo da linha e que
//ainda não passou pelo ponto
private Carro getCarroDaLinhaMaisProximoDoPonto(Linha l, Ponto p)
    throws SQLException{

    Carro carro = null;
    if(getQtdCarrosDaLinha(l) == 1){
        carro = getUltimoCarroQuePassouNoPonto(p);
    }

    while(carro == null){

        Ponto pAnterior = getPontoAnterior(p);

        Carro ultimoCarroP = getUltimoCarroQuePassouNoPonto(p);
        Carro ultimoCarroPAnterior = getUltimoCarroQuePassouNoPonto(pAnterior);

        if(ultimoCarroP != ultimoCarroPAnterior){
            carro = ultimoCarroPAnterior;
        }else{
            p = pAnterior;
        }
    }
    return carro;
}

```

Figura 13: Método “getCarroDaLinhaMaisProximoDoPonto”  
 Fonte: Dados da pesquisa

O restante dos métodos da classe “Gestor” (Figura 14) dão suporte aos métodos principais e fazem uso das classes do pacote “persistência” (classes DAO – *Data Access Object*).

```

private Integer getQtdCarrosDaLinha(Linha linha){
    LinhaDAO linDao = new LinhaDAO();
    return linDao.qtdCarrosLinha(linha);
}

private Carro getUltimoCarroQuePassouNoPonto(Ponto p) throws SQLException{
    RegistroDAO regDao = new RegistroDAO();
    Carro c = regDao.ultimoCarroPonto(p);
    return c;
}

private Ponto getPontoAnterior(Ponto p){
    //os pontos estão em sequencia na tabela de rotaPontos
    RotaDAO rotDao = new RotaDAO();
    Ponto ponto = rotDao.pontoAnterior(p);
    return ponto;
}

//retorna último registro do carro no ponto
private Ponto getUltimoPontoQueOCarroPassou(Carro carro){
    RegistroDAO regDAO = new RegistroDAO();
    Ponto ponto = regDAO.ultimoPontoCarro(carro);
    return ponto;
}

//somatório dos tempos dos trechos ainda não percorridos pelo ônibus
private Integer getTempoTotalDosTrechosNaoPercorridos
(Ponto pInicial, Ponto pFinal){
    RotaDAO rotDAO = new RotaDAO();
    return rotDAO.tempoTotalTrechosNaoPercorridos(pInicial, pFinal);
}

//retorna a quanto tempo que o ônibus passou no último ponto registrado
private Integer getTempoDecorridoDaPassagemDoOnibusPeloPonto
(Ponto p, Carro c){
    RegistroDAO regDAO = new RegistroDAO();
    return regDAO.tempoDecorridoDaPassagemDoOnibusPeloPonto(p, c);
}

```

Figura 14: Métodos da Classe "Gestor"

Fonte: Dados da pesquisa

### 5.7.2 A classe “SeteWebservice”

Esta classe representa o *web service* do SETE. Nela está contido o método remoto “consultaTempoEspera” que é acessado por qualquer aplicação cliente que possa ser construída baseando-se no arquivo WSDL do *web service*.

```
package br.sete.webservice;

import br.sete.entidade.Linha;
import br.sete.entidade.Ponto;
import br.sete.negocio.Gestor;
import java.sql.SQLException;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

@WebService()
public class SeteWebservice {

    @WebMethod(operationName = "consultaTempoEspera")
    public Integer consultaTempoEspera(@WebParam(name = "linha")
    Integer linha, @WebParam(name = "ponto")
    Integer ponto) throws SQLException {

        Linha l = new Linha();
        l.setId(linha);

        Ponto p = new Ponto();
        p.setId(ponto);

        Gestor g = new Gestor();
        return g.calculaTempoEspera(l,p);
    }
}
```

Figura 15: Classe “SeteWebservice”

Fonte: Dados da pesquisa

O *Web Service* é, na prática, implementado por uma aplicação *web* comum (HTTP/HTTPS), sem o uso de qualquer extensão proprietária. A publicação de um *Web Service* é, fisicamente, como a publicação de uma aplicação *web* comum.

### 5.7.3 Implementação do Cliente JME

Foi possível através da geração automática de uma aplicação cliente JME a criação dos componentes necessários ao consumo do *web service*. Esta aplicação foi desenvolvida com o auxílio do NetBeans 5.5 e tem suas especificações baseadas na configuração CLDC-1.0 e perfil MIDP-2.0.

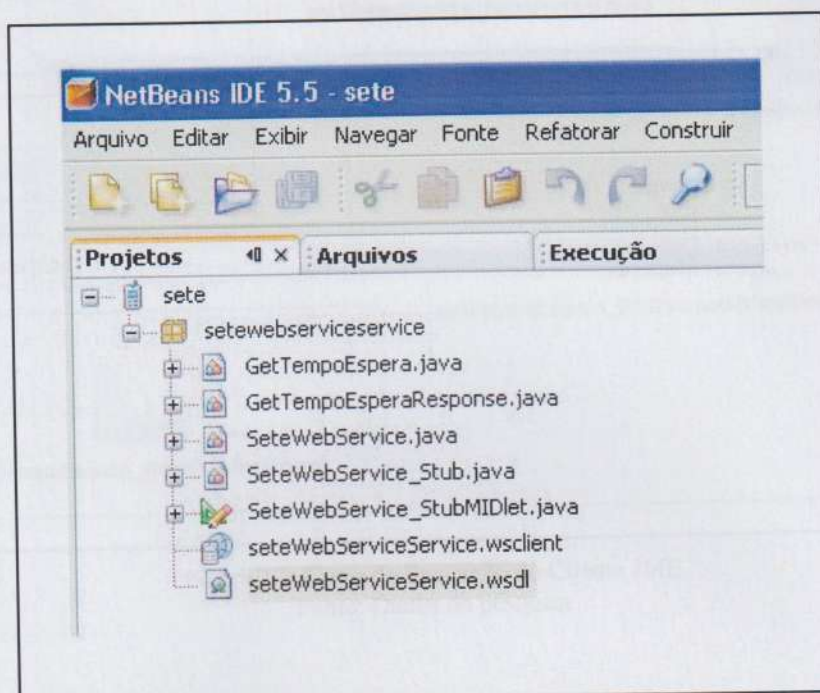


Figura 16: Estrutura do Cliente JME  
Fonte: Dados da pesquisa

Fazem parte desta aplicação a classe “GetTempoEspera” ( ver ANEXO C), a classe “GetTempoEsperaResponse” responsável por responder à solicitação do cliente, a interface “SeteWebService” e a implementação desta interface a classe “SeteWebService\_Stub”. Além dessas é gerado automaticamente um MIDlet denominado “SeteWebService\_StubMIDlet” contendo a priori alguns componentes necessários à execução do sistema como mostrado no *Flow Designer* abaixo (Figura 17).

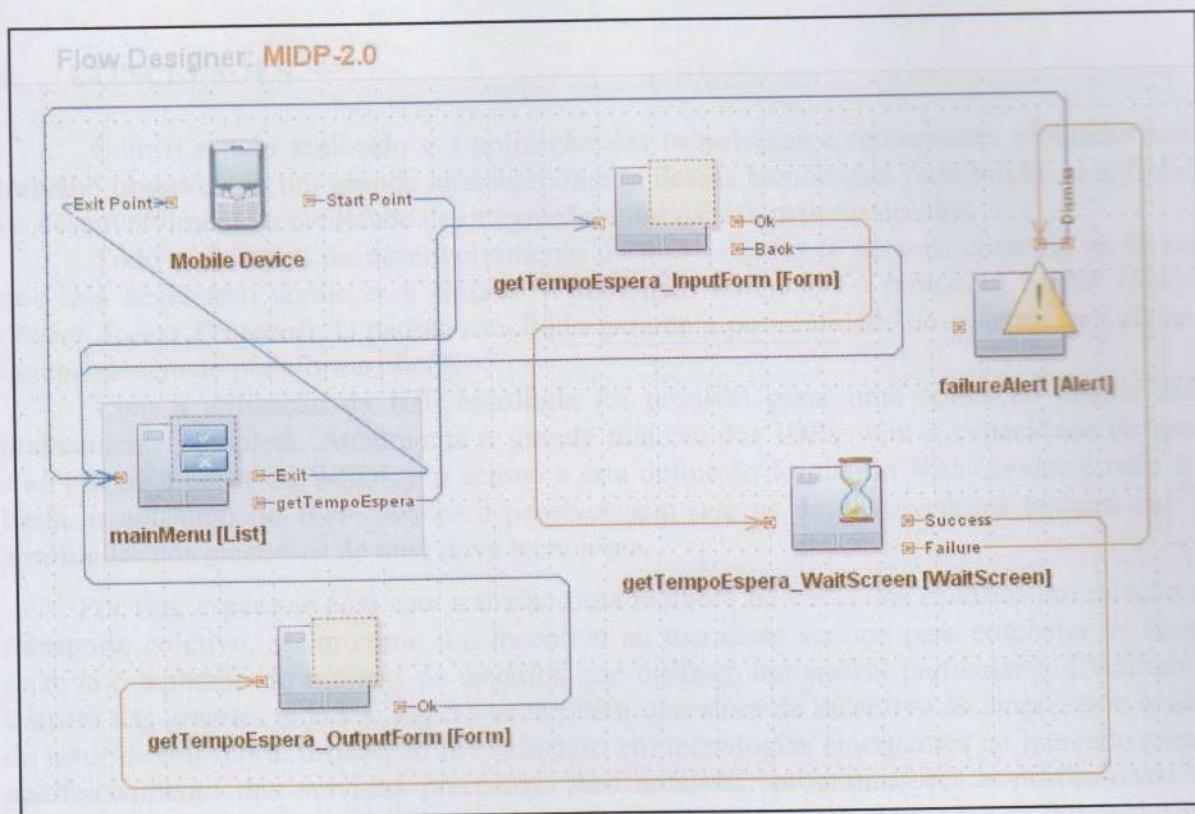


Figura 17: Fluxo de Execução do Cliente JME  
 Fonte: Dados da pesquisa

## 6. CONCLUSÕES

Com o estudo realizado e a aplicação das tecnologias e ferramentas utilizadas neste trabalho observou-se um grande amadurecimento dessas tecnologias possibilitando agilidade no desenvolvimento e facilidade de integração entre os sistemas envolvidos.

Todo o processo de desenvolvimento de *web services* (e de seus clientes) se dá sem que seja necessário conhecer a sintaxe WSDL nem tampouco o protocolo SOAP (*Simple Object Access Protocol*). O padrão escolhido garante a portabilidade do sistema sem afetar a independência de plataforma cliente.

Com a utilização da IDE escolhida foi possível gerar uma aplicação cliente JME praticamente completa. Atualmente a grande maioria das IDEs têm a capacidade de gerar *stub* cliente a partir do WSDL e o acesso a esta definição formal do *Web Service* é tudo que basta, o consumo do *Web Service* é possível sem que os desenvolvedores tenham que se aprofundar nos meandros de uma nova tecnologia.

Por fim, espera-se com este trabalho uma melhora na visão dos usuários em relação ao transporte coletivo, assim como um incentivo ao uso deste serviço para combater de forma criativa o aumento do número de usuários que utilizam automóvel particular e dificultam o trânsito nas grandes cidades. Espera-se também que sirva de incentivo às empresas e órgãos do setor de transporte urbano no investimento em tecnologias emergentes no mercado para o aperfeiçoamento dos serviços prestados. Aos analistas, programadores e profissionais de informática ficam os exemplos de como as tecnologias podem ser integradas de forma simples para a solução de problemas complexos do nosso cotidiano e de como atualmente temos um arsenal tecnológico de livre acesso que pode ser conhecido e utilizado como ferramenta facilitadora na elaboração de soluções, a exemplo JME, Internet, GPS e Webservices.

Para trabalhos futuros fica o melhoramento do sistema com a configuração das características do ônibus que o usuário deseja apanhar, pois existem ônibus preparados especialmente para o transporte de deficientes físicos (Ex.: elevador para cadeirantes, espaço para cão-guia, assentos especiais para obesos) e como não são todos os ônibus é de grande valia para o usuário deste tipo de serviço se beneficiar desta informação.

ANTP. São Paulo, ano 18, nº 37-40, jul-dez. 1995.

ITA - Divisão de Engenharia, engenharia. O Sistema GPS. Disponível em:  
<http://www.igara.com.br/infocliente/gps.asp>. Acesso em: 20 set. 2008.

JUNQUEIRA, Leonardo R. PINTO, Vercia Peres. O ônibus e o desenvolvimento. *Revista dos Transportes Públicos - ANTP*, São Paulo, ano 22, p. 53-74, jul-dez. 1999.

MONICO, J. G. Posicionamento pelo NAVSTAR-GPS Descrição, funcionamento e aplicações. Presidente Prudente, Editora UNESP, 2006.

PEREIRA, Wilfrido Alberto de Araújo; CRIZ, Denize Cássia; GARCIA, RITA. Amadurecimento. *Reportes dos transportes urbanos entre 1990 e 1999. Revista dos Transportes Públicos - ANTP*, São Paulo, ano 22, p. 19-27, jul-dez. 1999.

## REFERÊNCIAS

ASSOCIAÇÃO NACIONAL DOS TRANSPORTES PÚBLICOS – ANTP. **Anuário dos transportes públicos**. São Paulo, 1998.

AZAMBUJA, Ana Maria Volkmer de. **Análise de Eficiência na Gestão do Transporte Urbano por Ônibus em Municípios Brasileiros**. 2002. 80f. Tese (Doutorado). - UFSC. Florianópolis. Santa Catarina, 2002.

BERTOZZI, Patrícia Pacheco; LIMA JR., Orlando Fontes. A qualidade no serviço de transporte público sob as óticas do usuário, do operador e do órgão gestor. **Revista dos Transportes Públicos - ANTP**, São Paulo, ano 21, p. 53-61, out.-dez. 1998.

BICALHO, Marcos. A dívida social no transporte coletivo. **Revista dos Transportes Públicos - ANTP**, São Paulo, ano 20, p. 33-41, jul.-set. 1998. BLY.

BRASILEIRO, Anísio. A moderna empresa de ônibus urbanos no Brasil. **Revistas dos Transportes Públicos - ANTP**, São Paulo, ano 18, p. 39-51, abr.-jun. 1996.

CEDANO, Carlos Alberto. O consenso básico: a problemática do transporte urbano. **Revista dos Transportes Públicos - ANTP**, São Paulo, ano 22, , pp. 43-52. out.-dez. 1999.

FERNANDES, Fátima Sobral & BODMER, Milena. Gestão empresarial da qualidade nos transportes: aproximação entre teoria e prática. **Revista dos Transportes Públicos - ANTP**, São Paulo, ano 18, p. 33-43, out.-dez. 1995.

ITA - Divisão de Engenharia Aeronáutica. **O Sistema GPS**. Disponível em: <<http://www.figueroa.kit.net/didaticos/gps.html>>. Acesso em: 20 set. 2008.

JUNQUEIRA, Laurindo & PINTO, Valeska Peres. O trânsito e o desenvolvimento urbano. **Revista dos Transporte Públicos - ANTP**, São Paulo, ano 22, p. 53-74, out.-dez. 1999.

MONICO, J.F.G., **Posicionamento pelo NAVSTAR-GPS Descrição, fundamentos e aplicações**. Presidente Prudente: Editora UNESP, 2000.

PEREIRA, Willian Alberto de Aquino; CRUZ, Denise Cadete Guazzinelli; BRUNI, Bernadete Ballarin. Aspectos dos transportes urbanos entre 1998 e 1999. **Revista dos Transporte Públicos - ANTP**, São Paulo, ano 22, p. 19-27, out.-dez. 1999.

TRAVASSOS, Germano. Por que falam tão mal desse nosso transporte? Imagem e realidade dos sistemas de transporte público de passageiros. **Revista dos Transportes Públicos - ANTP**, ano 22, p. 77-93, jan.-mar. 2000.

W3C - a. **WebServices Architecture**. Disponível em: <<http://www.w3.org/TR/ws-arch/>>. Acesso em: 10 ago. 2008.

W3C - b. **WebServices Activity**. Disponível em: <<http://www.w3.org/2002/ws/>>. Acesso em: 05 set. 2008.

WEBBER, C; MEDEIROS, C; .Net: Uma Introdução A WebServices. **Revista de divulgação técnico-científica do ICPG**. v.2, n.5, p.33 - set. 2008. Disponível em: <[http://www.icpg.com.br/hp/revista/download.exec.php?rpa\\_chave=05df0a35fecc9d928629](http://www.icpg.com.br/hp/revista/download.exec.php?rpa_chave=05df0a35fecc9d928629)>. Acesso em: 08 set. 2008.

## ANEXO A - CLASSES DO PACOTE "ENTIDADE"

## Classe "Carro"

```
package br.senai.entidade;
```

```
public class Carro {
```

```
    private Integer id;
    private String descricao;
```

```
    public Integer getId() {
        return id;
    }
```

```
    public void setId(Integer id) {
        this.id = id;
    }
```

```
    public String getDescricao() {
        return descricao;
    }
```

```
    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
}
```

## ANEXO

## Classe "Carro"

## Classe "Limbo"

```
package br.senai.entidade;
```

```
import java.util.List;
```

```
public class Limbo {
```

```
    private Integer id;
    private String descricao;
    private List<Carro> carros;
    private Role role;
```

```
    public Integer getId() {
        return id;
    }
```

```
    public void setId(Integer id) {
        this.id = id;
    }
```

```
    public Role getRole() {
        return role;
    }
```

```
    public void setRole(Role role) {
        this.role = role;
    }
}
```

**ANEXO A – CLASSES DO PACOTE “ENTIDADE”****Classe “Carro”**

```
package br.sete.entidade;

public class Carro {

    private Integer id;
    private String descricao;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
}
```

Classe “Carro”

**Classe “Linha”**

```
package br.sete.entidade;

import java.util.List;

public class Linha {

    private Integer id;
    private String descricao;
    private List<Carro> carros;
    private Rota rota;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Rota getRota() {
        return rota;
    }

    public void setRota(Rota rota) {
        this.rota = rota;
    }
}
```

```

public List<Carro> getCarros() {
    return carros;
}

public void setCarros(List<Carro> carros) {
    this.carros = carros;
}

public String getDescricao() {
    return descricao;
}

public void setDescricao(String descricao) {
    this.descricao = descricao;
}
}

```

Classe "Linha"

Classe "Ponto"

```

package br.sete.entidade;

public class Ponto {

    private Integer id;
    private String descricao;

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}

```

Classe "Ponto"

Classe "Registro"

```

package br.sete.entidade;

import java.util.Date;

public class Registro {

    private Integer id;
    private Ponto ponto;
    private Carro carro;
    private Date datahora;
}

```

```

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public Carro getCarro() {
    return carro;
}

public void setCarro(Carro carro) {
    this.carro = carro;
}

public Date getDatahora() {
    return datahora;
}

public void setDatahora(Date datahora) {
    this.datahora = datahora;
}

public Ponto getPonto() {
    return ponto;
}

public void setPonto(Ponto ponto) {
    this.ponto = ponto;
}
}

```

Classe "Registro"

Classe "Rota"

```

package br.sete.entidade;

import java.util.List;

public class Rota {

    private Integer id;
    private String descricao;
    private List<Ponto> pontos;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getDescricao() {
        return descricao;
    }
}

```

```

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public List<Ponto> getPontos() {
        return pontos;
    }

    public void setTrechos(List<Ponto> pontos) {
        this.pontos = pontos;
    }
}

```

Classe "Rota"

Classe "Trecho"

```

package br.sete.entidade;

public class Trecho {

    private Integer id;
    private Ponto pInicial;
    private Ponto pFinal;
    private Integer tempoDeslocamento;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Integer getTempoDeslocamento() {
        return tempoDeslocamento;
    }

    public void setTempoDeslocamento(Integer tempoDeDeslocamento) {
        this.tempoDeslocamento = tempoDeDeslocamento;
    }

    public Ponto getPFinal() {
        return pFinal;
    }

    public void setPFinal(Ponto pFinal) {
        this.pFinal = pFinal;
    }

    public Ponto getPInicial() {
        return pInicial;
    }

    public void setPInicial(Ponto pInicial) {
        this.pInicial = pInicial;
    }
}

```

Classe "Trecho"

## ANEXO B – CLASSES DO PACOTE “NEGÓCIO”

### Classe “Gestor”

```

package br.sete.negocio;

import br.sete.entidade.Carro;
import br.sete.entidade.Linha;
import br.sete.entidade.Ponto;
import br.sete.persistencia.LinhaDAO;
import br.sete.persistencia.RegistroDAO;
import br.sete.persistencia.RotaDAO;
import java.sql.SQLException;

public class Gestor {

    public Integer calculaTempoEspera(Linha l, Ponto p)
        throws SQLException{

        //obter o carro da linha mais próximo do ponto
        Carro carro =
            getCarroDaLinhaMaisProximoDoPonto(l, p);

        //obter o último ponto onde o carro passou
        Ponto ponto =
            getUltimoPontoQueOCarroPassou(carro);

        //calcular o tempo total dos trechos onde o carro ainda não passou
        Integer tempoTrechosNaoPercorridos =
            getTempoTotalDosTrechosNaoPercorridos(ponto, p);

        //obter o tempo decorrido do momento da passagem do ônibus no
        //último ponto até o momento presente
        Integer tempoDecorrido =
            getTempoDecorridoDaPassagemDoOnibusPeloPonto(ponto, carro);

        //subtrair o tempo decorrido do tempo dos trechos não percorridos
        return tempoTrechosNaoPercorridos - tempoDecorrido;
    }

    //retorna o carro mais próximo da linha e que
    //ainda não passou pelo ponto
    private Carro getCarroDaLinhaMaisProximoDoPonto(Linha l, Ponto p)
        throws SQLException{

        Carro carro = null;
        if(getQtdCarrosDaLinha(l) == 1){
            carro = getUltimoCarroQuePassouNoPonto(p);
        }

        while(carro == null){

            Ponto pAnterior = getPontoAnterior(p);

            Carro ultimoCarroP = getUltimoCarroQuePassouNoPonto(p);
            Carro ultimoCarroPAnterior = getUltimoCarroQuePassouNoPonto(pAnterior);

```

```

        if(ultimoCarroP != ultimoCarroPAnterior){
            carro = ultimoCarroPAnterior;

        }else{
            p = pAnterior;
        }
    }
    return carro;
}

private Integer getQtdCarrosDaLinha(Linha linha){

    LinhaDAO linDao = new LinhaDAO();
    return linDao.qtdCarrosLinha(linha);
}

private Carro getUltimoCarroQuePassouNoPonto(Ponto p)throws SQLException{

    RegistroDAO regDao = new RegistroDAO();
    Carro c = regDao.ultimoCarroPonto(p);
    return c;
}

private Ponto getPontoAnterior(Ponto p){
    //os pontos estão em sequencia na tabela de rotaPontos
    RotaDAO rotDao = new RotaDAO();
    Ponto ponto = rotDao.pontoAnterior(p);
    return ponto;
}

//retorna último registro do carro no ponto
private Ponto getUltimoPontoQueOCarroPassou(Carro carro){

    RegistroDAO regDAO = new RegistroDAO();
    Ponto ponto = regDAO.ultimoPontoCarro(carro);
    return ponto;
}

//somatório dos tempos dos trechos ainda não percorridos pelo ônibus
private Integer getTempoTotalDosTrechosNaoPercorridos
    (Ponto pInicial, Ponto pFinal){

    RotaDAO rotDAO = new RotaDAO();
    return rotDAO.tempoTotalTrechosNaoPercorridos(pInicial, pFinal);
}

//retorna a quanto tempo que o ônibus passou no último ponto registrado
private Integer getTempoDecorridoDaPassagemDoOnibusPeloPonto
    (Ponto p, Carro c){

    RegistroDAO regDAO = new RegistroDAO();
    return regDAO.tempoDecorridoDaPassagemDoOnibusPeloPonto(p, c);
}
}

```

Classe "Gestor"

## ANEXO C – WEB SERVICE

### Classe “SeteWebservice”

```
package br.sete.webservice;

import br.sete.entidade.Linha;
import br.sete.entidade.Ponto;
import br.sete.negocio.Gestor;
import java.sql.SQLException;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

@WebService()
public class SeteWebservice {

    @WebMethod(operationName = "consultaTempoEspera")
    public Integer consultaTempoEspera(@WebParam(name = "linha")
    Integer linha, @WebParam(name = "ponto")
    Integer ponto) throws SQLException {

        Linha l = new Linha();
        l.setld(linha);

        Ponto p = new Ponto();
        p.setld(ponto);

        Gestor g = new Gestor();
        return g.calculaTempoEspera(l,p);
    }
}
```

Classe “SeteWebservice”

### WSDL

```
<?xml version="1.0" encoding="UTF-8"?><!-- Published by JAX-WS RI at http://jax-ws.dev.java.net.
RI's version is JAX-WS RI 2.1.3.1-hudson-417-SNAPSHOT. --><!-- Generated by JAX-WS RI at
http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.3.1-hudson-417-SNAPSHOT. -->

<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://webservice.sete.br/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://webservice.sete.br/" name="seteWebServiceService">

    <types>
        <xsd:schema>
            <xsd:import namespace="http://webservice.sete.br/"
            schemaLocation="http://localhost:8080/sete/seteWebServiceService?xsd=1"></xsd:import>
        </xsd:schema>
    </types>

    <message name="getTempoEspera">
        <part name="parameters" element="tns:getTempoEspera"></part>
    </message>

    <message name="getTempoEsperaResponse">
        <part name="parameters" element="tns:getTempoEsperaResponse"></part>
    </message>
```

```
<portType name="seteWebService">
<operation name="getTempoEspera">
<input message="tns:getTempoEspera"></input>
<output message="tns:getTempoEsperaResponse"></output>
</operation>
</portType>
<binding name="seteWebServicePortBinding" type="tns:seteWebService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"></soap:binding>
<operation name="getTempoEspera">
<soap:operation soapAction=""></soap:operation>
<input>
<soap:body use="literal"></soap:body>
</input>
<output>
<soap:body use="literal"></soap:body>
</output>
</operation>
</binding>
<service name="seteWebServiceService">
<port name="seteWebServicePort" binding="tns:seteWebServicePortBinding">
<soap:address location="http://localhost:8080/sete/seteWebServiceService"></soap:address>
</port>
</service>
</definitions>
```

WSDL

## ANEXO D – CLASSES DA APLICAÇÃO CLIENTE

### Classe “GetTempoEspera”

```
package setewebsserviceservice;

public class GetTempoEspera {
    protected int linha;
    protected int ponto;

    public GetTempoEspera() {
    }

    public GetTempoEspera(int linha, int ponto) {
        this.linha = linha;
        this.ponto = ponto;
    }

    public int getLinha() {
        return linha;
    }

    public void setLinha(int linha) {
        this.linha = linha;
    }

    public int getPonto() {
        return ponto;
    }

    public void setPonto(int ponto) {
        this.ponto = ponto;
    }
}
```

### Classe “GetTempoEspera”

### Classe “GetTempoEsperaResponse”

```
package setewebsserviceservice;

public class GetTempoEsperaResponse {
    protected java.lang.Integer _return;

    public GetTempoEsperaResponse() {
    }

    public GetTempoEsperaResponse(java.lang.Integer _return) {
        this._return = _return;
    }

    public java.lang.Integer get_return() {
        return _return;
    }

    public void set_return(java.lang.Integer _return) {
    }
}
```

```

        this._return = _return;
    }
}

```

Classe "GetTempoEsperaResponse"

Interface "SeteWebservice"

```

package setewebserviceservice;

public interface SeteWebService extends java.rmi.Remote {
    public java.lang.Integer getTempoEspera(int linha, int ponto) throws
    java.rmi.RemoteException;
}

```

Interface "SeteWebservice"

Classe "SeteWebService\_Stub"

```

package setewebserviceservice;

import javax.xml.rpc.JAXRPCException;
import javax.xml.namespace.QName;
import javax.microedition.xml.rpc.Operation;
import javax.microedition.xml.rpc.Type;
import javax.microedition.xml.rpc.ComplexType;
import javax.microedition.xml.rpc.Element;

public class SeteWebService_Stub implements setewebserviceservice.SeteWebService,
    javax.xml.rpc.Stub {
    private String[] _propertyNames;
    private Object[] _propertyValues;

    public SeteWebService_Stub() {
        _propertyNames = new String[] {ENDPOINT_ADDRESS_PROPERTY};
        _propertyValues = new Object[]
        {"http://localhost:8080/sete/seteWebServiceService"};
    }

    public void _setProperty(String name, Object value) {
        int size = _propertyNames.length;
        for (int i = 0; i < size; ++i) {
            if (_propertyNames[i].equals(name)) {
                _propertyValues[i] = value;
                return;
            }
        }
        // Need to expand our array for a new property
        String[] newPropNames = new String[size + 1];
        System.arraycopy(_propertyNames, 0, newPropNames, 0, size);
        _propertyNames = newPropNames;
        Object[] newPropValues = new Object[size + 1];
        System.arraycopy(_propertyValues, 0, newPropValues, 0, size);
        _propertyValues = newPropValues;

        _propertyNames[size] = name;
        _propertyValues[size] = value;
    }
}

```

```

public Object _getProperty(String name) {
    for (int i = 0; i < _propertyNames.length; ++i) {
        if (_propertyNames[i].equals(name)) {
            return _propertyValues[i];
        }
    }
    if (ENDPOINT_ADDRESS_PROPERTY.equals(name) ||
    USERNAME_PROPERTY.equals(name) || PASSWORD_PROPERTY.equals(name)) {
        return null;
    }
    if (SESSION_MAINTAIN_PROPERTY.equals(name)) {
        return new java.lang.Boolean(false);
    }
    throw new JAXRPCException("Stub does not recognize property: "+name);
}

protected void _prepOperation(Operation op) {
    for (int i = 0; i < _propertyNames.length; ++i) {
        op.setProperty(_propertyNames[i], _propertyValues[i].toString());
    }
}

//
// Begin user methods
//

public java.lang.Integer getTempoEspera(int linha, int ponto) throws
java.rmi.RemoteException {
    // Copy the incoming values into an Object array if needed.
    Object[] inputObject = new Object[2];
    inputObject[0] = new java.lang.Integer(linha);
    inputObject[1] = new java.lang.Integer(ponto);

    Operation op = Operation.newInstance(_qname_getTempoEspera,
    _type_getTempoEspera, _type_getTempoEsperaResponse);
    _prepOperation(op);
    op.setProperty(Operation.SOAPACTION_URI_PROPERTY, "");
    Object resultObj;
    try {
        resultObj = op.invoke(inputObject);
    } catch (JAXRPCException e) {
        Throwable cause = e.getLinkedCause();
        if (cause instanceof java.rmi.RemoteException) {
            throw (java.rmi.RemoteException) cause;
        }
        throw e;
    }
    java.lang.Integer result;
    // Convert the result into the right Java type.
    // Unwrapped return value
    Object _returnObj = ((Object[])resultObj)[0];
    result = (java.lang.Integer)_returnObj;
    return result;
}

//
// End user methods
//

protected static final QName _qname_linha = new QName("", "linha");
protected static final QName _qname_ponto = new QName("", "ponto");

```

```

        protected static final QName _qname_return = new QName("", "return");
        protected static final QName _qname_getTempoEspera = new
QName("http://webservice.sete.br/", "getTempoEspera");
        protected static final QName _qname_getTempoEsperaResponse = new
QName("http://webservice.sete.br/", "getTempoEsperaResponse");
        protected static final Element _type_getTempoEspera;
        protected static final Element _type_getTempoEsperaResponse;
        static {
            // Create all of the Type's that this stub uses, once.
            Element _type_linha;
            _type_linha = new Element(_qname_linha, Type.INT);
            Element _type_ponto;
            _type_ponto = new Element(_qname_ponto, Type.INT);
            ComplexType _complexType_getTempoEspera;
            _complexType_getTempoEspera = new ComplexType();
            _complexType_getTempoEspera.elements = new Element[2];
            _complexType_getTempoEspera.elements[0] = _type_linha;
            _complexType_getTempoEspera.elements[1] = _type_ponto;
            _type_getTempoEspera = new Element(_qname_getTempoEspera,
_complexType_getTempoEspera);
            Element _type_return;
            _type_return = new Element(_qname_return, Type.INT, 0, 1, false);
            ComplexType _complexType_getTempoEsperaResponse;
            _complexType_getTempoEsperaResponse = new ComplexType();
            _complexType_getTempoEsperaResponse.elements = new Element[1];
            _complexType_getTempoEsperaResponse.elements[0] = _type_return;
            _type_getTempoEsperaResponse = new
Element(_qname_getTempoEsperaResponse, _complexType_getTempoEsperaResponse);
        }
    }
}

```

Classe "SeteWebService\_Stub"

Classe "SeteWebService\_StubMIDlet"

```

/*
 * SeteWebService_StubMIDlet.java
 *
 * Created on 17 de Novembro de 2008, 16:54
 */

package setewebsserviceservice;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author heltonc
 */
public class SeteWebService_StubMIDlet extends MIDlet implements CommandListener {

    /** Creates a new instance of SeteWebService_StubMIDlet */
    * public SeteWebService_StubMIDlet() {
        initialize();
    }

    private List mainMenu;

```

```

private Command exitCommand1;
private org.netbeans.microedition.lcdui.WaitScreen getTempoEspera_WaitScreen;
private org.netbeans.microedition.util.SimpleCancellableTask getTempoEspera_Task;
private Alert failureAlert;
private Form getTempoEspera_InputForm;
private TextField getTempoEspera_linha_InputItem;
private TextField getTempoEspera_ponto_InputItem;
private Command backCommand1;
private Command okCommand1;
private Form getTempoEspera_OutputForm;
private StringItem getTempoEspera_OutputItem;
// End2End client generated code
private SeteWebService_Stub client = new SeteWebService_Stub();

private java.lang.Integer getTempoEspera_returnValue; // field for holding value returned from
method

// method calling the generated method
private void call_getTempoEspera() throws Exception {
    int linha = Integer.parseInt(getTempoEspera_linha_InputItem.getString());
    int ponto = Integer.parseInt(getTempoEspera_ponto_InputItem.getString());
    getTempoEspera_returnValue = client.getTempoEspera(linha, ponto);
}

// method for setting output values
private void set_getTempoEspera_OutputValues() {
    get_getTempoEspera_OutputItem().setText(""+getTempoEspera_returnValue);
}

/** Called by the system to indicate that a command has been invoked on a particular displayable.
 * @param command the Command that ws invoked
 * @param displayable the Displayable on which the command was invoked
 */
public void commandAction(Command command, Displayable displayable) {
    // Insert global pre-action code here
    if (displayable == getTempoEspera_InputForm) {
        if (command == okCommand1) {
            // Insert pre-action code here
            getDisplay().setCurrent(getTempoEspera_WaitScreen());
            // Insert post-action code here
        } else if (command == backCommand1) {
            // Insert pre-action code here
            getDisplay().setCurrent(get_mainMenu());
            // Insert post-action code here
        }
    }
    } else if (displayable == getTempoEspera_WaitScreen) {
        if (command == getTempoEspera_WaitScreen.FAILURE_COMMAND) {
            // Insert pre-action code here
            getDisplay().setCurrent(get_failureAlert(), get_mainMenu());
            // Insert post-action code here
        } else if (command == getTempoEspera_WaitScreen.SUCCESS_COMMAND) {
            // Insert pre-action code here
            getDisplay().setCurrent(get_getTempoEspera_OutputForm());
            // Insert post-action code here
        }
    }
    } else if (displayable == getTempoEspera_OutputForm) {
        if (command == okCommand1) {

```

```

        // Insert pre-action code here
        getDisplay().setCurrent(get_mainMenu());
        // Insert post-action code here
    }
} else if (displayable == mainMenu) {
    if (command == exitCommand1) {
        // Insert pre-action code here
        exitMIDlet();
        // Insert post-action code here
    } else if (command == mainMenu.SELECT_COMMAND) {
        switch (get_mainMenu().getSelectedIndex()) {
            case 0:
                // Insert pre-action code here
                getDisplay().setCurrent(get_getTempoEspera_InputForm());
                // Insert post-action code here
                break;
        }
    }
}
// Insert global post-action code here
}

/** This method initializes UI of the application.
 */
private void initialize() {
    // Insert pre-init code here
    getDisplay().setCurrent(get_mainMenu());
    // Insert post-init code here
}

/**
 * This method should return an instance of the display.
 */
public Display getDisplay() {
    return Display.getDisplay(this);
}

/**
 * This method should exit the midlet.
 */
public void exitMIDlet() {
    getDisplay().setCurrent(null);
    destroyApp(true);
    notifyDestroyed();
}

/** This method returns instance for mainMenu component and should be called instead of
accessing mainMenu field directly.
 * @return Instance for mainMenu component
 */
public List get_mainMenu() {
    if (mainMenu == null) {
        // Insert pre-init code here
        mainMenu = new List(null, Choice.IMPLICIT, new String[] {"getTempoEspera"}, new Image[]
{null});
        mainMenu.addCommand(get_exitCommand1());
        mainMenu.setCommandListener(this);
        mainMenu.setSelectedFlags(new boolean[] {false});
        // Insert post-init code here
    }
}

```

```

    return mainMenu;
}

/** This method returns instance for exitCommand1 component and should be called instead of
accessing exitCommand1 field directly.
 * @return Instance for exitCommand1 component
 */
public Command get_exitCommand1() {
    if (exitCommand1 == null) {
        // Insert pre-init code here
        exitCommand1 = new Command("Exit", Command.EXIT, 1);
        // Insert post-init code here
    }
    return exitCommand1;
}

/** This method returns instance for getTempoEspera_WaitScreen component and should be
called instead of accessing getTempoEspera_WaitScreen field directly.
 * @return Instance for getTempoEspera_WaitScreen component
 */
public org.netbeans.microedition.lcdui.WaitScreen get_getTempoEspera_WaitScreen() {
    if (getTempoEspera_WaitScreen == null) {
        // Insert pre-init code here
        getTempoEspera_WaitScreen = new
org.netbeans.microedition.lcdui.WaitScreen(getDisplay());
        getTempoEspera_WaitScreen.setCommandListener(this);
        getTempoEspera_WaitScreen.setText("Getting data from server ...");
        getTempoEspera_WaitScreen.setTask(get_getTempoEspera_Task());
        // Insert post-init code here
    }
    return getTempoEspera_WaitScreen;
}

/** This method returns instance for getTempoEspera_Task component and should be called
instead of accessing getTempoEspera_Task field directly.
 * @return Instance for getTempoEspera_Task component
 */
public org.netbeans.microedition.util.SimpleCancellableTask get_getTempoEspera_Task() {
    if (getTempoEspera_Task == null) {
        // Insert pre-init code here
        getTempoEspera_Task = new org.netbeans.microedition.util.SimpleCancellableTask();
        getTempoEspera_Task.setExecutable(new org.netbeans.microedition.util.Executable() {
            public void execute() throws Exception {
                // calls end-to-end method and setting result to output screens
                try {
                    call_getTempoEspera();
                } catch (java.io.IOException ioe) {
                    throw new RuntimeException(ioe.getMessage());
                }
            }
        });
        set_getTempoEspera_OutputValues();
    }
    // Insert post-init code here
}
return getTempoEspera_Task;
}

/** This method returns instance for failureAlert component and should be called instead of

```

accessing failureAlert field directly.

```
* @return Instance for failureAlert component
*/
public Alert get_failureAlert() {
    if (failureAlert == null) {
        // Insert pre-init code here
        failureAlert = new Alert("Failure.", "Communication failed.", null, AlertType.ERROR);
        failureAlert.setTimeout(-2);
        // Insert post-init code here
    }
    return failureAlert;
}
```

/\*\* This method returns instance for getTempoEspera\_InputForm component and should be called instead of accessing getTempoEspera\_InputForm field directly.

```
* @return Instance for getTempoEspera_InputForm component
*/
```

```
public Form get_getTempoEspera_InputForm() {
    if (getTempoEspera_InputForm == null) {
        // Insert pre-init code here
        getTempoEspera_InputForm = new Form("getTempoEspera", new Item[] {
            get_getTempoEspera_linha_InputItem(),
            get_getTempoEspera_ponto_InputItem()
        });
        getTempoEspera_InputForm.addCommand(get_backCommand1());
        getTempoEspera_InputForm.addCommand(get_okCommand1());
        getTempoEspera_InputForm.setCommandListener(this);
        // Insert post-init code here
    }
    return getTempoEspera_InputForm;
}
```

/\*\* This method returns instance for getTempoEspera\_linha\_InputItem component and should be called instead of accessing getTempoEspera\_linha\_InputItem field directly.

```
* @return Instance for getTempoEspera_linha_InputItem component
*/
```

```
public TextField get_getTempoEspera_linha_InputItem() {
    if (getTempoEspera_linha_InputItem == null) {
        // Insert pre-init code here
        getTempoEspera_linha_InputItem = new TextField("linha", "0", 120, TextField.ANY);
        // Insert post-init code here
    }
    return getTempoEspera_linha_InputItem;
}
```

/\*\* This method returns instance for getTempoEspera\_ponto\_InputItem component and should be called instead of accessing getTempoEspera\_ponto\_InputItem field directly.

```
* @return Instance for getTempoEspera_ponto_InputItem component
*/
```

```
public TextField get_getTempoEspera_ponto_InputItem() {
    if (getTempoEspera_ponto_InputItem == null) {
        // Insert pre-init code here
        getTempoEspera_ponto_InputItem = new TextField("ponto", "0", 120, TextField.ANY);
        // Insert post-init code here
    }
    return getTempoEspera_ponto_InputItem;
}
```

/\*\* This method returns instance for backCommand1 component and should be called instead of accessing backCommand1 field directly.

```

    * @return Instance for backCommand1 component
    */
    public Command get_backCommand1() {
        if (backCommand1 == null) {
            // Insert pre-init code here
            backCommand1 = new Command("Back", Command.BACK, 1);
            // Insert post-init code here
        }
        return backCommand1;
    }

    /** This method returns instance for okCommand1 component and should be called instead of
    accessing okCommand1 field directly.
    * @return Instance for okCommand1 component
    */
    public Command get_okCommand1() {
        if (okCommand1 == null) {
            // Insert pre-init code here
            okCommand1 = new Command("Ok", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand1;
    }

    /** This method returns instance for getTempoEspera_OutputForm component and should be
    called instead of accessing getTempoEspera_OutputForm field directly.
    * @return Instance for getTempoEspera_OutputForm component
    */
    public Form get_getTempoEspera_OutputForm() {
        if (getTempoEspera_OutputForm == null) {
            // Insert pre-init code here
            getTempoEspera_OutputForm = new Form("getTempoEspera Output", new Item[]
{get_getTempoEspera_OutputItem()});
            getTempoEspera_OutputForm.addCommand(get_okCommand1());
            getTempoEspera_OutputForm.setCommandListener(this);
            // Insert post-init code here
        }
        return getTempoEspera_OutputForm;
    }

    /** This method returns instance for getTempoEspera_OutputItem component and should be called
    instead of accessing getTempoEspera_OutputItem field directly.
    * @return Instance for getTempoEspera_OutputItem component
    */
    public StringItem get_getTempoEspera_OutputItem() {
        if (getTempoEspera_OutputItem == null) {
            // Insert pre-init code here
            getTempoEspera_OutputItem = new StringItem("getTempoEspera_OutputItem", "<Enter
Text>");
            // Insert post-init code here
        }
        return getTempoEspera_OutputItem;
    }

    public void startApp() {
    }

    public void pauseApp() {
    }

```

```
public void destroyApp(boolean unconditional) {  
    }  
}
```

Classe "SeteWebService\_StubMIDlet"