UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

FILIPE TESTA DAROS

DETECÇÃO DE ENGARRAFAMENTOS ATRAVÉS DE *SMARTPHONES*

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA - PR
2012

FILIPE TESTA DAROS

DETECÇÃO DE ENGARRAFAMENTOS ATRAVÉS DE *SMARTPHONES*

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de “Especialista em Tecnologia Java”.

Orientador: Prof. Dr. Adolfo Neto

CURITIBA - PR
2012

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria do Campus Curitiba
Gerência de Pesquisa e Pós-graduação
Departamento Acadêmico de Informática
Curso de Especialização em Tecnologia Java

TERMO DE APROVAÇÃO

Detecção de engarrafamentos através de *smartphones*

Essa monografia foi apresentada às _____ do dia ___ de _____ 2012 como requisito parcial para obtenção do título de Especialista em Tecnologia Java - Departamento Acadêmico de Informática - Universidade Tecnológica Federal do Paraná. O candidato apresentou o trabalho para a Banca Examinadora composta pelos professores abaixo assinados. Após a deliberação, a Banca Examinadora considerou o trabalho _____.

_____
Prof.
Membro
(UTFPR)

_____
Prof.
Membro
(UTFPR)

_____
Prof.
Membro
(UTFPR)

Visto da coordenação:

_____
Prof. Dr. João Alberto Fabro
Coordenador
Curso de Especialização em Tecnologia Java

DEDICATÓRIA

À minha esposa, Vanessa.

RESUMO

DAROS, Filipe T. Detecção de engarrafamento através de *smartphones*. 2012. Monografia (Especialização em Tecnologia Java) – Departamento Acadêmico de Informática. Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

Este trabalho apresenta, através do uso das tecnologias Android, GPS e Web Services, uma aplicação capaz de detectar e compartilhar informações sobre rotas congestionadas em centros urbanos.

Palavras-chave

Detecção de engarrafamentos, Android, GPS.

ABSTRACT

DAROS, Filipe T. Traffic jam detection using smartphones. 2012. Monografia (Especialização em Tecnologia Java) – Departamento Acadêmico de Informática. Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

This work presents, through use of Android, GPS and Web Services technologies, an application capable of detecting and sharing urban centres traffic jam information.

Key-words

Traffic jam detection, Android, GPS.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O trânsito é hoje um dos principais problemas dos grandes centros urbanos. As pessoas acabam despendendo tempo demais presas em engarrafamentos enquanto poderiam aproveitar esse tempo de maneira mais interessante. As prefeituras investem grande quantidade de recursos em vias expressas para tentar aumentar a fluidez do tráfego, porém essas avenidas congestionam rapidamente.

Existem muitos fatores que contribuem para o problema do trânsito. Entre os mais importantes podemos citar os semáforos mal sincronizados, cruzamentos bloqueados, túneis, curvas, radares mal posicionados, etc. Alguns estudos defendem que engarrafamentos surgiriam mesmo sem a existência destes gargalos, como experimentado em Sugiyama (2008).

Apesar de todos os fatores citados acima, um fator predominante nos engarrafamentos é um grande número de veículos tentando transitar por uma mesma rota simultaneamente. As vias expressas recém-criadas rapidamente se tornam referência para os condutores, contribuindo para a ocorrência de congestionamentos. Se os condutores pudessem se antecipar a uma rota engarrafada e escolher um caminho alternativo, os congestionamentos poderiam ter proporções menores. Nesse aspecto, um software que possua informações sobre o trânsito e que seja capaz de processar e compartilhar essas informações com uma rede de usuários torna-se interessante.

1.1 OBJETIVO GERAL

O trabalho tem por objetivo a criação de um software para *smartphones* capaz de calcular e compartilhar informações sobre o trânsito com uma rede de usuários.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos do trabalho são:

1. Abstrair informações sobre o trânsito através do cálculo de velocidade média em curtos espaços de tempo/deslocamento.

2. Enviar essas informações para um servidor através de *web services*.

3. Recuperar informações do servidor contendo as estatísticas atuais sobre as rotas próximas bem como sugestão de rotas alternativas caso o caminho atual esteja congestionado.

1.3 ESTRUTURA DO SOFTWARE

O software foi construído de modo a manter a maior quantidade de processamento possível no servidor. Os dispositivos móveis servem basicamente para recolher os dados e interagir com o usuário através de alertas visuais e sonoros. O servidor é responsável por atender aos pedidos dos dispositivos móveis, processar os dados recebidos e atualizar estatísticas bem como devolver esses dados calculados e sugerir rotas alternativas para o usuário caso ele se encontre em um local de tráfego intenso.

A primeira fase do projeto consistiu no desenvolvimento do módulo de captação de informações que roda nos dispositivos móveis e é usado para o levantamento inicial de dados que auxilia o desenvolvimento e testes do artefato.

A segunda fase abordou o desenvolvimento do lado servidor e da camada de comunicação com os clientes. Nesta fase são codificados os cálculos com os dados enviados pelos usuários e a sugestão de rotas alternativas caso necessárias.

Na terceira e última fase foi desenvolvido o *front-end* da aplicação que se comunica com o servidor para envio e recebimento de dados e orienta o usuário com as estatísticas atuais sobre o tráfego.

2. DESENVOLVIMENTO

2.1 ANDROID

No mundo todo aproximadamente três bilhões de pessoas possuem um aparelho celular. Isso torna esse dispositivo o de maior sucesso em vendas superando até mesmo os aparelhos televisores.

Os celulares evoluíram tremendamente desde o seu surgimento até os dias atuais. As pessoas não buscam mais um aparelho que somente sirva para falar. Existe uma grande convergência de funcionalidades que agora são essenciais para que um celular faça sucesso no mercado. Serviços como GPS, acesso a e-mails, Internet, redes sociais e agendas compartilhadas são alguns exemplos de serviços que não estavam disponíveis no mundo da telefonia móvel e agora tornaram-se indispensáveis.

Conforme visto em PurpleTalk (2012), desenvolvido a partir de um kernel modificado de Linux o Android forma um pacote de softwares que possui: SO, *middleware* e as principais aplicações necessárias. Como qualquer outro sistema operacional a principal função do Android é permitir, por meio de abstração, que as aplicações façam uso dos recursos de *hardware* e prover um ambiente bem definido de desenvolvimento. A seguir foi apresentado um breve histórico sobre a plataforma. Em seguida justificou-se o porquê de escolher essa tecnologia para o desenvolvimento do presente trabalho e foram feitas as considerações finais sobre o sistema operacional. Mais adiante foram apresentadas tecnologias complementares que compõem a solução apresentada, dentre elas destacam-se: navegação em mapas, GPS, *web services* e XML, entre outras.

Para o desenvolvimento desse trabalho foi usada a versão mais recente Ice Cream Sandwitch (ANDROID, 2012).

2.2 GPS

Segundo Kowoma (2012) o *Global Positioning System* (GPS) foi desenvolvido pelo departamento de defesa dos Estados Unidos com objetivo principal de auxílio à navegação. Através do uso de 27 satélites e suas respectivas estações base hoje é possível calcular a posição de um dispositivo na Terra com margem de erro de centímetros.

Originalmente o GPS servia apenas para monitorar equipamento militar como, por exemplo, submarinos e mísseis. Atualmente, especialmente graças ao avanço da tecnologia e consequente redução de preço dos receptores, essa tecnologia passou a ser utilizada em carros, celulares, barcos, aviões, etc.

Também de acordo com Kuwoma (2012) os satélites que compõe o GPS estão distribuídos em seis planos orbitais equidistantes ao redor do planeta. Cada plano orbital possui quatro *slots* para satélites. Esse arranjo garante que qualquer ponto da Terra seja visível por ao menos quatro satélites. Essa rede de satélites está em constante manutenção e por esse motivo é um mix de satélites antigos com novos. As duas principais empresas do segmento responsáveis pela manutenção e desenvolvimento de novos satélites são a Lockheed Martin e a Boeing, ambas norte americanas.

2.3 *WEB SERVICES*

Os *web services* foram a solução para um problema que surgiu com a vasta quantidade de softwares que foram desenvolvidos ao longo das últimas décadas. Eles visam permitir que sistemas de diferentes plataformas sejam capazes de trocar informações entre si de maneira eficiente e confiável (LEVITT, 2012).

O W3C define um *web service* da seguinte maneira:

> *Um web service é um software projetado para suportar a interação máquina-máquina um uma rede. Ele tem uma interface descrita em um formato compreensível por máquina (especificamente WSDL). Outros sistemas interagem com o web service de um modo prescrito por sua descrição usando mensagens SOAP, tipicamente transmitidas usando HTTP com uma serialização XML em conjunção com outros padrões relacionados a Web (W3SCHOOLS, 2012).*

Para permitir a comunicação entre aplicativos heterogêneos com o uso de *web services* foi necessário padronizar a forma com que esses softwares se comunicam. As tecnologias a seguir servem de base para os *web services (LEVITT, 2012)*:

2.3.1 XML

XML é sigla para *Extensible Markup Language*. Conforme visto em Quin (2012), é um formato texto extremamente flexível usado principalmente, mas não exclusivamente, para representar dados. Assim como HTML, XML também é uma linguagem de marcação, porém suas *tags* devem ser definidas pelo utilizador.

Apesar de ser uma linguagem, o XML não executa nada. Sua função não é apresentar conteúdo como o HTML faz nem processar informações como as linguagens de programação convencionais fazem. Ele foi criado para estruturar, representar e transportar informações.

Os dados representados em XML possuem um formato hierárquico. Um nó pai pode conter quantos nós filhos forem necessários para representar o modelo desejado. Os nós filhos por sua vez podem ser pais de outro nós. Em sua definição os nós podem também informar atributos, caso esses sejam necessários. A Figura 1 mostra uma estrutura XML simples para representar uma livraria.

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Figura 1 - Exemplo de XML. Fonte: W3Schools

2.3.2 SOAP

O *Simple Object Acess Protocol* (SOAP) é um protocolo baseado em XML e tem o objetivo de permitir que aplicações escritas em diferentes linguagens rodando em diferentes plataformas sejam capazes de se comunicar. Antes dele esse processo era feito através do uso de tecnologias como o CORBA e RPC, conforme visto em Levitt (2012), mas o protocolo HTTP não é capaz de suportá-los.

O SOAP é composto por três partes (W3SCHOOLS, 2012), sendo elas:

- *Envelope*: Descreve o que está na mensagem, quem deve tratá-la e quando é opcional ou não.
- *Encoding rules*: Define o mecanismo de serialização que pode ser usado para representar instâncias de tipos de dados das aplicações.
- RPC: Define uma convenção que pode ser usada para definir métodos remotos e suas respostas.

Como visto em W3schools (2012), uma mensagem SOAP é um XML composto por um envelope SOAP, que é obrigatório e também o elemento no topo da mensagem, um *header* opcional, e o SOAP body, que é obrigatório.

- SOAP Envelope: É o elemento do topo da mensagem e é responsável por abrigar todos os demais nós da mensagem bem como especificar que a mensagem é do tipo SOAP, através dos *namespaces* já citados.
- SOAP Header: É o primeiro nó filho do envelope e é usado para definir como a mensagem deve ser tratada. Contém informação específica da aplicação como autenticação.
- SOAP Body: É o nó que contém a informação da mensagem que é trocada entre os envolvidos. Pode conter nós específicos do *namespace* da aplicação e também nós responsáveis por indicar falhas no processo.

2.3.3 WSDL e UDDI

WSDL é o acrônimo para *Web Services Description Language*. É um formato baseado em XML e seu objetivo é descrever as funcionalidades de um serviço (CHRISTENSEN, 2012).

Em resumo um WSDL é um conjunto dos elementos apresentados acima aninhadas dentro de um nó *definitions*. O WSDL do Apêndice A é usado neste trabalho e é o descritor do serviço responsável pela comunicação dos agentes móveis com o servidor e vice-versa.

O UDDI por sua vez é o acrônimo para *Universal Description, Discovery and Integration*, conforme visto em W3schools (2012). É um serviço de diretório no qual empresas podem registrar ou buscar por determinado serviço sem necessariamente possuir sua URI. Ele permite buscas mais amplas, como por exemplo localizar os serviços que executem o processamento de passagens aéreas.

O UDDI surgiu em 2000 e foi iniciativa da Microsoft, IMB e Arriba. Ele faz parte dos padrões da tecnologia de *web services* e juntamente com o SOAP e o WSDL torna possível a descrição, descoberta e comunicação de um servico através da Internet. Hoje a iniciativa conta com a presença de grandes empresa do setor de tecnologia, como HP e Oracle.

3. SOLUÇÃO

Conforme planejado, o desenvolvimento do artefato foi divido em três partes, sendo elas:

1. Módulo de captura de informações
2. Servidor
3. *Front-end*

Toda a comunicação entre o servidor e os dispositivos móveis foi feita através de *web services* no servidor, bem como os cálculos necessários para determinar se a rota em questão está engarrafada.

A arquitetura macro do sistema é a seguinte:

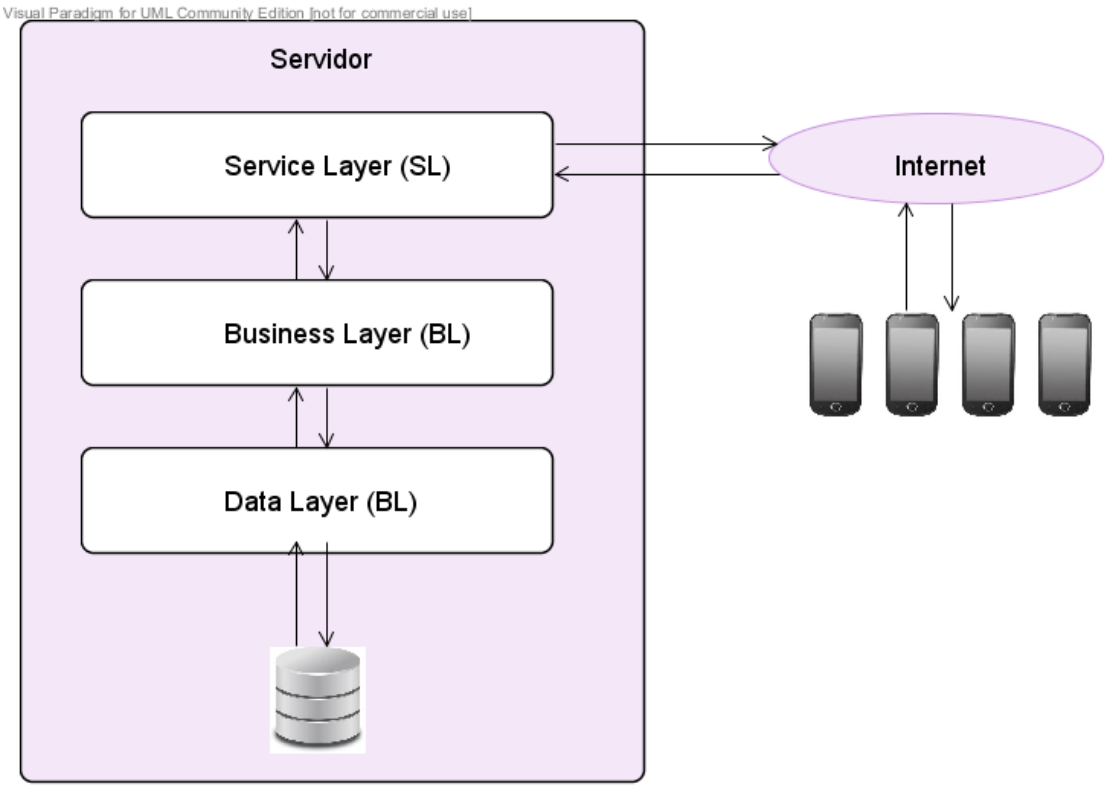

Figura 2 - Arquitetura da solução. Fonte: Autor

O artefato tem como objetivo principal a detecção de rotas engarrafadas através do processamento de pacotes que contêm a localização dos dispositivos móveis, enviados em curtos intervalos de tempo.

3.1 PROCESSO

Quando uma requisição de um cliente chega ao servidor e posteriormente ao *layer* de negócios para processamento da localização do cliente os seguintes passos são executados:

1. Axis recebe a requisição SOAP
2. Axis processa a mensagem e entrega para o SL
3. BL autentica o id do dispositivo móvel recebido
4. Caso o dispositivo seja válido o servidor salva a localização do cliente e salva um *log* da requisição
5. SL retorna um *boolean* = *true* para o cliente

O processo de recuperação das estatísticas de tráfego contidas no servidor segue um processo bastante semelhante:

1. Axis recebe a requisição SOAP
2. Axis processa a mensagem e entrega para o SL
3. BL autentica o id do dispositivo móvel recebido
4. BL recupera as estatísticas para a área do cliente
5. BL calcula velocidade média na área baseado em informações enviadas por todos os clientes.
6. BL monta XML com *markers* dos pontos engarrafados
7. SL retorna XML para o cliente

3.2 SERVIDOR

O Servidor efetua toda a sua comunicação com os dispositivos móveis através de *web services* utilizando o protocolo SOAP. Ele é responsável por todo o processamento necessário para atingir o objetivo do sistema.

O canal de entrada para os pacotes vindos dos dispositivos móveis são os serviços disponíveis no SL (*Service Layer*). Ele disponibiliza *webservices* operando com protocolo SOAP responsáveis pela interface do servidor com os clientes.

As chamadas recebidas no SL são então encaminhadas para o BL (*Business Layer*). Neste *layer* o servidor efetua todo o processamento em nível de regras de negócio para determinar se a rota atual do cliente está ou não engarrafada. É nele também que é montado o pacote de resposta para o dispositivo móvel.

No *layer* mais baixo da aplicação, o DL (*Data Layer*), é onde ocorre a persistência dos dados recebidos e processados na base de dados do sistema. Esse *layer* também é o responsável por disponibilizar para os *layers* superiores as informações contidas na base de dados.

3.2.1 SL – *Service Layer*

O *layer* de serviços está no topo da arquitetura do servidor. Ele é a interface para os agentes móveis. Este *layer* expõe a camada de negócios através de *web services* usando a *engine* Axis2.

De modo a reduzir o acoplamento dos agentes móveis com o servidor todos os parâmetros de entrada bem como os objetos de retorno do serviço foram desenvolvidos como tipos primários ou através de classes da própria linguagem (*java.util.Date* etc.), no caso Java.

Abaixo a especificação do serviço descrito.

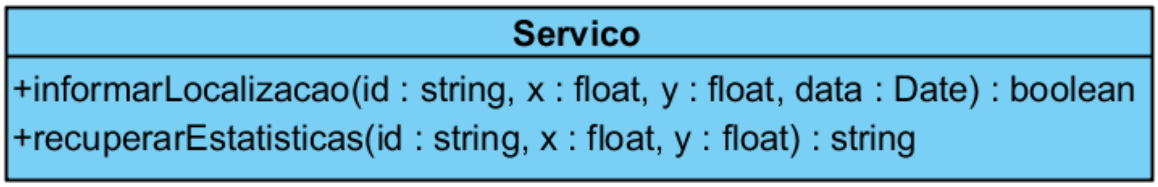

Figura 3 - Classe de serviço. Fonte: Autor

Antes de serem executadas, todas as requisições recebidas são logadas no banco de dados para permitir a rastreabilidade das requisições bem como possibilitar a extração de informações estatísticas sobre o uso dos serviços.

3.2.2 BL – *Business Layer*

O *layer* de negócios é o responsável pela execução de toda a lógica de negócio da aplicação. Suas funções incluem:

- Processamento das localizações dos clientes
- Interface entre o SL e o DL
- Cálculo de velocidade média baseado nas informações enviadas pelo cliente
- Cruzamento das informações enviadas pelos clientes para detectar engarrafamento de uma rota

Abaixo uma lista das classes que compõe o BL

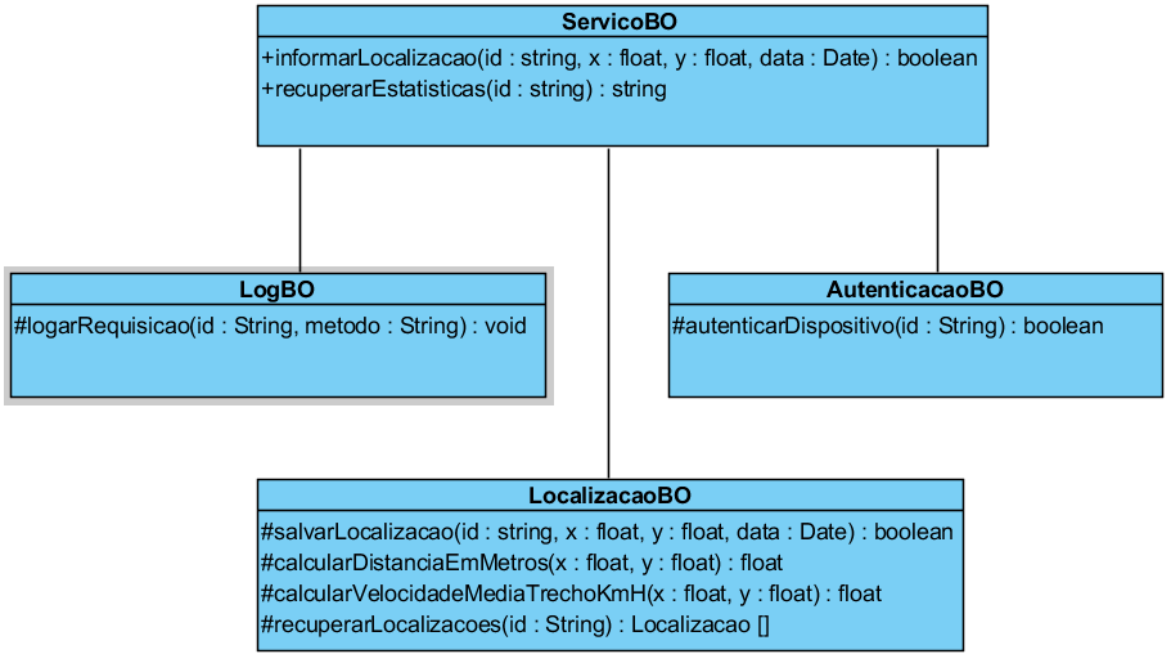

Figura 4 BL - *Business Layer*. Fonte: Autor

3.2.2.1 ServicoBO

Esta classe é a porta de entrada para o *layer*. Ela recebe todas as solicitações vindas do SL e processa as informações acessando as demais classes do *layer*.

3.2.2.2 LogBO

Esta classe é responsável por logar todas as requisições recebidas pelo servidor vindas do *layer* de serviço.

3.2.2.3 LocalizacaoBO

Esta classe é responsável pelo processamento referente aos cálculos de localização. Ela é a interface do *layer* de negócios com o DL no que se refere à persistência de informações de localização.

3.2.2.4 AutenticacaoBO

É a classe do SL responsável por autenticar um dispositivo móvel através do ID recebido pelo SL.

3.2.3 DL – *Data Layer*

O layer de acesso a dados é o responsável por fazer a interface entre o *layer* de negócios com a base de dados do sistema.

O modelo de dados é o seguinte:

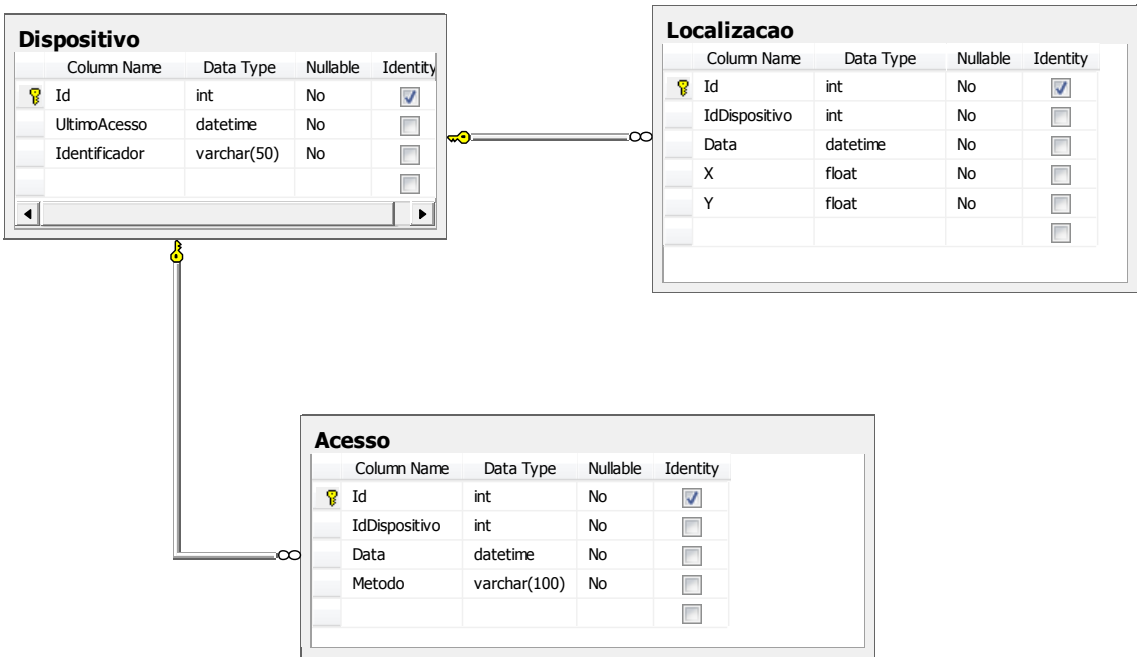

Figura 5 - Modelo de dados. Fonte: Autor

A tabela Dispositivo é a responsável por armazenar os dados dos dispositivos móveis que acessam o sistema. Seu principal uso é a autenticação dos dispositivos que fazem chamadas ao SL.

Coluna	Descrição
Id	Identificador único do banco de dados
UltimoAcesso	Ultimo acesso feito pelo dispositivo ao sistema
Identificador	Identificador do dispositivo. É um código formado a partir de algumas características dos dispositivos

Tabela 1 - Tabela Dispositivo. Fonte: Autor

A tabela Acesso é a responsável por armazenar os logs de acesso dos dispositivos ao sistema. Através dessa tabela podem ser retiradas estatísticas de uso do sistema

Coluna	Descrição
Id	Identificador único do banco de dados
IdDispositivo	Chave para a tabela Dispositivo. Identifica qual dispositivo efetuou o acesso.
Data	Data e hora da ação
Metodo	Método chamado pelo dispositivo. Por exemplo: +informarLocalizacao(id : string, x : float, y : float, data : Date) : boolean

Tabela 2 - Tabela Acesso. Fonte: Autor

A tabela Localização é a responsável por armazenar as informações de localização enviadas dos dispositivos móveis. Através dos dados armazenados nessa tabela é possível efetuar os cálculos necessários para atingir o objetivo do sistema.

Coluna	Descrição
Id	Identificador único do banco de dados
IdDispositivo	Chave para a tabela Dispositivo. Identifica qual dispositivo efetuou o acesso
Data	Data e hora da ação
X	Coordenada X do dispositivo na data e hora informada
Y	Coordenada Y do dispositivo na data e hora informada

Tabela 3 - Tabela Localização. Fonte: Autor

Essas tabelas são representadas no sistema através das classes do pacote *model* contidas dentro do DL, conforme visto na Figura 6.

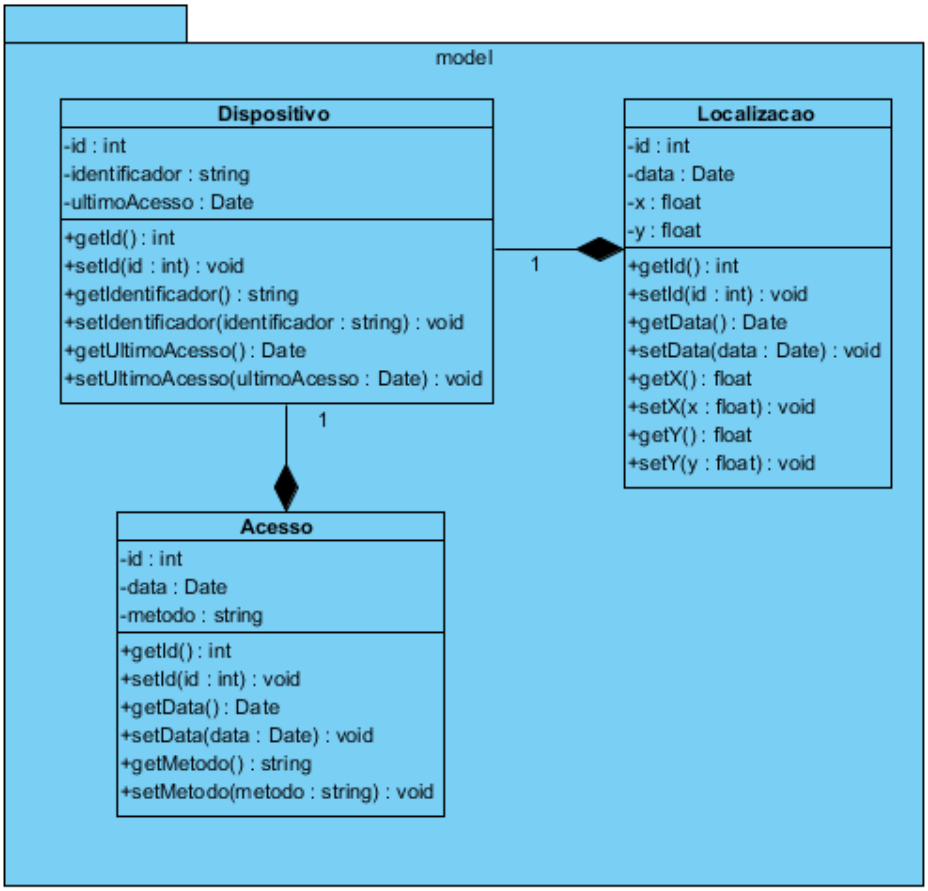

Figura 6 - Pacote *model*. Fonte: Autor

3.2.3.1 Acesso aos dados

Para acessar os dados armazenados no banco de dados foi usado o *framework* Hibernate. A classe responsável pelo acesso aos dados é a seguinte:

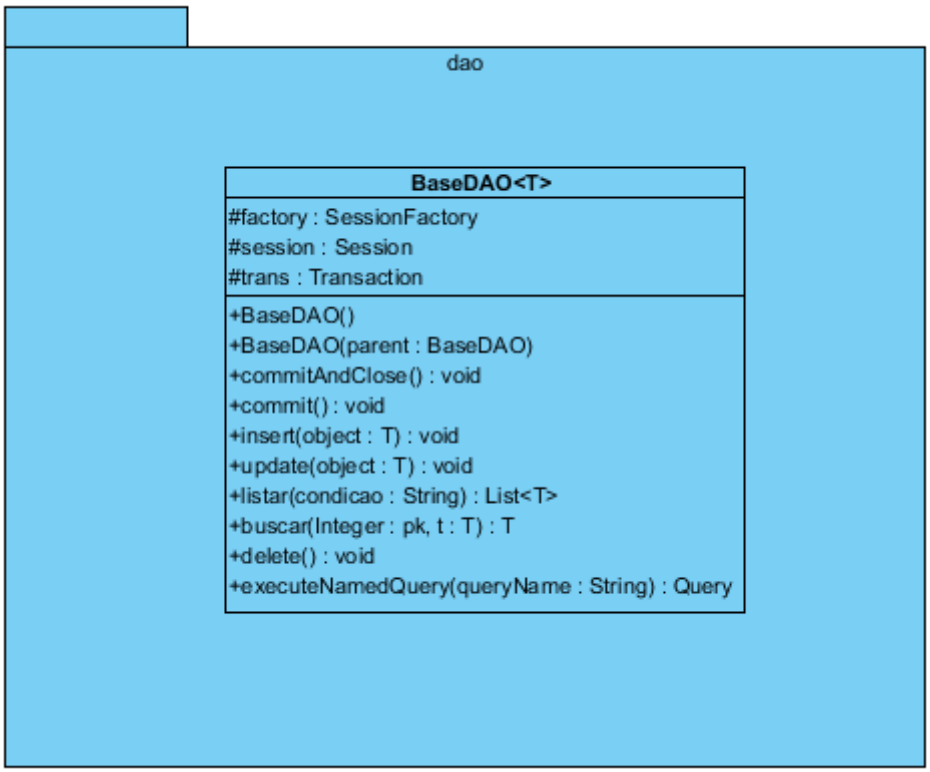

Figura 7 - Classe responsável pelo acesso aos dados. Fonte: Autor

Esta classe, através do uso de *generics*, é capaz de executar todas as operações necessárias pelo servidor contra a base de dados. Deste modo foi reduzida a necessidade de codificar uma classe DAO para cada entidade do banco e as classes do BL só precisam especificar a classe de destino no momento de criar uma instancia da DAO. Por exemplo, para acessar a tabela Acesso, a seguinte BaseDAO é criada:

BaseDAO<Acesso> daoAcesso = new BaseDAO<Acesso>();

A partir do momento que a classe é instanciada todas as operações da classe são executadas como se fossem codificadas em uma classe fortemente tipada.

3.3 CLIENTE

O cliente possui uma arquitetura bastante simples. Conforme dito na apresentação deste trabalho, de modo a evitar o consumo de recursos no dispositivo

móvel, a menor quantidade possível de processamento seria feita do lado do cliente. Suas atribuições se resumem ao envio das localizações geográficas para o servidor e a exibição dos dados já tratados vindos do servidor.

A conexão com o servidor é feita através do *web service* apresentado na Figura 3. Em um primeiro momento o cliente envia para o servidor um pacote contendo sua atual localização e a data atual. No segundo momento o cliente busca no servidor o XML que contém os dados que serão mostrados no mapa para indicar ao usuário os pontos que contêm tráfego lento. Esse processo se repete em uma *thread* em *background* enquanto o sistema estiver em execução.

O diagrama de classes do cliente é o seguinte:

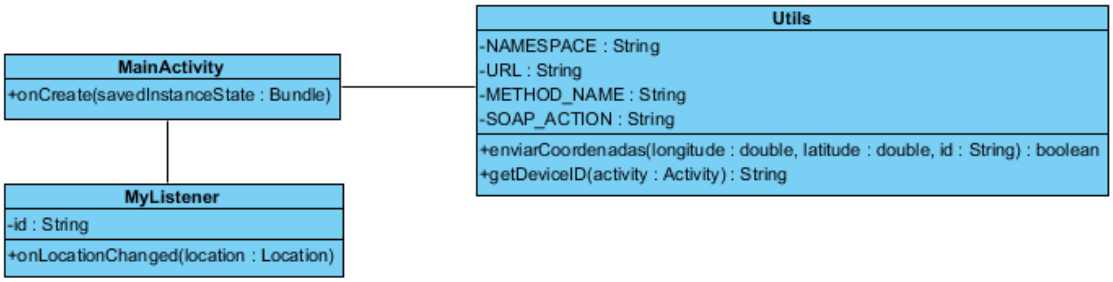

Figura 8 - Diagrama de classes da aplicação cliente. Fonte: Autor

O processo de comunicação com o servidor para o envio e recebimento é bastante simples, ele ocorre em uma *thread* auxiliar e seu diagrama de sequência está representado a seguir:

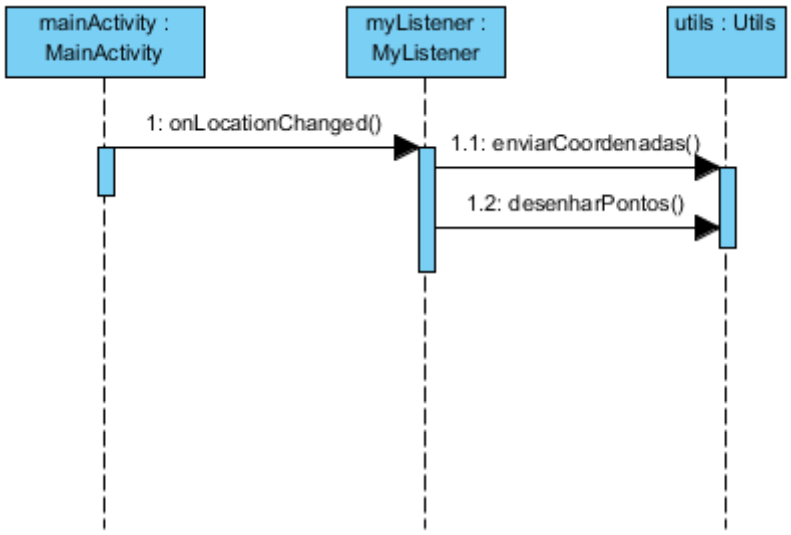

Figura 9 - Diagrama de sequência do cliente. Fonte: Autor

O disparo do evento de localização do passo um ocorre a cada cinco segundos. Este tempo é configurável e não necessariamente necessita ser tão curto. Vale lembrar que o processo de detecção da localização do cliente envolve o uso de vários sensores baseados em rádio e, portanto, é uma operação que consome uma quantidade considerável de bateria. Isso dito, esse intervalo poderia ser aumentado para cerca de cinco minutos para uso em ambiente de produção sem comprometer a usabilidade do sistema.

4. RESULTADOS

Através da solução apresentada no capítulo anterior foi possível detectar pontos de tráfego onde a velocidade média do veículo foi inferior a 20 Km/h. No cenário de teste simulado o servidor foi capaz de detectar o ponto de lentidão e montar o XML com os marcadores necessários para renderizar o mapa com a sinalização do engarrafamento, conforme visto abaixo:

```
<?xml version="1.0"?>
- <markers>
    <marker vm="14.999431195281483" y="-118.28666687011719"
      x="34.018287658691406" name="Ponto engarrafamento
      34"/>
    <marker vm="14.999431195281483" y="-118.28645324707031"
      x="34.01821517944336" name="Ponto engarrafamento 35"/>
</markers>
```

Figura 10 - XML retornado para o cliente. Fonte: Autor

No cenário mostrado na figura anterior é possível verificar que o sistema detectou um ponto de engarrafamento entre as coordenadas passadas. A velocidade média neste trecho, representada pelo elemento *vm*, foi de aproximadamente 15 Km/h.

Através deste XML o sistema móvel na ponta do processo é capaz de exibir para o usuário os pontos de engarrafamentos através de *markers*, conforme visto abaixo:

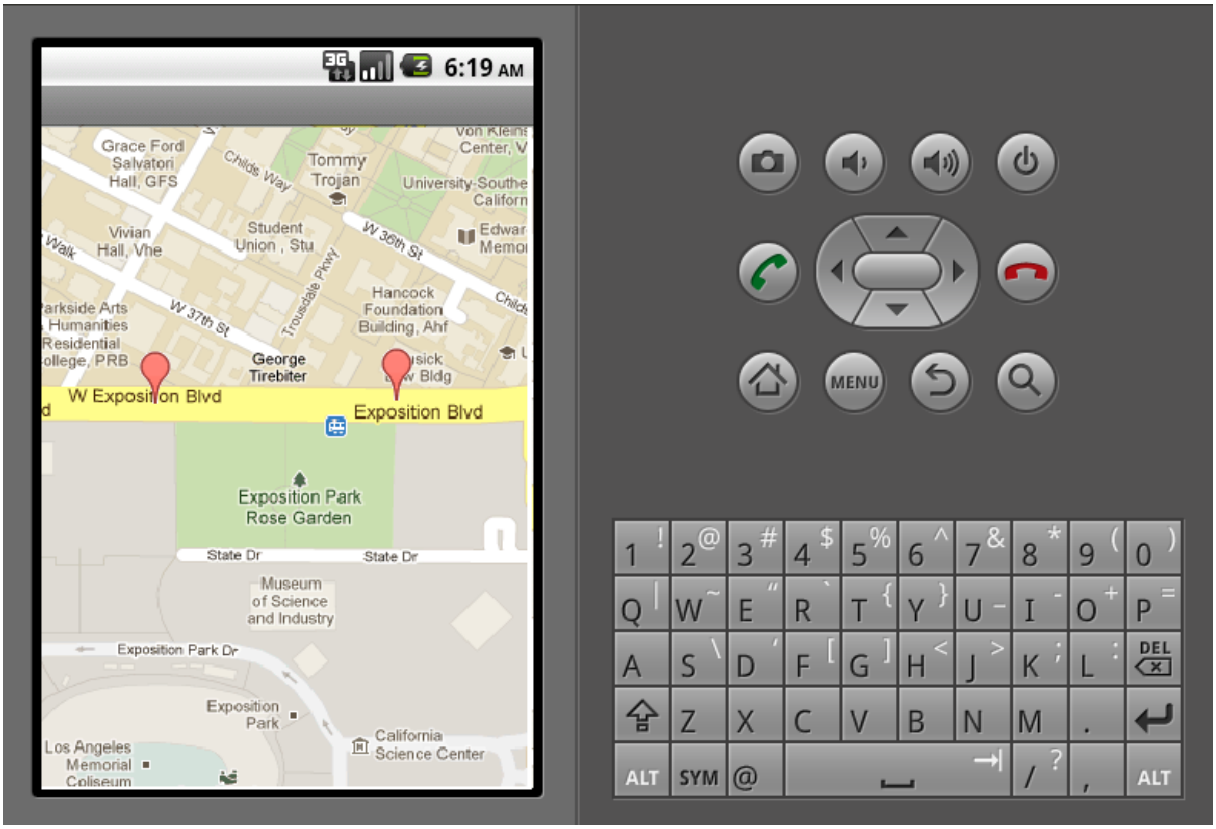

Figura 11 - *Markers* sinalizando trecho com engarrafamento. Fonte: Autor

Como os testes foram todos feitos em ambiente simulado, cabe lembrar que as informações sobre as rotas foram simuladas através de um arquivo GPX, que pode ser encontrado na Internet em diversos sites, como por exemplo em http://www.gpsvisualizer.com/examples/google_gpx.html.

Como demonstrado, o software depende de um servidor centralizado para operar e esse modelo de arquitetura gera custos de infraestrutura. Um dos modos para viabilizar economicamente esse projeto seria através do uso das informações capturadas sobre as rotas dos usuários para oferecer propagandas direcionadas. Deste modo, por exemplo, um shopping poderia ofertar produtos para clientes que costumam circular próximo ao seu endereço. Naturalmente nem todas as pessoas gostam de receber propagandas desse tipo, logo uma segunda abordagem seria a cobrança de um pequeno valor por licença de uso, que quando ativada não receberia anúncios a não ser que explicitamente solicitados pelo usuário.

Atualmente no mercado existem alguns concorrentes para a solução apresentada, destaca-se entre eles o Google, com a solução Google Maps, a Microsoft com a solução Bing Maps e a Tom Tom que apresenta software de monitoramento de tráfego embarcado em seus produtos. De modo a competir com tamanha concorrência alianças comerciais poderiam ser forjadas com empresários e governo local.

5. CONSIDERAÇÕES FINAIS

Através do sistema desenvolvido neste trabalho foi demonstrado como o uso de dispositivos móveis aliados com a vasta gama de ferramentas de desenvolvimento existentes no mercado pode ajudar pessoas a evitar o trânsito, um dos grandes problemas dos centros urbanos modernos.

Apesar de ter atingido o objetivo de informar o usuário sobre pontos de lentidão, o sistema apresentado pode ser melhorado em diversos aspectos, como por exemplo, uma melhor interface com o usuário, que poderia ser desenvolvida através de alertas sonoros e sugestão de caminhos alternativos para o usuário automaticamente evitando os pontos de engarrafamento previamente detectados e também a utilização de ruas pintadas com cores diferentes para sinalizar a intensidade do tráfego, em substituição aos marcadores.

Por fim, este trabalho, através do objeto desenvolvido, demonstrou a utilização de grande parte dos conceitos aprendidos no decorrer do curso.

6. REFERÊNCIAS

Android. **Introducing Android 4.0.** Disponível em: http://www.android.com/about/ice-cream-sandwich. Acesso em: Junho 2012

Christensen, Erik. **Web Services Description Language (WSDL) 1.1** Disponível em: http://www.w3.org/TR/wsdl. Acesso em: Junho de 2012

Kowoma. **History of navstar GPS**. Disponível em: http://www.kowoma.de/en/gps/history.htm. Acesso em: Junho de 2012

Levitt, Jason. **From EDI To XML And UDDI: A Brief History Of Web Services**. Disponível em: http://www.informationweek.com/news/6506480. Acesso em: Junho de 2012

PurpleTalk. **The Android Story.** Disponível em: http://www.xcubelabs.com/the-android-story.php. Acesso em: Junho de 2012

Quin, Liam. **Extensible Markup Language (XML).** Disponível em: http://www.w3.org/XML. Acesso em: Junho de 2012

Sugiyama, Yuki. **Traffic jams without bottlenecks - experimental evidence for the physical mechanism of the formation of a jam**. Disponível em: http://iopscience.iop.org/1367-2630/10/3/033001/pdf/1367-2630_10_3_033001.pdf. Acesso em: Junho de 2012

W3schools. **WSDL and UDDI.** Disponível em: http://www.w3schools.com/wsdl/wsdl_uddi.asp. Acesso em: Junho de 2012

W3schools. **Introduction to WSDL.** Disponível em: http://www.w3schools.com/wsdl/wsdl_intro.asp. Acesso em: Junho de 2012

W3schools. **SOAP Introduction.** Disponível em: http://www.w3schools.com/soap/soap_intro.asp. Acesso em: Junho de 2012