FRAMEWORK JAVA EM AMBIENTE DISTRIBUÍDO COM POSICIONAMENTO GLOBAL

Anhanguera Educacional S.A.

Correspondência/Contato
Alameda Maria Tereza, 2000
Valinhos, São Paulo - 13.278-181
rc.ipade@unianhanguera.edu.br
pic.ipade@unianhanguera.edu.br

Coordenação
Instituto de Pesquisas Aplicadas e Desenvolvimento Educacional - IPADE

Publicação: 09 de março de 2009

ANUÁRIO DA PRODUÇÃO DE INICIAÇÃO CIENTÍFICA DISCENTE
Vol. XI, Nº. 12, Ano 2008

Alexandre Vitorelli
Prof. Ms. Clayton Valdo
Ciência da Computação
FACULDADE POLITÉCNICA DE JUNDIAÍ
ANHANGUERA EDUCACIONAL S.A.

RESUMO

No mundo atual, a busca por informação se tornou fundamental e, em muitos casos, inevitável barreiras como distância e tempo não devem mais ser fontes impeditivas desta busca. Muitas vezes tal informação é um diferencial utilizado na execução confiável ou mesmo lucrativa de um determinado serviço. Um *framework* em ambiente distribuído com tecnologia Java e suporte a posicionamento global; fundamenta a elaboração e concepção de uma alternativa viável na tratativa de diversas soluções que atendam à questão levantada como: localização de rotas, direcionamento de chamados, compartilhamento de tarefas *online*, dentre outros. O objetivo deste trabalho foi elaborar um *framework* baseado nas premissas e tecnologias levantadas acima, bem como a implementação de uma solução em cima desta proposta.

Palavras-Chave: Sistema Distribuídos, GPS, Java ME, *Wireless*.

Trabalho realizado com o incentivo e fomento da Anhanguera Educacional S.A.

1. INTRODUÇÃO

As ferramentas de localização utilizadas por empresas cada vez mais precisam atender a serviços e processos de escala global, com velocidade, rapidez e, muitas vezes, em tempo real. Diversos segmentos como transportadoras, segurança patrimonial, engenharia de tráfego, dentre outros são usuários de ferramentas de localização para monitoração de frotas, vigilância, localização e mapeamento de rotas, entre outros serviços. Para execução de tais processos, muitas vezes, é necessária a utilização de diversos equipamentos e sistemas que podem, em muitos casos, não serem compatíveis; possuírem tecnologia ultrapassada ou necessitar ainda de interação humana, o que potencializa possíveis falhas no sistema.

Segundo Lopes (2008), "seja através de satélites GPS (*Global Positioning System*), ou de sistemas terrestres, serviços baseados em localização ou LBS (*Location-Based Services*) podem auxiliar os automobilistas, orientar os turistas, ou informar o público em geral sobre características e serviços referentes à determinada zona ou região". Além disso, é possível agregar valor aos serviços de localização, por meio de informações e serviços de orientação que auxiliem na localização de um determinado dispositivo móvel.

Vários problemas devem ser enfrentados como: diversidade de plataformas de hardware e software, controle e integração destas plataformas, localização e comunicação em locais remotos, dentre outros.

Segundo Alexandre Derani (*apud* ESTADÃO, 2008), existe um segmento em crescimento chamado *geobusiness*, que envolve planejamento de logística para empresas interessadas em serviços focados em planejamento de rotas, mapeamento e localização de cargas específicas do setor. Além disso, favorecendo a integração e crescimento deste setor, diversas empresas fabricantes de celulares estão iniciando operações comerciais de serviço de navegação no Brasil, oferecendo celulares com recursos GPS integrados. Juntamente com a plataforma JME nos celulares, com suporte à API JSR (*Java Specification Request*) 179 de localização e posicionamento, é possível realizar a integração entre aplicações e GPS.

Um *framework* em ambiente distribuído com suporte a posicionamento global (GPS) operando com tecnologia Java e Java ME traz inúmeras vantagens como a independência de plataformas, comunicação *wireless* e localização global; desta forma consegue-se a implementação de sistemas para gerenciamento e integração de recursos locais ou remotos, fornecendo, facilidades de localização, roteirizarão, emissão de alertas, dentre outros benefícios que podem auxiliar de forma eficiente às questões levantadas.

A conclusão desta pesquisa foi a contribuição na área de sistemas distribuídos, em especial para dispositivos móveis tais como: PDAs, celulares, *smart phones* e *pagers* dentre outros, com a aplicação de mecanismos de posicionamento global (GPS) em plataforma Java.

Tal proposta, aplicada ao framework, torna possível a implementação de um modelo de sistema *Wireless* com suporte a GPS para integração de aplicações em ambientes distribuídos e remotos.

2. OBJETIVO

Desenvolver um *framework* em ambientes móveis (JME) com suporte a GPS para integração e compartilhamento de processos e serviços viáveis na tratativa de diversas soluções que atendam à questão levantada como: localização de rotas, direcionamento de chamados, compartilhamento de tarefas *online*, dentre outros.

3. METODOLOGIA

Para o projeto em questão foram utilizados o embasamento teórico inicial através de leitura e pesquisa de livros, revistas e periódicos especializados, além de artigos e acesso à *Internet*.

Após a leitura e embasamento necessários, foi projetado um *framework* para atender ao projeto proposto com o uso de ferramentas case Argo UML.

Por último, foram realizadas a codificação de uma solução baseado em ferramentas *Open-Source* da tecnologia Java, através de IDE (*Integrated Development Environment*) *freeware* e emuladores para acesso remoto.

A implementação seguiu o *framework* proposto e mostra uma aplicação teste que utilizou dos recursos deste *framework* baseado nas premissas levantadas.

4. DESENVOLVIMENTO

O desenvolvimento da pesquisa está subdividido em: conceitos sobre tecnologia *Wireless*; GPS; arquitetura do *framework* e a aplicação de um sistema nesta proposta.

4.1. Tecnologias Wireless

Segundo MICHAELIS (1989), o termo *wireless* significa sistema de comunicação que não requer fios para transportar sinais. O termo *wireless* também pode ser utilizado tanto para as redes *wi-fi* como as redes *bluetooth*.

Utilizar equipamentos sem fio traz grande alivio para instalação pois não há dependência de ligações físicas para se preocupar ou se o comprimento do fio irá atingir o equipamento e segundo Tnembaum (2003) "Muitas pessoas sonhavam com o dia em que entrariam em um escritório e magicamente seu notebook se conectaria a Internet" e isso já começou a se realizar.

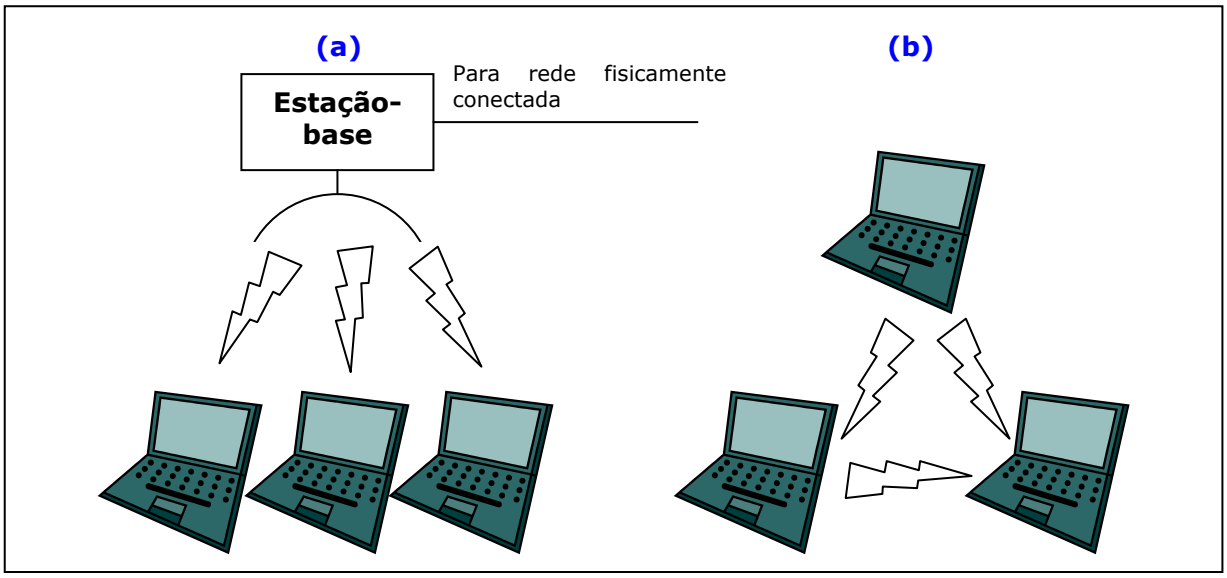
As redes *wi-fi wireless* têm como principais vantagens o acesso a longas distâncias, multiplicidade de conexões e alta taxa de velocidade – atualmente podendo atingir 108 *Mbps* (*Megabits Per Second*).

IEEE (Instituto dos Engenheiros Elétricos e Eletrônicos) padronizou as redes *wi-fi* no padrão 802.11, sendo este composto de 3 sub-padrões:

- padrão 802.11b: operam com taxas de no máximo 11 *Mbps* e a 2.4 *GHz (GigaHertz)*;
- padrão 80211a: opera com taxas de no máximo 54 *Mbps* e a 5 *GHz* e são mais resistentes à interferência comparada ao padrão anterior porém, tem um ponto negativo: o seu alcance é menor que o 802.11b e;
- padrão 802.11g: possui o melhor dos dois padrões anteriores; trafega no máximo a 54 *Mbps* e a 2.4 *GHz*. Caso existam diferentes tipos de conexões na mesma, este padrão se submete ao padrão inferior para manter compatibilidade.

As redes w*i-fi wireless* podem se comunicar de duas formas segundo Tanembaum (2003):

- Toda comunicação deve passar por uma estação base chamada *ponto de acesso* (Figura 1a) ou;
- Comunicar diretamente com outros computadores, este modo é chamado interligação de redes *ad hoc* (Figura 1b).



Fonte: (TANEMBAUM, 2006, p. 73)

Figura 1. (a) Rede sem fio com uma estação-base. (b) Rede *ad hoc*.

Além disso, as redes *wi-fi* têm três tipos de encriptação: WEP (*Wired Equivalenmnt Privacy*) de 64bits, WEP de 128 bits e WPA (*Wi-Fi Protected Access*), sendo essa última a mais segura; com a encriptação ativada o acesso é bem simples, apenas aqueles que possuírem a chave correta poderão acessar a rede.

Para uma chave WEP de 64 bits são definidos dez caracteres em hexadecimal, ou seja, 0 até 9 e de A até F e para uma chave com 128 bits são definidos vinte e seis caracteres em hexadecimal. As chaves WEP são facilmente quebradas com programas como *kismet* e o *aircrack*. As chaves WPA também conhecidas como WEP2 tem melhoria na criptografia dos dados usando TKIP (*Temporal Key Integrity Protocol*) onde as chaves são alteradas a cada novo envio.

Já as redes *bluetooth* conhecidas como WPAN (*Wireless Personal Area Network*) assim como a rede *wi-fi* foi padronizada pelo órgão IEEE no padrão 802.15. O *bluetooth* tem como principal vantagem menor consumo de energia comparado às redes *wi-fi*; sendo por esta razão, melhor aceito por aparelhos *mobile* porém, possui como desvantagens:

- alcance reduzido: máximo de 10 metros;
- limite de conexões: máximo de 8 e;
- taxa de conexão: 1 *Mbps*.

O padrão *bluetooth* 2.0 + EDR (*Enhanced Data Rate*) pode atingir uma taxa de 3*Mbps* e, se o adaptador *bluetooth* for de classe 1, pode atingir uma distância de até 100 metros.

Da mesma forma que em uma rede *wi-fi*, caso haja um padrão inferior na mesma rede, este operará em padrão inferior por questão de compatibilidade; desta forma um padrão *bluetooth* 1.2 pode se comunicar com o padrão 2.0.

4.2. GPS

GPS significa *Global Positioning System*, e é um aparelho de rádio especial que mede a localização a partir de satélites que orbitam a Terra e que transmitem essa informação por sinais rádio (MACNAMARA, 2004).

O GPS, conforme Oliveira (2006), "Consiste de três órbitas médias (aproximadamente 20.000 km) com oito satélites cada. As órbitas foram projetadas de forma que cada ponto da Terra seja capaz de "enxergar" pelo menos quatro satélites em qualquer instante do dia". Utiliza baixa freqüência; devido a diversos fatores:

- baixa perda de potência ao se deslocar no espaço;
- baixa atenuação devido à chuva;
- baixa absorção da atmosfera e;
- relativa capacidade de atravessar objetos.

Para "descobrir" a posição atual segundo OLIVEIRA (2006), "o receptor GPS sabe que cada satélite se encontra na superfície de uma esfera imaginária centrada no próprio satélite e com raio igual a distância calculada. A intersecção de duas esferas forma um círculo. O receptor então, está em algum ponto desse círculo. Se uma terceira esfera for usada, ela cortará o círculo em dois pontos, logo o receptor se encontra em algum desses dois pontos. Uma quarta esfera determina qual desses pontos é a posição correta em relação aos quatro satélites".

Uma vez obtidos os dados do GPS segundo OLIVEIRA (2006), "o mesmo deve formatá-los no padrão NMEA (*National Marines Eletronic Association*) chamado de NMEA-0183, existem atualmente vinte e seis tipos de mensagens em formatos NMEA-0183, a lógica nas mensagem segue o seguinte padrão, cada sentença começa com o símbolo $ e termina com *carrige return* e *line feed*, os dados são separados por virgula portanto os números não usam ponto e virgula para separar as casas decimais e sim o ponto". As principais mensagens em formato NMEA-0183 que são retornadas por um dispositivo GPS são:

- GPGGA (*Global Positioning System Fix Data*)
- GPGSA (*GPS DOP and Active Satellites*)
- GPGSV (*GPS Satellites in View*)
- GPRMC (*Recommended Minimum Specific GPS/TRANSIT Data*)

Uma mensagem no padrão NMEA-0183 em formato GPRMC pode ser exemplificada conforme Dale na Figura 2.

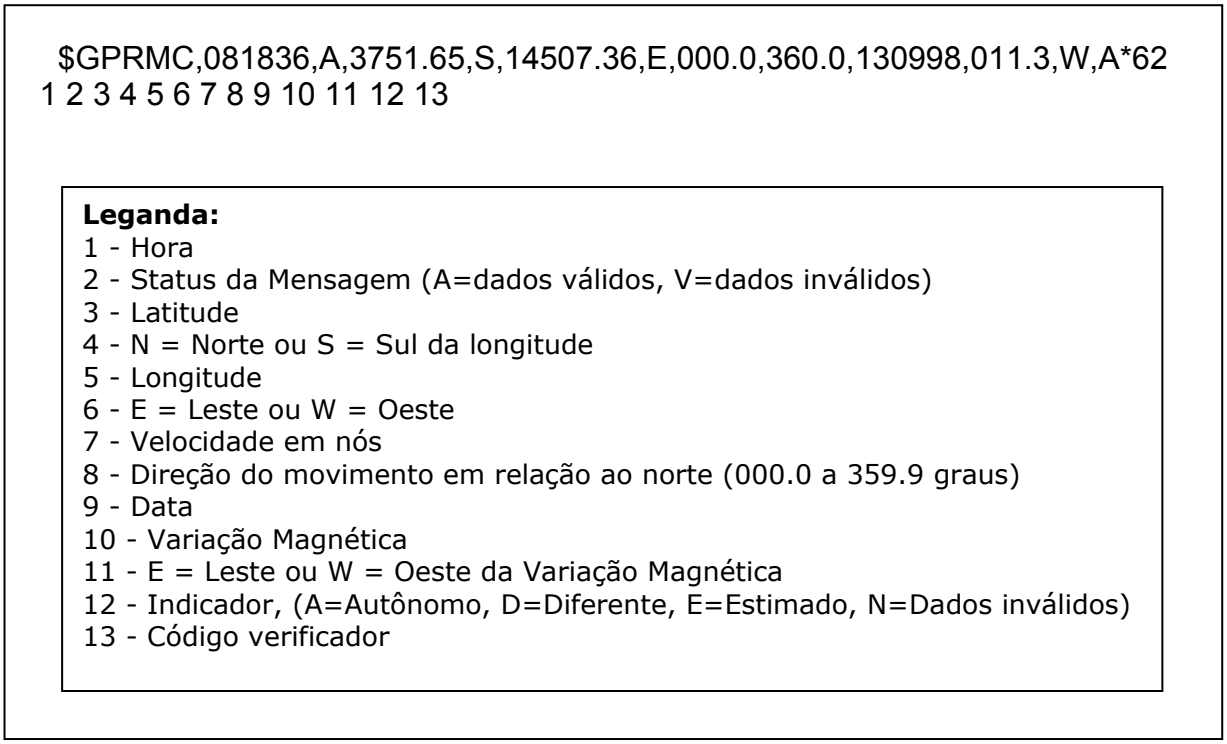


Figura 2. Mensagem no padrão NMEA-0183 em formato GPRMC.

4.3. *Framework* proposto para integração de GPS e Tecnologia Wireless

Após embasamento das tecnologias necessárias analisou-se a viabilidade técnica das mesmas e foi concebido um modelo de *framework* integrando as tecnologias e premissas do projeto proposto inicialmente que foram: integração, heterogeneidade e mobilidade.

As ferramentas usadas para a implementação do *framework* estão baseadas em tecnologias *open-source* com grande utilização no mercado: IDE *Net Beans* 6.0.1, *MySQL* 5.0 e gravação dos dados em um banco de dados foi utilizado a biblioteca *Hibernate*, facilitando a portabilidade em diversos bancos de dados.

O framework proposto foi dividido em 2 camadas: Negócio e Interface e, esta última em 2 sub-camadas: GPS e *Mobile*, conforme mostrado na Figura 3.

A Camada de Interface realiza toda a comunicação da aplicação está dividida em:

- Interface GPS: tem por finalidade encapsular uso do GPS, com busca de dados de geoposicionamento.
- Interface Mobile: tem por finalidade bibliotecas de funções em linguagem Java para facilitar o desenvolvimento como realizar um *split* em um texto, persistir e recuperar dados no dispositivo *Mobile*, recuperar o *IMEI* (*International Mobile Equipment Identity*) do dispositivo e classes de interop para uso do *web service*.

- A Camada de Negócio de negócio é a aplicação que utilizará os dados enviados por um dispositivo *Mobile*, ou seja, seguindo uma regra de negócio poderá ser gerado um relatório dizendo qual caminho percorreu uma determinada fonte ou onde ela está localizada e quanto tempo levou até seu destino; em tempo real ou não.

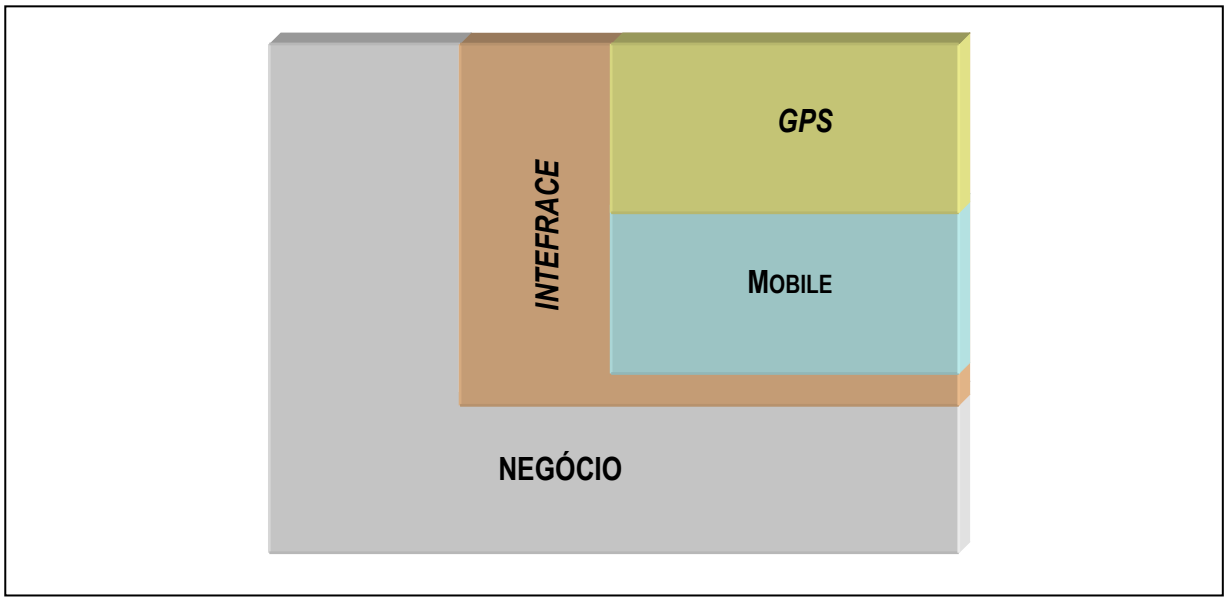


Figura 3. *Framework* proposto.

4.4. Camada de negócio

A camada de negócio se destina para uso de negócio da aplicação que utilizará o *framework*, portanto fica a critério de quem irá implementar uma aplicação baseada neste *framework,* por exemplo, na aplicação de teste foi utilizado os dados de posicionamento para encontrar um determinado dispositivo e exibi-los no mapa.

4.5. Camada de interface

A divisão em camadas foi realizada para encapsular o uso do GPS, pois futuramente se houver outro tipo de conexão com um GPS que não seja por API JSR 179, somente essa interface sofrerá alteração. A interface *mobile* tem por finalidade facilitar o desenvolvimento.

A camada de interface GPS tem como dependência CLDC 1.1 (*Connected, Limited Device Configuration*) por usar cálculos com ponto flutuante, MIDP 2.0 (*Mobile Information Device Profile*) e API JSR 179 (*Location*).

Essa interface tem como principal API a API JSR 179 de localização e de posicionamento esta API foi utilizada por sua facilidade de uso, pois sem ela o trabalho seria muito maior na hora de usar o GPS e se ao invés usar essa API fosse usado a API JSR 82 (*Bluetooth*) teria que ser realizado um processo de busca na rede *Bluetooth* chamado

inquiry, recuperar os dispositivos remotos feito pela busca, conectar no serviço *bluetooth* disponível no dispositivo remoto escolhido, abrir um *stream* para tráfegos de dados e após o término da transmissão trabalhar com a string em formato NMEA-0183 (OLIVEIRA 2008).

As mensagens em formato NMEA-0183 tem várias formatações dependendo do nível de precisão requisitado e para requisitar precisamos ter um meio de comunicação entre a aplicação e o GPS que podem ser manual usando API JSR 82 ou de forma automática API JSR 179, porém manualmente pode ocasionar duas falhas usando a API JSR 82 uma pelo *bluetooth* estar desligado necessitando de intervenção humana e a outra a busca na rede *bluetooth* seria de todos os aparelhos e poderia ser escolhido um dispositivo que não fosse um GPS, já da forma automática com a API JSR 179 quando se é requisitado a posição do GPS, a mesma informa uma mensagem amigável pedindo permissão ao usuário para ligar o *bluetooth,* caso esteja desligado, faz a busca somente por dispositivos GPS e com métodos simples como *getLatitude()* e *getLongitude()* podemos ter a localização facilmente.

A implementação usando a API JSR 179 é bem simples, primeiro foi adicionado um *import* ao *package javax.microedition.location*, criou se um objeto *Criteria* onde foi configurado o mínimo possível de informação que será retornada pelo GPS, com isso o retorno do GPS será mais rápido e conterá apenas informações que realmente será usado, o objeto *LocationProvider* com seu método *getInstance()* será responsável em se conectar ao GPS, e recuperar as informações requisitadas pelo objeto *Criteria*, o objeto *Location* contém métodos para poder recuperar a latitude e longitude, mas para criá-lo foi usado o método *getLocation* com o parâmetro *LocationProvider.AVAILABLE* do objeto *LocationProvider*.

A interface GPS tem como sua principal classe a classe GPS no package *framework.mobile.library.gps* com seu único método estático *getLocation()* no qual é responsável em retornar um objeto *Location* para uso de retorno da posição atual, já as classes dentro do package *framework.mobile.library.nmea* servem para controle da string GPRMC em formato NMEA-0183, o formato GPRMC foi escolhido por conter o mínimo de informações que realmente serão utilizadas, a classe *Position* contem dois métodos *getCalcLatitude* e *getCalcLongitude* que por sua vez calculam respectivamente a latitude e longitude, pois se no futuro for implementado a recuperação da posição manualmente por API JSR 82 já é suportado o calculo, o calculo utiliza a seguinte regra:

- Para Latitude GG + (MM.DDDD / 60) e;
- Para Longitude GGG + (MM.DDDD / 60).

Respectivamente G = graus, M = minutos e D = décimo de segundos, se as direções forem sul ou oeste os números devem ser negativos.

Para a camada de interface *Mobile* tem como dependência CLDC 1.1 por usar cálculos com ponto flutuante, MIDP 2.0 e API JSR 172 (*Web Services Specification*).

A camada de interface *Mobile* contém utilitários de uso imprescindível como *split* não suportado pelo JME, a classe do *Store* do *package framework.mobile.utils* tem como finalidade realizar o trabalho de armazenamento usando *RecordStrore* que funciona da seguinte forma; adiciona se um *import* para o *package javax.microedition.rms* cria se uma variável do tipo *RecordStore* usando o método *openRecordStore*, este método ter por finalidade abrir uma espécie de "*stream*", para gravar usa se o método *addRecord* e para fechar use o método *closeRecordStore*, para leitura apenas use o método *getRecord*. A também um utilitário para se retornar o IMEI usando a função *System.getProperty* passando como parâmetro um código disponibilizado por cada fabricante, com o IMEI é possível diferenciar cada dispositivo conectado ao sistema pois é o código físico do dispositivo.

4.6. Aplicação teste

Para fundamentação do *framework* proposto, implementou-se uma aplicação seguindo tal modelo.

Para a camada de negócio, foi implementado o uso das informações recebidas dos dispositivos *mobile* que estão na tabela *location* do banco de dados deste *framework*, o campo NMEA0183 desta tabela contém uma mensagem em formato GPRMC que para utilizá-la foi implementado o uso da classe GPRMC do *package framework.mobile.library.nmea* para separar as informações e com isso calcular a latitude e longitude usando a classe *Position*, com a latitude e longitude calculados pode se passar para uma aplicação de mapas digitalizados e com isso ver onde o dispositivo estava naquele momento.

Para a camada de interface *Mobile* foi implementado o uso das classes do *package framework.mobile.utils, Store* para armazenar dados de configurações como também usuário ou senha se o usuário permitir e *Phone* para recuperar o IMEI do dispositivo.

Também foram implementadas a classe *WebServiceService_Stub* do *package framework.client.webserviceservice* para enviar dados de posicionamento para o servidor usando os métodos *EnviarPosicao* e *getMobile*, para autenticação do sistema foi utilizado o método *Login* desta mesma classe.

Para a camada de interface GPS foi implementado o uso da Classe GPS do *package framework.mobile.library.gps*, essa classe tem a funcionalidade de conectar no GPS e retornar as informações de posicionamento, para isso foi utilizado o principal método estático dessa classe o *getLocation()* com isso a cada intervalo de tempo é possível enviar a posição do dispositivo.

A Figura 4 representa o diagrama do banco de dados usado no projeto, com as seguintes tabelas:

- *Login*: contém informações de usuário e senha que podem acessar o sistema.
- *Location*: contém dados de posicionamento enviado em um determinado intervalo de tempo, o campo NMEA0183 contém uma mensagem em formato GPRMC contendo latitude e longitude.
- *Mobile*: contém informações sobre os dispositivos utilizados no sistema, com isso é possível identificar cada um dentro do sistema.

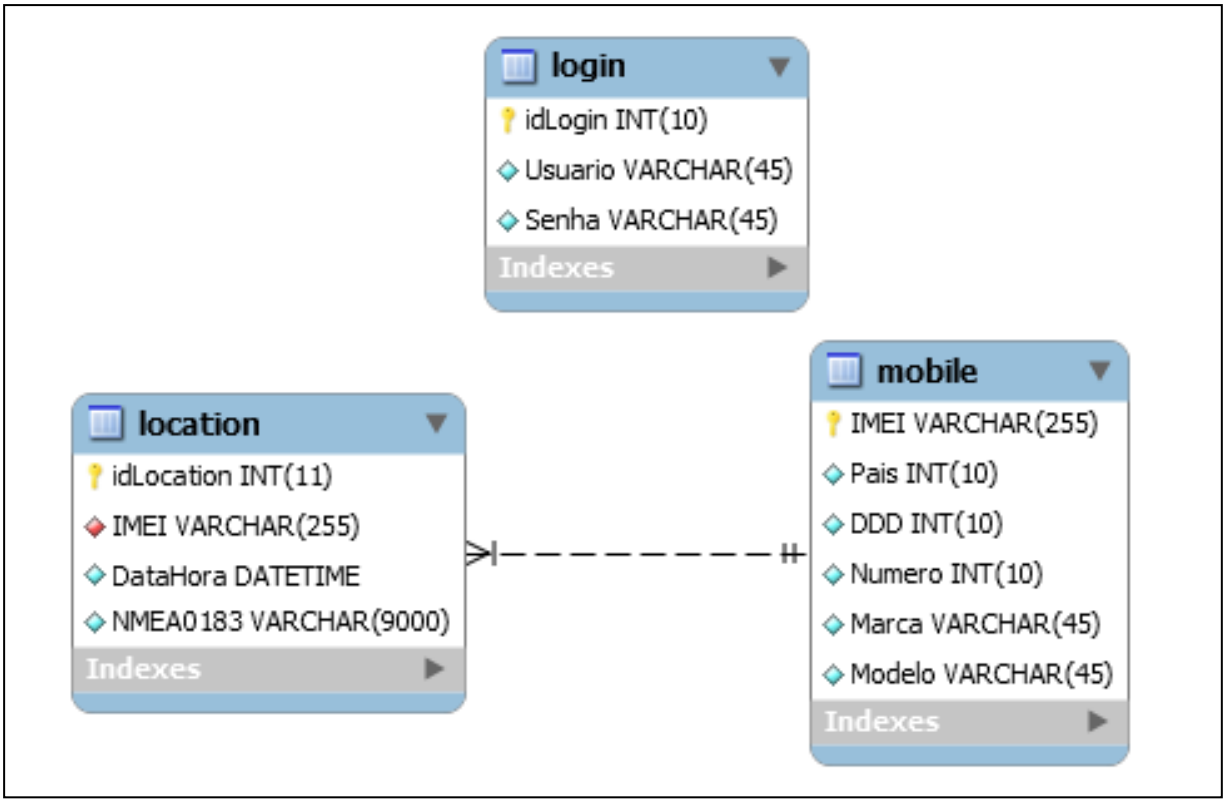


Figura 4. Modelagem do Banco de Dados

A Figura 5 representa a classe principal do sistema. Foi criado um *MIDlet*, que foi adicionado uma referência ao *framework* que é capaz de obter a posição, enviar dados para o servidor, realizar o *login* e tratar os dados do GPS.

Esta classe foi isolada no projeto de forma que pudesse ser referenciado tanto em um *Web Service* como em um *MIDlet*.

As Figuras 6, 7, 8, 9 e 10 são as telas da aplicação de teste em funcionamento.

A Figura 6 representa a tela de *login*. O item menu na tela do celular permite acesso à tela de configurações e ao botão "Entrar" onde se realiza o *login* efetivamente, caso o retorno do *login* seja falso uma mensagem de usuário e senha invalido é mostrada na tela, caso seja retornado verdadeiro, é passado para a tela de envio de posição.

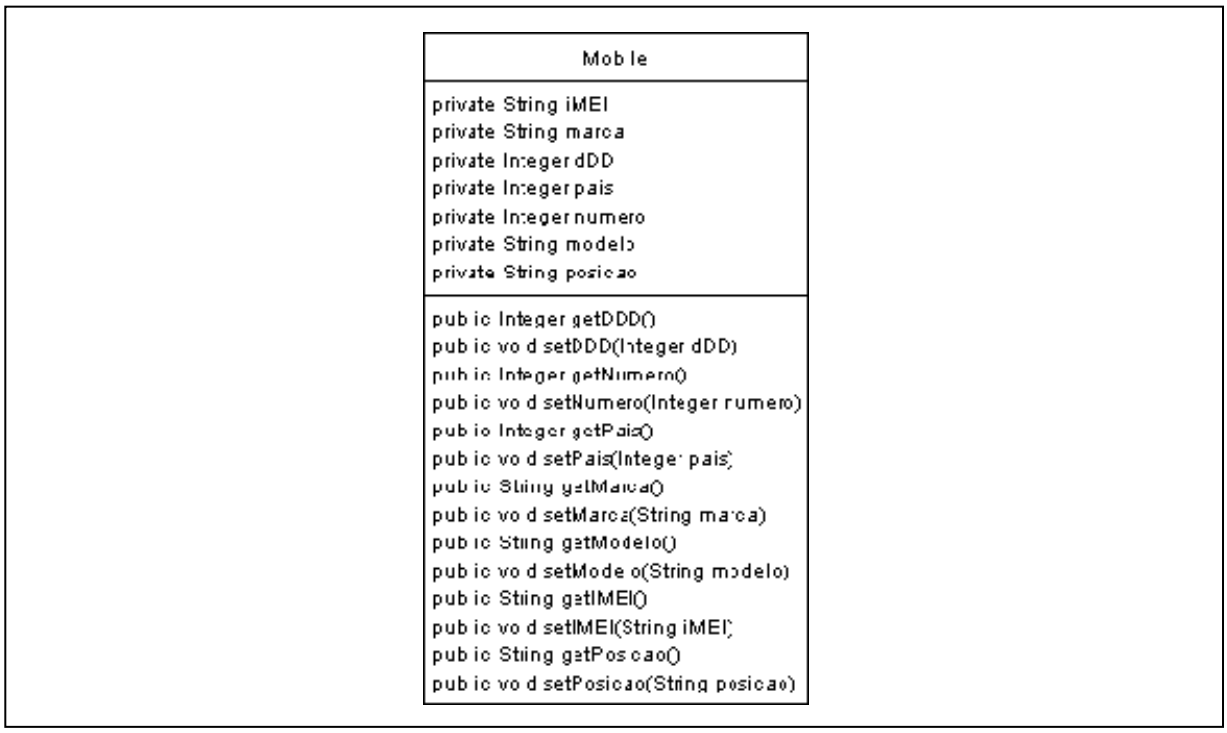


Figura 5. Classe principal do sistema.

As opções "Lembre-me" e "Salvar senha" são para uso cômodo do usuário caso o mesmo não queira digitar usuário ou senha pela segunda vez.

A Figura 7, exibe a tela de configurações de *Web Service*, os parâmetros que aparecem nessa tela URL (*Uniform Resource Locator*) é o endereço físico do servidor onde o *Web Service* está hospedado, o parâmetro IMEI é o código físico do dispositivo que nunca se repete este parâmetro é apenas informativo, ou seja, somente leitura.

A Figura 8 exibe a tela que é feito o envio da posição a cada intervalo de tempo, caso ocorra um erro uma mensagem é exibida ao usuário informando o que ocorreu. O botão "*Log Off*" cancela o envio da posição e volta para a tela de *login*.

As Figuras 9 e 10 exibem uma implementação de aplicação de negócio que exibi em uma página de internet a última posição dos dispositivos; o botão "busca" ao lado de cada dispositivo mostra a posição do mesmo no mapa.

Figura 6. Tela de *Login*.

Figura 7. Tela de Configuração.

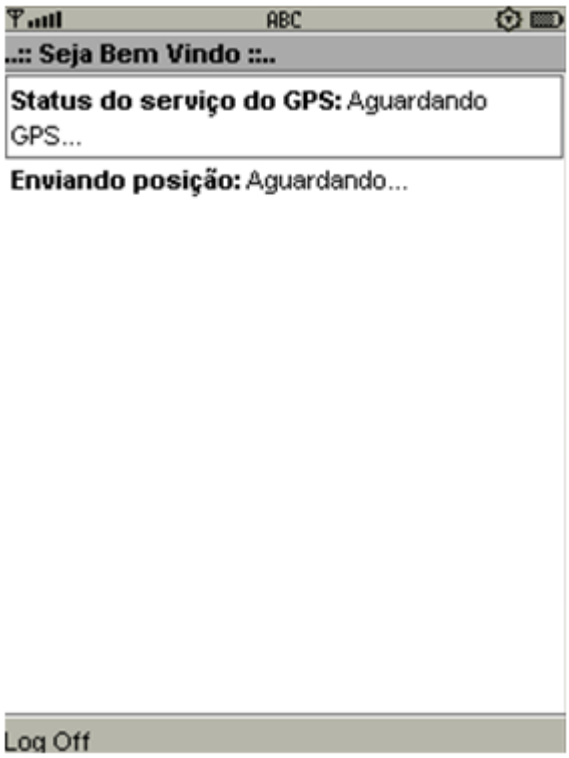


Figura 8. Tela de envio da posição.

5. RESULTADOS

Para os testes do *framework* foram utilizados um *smartphone* Nokia E65, um módulo GPS com suporte a *bluetooth* e padrão NMEA-0183 Nokia LD-3W, um ponto de acesso *wireless* e um computador para hospedagem dos *Web Services*.

Depois de realizado toda a instalação: servidor, *smartphone* e módulo GPS no celular, utilizando o manual do fabricante; foi realizado o processo de comunicação entre o *smartphone* com o GPS por *Bluetooth* e *smartphone* com o servidor por *wireless*.

A camada do GPS foi testado usando a API JSR 179; diversos problemas foram enfrentados, principalmente pela interferência; segundo Nokia (2006), "o módulo GPS pode ter interferência senão estiver em céu aberto onde a antena GPS pode se sintonizar". Tal problema foi solucionado seguindo tais recomendações; com isso, a conexão com o satélite foi estabelecida e a aplicação funcionou perfeitamente, com tráfego dos dados e gravação no servidor através do *Web Service*.

Além disso, foi realizada a movimentação do módulo GPS e conseguiu-se a mudança dos pontos de localização, conforme indicado nas Figuras 9 e 10.

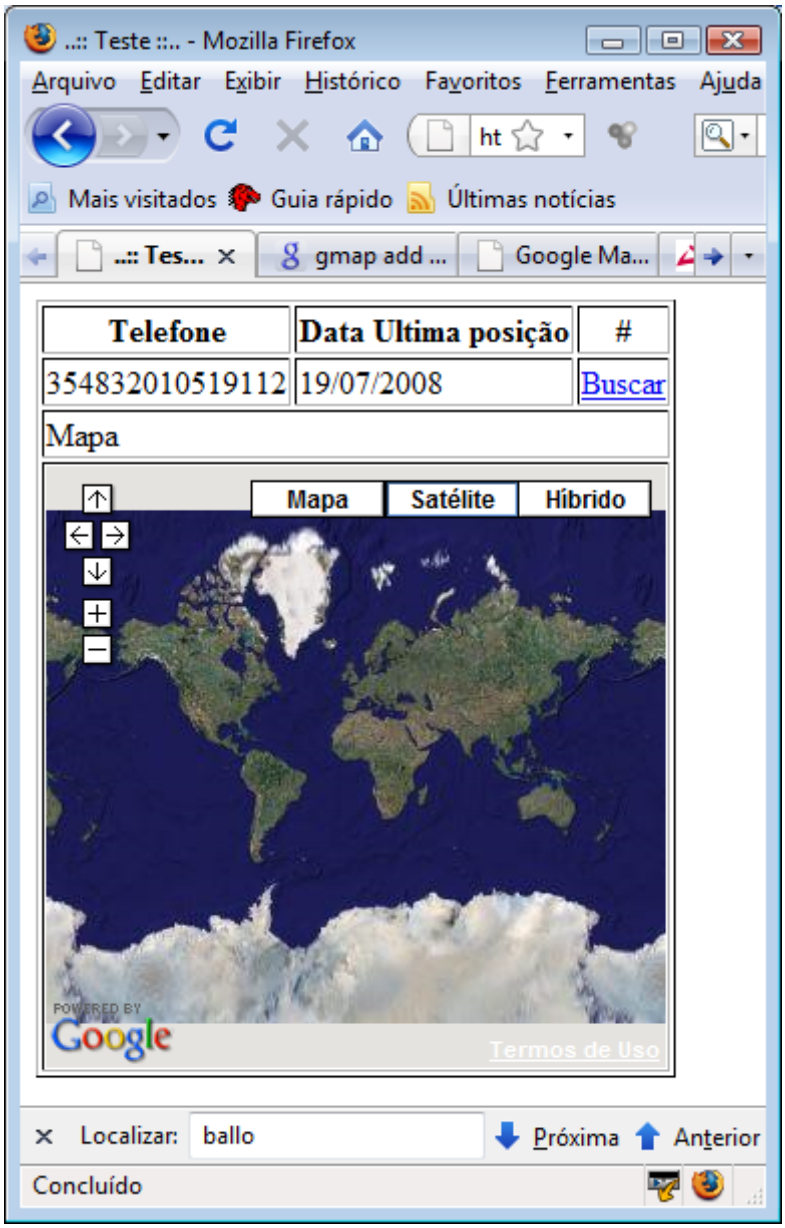


Figura 9. Tela Inicial da Aplicação de Negócio.

A aplicação funcionou perfeitamente no *smartphone*, pois o *framework* apenas utiliza API's JSR padronizadas isso garante compatibilidade entres outros dispositivos que suporte as API's JSR utilizadas neste *framework*.

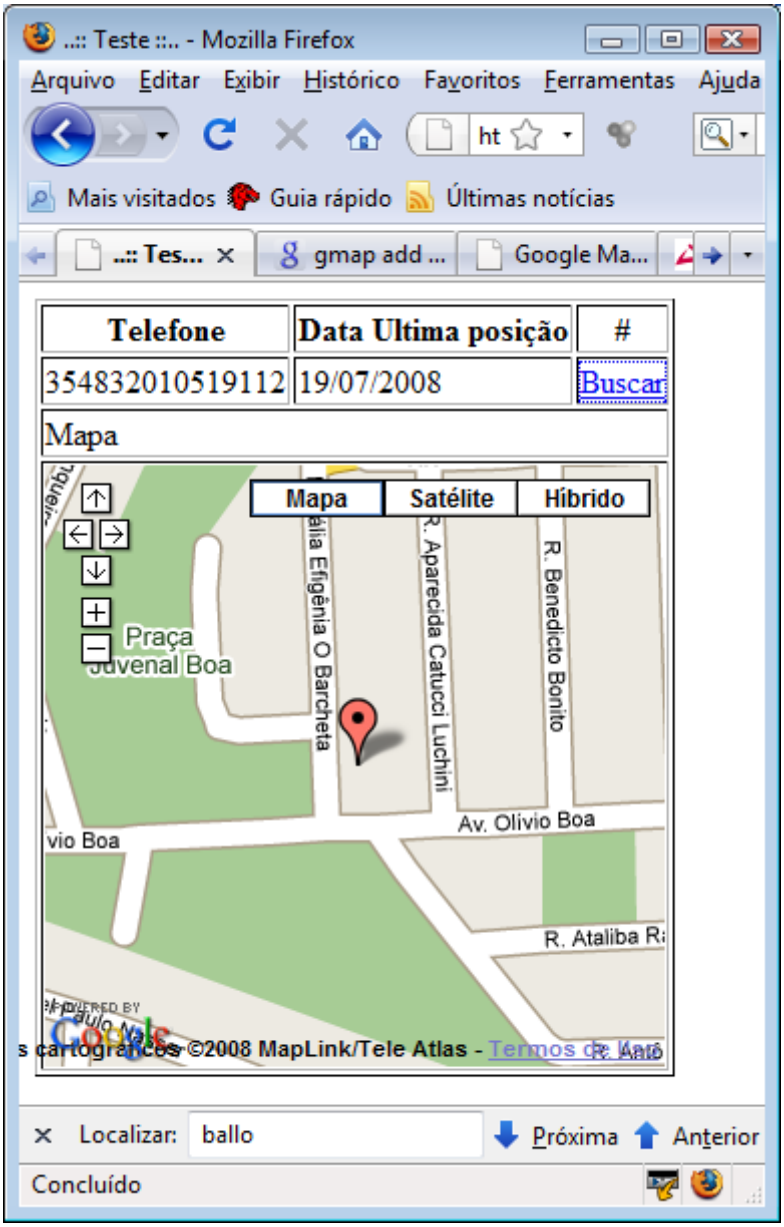


Figura 10. Tela com mapa da posição do dispositivo.

6. CONSIDERAÇÕES FINAIS

O mercado para o chamado *geobusiness* está em franco crescimento; cada dia, novas aplicações de negócio e tecnologias surgem crescendo seu mercado e agregando novos serviços e estratégias.

Tal crescimento é devido, principalmente pela popularização e baixa de preços dos dispositivos GPS, em especial os *smartphones* e aparelhos celulares com este recurso incorporado; o que aumenta pela demanda de serviços e aplicações de forma preponderante.

A busca por um aplicativo portável para qualquer ambiente e *open-source* para criação de tais aplicações possibilita a rápida popularização e evolução com a portabilidade e baixo custo inerentes.

Este trabalho mostra a concepção de um *framework* genérico com tecnologia Java com biblioteca GPS, agregando portabilidade e baixo-custo na criação de aplicações diversas para o mercado de *geobusiness*.

Tal proposta foi testada com a implementação de uma aplicação teste para busca e mapeamento de um ponto no celular com GPS.

Empresas que precisam utilizar de ferramentas de localização, porém não tem recursos financeiros para tal, pode utilizar de tecnologias *open-source* para atender seus serviços e processos em escala global e ainda reduzir custos.

REFERÊNCIAS

ESTADÃO: **Preço menor abre caminho para aparelhos de GPS no País, 2008**. Disponível em: <http://www.estadao.com.br/tecnologia/not_tec201958,0.htm>. Acesso em: 08 jul. 2008.

LOPES, Eli. **LBS com a facilidades da API Location (J2ME)**: Proposta de Arquitetura. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=5356>. Acesso em: 30 maio 2008.

MCNAMARA, Joel. **GPS for Dummies**. 1. ed., EUA: John Wiley Consumer. 2004.

MICHAELIS, Michael C., et al. **Dicionário Michaelis**: pequeno discionário Inglês - Português, Português - Inglês. 70. ed. rev. São Paulo: Cia de Melhoramentos de São Paulo, 1989.

NOKIA: Nokia Wireless GPS Module (LD-3W) User Guide, USA, 2006.

OLIVEIRA, Daniel da Silva. JavaME usando GPS. **Revista WebMobile**. , 11. ed., ano 02, Rio de Janeiro: DevMedia Group, p. 20-29, 2006.

TANENBAUM, Andrew S. **Redes de Computadores**. 4. ed., São Paulo: Campus, 2003.

DEPRIEST, Dale. **NMEA data**. Disponível em: <http://www.gpsinformation.org/dale/nmea.htm>. Acesso em: 08 jul. 2008.