UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VISUALIZAÇÃO GENÉRICA DE SINAIS E DADOS BIOMÉDICOS EM DISPOSITIVOS MÓVEIS

RAFAEL SANTOS ORTIS

ORIENTADOR: HERVALDO SAMPAIO CARVALHO

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM - 411/09

BRASÍLIA/DF: DEZEMBRO – 2009

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VISUALIZAÇÃO GENÉRICA DE SINAIS E DADOS BIOMÉDICOS EM DISPOSITIVOS MÓVEIS

RAFAEL SANTOS ORTIS

DISSERTAÇÃO DE MESTRADO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

Prof. Hervaldo Sampaio Carvalho, Doutor (FM/ENE/UnB)
(Orientador)

Prof. Adson Ferreira da Rocha, Doutor (ENE/UnB)
(Examinador Interno)

Prof. Paulo César de Jesus, Doutor (FM/UnB)
(Examinador Externo)

BRASÍLIA/DF: 18 DE DEZEMBRO DE 2009

FICHA CATALOGRÁFICA

ORTIS, RAFAEL SANTOS

Visualização genérica de sinais e dados biomédicos em dispositivos móveis [Distrito Federal] 2009.

xvii, 62p., 210 x 297 mm (ENE/FT/UnB, Mestre, Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

1. Visualização de sinais 2. Visualização genérica

3. Dispositivos móveis 4. SOA

I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

ORTIS, R. S. (2009). Visualização genérica de sinais e dados biomédicos em dispositivos móveis. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-411/09, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 62p.

CESSÃO DE DIREITOS

AUTOR: Rafael Santos Ortis.

TÍTULO: Visualização genérica de sinais e dados biomédicos em dispositivos móveis.

GRAU: Mestre ANO: 2009

_____

Rafael Santos Ortis

SHCGN 710 Bloco N Casa 13, Asa Norte.

70750-744 Brasília – DF – Brasil.

Dedicado a minha família e aos meus amigos.

AGRADECIMENTOS

Agradeço a Deus pela minha vida, por me ajudar e sustentar nas etapas deste trabalho.

Agradeço ao Departamento de Engenharia Elétrica por permitir a conclusão deste trabalho.

Agradeço ao professor Hervaldo Sampaio Carvalho que acompanhou e orientou o desenvolvimento deste trabalho.

Agradeço a Lara Vida por sua paciência, carinho e compreensão que foram essenciais no final deste trabalho.

Agradeço também aos meus amigos e familiares que me ajudaram no decorrer deste projeto.

RESUMO

VISUALIZAÇÃO GENÉRICA DE SINAIS E DADOS BIOMÉDICOS EM DISPOSITIVOS MÓVEIS

Autor: Rafael Santos Ortis

Orientador: Hervaldo Sampaio Carvalho

Programa de Pós-graduação em Engenharia Elétrica

Brasília, dezembro de 2009

O objetivo deste trabalho é o desenvolver de um sistema para a visualização móvel de sinais biomédicos de forma genérica e abrangendo a maior parte dos dispositivos móveis utilizando J2ME. O sistema proposto acessa os dados e sinais biomédicos de pacientes utilizando a arquitetura SOA. A visualização de sinais proposta neste trabalho se mostra eficiente para os dispositivos com resolução de tela pequena. A implementação de comunicação baseado em WebServices também garante maior interoperabilidade do sistema desenvolvido com diversos outros sistemas. A compressão dos dados no acesso ao serviço web mostra que é possível ter economia considerável (30%) na transmissão de dados. O desenvolvimento deste trabalho criou a base de um sistema genérico para a visualização de dados em dispositivos móveis.

ABSTRACT

BIOMEDIC DATA VISUALIZATION ON MOBILE DEVICES

Author: Rafael Santos Ortis

Supervisor: Hervaldo Sampaio Carvalho

Programa de Pós-graduação em Engenharia Elétrica

Brasília, December 2009

The objective of this work is to develop a system for visualization of biomedical signals in a generic way. The proposed system uses the SOA architecture to access data and biomedical signals of patients. The display of signals proposed in this paper proves to be effective for devices with a small screen resolution. The implementation of Web services based on communication also ensures a greater interoperability of the system developed with several other systems. The compression of data transmitted to web service shows that it is possible to have considerable savings (30%) in the transmission of data. Also the development of this work created the basis for a generic system for data visualization on mobile devices.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

ECG	- EletroCardioGrama
HTTP	- *Hypertext Transfer Protocol*
J2ME	- *Java 2 Plataform, Micro Edition*
JAX-WS	- *Java API for XML Web Services*
PDA	- *Personal Digital Assistant*
SOAP	- *Simple Object Access Protocol*
JSR	- *Java Specification Request*
API	- Interface para Programação de Aplicações
PHMS	- *Personal Health Monitor System*
GIMPA	- Gerenciamento de Informações Médicas do Paciente
TCP	- *Transmission Control Protocol*
UDP	- *User Datagram Protocol*
AWT	- *Abtract Window Toolkit*
Swing	- API gráfica do Java
Applet	- Aplicativo que é executado dentro de outro aplicativo.
WiFi	- Redes em fio padrão IEEE 802.11
J2SE	- *Java 2 Plataform, Standard Edition*
MIDP	- *Mobile Information Device Profile*
IMP	- *Information Module Profile*
XML	- *eXtensible Markup Language*
WSDL	- *Web Services Definition Language*
URI	- *Uniform Resource Indentifier*
WSS	- *Web Services Security*
MPS	- *Monitoramento Pessoal de Saúde*

1. INTRODUÇÃO

1.1. *CONSIDERAÇÕES INICIAIS*

A visualização de sinais está presente de várias formas no dia-a-dia das pessoas como no saldo da conta do banco, resultado de um jogo de futebol, variação da bolsa de valores durante o dia, e vários outros. Estas informações podem ser referentes a uma única forma de dado como o saldo do banco e outras vezes pode se referir a um conjunto de dados grande a ser visualizado como um exame de eletrocardiograma. A visualização de dados nos permite transformar os dados do seu estado bruto em gráficos e imagens que possibilitam tomada de decisão fácil e rápida.

Os principais dispositivos móveis atualmente são os celulares, assistentes pessoais (*handhelds*) – palmtops ou *PDA*s (Personal Digital Assistants) – e *smartphones* (dispositivos convergentes que agregam as funcionalidades de celular e assistentes pessoais). Os celulares estão atualmente presentes na maior parte da população, no Brasil temos uma densidade de 47,47[1] celulares a cada 100 habitantes e no Distrito Federal temos uma densidade de mais de um celular por habitante [1]. Já os assistentes pessoais até hoje não se tornaram um dispositivo popular, não só no Brasil como também em nações desenvolvidas, estando atualmente com as vendas em forte declínio [2].

Na maioria dos celulares atuais está presente uma máquina virtual Java (JVM) para sistemas móveis – Java J2ME. Muitas vezes, principalmente nos celulares, a única forma de se rodar um programa no dispositivo móvel é por meio de um programa escrito para esta máquina Java. Alguns celulares possuem sistemas operacionais mais sofisticados e permitem rodar programas compilados em C ou C++, como é o caso dos celulares com o sistema operacional SimbianOS®, Linux® embarcado e o WindowsMobile® (neste caso até C#), mas estes modelos (SmartPhones) não são maioria no mercado de celulares (cerca de 10% do mercado) [2].

Este trabalho tem como objetivo principal desenvolver um sistema genérico para a visualização de dados, em especial dados biomédicos, para dispositivos móveis com baixo poder computacional e com limitação de energia.

1.2. VISÃO GERAL DESTE DOCUMENTO

Este documento está dividido da seguinte forma:

1. Introdução;
2. Método utilizado;
3. Resultados;
4. Discussão e conclusões;
5. Referências bibliográficas.

1.3. CONVERGÊNCIA DIGITAL

A evolução tecnológica dos dispositivos móveis, em especial os celulares, nos últimos anos mostra uma tendência de unificar vários dispositivos que antes eram separados em um único. No início os celulares eram apenas para fazer e receber chamadas. Sistemas de agenda eletrônica, câmeras fotográficas, tocadores de áudio e posicionamento global por satélite (GPS) eram produtos separados. Com sua evolução foi possibilitado executar programas em Java, foram acrescentados sistemas de agenda eletrônica, câmera digital, tocadores de música e vídeo digitais, GPS e acesso a redes de dados sem fio de alta capacidade como as redes da terceira geração e as redes sem fio.

Um bom exemplo desta evolução são os produtos da empresa norte americana Apple® que desenvolveu um dos primeiros PDAs, o Apple Newton®, e depois criou o tocador de mídia IPod®,que se tornou seu principal produto, e após várias gerações deste tocador evoluiu para seu telefone celular que une as características de todos estes. Esta união de vários dispositivos em um único é a convergência digital.

Dentro desta nova realidade de convergência digital, o desenvolvimento de um sistema para a visualização genérica de sinais em dispositivos móveis não deve estar alheia às tecnologias voltadas para o mundo da telefonia celular. Estas tecnologias permitem criar um sistema genérico de visualização de sinais que possibilite a sua execução em diversos dispositivos de fabricantes diferentes. Embora seja necessária a alteração em casos específicos onde os padrões não foram seguidos pelos fabricantes, em geral estas tecnologias garantem uma grande compatibilidade do código gerado. A seguir será feita uma revisão destas tecnologias móveis.

1.4. TECNOLOGIAS MÓVEIS

1.4.1. J2ME

O J2ME [10] é uma das três edições da plataforma Java 2. O J2ME não define uma nova linguagem de programação, ele adapta a tecnologia Java para *handhelds* e dispositivos embarcados. E para atender as limitações dos dispositivos móveis o J2ME substitui certos APIs do J2SE e adiciona novas interfaces.

O J2ME é uma plataforma em três camadas de software: Configuração (*configuration*), Perfil (*profile*) e pacotes opcionais. A configuração disponibiliza os serviços fundamentais para uma ampla categoria de dispositivos de forma a definir uma máquina virtual Java (JVM) para estes dispositivos. O perfil define os serviços mais específicos em comum para um conjunto mais específico de dispositivos. Os pacotes opcionais, como o próprio nome diz, são serviços especializados que são úteis para vários tipos de dispositivos, mas não são necessários para todos.

A figura 1 mostra as divisões em um nível mais alto do J2ME, o qual foi dividido entre o CDC – (*Connected Device Configuration*) e o CLDC (*Connected Limited Device Configuration*). O CDC é voltado para computadores de mão como os mais potentes palmtops enquanto o CLDC é feito para os dispositivos com mais limitações de recursos como os telefones celulares.

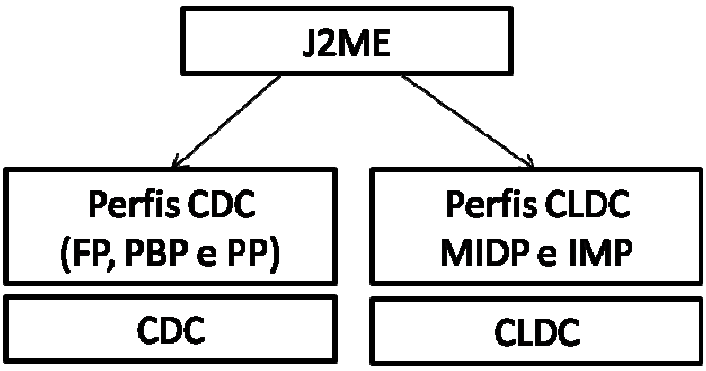

Figura 1 – Tecnologias Java para dispositivos com recursos limitados [10].

A figura 2 mostra a organização geral da plataforma J2ME. Nos níveis mais baixos temos o hardware que é controlado pelo sistema operacional. A máquina virtual usa os recursos do sistema operacional e provê a base para os perfis e os pacotes opcionais. A aplicação pode usar os recursos da JVM, das bibliotecas opcionais e as específicas do dispositivo.

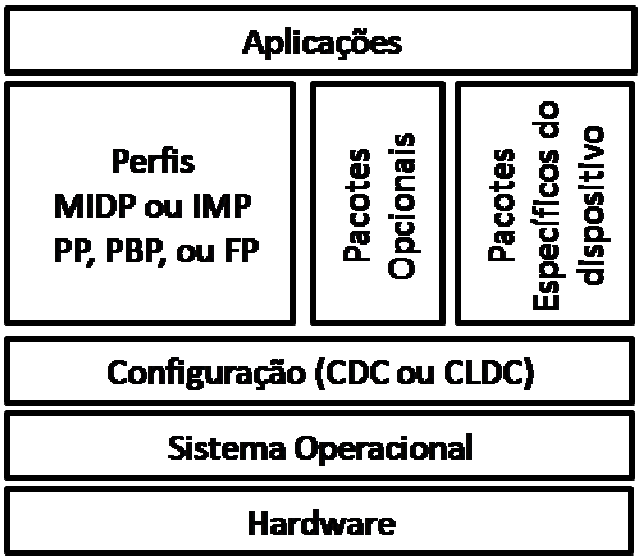

Figura 2 – Organização das camadas no J2ME [10]

1.4.1.1. CONFIGURAÇÕES DO J2ME

A configuração define o ambiente de execução básico do J2ME, incluindo a máquina virtual e o conjunto das classes básicas derivadas do J2SE.

1.4.1.1.1. CLDC

O CLDC é uma configuração mínima do J2ME. Foi criado para dispositivos com baixo poder computacional, tempo de vida da bateria, memória e banda de transmissão. No núcleo do CLDC está a máquina virtual Java usando um subconjunto dos pacotes do núcleo da linguagem Java em conjunto com as classes adaptadas para dispositivos limitados. Na versão 1.1 do CLDC foi adicionado o suporte ao processamento usando ponto flutuante.

1.4.1.1.2. CDC

O CDC foi desenvolvido para dispositivos mais potentes que os suportados pelo CLDC, como os celulares e PDAs mais avançados, os dispositivos embarcados mais sofisticados como os sistemas de navegação dos carros. O CDC engloba as classes do CLDC e as classes principais do J2SE. A figura 3 mostra a relação entre o CDC e o CLDC.

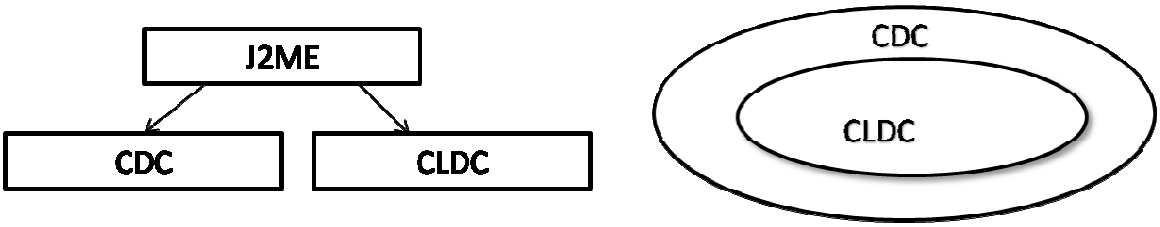

Figura 3 – Relação entre o CDC e o CLDC

1.4.1.2. PROFILES

As configurações (CDC e CLDC) não provêem as classes para gerenciar o ciclo de vida da aplicação, gerenciar a interface com o usuário, para manter e atualizar os dados no dispositivo ou para acessar informações seguras em um servidor na rede. Esse tipo de funcionalidade é provida pelos perfis (*profiles*) ou por pacotes opcionais.

Existem dois *profiles* baseados no CLDC: o MIDP (*Mobile Information Device Profile*) e o IMP (*Information Module Profile*). Existem também três *profiles* baseados no CDC: o Foundation Profile (FP), o Personal Basis Profile (PBP) e o Personal Profile (PP).

O MIDP é o mais adotado nos telefones celulares e PDAs. O MIDP 2.0 (JSR 118) melhora as capacidades do perfil adicionando suporte ao TCP, UDP e comunicação serial, conexões seguras, APIs para políticas de segurança, APIs para multimídia.

1.4.2. Pacotes Opcionais

Os pacotes opcionais são extensões da arquitetura J2ME que adicionam funcionalidades importantes para alguns dispositivos e aplicações mas não para todos, como envio de mensagens (SMS e MMS) e multimídia. A inclusão destes pacotes depende de cada fabricante e geralmente varia de produto para produto. Os pacotes principais são:

- Java APIs for Bluetooth (JABWT) - implementa a pilha de protocolos Bluetooth.
- Wireless Messaging API (WMA) – API para enviar e receber mensagens SMS (*Short Messaging Service*) e o MMS (*Multimedia Messaging Service*).
- Mobile Media API (MMAPI) – API genérica para o processamento multimídia.
- Web Services API for J2ME (WSA) – API para acessar e processar os dados dentro de um documento XML (XML parsing).

1.5. INTERFACE GRÁFICA EM J2ME

No J2ME a interface gráfica padrão é bastante restrita para um uso aceitável atualmente (especialmente se comparado com os aplicativos de celulares mais recentes como o iPhone® da Apple®). Na figura 4 temos um exemplo de um aplicativo utilizando a interface do J2ME em comparação com um aplicativo do iPhone.

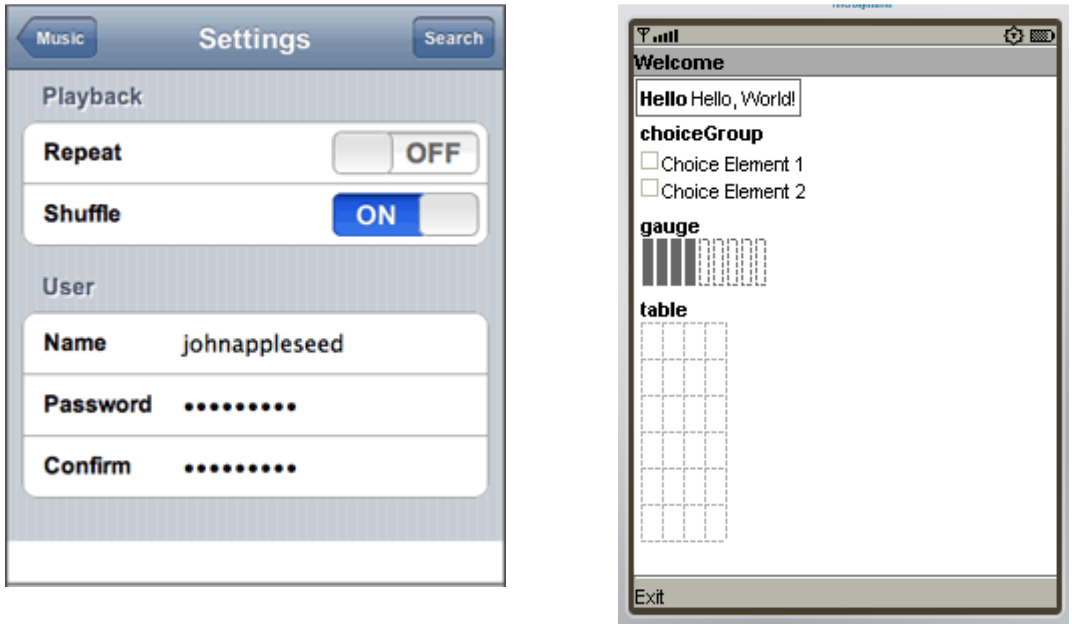

Figura 4 – Comparação entre aplicativos iPhone [42] (imagem da esquerda) e J2ME (imagem da direita)

Para contornar as limitações do J2ME várias bibliotecas foram desenvolvidas de forma a adicionar funcionalidades de modo que as aplicações móveis fiquem mais parecidas com as aplicações do desktop. Podemos destacar as seguintes bibliotecas: Apime[11], Byblos[12], Fire [13], LwVCL[14], J4ME[15], jMobileCore[16], kUI[17], MWT[18], Nextel Open Source Tollkits[19], Thinlet[20] , Synclast[21] e J2ME Polish[37].

Dentre estas bibliotecas se estaca a Thinlet que separa claramente a camada de visualização que é descrita como um arquivo XML e que foi parcialmente portada para o MIDP 2.0 pela Porsche Engineering [20].

1.5.1. Thinlet

Esta biblioteca permite criar as interfaces gráficas a partir de sua descrição em um arquivo XML de modo que é possível ter os seguintes componentes:

1. Label
2. Button
3. CheckBox
4. ToggleButton
5. ComboBox
6. TextField
7. PasswordField
8. TextArea
9. TabbedPane
10. Panel
11. Dialog
12. SpinBox
13. ProgressBar
14. Slider
15. SplitPane
16. List
17. Table
18. Tree
19. Separator
20. MenuBar

A versão mais atual da biblioteca Thinlet, atualmente, está disponível somente para o J2SE 1.1 a 1.4 e para o personal profile do J2ME.

1.6. ORIENTAÇÃO A SERVIÇOS (SOA) USANDO SERVIÇOS WEB

No desenvolvimento anterior [23] a troca de dados entre o dispositivo móvel e a central era realizada escrevendo diretamente no *socket* de transmissão de dados. Neste trabalho será abordada outra forma de trocar as informações com um servidor que é utilizando a orientação a serviços.

Na arquitetura orientada a serviços, os serviços Web (*Web Services*) são utilizados para disponibilizar funcionalidades de negócio, ou parte de uma aplicação, entre plataformas heterogêneas. A principal vantagem da utilização desta tecnologia é que o canal de comunicação está geralmente disponível (protocolo HTTP) entre os sistemas, ou seja, não é barrado por firewalls [3].

Na arquitetura e-PING, a qual define um conjunto de especificações que regulamentam a utilização da Tecnologia de Informação e Comunicação no Governo Federal [4], os

padrões de interoperabilidade de governo eletrônico recomendam que os serviços Web tenham duas características básicas [5]:

a. Que seja possível descobrir o serviço através de um catálogo de serviços.
b. Que os serviços forneçam uma descrição completa de como interagir com eles através do WSDL (Web Service Description Language).

O funcionamento do serviço Web pode ser resumindo como visto na figura 5:

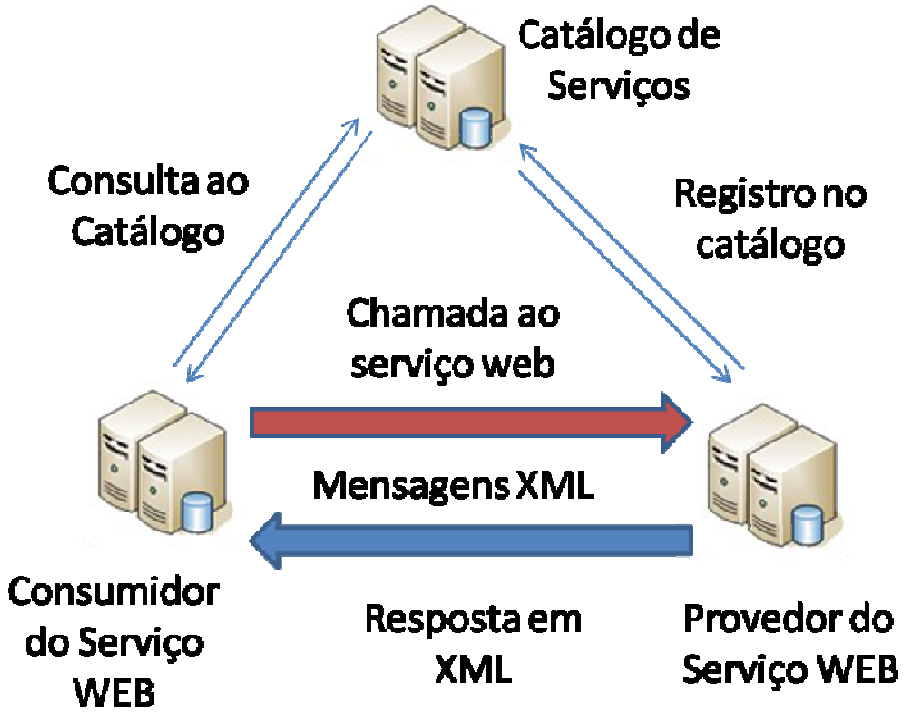

Figura 5 – Funcionamento de um WebService.

1.6.1. Serviços web RESTful e baseados em SOAP

Atualmente os serviços web podem ser separados em dois tipos: Serviços web RESTful e baseados em SOAP (*Simple Object Access Protocol*). Existe um terceiro tipo, que são os serviços baseados em JAX-RPC (chamada remota de procedimentos), mas este foi substituído pelo JAX_WS [6].

Um serviço RESTful é uma coleção de recursos Web identificados por uma URI (identificador único de recursos). Todos os documentos e processos são modelados como um recurso web, por exemplo: HTTP://localhost:8080/Servico/usuario/1/. As manipulações destes recursos são feitas pelo cabeçalho HTTP (GET, PUT, POST e DELETE) [7].

Já os serviços que usam SOAP expõem para a Web as funcionalidades definidas no código fonte (podem ser feitos em diversas linguagens) usando a linguagem WSDL. As mensagens são trocadas utilizando este protocolo e os clientes que desejam acessar os serviços o fazem baseados na sua descrição (WSDL) [7].

1.6.2. Segurança em Web Services

Existem duas formas usuais de garantir a segurança em um serviço Web: utilizando criptografia na camada de transporte por meio do protocolo HTTPS e dos padrões definidos no protocolo *Web Service Security* (WSS) [8]. Esta segurança visa reduzir os riscos e garantir a integridade e confidencialidade da informação transmitida. Usando o HTTPS estamos delegando a segurança para o protocolo TLS ou SSL [9] o que simplifica o desenvolvimento devido ao seu uso disseminado.

Com o WSS a segurança pode ser feita por meio do nome e senha do usuário, via kerberos, certificados X.509 e serviço de token de segurança (STS – Security Token Service). O WSS possibilita o uso seguro de serviços web em sistemas cujo acesso fim-a-fim possui intermediários [22], pois os esquemas de segurança estão na mensagem XML e não no protocolo de comunicação.

A escolha do esquema de segurança a ser utilizado deve se basear na escolha da segurança: (a) se estará na camada de transporte ou (b) na mensagem. No caso da mensagem é possível configurar:

1. A autenticação do cliente: direta ou por intermediários usando: certificado X.509, protocolo Kerberos 5 ou serviço de token de segurança (STS).
2. E os requisitos de segurança da mensagem como: confidencialidade de seu

conteúdo, detecção de mensagens adulteradas, e detecção de tentativas de reenvio de mensagens.

1.7. OBJETIVOS

Este trabalho tem como objetivo principal o desenvolver um sistema genérico para a visualização de dados, em especial dados biomédicos, para dispositivos móveis com baixo poder computacional e com limitação de energia. Como o sistema deve levar em consideração a convergência digital, sua utilização está voltada para telefones celulares e computadores de mão. A utilização principal do sistema será no Sistema Pessoal Móvel de Monitoração da Saúde - SPMMS [25] onde o sistema desenvolvido neste trabalho visa possibilitar a visualização local dos sinais capturados pelo dispositivo móvel (dados biomédicos).

Os objetivos secundários são:

- Utilizar a arquitetura orientada a serviço (SOA – *Service Oriented Arquitecture*) para acessar os dados e sinais biomédicos.
- Compressão dos dados enviados para diminuir o *overhead* do protocolo SOAP na transmissão das requisições.
- O sistema deve possuir interface com o usuário mais versátil do que a interface padrão do J2ME.

2. MÉTODOS

Este trabalho faz parte da pesquisa intitulada projeto GIMPA, desenvolvida por Carvalho [38][39], que consiste no monitoramento remoto de pacientes. As figuras 6 e 7 mostram o diagrama geral do projeto GIMPA. Como podemos ver na figura 6, o centro do projeto é a informação, que representa um *middleware* que unifica os módulos de Recuperação de Informação baseada em Evidências, o Prontuário eletrônico do Paciente e a Monitoração continua de Variáveis Fisiológicas.

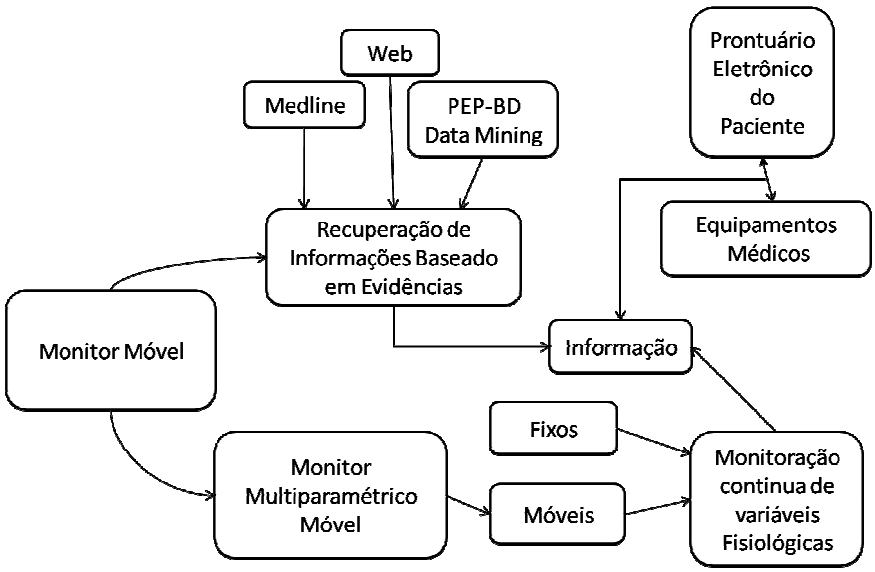

Figura 6 – Diagrama geral do projeto GIMPA [38][39].

A figura 7 mostra a estrutura geral do Monitoramento Pessoal de Saúde (MPS) baseada no corpo [40] o qual está dividido em três segmentos: o monitoramento de sinais fisiológicos, o monitoramento de sintomas e a captura de informações baseadas em evidências.

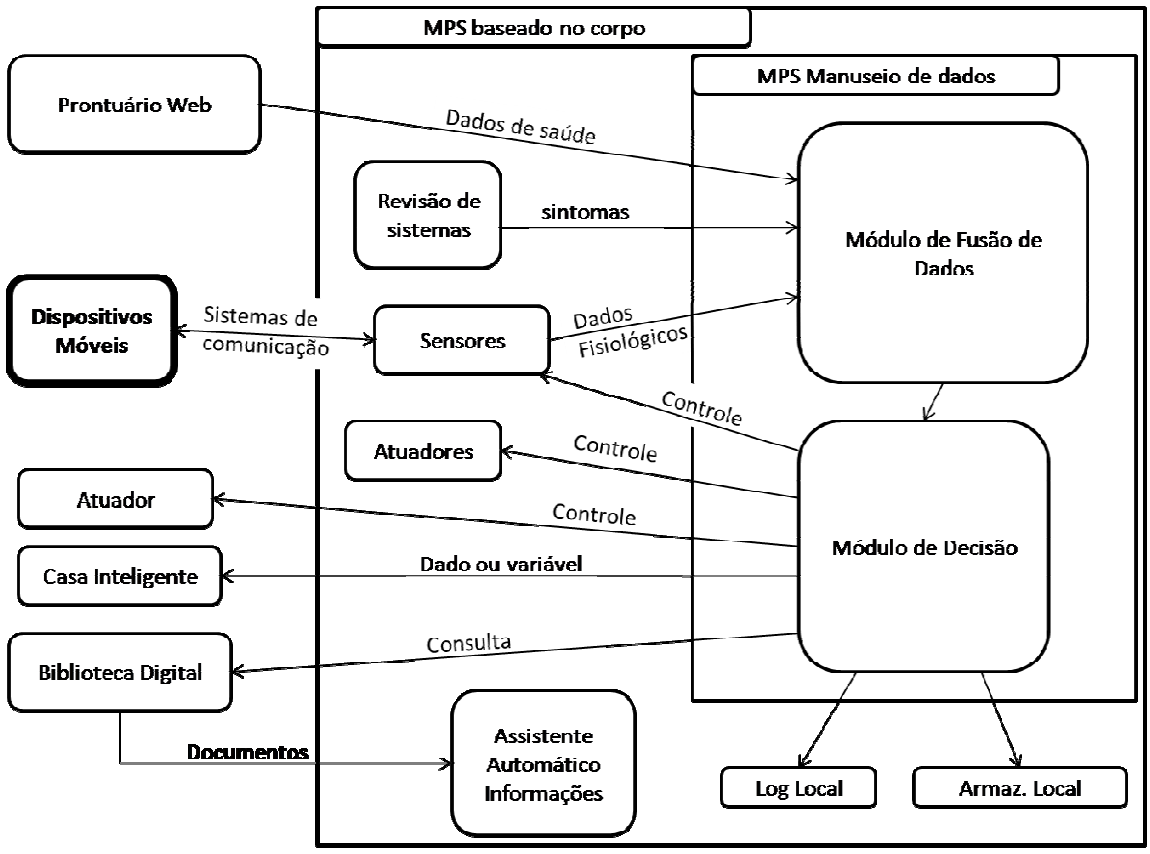

Figura 7 – Estrutura Geral do Projeto de Monitoramento Pessoal de Saúde baseada no corpo [40].

Anteriormente [23] foi desenvolvido um sistema para receber e enviar sinais biomédicos utilizando um PDA, rodando o sistema operacional Linux utilizando a biblioteca Qtopia. Este sistema também é capaz de mostrar na tela os sinais de eletrocardiograma, eletromiograma, pressão arterial, freqüência cardíaca, temperatura e oximetria de pulso.

A limitação deste sistema se mostrou no fato de existirem poucos dispositivos móveis comerciais que rodem Linux e suportem a biblioteca Qtopia. Este fato é relevante quando se tem em mente que estes dispositivos poderiam ser utilizados em órgão públicos, onde um sistema mais genérico se mostra mais promissor.

O sistema desenvolvido visa possibilitar o acesso a dados biomédicos em dispositivos móveis tendo em mente as restrições do processador destes dispositivos que geralmente são processadores de ponto fixo. O objetivo é criar uma aplicação capaz de acessar as informações e exames de um Prontuário Eletrônico de modo seguro utilizando os principais padrões de comunicação existentes para dispositivos móveis. A arquitetura do sistema pode ser vista na figura 8 e 9:

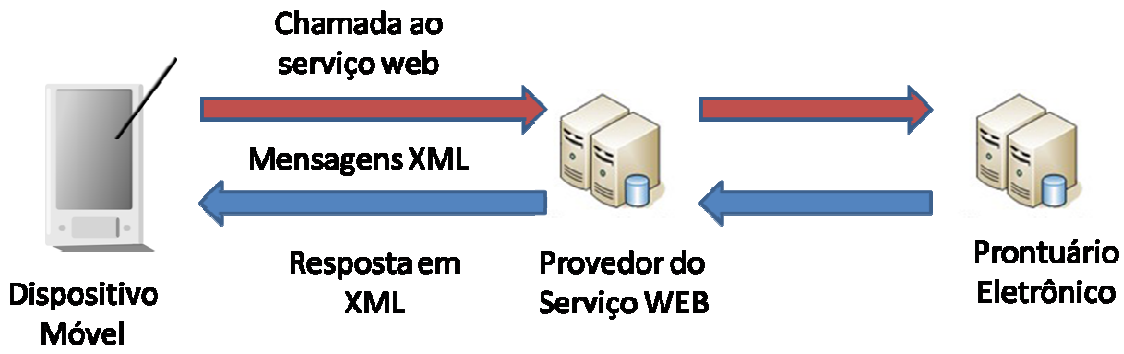

Figura 8 – Arquitetura básica do sistema.

Nesta arquitetura, as informações do prontuário eletrônico devem estar disponíveis em um ou mais serviços web e o dispositivo móvel acessa estas informações diretamente (figura 8) ou através de um gateway que adiciona compressão aos dados trocados entre o serviço e o dispositivo. Este mesmo gateway pode configurar qual o nível de segurança utilizado de modo a encontrar uma relação segurança/uso de processamento no dispositivo (figura 9).

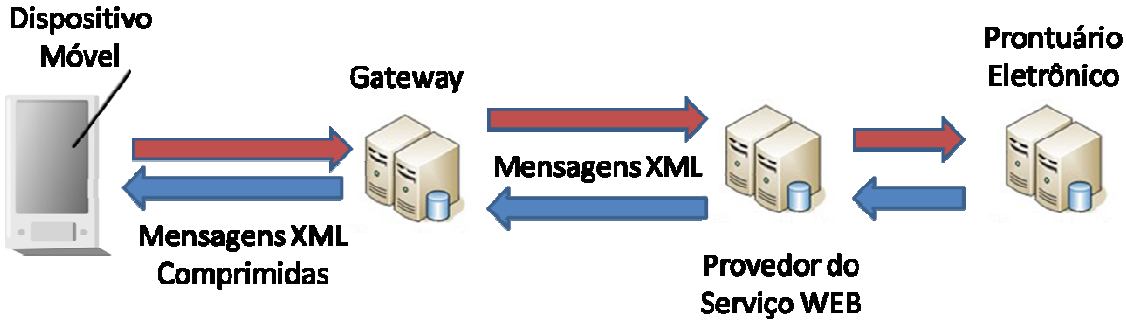

Figura 9 – Arquitetura básica do sistema com o gateway.

A arquitetura do sistema pode ser também vista como a implementação dos seguintes casos de uso (figura 10):

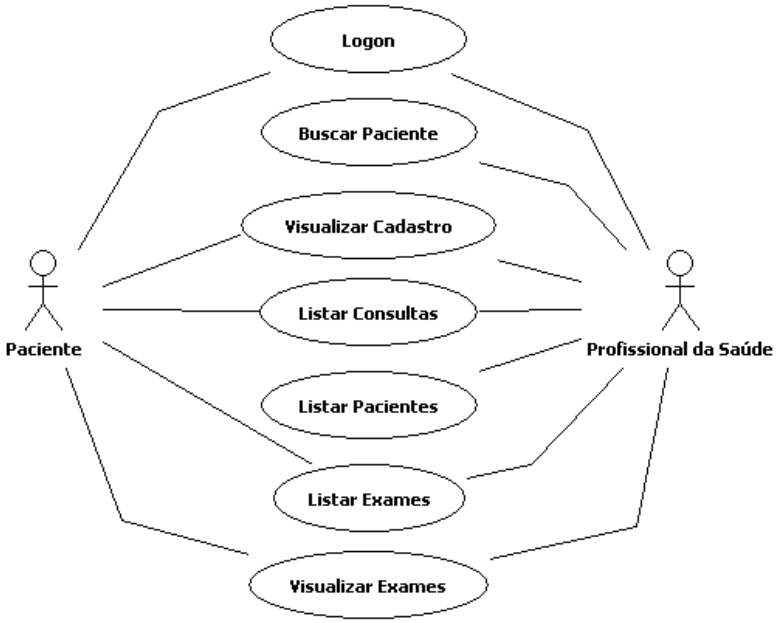

Figura 10 – Casos de uso do sistema.

A realização destes casos de uso está distribuída entre o visualizador e o serviço web. No serviço web é realizado o acesso ao banco, conversão de dados e controle de sessão. No visualizador estão as funcionalidades da interface com o usuário e a visualização dos sinais.

A visualização dos exames será realizada por meio de um visualizador genérico que tem como objetivo mostrar:

- Gráficos de linhas – Para ECG e outros sinais com amostragem alta (figura 11);

Figura 11 – Gráfico de Linhas – Mostrando um ECG.

- Gráficos de barras – Histograma como o exemplo da figura 12;

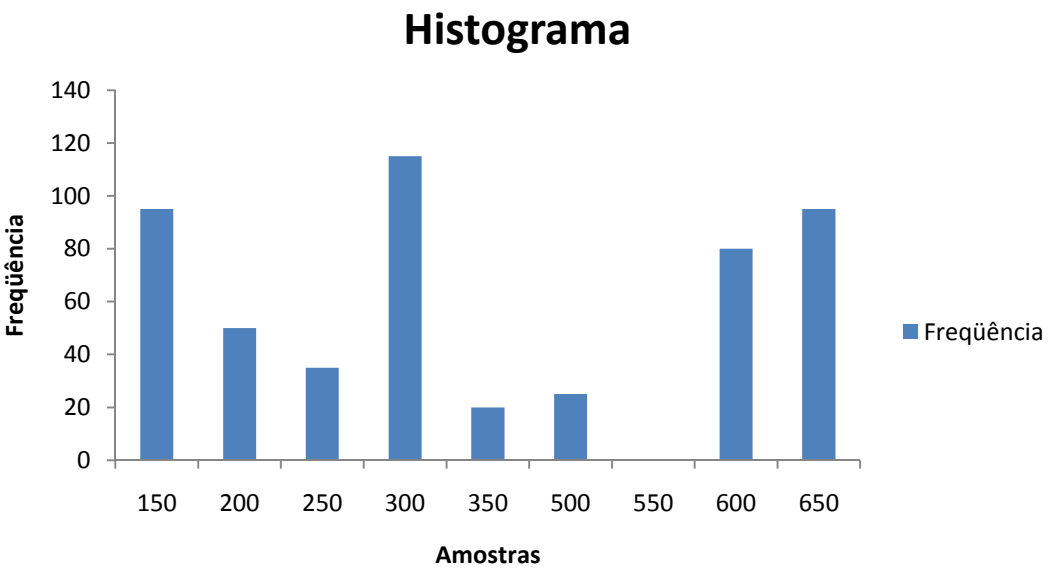

Figura 12 – Gráfico de Barras - Histograma.

- Gráfico de barras com delimitação de valores máximo e mínimo, o qual pode ser utilizado para mostrar a pressão arterial como visto na figura 13;

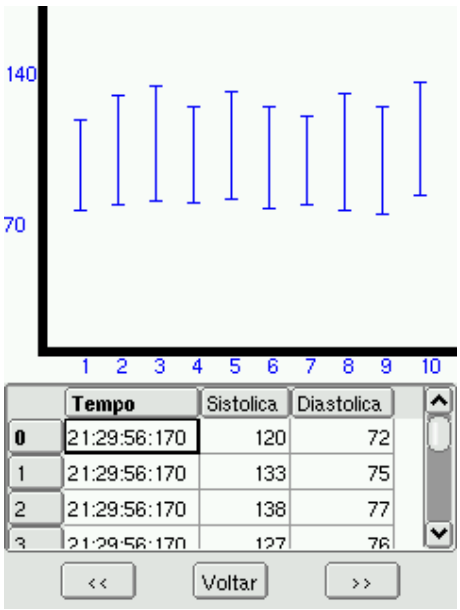

Figura 13 – Gráfico de Barras.

- Gráficos do tipo Candlestick (figura 14). Embora este tipo de visualização não seja usual na medicina, em outras áreas é utilizada. Por exemplo, ele é utilizado na analise do comportamento da bolsa de valores [25].

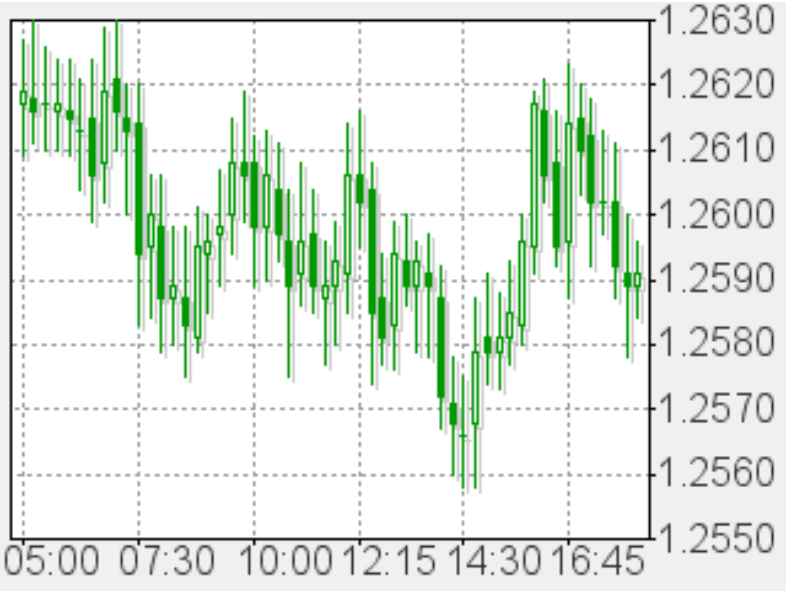

Figura 14 – Gráfico do tipo Candle Stick. Fonte [41]

- Imagens de exames guardadas no prontuário eletrônico do paciente. Embora as dimensões reduzidas dificultem a visualização destes exames. As imagens podem ser utilizadas em outros tipos de equipamento que permitem a visualização como os projetores de imagens embarcados ou portáteis [26].

Outra necessidade do visualizador é a necessidade de interatividade com os dados disponibilizados. Nos gráficos de linhas e imagens deve ser possível a possibilidade de aproximar ou afastar (zoom in/out) e a possibilidade de deslocar a posição de visualização dos dados (*scroll*) de modo a mostrar apenas uma parte dos dados.

Alem do visualizador de sinais, o sistema no dispositivo móvel possui a interface de *login*, a interface de navegação do sistema, e as interfaces para a apresentação dos dados dos pacientes no Prontuário do paciente, que são disponibilizados pelos serviços web.

2.1. FERRAMENTAS UTILIZADAS NO DESENVOLVIMENTO

O sistema foi desenvolvido utilizando ferramentas de código aberto de forma a possibilitar um maior controle sobre os componentes utilizados. A linguagem de programação utilizada foi o Java no serviço web e J2ME no visualizador. Vários componentes foram modificados para atender as necessidades que antes não eram contempladas, por exemplo: compressão dos dados transferidos pelo *WebService*; criptografia em um *socket* do WebService; melhoramentos na camada de interface do usuário usando o Thinlet para o MIDP 2.0.

O desenvolvimento foi separado em três programas: o WebService, o gateway de compressão para o WebService e o visualizador em J2ME.

No desenvolvimento dos três programas foram utilizadas as seguintes ferramentas:

1. Netbeans – IDE de programação;
2. JDK 1.6 – ferramentas de desenvolvimento java;
3. Wireless Tools 2.5 – emulador J2ME de dispositivos móveis (figura 15)
4. Apache Tomcat – Servidor de aplicação

Figura 15 – Captura da tela do Wireless Tools 2.5 em execução.

No visualizador, as seguintes bibliotecas foram utilizadas:

1. kSOAP2 – acesso aos serviços web;
2. kXML2 – biblioteca de manipulação de XML;

3. Thinlet – interface gráfica definida por arquivos XML;
4. BouncyCastle – biblioteca de criptografia;
5. Jzlib – biblioteca de compressão.

Nos próximos itens 2.2 a 2.5 estão relacionadas resumidamente as funcionalidades de cada biblioteca e as mudanças nelas efetuadas. No item 2.6 estão os detalhes do desenvolvimento da comunicação entre o dispositivo móvel e os serviços web. No item 2.7 os detalhes do desenvolvimento do gateway de compressão. E finalmente no item 2.8 os detalhes do visualizador genérico desenvolvido para os dispositivos móveis.

2.1.1. kSOAP2

A biblioteca kSOAP2 foi desenvolvida para proporcionar a utilização de SOAP em dispositivos limitados [27]. A utilização desta biblioteca nos possibilita um controle fino sobre o chamado ao WebService, pois o seu código fonte é aberto. O código fonte desta biblioteca foi incluído no projeto do visualizador para facilitar as modificações. Outra vantagem de utilizar esta biblioteca é que a máquina Java do dispositivo não precisa da implementação do pacote opcional de serviços web - JSR-172. Isto possibilita que uma quantidade maior de dispositivos seja compatível com a aplicação.

Esta biblioteca foi modificada para permitir comprimir a mensagem SOAP e descomprimir a resposta antes de realizar a leitura da resposta. Outra modificação foi a inclusão da criptografia SSL onde é possível utilizar a biblioteca nativa do J2ME ou a biblioteca BouncyCastle [28], que é mais genérica e que possibilita definir o tipo de segurança utilizada na conexão.

2.1.2. Thinlet

Para criar a interface com o usuário primeiramente foi levantada a hipótese de utilizar a interface padrão do MIDP 2.0, mas mesmo esta ainda é muito simples para as necessidades do programa. A segunda opção levantada foi desenhar a interface gráfica janela a janela utilizando funções básicas como: desenhar quadrado, copiar figura, desenhar linhas, etc. Mas a complexidade desta tarefa não a torna viável neste trabalho.

Como um dos objetivos é a visualização genérica, torna-se mais interessante que a

interface seja genérica também. Para criar esta generalização foi levantada a hipótese de utilizar um browser XHMTL embutido no software para criar a interface com o usuário. Deste modo para criar a interface bastaria criar um arquivo XHTML (página web). Esta possibilidade mostrou-se inviável pois não existia nenhuma implementação completa do XHTML em J2ME com o código fonte disponível. Uma segunda opção seria programar um sub-set do XHTML para descrever a interface, mas a complexidade desta tarefa é muito alta o que demandaria muito tempo.

Portanto, embora não seja um padrão de desenvolvimento, foi utilizada a biblioteca gráfica Thinlet que se aproxima bastante ao objetivo desejado. Nesta biblioteca a interface é definida utilizando arquivos XML. A interação com o usuário é gerenciada pela biblioteca, que chama a lógica de negócio separando assim a camada de apresentação (interface gráfica) dos métodos da aplicação.

Esta biblioteca possibilita a criação de interfaces gráficas genéricas a partir de um arquivo XML de modo a criar um sistema MVC (*Model View Controller*). O modelo é definido por um arquivo XML e o controlador é criado herdando a classe Thinlet, a visão é criada utilizando as funções básicas para desenhar na tela (texto, imagens, linhas, retângulos e arcos). A implementação original utiliza a biblioteca AWT do J2SE para criar a interface. Como a AWT não está disponível no J2ME a biblioteca foi convertida para o MIDP 2.0 pela Porsche Engineering [20] e o código fonte foi disponibilizado no site do Thinlet.

Esta versão MIDP 2.0 ainda possuía alguns problemas, pois referenciava a uma classe que não existe no CLDC padrão (java.lang.Method). Esta classe era utilizada para chamar o método definido para uma ação do usuário na interface gráfica. Para resolver este problema, simplificou-se o sistema e adicionou-se uma função abstrata click() que deve ser implementada pela classe que herda do Thinlet para realizar o controle da aplicação. Diversos outros erros foram retirados da biblioteca especialmente em relação à visualização de imagens e nas interfaces da tabela. A interface gráfica utilizada pode ter os seguintes componentes:

1. Tela de Abertura (*Splash Screen*)

2. *Labels*
3. Caixas de Texto
4. Botões
5. Tabelas
6. Listas
7. Barra de progresso
8. Imagens

Para criar as telas foi utilizado o programa ThingG, que possibilita criar a interface gráfica visualmente e salva a interface definida como um arquivo XML que é carregado pela biblioteca Thinlet. Por exemplo, o conteúdo do arquivo XML gerado para a tela de logon é:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel background="#99cc99" columns="1" gap="5"
 height="237" left="19" top="14" width="240">
    <label alignment="center" font="16 bold"
     height="22" text="MedPlot" width="64"/>
    <panel background="#ffffff" border="true" columns="1"
     foreground="#000000" gap="4" height="173" left="20"
top="12" width="200">
        <label font="13 bold" height="20" text="Login:"
width="40"/>
        <textfield background="#008b00" editable="false"
       end="5" height="19" name="login" start="5"
     width="147"/>
        <label font="13 bold" height="20" text="Senha:"
width="40"/>
        <passwordfield editable="false" end="5" height="19"
        name="senha" start="5" width="147"/>
        <panel columns="2" gap="11" height="53" top="15"
width="158">
            <button action="click()" background="#cbcbff"
font="13 bold"
          halign="center" height="22" icon="/login.png"
          name="conectar" text="Conectar" valign="center"
          width="90"/>
         <button action="click()" background="#fbffaf"
         font="13 bold"
          halign="center" height="22" icon="/exit.png"
         name="sair"
          text="Sair"    valign="center" width="56"/>
        </panel>
```

```
        <progressbar halign="center" name="progresso"
visible="false"/>
    </panel>
</panel>
```

A figura 16 abaixo mostra o ThingG em execução e o a figura 17 mostra a mesma tela sendo executada no emulador do J2ME.

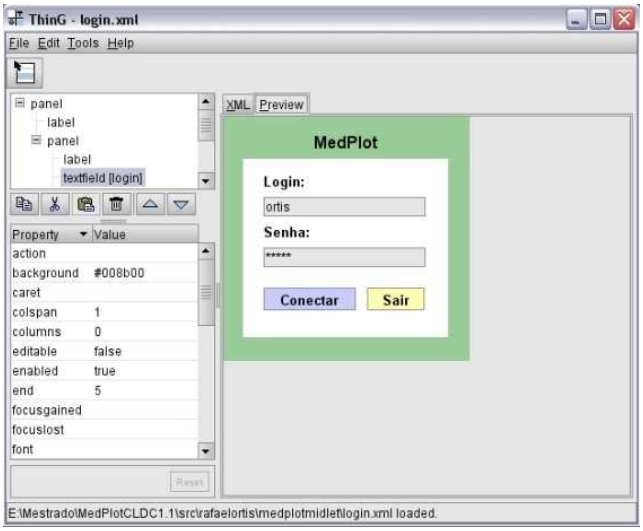

Figura 16 – ThinG em execução

Figura 17 – Tela de Logon no Emulador

2.1.3. BouncyCastle

Esta biblioteca possui um conjunto de APIs de criptografia e é utilizada na codificação e decodificação hexadecimal ou base64 dos parâmetros das informações trocadas pelo dispositivo móvel e o gateway de compressão.

O visualizador pode também utilizar a implementação do TLS desta biblioteca (microTLS) que possibilita assegurar, se necessário, um nível de criptografia mais alto que o padrão implementado na JVM. Uma vez que a implementação é puramente em Java o desempenho é consideravelmente inferior ao da implementação da JVM do dispositivo a qual é otimizada pelo fabricante.

Esta biblioteca é disponível sob a licença MIT que permite alterar, modificar e distribuir o código [29].

2.1.4. jZlib

Esta biblioteca é utilizada pelo sistema móvel para comprimir e descomprimir as mensagens do WebService de modo a utilizar um menor volume de dados na transmissão. Isto visa diminuir o custo de cada operação efetuada pelo sistema. A biblioteca é uma adaptação [30] da biblioteca zlib [31] do Unix para o Java a qual usa o método DEFLATE [34] [35] para realizar a compressão dos dados sem perdas. Esta biblioteca foi utilizada pois permite acessar os dados recebidos sem descompactar através da classe ZOuputStream. Como os dispositivos móveis possuem pouca memória (em comparação com um PC) manter os dados comprimidos na memória diminui a quantidade de memória necessária.

2.2. COMUNICAÇÃO

A comunicação dos dados é feita utilizando serviços web. O visualizador acessa os dados biomédicos disponibilizados em um serviço utilizando a biblioteca kSOAP2. É utilizada uma máquina de estados para controlar a comunicação entre o visualizador e o WebService. O funcionamento desta máquina de estados é bem simples:

1. A máquina de estados no estado Connect. No estado de conexão é aberta a sessão com o serviço web enviando o login e a senha do usuário;
2. Se a conexão for bem sucedida o visualizador recebe um identificador da sessão e o estado muda para Idle (estado de espera), se não for o estado volta para desligado (OFF);
3. No estado de espera o usuário pode receber dados do serviço web (estado GET) ou enviar os dados para o serviço (estado POST). Do estado Idle a conexão é finalizada passando para o estado Disconnect;
4. No estado Disconnect sessão do serviço web é finalizada e o estado vai para o estado desligado (OFF)

O diagrama a seguir mostra os estados da máquina de estados (figura 18):

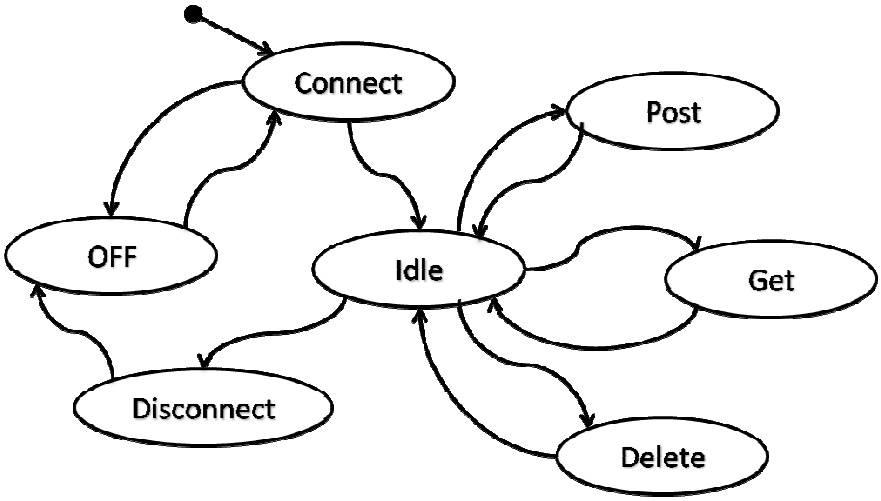

Figura 18 – Diagrama da máquina de estados.

Observe que esta máquina de estados possibilita realizar as funções básicas do CRUD (do inglês *Create, Read, Update and Delete*): Criar, Selecionar, Atualizar e Apagar. No estado Post temos as funcionalidades de criar e atualizar um registro e no Get temos a funcionalidade de selecionar um registro ou excluir um registro.

2.2.1. Serviços web

O objetivo dos serviços web são prover acesso a camada de dados e a camada de negócio do aplicativo (regras de negócio). Para manter o desenvolvimento simples as funcionalidades foram colocadas em um dois serviços web que fazem o controle de login, controle de sessão e acesso ao banco de dados PostgreSQL. O serviço faz o controle de acesso e busca os usuários em um banco de desenvolvimento do software GSWeb [33] utilizado no Hospital Universitário de Brasília.

Para criar o serviço foi utilizado a ferramenta de desenvolvimento Netbeans (versões 5.0 a 6.5) e o servidor de aplicação Apache Tomcat.

O diagrama de classes do WebService para acessar o banco de dados do GS-Web é o seguinte (figura 19):

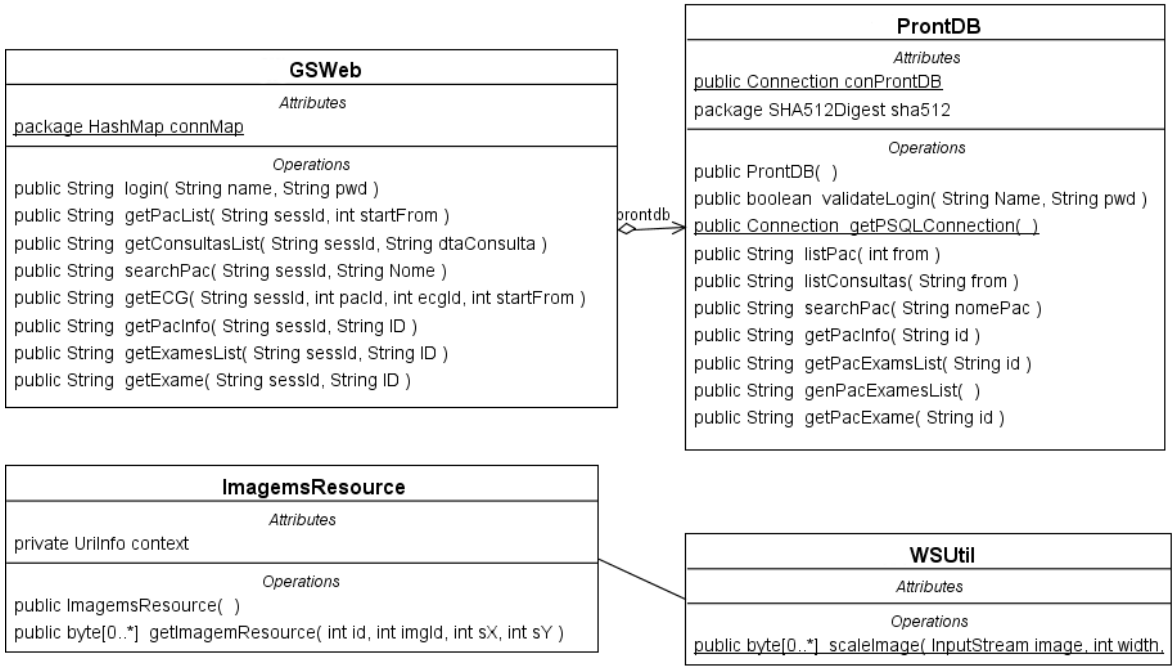

Figura 19 – Diagrama de classes.

O serviço expõe as seguintes funções da Classe GSWeb:

- **login**: faz o login do usuário recebendo o nome do usuário e a senha;
- **getPacList**: retorna a lista completa de pacientes começando de uma

posição;

- **searchPac**: busca um paciente pelo nome do paciente – pode retornar um paciente ou uma lista de pacientes;
- **getPacInfo**: retorna as informações do cadastro do paciente;
- **getExamesList**: lista todos os exames de um paciente;
- **getExame**: busca um exame específico de um paciente para a visualização de seus dados;
- **putExame**: envia os dados de um exame capturado de um dispositivo conectado ao PDA ou celular para ser adicionado ao banco de dados. Passa a identificação do paciente, o tipo e os dados capturados.
- **getConsultasList**: envia a lista de consultas marcadas do profissional logado no sistema.

Foi definido também um serviço *RESTful* para acessar as imagens dos exames guardadas que foi denominado *ImagemsResource*. Caso as imagens fossem transmitidas utilizando um serviço web padrão (SOAP) seria necessário converter os bytes da imagem para base64. Este serviço simplifica, para o sistema no dispositivo móvel, o acesso a imagens que deste modo não necessitam ser convertidas para base64 para serem transmitidas e diminui a quantidade de dados transmitidos.

As funcionalidades destes dois serviços possibilitam a troca da maior de parte das informações entre o dispositivo móvel e o Prontuário eletrônico. A seguir são apresentados os detalhes do gateway de compressão utilizado para diminuir o volume de dados transmitidos.

2.2.2. Gateway de compressão

A transmissão dos dados dos serviços web utiliza o protocolo SOAP e os dados transmitidos geralmente utilizam o XML. Este protocolo utiliza geralmente como camada de transporte o protocolo HTTP, utilizando as funções GET e POST. Em dispositivos móveis, a transmissão possui um custo de energia elevado e, se estiver usando a rede celular de telefonia, possui também um custo financeiro da transmissão bem mais caro que o da rede local. Para diminuir estes custos na transmissão existe a possibilidade de

comprimir as chamadas do WebService utilizando a compressão do protocolo HTTP [32], mas esta nem sempre pode ser utilizada através das redes de telefonia celular devido à configuração dos servidores de Proxy das prestadoras de telefonia.

Para deixar o sistema mais abrangente foi desenvolvido um gateway de compressão para o serviço web de modo a diminuir os custos de utilização, sem depender da configuração de rede utilizada. A figura abaixo (figura 20) mostra um diagrama do papel do gateway na transmissão.

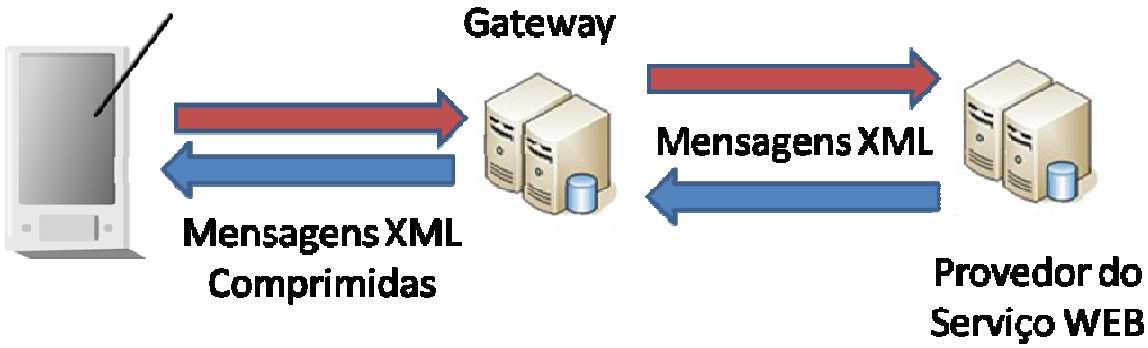

Figura 20 – Diagrama do gateway de compressão.

O funcionamento da compressão utiliza o seguinte protocolo de comunicação:

1. O dispositivo gera a requisição XML normalmente e a envia para o servidor formatando a chamada com os campos:
    1. Tamanho da URL do serviço (inteiro de 32 bits)
    2. String da URL do serviço (string unicode)
    3. Tamanho da função SOAP (inteiro 32 bits)
    4. Função SOAP (string unicode)
    5. Tamanho da requisição (inteiro 32 bits)
    6. Requisição comprimida (utilizando DEFLATE)
2. A partir das informações enviadas o gateway formata a mensagem SOAP em cima do HTTP e realiza a chamada no servidor e devolve a resposta com os seguintes campos:
    1. Tamanho da resposta (inteiro 32 bits)
    2. Resposta comprimida (DEFLATE)

2.3. VISUALIZADOR DE SINAIS

O visualizador é o componente do sistema no dispositivo móvel que desenha os sinais biomédicos na tela do dispositivo móvel. Este componente, que inicialmente foi desenvolvido usando o Qtopia [23], foi generalizado tendo como alvo a utilização em telefones celulares. Esta generalização deveu-se ao fato que a implementação inicial era em C++ e lidava com os dados de forma específica. Foi necessário desenvolver este componente pois ainda não existe nenhum visualizador com as características necessárias com o código fonte aberto.

Inicialmente foi levantada a hipótese de programar utilizando o CLDC 1.0 (sem suporte a ponto flutuante), mas este desenvolvimento mostrou-se impraticável devido a complexidade das operações necessárias para tornar o visualizador genérico.

O visualizador divide-se em dois componentes: um específico para o eletrocardiograma e um genérico. O específico é uma adaptação do visualizador desenvolvido no trabalho anterior [23] para o J2ME.

O visualização do ECG está separada pois leva em consideração as dimensões do visor do dispositivo móvel para mostrar o sinal do ECG o mais próximo ao de um sinal impresso em papel. Como não é possível saber o tamanho da tela do dispositivo programaticamente, este valor é programado dentro da aplicação.

O visualizador genérico possibilita mostrar:

- Gráficos de Linhas;
- Histogramas;
- Barras com delimitação dos valores máximo e mínimo.
- Imagens

E para cada um destes tipos possuem:

- Rolar a tela;
- Divisões das linhas de escala programáveis.

Na visualização de imagens a imagem não é transmitida na sua resolução nativa. O sistema envia a imagem em uma escala menor e cada vez que o usuário aumenta o zoom na imagem, o serviço envia uma imagem com a resolução maior.

O visualizador utiliza o J2ME e tem como plataforma mínima o MIDP 2.0 e o CLDC 1.1. A necessidade de ter o MIDP 2.0 se dá com a necessidade da visualização de dados. No MIDP 1.0 as funcionalidades gráficas são muito restritas (por exemplo, o programa não possui recursos para descobrir a resolução da tela do dispositivo), já na versão 2.0 o programa dispõe de diversas classes especificas para desenhar na tela.

3. DESENVOLVIMENTO E RESULTADOS

Neste capitulo serão apresentados os resultados da implementação de cada parte do sistema: O visualizado, denominado MedPlot, o gateway de compressão e os serviços web.

3.1. MEDPLOT

O objetivo deste programa é a consultas e/ou modificações no banco de dados de um prontuário eletrônico. As funcionalidades desenvolvidas no sistema são:

1. Acesso às informações básicas do paciente – Nome, endereço, telefone.
2. Acesso a exames do paciente como:
    1. ECG;
    2. Pressão arterial;
    3. Temperatura
    4. Raio-X
3. Busca de pacientes
4. Lista de consultas marcadas

Para programar o sistema foi escrito um WebService que acessa o banco de dados do Prontuário eletrônico e envia estes dados compactados para o dispositivo móvel.

3.2. VISUALIZADOR

Nesta sessão serão apresentadas as telas desenvolvidas para o sistema móvel. As telas foram capturadas no emulador do *Sun Java Wireless Toolkit* versão 2.5.2.

3.2.1. Tela de Abertura

A tela de abertura é uma tela estática mostrando uma imagem do tipo PNG e uma caixa de texto não editável no centro da tela simulando um splash screen, como visto na figura 21.

Figura 21 – Tela de abertura do programa.

3.2.2. Logon

Na tela de Logon (figura 22) o usuário insere o login e a senha para acessar o sistema. O serviço web é o responsável por autorizar o usuário a utilizar o sistema. A interface gráfica padrão do visualizador é composta por painéis dentro de painéis. Onde o painel principal contém o título da tela e o painel central. No painel central estão as funcionalidades da página.

Figura 22 – Tela de Login do MedPlot.

3.2.3. Inserção de texto

A inserção de texto dos campos utiliza o editor configurado para a JVM instalada no dispositivo móvel. Deste modo cada dispositivo pode mostrar uma interface diferente para a inserção do texto. A figura 23 mostra a edição dos campos do login e senha no emulador:

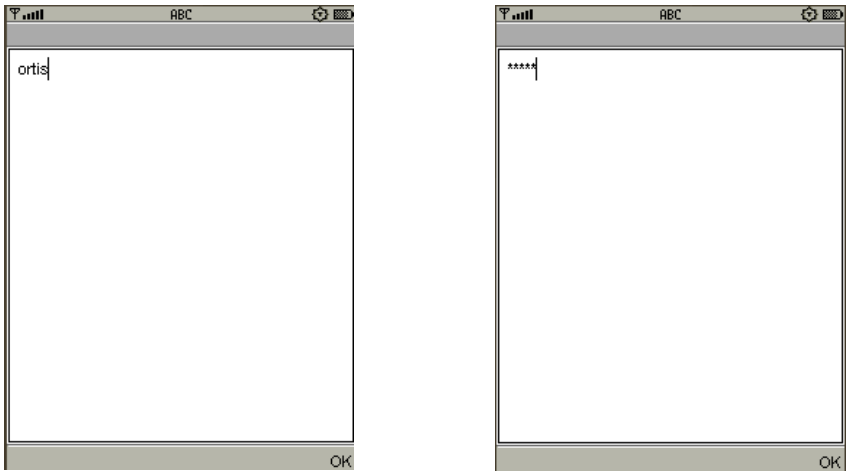

Figura 23 – Entrada de texto no emulador.

3.2.4. Menu do MedPlot

No menu estão localizados o início das funcionalidades do sistema: Listar os pacientes, Procurar um paciente e listar as consultas marcadas caso o usuário seja um profissional de saúde. Caso o usuário seja um paciente as funcionalidades mostradas são: Minhas informações e Consultas Agendadas. Para escolher a funcionalidade, a entrada do usuário pode ser por meio da tela sensível ao toque, navegar utilizando o botão “prox” para mudar o foco ou utilizar o botão de navegação e escolher a opção utilizando o botão central. Para marcar que um botão está selecionado a biblioteca Thinlet foi modificada para mudar a borda do botão para vermelho como visto na figura 24.

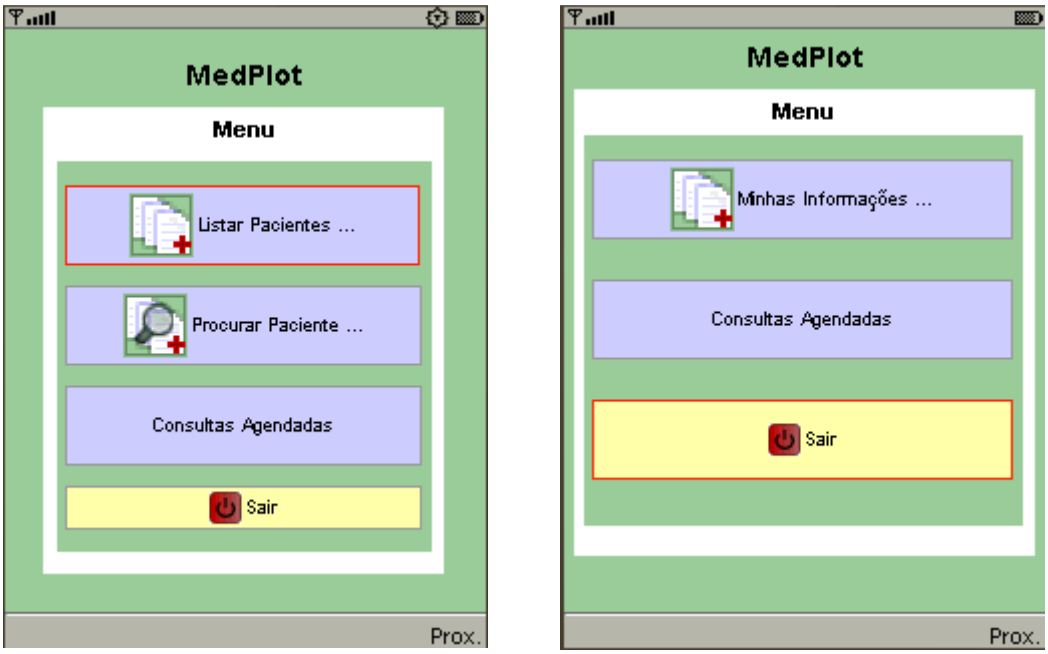

Figura 24 – Telas do menu do sistema. Na figura a esquerda a Tela do Profissional de saúde e na tela a direita a tela do paciente.

3.2.5. Busca de pacientes:

Ao escolher a opção Procurar Paciente ... (figura 25) o usuário deve digitar o nome do paciente para a busca, e acionar a busca no botão Procurar Paciente. O sistema irá buscar o paciente cujo nome comece com a entrada do usuário e irá retornar uma lista de pacientes.

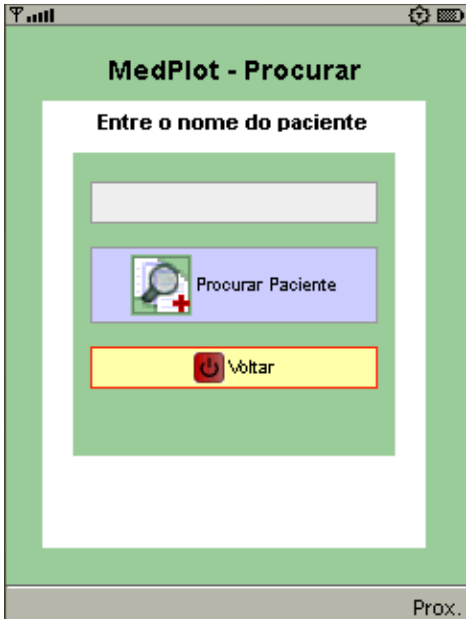

Figura 25 – Tela de busca por um paciente pelo nome.

3.2.6. Lista de pacientes:

O resultado da ação de procurar um paciente e listar os pacientes utiliza a mesma tela (figura 26). Nesta tela são listados os pacientes. Ao apertar o botão central em uma entrada da lista, o sistema irá buscar as informações do paciente. Outra forma de ver as informações é selecionando uma linha e utilizar o comando “Selecionar”. Se o usuário apertar o botão voltar, o sistema retorna para a tela anterior.

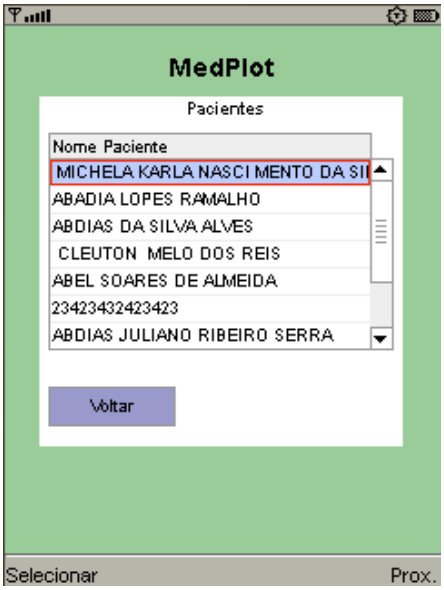

Figura 26 – Lista de Pacientes.

3.2.7. Informações dos pacientes

Nesta tela (figura 27) é mostrado o resultado da consulta dos dados básicos do paciente que estão armazenados no prontuário eletrônico. Para simplificar o sistema, são mostrados apenas o nome, nome da mãe, data de nascimento e telefone. Ao apertar o botão exames, o sistema busca a lista de exames do paciente.

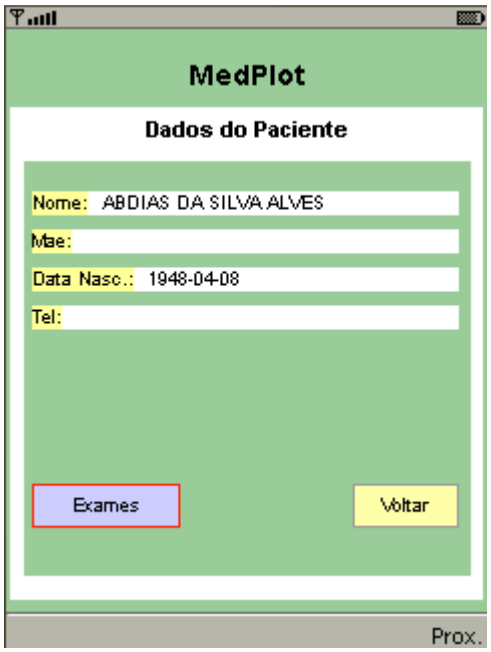

Figura 27 – Dados básicos do prontuário do paciente.

3.2.8. Lista de exames

Os exames do paciente selecionado são listados nesta tela, mostrando o tipo do exame e a data da realização dos exames (figura 28). Para visualizar o exame, basta pressionar o botão de ação ou selecionar o comando "*Prox*." e depois pressionar o botão *Ver Exame*.

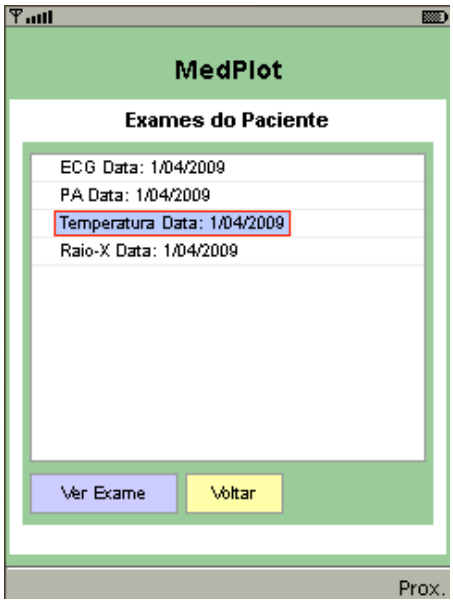

Figura 28 – Lista de Exames do Paciente.

3.2.9. Visualização de exames

Nesta sessão serão mostrados os resultados do desenvolvimento do visualizador genérico de sinais biomédicos. O sistema demonstra a visualização de quatro tipos de exames: ECG, pressão arterial, temperatura e uma imagem de um raios-x.

O funcionamento do sistema segue os seguintes passos: o visualizador faz uma requisição para o WebService enviando o identificador do exame e recebe dos dados do exame; O sistema então calcula a forma de melhor mostrar os dados de acordo com o tipo recebido.

Na figura 29 é mostrada a visualização de um sinal de ECG. O comando *Rotacionar* modifica a orientação de horizontal para vertical. O comando Voltar traz a tela de lista de exames.

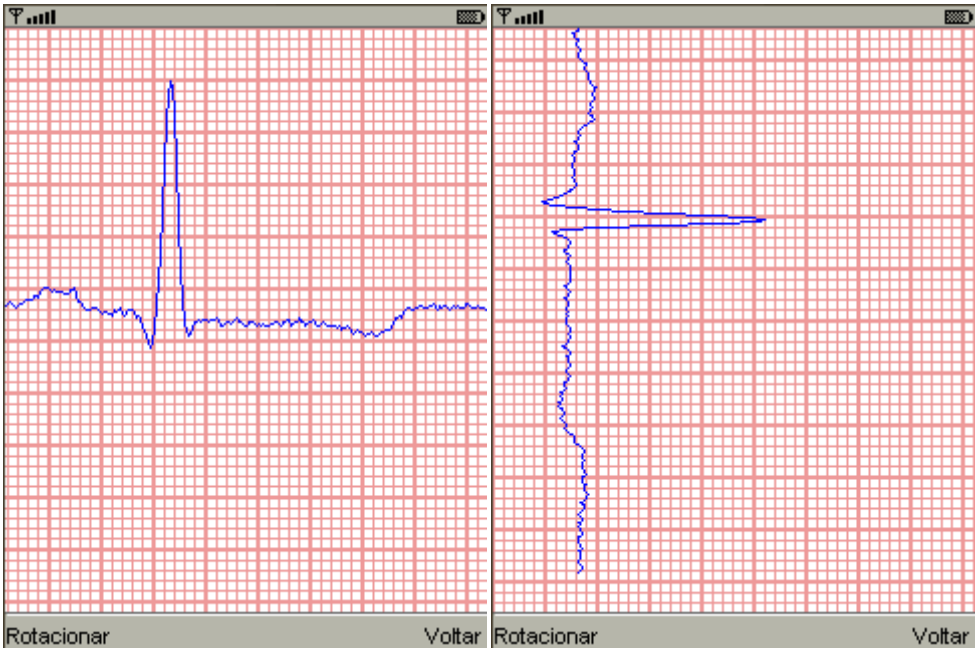

Figura 29 – Visualização do ECG.

Na figura 30 temos a visualização gráfica de um histórico de 10 amostras de pressão arterial. Cada conjunto de pressão sistólica e diastólica é desenhado usando uma barra com delimitação de valores máximo e mínimo. No eixo das abscissas são mostradas as datas de início e fim das amostras e no eixo das ordenadas temos o valor da pressão sistólica e diastólica.

Nesta tela também é mostrada outra funcionalidade do visualizador, que é a configuração das divisões dos eixos no fundo do gráfico. O eixo x possui duas divisões principais (mais escuras) e duas secundárias (mais claras). No eixo y temos quatro divisões principais e quatro secundárias.

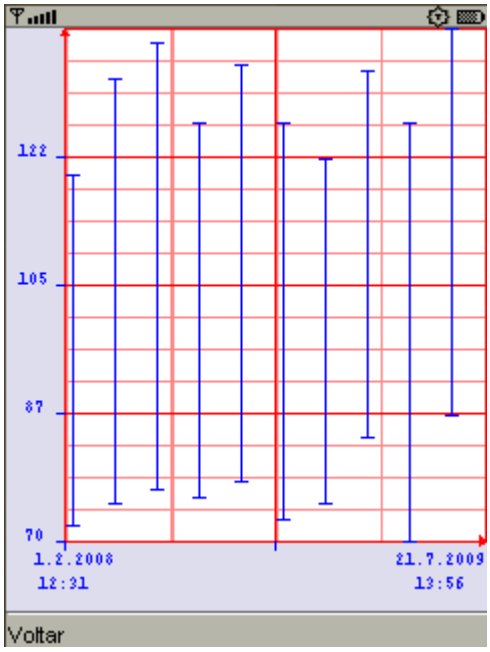

Figura 30 – Visualização de gráfico de histórico de pressão arterial.

A figura 31 mostra a visualização gráfica de um histórico de temperatura mostrando 35 amostras de temperatura. No eixo x temos cinco divisões principais e duas secundárias (mais claras). No eixo y temos quatro divisões principais e quatro secundárias.

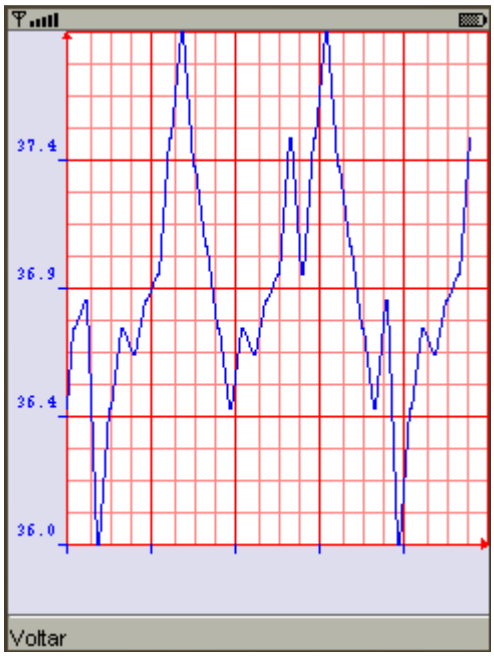

Figura 31 – Visualização de gráfico de histórico de temperatura.

A visualização de imagens utiliza um serviço RESTful para receber os dados. O visualizador manda a resolução da imagem que deseja receber. O serviço acessa a imagem (no banco de dados ou sistema de arquivos) e redimensiona antes de mandar a imagem para o dispositivo móvel.

Na figura 32 temos a imagem de um exame de raios-x. Ao pressionar o botão de ação o sistema faz uma requisição por uma imagem com uma resolução maior. Pressionando as teclas direcionais o sistema movimenta a imagem na tela.

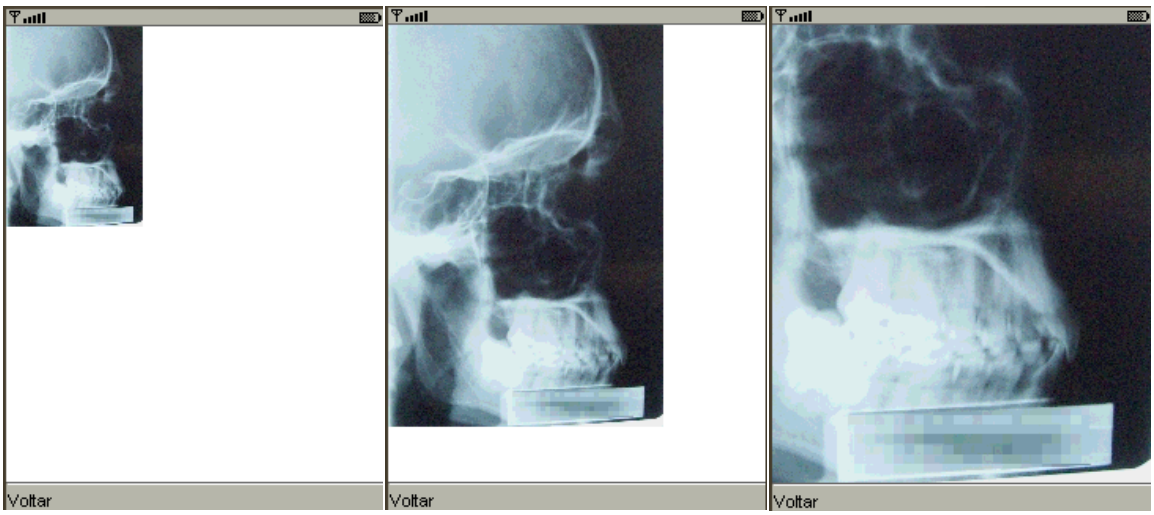

Figura 32 – Visualização de imagem de raio-x. Fonte da imagem [43].

3.2.10. Lista de Consultas Marcadas

A tela da lista de consultas é diferenciada. Quando um paciente está utilizando o sistema, a lista mostra as suas próximas consultas marcadas, mostrando a data e a especialidade médica da consulta. Quando um médico está utilizando o sistema, a tela mostra a lista de pacientes com consulta agendada.

Na figura 33 temos a visualização da lista de consultas marcadas para o profissional da saúde mostrando o nome do paciente e a data e hora da consulta. Ao pressionar o botão central em qualquer um dos pacientes o sistema traz as informações do paciente.

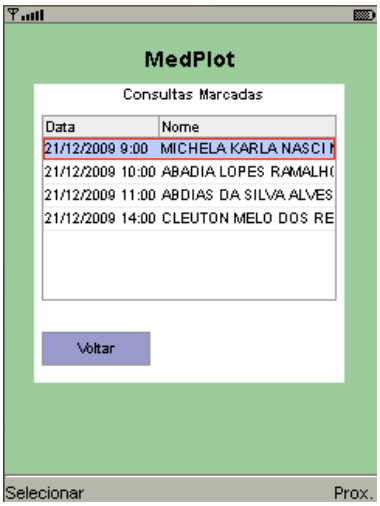

Figura 33 – Lista de consultas marcadas (profissional de saúde).

Na figura 34 temos a lista das próximas consultas do paciente.

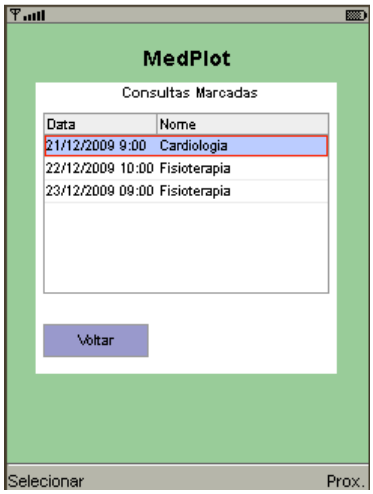

Figura 34 – Lista de consultas marcadas para o paciente.

3.3. GATEWAY DE COMPRESSÃO

O gateway de compressão desenvolvido funciona entre o visualizador e o WebService como mostrado na figura 21. Em vez do sistema mandar as requisições para o WebService, as requisições são comprimidas e direcionadas para o serviço de compressão que executa a chamada ao serviço. A resposta também volta comprimida. O nível de compressão utilizado nas requisições é o mais rápido que a biblioteca possui.

Nas tabelas 1 e 2 temos os ganhos obtidos utilizando esse método para melhorar o desempenho dos serviços.

Tabela 1 – Resultados da Utilização da Compressão na Requisição para o WebService

Ação	Tamanho da Requisição (bytes)		
	Sem Compressão	Comprimido	Ganho
Logon no Sistema	401	269	33%
Lista de Pacientes	482	320	34%
Lista de Consultas	498	325	35%
Informações do Paciente	471	319	32%
Lista de exames	477	322	32%
ECG	464	316	32%
Temperatura	464	316	32%
PA	464	316	32%

Tabela 2 – Resultados da Utilização da Compressão na Resposta do Webservice

Ação	Tamanho da Resposta (bytes)		
	Sem Compressão	Comprimido	Ganho
Logon	273	196	28%
Lista de Pacientes	521	347	33%
Lista de Consultas	419	277	34%
Informações do Paciente	497	332	33%
Lista de exames	427	228	47%
ECG	14628	3143	79%
Temperatura	431	204	53%
PA	459	269	41%

Como podemos observar nas requisições, o ganho médio é de 33% e o desvio padrão 1%, o que é expressivo se levarmos em consideração uma utilização continuada do sistema. Este ganho possibilita uma maior vida da carga da bateria. Nas respostas o ganho varia de 28% a 79%. No caso do ECG, utilizar um gateway de compressão (ou apenas a compressão antes de enviar) se mostra excelente tendo em mente a simplicidade do algoritmo utilizado.

Antes de utilizar a compressão na transmissão, o custo do processamento deve ser levado também em consideração. Geralmente o custo da transmissão é maior que o custo do processamento do algoritmo DEFLATE, mas isto pode variar de acordo com o hardware utilizado. A tabela 3 mostra o impacto da compressão da transmissão de 500 requisições de um sinal de ECG (tamanho 15KB cada).

O hardware utilizado para fazer este teste foi um iPAQ hx2495b que possui um processador Intel PXA270 de 520 MHz. Neste caso a transmissão utilizando a compressão utiliza menos energia do que a transmissão sem compressão.

Tabela 3 – Resultados da Utilização da Compressão no uso de energia.

500 ECG	Carga Inicial	Carga Final
Sem compressão	100%	91%
Com compressão	100%	95%

A grande desvantagem desta abordagem é a utilização de um protocolo não padrão, o que pode gerar dificuldades de implantação do sistema. Outra desvantagem é que o gateway é a manutenção do código em Java. Esta manutenção pode ser considerada complexa, uma vez que cria um protocolo utilizando a interface mais básica de comunicação (socket).

3.4. WEB SERVICE

Os serviços desenvolvidos foram implantados em um servidor de aplicação web *Tomcat* e utiliza a implementação de *WebServices* da Sun JAX-WS RI versão 2.1. O serviço GSWeb foi disponibilizado no contexto /MedPlotWebService/GSWeb e o serviço RESTful foi disponibilizado no contexto / MedPlotWebService/resources/imagem.

3.5. DISPOSITIVOS DE TESTES

Os dispositivos utilizados para os testes do sistema foram:

1. Sun wireless toolkit
2. IPAQ hx2495b
3. Sony PSP
4. Celular Sony Ericsson W980

O emulador da Sun foi utilizado para o desenvolvimento do sistema e para gerar as figuras deste documento. E para mostrar a diversidade de dispositivos que o J2ME permite, as figuras 35 e 36 a seguir mostram o MedPlot rodando no Playstation Portable da Sony usando a máquina Java PSPKVM [36] .

Figura 35 – MedPlot rodando em um PSP

Figura 36 – MedPlot mostrando um ECG em um PSP.

As figuras 37 a 39 mostram o sistema rodando no IPAQ 2495b rodando o Windows Mobile 6.1. Neste caso a máquina virtual utilizada foi a J9 da IBM.

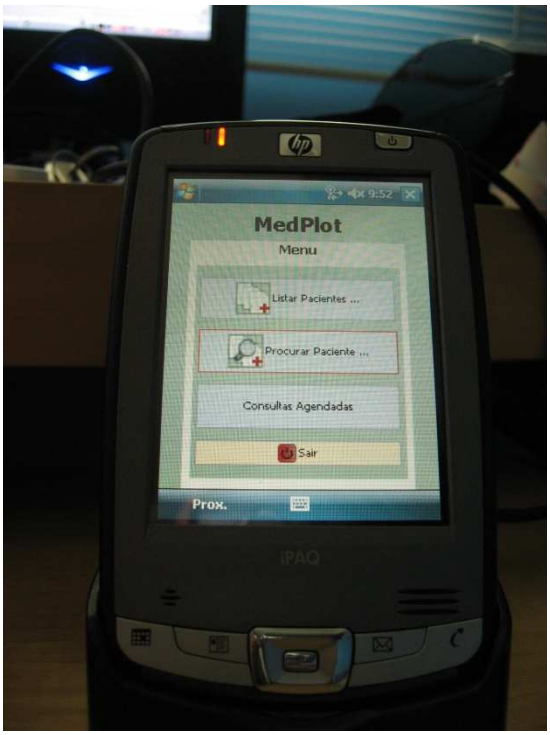

Figura 37 – MedPlot no IPAQ hx2495b

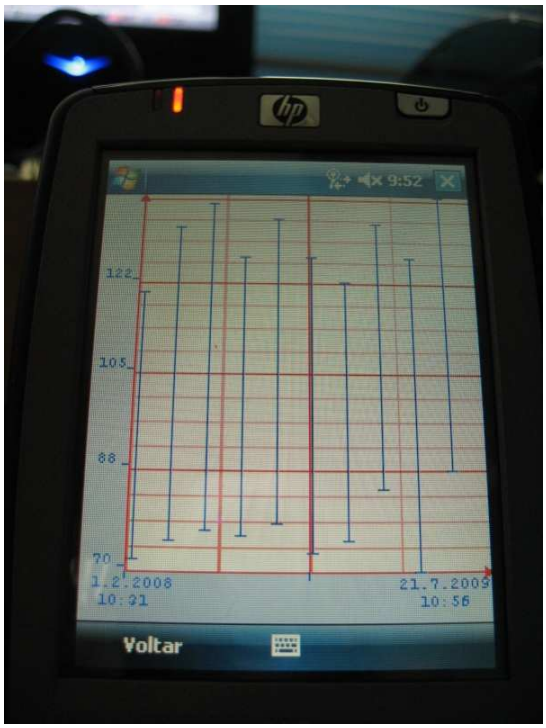

Figura 38 – MedPlot mostrando um gráfico de pressão arterial no iPaq

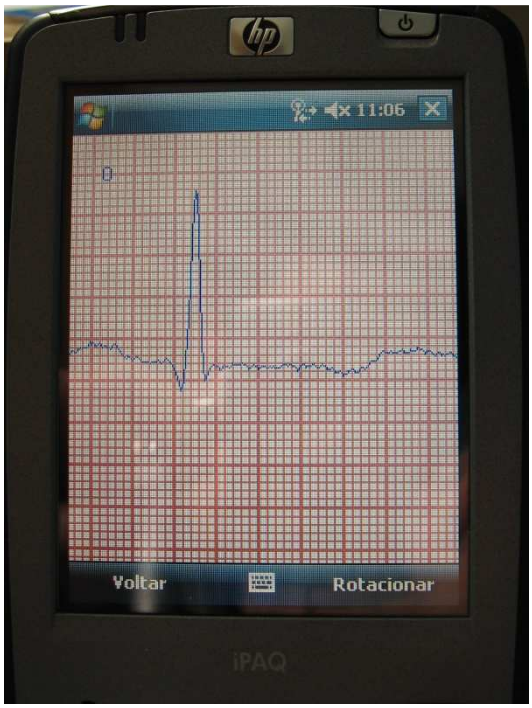

Figura 39 – MedPlot mostrando um ECG no iPaq.

As figuras 40 e 41 mostram o sistema rodando no telefone celular W980 da Sony Ericsson.

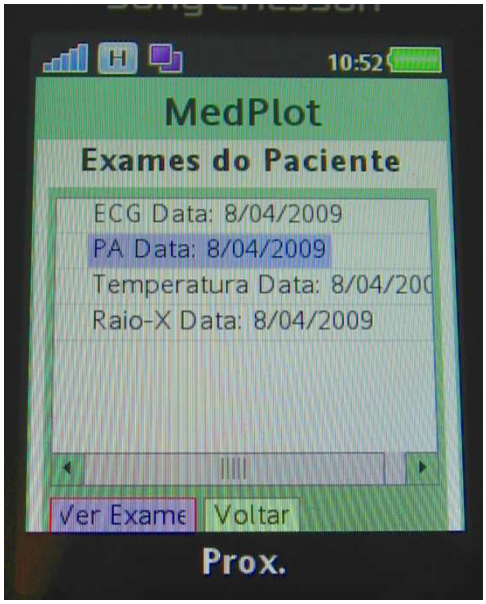

Figura 40 – MedPlot mostrando a lista de exames no W980 (Sony).

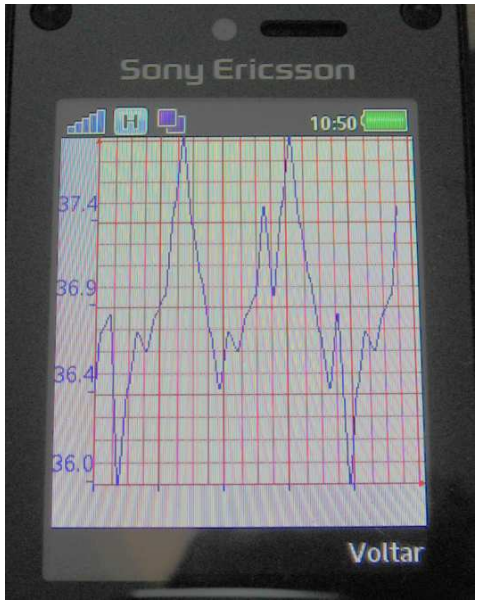

Figura 41 – MedPlot mostrando um gráfico de temperatura no W980 (Sony).

3.6. TESTES REALIZADOS

Para realizar os testes do sistema foi criada uma conta no site de DNS dinâmico dyndns.org. Nesta conta foi definido o endereço “medplot.dyndns.org” para o servidor de aplicação de modo a possibilitar os testes com o WebService. Durante os testes este endereço apontava para o IP dinâmico do computador que executava o servidor de aplicação.

Foi utilizado o servidor de aplicação Tomcat para rodar os serviços web durante os testes. Para configurar o acesso seguro aos serviços (HTTPS) primeiramente foi criada uma autoridade certificadora utilizando o OpenSSL. Esta autoridade certificadora foi utilizada para assinar o certificado x509 do servidor medplot.dyndns.org. Antes de testar o sistema o certificado da autoridade certificadora foi instalado nos dispositivos de testes para possibilitar o acesso seguro utilizado o SSL/TLS.

Os dispositivos de testes foram o emulador, o celular W980 e o iPAQ hx2495b. O PSP não foi utilizado pois não possuía ainda o acesso por HTTPS. Na tabela 4 temos os resultados dos testes realizados.

Tabela 4 – Testes dos casos de usos do sistema

Caso de Uso	Emulador	W980	iPAQ
Login	OK	Parcial	OK
Buscar Paciente	OK	Parcial	OK
Visualizar Cadastro	OK	Parcial	OK
Listar Consultas	OK	Parcial	OK
Listar Pacientes	OK	Parcial	OK
Listar Exames	OK	Parcial	OK
Visualizar Exames	OK	OK	OK

No W980 os testes foram apenas parcialmente corretos. Isto ocorreu devido ao tamanho da fonte padrão instalada no celular pelo fabricante. Mesmo utilizando a menor fonte do celular o texto ficava maior que o tamanho dos botões e das caixas de texto.

4. DISCUSSÃO E CONCLUSÕES

4.1. DISCUSSÃO E CONCLUSÕES GERAIS

O sistema desenvolvido possibilita o acesso aos dados (sinais) e informações biomédicas em uma vasta gama de dispositivos móveis que trabalhem com MIDP 2.0 e CLDC 1.1. Entretanto pode ser necessárias algumas adaptações de interface em alguns dispositivos móveis tipo celular. A implementação de comunicação baseada em WebService também garante uma maior interoperabilidade do sistema desenvolvido com diversos outros sistemas.

O tamanho do aplicativo, após a compressão (“obfuscation”) das classes (retirada das classes não utilizadas do aplicativo), é menor que 200 KB o que permite instalar em diversos celulares atualmente.

O acesso aos serviços web (usando SOA), de forma segura e eficiente nos dispositivos móveis, possibilita o desenvolvimento de sistemas com um grau de complexidade elevado em dispositivos com baixo poder computacional.

A visualização de sinais proposta no presente trabalho mostra-se eficiente para os dispositivos com uma resolução de tela pequena. Em dispositivos com uma resolução maior, uma solução web provavelmente seria uma opção melhor que o apresentado neste trabalho em relação à produtividade no desenvolvimento do software. Quanto à otimização do uso de energia, o sistema desenvolvido é mais eficiente que um baseado em web. Isto ocorre pois a implementação é mais simples e mais específica. Um browser com suporte a HTML, CSS e Javascript requer mais recursos do que o proposto no presente trabalho.

Outra vantagem deste sistema é a possibilidade de acessar os dados biomédicos diretamente de um sistema de captura de sinais formando um monitor multiparamétrico móvel. Esta possibilidade necessita ainda do desenvolvimento do software para acessar o hardware que captura os sinais.

A compressão dos dados no acesso ao serviço web mostra que é possível ter uma economia

considerável (30%) na transmissão de dados. Isto gera uma maior economia de energia e retira parte da desvantagem de se usar um serviço web (Custo do XML e do SOAP). A desvantagem está no aumento da complexidade na manutenção do sistema.

O desenvolvimento deste trabalho criou a base de um sistema genérico para a visualização de dados em dispositivos móveis. Os implementos da adaptação da biblioteca Thinlet para o J2ME e o visualizador genérico possibilitam a utilização da base do sistema para o desenvolvimento de outros sistemas para celulares.

Em comparação com a interface padrão do J2ME a utilização da biblioteca Thinlet possibilita ganhos consideráveis. O XML que define a interface gráfica pode ser gerado, de forma simples, em tempo de execução facilitando o desenvolvimento de interfaces dinâmicas no dispositivo móvel. No J2ME esta generalização não existe e a interface do usuário é padrão (definida antes da compilação do software) sendo atualizado apenas o conteúdo dinâmico. Por exemplo, com o uso da biblioteca Thinlet a interface inteira da tela pode ser definida no WebService que retorna o XML definindo dinamicamente a interface no dispositivo móvel.

4.2. RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Em trabalhos futuros o sistema desenvolvido pode ser modificado para ser inicializado trabalhar com mensagens SMS. Os dados de um exame podem ser enviados via SMS para um profissional da saúde e serem visualizados no sistema.

Outra modificação desejada é o desenvolvimento de um sistema para receber dados de um hardware de captura de sinais utilizando porta ZIGBEE e/ou Bluetooth em celulares ou a porta serial/USB de um PDA.

Embora tenha sido feito um grande trabalho na biblioteca Thinlet algumas questões de ainda estão em aberto como: a variação do tamanho da fonte em dispositivos diferentes; melhor orientação a objeto da biblioteca para melhorar o desempenho do obfuscador. Outra modificação desejável nesta biblioteca é a substituição da função de desenho de texto por uma função que seja independente da máquina Java instalada no dispositivo.

5. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Total de Acessos Móveis Por UF , de 18/03/2008. [http://www.anatel.gov.br/Portal/exibirPortalRedireciona.do?codigoDocumento=210506&caminhoRel=Cidadao-Telefonia%20M%F3vel-Dados%20do%20SMP] em 09/04/2008.

[2] Canalys research release 2008/021. [http://www.canalys.com/pr/2008/r2008021.pdf] em 09/04/2008.

[3] Web Service Processing and Interaction Models [http://java.sun.com/blueprints/webservices/using/webservbp2.html] em 03/01/2010.

[4] e-Ping Governo Eletrônico. [https://www.governoeletronico.gov.br/acoes-e-projetos/e-ping-padroes-de-interoperabilidade] em 03/01/2010.

[5] Comitê Executivo de Governo Eletrônico, e-PING Padrões de Interoperabilidade de Governo Eletrônico. Documento de Referência Versão 3.0, 2007.

[6] JSR-000224 - Java™API for XML-Based Web Services. Sun Microsystems, Inc.

[7] Introduction to Web Services. [http://www.netbeans.org/kb/60/websvc/intro-ws.html] em 03/01/2010.

[8] OASIS Standard Specification - Web Services Security: SOAP Message Security 1.1 [http://www.oasis-open.org/committees/download.php/21257/wss-v1.1-spec-errata-os-SOAPMessageSecurity.htm] em 03/01/2010.

[9] Dierks T., Allen C. Request for Comments: 2246 - The TLS Protocol.

[10] Ortiz, E. A Survey of J2ME Today. Outubro 2004, Sun Microsystems.

[http://developers.sun.com/techtopics/mobility/getstart/articles/survey/] em

[11] Apime framework. [http://www.java4ever.com/index.php?section=j2me &project=apime] em

[12] Coleção de componentes byblos. [http://code.google.com/p/byblos/] em 03/01/2010.

[13] Fire j2me - http://sourceforge.net/projects/fire-j2me] em 03/01/2010.

[14] J2ME Lightweght Visual Component Library - http://www.lwvcl.com/j2me.php] em 03/01/2010.

[15] J4ME: Java For Me - http://code.google.com/p/j4me/] em 03/01/2010.

[16] Biblioteca jMobileCore - http://jmobilecore.sourceforge.net/] em 03/01/2010.

[17] kUI - http://www.kobjects.org/kui] em 03/01/2010.

[18] Micro Window Toolkit – MWT - http://j2me-mwt.sourceforge.net/] em 03/01/2010.

[19] Nextel Open Source Tollkits - http://nextel.sourceforge.net/] em 03/01/2010.

[20] Biblioteca Thinlet – http://thinlet.sourceforge.net/home.html] em 03/01/2010.

[21] Biblioteca Synclast UI - http://www.synclast.com/ui_api.jsp] em 03/01/2010.

[22] Web Service Security Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0

[23] Ortis R. Monitorização de Sinais Biomédicos em Assistentes Pessoais Digitais. Relatório de Projeto Final de Graduação. 2004.

[25] Online Trading Concepts. http://www.onlinetradingconcepts.com /TechnicalAnalysis/Candlesticks/CandlestickBasics.html] em 03/01/2010.

[26] Pico Projector Displays. [http://www.microvision.com/pico_projector_displays/] em 01/06/2008

[27] Biblioteca kSOAP2 – [http://ksoap2.sourceforge.net/] em 02/06/2006

[28] The Legion Of The Bouncy Castle – [http://www.bouncycastle.org/java.html] em 30/05/2006

[29] API criptográfica BouncyCastle – [http http://bouncycastle.org/licence.html] em 30/05/2006

[30] Biblioteca jzlib – [http://www.jcraft.com/jzlib/] em 03/01/2010.

[31] Biblioteca zlib – [http://zlib.net/] em 03/01/2010.

[32] Retrieving Data from Web Services using Standard HTTP 1.1 Compression. [http://www.dotnetjunkies.ddj.com/Tutorial/90D3B3E0-6544-4594-B3BA-E41D8F381324.dcik] em 09/06/2006.

[33] GS-WEB - Gestão de Saúde via Web. [http://www.humanoweb.com.br/ solucoes/solGsweb.php] em 10/06/2008

[34] Deutsch P., Gailly J.L. RFC:1950 - ZLIB Compressed Data Format Specification version 3.3. 1996.

[35] Deutsch P. RFC: 1951: DEFLATE Compressed Data Format Specification version 1.3. 1996.

[36] Máquina virtual KVM para o PSP. [http://www.pspkvm.com/] em 03/01/2010.

[37] Biblioteca J2ME Polish. [http://www.j2mepolish.org] em 03/01/2010.

[38] Carvalho H., Coelho C., Heinzelman W. Gerenciamento de Informações Médicas do Paciente (Projeto GIMPA). CBIS 2002.

[39] Carvalho H., Coelho C., Heinzelman W., Murphy A. Body-worn sensor networks. Computers in Cardiology, 2003.

[40] Sene JR, I. G. (2009). Arquitetura para Desenvolvimento de Aplicação de Rede de Sensores para o Monitoramento da Saúde Humana. Tese de Doutorado em Engenharia Elétrica, Publicação PPGENE.TD-035/09, Departamento de Engenharia Elétrica, Universidade de Brasília, DF, 178p.

[41] Imagem de domínio publico. [http://en.wikipedia.org/wiki/File:Candlestick-chart.png] em 03/12/2009.

[42] iUI 0.13 - An Overview. [http://www.k10design.net/articles/iui/] em 03/01/2010.

[43] Lateral projection of the paranasal sinuses. [http://en.wikipedia.org/wiki/File:X-Ray_Skull.jpg] em 03/12/2009.