DESENVOLVIMENTO DE UMA FERRAMENTA UTILIZANDO A TECNOLOGIA J2ME

Ana Paula A. ZANELATO[1]
Eliezer Gomes Paraganba FILHO[2]
Emerson Silas DÓRIA[3]

RESUMO: Este artigo pretende apresentar a tecnologia J2ME (linguagem Java para computação móvel), por meio de uma solução de *software* para dispositivos móveis, uma alternativa de integração da tecnologia Java para micros dispositivos no âmbito acadêmico. Tal solução tem como objetivo facilitar a consulta de notas do aluno, sendo que essa consulta será realizada remotamente de um dispositivo móvel diretamente para a base de dados acadêmica. Por tratar-se da tecnologia Java a aplicação irá funcionar independente de plataforma, bastando somente que o dispositivo suporte a configuração e o perfil desejado.

Palavras-chave: J2ME; Configuração; Perfil; Dispositivos Móveis; Mobilidade.

1 INTRODUÇÃO

Devido ao rápido desenvolvimento ocorrido nas áreas de comunicação móvel, atualmente, é possível obter informações a qualquer hora e lugar. Apesar das limitações tecnológicas e físicas dos dispositivos móveis, é possível utilizá-los para

[1] Discente do curso de Pós-Graduação em Desenvolvimento de Sistemas para Ambientes Web baseados em Tecnologia Java da Universidade do Oeste Paulista. anapaulazanelato@gmail.com.
[2] Docente do curso de Pós-Graduação em Desenvolvimento de Sistemas para Ambientes Web baseados em Tecnologia Java da Universidade do Oeste Paulista. Especialista em aplicações para a web utilizando EJB pela Universidade Estadual de Maringá. eliezer@unoeste.br. Orientador do trabalho.
[3] Docente do curso de Pós-Graduação em Desenvolvimento de Sistemas para Ambientes Web baseados em Tecnologia Java da Universidade do Oeste Paulista. Mestre Estudos Empíricos em Engenharia de Software pela Universidade de São Paulo. emerson@unoeste.br. Co-orientador do trabalho.

oferecer diversas aplicações, como por exemplo, checar a caixa de e-mails pelo celular ou atualizar a agenda de compromissos direto do servidor de sua empresa. A mobilidade aliada à tecnologia sem fio trouxe para o mercado um novo paradigma de comunicação**:** a comunicação móvel sem fio.

Este artigo tem como objetivo apresentar a tecnologia J2ME (linguagem Java para computação móvel), tendo como exemplo uma aplicação de consulta de notas pelo celular. Sendo assim, a aplicação irá funcionar independente de plataforma, bastando somente que o dispositivo suporte a configuração e o perfil desejado.

2 DESENVOLVIMENTO

2.1 Introdução ao J2ME

2.1.1 Edições de Java

Desde o lançamento da linguagem Java, em 1995, a idéia era desenvolver uma linguagem que seu código fosse escrito apenas uma vez, e então fosse executada em qualquer plataforma que suportasse uma máquina virtual Java (*"Write once, Run Anywhere"*). A linguagem Java ampliou muito seu alcance, dois anos após a introdução da linguagem, atualmente conhecida como versão J2SE (*Java 2 Standard Edition*), uma nova edição foi lançada, a *Java 2 Enterprise Edition* (J2EE), fornecendo suporte para aplicativos empresariais. A última versão lançada da família é a *Micro Edition* (J2ME), desenvolvida para dispositivos móveis, variando desde máquinas ligadas à TV habilitadas para Internet a telefones celulares.

Em resumo, as plataformas Java atualmente disponíveis são:

- *Standard Edition* (J2SE): objetiva o seu uso em computadores pessoais e estações de trabalho.

- *Enterprise Edition* (J2EE): destinada a aplicações executadas no servidor, fornece suporte para Servlets, JSP e XML.
- *Micro Edition* (J2ME): projetada para dispositivos com memória, vídeo e poder de processamento limitados.

2.1.2 J2ME

Segundo RABELLO (p .03):

> J2ME *(Java 2 Platform Micro Edition)* pode ser definido como uma especificação da plataforma Java que foi projetada para suportar aplicações desenvolvidas para dispositivos móveis sem fio, como *PDAs, pagers*, telefones celulares, entre outros. Nesta especificação estão inclusos máquinas virtuais Java (JVM) e um conjunto de padrões da Java API desenvolvido pelo *Java Community Process (JCP)*.

O uso de J2ME é destinado aos dispositivos com poder limitado, muitos desses dispositivos não possuem opção de download e software de instalação, além do que foi realizado durante o processo de fabricação, uma implementação de J2ME em um dispositivo permite a opção de navegar, realizar downloads e instalar aplicações Java.

Os dispositivos móveis nos permitem comunicar em qualquer lugar, acessar e-mails, navegar na internet e executar aplicativos. Com a introdução do Java nesses dispositivos, temos acesso aos recursos da linguagem e da plataforma Java, sem mencionar as qualidades da linguagem, fácil domínio, plataforma segura e portável, acesso ao conteúdo dinâmico e milhares de desenvolvedores.

Devido a todas as limitações dos dispositivos móveis, não é possível ter todas as funcionalidades da API (*Application Program Interface,* interface de aplicação de aplicativos) J2SE, por isso foi lançada a *Micro Edition*, para tratar dessas necessidades especiais, que não estão fora da abrangência do J2SE e J2EE.

Apesar da especificação, os recursos dos dispositivos que estão dentro da *Micro Edition* podem variar bastante, como por exemplo no tamanho da tela. Para que essa diversidade de recursos seja acomodada dentro do J2ME, será necessário entender dois conceitos: as configurações e os perfis. Basicamente, os perfis são mais específicos que as configurações, e fazendo uma analogia, temos uma

abstração sobre o que é um carro e como ele é fabricado (configuração) e como um Ford é fabricado (perfil).

2.1.3 Configurações

Com o objetivo de suportar a variedade de recursos, a Sun introduziu a Configuração. De acordo com a *Sun Microsystems (2002)*, “uma configuração diz respeito a dispositivos que possuem restrições e limitações de recursos, como poder de processamento, conectividade de rede, de memória e de capacidade gráfica.”.

Uma Configuração esta totalmente ligada a uma máquina virtual Java (JVM, *Java Virtual Machine*), e define os recursos da linguagem e as bibliotecas básicas da JVM para essa configuração.

A “versão móvel” de Java dá suporte a dois tipos de configurações denominadas *CLDC* (*Connected Limited Device Configuration*) e *CDC (Connected Device Configuration).*

2.1.4 Perfis

A Sun introduziu o conceito de perfil na plataforma J2ME para tratar da variação de recursos dentro de uma mesma configuração.

“Um perfil é uma extensão de uma configuração. Ele fornece as bibliotecas para um desenvolvedor escrever aplicativos para um tipo particular de dispositivo.” (MUCHOW, 2004, p. 05). Como por exemplo, o MIDP (*Mobile Information Device Profile,* Perfil de Dispositivo de Informação Móvel) e também o PDAP (*Personal Digital Assistant Profile*) e ambos estão acima do CLDC.

2.1.5 Máquinas Virtuais Java

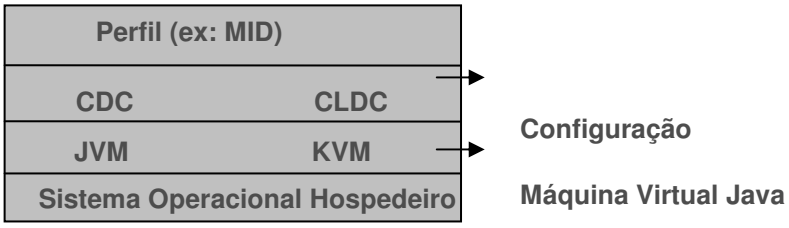
Uma JVM transforma os arquivos de classe em Java (o código de byte nos arquivos de classe) no código de máquina para a plataforma que está executando a JVM. A JVM também é responsável por fornecer segurança, alocar ou não a memória, e gerenciar linhas de execução.

Para a configuração CDC, a máquina virtual é a mesma do J2SE, entretanto, para a CLDC, a Sun desenvolveu uma máquina virtual projetada para manipular as necessidades especiais dos dispositivos de recursos limitados, que ficou conhecida como *K Virtual Machine* (KVM).

2.1.6 Arquitetura

Em resumo, a arquitetura de software do J2ME pode ser apresentada como na figura abaixo, onde inicia com o sistema operacional hospedeiros, seguida da máquina virtual, que poderá assumir duas formas: JVM (para sistemas compatíveis com CDC) ou KVM (para sistema que atendam as especificações da CLDC). A configuração é a próxima na hierarquia, representadas pela CDC e CLDC. Os perfis é a camada superior e fornecem um kit de ferramentas para desenvolver aplicativos para uma família de dispositivos especifica, neste caso será o perfil MID.

FIGURA 1- Arquitetura J2ME



Fonte: a própria pesquisadora

2.2 CLDC (CONFIGURAÇÃO DE DISPOSITIVO CONECTADO LIMITADO)

A CLCD possui dois objetivos, o primeiro é definir uma especificação para uma JVM, e o segundo é definir um conjunto de classes Java. Entretanto ambos objetivos possuem uma função em comum, que devem fornecer suporte para diversos dispositivos com memória, capacidade de vídeo e recursos limitados.

2.3 O Conjunto de MIDlets

Um MIDlet é um aplicativo Java, que possui como classes básicas a CLDC e o MIDP. Foi projetado para ser executado em um dispositivo móvel. Um

conjunto de MIDlets pode ser formado por um ou mais MIDlets, que são empacotados em um único arquivo JAR (*Java Archive*).

2.3.1 JAR (Arquivo Java *Archive*)

Geralmente, um aplicativo consiste em muitos arquivos, sejam as classes Java, ou outros arquivos, como imagens e dados de aplicativos. Para facilitar a distribuição, esses arquivos são empacotados em uma única entidade conhecida como arquivo JAR.

2.3.2 JAD (Arquivo Java *Application Descriptor*)

Um arquivo JAD fornece informações sobre o(s) MIDlet(s) dentro do arquivo JAR. O objetivo do arquivo JAD é fornecer informações para o gerenciador de aplicativos sobre o conteúdo de um arquivo JAR, e disponibilizar um meio para que parâmetros sejam passados para um ou mais MIDlet(s), sem fazer alterações no arquivo JAR.

2.4 PROGRAMAÇÃO COM MIDLETS

"Um MIDlet é um aplicativo construído com a classe MIDlet." (MUCHOW, 2004, p.61). A comunicação do MIDlet com o gerenciador de aplicativos, se dá por meio dos métodos, sendo que tanto o gerenciador de aplicativos pode pausar um MIDlet para que o usuário atenda a uma chamada telefônica, quanto o MIDlet pode fazer um pedido de pausa.

2.4.1 Ciclo de Vida do MIDlet

Um MIDlet sempre estará em um dos estados mencionados abaixo durante seu ciclo de vida:

- Pausa: um MIDlet encontra-se no estado de pausa após a chamada do construtor e antes de ser iniciado pelo gerenciador de aplicativos. Durante o ciclo de vida, um MIDlet pode alternar entre o estado de ativo e pausado qualquer número de vezes.

- Ativo: quando o MIDlet está em execução.
- Destruído: este estado ocorre quando o MIDlet já liberou todos os recursos que usou e foi desligado pelo gerenciador de aplicativos.

2.4.2 Display

O objeto *Display* pode ser definido como o gerenciador da tela que controla o que é mostrado na tela do dispositivo e quando. Cada MIDlet referência apenas um único *Display*, que pode capturar características da tela atual do dispositivo, como por exemplo, quantas cores são suportadas, e exibir diversos objetos na tela, como *Forms, TextBox*, etc. Apesar do MIDlet possuir um único *display*, ela pode exibir diversos objetos (*Forms,TextBox, ChoiceGroups*, etc), porém, em um dado momento, apenas um desses objetos poderá ser exibido na tela.

O objeto *display* pode tornar-se disponível em um MIDlet por meio da chamada do método estático *Display.getDisplay(MIDlet m)*.

Os objetos que podem ser exibidos no *Display*, como *Forms,TexBox, Canvas*, etc., são todos herdados da classe *Displayable*.

2.4.2.1 Displayable

Como já foi mencionado um objeto *Displayable* é aquele que pode ser exibido em um dispositivo. O MIDP inclui duas subclasses de *Displayable: Screen* e *Canvas*. Os objetos da classe *Screen* (*TextBox, List, Form* e *Alert*) são componentes de alto nível de interface com o usuário. A classe *Canvas* é usada para elementos gráficos personalizados e tratamento de eventos de baixo nível.

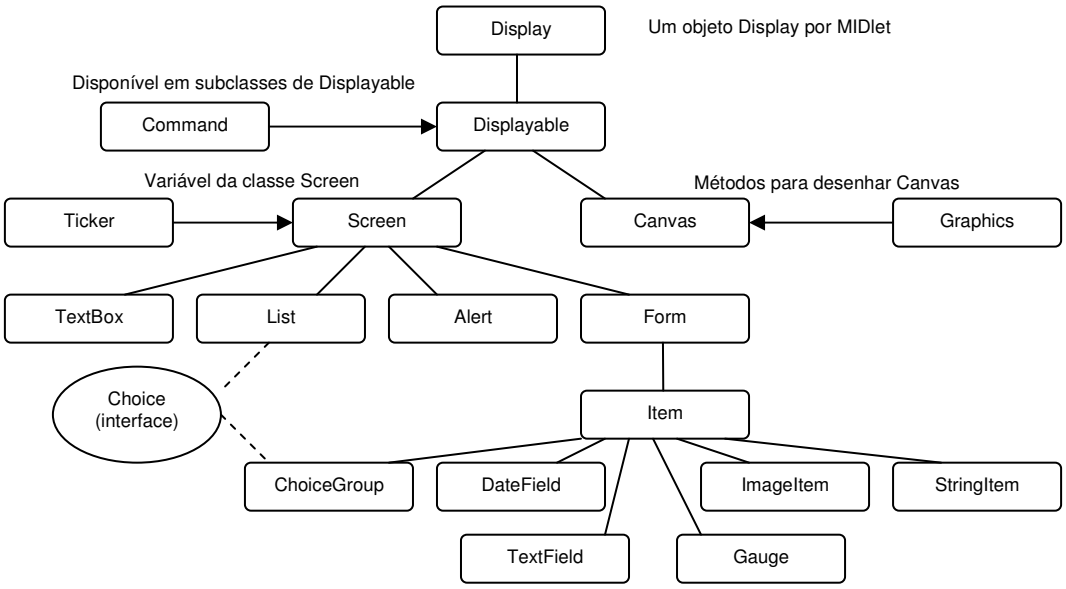
A implementação da classe *CommandListener* permite que cada um dos formulários gerencie seus próprios eventos. Os objetos *Command* podem fornecer um meio de interação com o usuário, são objetos que possuem informações sobre uma ação, como por exemplo, um usuário pedindo para sair de um MIDlet.

Em conjunto com o objeto *Command* existem os receptores, para receber os eventos. Quando um evento ocorrer, o receptor chamará o método *public void commandAction(Command c, Displayable s)*. Este método é capaz de identificar qual objeto *command* iniciou o evento.

2.5 INTERFACE COM O USUÁRIO

Neste capítulo será apresentado como se dá a interface com o usuário em um MIDlet. A Classe *Screen*, que é um dos dois objetos *Displayable* será focalizada, especificamente o objeto *Form*, derivado desta classe.

FIGURA 2- Hierarquia da classe *Displayable*.



Fonte: a própria pesquisadora

2.5.1 *Screen*

Screen é uma classe progenitora de componentes que têm uma aparência e um comportamento na tela. A classe *Screen* e suas subclasses (*Form, List TextBox* e *Alert*) são componentes de interface com o usuário de alto nível.

2.5.1.1 *Form*

Um objeto *Form* pode ser visto como um contêiner que pode ter qualquer número de componentes, em que cada um é uma subclasse de Item. Este objeto apresenta métodos para anexar, inserir, substituir e excluir componentes.

2.5.1.2 *Item*

Item é um componente que pode ser adicionado em um *Form*. As subclasses de Item são: *ChoiceGroup, DateField, Gauge, ImageItem, StringItem* e *TextField*. A classe *ItemStateListener* trabalha em conjunto com a classe Item, para processar eventos em um objeto Item.

2.5.1.3 *List*

O Objeto *List* estende a classe *Screen*, assim como um objeto *Form*. Pode-se dizer que, esses objetos encontram-se no mesmo nível como mostrado na Figura 1. A diferença, é que um objeto *Form* pode ser visto como um container de objetos Item, alocando esses objetos na tela, enquanto o objeto *List*, assim como *TextBox* e *Alert*, operam independentemente. Quando um objeto *List* é configurado como tela ativa, ele é o único componente visível, sem levar em consideração os objetos *Command*.

2.5.1.4 *TextBox*

O objeto *TextBox* é uma tela de entrada de texto de várias linhas. É possível filtrar a entrada do usuário inserindo restrições, assim como as restrições descritas para o objeto *TextField*.

2.5.1.5 *Alert*

Um objeto *Alert* é uma caixa de diálogo que suporta um texto e um objeto *Image*, seu uso mais comum é para exibir mensagens de aviso e erro. Este objeto possui três atributos: título, imagem e texto.

2.6 ESTRUTURA DE CONEXÃO GENÉRICA (GCF)

A GCF (*Generic Connection Framework,* Estrutura de Conexão Genérica) foi desenvolvida para fornecer uma estrutura extensível para I/O e interligação em rede nos dispositivos móveis, levando em consideração suas limitações e variações. As classes da GCF formam um subconjunto das classes e interfaces do J2SE que dão suporte para protocolos de rede e sistemas de arquivo, com o objetivo de serem suportadas nos dispositivos que implementam o MIDP.

2.6.1 Conexão HTTP

De acordo com MUCHOW (2004, p.418):

> A classe *HttpConnection* permite que um servidor Web se comunique com qualquer outro dispositivo remoto que suporte o protocolo HTTP. O HTTP pode ser definido como um protocolo de pedido/resposta. Um cliente inicia um pedido, envia-o para um servidor com o endereço especificado como um URL (*Uniform Resource Locator*) e uma resposta é retornada do servidor.

2.6.1.1 Requisição do Cliente

Como já foi dito, o HTTP é um protocolo de pedido/resposta, um cliente realiza uma solicitação e o servidor envia uma resposta. Um pedido de cliente consiste em três seções: método do pedido, cabeçalho e corpo.

2.6.1.2 Resposta do Servidor

Após um cliente empacotar o método do pedido e enviar pela rede, o servidor deverá interpretar o pedido, processar e gerar uma resposta, conhecida como entidade de resposta. A resposta do servidor consiste em três seções: linha de status, cabeçalho e corpo.

2.6.1.3 Gerenciamento de Sessão

O protocolo HTTP funciona basicamente da seguinte forma, um cliente faz um pedido e um servidor gera uma resposta, sendo assim, não há nenhuma conexão persistente entre os dois que o protocolo possa oferecer. Entretanto, é muito interessante que o servidor possa reconhecer os pedidos subseqüentes de um cliente. Para que a interação entre o servidor e o cliente possa acontecer, é necessário que o servidor reconheça que os pedidos são do mesmo cliente, isso é

chamado de gerenciamento de sessão. Com um *Servlet* Java, é possível fazer isso de duas formas: reescrita de URL e *cookies*.

2.7 Proposta de Pesquisa

2.7.1 Formulação do Problema

Atualmente, com a disponibilidade de diversas redes, é possível obter informações a qualquer hora e lugar. Apesar das limitações tecnológicas e físicas dos dispositivos móveis, é possível cada vez mais oferecer diversas aplicações para serem executadas em celulares e PDAs. Isso se deve ao fato do avanço tecnológico que está ocorrendo na área da comunicação móvel, o objetivo é sempre facilitar a vida do usuário, oferecendo soluções mais rápidas e simples, e por conseqüência trazendo mais comodidade.

Dessa forma, um usuário que não tenha disponibilidade em um dado momento, de obter acesso à rede utilizando o computador, pode obter a mesma informação por meio de um dispositivo móvel. Existem diversas situações que podem ser citadas, por exemplo, um aluno que esteja em uma viagem dentro de um veículo, e que recebeu uma mensagem da faculdade neste exato momento, informando que suas notas foram disponibilizadas para consulta. Este aluno pode consultar suas notas acessando o servidor da faculdade por meio de um celular no mesmo instante, sem a necessidade de aguardar o momento em que teria acesso a um computador.

2.7.2 Objetivos e Justificativas

O objetivo deste trabalho é fornecer uma solução de software para dispositivos móveis, uma alternativa de integração da tecnologia Java para micros dispositivos no âmbito acadêmico. Tal solução tem como objetivo facilitar a consulta de notas do aluno, sendo que essa consulta será realizada remotamente de um dispositivo móvel diretamente para a base de dados acadêmica.

2.8 Desenvolvimento

Para o desenvolvimento das ferramentas, foi utilizado um computador *AMD Atlon XP 3500*, com, 1 GB de memória RAM, utilizando-se o sistema operacional Windows XP Professional e o ambiente de desenvolvimento *NetBeans ID 6.0*, existe uma versão disponível para download no site http://www.netbeans.org/ (último acesso em 16/02/2008).

O *NetBeans ID 6.0* possui integrado a ele a plataforma *J2ME Wireless Toolkit 2.2.* O *J2ME Wireless Toolkit* (também conhecido como *Sun Java Wireless Toolkit*) é um conjunto de ferramentas para criar aplicativos do *Java Micro Edition* (Java ME) compatíveis com as tecnologias Configuração de dispositivo limitado conectado (CLDC) e Perfil de dispositivo de informação móvel (MIDP). O *J2ME Wireless Toolkit* contém ferramentas para criação de aplicações, utilitários e um emulador de dispositivo.

Para disponibilizar os recursos do *Servlet* (tecnologia que insere novos recursos a um servidor, gerando conteúdos dinâmicos e interagindo com clientes por meio do modelo resquest/response) aos dispositivos móveis foi utilizado o servidor de aplicações *Apache Tomcat 5.5.17 Server*, existe uma versão disponível para download no site http://tomcat.apache.org/ (último acesso em 16/02/2008).

Por fim, para armazenar a base de dados acadêmica com informações dos alunos foi utilizado o banco de dados *PostgresSQL 8.1*, por se tratar de um gerenciador de banco de dados *free*, existe uma versão disponível para download no site http://www.postgresql.org/ (último acesso em 16/02/2008).

2.9 Protótipo

Este capítulo tem por objetivo realizar uma introdução quanto às ferramentas desenvolvidas a fim de validar os resultados desta pesquisa.

2.9.1 Arquitetura proposta

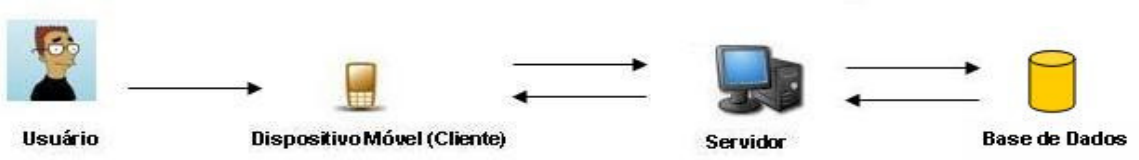
Para desenvolver a ferramenta foi necessário dividir o projeto em duas etapas. Na etapa inicial foi desenvolvido um *Servlet* que disponibiliza recursos de autenticação do aluno, consulta de disciplinas cursadas e notas, com acesso a base de dados.

Na etapa seguinte foi desenvolvida a aplicação para o dispositivo móvel, com uso do MIDlet. Esta aplicação se utiliza dos recursos do *Servlet* descrito acima, e mantém uma interação com o usuário, de forma que o mesmo possa se autenticar e visualizar suas disciplinas e notas.

2.9.2 Ferramenta SCN (Sistema para Consulta de Notas)

Como mencionado, a ferramenta SCN possui o MIDlet para o dispositivo móvel e o *Servlet* para disponibilizar os recursos necessários. O projeto para o dispositivo móvel possui um único MIDlet que se comunica com o *Servlet* por meio do protocolo HTTP. Após o primeiro acesso, é criada uma sessão entre o usuário e o servidor, utilizando para isso o gerenciamento de sessão com base na reescrita de URL (o gerenciamento por *cookies* não foi utilizado para evitar problemas de incompatibilidade com dispositivos e/ou redes).

FIGURA 3- Arquitetura da Ferramenta SCN



Fonte: a própria pesquisadora

2.9.2.1.1 Diagrama (MIDlet)

O *NetBeans ID 6.0* oferece um design visual para facilitar o desenvolvimento de aplicações móveis. A figura abaixo representa o fluxo do MIDlet.

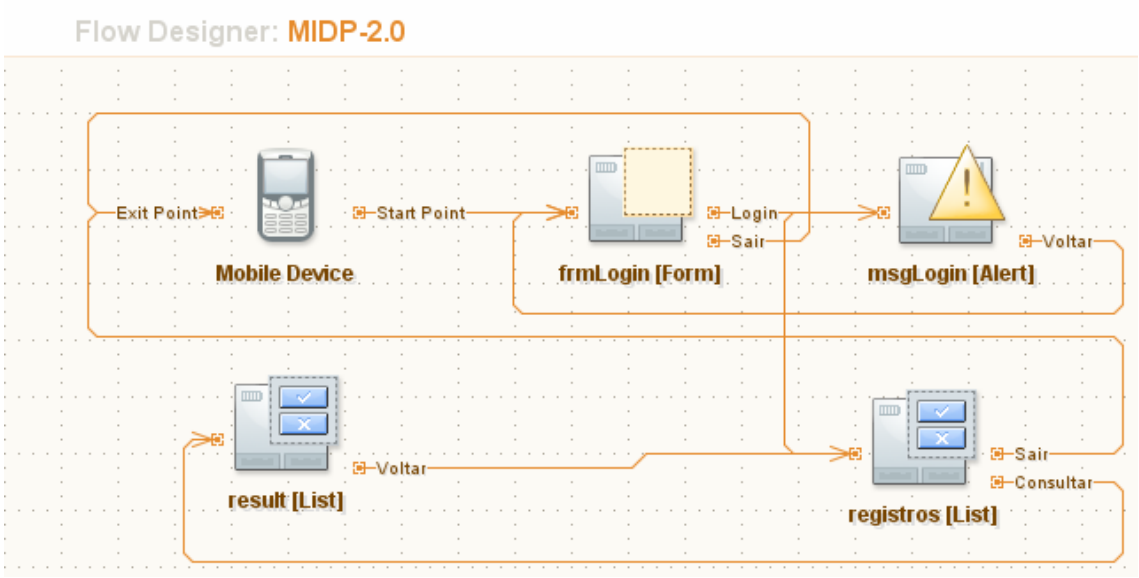
Assim que a aplicação é iniciada é exibido a tela de login (*FrmLogin*), que trata-se de um objeto do tipo *Form*. Esta tela apresenta dois objetos *Command* Sair e Login. Se escolhida a opção Sair, a aplicação será encerrada, caso contrário, será realizado uma validação do usuário. Caso a autenticação do usuário obtiver sucesso, será exibida uma lista com as disciplinas do aluno por meio do objeto *List*, chamado de registros. Se o retorno da autenticação obtiver fracasso, será exibida

uma mensagem por meio do objeto *Alert* chamado msgLogin. Neste último caso a única opção do usuário é de retornar a tela de login.

Caso o login tenha sido realizado com sucesso e as disciplinas estejam sendo exibidas no dispositivo, o usuário possui duas opções Sair (a aplicação será encerrada) ou Consultar (visualizar as notas da disciplina).

Ao selecionar a opção Consultar, será retornado as notas da disciplina selecionada e exibidas no objeto *List* chamado result. Após visualizar as notas o usuário possui apenas uma única opção que retornará à lista de disciplinas.

FIGURA 4- Design visual do fluxo do MIDlet



Fonte: Layout criado pelo software NetBeans

2.9.2.5 Disponibilização do aplicativo

Com o objetivo de disponibilizar a ferramenta SCN foram realizadas diversas ações. O *Servlet* foi disponibilizado em um servidor que executa o servidor de aplicações *ApacheTomCat*.

Para que aplicação móvel pudesse ser distribuída aos usuários foram criados os arquivos .JAR, com os arquivos do projeto referentes o MIDlet, e .JAD, com informações do arquivo .JAR.

Além disso, foi criada uma página em WML, linguagem baseada em XML e interpretada por dispositivos móveis, que apresenta um link para o arquivo

.JAD e por conseqüência o download do arquivo .JAR. A seguir o código da página WML criada para o projeto.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card title="Clique aqui para instalar " >
<p><a href="http://127.0.0.1/ConsultaNotasVisual.jad">Clique aqui para instalar</a></p>
</card>
</wml>
```

Assim, o dispositivo móvel acessa a página WML por meio de um navegador WML. O software de gerenciamento de aplicativos do dispositivo detecta o arquivo JAD e verifica os atributos desse arquivo. Se o dispositivo detectar que há alguma discrepância entre os requisitos, conforme estabelecido no descritor de aplicativo, e os seus recursos, normalmente ele tentará resolver o problema, inibindo ou não a continuação da instalação. Abaixo é exibido o código do arquivo .JAD utilizado neste projeto.

```
MIDlet-1: ConsultaNotasVisual,/res/imagem.png,app.ConsultaNotasVisual
MIDlet-Jar-Size: 68463
MIDlet-Jar-URL: http://localhost/ConsultaNotasVisual.jar
MIDlet-Name: ConsultaNotasVisual
MIDlet-Vendor: Vendor
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

Conforme já foi dito, o arquivo JAD especifica além de outras informações, a localização do arquivo JAR. Caso tudo esteja certo, o software de gerenciamento de aplicativos do dispositivo iniciará o download do arquivo JAR, a instalação e a execução do aplicativo.

2.10 Testes e Resultados

2.10.1 Considerações iniciais

Este capítulo tem por objetivo realizar uma metodologia de testes afim de validar o funcionamento da ferramenta SCN. Para visualização da aplicação, mostraremos telas capturadas durante a execução da mesma no emulador de dispositivos móveis do *NetBeans*. As principais telas obtidas durante a emulação da

aplicação serão mostradas a seguir. Ao final do capítulo será descrito os resultados obtidos.

2.10.2 Emulador

Para que pudesse ser realizada a comunicação do sistema móvel com o sistema servidor, vários testes foram realizados (testes de interfaces, de conexão, de desempenho). Para isso, foi utilizado o simulador *Sun Java Wireless Toolkit 2.2*, que pode ser definido como: "uma ferramenta de desenvolvimento do J2ME que vem empacotada com um simulador que pode simular vários telefones móveis, assim como *PDAs* e *pagers.*" (MUCHOW, 2004, p.521).

Este simulador permite que o programador configure seu ambiente de desenvolvimento a fim de "prever" qual seria o comportamento de uma aplicação J2ME rodando em um dispositivo móvel real. Variáveis reais como capacidade de armazenamento, velocidade de processamento e velocidade da rede podem ser configuradas.

Também foram realizados testes em dispositivos móveis reais, porém, as telas aqui apresentadas serão do simulador.

2.10.3 Processo de Execução

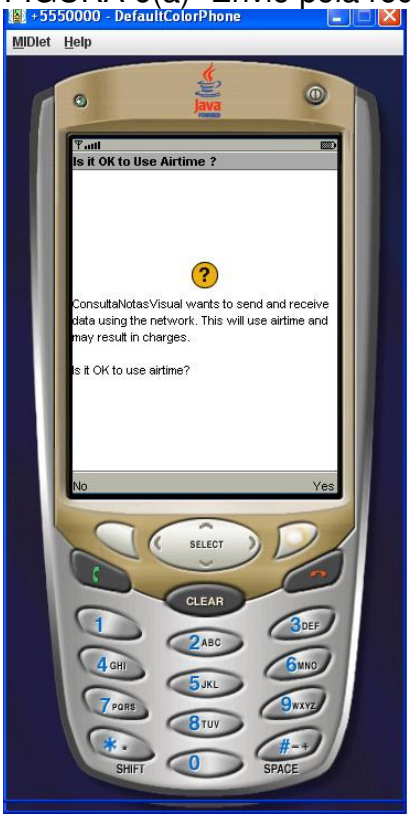
Ao iniciar a ferramenta é exibida a tela de login, como na figura abaixo. Nesta tela o usuário deve informar seu Registro Acadêmico e sua senha.

FIGURA 5- Tela de Login

Fonte: a própria pesquisadora

Ao escolher a opção Login, o dispositivo detecta a requisição do envio de dados ao servidor e questiona o usuário se deseja enviar os dados pela rede (FIGURA 6(a)). Após a confirmação do envio os dados, a conexão é estabelecida (FIGURA 6(b)) com o servidor, que valida os dados. Se estiverem incorretos, é exibida uma tela informativa com um sinal sonoro (FIGURA 6(c)), e em seguida, será retornada a tela de login para que o usuário possa digitar os dados corretos.

FIGURA 6(a)- Envio pela rede

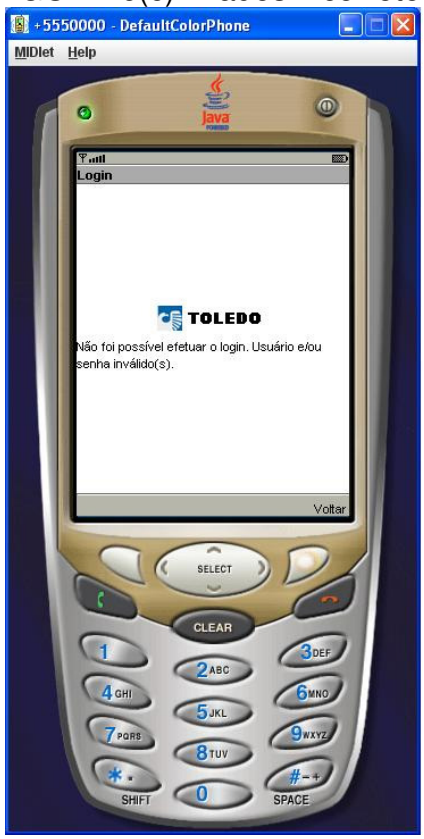


Fonte: a própria pesquisadora

FIGURA 6(b)- Estabelecendo conexão

Fonte: a própria pesquisadora

FIGURA 6(c)- Dados incorretos



Fonte: a própria pesquisadora

Caso o Registro Acadêmico e a senha do aluno estiverem corretos, o servidor irá retornar as disciplinas do aluno, que serão exibidas na tela como na FIGURA 7. É possível notar que no topo do objeto *List* é exibido o nome do aluno.

FIGURA 7- Disciplinas do aluno

Fonte: a própria pesquisadora

Para consultar as notas de uma determinada disciplina basta que o usuário selecione a disciplina e clique na opção Consultar, ou apenas clique no botão *Select* do dispositivo. Uma nova requisição será realizada ao servidor, que retornará as notas da disciplina escolhida, como mostrado na FIGURA 8.

FIGURA 8- Notas do aluno

Fonte: a própria pesquisadora

Para retornar a tela anterior o aluno deverá selecionar a opção Voltar, assim, retornará a tela de disciplinas, podendo realizar outra consulta ou finalizar a aplicação selecionando a opção Sair.

3 CONCLUSÃO

No decorrer deste artigo, tivemos a oportunidade de analisar de forma detalhada a tecnologia J2ME. Essa análise se deu através da descrição e exemplificação dessa tecnologia com relação a um breve histórico, arquitetura, organização e configuração. Foi desenvolvida uma aplicação de Consulta de Notas

(SCN) baseada nessa tecnologia. Essa aplicação serviu para colocar-mos em prática alguns dos conceitos expostos por esse artigo.

No entanto, ao decorrer do desenvolvimento desse artigo, algumas dificuldades foram encontradas, são elas:

- Apesar da melhora de capacidade de processamento e armazenamento de dados dos dispositivos móveis, ainda enfrentamos problemas como o tamanho limite das aplicações J2ME;
- Entrada de dados provenientes de usuários em aplicações móveis é sacrificada, uma vez que em geral é feita através do teclado de um celular;
- Dificuldade em manter uma configuração compatível com os diversos modelos de dispositivos reais, devido à divergência de configuração nos aparelhos.

Como esse artigo tinha como um de seus objetivos servir de referência para novos desenvolvedores e novas aplicações que viessem a fazer uso da tecnologia J2ME, ele poderá de maneira geral servir de bibliografia para futuros trabalhos nessa área de conhecimento, ou também poderá servir de documento referência para implementações de aplicações que irão fazer uso dos benefícios dessa tecnologia nas diversas áreas de conhecimento.

BIBLIOGRAFIA

BOUZAN, B. P.; FONSECA, E. D. **WAP - Wireless Application Protocol**. Universidade Federal do Rio de Janeiro, Rio de Janeiro. Publicado em 9 de Junho de 2006. Disponível em: <http://www.gta.ufrj.br/grad/06_1/wap/index.htm>. Acesso em: 05 fev. 2008.

CARNIEL, J.; TEIXEIRA, C. **Apostila de J2ME- versão 1.0**. 2003. 45p.

MAHMOUND, Q. H. **Secure Java MIDP Programming Using HTTPS with MIDP**. 2002. Disponível em: <http://developers.sun.com/mobility/midp/articles/https/ >. Acesso em: 20 out. 2007.

MUCHOW, J. **Core J2ME – Tecnologia & MIDP.** São Paulo: Pearson Makron Books, 2004. p.

RABELLO, R. R.; TRECCANI, P. J.; JOHNSON, T. M. **Integrando a Tecnologia J2ME no Âmbito Acadêmico**. Universidade da Amazônia, Belém, Pará. 10p.

SILVA, M. S. **Introdução a tecnologia WAP**. Publicado em: 10/05/2004. Disponível em: <http://www.linhadecodigo.com.br/Artigo.aspx?id=319>. Acesso em 05 fev. 2008.

SUN MICROSYSTEMS. **Connected Limited Device Configuration (CLDC); JSR 30, JSR 139 Overview**. 2002. Disponível em: <http://java.sun.com/products/cldc/overview.html>. Acesso em: 10 out. 2007.

SUN MICROSYSTEMS. **Mobile Information Device Profile (MIDP); JSR 37, JSR 118**. 2002. Disponível em: <http://java.sun.com/products/midp/>. Acesso em: 10 out. 2007.