**UNIVERSIDADE FEDERAL DE SANTA MARIA**
**CENTRO DE TECNOLOGIA**
**CURSO DE ESPECIALIZAÇÃO EM SISTEMAS DE COMPUTAÇÃO PARA WEB**

# UMA APLICAÇÃO DA COMPUTAÇÃO MÓVEL PARA PESQUISA DE DÉBITOS NA PREFEITURA DE ALEGRETE

**MONOGRAFIA DE PÓS-GRADUAÇÃO**

**CRISTIAN TALES FAGUNDES BANDEIRA**

**SANTA MARIA, 11 DE DEZEMBRO DE 2006.**

# UMA APLICAÇÃO DA COMPUTAÇÃO MÓVEL PARA PESQUISA DE DÉBITOS NA PREFEITURA DE ALEGRETE

**por**

**Cristian Tales Fagundes Bandeira**

Monografia apresentada ao Curso de Especialização em Sistemas de Computação para Web, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
**Especialista em Sistema de Computação**.

**Orientador: Profa. IARA AUGUSTIN**

**Santa Maria, RS, Brasil**
**2006**

**Universidade Federal de Santa Maria**
**Centro de Tecnologia**
**Curso de Especialização em Sistemas de Computação para Web**

A Comissão Examinadora, abaixo assinada,
aprova a Especialização

**UMA APLICAÇÃO DA COMPUTAÇÃO MÓVEL PARA PESQUISA DE DÉBITOS NA PREFEITURA DE ALEGRETE**

elaborada por
**Cristian Tales Fagundes Bandeira**

como requisito parcial para obtenção do grau de
**Especialização em Sistemas de Computação para Web**

**COMISÃO EXAMINADORA:**

**Iara Augustin, Drª.**
(Presidente/Orientador)

**Raul Ceretta Nunes, Dr.**

**João Carlos Damasceno Lima**, **Ms.**

Santa Maria, 11 de Dezembro de 2006.

# SUMÁRIO

INTRODUÇÃO ..... 7

1. TECNOLOGIA PARA O DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS ..... 10

1.1 Requisitos das Aplicações Móveis ..... 10

1.2 A Plataforma J2ME ..... 12

1.2.1 Entendento J2ME ..... 13

1.2.2 Detalhes do MIDP ..... 14

1.2.2.1 MIDP 2 .0 ..... 16

1.2.3 MIDlets ..... 16

1.2.3.1 Ciclo de vida da MIDlet ..... 17

1.2.3.2 Componentes visuais – Interface gráfica ..... 19

1.3 Ambiente de Desenvolvimento ..... 20

1.3.1 Como usar o *wireless toolkit* ..... 21

2. Estudo de caso: A Aplicação PESQUISA DE DÉBITOS NA PREFEITURA DE ALEGRETE ..... 25

2.1 Descrição e requisitos ..... 26

2.1.1 Recursos de Hardware ..... 26

2.1.2 Recursos de Software ..... 27

2.2 Modelagem ..... 27

2.2.1 Módulo PHP ..... 27

2.2.2 Módulo Aplicação Móvel ..... 31

2.3 Implementação ..... 33

2.3.1 Aplicação Prefeitura ..... 33

2.3.2 O Processo de Desenvolvimento ..... 34

2.3.3 Estrutura da Aplicação Prefeitura ..... 34

2.3.3.1 Classe `Prefeitura` ..... 35
2.3.3.2 Classe `Community` ..... 35
2.3.3.3 Classe `ButtonListView` ..... 36
2.3.3.4 Classe `Dialog` ..... 37
2.3.3.5 Classe `LoginView` ..... 38
2.3.3.6 Classe `GetHTTP` ..... 39
2.3.4 Resultados da Aplicação Prefeitura ..... 41
2.3.4.1 Tela de Resultados ..... 42
3. CONCLUSÕES E TRABALHOS FUTUROS ..... 43

REFERÊNCIAS BIBLIOGRÁFICAS ..... 44
APÊNDICES ..... 45

# LISTA DE FIGURAS

FIGURA 1.1 – Plataforma Java 2 Micro Edition (REIS,2006). ..... 11
FIGURA 1.2 – Ciclo de Vida da MIDlet. ..... 19
FIGURA 1.3 - Componentes Visuais. ..... 20
FIGURA 1.4 – Apresentação do KToolBar. ..... 21
FIGURA 1.5 – Formulário com o Nome Do Projeto. ..... 22
FIGURA 1.6 – Formulário de Configuração do Projeto. ..... 23
FIGURA 1.7 – Formulário de Diretório do Projeto. ..... 23
FIGURA 1.8 – Emulador Genério Toolkit. ..... 25
FIGURA 2.1 – Caso de Uso do Pedido de Login e Senha. ..... 28
FIGURA 2.2 – Diagrama de Seqüência do Pedido de Login e Senha. ..... 29
FIGURA 2.3 – Pedido de Login e Senha na página da Prefeitura. ..... 29
FIGURA 2.4 – Tributos Lançado para o Contribuinte. ..... 30
FIGURA 2.5 – Tributos Lançado para o Contribuinte. ..... 31
FIGURA 2.6 – Valor do Débito. ..... 31
FIGURA 2.7 – Caso de Uso Consulta Débitos. ..... 32
FIGURA 2.8 – Diagrama de Seqüência Consulta Débitos. ..... 32
FIGURA 2.9 – Classe `Prefeitura.` ..... 35
FIGURA 2.10 – Classe `Community`. ..... 36
FIGURA 2.11 – Classe `ButtonListView`. ..... 37
FIGURA 2.12 – Classe `Dialog`. ..... 38
FIGURA 2.13 – Classe `LoginView`. ..... 39
FIGURA 2.14 – Aplicação `cel001.php` no servidor da prefeitura. ..... 40
FIGURA 2.15 – Classe `GetHTTP`. ..... 41
FIGURA 2.16 – Telas de Abertura da Aplicação. ..... 42
FIGURA 2.17 – Resultado da Consulta de Débitos. ..... 42

# APÊNDICES

APÊNDICE A – Classe Prefeitura ..... 45
APÊNDICE B – Classe Community ..... 46
APÊNDICE C – Classe ButtonListView ..... 56
APÊNDICE D – Classe `Dialog` ..... 58
APÊNDICE E – Classe `LoginView` ..... 61
APÊNDICE F – Classe `GetHTTP` ..... 66

# RESUMO

Monografia de Especialização
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

## UMA APLICAÇÃO DA COMPUTAÇÃO MÓVEL PARA PESQUISA DE DÉBITOS NA PREFEITURA DE ALEGRETE

AUTOR: CRISTIAN TALES FAGUNDES BANDEIRA
ORIENTADORA: IARA AUGUSTIN
Data e Local da Defesa: Santa Maria, 11 de dezembro de 2006.

Este trabalho apresenta um estudo sobre a tecnologia Java para o desenvolvimento de aplicações em dispositivos móveis, mas especificamente sobre a plataforma J2ME. O estudo faz uma interação do J2ME com aplicativos já existentes e desenvolvidos na linguagem de programação PHP. O trabalho está dividido em quatro capítulos tratando respectivamente de: introdução (Evolução das tecnologias e as novas tecnologias); tecnologia para o desenvolvimento de aplicações móveis (Plataforma Java); estudo de caso (A aplicação Pesquisa de Débitos na Prefeitura de Alegrete); conclusões e trabalhos futuros.



Palavras-chave: estudo da tecnologia; desenvolvimento; aplicação

## ABSTRACT

Monografh of Especialization
Pos-Graduation Program in Computer Science
Federal University of Santa Maria

### AN APPLICATION OF THE MOBILE COMPUTATION FOR RESEARCH OF DEBITS IN THE ALEGRETE CITY HALL

AUTHOR: CRISTIAN TALES FAGUNDES BANDEIRA
ADVISOR: IARA AUGUSTIN
Date and Place: Santa Maria, December 11th, 2006.

This work presents a study on the Java technology for the development of applications in mobile devices, but specifically on platform J2ME. The study it makes an interaction of the applicatory already existing and developed J2ME with in the programming language PHP. The work is divided in four chapters treating respectively to: introduction (Evolution of the technologies and the new technologies); technology for the development of mobile applications (Java Platform); case study (the application Research of Debits in the City hall of Alegrete); future conclusions and works.



Key word: study of the technology; development; application

# INTRODUÇÃO

No mundo atual, a necessidade de integrar comunicação e mobilidade se torna cada vez maior. Com a quebra da fronteira entre os mercados, as empresas buscam soluções que viabilizem seus negócios, tanto no nível gerencial como no operacional, dentro desta nova realidade que impõe a realização de comunicações e operações de acesso a dados a qualquer hora, a partir de qualquer lugar do planeta.

> Mais do que a própria Internet, a tecnologia móvel forma a base da principal revolução tecnológica do século XXI. Ao permitir ao indivíduo se comunicar a qualquer momento e em qualquer lugar, a mobilidade muda a forma dos seres humanos interagirem, afetando suas relações sociais, familiares, afetivas e profissionais. (PROMON, 2005).

A globalização do mercado tem provocado um crescente aumento da competitividade no ambiente corporativo. Como reflexo disso, o tempo de maturação de novos negócios e de lançamento de novos produtos se reduz cada vez mais, forçando ciclos de tomada de decisão cada vez mais curtos.

> O mundo dos negócios cada vez mais on-line. Dados da FGV pela Folha de São Paulo no último mês de abril indicam que até março de 2004 as transações comerciais *on-line* realizadas entre empresas eram mais de 9% do total do mercado. Dois anos depois, esse tipo de transação mais do que dobrou e já responde por 19,6%(BRASIL TELECOM, 2006).

Por outro lado, a evolução tecnológica (notadamente o crescimento exponencial da capacidade de processamento dos circuitos integrados e o aumento da disponibilidade de banda propiciado pelas modernas redes de telecomunicações) tem sido o catalisador do enriquecimento da comunicação eletrônica, num esforço de torná-la cada vez mais semelhante à interação pessoal. O melhor exemplo desse enriquecimento é a crescente presença do vídeo nos meios de comunicação pessoais: videoconferência em telefones celulares, *instant messengers* e telefones fixos estão entre as formas mais comuns de comunicação multimídia.

Entre todas as aplicações e tecnologias móveis, a telefonia celular é, sem dúvida, a mais bem-sucedida, popular e madura. Como toda tecnologia madura, passou por uma série de estágios evolutivos, partindo de sua proposta inicial de oferecer ao usuário móvel um serviço similar ao da telefonia tradicional, até atingir o *status* atual de tecnologia convergente de alta *performance.*

Além do celular, existem diversos equipamentos que nos permitem ganhar mobilidade, como os handhelds, PDAs(*Personal Digital Assistants* ), Palmtops e que nos é possível fazer

uma associação, mobilidade/portabilidade.

A mobilidade é fundamental e seu potencial é amplo em sistemas de informação. O comércio eletrônico tradicional, já sinaliza claramente que a vantagem da conveniência é bem-aceita. Pode-se fazer compras pela Web em supermercados, sem precisar enfrentar trânsito ou perder tempo em deslocamentos. Se essa conveniência puder ser disponibilizada sem a necessidade de se dirigir a um ponto fixo onde esteja instalado um microcomputador, mas oferecida por um equipamento móvel como um telefone celular, com certeza ter-se-á um conforto muito maior no uso dos serviços.

Aliado a essa demanda, vê-se a evolução tecnológica. Ao lado dos aparelhos móveis, crescentemente sofisticados, e tecnologias de comunicação cada vez mais potentes em termos de capacidade de transmissão, permitindo tráfego de imagens e vídeos de alta resolução, tem-se tecnologia como o ambiente Java, cada vez mais presente no dia-a-dia, seja em produtos fixos ou móveis e que apresentam entre outras características, alguma forma de comunicação e elevado grau de sofisticação em relação às tarefas que executam.

A plataforma Java é a escolha indicada para o desenvolvimento sem fio por muitas razões. Uma delas é a segurança. O código Java executa sempre dentro dos limites da Java Virtual Machine (JVM), o qual oferece um ambiente seguro para executar o código instalado. Outra é o encorajamento à programação robusta com os mecanismos da linguagem para o tratamento de exceções e o *garbage collector* que facilita a programação com o controle automático de objetos não mais referenciados na memória. Por outro lado, a portabilidade é a razão mais preponderante. Um programa simples pode ser executado em vários dispositivos diferentes. Um segundo benefício da portabilidade é a tranqüilidade na liberação de aplicações para um dispositivo através de uma rede sem fio, pelo fato do código Java rodar na JVM, ele pode ser baixado para ser executado seguramente.

O desenvolvimento de aplicações para dispositivos móveis utilizando a linguagem de programação Java é possível utilizando o ambiente Java 2 Plataform Micro Edition (J2ME).

> J2ME é a adição a mais nova e a menor à família de Java. É o irmão menor de J2SE (edição padrão) e do J2EE (edição da empresa). Como mencionado, J2ME fornece um ambiente do desenvolvimento para uma escala de dispositivos pequenos, confinados. Mesmo que J2ME fosse alvejado em dispositivos com potencialidades limitadas, foi derivado de J2SE e mostra todas as características da língua de Java (JODE, 2004).

A plataforma J2ME oferece uma máquina virtual Java, chamada de KVM, compacta o suficiente para ser suportada pelas restrições de memória destes dispositivos. Porém, essa máquina não suporta todas as classes e funcionalidades do Java, sendo formado basicamente

por duas outras especificações, a CLDC (*Connected Limited Device Configuration*), e MIDP (*Mobile Information Device Profile*).

A configuração CLDC fornece um conjunto de APIs para as aplicações sem fios. Essa especificação fornece as classes responsáveis pela conexão, entrada e saída de dados, classes de manipulação de strings e operações matemáticas.

O perfil MIDP define melhor a configuração CLDC para os aparelhos celulares e PDAs com pouca capacidade de memória e processamento. Oferece a biblioteca de interfaces gráficas. Provê ainda as classes para memória persistente e algumas definições de objetos de formulário.

O uso dessas tecnologias motivou o desenvolvimento deste trabalho, cujo objetivo geral é a adaptação de uma aplicação existente na prefeitura de Alegrete para pesquisa de débitos via aparelho móvel.

O texto possui a seguinte estrutura: o capítulo 1 apresenta os principais conceitos das tecnologias envolvidas neste trabalho, o capítulo 2 apresenta um estudo de caso de uma pesquisa de débitos na Prefeitura de Alegrete, o capítulo 3 apresenta as considerações finais e conclusões.

# 1. TECNOLOGIA PARA O DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

Neste capítulo são apresentadas algumas tecnologias utilizadas para o desenvolvimento de sistemas para os dispositivos móveis, em particular a tecnologia Java, na plataforma J2ME, que vem sendo a plataforma mais utilizada, principalmente pela telefonia celular, que cria assim oportunidades de expansão para seus serviços através de inovações e meios de entretenimento, como os jogos, além do desenvolvimento de soluções corporativas.

## 1.1 Requisitos das Aplicações Móveis

Java atualmente encontra-se na sua segunda edição, sendo que a versão mais recente é a 5.0, também conhecida como J2SE. O termo segunda edição refere-se ao fato que quando a *Sun* lançou a versão 1.2 as modificações eram tão grandes que resolveu mudar o nome para Java 2. Com isso houve também uma reestruturação dos pacotes que agora passaram a contar com o prefixo J2, seguido do tipo da versão (como em J2ME, por exemplo).

A plataforma Java conta com 3 diferentes tipos de versões, ilustradas na Figura 1.1, sendo eles:

- *Java 2 Standard Edition* (J2SE): É a versão básica, destinada ao desenvolvimento da maior parte das aplicações de *desktop* e estações de trabalho;
- *Java 2 Enterprise Edition* (J2EE): Versão destinada ao desenvolvimento de aplicações de grande porte, as quais fazem extenso uso de EJB ( *Enterprise Java Beans* ) e Servidores de Aplicação, por exemplo. A implementação padrão chama-se J2SDKEE (*Java 2 Software Development Kit Enterprise Edition* );
- *Java 2 Micro Edition* (J2ME): Destinada ao desenvolvimento de programas para periféricos móveis ou de pequeno porte.


FIGURA 1.1 – Plataforma Java 2 Micro Edition (REIS,2006).

Outras definições que são importantes:

- *Java Virtual Machine* (JVM): é o mecanismo que interpreta os arquivos *.class* (pré-compilados), fazendo o programa funcionar em qualquer plataforma de hardware. A JVM pode ser considerada o coração do Java, ela tem este nome pois faz o processamento. A JVM não apenas interpreta o código, às vezes, compila alguns trechos deste para acelerar a execução (*just in time compiler*);
- *Java Software Development Kit* (JSDK): ou simplesmente JDK, é o ambiente de desenvolvimento Java. Entre outras coisas, inclue: *Java Virtual Machine*, compilador, *appletviewer* par executar *applets*, códigos de exemplo e bibliotecas básicas para auxiliar na programação;
- *Java Runtime Environment* (JRE): usado no caso de apenas executar as aplicações Java, sendo que o pacote contém somente as bibliotecas necessárias para tal e a JVM;
- *Java Community Process* (JCP): é uma organização internacional de desenvolvedores e fabricantes cuja missão é desenvolver e revisar as especificações, referências e outras implementações relacionadas ao Java. Para maiores informações, a página oficial é http://www.jcp.org;
- *Java Specification Request* (JSR): são as propostas de mudanças e novas implementações para o Java. Em outras palavras, são os "documentos" que

definem tal funcionalidade, ou propõem mudanças em algo existente, por exemplo;

- *Web Services Developers Pack* (WSDP): conjunto de interfaces de programação (APIs) e ferramentas para o desenvolvimento de *WebServices* e *servlets* em Java;
- *Java Server Pages* (JSP): tecnologia baseada em Java para o desenvolvimento de aplicações Web. É uma linguagem de *script* parecida com *Personal Home Page* (PHP), com a diferença de usar Java e ser traduzida para puro Java no momento da execução;
- *Java DataBase Connectivity* (JDBC): especificação que deve ser seguida para oferecer suporte ao acesso a Banco de Dados em Java.

Tendo em vista que a máquina Java tende a ser incorporada em aparelhos das redes 2.5G, a tecnologia J2ME torna-se a principal ferramenta de desenvolvimento para aplicativos web móvel e muitos benefícios. Altos investimentos de fabricantes, como a Motorola, Nokia, Siemens e operadoras telefônicas, como a Nextel, NTTDoCoMo, tornando essa tecnologia a grande promessa para oportunidades emergentes.

## 1.2 A Plataforma J2ME

Segundo Muchow (2005), a plataforma *Java Micro Edition*, ou J2ME como é mais conhecida, é o segmento do Java destinado aos dispositivos de pequeno porte, tais como telefones celulares, PDAs(*Personal Data Assistants* - Assistentes de dados pessoais) , *Internet screenphones*, televisores digitais de nova geração, sistemas de navegação automotiva, comutadores e roteadores de rede, componentes para automação residencial, etc.

Com a tecnologia J2ME é possível criar aplicações especiais para tirar proveito das vantagens e particularidades dos aparelhos móveis, aplicações integradas ao serviço das operadoras, que tenham a interatividade esperada para um pequeno aparelho e que respondam de acordo com a situação e a atenção do usuário.

J2ME é, um conjunto de especificações, que define uma JVM (Máquina Virtual Java) simplificada, um conjunto de APIs (Interfaces para Programação de Aplicações) especializadas para pequenos dispositivos e ferramentas direcionadas aos dispositivos com poder de processamento, memória e conectividade menor que um computador *desktop*, com

capacidade de executar as aplicações localmente.

Possui uma série de vantagens em relação às demais tecnologias sem fio (*wireless)*: a portabilidade, ou seja, a capacidade de desenvolver aplicações que executem em qualquer dispositivo de qualquer fabricante, independente de tecnologia de operadora ou fabricante; os recursos gerais de Java podem ser utilizados; programação orientada a objetos; suporte à manipulação da tela gráfica e teclado dos dispositivos; maior suporte à conectividade com outros dispositivos, e programação em rede com o protocolo HTTP (*Hipper Text Transfer Protocol).*

Diante dessas vantagens e por ser esta tecnologia uma das principais ferramentas de desenvolvimento de aplicações para telefones celulares, e conseqüentemente, o objeto de estudo desta monografia, a tecnologia J2ME é discutida na seqüência em maiores detalhes.

### 1.2.1 Entendento J2ME

J2ME está dividida em configurações *(configurations*) e perfis *(profiles*) e, ainda, em APIs opcionais, que o fabricante coloca ou não, como multimídia e mensagens SMS. O desenvolvimento de configurações e perfis acontece de acordo com processos estabelecidos pela Comunidade Java, podendo participar o setor privado, público, ou mesmo qualquer indivíduo.

De acordo com Miranda (2003), uma configuração é uma JVM e API que contém os requisitos mínimos de funcionamento (a API pode ser um subconjunto de J2SE), destinada a uma larga faixa de dispositivos, com capacidade de processamento e memória semelhantes.

Duas configurações estão definidas hoje, a CDC (*Connected Device Configuration*) e a CLDC (*Connected Limited Device Configuration*). A configuração CDC destina-se a ambientes e dispositivos com mais capacidade, como TVs a cabo, sistemas automotivos, alguns PDAs mais poderosos, etc. A configuração CLDC objetiva aparelhos móveis, *smartphones*, *pagers*, PDAs, e outros dispositivos menores. Para aparelhos celulares a CLDC tem algumas características como limites de processamento e memória, tem entre 160 Kb e 512 Kb disponíveis para Java, conexão limitada intermitente e lenta (mais ou menor 9600 bps), tela de tamanho pequeno, fonte de energia limitada, um software que inclua suporte a um subconjunto da linguagem Java e a um subconjunto da máquina virtual Java, definindo algumas funções para permitir o desenvolvimento de aplicações móveis.

A versão suportada pela maioria dos dispositivos celulares no Brasil é a CLDC 1.0 e o MIDP 1.0, as versões mais recentes são CLDC 1.1 e MIDP 2.0.

A CLDC está baseada no chamado "*Sandbox Security Model*", isto é, a máquina virtual terá um espaço de memória independente do restante das aplicações do aparelho celular (tais como agenda, tons, imagens, configuração, *browser* WAP, etc). Nesse modelo de segurança as operações que podem ser executadas estão restritas ao conjunto de funções da API. Nenhuma outra operação é permitida, isto quer dizer que a aplicação não consegue acessar a área de memória do calendário ou agenda de contatos do aparelho celular, por exemplo.

Mesmo com a divisão do J2ME entre as configurações, a variedade entre dispositivos ainda é muito grande. Para resolver isso a *Sun* criou extensões das configurações chamadas perfis. Um perfil define as características da linguagem para um tipo particular de dispositivo. Logo, os perfis provêem uma série de API´s padrões que combinados com uma configuração provêem um serviço completo para que as aplicações possam ser executadas (WEBMOBILE, 2005).

No momento alguns dos perfis existentes são: MIDP, RMI Profile, Foundation Profile, PDA Profile, PersonalJava (WEBMOBILE, 2005).

O CLDC 1.1 define em sua API os pacotes java.io, java.lang, java.util e javax.microedition.io (conexão e interfaces). Não há suporte a ponto flutuante. O MIDP define o que há no CLDC mais os pacotes `javax.microedition.lcdui` (interface com o usuário), `javax.microedition.rms` (sistema de gerência de registros para persistência de informações), `javax.microedition.midlet` (suporte para aplicações MIDP, os chamados *midlets*).

A plataforma J2ME atual é formada pelas configurações:

- CLDC 1.0 e 1.1
- MIDP 1.0.3 e J2ME Toolkit 1.0.4 (maioria dos celulares brasileiros)
- MIDP 2.0 e J2ME Toolkit 2.0
- CDC 1.0
- Foundation Profile 1.0
- Personal Basis Profile 1.0

### 1.2.2 Detalhes do MIDP

De acordo com MUCHOW(2005), o MIDP permite a execução de aplicações múltiplas de MIDP, conhecidas como MIDlets. O modelo definido como o MIDlet é empacotado. O modelo define também como vários MIDlets podem ser empacotado junto nos *suites* e o

compartilhamento dos recursos entre eles, tais como os gráficos e os dados armazenados na facilidade de base de dados pequena, conhecida como o *Record Management System*(RMS).

Cada *suite* de MIDlet tem também uma descrição chamada *Java Aplication Descriptor* (JAD*)*, que permite que o software gerencie a aplicação no dispositivo e obtenha informações para a instalação. O modelo define também um ciclo de vida para um MIDlet, que permita iniciar, parar e continuar, terminar um MIDlet.

Os aparelhos que suportam o MIDP trabalham basicamente com dois tipos de arquivos, os com extensão JAD e os de extensão JAR (*Java Archive*). O arquivo `.jad` é um arquivo de descrição do aplicativo e dentro dele há algumas informações referentes ao fabricante, versão, *link* para o arquivo `.jar`, entre outras. Já o arquivo `.jar` é o aplicativo propriamente dito, todas as classes compiladas estão lá dentro, bem como as imagens usadas nas aplicações. O arquivo `.jad` é necessário apenas para fazer o *download* do aplicativo direto pelo celular via web.

Vários fabricantes adicionam ao MIDP algumas outras APIs, tais como suporte a uma forma simplificada de AWT (*Abstract Window Toolkit*), controles de vibração, som e display.

Como cada fabricante adiciona livremente o que considerar importante, isso dificulta a portabilidade da aplicação entre os vários modelos de aparelhos e fabricantes, a menos que se use sempre o conjunto mínimo especificado no MIDP 1.0, para garantir o "*Write Once, Run Anywhere*" (WORA).

Conforme WEBMOBILE (2005), o MIDP exige dispositivos com os seguintes requisitos mínimos:

- Mínimo de 160kb de memória não-volátil para Java
- Um processador de 16bits ou 32 bits com um clock de no mínimo 16MHz
- 32kb de memória volátil para tempo de execução
- Pelo menos 192kb livres para Java
- Uma tela de pelo menos 96x54 pixels
- Capacidade de entrada de dados seja por teclado (do celular), teclado externo ou mesmo touch-screen
- Possibilidade de enviar e receber dados em conexão possivelmente intermitente e banda reduzida.

O MIDP 2.0 suporta um conjunto maior de especificações que inclui controles de vibração, som e display entre outras capacidades, garantindo maior portabilidade.

#### 1.2.2.1 MIDP 2 .0

> O padrão MIDP 2.0 *New Generation* oferece novos recursos como aumento de performance, melhorias nas APIs para interfaces gráficas , maior segurança e APIs para jogos (funcionalidades 2D) e conectividade, tornando possível o desenvolvimento e a utilização de uma maior variedade de aplicações (LIMA, 2005, p.47).

A segurança neste padrão está mais confiável. Através da assinatura criptográfica a *Suite* de MIDlets a ser baixada para o telefone fica mais confiável, validando a origem dos MIDlets, podendo ser assinados e tratados semelhante à J2SE.

O conceito de *sandbox* também foi melhorado. As aplicações confiáveis *(trusted)* tem acesso a recursos específicos, enquanto as não-confiáveis *(untrusted)* precisam de confirmação do usuário para acessar os protocolos HTTP e HTTPS.

Quanto à rede, além do HTTP há suporte a cinco protocolos: HTTPS, Comunicação Serial, Sockets, Server Socketse Datagramas, através do *Generic Connection Framework.* Também suporta a tecnologia *Push*: MIDlets podem registrar um dispositivo para receber eventos (conexões de rede, mensagens enviadas), mesmo quando o dispositivo não esteja em modo de execução de aplicações Java.

É requerida a implementação do OTA *Recommended Practice* (*Over The Air*), padronizando o *download* e a instalação de MIDlets através do *browser* embutido. A padronização inclui notificações para o servidor quando as aplicações são instaladas, gerenciamento de atualizações de MIDlets e o suporte a sessões através de URL-Rewriting.

### 1.2.3 MIDlets

Segundo Jode (2004), uma vez que uma MIDlet foi inicializada, ela reside em um de três estados possíveis. Um estado é projetado assegurar-se de que o comportamento de uma aplicação seja consistente com as expectativas dos usuários finais e do fabricante do dispositivo. A iniciação da aplicação deve ser curta; deve ser possível pôr uma aplicação em um estado *non-active*; e deve também ser possível destruir uma aplicação a qualquer hora.

Existem três estados válidos da MIDlet:

1. *PAUSED* (Pausado)

   O MIDlet foi inicializado, mas está em um estado de pausa. Este estado é incorporado a uma de quatro maneiras:

   - depois que o MIDlet foi instanciado pelo software da aplicação que invoca

seu construtor; se uma exceção ocorrer, o estado DESTRUÍDO está incorporado;

- no estado *ACTIVE* (Ativo), se o método do `pauseApp()` for chamado pelo software da aplicação;
- no estado *ACTIVE* (Ativo), se o método do `startApp()` for chamado mas uma exceção foi levantada;
- no estado *ACTIVE* (Ativo), se o método do `notifyPaused()` for invocado e retornado com sucesso.

Quando uma MIDlet bem-escrito é pausado, deve geralmente liberar todos os recursos compartilhados.

2. *ACTIVE* (Ativo)

   O MIDlet está funcionando normalmente. Este estado é incorporado depois que o software da aplicação chamou o método do `startApp()`. O método do `startApp()` pode ser chamado mais de uma ocasião durante o ciclo de vida da MIDlet.

3. *DESTROYED* (Destruído)

   A MIDlet liberou todos os recursos e terminou-os. Este estado, que pode somente ser incorporado uma vez, é incorporado para as seguintes duas razões:

   - o método incondicional `destroyApp (boolean)` foi chamado pelo software da aplicação e retornado com sucesso; se o argumento incondicional for falso um `MIDletStateChangedException` puder ser jogado e a MIDlet não se moverá para o estado DESTRUÍDO; a execução do método do `destroyApp()` deve liberar todos os recursos e terminar todas as `threads`.
   - quando o método `notifyDestroyed()` retornar com sucesso; a aplicação deve liberar todos os recursos e terminar todas as `threads` antes de chamar o método `notifyDestroyed()`.

1.2.3.1 Ciclo de vida da MIDlet

Uma MIDlet não deve possuir um método `public void static main()`. O software de gerenciamento de aplicação deve suprir a classe inicial necessária pelo CLDC para iniciar a MIDlet. Quando uma MIDlet é instalada, ela é mantida no aparelho e fica pronta

para uso. Figura 1.2.

Quando a MIDlet é executada, uma instância é criada através de seu construtor público sem argumentos, e seus métodos são chamados para mudar pelos estados da MIDlet. Quando ela é terminada, ou destruída, os recursos utilizados por ela podem ser recuperados, incluindo os objetos criados e suas classes.

O software de gerenciamento de aplicação no aparelho (midp.exe) disponibiliza um ambiente no qual a MIDlet é instalado, iniciado, parado e desinstalado. Também é responsável por manusear os erros que podem ocorrer durante alguma destas etapas.

O compartilhamento de dados e outras informações entre MIDlets são controlados pelas APIs individuais e suas implementações. Assim, por exemplo, os métodos da API de um sistema de gerenciamento de registros devem ser especificados para manusear com dados que podem ser compartilhados com outras MIDlets.

O Software de Gerenciamento de Aplicação pode administrar as atividades de múltiplos MIDlets dentro de um ambiente em tempo de execução (*runtime*). Além disso, a MIDlet pode iniciar sozinha algumas mudanças de estados e notificar para o software de gerenciamento de aplicação que estas mudanças ocorreram.

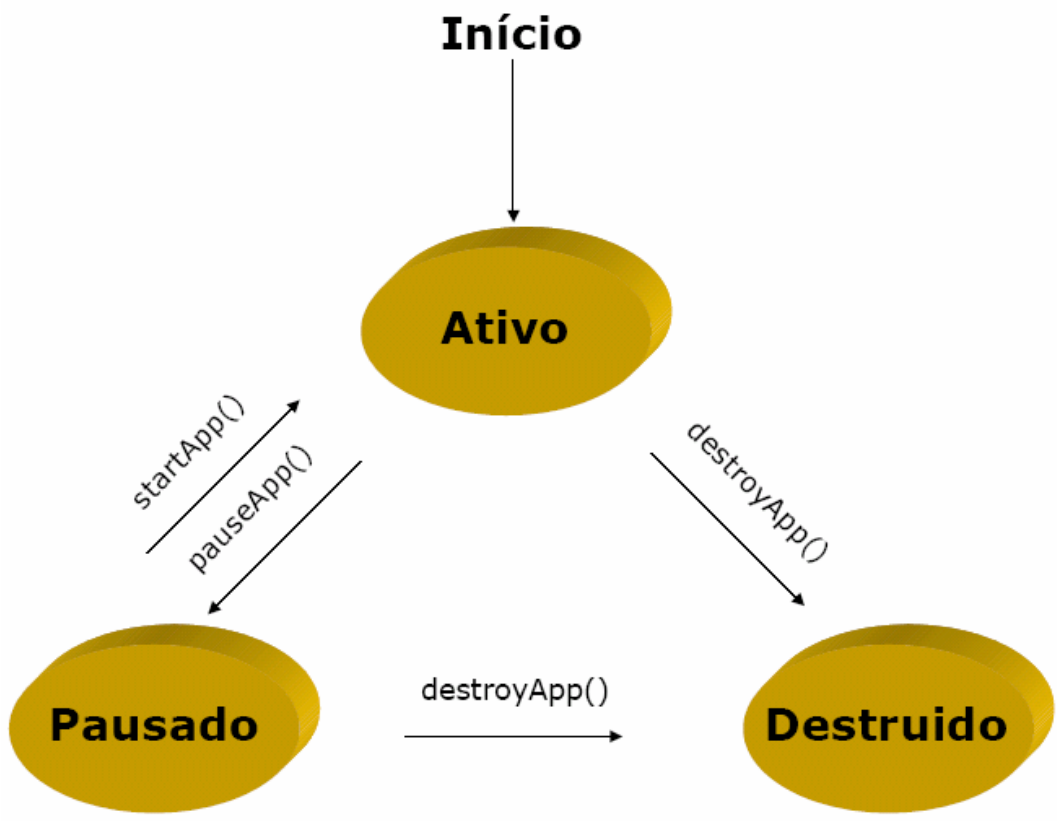

FIGURA 1.2 – Ciclo de Vida da MIDlet.

1.2.3.2 Componentes visuais – Interface gráfica

O pacote `javax.microedition.lcdui` contém os elementos (pré-fabricados) de interface gráfica utilizados por aplicativos MIDP. Há basicamente três famílias de componentes *Graphical User Interface* (GUI) definidos no pacote `lcdui`, que são agrupados baseados em suas classes base:

**- Componentes de tela :** são descendentes da classe abstrata "*Screen*"(tela) e provêem os componentes de interface gráfica (*widgets)* em forma de janela. O objeto "*Form*" (formulário) é um descendente da classe "*Screen*", que contém e forma controles de interface gráfica. Outros componentes de "*Screen*" incluem alertas (*Alert*), caixas de diálogo (*dialog box*), listas (*List*) e caixas de texto (*TextBox*), sendo esta última uma entrada de texto que suporta múltiplas linhas.

**- Componentes de item :** são controles tradicionais, como o campo de texto. Estes controles todos são descendentes da classe "Item". Esta classe provê uma API uniforme para colocar etiquetas de nome, tratamento de eventos, e exibição de controles. *ChoiceGroup, DateField, Gauge, ImageItem e StringItem* são outros componentes dessa classe.

**- Componentes diversos de display :** a grande maioria são descendentes da classe de hierarquia mais alta e abstrata "*Displayable*". Este grupo inclui componentes como a classe "*Command*", que integra os botões de comando; "*Ticker*", que integra as caixas de texto rolantes; "*Graphics*", que exibe os gráficos; e "*Choice*", que é uma interface de manipulação de seleções pré-definidas.

Toda a parte gráfica é gerenciada por um objeto `Display,` à partir do qual, cada aplicação tem acesso à uma única e privada instância. Essa pode ser obtida através do método estático `Display.getDisplay.` Além dos métodos para concentrar o foco da tela em um elemento em particular (*setCurrent*) e descobrir qual o elemento com o foco (*getCurrent*), a classe `Display` também expõe vários outros métodos muito úteis para obtenção de informação sobre as capacidades do display do dispositivo. Entre estas capacidades estão a detecção de suporte à cores (*isColor*) e quantas cores são suportadas (*numColors*).

Na figura 1.3 tem-se a composição do MIDP.

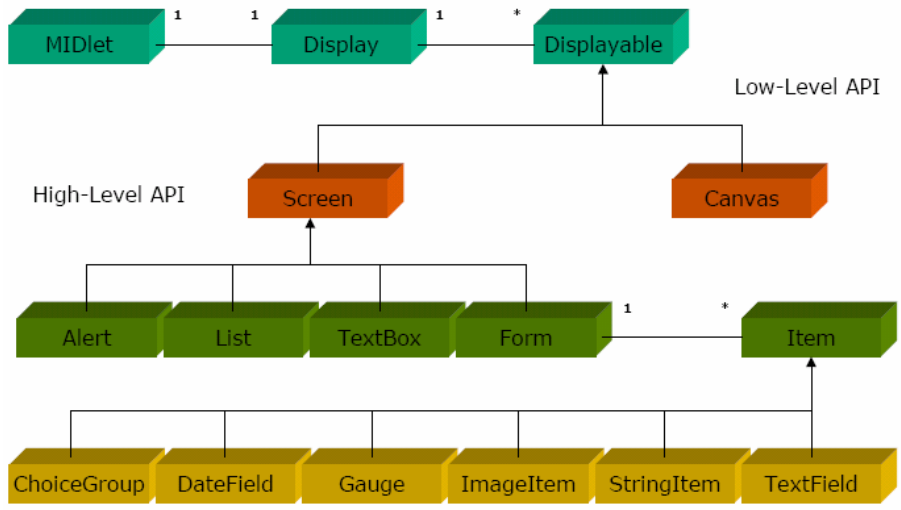


FIGURA 1.3 - Componentes Visuais.

### 1.3 Ambiente de Desenvolvimento

Para iniciar o desenvolvimento em J2ME, primeiramente é necessário montar o ambiente de desenvolvimento. No site da SUN *Microsystems* podem ser encontradas ferramentas que permitem gerar os arquivos necessários para rodar aplicações em aparelhos compatíveis com o J2ME gratuito.

Os elementos básicos para iniciar o desenvolvimento em J2ME, tanto no ambiente *Windows* quanto *Linux*, são:

- J2SE (kit de desenvolvimento Java);
- *Java Wireless Toolkit* (WTK) que é a plataforma de desenvolvimento Java para dispositivos móveis.

É importante ressaltar que o WTK não é um editor de código, ou seja, não se consegue escrever código Java através dele. Ele possibilita utilizar o IDE (*Integrated Development Environment*) de preferência para o desenvolvimento para dispositivos móveis. O JCreator pode ser utilizado como uma IDE para desenvolvimento ( *http://www.jcreator.com.).*

Com relação a *hardware*, o mínimo necessário para um bom funcionamento das aplicações utilizadas nesta abordagem e:

- 50 MB(*Mega Byte)* de espaço em seu HD (*Hard Disk*);
- 128 MB (*Mega Beyte*) de memória RAM (*Random Access Memory*);
- Um computador com no mínimo 800 MHz.

O primeiro passo é instalar o J2SE SDK (*Software Development Kit*) e JRE (*Java Runtime Environment*) da SUN, escolhendo o sistema operacional. Existem versões para *Windows, Linux e Solaris*. O *download* dos programas pode ser feito nos seguintes endereços:

- *http://java.sun.com/j2se/downloads.html*
- *http://java.sun.com/products/j2newtoolkit/download-2_2.html*

### 1.3.1 Como usar o *wireless toolkit*

No *linux* o executável encontra-se no diretório *bin* aonde foi intalado o WTK execute o seguinte comando "/usr/java/WTK2.2/bin/ktoolbar", já no *windows* vá até "Iniciar->Programas->J2ME Wireless Toolkit->Ktoolbar".

Após aberta a ferramenta *KtoolBar* temos as principais funcionalidades, conforme figura 1.4, definidas em botões, sendo eles:

- *New Project*: cria um novo projeto dentro do WTK;
- *Open Project*: abre um projeto já existente;
- *Settings*: abre uma tela para diversas configurações;
- *Build*: compila o projeto que está aberto no WTK;
- *Run*: roda o projeto com o emulador definido no combo (*Device*) abaixo dos botões;
- *Clear Console*: limpa a tela de saída.

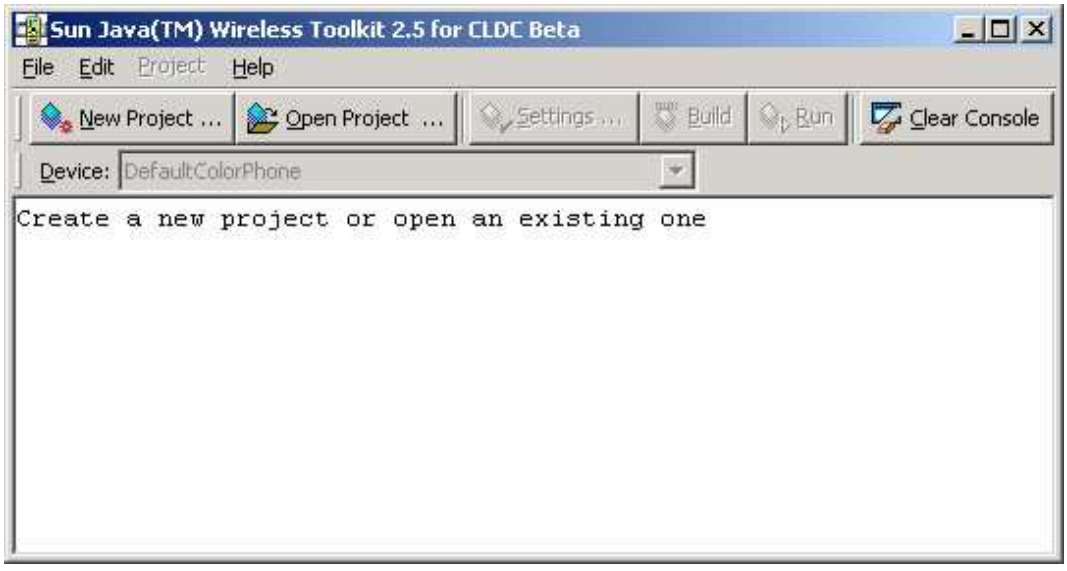


FIGURA 1.4 – Apresentação do KToolBar.

Nesta tela, um novo projeto deve ser aberto. Clicando no botão "New Project..." um formulário solicitando o nome do projeto e o da classe MIDlet que rodará o programa J2ME. Figura 1.5:

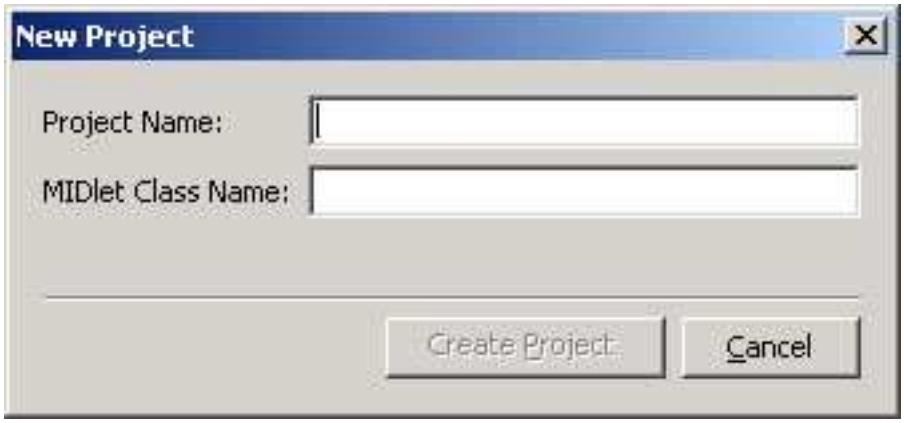


FIGURA 1.5 – Formulário com o Nome Do Projeto.

Ao preencher os nomes o botão "Create Project" se habilitará. Ao clicar nele o sistema automaticamente criará uma estrutura de diretórios referentes ao projeto dentro do diretório do ToolKit. Figura 1.6.

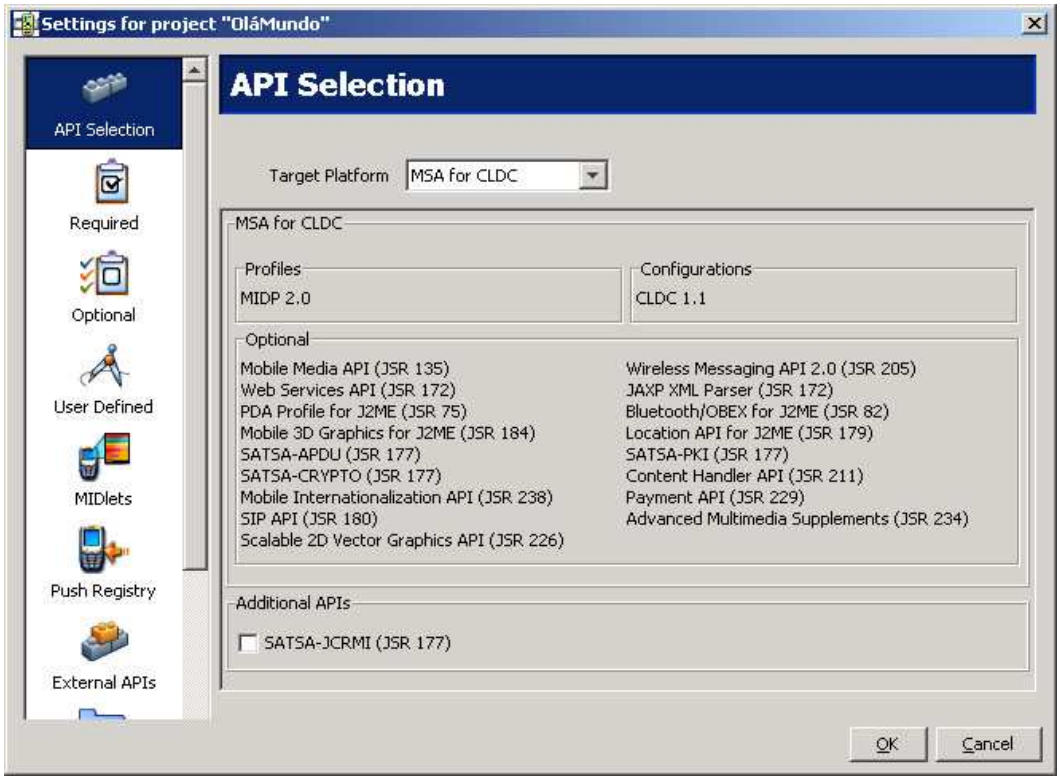


FIGURA 1.6 – Formulário de Configuração do Projeto.

Depois de clicar no botão "OK", o ToolKit indicará,que os diretórios foram criados. Figura 1.7.

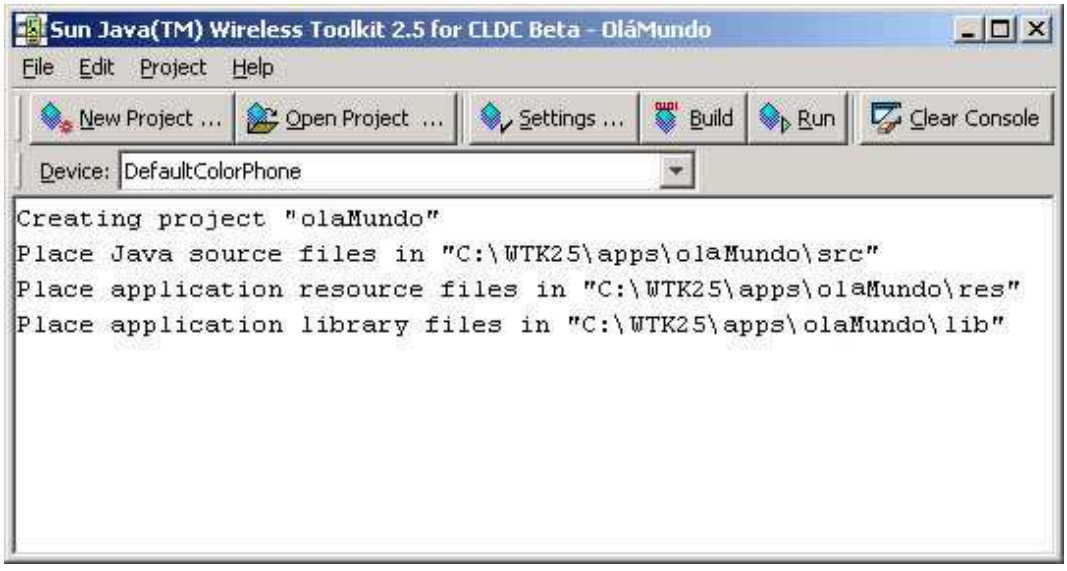


FIGURA 1.7 – Formulário de Diretório do Projeto.

Finalmente, copiar o arquivo fonte, no exemplo o olaJava, para o diretório "...\apps\olaMundo\src\". Agora basta clicar no botão "*Build*" que o *ToolKit* compilará os arquivos fonte e criará os *.class* que serão usados no simulador. Não ocorrendo nenhum problema a mensagem "*Build Complete*" aparecerá no final das mensagens, caso contrário, os erros serão apresentados.

Para executar a aplicação o botão "*Run*" deve ser pressionado. Uma nova janela com o simulador genérico aparecerá.

O botão em destaque, logo abaixo da palavra "*Launch*" e indicado pela seta, executará a aplicação. Figura 1.8.

Uma vez executado e aprovado, o arquivo .JAR com todas as classes, necessário para executar o programa no aparelho celular deve ser criado. Para cria-lo, deve-se selecionar "*Project Package*" no menu da *Ktoolbar*. A opção "*Create Obfuscated Package*" é aconselhada por questões de segurança.

Então é só enviar a aplicação para o celular. Pode se optar por vários métodos que devem ser escolhidos de acordo com as características do hardware que se tem. Alguns deles: por *Bluetooth*, por infravermelho, por *download* num site WAP, por email pop, por cabo, se o sistema operacional para *linux*, tem um programa chamado *OpenObex*, se Windows pode ser por infravermelho.


FIGURA 1.8 – Emulador Genério Toolkit.

## 2. Estudo de caso: A Aplicação PESQUISA DE DÉBITOS NA PREFEITURA DE ALEGRETE

Neste capítulo é apresentada a proposta de adaptação da aplicação de consulta de débitos na Prefeitura de Alegrete para acesso via dispositivos móveis. A Prefeitura de Alegrete disponibilizou para os contribuintes uma consulta de débitos via *Web*, endereço www.alegrete.gov.br, porém o meio de conexão deve ser um dispositivo fixo. Para atender melhor aos contribuintes, novos meios de acesso às informações devem ser disponibilizados.

O pedido de *login* e senha podem ser efetuados no próprio site da prefeitura, os dados cadastrados estando correto é retornado por *email* ao contribuinte, podendo assim acessar a página ou o celular.

As informações retornadas são referentes aos débitos que estão vencendo ou vencidos, os mesmos retornam atualizados com o cálculo de multa e juro atualizados.

Através desta consulta o contribuinte fica informado do valor real do seu débito, evitando assim entrar na fila para a referida consulta.

## 2.1 Descrição e requisitos

Considerando a importância da mobilidade para o crescimento das organizações e levando em conta o escopo do projeto, podemos destacar os seguintes requisitos do sistema:

- Dar acesso a consulta de débitos da prefeitura de Alegrete através do *login* e senha;
- Permitir a utilização da aplicação móvel fora da área de cobertura;
- Assegurar que a aplicação móvel possa ser instalada na maioria dos aparelhos celulares em operação;
- Verificar *login* e senha antes de retornar as informações;
- Retornar os dados atualizados.

### 2.1.1 Recursos de Hardware

Os requisitos de *hardware* usados no desenvolvimento, testes e operação do protótipo são:

- Micro-computador Pentium III 800 MHz ou superior;
- Memória 256 Mb ou mais;
- Placa de vídeo SVGA 800x600, Hi-Color;
- Disco rígido com 5 Mb livres ou mais;
- Servidor com linguagem PHP instalado e acesso a aplicação da prefeitura;

- Servidor de banco de dados;
- Aparelho celular com suporte para CLCD 1.0 e MIDP 1.0.

### 2.1.2 Recursos de Software

Os requisitos de *software* usados no desenvolvimento, testes e operação do protótipo são:

- JCreator
- J2SE
- Wireless Toolkit (J2ME)
- Banco de dados PostgreSQL 8.0;
- Linguagem PHP
- Acesso ao módulo servidor da aplicação da prefeitura;
- Navegador web.

## 2.2 Modelagem

Por limitações de escopo, o projeto foi implementado como um protótipo composto por dois módulos:

- Módulo PHP;
- Módulo Aplicação Móvel.

### 2.2.1 Módulo PHP

Primeiramente, o contribuinte deverá fazer a atualização do seu cadastro junto a prefeitura, se o cadastro já esta atualizado pode ser feito o pedido de senha direto na página da prefeitura acessando esse endereço <http://www.alegrete.rs.gov.br/dbpref/online>. Será retornado o *login* e senha para o e-mail cadastrado, conforme ilustra o caso de uso na Figura 2.1.

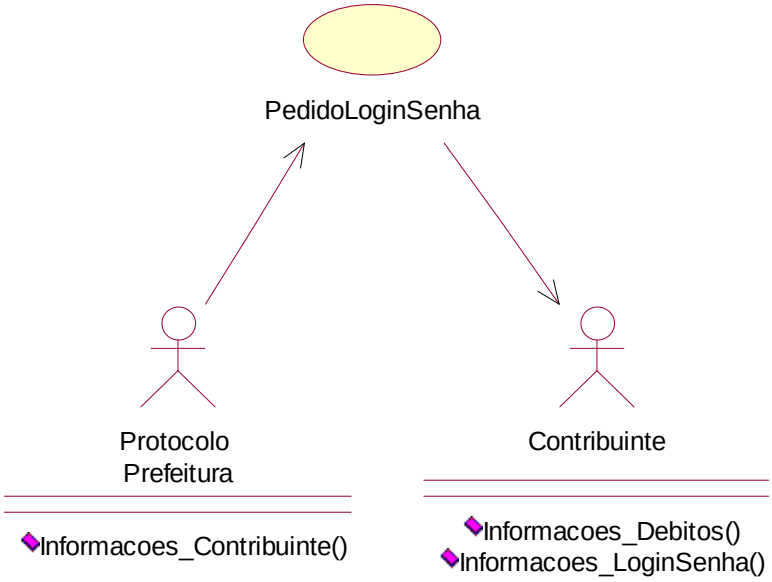


FIGURA 2.1 – Caso de Uso do Pedido de Login e Senha.

Através da notação UML (*Unified Modeling Language*), o diagrama de seqüência (Figura 2.2) mostra as funcionalidades do pedido de senha.

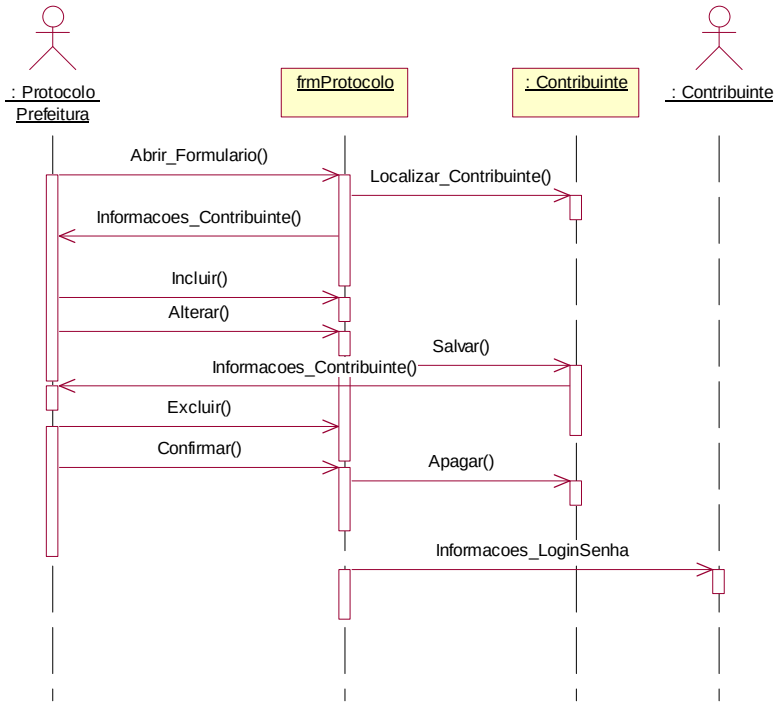

FIGURA 2.2 – Diagrama de Seqüência do Pedido de Login e Senha.

Na figura 2.3 tem-se uma representação de pedido de senha direto na página da prefeitura.

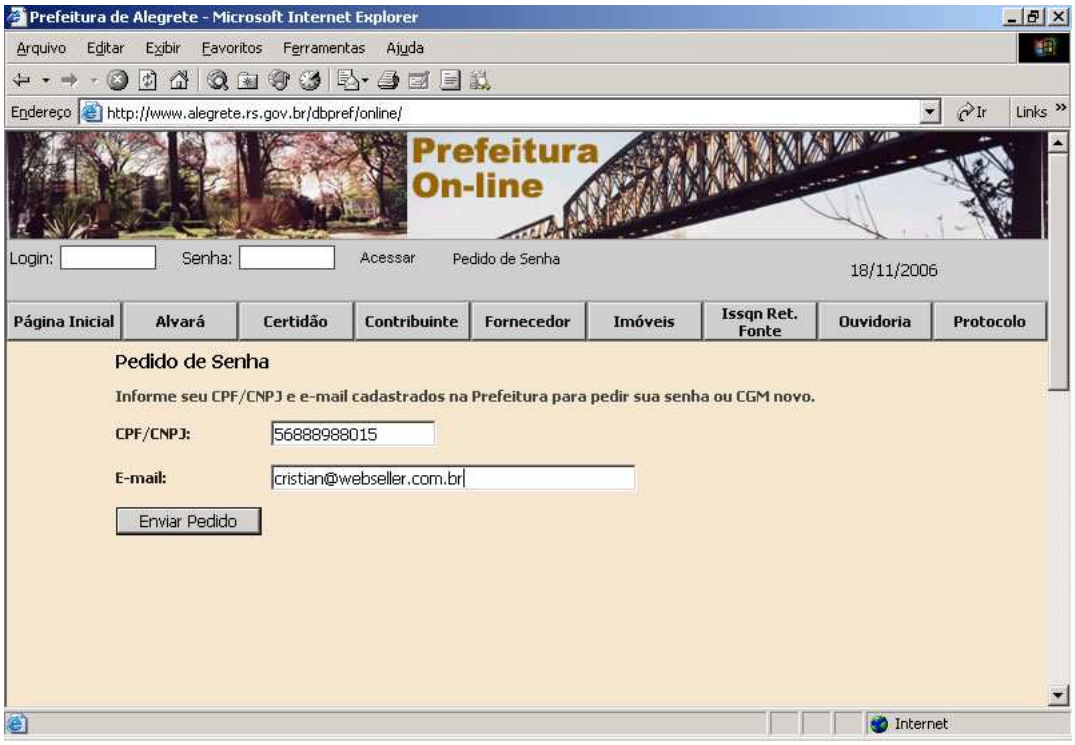


FIGURA 2.3 – Pedido de Login e Senha na página da Prefeitura.

O arquivo `pedido_senha.php` é responsável pela montagem da tela para o contribuinte fazer o seu pedido de *login* e senha ou fazer o acesso sistema, se o mesmo já tem permissão. A seguir, é mostrado um segmento de código do arquivo `pedido_senha.php` na Figura 2.4

```
...
<?
if(isset($cgccpf)){
 if($email_contribuinte == "") {
  msgbox("Email em branco");
  exit;
 }
 if($cgccpf == "") {
  msgbox("Erro: digite seu cgc ou cpf.");
  exit;
 }
 $cgccpf = str_replace(".","",$cgccpf);
 $cgccpf = str_replace("/","",$cgccpf);
 $cgccpf = str_replace("-","",$cgccpf);
 $sql = "select z01_numcgm,z01_nome,z01_email,z01_cgccpf from cgm where trim(z01_cgccpf) = '$cgccpf'";
 $result = @pg_query($sql) or die(@pg_errormessage());
 if(@pg_num_rows($result) == 0){
  db_logs("","",0,"Solicitação de senha para fornecedor: cgc ou cpf não encontrado. $cgccpf");
  if($w13_liberaatucgm=="t"){
   msgbox("Dados informados NÃO encontrados/atualizados no cadastro da Prefeitura! Você será direcionado para realizar o pedido
de cadastro ou atualização de CGM agora.");
   db_redireciona("atualizaendereco.php?w11_cgccpf=$cgccpf&w11_email=$email_contribuinte&cgmlogin=0");
   db_redireciona("pedido_senha.php");
  }else{
   msgbox("Dados informados NÃO encontrados no cadastro da Prefeitura! Procure o balcão da Prefeitura para realizar seu
cadastro.");
  }
  exit;
 }else{
  db_logs("","",0,"Solicitação de senha para fornecedor: cgc ou cpf - $cgccpf");
  db_fieldsmemory($result,0);
  //verifica se email confere com o cadastrado no cgm
  if($email_contribuinte!=$z01_email || empty($z01_email)){
   msgbox("Seu e-mail cadastrado na Prefeitura NÃO confere ou está em branco. Entre em contato com a Prefeitura para atualizar seu
cadastrado.");
   if($w13_liberaatucgm=="t"){
    db_redireciona("atualizaendereco.php?w11_cgccpf=$cgccpf&w11_email=$email_contribuinte&cgmlogin=0");
   }else{
    db_redireciona("pedido_senha.php");
   }
   exit;
  }
...
```

FIGURA 2.4 – Tributos Lançado para o Contribuinte.

Após o contribuinte efetuar o *login*, a aplicação verifica quais os atributos que foram lançados para ele, conforme ilustra a Figura 2.5.

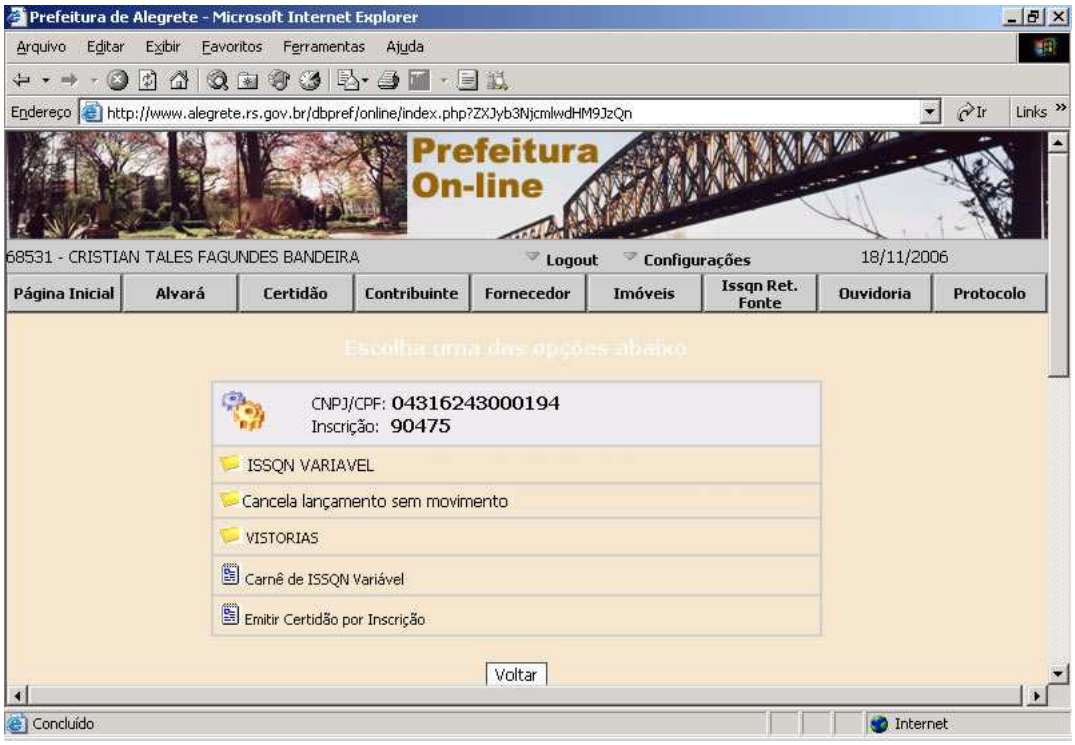

FIGURA 2.5 – Tributos Lançado para o Contribuinte.

O contribuinte quando seleciona um dos débitos é apresentado o valor do débito que foi lançado. Uma tela correspondente a este processo é mostrada na Figura 2.6

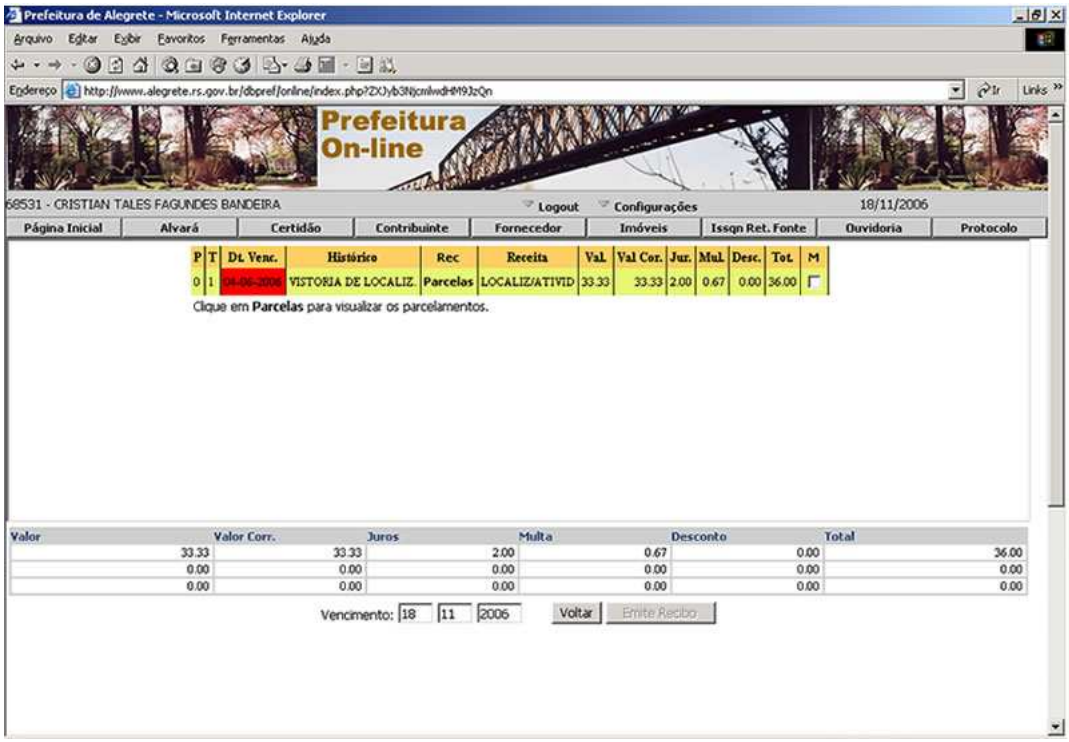


FIGURA 2.6 – Valor do Débito.

### 2.2.2 Módulo Aplicação Móvel

A aplicação desenvolvida corresponde a uma consulta de débitos da Prefeitura de Alegrete que utiliza a plataforma J2ME, ou mais especificamente, a configuração CLDC e o perfil MIDP, vistos anteriormente. Trata-se de uma aplicação simples, para ser utilizada em telefones celulares, que apresenta as seguintes opções:

- Consulta débitos;
- Tela de *login* e senha;
- Sair da aplicação;
- Voltar para uma nova consulta.

Através da notação em UML, o diagrama de casos de uso (Figura 2.7) mostra as

funcionalidades da aplicação.

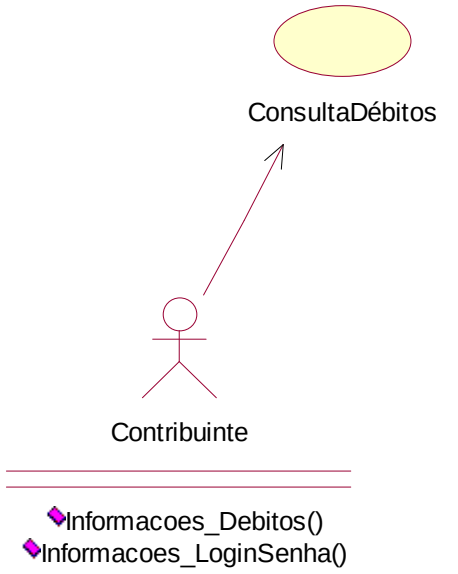


FIGURA 2.7 – Caso de Uso Consulta Débitos.

Na Figura 2.8, tem-se uma visão do digrama de seqüência.

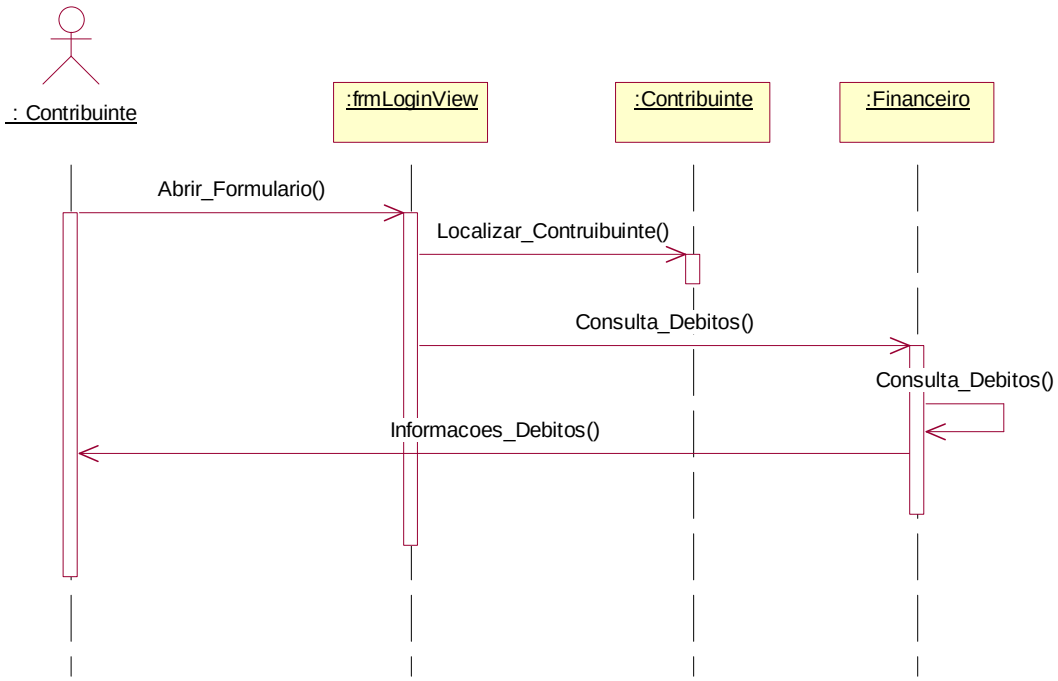


FIGURA 2.8 – Diagrama de Seqüência Consulta Débitos.

O módulo apresenta opção de configuração do sistema que pode ser alterado pelo contribuinte:

- Solicitar senha: o padrão do sistema é solicitar os dados de identificação (*login* e senha) apenas uma vez, armazenando-os no aparelho celular. Se desejar, o contribuinte poderá definir que o *login* e senha sejam solicitados sempre que entrar na aplicação.

A conexão com o servidor *web* é feita em uma linha de execução separada permitindo que o dispositivo continue disponível durante a operação. Para implementar a execução de processos concorrentes foi usada uma instância da classe `Thread`.

A padronização das *interfaces* é muito difícil no ambiente J2ME. O desenvolvedor pode e deve usar um padrão mas cada dispositivo irá implementa-lo à sua maneira.

As *interfaces* da aplicação são instanciadas pela classe `Community` onde são adicionados elementos do tipo `Button`, para escolha de algumas opções, `Label`, `TextField e Choice`.

## 2.3 Implementação

Para a exemplificação de alguns dos aspectos ressaltados no decorrer da elaboração desse trabalho, foi criada uma aplicação móvel chamada Prefeitura.

### 2.3.1 Aplicação Prefeitura

Quando iniciada, a aplicação apresenta uma opção de “Consulta Débitos” para o contribuinte selecionar. Após é apresentado uma tela para efetuar o *login*. A aplicação móvel é formada por um contêiner *web* onde estarão as *midlets* (clientes da aplicação) que em acesso ao servidor PHP, por sua vez, faz a conexão com a base de dados da prefeitura.

Esta aplicação utiliza o pacote `javax.microedition.rms`, *Record Management System*, que contém as classes necessárias para armazenamento persistente no dispositivo, no caso *login* e senha. É chamada de persistente, porque mesmo quando o dispositivo sofre *reboots*, troca de bateria ou desligamentos, os dados gravados permanecem inalterados para futuramente não precisar digitá-los novamente.

Para efetuar a comunicação com servidor da prefeitura é utilizado uma conexão HTTP através do pacote `javax.microedition.io`, por meio da classe `HttpConnection`.

> O HTTP é conhecido como um protocolo de pedido/resposta. Um cliente inicia um pedido, envia-o para um servidor com um endereço especificado como um URL (*Uniform Resource Locator*) e uma resposta é retornada do servidor. É assim que um navegador e um servidor Web trabalham em conjunto. (MUCHOW, 2005)

### 2.3.2 O Processo de Desenvolvimento

O código fonte da aplicação Prefeitura foi implementado em J2ME. Como ferramenta de edição de arquivos foi utilizada JCreator que é um IDE para desenvolvimento de aplicações *Open Source* (http://www.jcreator.com). Para a criação e compilação de arquivos na máquina virtual Java para aplicações móveis, utilizou-se o J2ME Wireless Toolkit da Sun, na versão 2.5 (http://www.sum.com). Através de um componente desse Toolkit, o `KtoolBar`, pode-se emular a aplicação desenvolvida e, com isso, verificar o funcionamento da aplicação.

### 2.3.3 Estrutura da Aplicação Prefeitura

A aplicação Prefeitura foi divida em cinco classes:

- `Prefeitura`
- `Community`;
- `View;`
- `ButtonListView`;
- `Dialog`;
- `LoginView`;
- `GetHTTP`.

Uma observação, que se faz necessária ser mencionada, é sobre a implementação da classe `LoginView`, é que para qualquer chamada de método para efetuar o *login*, se faz necessário a criação de um `Thread` para que essa troca de dados possa ser feita. A criação dessa `Thread`, é importante pois, caso o servidor da prefeitura não possa receber o dado enviado ou o próprio dispositivo local não consiga enviar esse dado, a aplicação não ficará travada esperando que essa transação seja completada.

#### 2.3.3.1 Classe `Prefeitura`

A classe `Prefeitura`, é responsavel pelo inicio da aplicação. Essa classe é a classe que estende `MIDLet` e por conseqüência implementa os métodos `startApp()`, `pauseApp()` e `destroyApp(boolean unc)` que devem ser definidos por qualquer objeto que *extends* de um `MIDLet`. A seguir mostramos o diagrama da classe `Prefeitura`, conforme ilustra a Figura 2.9.

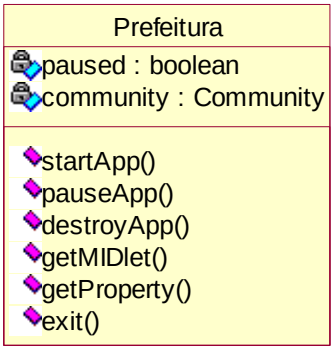


FIGURA 2.9 – Classe `Prefeitura`.

No apêndice A, tem-se uma descrição completa do código do referida objeto.

#### 2.3.3.2 Classe `Community`

A classe `Community` é responsável pela inicialização dos seguintes objetos:

- `ButtonListView;`
- `Dialog;`
- `LoginView;`
- `TextView.`

A seguir o diagrama da classe `Community`, conforme ilustra a Figura 2.10.

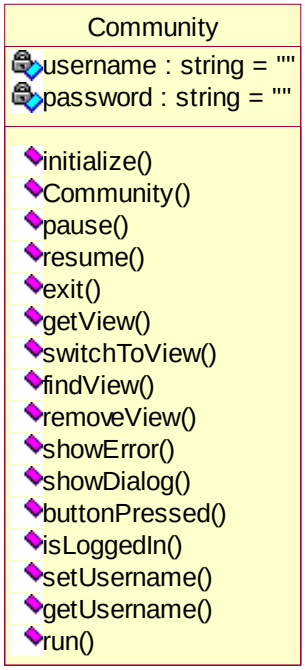


FIGURA 2.10 – Classe `Community`.

No apêndice B, tem-se uma descrição completa do código do referido objeto.

O método `View` é responsável pela montagem e ações da tela da aplicação, ou seja, através dessa classe toda a interface gráfica é criada e exibida na tela do celular. Essa classe é importada do pacote `samples.ui`.

### 2.3.3.3 Classe `ButtonListView`

Esta classe tem a funcionalidade de criar e definir o *layout* dos botões, podendo inicializar os botões com fontes e imagens especificadas. A seguir o diagrama da classe `ButtonListView,` conforme ilustra a Figura 2.11.

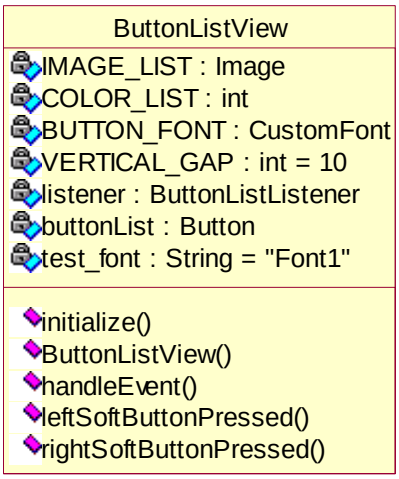


FIGURA 2.11 – Classe `ButtonListView`.

No apêndice C, tem-se uma descrição completa do código do referido objeto.

#### 2.3.3.4 Classe `Dialog`

Essa classe implementa uma tela com múltiplas linhas de texto ou diálogo de entrada de dados, também pode-se configurar as tecla de opções, com seus *label* e posicionamento (direita ou esquerda). Utilizada na aplicação para informar mensagem ao contribuinte ou pedir alguma confirmação. A seguir o diagrama da classe `Dialog`, conforme ilustra a Figura 2.12.

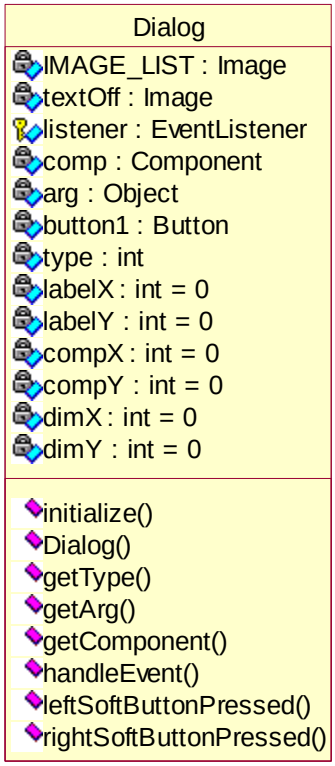


FIGURA 2.12 – Classe `Dialog`.

No apêndice D, tem-se uma descrição completa do código do referido objeto.

2.3.3.5 Classe `LoginView`

A classe `LoginView` implementa uma tela com `Label`, `TextField`, `Choice` e `Button` com esses métodos é montada a tela para o contribuinte informar o *login* e senha. Na mesma classe tem-se um método `validate` utilizado para verificar se todas as informações foram preenchidas, para que possa ser enviado ao servidor da prefeitura o *login* e senha, aonde será feita a consulta dos débitos. Logo a baixo tem-se o diagrama da classe `LoginView`, conforme ilustra a Figura 2.13.

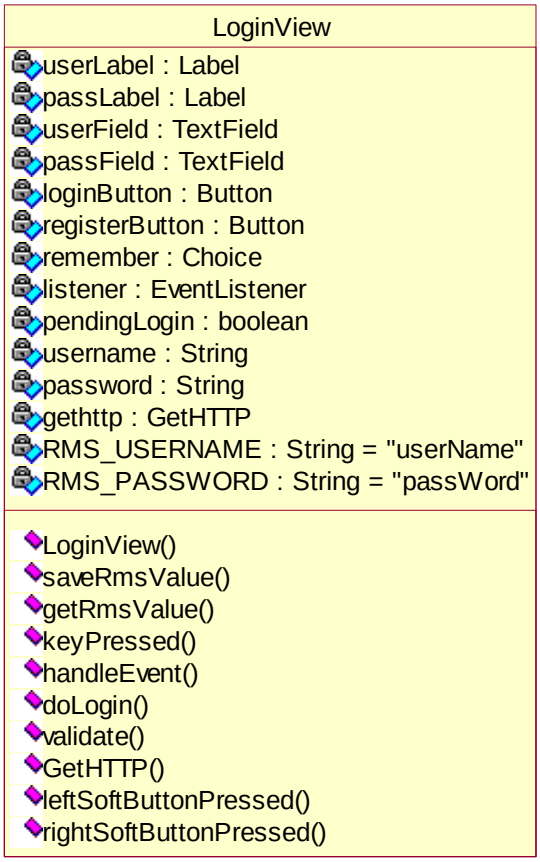


FIGURA 2.13 – Classe `LoginView`.

No apêndice E, tem-se uma descrição completa do código do referido objeto.

#### 2.3.3.6 Classe `GetHTTP`

A classe `GetHTTP`, faz a conexão com o servidor da prefeitura, o protocolo http é implementado pela classe `HttpConnection`, podendo comunicar-se com o servidor da prefeitura. Esse protocolo faz uso da abordagem de pedido/resposta, ou seja, o contribuinte inicia um pedido para o servidor através de um endereço contido em uma URL, que é informada na classe `LoginView`, a tarefa do servidor é responder essa requisição através da aplicação `cel001.php`, que encontra-se no servidor aonde recebe as informações de *login* e senha repassados pelo método `GetHTTP`. Na figura 2.14 tem-se uma descrição do código da aplicação `cel001.php`.

```
...
<?
  //conecta
  require("db_conn.php");
  include("db_fieldsmemory.php");
  parse_str($HTTP_SERVER_VARS['QUERY_STRING']);
  if(!($conn = pg_connect("host=$DB_SERVIDOR dbname=$DB_BASE port=$DB_PORTA user=$DB_USUARIO
password=$DB_SENHA"))) {
     echo "erro ao conectar...\n";
     exit;
  }

  if( empty($str_login) && empty($str_senha) ){
     echo "Login e Senha não informado.";
     exit;
  }
  //Pesquisa Login
  $str_sql = "select id_usuario,senha,login
                from db_usuarios
               where login = '$str_login'";
  $result = pg_exec( $str_sql );
...
   //Valida Senha
  $str_senha2 = pg_result($result,0,"senha");
  if($str_senha != $str_senha2){
     fwrite($arq_cel, "Senha Inválida.\n");
     echo "Senha Inválida.\n";
     exit;
  }

  $sql = "select * from cgm where z01_numcgm = ".pg_result($result,0,"login");
  $result = pg_exec( $sql ) or die ("ERRO >>> $sql");
  $z01_nome = pg_result($result,0,"z01_nome");
...
  //Pesquisa Débitos
  $str_sql = " select k00_descr,
                      sum(substr(fc_calcula,15,13)::float8+
                       substr(fc_calcula,28,13)::float8+
                       substr(fc_calcula,41,13)::float8-
                       substr(fc_calcula,54,13)::float8) as valor
               from (
                      select t.k00_tipo,t.k00_descr,

fc_calcula(r.k00_numpre,r.k00_numpar,r.k00_receit,r.k00_dtvenc,r.k00_dtvenc,'2006')
                        from arrenumcgm ar
                       inner join arrecad r on r.k00_numpre = ar.k00_numpre
                       inner join arretipo t on t.k00_tipo   = r.k00_tipo
                       where ar.k00_numcgm = $str_login
                         and r.k00_dtvenc <= now()
                     ) as unica
               group by k00_tipo,k00_descr;";

  $result = pg_exec( $str_sql );
...
```

FIGURA 2.14 – Aplicação `cel001.php` no servidor da prefeitura.

Logo a baixo tem-se o diagrama da classe `GetHTTP`, conforme ilustra a Figura 2.15 :

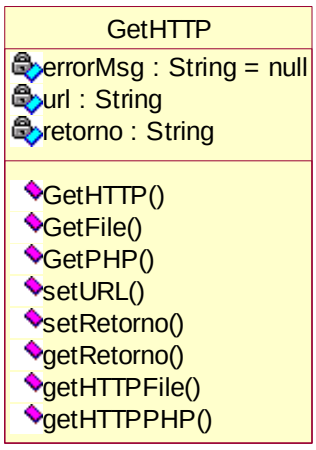

FIGURA 2.15 – Classe `GetHTTP`.

No apêndice F, tem-se uma descrição completa do código do referido objeto.

### 2.3.4 Resultados da Aplicação Prefeitura

Para visualização da aplicação, mostraremos telas capturadas durante a execução da mesma no emulador de dispositivos móveis (KtoolBar). As principais telas obtidas durante a emulação da aplicação serão mostradas a seguir.

Na Figura 2.16 (a), tem-se o uso um botão na aplicação, com o texto “Consulta Debitos”, e também os botões selecionar ou sair da aplicação.

Na Figura 2.16 (b), tem-se a tela de *login*, o contribuinte deverá informar o *login* e senha, após selecionar o botão *login*, que será utilizado para efetuar a conexão com o servidor da prefeitura.

(a) (b)

FIGURA 2.16 – Telas de Abertura da Aplicação.

2.3.4.1 Tela de Resultados

Na Figura 2.17, tem-se o resultado da consulta dos débitos do contribuinte, vindo os tributos e o seu valor.


FIGURA 2.17 – Resultado da Consulta de Débitos.

# 3. CONCLUSÕES E TRABALHOS FUTUROS

Nunca os dispositivos portáteis foram tão poderosos, oferecendo aos usuários novas e fascinantes formas de comunicação e de administração para seus negócios.

Java é uma tecnologia que possibilita aos desenvolvedores estender as aplicações existentes a estes dispositivos. Ela vem evoluindo com muita velocidade e nem sempre é fácil acompanhar as mudanças. Isso muitas vezes acaba se refletindo em uma grande quantidade de produtos e uma maior ainda variedade de versões.

É claro que o desafiador cenário globalizado estende o alcance e a importância do fluxo de informações. No mundo atual, a estrutura atual da organização possibilita uma melhor visão global das regras de conduta normativas. O empenho em analisar as diversas necessidades desafia a capacidade de equalização das direções preferenciais no sentido do progresso. Pensando mais a longo prazo, a constante divulgação das informações auxilia a preparação e a composição das diretrizes de desenvolvimento de novas aplicações.

O incentivo ao avanço tecnológico, assim como a competitividade nas transações comerciais talvez venha a ressaltar a relatividade do sistema de participação geral.

Neste trabalho, foi abordado o uso dessa tecnologia para o desenvolvimento de uma aplicação móvel para os contribuintes da Prefeitura de Alegrete. Essa aplicação tem o objetivo de agilizar o processo na consulta dos débitos disponibilizando em tempo hábil os referidos valores dos tributos atualizados.

Como trabalhos futuros pretendem-se definir novos meios para que o contribuinte atualize seu cadastro sem precisar ir à prefeitura, disponibilizando assim o meio de acesso à consulta de débitos mais rapidamente.

Segundo Burkhard (2007), a aternativa de maior interesse é o aviso da data de vencimento de tributos/taxas.

## REFERÊNCIAS BIBLIOGRÁFICAS


ALENCAR, PAULO. **Internet tem 91 milhões de domínios registrados**. Disponível em: http://info.abril.com.br/aberto/infonews/052006/25052006-6.sh. Acesso em 16 jun. 2006.

BURKHARD, LUCIANO M. **Serviços de M-Gov: Estudo de Caso na Administração Tributária.** 2007. 1 dispositivo, color.

BRASIL TELECOM, **Informativo nº 1 2006**. Disponível em: http://www.brasiltelecom.com.br. Acesso em 27 de jun. 2006.

CAÇADOR, FÁTIMA. **Sobre a Internet, WAP**. Disponível em: http://www.estudar.org/pessoa/internet/01internet/conceito-wap.html. Acesso em 16 jun. 2006.

JODE, MARTIN DE, **Programming Java 2 Micro Edition on Symbian OS**. Symbian Ltda, 2004.

LIMA, A. L. S. **Identificação e exemplificação de aspectos da computação distribuída no desenvolvimento de aplicações para celulares usando J2ME e Bluetooh**. Monografia (Graduação em Ciência da Computação) – Universidade Federal de Pernambuco, 2005.

MIRANDA, CLÁUDIO. **Dados em J2ME.** Curitiba- PR. Java Magazine, 2003.

MUCHOW, JOHN W. **Core J2ME: Tecnologia & MIDP**. Makron Books, 2005.

OLIVEIRA, ADELIZE GENERINI. **Desenvolvendo Site WAP**. Serial Sistemas e Consultoria Ltda, 2000.

PLUMMER, DARYL. **The Java Scenario, Simpósio Gartner 2000.** Disponível em: http://www.gartner.com/research/fellows/asset_55287_1175.jsp. Acesso em 30 de jun. 2006.

REIS, LUIS M. **Aplicações para Dispositivos Móveis utilizando a plataforma J2ME.** 2006. 1 dispositivo, color.

REVISTA WEBMOBILE, Rio de Janeiro, fev./mar. 2005

SASAKI, CURTIS. **Centro de Imprensa**. Disponível em: http://www.sun.com/aboutsun/media/bios/bios-sasaki.html. Acesso em 30 de jun. 2006.

SILVA, FLÁVIO, **Agentes Móveis**. Disponível em: http://www.lasid.ufba.br/public/resumos/sbc_wh.html. Acesso em 27 de jun 2006.

# APÊNDICES

## APÊNDICE A – Classe Prefeitura

```
/* ----------------------------------------------------------------------- *
        Wireless Software Solutions
 * ----------------------------------------------------------------------- */

package samples.commui;

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


public class Prefeitura extends MIDlet implements MainApp {
   private boolean paused;
   private Community community;

   public void startApp() {
      if (!paused && community == null) {
            community = new Community(this);
      } else {
            community.resume();
      }
      paused = false;
   }
   /**
    * Called by the handset when the MIDlet is paused.
    */
   public void pauseApp() {
      paused = true;
      if (community != null) community.pause();
   }

   /**
    * Destroys and exits MIDlet.
    */
   public void destroyApp(boolean unc) {
      Display.getDisplay(this).setCurrent(null);
      notifyDestroyed();
   }

   /**
    * Returns MIDlet reference.  Method inherited from interface
    *
    * @return MIDlet reference
    */
   public MIDlet getMIDlet() {
      return this;
   }

   /**
    * Returns MIDlet property.  Method inherited from interface
    *
    * @param string Name of property
    * @return Value of property
    */
```

```
    public String getProperty(String string) {
        return getAppProperty(string);
    }

    /**
     * Exits MIDlet.  Method inherited from interface
     */
    public void exit() {
        destroyApp(true);
    }
}
```

## APÊNDICE B – Classe Community

```
/* ---------------------------------------------------------------------- *
        Wireless Software Solutions
 * ---------------------------------------------------------------------- */

package samples.commui;
import java.util.Enumeration;
import java.util.Random;
import java.util.Vector;
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Image;
import samples.ui.Component;
import samples.ui.Event;
import samples.ui.EventListener;
import samples.ui.ResourceManager;
import samples.ui.TextBox;
import samples.ui.TextField;
import samples.ui.View;
import samples.ui.ViewCanvas;

import com.nokia.sm.net.ItemList;
import com.nokia.sm.net.ServerComm;
import com.nokia.sm.net.SnapEventListener;


public class Community implements SnapEventListener, ButtonListListener, EventListener, Runnable {
    public static final String CHALLENGE_PREFIX = "+++ ChAlLeNgE +++: ";
    public static final String NON_CRITICAL = "non-critical";
    public static int X=0;
    public static int Y=0;

    static final int    MAX_NEW_ACCOUNTS_PER_DAY= 5;

    private static final Random rnd = new Random();

    static public final Image FADING_BACKGROUND =
ResourceManager.getImage("/fading_background.png");
    static final Image OVERLAY_BACKGROUND =
ResourceManager.getImage("/overlay_background.png");
    static final Image SELECTED_SECTION   = ResourceManager.getImage("/selected_section.png");
    static final Image SPLASH_IMG        = ResourceManager.getImage("/device_splash.png");

    static final String About_txt                = "/about.txt";
    static final String Motd_txt                 = "/motd.txt";
```

```
    static final String Game_Help_txt   = "/game_help.txt";
    static final String SM_Help_txt     = "/sm_help.txt";
    static final String Operator_Help_txt = "/operator_help.txt";

    static final String SPLASH           = "Splash";
    static final String GAME             = "Game";
    static final String MOTD             = "Message of the day";
    static final String WELCOME                                    = "Bem Vindo!";
    static final String GO_ONLINE        = "Consulta Debitos";
    static final String SINGLE_USER      = "single user game";
    static final String ABOUT                                      = "about";
    static final String CREATE_ACCOUNT   = "create account";
    static final String DIALOG           = "Dialog";
    static final String CHAT             = "chat messages";
    static final String FRIENDS          = "Friends";
    static final String HOME             = "Home";
    static final String PLAY             = "Play";
    static final String PENDING_GAMESTART  = "Waiting for player";
    static final String RANKING          = "rankings";
    static final String TOP_10           = "Top 10";
    static final String MY_RANKINGS      = "My Rankings";
    static final String STATS            = "Stats";
    static  String HELP                          = "help";
    static final String SM_HELP          = "SNAP Mobile Help";
    static final String GAME_HELP        = "Game Help";
    static final String OPERATOR_HELP    = "Operator Help";
    static final String EMAIL_DOB        = "E-mail & Birthdate";
    static final String USER_PASS        = "Login & Senha";
    static final String TERMS            = "Terms";
    static final String SUBMIT_SCORES    = "SubmitScores";

    static final String SELECT           = "selecionar";
    static final String SEND             = "enviar";
    static final String CANCEL           = "cancelar";
    static final String BACK             = "voltar";
    static final String MAZEEXIT                 = "mazeexit";
    static final String EXIT             = "sair";
    static final String REALLY_EXIT      = "sair!";
    static final String LOGIN            = "login";
    static final String LOGOUT           = "log out";
    static final String REALLY_LOGOUT    = "log out!";
    static final String NEXT             = "próximo";
    static final String OK               = "ok";
    static final String YES              = "sim";
    static final String NO               = "nao";

    static final String RMS_USERNAME     = "userName";
    static final String RMS_PASSWORD     = "passWord";
//    static final String RMS_SAVE_PASSWORD  = "savePass";

    //  GUI data
    private ViewCanvas canvas;
    /*private MazeRacer mazeRacer;*/
    private Vector cmdList;
    //private BuddyList buddyList;
    private String[] statList;
    private boolean inGame;

    //Snap specific data
    private String webSessionID;
```

```
    private String impsSessionID;
    private String snapSessionID;
    private String snapUserID;
    private ServerComm comm;
    private Vector viewList;
    private View motd;
    private MainApp main;
    private String lastError;
    private int lastErrorSeverity;
    private Integer gcid;
    private String operatorId;
    private boolean done;
    private boolean waitingForGame;
    //private AsyncCommandListener asyncListener=null;
    //private String asyncCommand=null;


    // User login/registration info -- save here for later retrieval
    public String      username          = "";
    public String      password          = "";
    public String      password2         = "";
    public String      dateOfBirth       = "";
    public String      emailAddress      = "";
    public int         minAge            = 13;

    protected int      getMinAge()           { return minAge; }
    protected MainApp  getMainApp()          { return main; }


    /*
     * ========================================================
*   Instance lifecycle methods: static initializer, constructor, pause
     *  resume and exit methods.
     * ========================================================    */

    /**
     * Calls static initializers on View and SnapLoginView.
     */

    static void initialize() {
       CommunityView.initialize();
       View.initialize();
       ButtonListView.initialize();
       Dialog.initialize();
       LoginView.initialize();
       TextView.initialize();
    }

    /**
     * Constructor.  Loads settings from .jad file.  Performs network
     * connectivity check if connectivity has not been established in the
     * past.  Then, instantiates and displays LoginView screen.
     *
     * @param main Reference to caller's code, via the <code>MainApp</code>
     *  interface.  This reference is used to return control back to the
     *  calling code once login and/or registration are complete, by means
     *  of the handleEvent() method.
     */
    public Community(MainApp main) {
```

```
    this.main = main;

    cmdList = new Vector();
    viewList = new Vector();

    gcid = new Integer( 49721);
    username = "";

    View splash = getView(SPLASH);
    splash.setActive(true);


    canvas = new ViewCanvas(splash);
    Display.getDisplay(main.getMIDlet()).setCurrent(canvas);
    canvas.waitForResize();


    initialize();


    switchToView(WELCOME);

    Thread thread = new Thread(this);
    thread.start();
}

public void pause(){}

public void resume(){}

/**
 * Saves username/password if login saving is enabled, and is checked
 * "on" by the user. Then, returns control back to calling code via the
 */
public void exit() {
    System.out.println("Saindo MIDLET");
    main.exit();
}


/**
 * Returns active View.
 *
 * @return View
 */
public View getView()
{
        return canvas.getView();
}

/**
 * Returns a particular View by name.  View is created if it does not
 * already exist, or if View caching is turned off.  Requesting View
 * "back" will pop the current View off the View stack, destroy it,
 * and return the previous View.  Requesting View "exit" will cause
 * SnapLogin to exit.
 *
 * @param name Name of the view to return
 */
View getView(String name) {
```

```
String url;
View view;
int i;

view = null;

synchronized (viewList) {
   if (viewList.size() > 1 && name.equals(BACK)) {
      viewList.removeElementAt(viewList.size() - 1);
      view = (View)viewList.elementAt(viewList.size() - 1);
      return view;
   }
}

view = findView(name);

if (view != null) return view;


else if (name.equals(USER_PASS))
   view = new UserPassView  ( this, USER_PASS);


else if (name.equals(EXIT)) {
       showDialog( "Sair", "Voce quer sair?",
                           null, Community.REALLY_EXIT, Dialog.YES_NO
                           );
}
else if (name.equals(REALLY_EXIT)) {
   view = null;
   exit();
}


else if (name.equals(SPLASH)) {
   view = new SplashView(
      this,
      SPLASH,
      null,
      SPLASH_IMG,
      WELCOME,
      3000
   );
}


else if (name.equals(Community.WELCOME)) {
   view = new ButtonListView(
      this,
      Community.WELCOME,
      this,
      null,
      new String[] {Community.GO_ONLINE },
      Community.SELECT,
      Community.EXIT
   );
}


else if (name.equals(LOGIN)) {
```

```
            view = new LoginView(
                this,
                LOGIN,
                this
            );
        }
        return view;
    }

    /**
     * Tells GUI to instantiate (if necessary) and display a
     * particular named view.  "back" and "exit" have special
     * behavior (see <code>getView</code>).
     *
     * @param name Name of View to display
     */

    public void switchToView(String name) {
        switchToView(getView(name), !name.equals(BACK));
    }

    /**
     * Tells GUI to instantiate (if necessary) and display a
     * particular named view.  "back" and "exit" have special
     * behavior (see <code>getView</code>).
     *
     * @param view View to display
     * @param cache Caching flag; if <code>false</code>, current
     *  view will not be preserved on the View stack (disabling
     *  "back" behavior for that View)
     */
    public void switchToView(View view, boolean cache) {
        View prev;

        if (view == null) {
            return;
        }

        prev = canvas.getView();
        prev.setActive(false);
        view.setActive(true);
        canvas.setView(view);

        if (cache) {
            synchronized (viewList) {
                while (viewList.contains( view)) {
                    viewList.removeElementAt( viewList.size()-1);
                }
                viewList.addElement(view);
            }
        }
    }

    /**
     * Searches for a particular named View in the View stack (cache).
     *
     * @param name Name of View to find
     * @return Returns named View, if found
     */
    public View findView(String name) {
```

```
    View view;
    int i;

    synchronized (viewList) {
        for (i=viewList.size()-1; i>=0; i--) {
            view = (View)viewList.elementAt(i);
            if (name.equals(view.getName())) return view;
        }
    }

    return null;
}

/**
 * Removes a particular named View from the View stack.
 * @param name Name of View to remove.
 */
public void removeView(String name) {
    synchronized (viewList) {
        View view = findView(name);
        if (view != null) viewList.removeElement(view);
    }
}

/**
 * Causes currently visible canvas to repaint itself.
 *
 */
public void repaint()
{
    getCanvas().repaint();
}

/**
 * Returns currently visible canvas.
 * @return
 */
public Canvas getCanvas() {
    return canvas;
}


/**
 * Shows an error dialog w/ a string message and one "OK"
 * softkey in the left corner of the screen.  "OK" just
 * goes back to the previous screen.
 *
 * @param msg text of error message to display
 */
public void showError(String msg) {
    showError( msg, null, Dialog.ALERT);
}

/**
 * Shows error message passed in, in full-screen Dialog.
 *
 * @param msg The error message to be displayed
 * @param target If not <code>null</code>, error dialog will
 *   go to this View upon exiting.  Otherwise, will typically
 *   go "BACK"
```

```
     * @param type type of dialog.  See <code>Dialog<code> class.
     */
    public void showError(String msg, String target, int type)
    {
       //System.out.println("Community.showError(), msg = " + msg);
            String name = "Error";
            switch (type) {
            case Dialog.ALERT:
                    name = "Alerta";
                    break;
            case Dialog.ALERT_FATAL:
            case Dialog.ALERT_LOGOUT:
                    name = "Logout Error";
          msg += " -- You must log back in from the main menu.";
                    break;
            }
       showDialog( name, msg, null, target, type);
    }


    /**
     * Shows message passed in, in a full-screen Dialog.
     *
     * @param title
     * @param contents Must be either a String or a Component
     * @param arg Argument passed to listener when activated via SELECT
     * @param targetView If not null, View to go to upon exiting
     * @param type Type of dialog (see Dialog class for types).
     */
    public void showDialog(String title, Object contents, Object arg, String targetView, int type ) {

       String rightSoft = null;
       String leftSoft = null;

       // Contents to be displayed - must be either a String or a Component
       Component comp = null;
       if (contents instanceof Component) {
            comp = (Component)contents;
       } else {
         TextBox box = new TextBox();
         if (contents instanceof String) {
            box.setText( (String)contents);
         } else {
            box.setText(
                        "Error! Unrecognized contents: " +
                        contents.toString()
                        );
         }
         box.setBackground(0x00ffffff);
         box.setForeground(0x00c0c0c0);
         box.setFocusable(false);
         box.setDimension( canvas.getWidth() - 10, 12 * box.getFont().getHeight() +
TextBox.WIDTH_OFFSET);
         comp = box;
       }

       // Set left/right softkeys based on type
       switch (type) {
       case Dialog.YES_NO:
            leftSoft = Community.YES;
```

```
            rightSoft = Community.NO;
            break;
        case Dialog.DATA_ENTRY:
        case Dialog.OK_CANCEL:
                leftSoft = Community.OK;
                rightSoft = Community.CANCEL;
                break;
        // All other alerts have only "OK" on left softkey
        default:
                leftSoft = Community.OK;
            break;
        }

        Dialog dialog = new Dialog(
            this,
            title,
            this,
            leftSoft,
            rightSoft,
            comp,
            arg,
            type,
            targetView
        );

        switchToView(dialog, true);
    }


    /**
     * Callback for button list views
     *
     * @param view Name of View
     * @param button Name of Button
     */
    public void buttonPressed(String view, String button) {
        if (view.equals(Community.WELCOME)) {
            if (button.equals( Community.SINGLE_USER)) {
                /*startGame(true);*/
            }

            else if (button.equals( Community.GO_ONLINE)) {
                switchToView(LOGIN);
            }
            else if (button.equals( Community.ABOUT)) {
                switchToView( ABOUT);
            }
            else if (button.equals( Community.HELP)) {
                switchToView( HELP);
            }
        }

        else if (view.equals(LOGIN)) {
                if (button.equals(Community.LOGIN)) {
                        switchToView(LOGIN);
                }

                else if (button.equals(Community.CREATE_ACCOUNT)) {
                        switchToView(CREATE_ACCOUNT);
                }
```

```
            }

        }

        /*
         * Event handler implemented from samples.ui.EventListener interface.
         *
         * @param e The event
         */
        public void handleEvent(Event e) {
            //System.out.println("Community.handleEvent(), source = " + e.getSource() + ", value = " +
e.getValue());
            View view = (View)e.getSource();
            String action = (String)e.getValue();
            ItemList il;

            if (e.getSource() instanceof LoginView) switchToView(HOME);

            // Splash screen
            else if (e.getSource() instanceof SplashView) {
                switchToView(Community.WELCOME);
            }
            // Dialogs
            else if (e.getSource() instanceof Dialog) {
                Dialog dialog = (Dialog)e.getSource();
                String name = dialog.getName();


                if (name.equals("Alerta") || name.equals("Error") || name.equals("Note")) {
                    switchToView(Community.BACK);
                }

            }

        }


        /** Returns <code>true</code> if user has successfully logged in. */
        public boolean isLoggedIn() {
            return webSessionID != null;
        }

        public String getProperty(String name) {
            return main.getProperty(name);
        }

        public Integer getGCID() {
            return gcid;
        }

        void setUsername(String username) {
            this.username = username;
        }

        /** Returns user's username */
        String getUsername() {
            return username;
        }
```

```
    public void processServerError(int error, String msg, int severity) {
        synchronized (this) {
            lastError = msg + " " + error;
            lastErrorSeverity = severity;
        }

    }

    public void processEvents(Vector list) {
        ItemList il;
        String from, msg, gameRoom;
        byte[] data;
        String event = "";
    }


    /**
     * Main event loop for sending SNAP commands to server.
     * Handles sending commands, catching error results, and notifying
     * Views of the success/failure of their requests.
     */
    public void run() {
        /*AsyncCommandListener listener;*/
        ItemList itemList, returnValues;
        String error, cmd;

    }
}
```

## APÊNDICE C – Classe ButtonListView

```
/* ---------------------------------------------------------------------------- *
        Wireless Software Solutions
 * ---------------------------------------------------------------------------- */

package samples.commui;

import javax.microedition.lcdui.*;
import samples.ui.*;

/**
 * Class to define the layout of the Button List
 */
public class ButtonListView extends CommunityView implements EventListener {
    public static Image[] IMAGE_LIST;
    public static int[] COLOR_LIST;
    public static CustomFont BUTTON_FONT;
   // public static Font BUTTON_FONT;


    private static final int VERTICAL_GAP = 10;

    private ButtonListListener listener;
    private Button[] buttonList;
    private static String test_font="Font1";

    /**
```

```
     * iitialize buttons with fonts and images from resources
     */
    public static void initialize() {
        //BUTTON_FONT = Font.getFont(Font.FACE_PROPORTIONAL, Font.STYLE_BOLD,
Font.SIZE_LARGE);
            BUTTON_FONT = new CustomFont(test_font);

        IMAGE_LIST = new Image[] {
            ResourceManager.getImage("/button_on.png"),
            ResourceManager.getImage("/button_off.png"),
            ResourceManager.getImage("/button_pressed.png")
        };

        COLOR_LIST = new int[] {0x001e64b4, 0x00787d84, 0x00ffffff};
    }

    /**
     * Sets up widget appearance and layout.
     * @param community Main <code>Community</code> instance
     * @param name Name of view (used for navigation)
     * @param listener
     * @param buddyList
     * @param buttonNameList
     * @param leftSoft
     * @param rightSoft
     */
    public ButtonListView(Community community, String name, ButtonListListener listener,
        BuddyList buddyList, String[] buttonNameList, String leftSoft, String rightSoft)
    {
        super(community, name);

        int i, y, height;

        this.listener = listener;

        setBackgoundImage(Community.FADING_BACKGROUND);
        setLeftSoftButton( leftSoft);
        setRightSoftButton( rightSoft);

        if (buddyList != null && community.isLoggedIn()) {
            buddyList.setLocation(getWidth() - buddyList.getWidth() - 2, 0);
            add(buddyList);
        }

        i = buttonNameList.length;
        height = i * IMAGE_LIST[0].getHeight() + (i - 1) * VERTICAL_GAP;
        y = getHeight() / 2 - height / 2;

        buttonList = new Button[buttonNameList.length];
        for (i=0; i<buttonList.length; i++) {
            buttonList[i] = new Button(buttonNameList[i]);
            buttonList[i].setFont(BUTTON_FONT);
            buttonList[i].setStateData(IMAGE_LIST, COLOR_LIST);
            buttonList[i].addEventListener(this);
            buttonList[i].setDimension(
                IMAGE_LIST[0].getWidth(),
                IMAGE_LIST[0].getHeight()
            );
            buttonList[i].setLocation((getWidth() - buttonList[i].getWidth()) / 2, y);
```

```
            add(buttonList[i]);

            y += IMAGE_LIST[0].getHeight() + VERTICAL_GAP;
        }
    }

    /**
     * Handles "SELECTED" events for pushbuttons:
     *
     */
    public void handleEvent(Event e) {
        Button button = (Button)e.getSource();
        if (e.getType() == Event.ITEM_DESELECTED) {
            listener.buttonPressed(getName(), button.getText());
        }
    }

    public void leftSoftButtonPressed(String label) {
        if (label.equals(Community.SELECT)) {
            keyPressed(ENTER_BUTTON);
            keyReleased(ENTER_BUTTON);
        }
    }

    public void rightSoftButtonPressed(String label) {
                community.switchToView( label);
    }
}
```

## APÊNDICE D – Classe `Dialog`

```
/* ------------------------------------------------------------------------ *
        Wireless Software Solutions
 * ------------------------------------------------------------------------ */

package samples.commui;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Image;
import samples.ui.*;

/**
 * Implements a full-screen multi-line text or data input dialog, with
 * configurable softkeys (labels and targets.)
 *
 */
public class Dialog extends CommunityView implements EventListener {

    private String targetView = null;

    private static Image[] IMAGE_LIST;
    private static Image textOff;

    protected EventListener listener;
    private Component comp;
    private Object arg;
    private Button button1;
```

```
    private int type;
    private int labelX=0, labelY=0, compX=0, compY=0, dimX, dimY;

    static public final int       ALERT                 = 1;
    static public final int ALERT_LOGOUT    = 2;
    static public final int ALERT_FATAL               = 3;
    static public final int DATA_ENTRY                = 4;
    static public final int YES_NO                    = 5;
    static public final int OK_CANCEL                 = 6;

    public static void initialize() {
       textOff = ResourceManager.getImage("/text_off.png");

       IMAGE_LIST = new Image[] {
          ResourceManager.getImage("/short_button_on.png"),
          ResourceManager.getImage("/short_button_off.png"),
          ResourceManager.getImage("/short_button_pressed.png")
       };
    }

    /**
     * Creates and sets up widgets and softkeys for dialog.
     *
    * @param community <code>Community</code> main instance
    * @param name Name of screen (used for inter-screen navigation by <SnapLogin>)
     * @param listener Eventlistener
     * @param left Target view visited when left softkey pressed
     * @param right Target view visited when right softkey pressed
     * @param type Body of the message to display in the dialog
     *
    */

    public Dialog(Community community, String name, EventListener listener, String left, String right,
Component comp, Object arg, int type, String target) {
       super(community, name);

       this.comp = comp;
       this.arg = arg;
       this.type = type;
       this.targetView = target;
       this.listener = listener;

       Button button2;
       Label label;

       showTitle = false;
       setLeftSoftButton(left);
       setRightSoftButton(right);
       setBackgoundImage(Community.OVERLAY_BACKGROUND);

       dimX   = 70;
       dimY   = 100;
       if( Community.X == 0){
          dimX   = 0;
       }
       if( Community.Y == 0){
          dimY   = 0;
       }

       textOff = ResourceManager.getImage("/text_off.png");
```

```
    label = new Label(name, true);
    label.setBackgroundImage(new Image[] {textOff});
    label.setDimension(textOff.getWidth()+dimX, textOff.getHeight());
    // Center the dialog contents relative to the the background image
    labelX = Community.X;
    labelY = Community.Y;

    if( Community.X == 0){
            labelX = (getWidth() - label.getWidth()) / 2;
            compX = labelX;
    }
    if( Community.Y == 0 ){
       labelY = ((getHeight() - getBackgroundImage().getHeight()) / 2) + 7;
       compY = labelY + getHeight();
    }
    label.setLocation( labelX, labelY );
    add(label);

    //comp.setLocation(label.getX(), label.getY() + label.getHeight());
    comp.setLocation(compX, label.getY()+label.getHeight());
    comp.setDimension(label.getWidth()+dimX, comp.getHeight()+dimY);
    if (comp instanceof TextField) setFocus(comp);
    add(comp);
}

public int getType() {
            return type;
}

public Object getArg() {
    return arg;
}

public Component getComponent() {
    return comp;
}

/**
 * EventHandler
 * @param e Event to handle
 */

public void handleEvent(Event e) {
    if (e.getType() == Event.ITEM_DESELECTED) {
            if (targetView==null) {
                        listener.handleEvent(new Event(this, e.getType(), e.getValue()));
        } else {
                        community.switchToView( targetView);
            }
    }
}

/**
 * If there is a <code>leftTarget</code> set, visits the
 * <code>View</code> with that name.
 */
public void leftSoftButtonPressed(String label) {
            if (targetView==null) {
                        if (leftSoft != null) {
                                    listener.handleEvent(new          Event(this,          Event.ITEM_DESELECTED,
```

```
leftSoft));
                        }
            } else {
                        community.switchToView( targetView);
                }
        }

        /**
         * If there is a <code>rightTarget</code> set, visits the
         * <code>View</code> with that name.
         */
        public void rightSoftButtonPressed(String label) {
                if (label == null) return;
                if (label.equals(Community.CANCEL) || label.equals(Community.BACK)
                                || label.equals(Community.NO))
                {
                        community.switchToView( Community.BACK);
                }
        }
}
```

## APÊNDICE E – Classe `LoginView`

```
/* ---------------------------------------------------------------------- *
        Wireless Software Solutions
 * ---------------------------------------------------------------------- */

package samples.commui;

import java.io.*;
import java.lang.*;
import javax.microedition.io.*;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Image;
import javax.microedition.rms.RecordStore;
import com.nokia.sm.net.ItemList;
import com.nokia.sm.net.SnapEventListener;

import samples.ui.*;
public class LoginView extends CommunityView implements EventListener{
    private Label                   userLabel;
    private Label                   passLabel;
    private TextField               userField;
    private TextField               passField;
    private Button                  loginButton;
    private Button                  registerButton;
    private Choice                  remember;
    private EventListener   listener;
    private boolean         pendingLogin;
    private String                  username;
    private String                  password;
    private GetHTTP         gethttp;

    static final String RMS_USERNAME    = "userName";
    static final String RMS_PASSWORD    = "passWord";
```

```
/**
 * Sets up UI appearance and layout.
 *
 * @param community <code>Community</code> main instance
 * @param name Name of screen (used for inter-screen navigation)
 */
public LoginView(Community community, String name, EventListener listener) {
   super(community, name);
   String uid, pw;

   username = "";
   password = "";
   this.listener = listener;
   setLeftSoftButton( Community.SELECT);
   setRightSoftButton( Community.BACK);
   setBackgoundImage( Community.FADING_BACKGROUND);

   userLabel = new Label("Login", false);
   userLabel.setBackgroundImage(new Image[] {text_on});
   userLabel.setDrawShadows(false);
   userLabel.setDimension(text_off.getWidth(), text_off.getHeight());
   userLabel.setLocation((getWidth() - userLabel.getWidth()) / 2, 18);

   add(userLabel);

   uid = getRmsValue( RMS_USERNAME);
   if (uid.length() == 0) uid = "";

   userField = new TextField(15);
   userField.setText( uid);
   userField.setDrawShadows( false);
   userField.setForeground( 0x00c0c0c0);
   userField.setFont( TEXTFIELD_FONT);
   userField.setEntryMode( TextField.ENTRY_USERNAME);
   userField.setLocation( userLabel.getX(), userLabel.getY() + userLabel.getHeight());
   userField.setDimension( userLabel.getWidth(), 20);

   add(userField);

   passLabel = new Label("Senha", false);
   passLabel.setBackgroundImage(new Image[] {text_off});
   passLabel.setDrawShadows(false);
   passLabel.setLocation(userField.getX(), userField.getY() + userField.getHeight() + 5);
   passLabel.setDimension(text_off.getWidth(), text_off.getHeight());

   add(passLabel);

   pw = getRmsValue( RMS_PASSWORD);
   if (pw.length() == 0) pw = "";

   passField = new TextField(15);
   passField.setText(pw);
   passField.setDrawShadows(false);
   passField.setFont(TEXTFIELD_FONT);
   passField.setForeground(0x00c0c0c0);
   passField.setEntryMode( TextField.ENTRY_ASCII);
   passField.setDispMode( TextField.DISP_PASSWORD);
   passField.setLocation(passLabel.getX(), passLabel.getY() + passLabel.getHeight());
   passField.setDimension(passLabel.getWidth(), 20);
```

```
            add(passField);
            remember = new Choice(check_off, check_on, "Salva Login?");
            remember.setState( !(uid.equals("") && pw.equals("")));
            remember.setDrawShadows(false);
            remember.setBackgroundImage(new Image[] {text_off, text_on});
            remember.setLocation(passField.getX(), passField.getY() + passField.getHeight() + 5);
            remember.setDimension(text_off.getWidth(), text_off.getHeight());
            add(remember);
            loginButton = new Button("login");
            loginButton.setFont(ButtonListView.BUTTON_FONT);
            loginButton.setStateData(ButtonListView.IMAGE_LIST, ButtonListView.COLOR_LIST);
            loginButton.addEventListener(this);
            loginButton.setDimension(
                  ButtonListView.IMAGE_LIST[0].getWidth(),
                  ButtonListView.IMAGE_LIST[0].getHeight()
            );
            loginButton.setLocation((getWidth() - loginButton.getWidth()) / 2, remember.getY() +
remember.getHeight() + 5);

            add(loginButton);
         }
         /**
          * Save LoginView info in database
          * @param name  Name of the LoginView Field
          * @param value LoginView ID value
          */
         private void saveRmsValue(String name, String value) {
            try {
               RecordStore store = RecordStore.openRecordStore(name, true);

               if (store.getNumRecords() > 0) {
                  store.setRecord(1, value.getBytes(), 0, value.length());
               } else {
                  store.addRecord(value.getBytes(), 0, value.length());
               }
               store.closeRecordStore();
            } catch (Exception e) {
               e.printStackTrace();
            }
         }

         /**
          * Get LoginView from RMS
          * @param name LoginView Name
          * @return login String
          */
         private String getRmsValue(String name) {
            String value = "";
            try {
               RecordStore store = RecordStore.openRecordStore(name, true);
               if (store.getNumRecords() > 0) {
                  byte[] buf = store.getRecord(1);
                  value = buf == null ? "" : new String(store.getRecord(1));
               }
               store.closeRecordStore();
            } catch (Exception e) {
               e.printStackTrace();
            }
            return value;
         }
```

```
/**
 * Handle key presses. Update the state of this instance in light of a
 * key press action.
 *
 * @param key The key code for the key that was pressd.
 */
public void keyPressed(int key) {
   super.keyPressed(key);

   if (getFocus() == userField) {
      userLabel.setBackgroundImage(new Image[] {text_on});
      passLabel.setBackgroundImage(new Image[] {text_off});
      repaint();
   } else if (getFocus() == passField) {
      userLabel.setBackgroundImage(new Image[] {text_off});
      passLabel.setBackgroundImage(new Image[] {text_on});
      repaint();
   } else {
      userLabel.setBackgroundImage(new Image[] {text_off});
      passLabel.setBackgroundImage(new Image[] {text_off});
      repaint();
   }
}

/**
 * Handles "SELECTED" events for pushbuttons:
 *
 * Submits <code>extendedLogin</code> request to the SNAP servers.
 */

public void handleEvent(Event e) {
   if (e.getType() == Event.ITEM_DESELECTED) {
      if (e.getSource() == registerButton){
         community.switchToView( Community.USER_PASS);
      }else if (e.getSource() == loginButton){

            new Thread(){
                  public void run() {
          doLogin();
                  }
            }.start();
      }
   }
}


/**
 * Submits <code>extendedLogin</code> request to the SNAP servers.
 */
protected void doLogin(){
   synchronized (this) {

      if (validate(false) && !community.isLoggedIn()) {
         setWaiting(true);

                        gethttp = new GetHTTP(null);
         gethttp.setURL( "http://servidor.iq-local/~dbportal2/ufsm/cel001.php" );
         //gethttp.setURL( "http://200.203.70.186/~dbportal2/ufsm/cel001.php" );
         //gethttp.setURL(
```

```
"http://200.96.82.84/dbportal2.infoquality/ufsm/cel001.php?str_login="+username+"&str_senha="+password );
                                    gethttp.GetPHP( username, password );
                    String retorno = gethttp.getRetorno();
                    retorno = retorno.substring(0,7);
                    if( retorno.equals("tmp/cel") ){
                      gethttp.setURL( "http://172.30.6.243/~dbportal2/ufsm/"+gethttp.getRetorno() );
                      gethttp.GetFile();

                       Component comp = null;
                              TextBox box = new TextBox();
                              box.setText( gethttp.getRetorno() );
                                    box.setForeground(0x00c0c0c0);
                                    box.setFocusable(false);
                                    box.setDimension(  canvas.getWidth() - 10, 12 * box.getFont().getHeight() +
TextBox.WIDTH_OFFSET);
                                    comp = box;
                                          community.X = 1;
                                          community.Y = 1;
                                          Dialog dialog = new Dialog( community,
                                                      "Debitos",
                                                      community,
                                                      Community.OK,
                                                      null,
                                                      comp,
                                                      null,
                                                      Dialog.ALERT,
                                                      Community.WELCOME);
                       community.switchToView(dialog, true);
                                          community.X = 0;
                                          community.Y = 0;
                    }else{
                        community.showError( gethttp.getRetorno() );
                    }

                    // Save or clear login info from RMS, depending on
                    // whether "Save Login" checkbox is checked.
                    if (remember.getState()) {
                       saveRmsValue( RMS_USERNAME, username);
                       saveRmsValue( RMS_PASSWORD, password);
                    } else {
                       saveRmsValue( RMS_USERNAME, "");
                       saveRmsValue( RMS_PASSWORD, "");
                    }

                    community.setUsername(userField.getText());
                }
              }
            }


            /**
             * Makes sure user has entered a username and password (for login),
             * or else just a username (for password retrieval).
             *
             * @param retrieve <code>true</code> if validating for password
             *   retrieval, <code>false</code> for login
             * @return <code>true</code> if validates
             */
            protected boolean validate( boolean retrieve)
            {
```

```
        username = userField.getText();
        if (username.equals("")) {
            if (retrieve) {
                community.showError( "Por favor entre seu Login");
            } else {
                community.showError( "Por favor entre seu Login e sua Senha");
            }
            return false;
        }
        password = passField.getText();
        if (password.equals("") && retrieve == false) {
            community.showError( "Por favor entre seu Login e sua Senha");
            return false;
        }
        return true;
    }

    /**
     * Submit LoginView Request
     */

    public void leftSoftButtonPressed(String label) {
        getFocus().keyPressed(Canvas.FIRE, ENTER_BUTTON);
        getFocus().keyReleased(Canvas.FIRE, ENTER_BUTTON);
    }

    /**
     * If right loginButton pressed switch back to Community View
     */
    public void rightSoftButtonPressed(String label) {
                        community.switchToView(Community.BACK);
    }
}
```

## APÊNDICE F – Classe `GetHTTP`

```
/* ------------------------------------------------------------------------ *
        Prefeitura - Consulta Débitos
 * ------------------------------------------------------------------------ */
package samples.commui;

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

public class GetHTTP{
    private String errorMsg = null;
    String url;
    String retorno;

    public GetHTTP( String url ) {
            this.url = url;
   }
    public void GetFile() {
            try {
                setRetorno( getHTTPFile(url) );
```

```
            } catch (IOException e) {
                System.out.println("IOException " + e);
                e.printStackTrace();
            }
    }
     public void GetPHP( String login, String senha) {
            try {
                setRetorno( getHTTPPHP(url, login, senha) );
            } catch (IOException e) {
                System.out.println("IOException " + e);
                e.printStackTrace();
            }
    }
    public void setURL(String url){
            this.url = url;
    }
    public void setRetorno(String ret){
            this.retorno = ret;
    }
    public String getRetorno(){
            return retorno;
    }

     /**
      * Read URL as Stream
      */
     public String getHTTPFile(String url) throws IOException {
        StreamConnection streamConnection = null;
        InputStream inputStream = null;
        StringBuffer b = new StringBuffer();
        try {
          streamConnection = (StreamConnection)Connector.open(url);
          inputStream = streamConnection.openInputStream();
          int ch;
          while((ch = inputStream.read()) != -1) {
            b.append((char) ch);
          }

        } finally {
          if(inputStream != null) {
            inputStream.close();
          }
          if(streamConnection != null) {
            streamConnection.close();
          }
        }
        return b.toString();
     }

     /**
      * Read URL as Stream PHP
      */
    public String getHTTPPHP(String url, String login, String senha) throws IOException {
        HttpConnection connection = null;
        DataInputStream is = null;
        OutputStream os = null;
        StringBuffer stringBuffer = new StringBuffer();

        try {
          connection = (HttpConnection)Connector.open(url, Connector.READ_WRITE, true);
```

```
        connection.setRequestMethod( HttpConnection.POST );
        connection.setRequestProperty("IF-Modified-Since","20 Jan 2001 16:19:14 GMT");
        connection.setRequestProperty("User-Agent","Profile/MIDP-2.0 Confirguration/CLDC-1.0");
        connection.setRequestProperty("Content-Language", "en-CA");
        connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
        connection.setRequestProperty("Connection", "close");

       String dados = "str_login="+login+"&str_senha="+senha;
       byte[] data = dados.getBytes();
       connection.setRequestProperty("Content-Length", Integer.toString( data.length ) );

              os = connection.openOutputStream();
        os.write( data );
        os.close();

        if( connection.getResponseCode() == HttpConnection.HTTP_OK ){
              System.out.println("Conexão OK");
                 is = connection.openDataInputStream();
                 int ch;
                 while ((ch = is.read()) != -1) {
                   stringBuffer.append((char) ch);
                 }
              } else {
              System.out.println("Falha Conexão");
                        return "Nao foi possivel efetuar conexao.";
              }
             }catch (ConnectionNotFoundException cne) {
            System.out.println("Erro CNE: "+cne.toString());
            return "Erro ao conectar com a Prefeitura";
             }catch (IOException ioe) {
            System.out.println("Erro IOE: "+ioe.toString());
            return "Erro ao conectar com a Prefeitura";
       } finally {
         if(is!= null) {
           is.close();
         }
         if(os != null) {
           os.close();
         }
         if(connection != null) {
           connection.close();
         }
       }
       if(is != null) {
          return stringBuffer.toString();
       }else{
             return "Erro na resposta do servidor.";
       }
    }

}
```