BrokerCell: programando celular com J2ME para cotar papéis na Bovespa

Nairon Neri Silva[1], Luís Augusto Mattos Mendes(Orientador)[1]

[1]Ciência da Computação - Universidade Presidente Antônio Carlos (UNIPAC)

naironneri@yahoo.com.br, luisaugustomendes@yahoo.com.br

Resumo. *Este artigo aborda os principais aspectos e conceitos da plataforma Java 2 Micro Edition (J2ME), com foco principal no desenvolvimento para telefones celulares, aplicando os conhecimentos adquiridos no desenvolvimento de uma aplicação J2ME voltada para o acompanhamento econômico da bolsa de valores, o BrokerCell.*

Palavras-chave: *J2ME; Java; Aplicações Móveis; Bolsa de Valores; Bovespa.*

1. Introdução

Inicialmente a telefonia móvel oferecia somente a função de efetuar e receber ligações. Com os constantes avanços do setor, hoje os celulares agregam diversas funcionalidades importantes, tais como: conexão à internet, comunicação por *e-mail*, transferência de arquivos multimídia, instalação de *softwares* e jogos em Java, etc.

A plataforma J2ME torna possível a criação de aplicações voltadas para esses dispositivos, desde celulares até dispositivos embarcados em automóveis. Através do uso de máquina virtual, abstrai-se o *hardware* da aplicação desenvolvida, facilitando a compatibilidade entre os dispositivos de diferentes fabricantes.

Este trabalho tem como objetivo descrever as principais funcionalidades e características da plataforma J2ME e aplicar os conhecimentos no desenvolvimento de um sistema baseado na plataforma J2ME. O sistema consiste em uma aplicação para consulta de cotações das ações de empresas presentes na Bovespa (Bolsa de Valores de São Paulo).

A segunda seção aborda a fundamentação teórica para o desenvolvimento do trabalho, contemplando os principais aspectos da plataforma J2ME. A seção 3 tem como objetivo a aplicação prática dos conceitos vistos na seção 2, através da aplicação *BrokerCell*. Por fim, a seção 4 trata das considerações finais do artigo.

2. Revisão Bibliográfica

Esta seção contempla a fundamentação teórica sobre a plataforma J2ME.

2.1. Visão Geral da Plataforma J2ME

A plataforma J2ME (Java 2 *Micro Edition*) desenvolvida pela *Sun Microsystems* consiste em um conjunto de API's (*Application Programming Interface*) voltadas para o desenvolvimento de aplicações compatíveis com dispositivos móveis, dentre eles destacam-se os *PDA's* e celulares. Uma das principais características desses dispositivos

é o baixo poder de processamento, o que compromete em parte a complexidade do sistema a ser desenvolvido. Vale ressaltar também o baixo poder de energia. Para executar aplicações J2ME, o dispositivo deve possuir uma JVM (Java *Virtual Machine*), que difere da máquina virtual utilizada em outras plataformas Java. A função da JVM é promover comunicação da aplicação com o sistema operacional e consequentemente ao *hardware* do dispositivo.

"Com o J2ME, as aplicações são escritas somente uma vez para um grande número de dispositivos e são baixadas dinamicamente" [1]. Essa portabilidade de código se deve pelo fato da plataforma Java não gerar códigos nativos de um determinado sistema operacional, transformando a sequência de comandos em *bytecodes*, que é o próprio código já compilado. Os *bytecodes* são executados pela JVM, que por sua vez abstrai do programador o sistema operacional e o *hardware* ao qual ela se comunica. Além da portabilidade herdada da plataforma Java, a API do J2ME dispõe de recursos importantes como o armazenamento de dados, comunicação através de protocolos e suporte a interface gráfica.

2.2. Tecnologia J2ME

Além da máquina virtual, a plataforma J2ME é composta por configuração (*configuration*) e perfil (*profile*), conforme mostra a Figura 1.

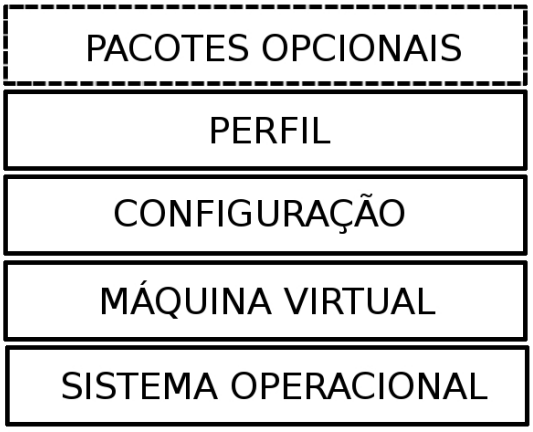

Figura 1. Divisão da plataforma J2ME em camadas

Pode contar também com os pacotes opcionais, que são bibliotecas específicas para oferecer suporte a uma determinada tecnologia, a comunicação por *bluetooth* e SMS (*Short Message Service*) são exemplos de pacotes opcionais.

2.2.1 Configurações (*Configurations*)

Segundo [2], as configurações são conjuntos de bibliotecas que atendem a um determinado conjunto de dispositivos, que apesar de possuírem aplicações distintas, compartilham características semelhantes.

De acordo com [3], a plataforma J2ME é composta por duas configurações: CLDC e CDC, como apresentado na Figura 2 [4].

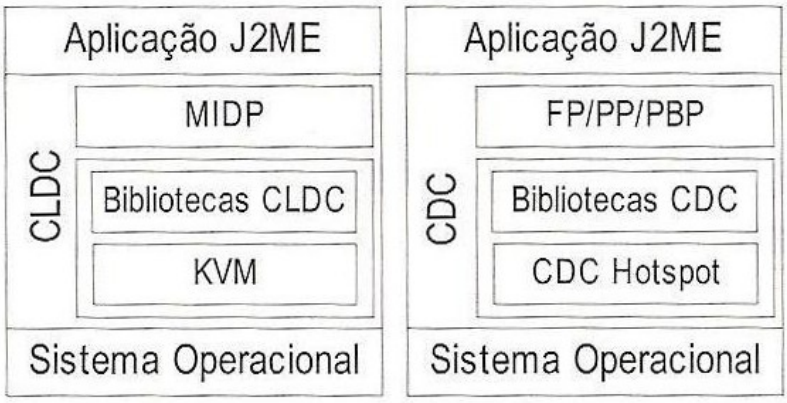

Figura 2. Divisão das Configurações

i) CLDC (*Connected Limited Device Configuration*)

É a configuração mais utilizada, pois abrange um grande número de dispositivos. Esses dispositivos são caracterizados por possuírem baixo poder de processamento, baixa capacidade gráfica e também pouca memória para executar as aplicações. Em contrapartida, esses dispositivos são bem portáteis, dentre eles podemos citar os celulares e os *PDA's*.

De acordo com [4], os processadores utilizados nesses dispositivos são geralmente de 16 ou 32 *Bits* e a memória disponível normalmente é de 160Kb não-volátil e 192Kb para a plataforma Java.

O suporte à comunicação geralmente é feito por conexões sem fio e não contínuas. A máquina virtual utilizada pela configuração CLDC é chamada de KVM (*Kilobyte Virtual Machine*) em analogia ao seu tamanho.

ii) CDC (*Connected Device Configuration*)

Oferece suporte a uma classe de dispositivos com maior poder de processamento que os suportados pela CLDC. Consequentemente, possuem maior memória e capacidade de processamento disponíveis para executar aplicações, “eles devem ter, pelo menos, processadores de 32-*Bit*, 2,5MB de memória não volátil e 2MB de memória RAM” [4] , além de contarem com conexões mais rápidas. Trazem também uma maior parte da plataforma J2SE (Java 2 *Standard Edition*) incorporada. Os *set-top-box* de televisores digitais e os sistemas de navegação de automóveis são exemplos de dispositivos suportados pela configuração CDC. A máquina virtual utilizada pela configuração CDC é chamada de CDC *HotSpot*.

2.2.2 Perfis (*Profiles*)

Consiste de uma extensão da configuração, oferecendo bibliotecas mais específicas de um determinado tipo de dispositivo. Definem o ciclo de vida da aplicação e diversos outras funcionalidades as quais um aplicativo poderá utilizar. Um perfil é sempre destinado a somente uma configuração, porém uma configuração pode compreender diversos perfis.

Dentro da configuração CDC há os seguinte perfis:

- FP (*Foundation Profile*) – É o perfil base da configuração CDC, seu foco é atender dispositivos sem interface com o usuário (*hardware* embutido), por isso não oferece suporte a interface gráfica.
- PBP (*Personal Basis Profile*) – Direcionado a dispositivos que operam com interface gráfica, porém não traz suporte completo ao AWT[1] (*Abstract Window Toolkit*), utilizando *toolkits* específicos. Oferece também as funcionalidades do FP.
- PP (*Personal Profile*) – Além de prover as funcionalidades dos perfis anteriores, agrega duas funcionalidades importantes, que é o suporte à *applets* e o suporte completo ao AWT. Um exemplo de utilização é em consoles de videogames.

Dentro da Configuração CLDC há o perfil MIDP.

> "O MIDP foi feito para celulares e *PDA's*. Ele oferece a base para a funcionalidade requerida pelas aplicações móveis, incluindo interfaces do usuário, conexões com rede, persistência de dados e controle de aplicações. Combinado com o CLDC, o MIDP fornece um ambiente de execução Java completo que alavanca a capacidade de *handhelds* e minimiza o consumo de memória e energia" [1].

2.2.3. *MIDlet*

Em aplicações MIDP, deve-se utilizar a classe abstrata *MIDlet*. O ciclo de vida dos aplicativos é gerenciado pela classe *MIDlet* e os métodos *startApp*(), *pauseApp*() e *destroyApp*() devem ser sobrescritos.

Como observado na Figura 3 [2], após ser criado pelo construtor, o aplicativo fica em estado de espera aguardando a ativação que é realizada através de uma chamada ao método *startApp*() e ficará nesse estado até que seja chamado um dos outros métodos. Se for necessário interromper a execução da aplicação temporariamente, chama-se o método *pauseApp*(), que será responsável por liberar o dispositivo para executar outra tarefa. O método *destroyApp*() é invocado se houver a necessidade de terminar a aplicação, seja por solicitação do usuário ou por algum erro que impeça o aplicativo de continuar sua execução. A transição do estado ativo (*startApp*()) para o estado pausado (*pauseApp*()) ou vice-versa, pode ser realizada quantas vezes for necessário, porém a transição de um desses estados para o estado terminado (*destroyApp*()) somente poderá ser realizada uma vez. Todos os aplicativos MIDP devem implementar no mínimo esses três métodos.

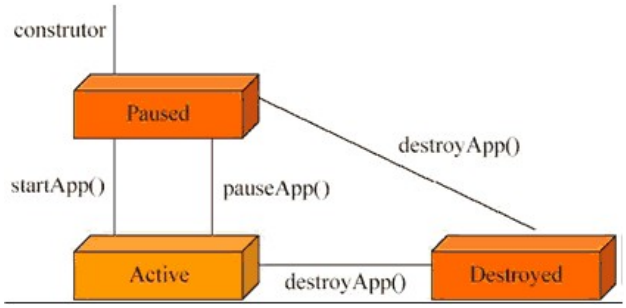

Figura 3. Ciclo de vida de um *MIDlet*

1 De acordo com [5], o AWT é um conjunto de classes voltadas para operações gráficas, porém independentes da plataforma em que será utilizada.

Conforme [6], uma aplicação MIDP deve ter no mínimo uma classe *MIDlet*, nos casos em que possui mais de um *MIDlet* é chamada de *MIDlet Suite*. O *MIDlet Suite* ao ser executado, lista para o usuário todos os *MIDlets* disponíveis e aguarda o usuário solicitar a execução de algum. As aplicações são descritas por um arquivo JAD (Java *Application Descriptor*), que contém informações como a configuração e o perfil. Para ser executado, necessita de um arquivo JAR (Java *Archive*), que é o empacotamento da aplicação.

2.2.4. Comunicação com o usuário

Uma aplicação *MIDlet* é composta por formulários que comunicam com o usuário através de comandos. Há algum tempo atrás, a criação de formulários era um pouco complicada devido a baixa resolução de tela dos dispositivos, atualmente não é um quesito preocupante, já que boa parte dos dispositivos possuem visores de alta resolução.

Apesar das resoluções não representarem problemas atualmente, os equipamentos móveis podem apresentar telas de tamanhos diferentes. Essa divergência é corrigida pela classe *Display*, que possui dentre suas funções a comunicação da aplicação com o *MIDlet*, agregando o benefício da aplicação abstrair o *hardware* do equipamento, conforme é mostrado na Figura 4 [7].

"O *Display* faz a ponte entre as telas da aplicação (*Displayables*) e a aplicação (*MIDlet*). É a classe que recebe os *Displayables*"[7]. Os *Displayables* representam o que se pode exibir na tela do dispositivo. Assim, uma aplicação possui somente uma classe *Display*, mas pode conter diversas classes *Displayables* que são exibidas uma por vez.

A classe *Displayable* possui como generalizações as classes *Screen* e *Canvas*. A classe *Screen* e suas derivadas são responsáveis por montar os principais objetos gráficos de interface com o usuário, como por exemplo, os *forms* e *lists.* Como esses objetos são disponibilizados praticamente no ponto de serem incluídos na aplicação, a classe *Screen* é considerada de alto nível (*High-Level*). Já a classe *Canvas* é utilizada normalmente para criação de telas de desenho e animação para a criação de jogos e aplicações gráficas. Como possui maior poder de manipulação dos objetos, ela é considerada de baixo nível (*Low-Level*).

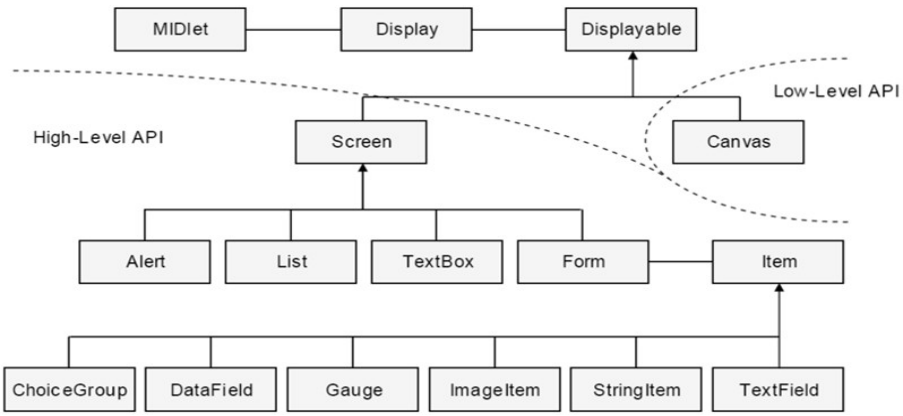

Figura 4. Diagrama de classes de um *Midlet*

A plataforma também inclui a classe *Command* que permite atribuir comandos (funções) aos objetos que compõem a tela do aplicativo.

2.2.5. Conexão e armazenamento

Geralmente em uma aplicação *MIDlet*, a conexão é realizada por meio do protocolo HTTP(*Hypertext Transfer Protocol*), trazendo assim a independência da tecnologia a ser utilizada do lado servidor, pois basta que essa tenha suporte à comunicação por HTTP. O armazenamento de dados é realizado pela classe *RecordStore*, que armazena os dados de uma forma simplificada.

3. Estudo de Caso

Segundo [8], até 2014 o número de celulares com acesso à internet crescerá 600%, assim, as aplicações móveis vem ganhando um notável espaço na área de desenvolvimento de *software*. Esse cenário se deve principalmente pela alta competitividade do mundo globalizado, onde é comum o acesso às informações através desses dispositivos.

Partindo dessa ideia, a proposta do artigo é o desenvolvimento de uma aplicação voltada para a área de acompanhamento de ações na Bolsa de Valores.

3.1. Aplicação *BrokerCell*

A aplicação *BrokerCell* tem como objetivo principal a consulta do valor dos papéis das empresas com ações na Bovespa. Além disso, o sistema também é capaz de realizar cotação e conversão de moedas. A aplicação J2ME realiza o acesso a um w*eb service*[2] que é responsável por buscar as informações e formatá-las para envio e exibição no celular, essas informações são retornadas pelo servidor do *Yahoo Finances*[3] com defasagem de 15 minutos em relação a informação em tempo real. O *Yahoo Finances* foi escolhido pelo fato de disponibilizar as informações da Bovespa já formatadas. Assim, elimina-se a utilização de um *parser* para ler diretamente a página da Bovespa e captar os dados, o que poderia ficar inoperante com a alteração do *layout* da página. Esse intermédio entre a aplicação J2ME e o servidor do *Yahoo Finances* é importante, pois cria uma independência da forma como os dados são disponibilizados pela Bovespa. Assim, se a estrutura das informações sofrer alterações, será necessário somente a alteração no *web service*, a aplicação J2ME continua a comunicar da mesma forma.

A aplicação foi desenvolvida utilizando a Configuração CLDC juntamente com o Perfil MIDP, com o foco principal de utilização em celulares.

3.1.2. Modelagem

A modelagem contempla o lado cliente denominado *BrokerCell* que é uma aplicação J2ME responsável por realizar requisição de informações e o lado servidor que é composto pelo *web service* denominado *WebserviceBroker.*

---

2 *Web services* são aplicações com a capacidade de prover comunicação entre sistemas programados em diferentes linguagens e também sob arquiteturas diferentes, seguindo um determinado padrão de comunicação.

3 Serviço do *Yahoo* para a divulgação de informações do mercado financeiro.

O acesso aos dados é por meio de *web service*, logo as informações entre as aplicações cliente e servidor são trafegadas no formato *XML*[4].

O sistema é dividido basicamente em três componentes: o cliente(*BrokerCell*), o *web service* (*WebServiceBroker*) e o servidor do *Yahoo Finances* (responsável por fornecer as informações da bolsa de valores), como é mostrado na Figura 5.

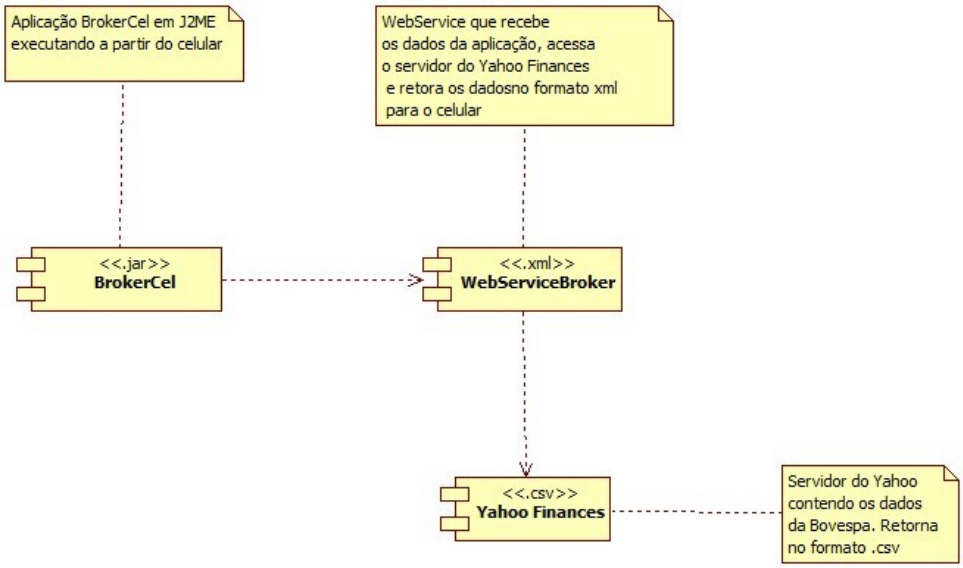

Figura 5. Diagrama de componentes

O sistema provê os seguintes serviços ao usuário:

- Acessar cotação: Permite verificar o valor e demais informações referentes à empresas presentes na Bovespa;
- Índice Bovespa: Retorna o índice em pontos da Bovespa;
- Verificar Valor de Moedas: Permite acessar o valor atual de uma determinada moeda perante o real;
- Conversão de Moedas: Serviço destinado à realização de conversão de valores entre moedas, baseado no valor atual de mercado;

O diagrama de classes da Figura 6 apresenta os principais métodos e atributos do lado cliente, que conta com a classe Principal composta pelos formulários da aplicação que estende da classe *MIDlet*, com a classe Conexao responsável pelo acesso ao *web service* e as classes Cotacao e Moeda que são responsáveis pelo escopo dos objetos manipulados pelo sistema.

4 "*Extensible Markup Language* (XML) é linguagem de marcação de dados (*meta-markup language*) que provê um formato para descrever dados estruturados." [9]

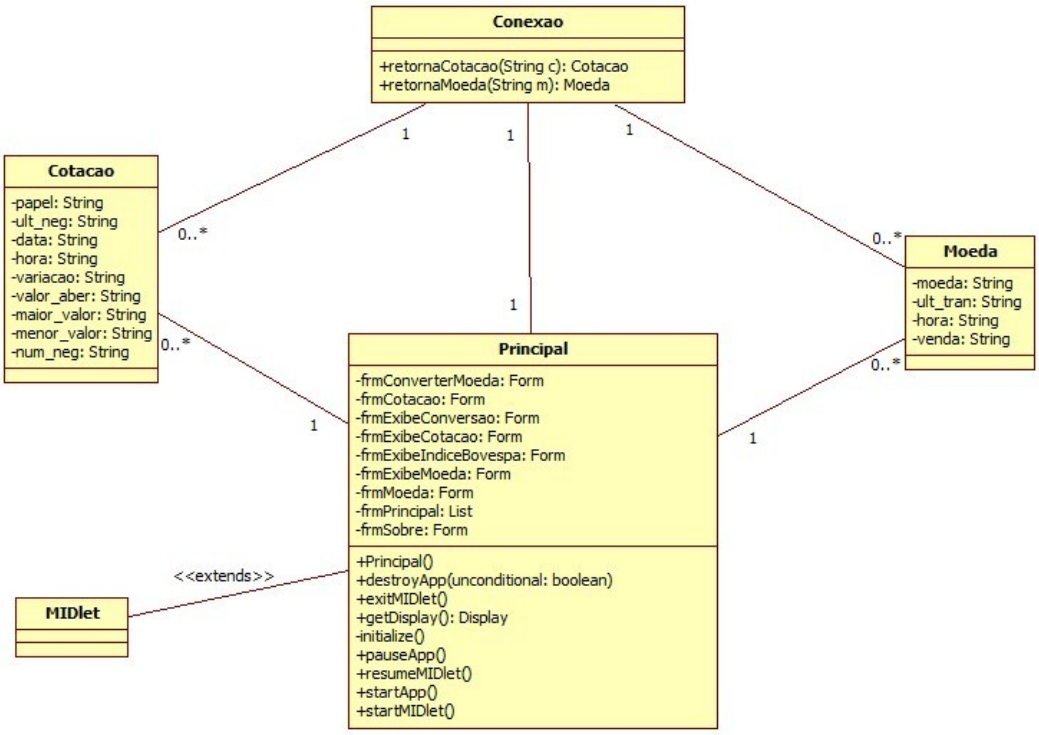

Figura 6. Diagrama de classe do cliente J2ME

O lado servidor, conforme mostra o diagrama de classe da Figura 7, é composto pelas classes Cotacao e Moeda que possuem as mesmas funções do lado cliente, pelas classes retornaCotacao e retornaMoeda responsáveis por acessar o servidor do *Yahoo Finances*, recuperar as informações no formato de arquivo *.CSV*[5], formatar e retornar os dados para a aplicação cliente no formato *XML*.

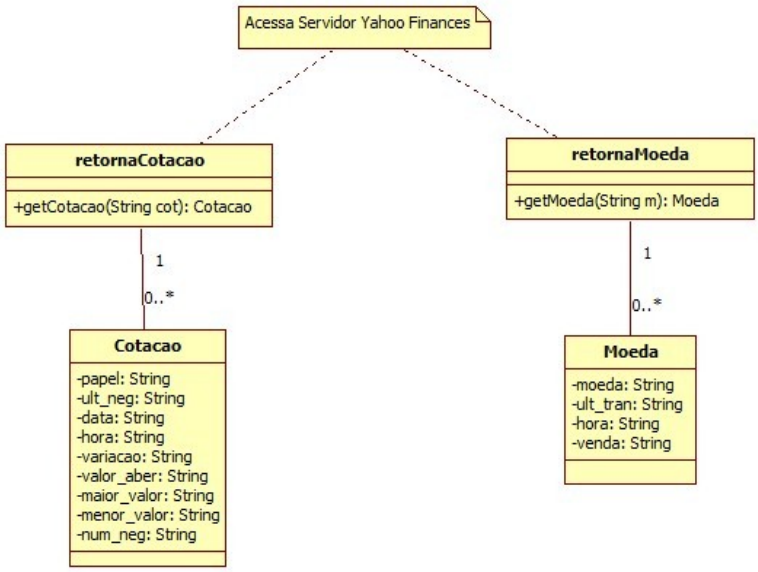

Figura 7. Diagrama de classe do *web service*

5 "*Comma-separated values* (ou CSV) é um formato de arquivo que armazena dados tabelados, cujo grande uso data da época dos *mainframes*." [10].

A sequência de operações do sistema para a funcionalidade de obter cotação de um determinado papel[6] e também para obter o índice da Bovespa é apresentado na Figura 8.

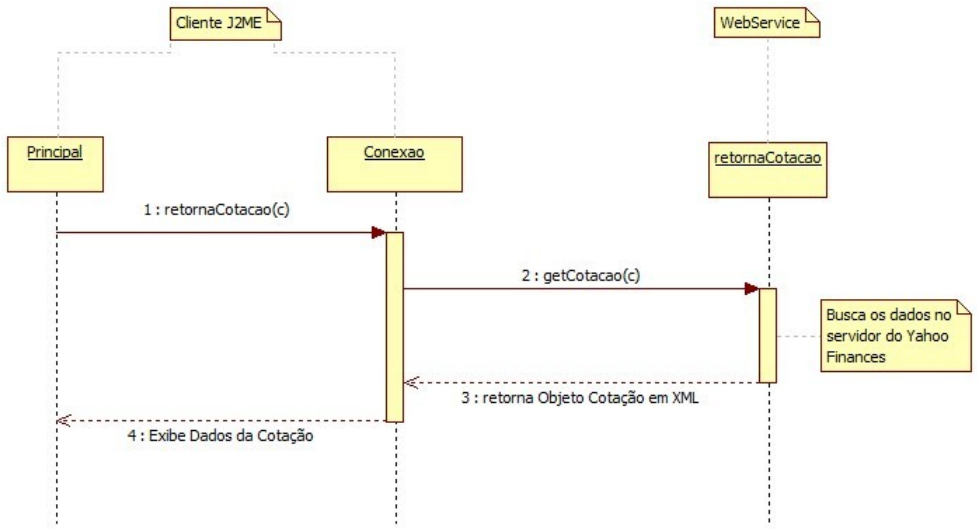

Figura 8. Diagrama de sequência para obter cotação de papéis

A sequência de operações do sistema para obter a cotação de moedas e realizar a conversão entre moedas é apresentado na Figura 9.

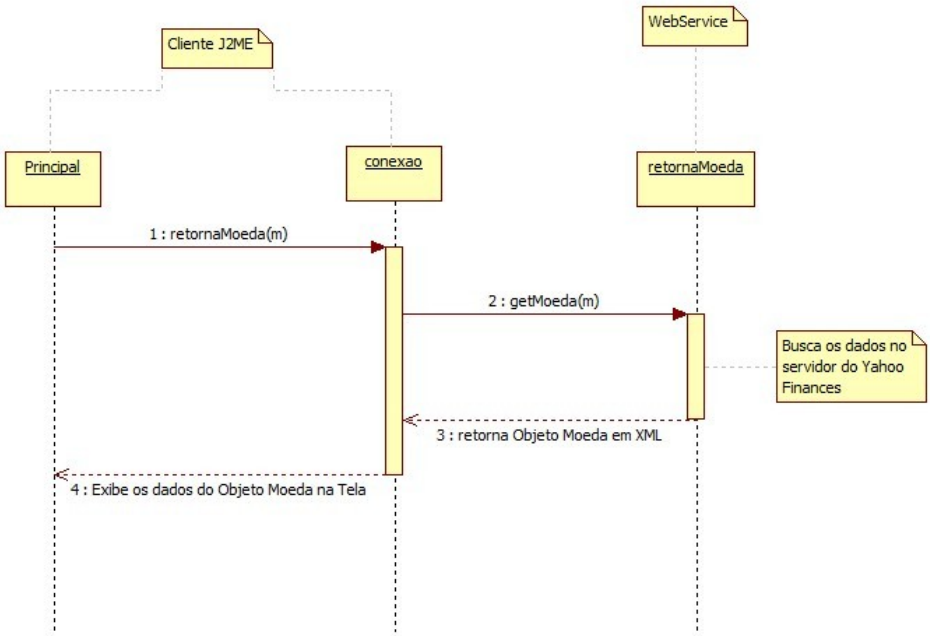

Figura 9. Diagrama de sequência para obter cotação de moedas

3.1.3 Implementação

A implementação da aplicação proposta se divide em duas partes, cliente e servidor.

---

6 Ações da Bolsa de Valores de São Paulo.

3.1.3.1. Cliente

O desenvolvimento da aplicação utiliza o *IDE (Integrated Development Environment) Netbeans* com o *Mobility Pack*, que consiste em um ambiente de desenvolvimento de código integrado com recursos de compilação, testes e empacotamento.

Como a plataforma J2ME não suporta nativamente a comunicação com *web service*, a aplicação *BrokerCell* utiliza as classes do projeto *Ksoap2*[7] para esse fim.

A classe Principal é a que deriva da classe *MIDlet* e comporta todos os métodos, *forms, alerts, lists e commands* da aplicação J2ME, conforme pode ser observado no diagrama da Figura 10 gerado pelo *Netbeans*.

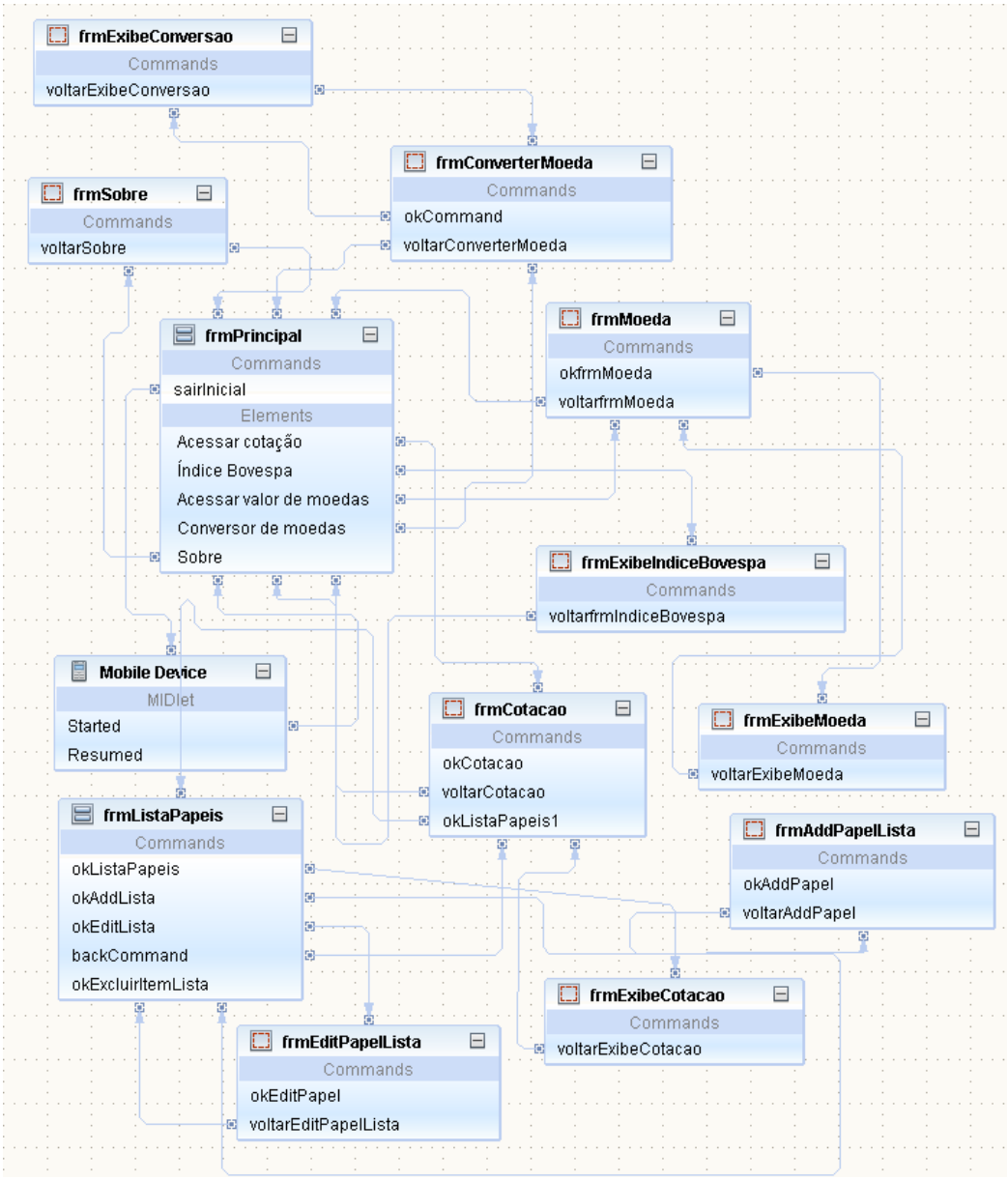

Figura 10. Diagrama interno da classe Principal

7 Projeto destinado a ambientes limitados de desenvolvimento com o intuito de possibilitar a comunicação com *web service.*

Ao ser compilada, a aplicação gera na raiz do projeto uma pasta denominada “dist” que contém um arquivo com a extensão .jar, esse arquivo é o “executável” da aplicação J2ME e deve ser enviado e instalado no dispositivo móvel. Vale ressaltar, que o dispositivo deve contar com algum tipo de conexão de dados e a aplicação possuir direito de acesso à internet, para conseguir se comunicar com o servidor remoto.

A classe Principal da aplicação J2ME não realiza a comunicação diretamente com o *web service*, toda a estrutura de conexão é de responsabilidade da classe Conexão.

3.1.3.2. Servidor

A aplicação do servidor, *WebServiceBroker*, desenvolvida em Java, é executada no servidor de aplicação *Tomcat*[8] com suporte ao *Axis2*[9] que agrega a funcionalidade de *web service* ao servidor.

As classes retornaCotacao e retornaMoeda são as responsáveis por prover os serviços do *web service*. Ao receber uma requisição da aplicação J2ME, a aplicação servidora busca os dados no *Yahoo Finances* que estão no formato .CSV, instancia um objeto e o retorna para a aplicação J2ME em formato *XML*.

3.1.4. Utilização do *BrokerCell*

A utilização do *BrokerCell* é bastante simples, basta que o usuário possua conhecimentos do mercado financeiro, como o código dos papéis da Bovespa e moedas.

A Figura 11 apresenta a tela inicial do sistema, com todos os serviços disponíveis ao usuário.

Figura 11. Tela inicial da aplicação

8 O *Tomcat* é um servidor web que executa aplicações desenvolvidas em Java.

9 Projeto destinado a incorporar ao servidor *Tomcat* as funcionalidades de *web service*, inclui painel de administração e visualização dos serviços instalados.

A Figura 12 (a) mostra a tela de cotação de papéis, onde é inserido o código do papel e retorna os dados solicitados conforme mostra a Figura 12 (b). O índice da Bovespa é retornado da mesma forma que uma cotação de papel, com a diferença de não apresentar o número de negócios.

A inclusão do sufixo ".sa" no código do papel é necessária para que o servidor do *Yahoo Finances* identifique que se trata do mercado de São Paulo (Bovespa). Assim, a aplicação pode acessar papéis de outros mercados, basta a inclusão do sufixo que identifique o mercado, por exemplo, os códigos da bolsa de Buenos Aires possui sufixo ".ba".

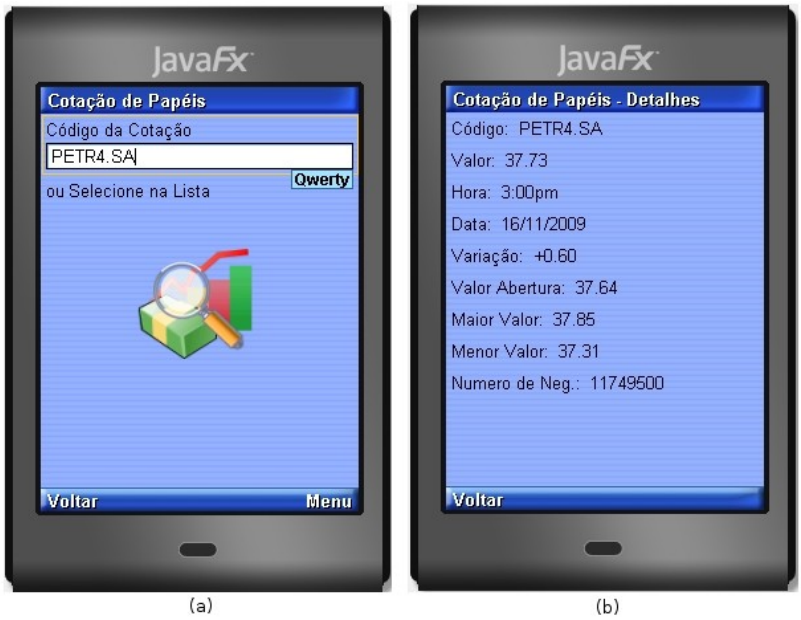

Figura 12. Cotação de papéis

Também é possível criar uma lista de papéis, contendo os códigos para futuras consultas sem a necessidade de memorizá-los, conforme apresentado na Figura 13 (a). A Figura 13 (b) mostra a tela para a manipulação dos códigos armazenados.

Figura 13. Lista de papéis salvos

A Figura 14 (a) apresenta a tela para a inserção do código da moeda que é retornado conforme a Figura 14 (b).

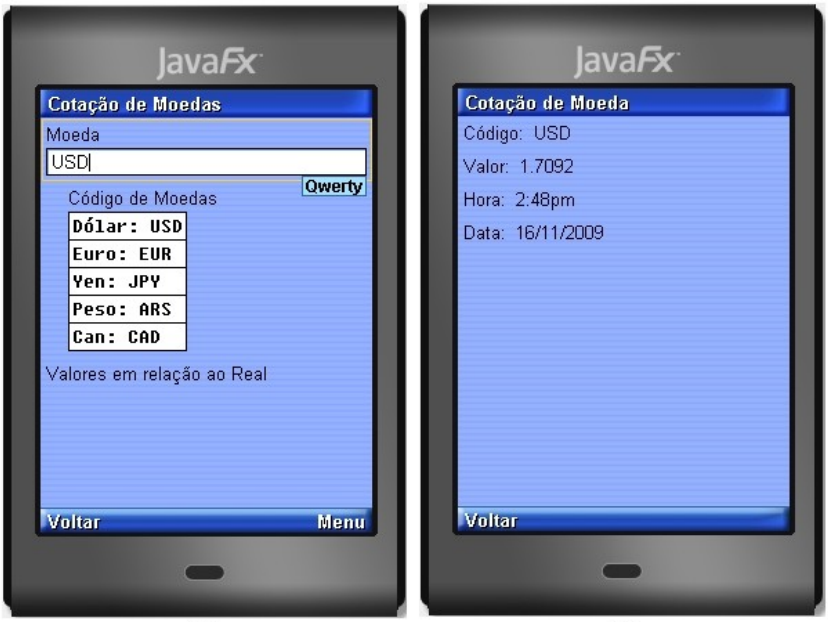

Figura 14. Cotação de moedas

A Figura 15 (a) mostra a tela de inserção de dados para a conversão de moedas, que são retornados conforme apresentado na Figura 15 (b).

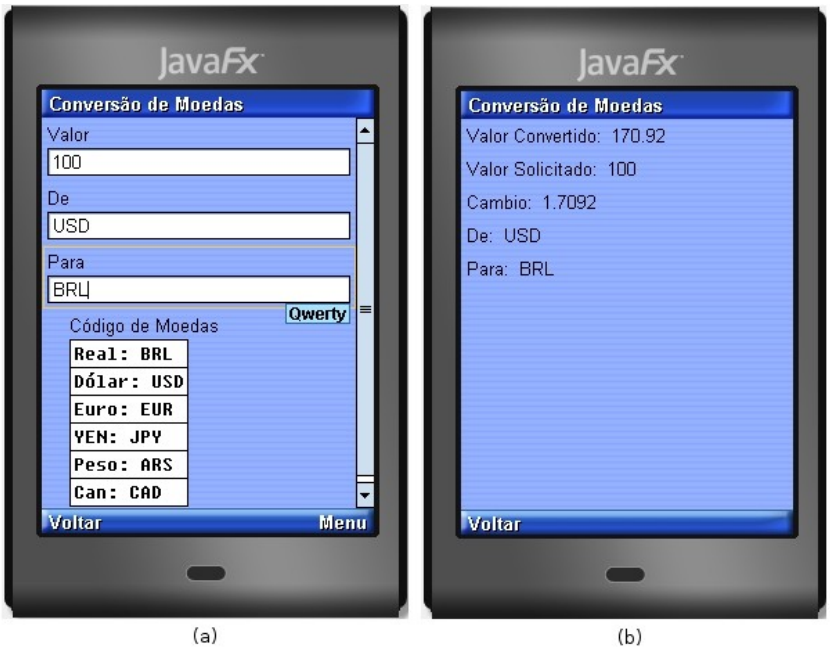

Figura 15. Conversão de moedas

4. Considerações Finais

A partir da revisão bibliográfica e com a aplicação do estudo de caso, conclui-se que através da plataforma J2ME é possível criar ferramentas estáveis e de grande portabilidade, visto que é necessário somente uma máquina virtual compatível. No que

se refere ao desenvolvimento, a integração com o *Netbeans* é de suma importância, pois além de descomplicar a codificação, funciona perfeitamente com o emulador de aplicações J2ME.

De acordo com [11], o mercado mundial de aplicativos móveis deverá crescer em média 102% ao ano até 2012. Assim, a tecnologia J2ME se encaixa perfeitamente nesse mercado, conseguindo levar até os dispositivos móveis uma das plataformas de desenvolvimento mais utilizadas no mundo, a plataforma Java.

Dentro desse contexto, a aplicação *BrokerCell* pode ser utilizada por diversos segmentos da área econômica, tanto pelos corretores de ações como pelos pequenos investidores, facilitando o acesso aos dados em qualquer lugar, desde que o celular possua uma conexão de dados.

Um dos problemas apresentados durante os testes da aplicação foi o congestionamento da rede de dados do celular, dificultando a comunicação da aplicação com o servidor.

Uma sugestão para futuros trabalhos é a implementação de outros tipos de informações da Bovespa, como o histórico de cotações e fundos imobiliários.

5. Referências Bibliográficas

[1] PINHEIRO, Christiano. **J2ME - Java para os portáteis**. Disponível em: <http://imasters.uol.com.br/artigo/1539/java/j2me_-_java_para_os_portateis/>. Acessado em 13 de março de 2009.

[2] ANDREÃO, Cristiano Fioresi. **Conceitos Básicos das Plataformas Java e J2ME**. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=6484>. Acessado em 13 de março de 2009.

[3] SUN Microsystems. **Java ME Platform Overview.** Disponível em:<http://java.sun.com/javame/technology/index.jsp>.Acessado em 3 de abril de 2009.

[4] JOHNSON, Thienne M.. **Java para Dispositivos Móveis**. 1 ed. São Paulo: Novatec, 2007. 336 p.

[5] TRINDADE, Cristiano. **Uma visão geral sobre AWT.** Disponível em: <http://imasters.uol.com.br/artigo/468/java/uma_visao_geral_sobre_awt/>. Acessado em 7 de dezembro de 2009.

[6] ROSA, Fabiano. **Protótipo de um diário de classe em dispositivos móveis utilizando J2ME**. 2005. 100f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

[7] MOVEL, Java. **Interface com o usuário: Display, Displayable e Command.** Disponível em: <http://www.javamovel.com/2009/05/interface-com-o-usuario-display.html>. Acessado em 26 de setembro de 2009.

[8] OLHAR DIGITAL. **Celulares com Internet: mais 600% até 2014**. Disponível em: <http://olhardigital.uol.com.br/digital_news/noticia.php?id_conteudo=9736>. Acessado em 7 de dezembro de 2009.

[9] GRONER, Loiane. **Introdução a XML**. Disponível em: <https://www.ibm.com/developerworks/mydeveloperworks/blogs/loiane/entry/introdu_c3_a7_c3_a3o_a_xml?lang=en>. Acessado em 7 de dezembro de 2009.

[10] ROGÉRIO, Pedro. **Ler arquivos CSV com jQuery**. Disponível em: <http://www.pinceladasdaweb.com.br/blog/2008/12/05/ler-arquivos-csv-com-jquery>. Acessado em 7 de dezembro de 2009.

[11] SPOSITO, Rosa. **Aplicações móveis devem crescer 102% ao ano**. Disponível em: <http://info.abril.com.br/aberto/infonews/102007/29102007-13.shl>. Acessado em 7 de dezembro de 2009.