UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIAS, GERÊNCIA E SEGURANÇA
DE REDES DE COMPUTADORES

ADILSON SILVEIRA DE SOUZA

Utilização da Tecnologia J2ME para Desenvolvimento de Sistemas de M-Banking

Trabalho de Conclusão apresentado como requisito parcial para a obtenção do grau de Especialista

Prof. Dr. Raul Fernando Weber
Orientador

Prof. Dr. Sérgio Luis Cechin
Prof. Dr. Luciano Paschoal Gaspary
Coordenadores do Curso

Porto Alegre, dezembro de 2008.

AGRADECIMENTOS

Inicialmente agradeço a Deus que me deu forças e condições de superar mais esse desafio em minha vida. Ao grande amor de minha vida Denise que me apoiou e incentivou durante estes dois anos. Aos meus filhos Grazielle, Matheus e Laura que foram privados de minha presença durante vários fins-de-semana e mesmo assim me receberam com sorrisos e carinhos revigorantes. Que minha atitude sirva de exemplo para suas vidas e os incentivem a estudar em busca de tranqüilidade futura.

Aos meus pais Rosa e Viana, cujo amor e carinho resultaram em minha existência. Á Adriana e Alexandre, meus irmãos amados e companheiros de jornada. Aos meus cunhados Juliano e Luciana, meus sobrinhos queridos Rafaela, Lucca, Juliana e Adriano. Aos meus grandes amigos Torres, Caon, Maurício, Cristina, que de uma maneira ou outra me incentivaram nesse desafio. Ao meu grande amigo Vinicius Ribeiro que em 2004 me incentivou a estudar e desenvolver os conhecimentos que possuo de Java.

Agradeço aos meus colegas de aula, em especial ao grande Vanderlei Pollon (“Pollanski” para íntimos), pois juntos conseguimos realizar um grande trabalho durante o curso.

Com muito reconhecimento e respeito, quero agradecer ao professor Weber, que me auxiliou no desenvolvimento desta monografia.

Por fim, sou imensamente grato a todos que contribuíram de alguma forma para que eu conseguisse lograr êxito nesse novo desafio de minha vida.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

3DES	*Triple Data Encryption Standart*
3G	*Third Generation*
ABNT	Associação Brasileira de Normas Técnicas
API	*Application Program Interface*
ANATEL	Agência Nacional de Telecomunicações
B2B	*Business-to-Business*
B2C	*Business-to-Consumer*
B2E	*Business-to-Employee*
C2C	*Consumer-to-Consumer*
CE	*Community Edition*
CLDC	*Connected Limited Device Configuration*
CORBA	*Common Object Request Broker Architecture*
CPU	*Central Processing Unit*
DA	*Discovery Application*
DES	*Data Encryption Standart*
DH	*Diffie-Hellman*
ECC	*Elliptic Curve Cryptography*
GCF	*Generic Connection Framework*
HTTP	*Hyper Text Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
IDE	*Integrated Development Environment*
IDEA	*International Data Encryption Algorithm*
IEC	*International Engineering Consortium*
IETF	*Internet Enginering Task Force*

ISO	*International Organization for Standardization*
J2EE	*Java 2 Enterprise Edition*
J2ME	*Java 2 Micro Edition*
J2SE	*Java 2 Standart Edition*
JAM	*Java Application Manager*
JCA	*Java Criptography Architecture*
JCE	*Java Criptography Extension*
JCP	*Java Community Process*
JDBC	*Java DataBase Conectivity*
JSP	*Java Server Page*
JVM	*Java Virtual Machine*
KVM	*Kilobyte Virtual Machine*
MAC	*Message Authentication Code*
Mbps	Mega Bits por Segundo
MD4	*Message Digest 4*
MD5	*Message Digest 5*
MIDP	*Mobile Information Device Profile*
OSI	*Open Systems Interconnection*
OTA	*Over the Air Provisioning*
PDA	*Personal Digital Assistant*
RC4	*Rivest Cipher 4*
RC5	*Rivest Cipher 5*
RMI	*Remote Method Interface*
RSA	*Rivest Shamir Adelman*
SHA-1	*Secure Hash Algorithm - 1*
SGDB	Sistema Gerenciador de Banco de Dados
SOAP	*Simple Object Access Protocol*
SSL	*Secure Socket Layer*
TCC	Trabalho de Conclusão de Curso
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*

TI	Tecnologia da Informação
WTK20	*J2ME Wireless ToolKit 2.0*
XML	*Extensible Markup Language*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com o amadurecimento da tecnologia Java para dispositivos móveis, a universalização do uso de telefones celulares e a crescente utilização desses equipamentos como meio de acesso à Internet, surgem às condições necessárias para o crescimento do comércio móvel (*M-Commerce*). Elementos de segurança e respeito às limitações físicas desses equipamentos (tamanho de tela, baterias e entrada de dados) são fatores fundamentais para o sucesso dessa modalidade de comércio eletrônico.

O objetivo desse trabalho de conclusão é mostrar a potencialidade da tecnologia *Java 2 Micro Edition* utilizando como base o sistema de *M-Banking* desenvolvido pelo autor. São estudados e discutidos os principais aspectos da tecnologia J2ME, no que tange a facilidade de desenvolvimento bem como os aspectos de segurança que norteiam esta tecnologia. Além disso, são mostradas as soluções desenvolvidas pelo autor (autenticação e controle de sessão) em seu protótipo *M-Banking.*

Palavras-Chave: Dispositivos Móveis, J2ME, CLDC, MIDP, M-Banking, Segurança.

J2ME use of technology for Development of M-Banking Systems

ABSTRACT

With the maturing of the Java technology for mobile devices, the universalization of the use of mobile phones and the increasing use of such equipment as a means of Internet access, the necessary conditions for the growth of mobile commerce (M-Commerce) appears. Elements of security and respect for the physical limitations of the equipment (size of screen, battery and data entry) are key factors for the success of this type of e-commerce.

The purpose of this work is to show the potentiality of Java 2 Micro Edition using as a base the M-Banking system developed by the author. The main features of the J2ME technology are studied and discussed, as well as its ease of development and the security aspects that guide this technology. Moreover, the solutions developed by the author (authentication and session control) are shown in his prototype M-Banking.

Keywords: Mobile Devices, J2ME, CLDC, MIDP, M-Banking, Security.

1 INTRODUÇÃO

A consolidação da Internet como uma das maiores revoluções tecnológicas da humanidade, levaram ao que Lessa (2001) denominou de "Nova Economia". A possibilidade de acessar conteúdos em qualquer parte do mundo com os mais diversos objetivos (cultura, lazer, negócios, etc) foi o "empurrão" necessário para o sucesso dessa nova revolução.

Este conglomerado de equipamentos, programas e conteúdos é utilizado, conforme dados da Internet World Stats (2008), por cerca de 1,4 bilhão de pessoas no mundo todo. Nesta estatística o Brasil aparece com cerca de 50 milhões de usuários, tornando-se assim o país líder em número de usuários na América Latina e 10ª posição no rank mundial.

O montante expressivo de usuários criou inúmeras oportunidades de negócios. Essa nova modalidade de comércio foi denominada de "*E-Commerce*" e é explorada por inúmeras empresas no mundo todo, tais como: Amazon, Submarino, Lojas Americanas, entre outras. As vendas a varejo online movimentaram cerca de 6.4 bilhões de reais em 2007 sem considerar a venda de automóveis, passagens aéreas e leilões online, segundo dados do site *E-CommerceOrg* (2008).

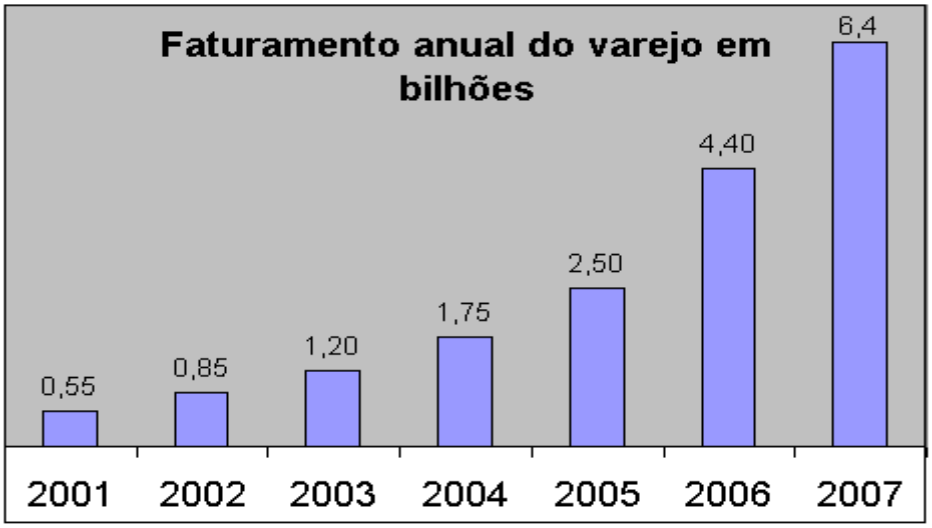

Figura 1.1: Evolução do Varejo Online

E-CommerceOrg (2008).

Este novo tipo de comércio pode ser categorizado segundo Ayres (2008), da seguinte forma:

- Business-to-Consumer (B2C): comércio eletrônico que relaciona de forma direta as empresas e seus consumidores, alguns autores relacionam este tipo de comércio como varejo eletrônico;
- Business-to-Business (B2B): comercio eletrônico de produtos e serviços entre empresas;
- Business-to-Employee (B2E): relação de negócios entre as empresas e seus funcionários;
- Consumer-to-Consumer (C2C): este tipo de comércio refere-se as transações que são efetuadas entre os consumidores;

Outro setor que obteve um crescimento gigantesco foi o de dispositivos móveis, sobretudo o de telefonia celular. Já são mais de 133 milhões de linhas móveis no Brasil, conforme dados da agência reguladora do setor a ANATEL (2008).

As grandes Empresas de tecnologia do setor de dispositivos móveis têm investido milhares de dólares no desenvolvimento de novos equipamentos, mas estes equipamentos continuam a ser caracterizados por suas limitações de processamento, memória e energia. Apesar das melhoras tecnológicas nestes dispositivos, ainda são incomparáveis aos recursos de servidores e alguns tipos de *desktops*.

As maiores restrições dos dispositivos móveis são as seguintes (GUPTA, 2002);

– poder de processamento, memória e energia limitadas;

– limitações para entrada de dados;

– sensibilidade às variações das condições da rede *wireless*; e,

– tamanho e resolução das telas;

Mesmo com as restrições apontadas, a evolução da tecnologia dos equipamentos celulares e de telecomunicações (comunicação 3G com velocidades em torno de 2 Mbps) alavancaram o crescimento da computação móvel através de dispositivos de mão (telefones celulares, *pagers* e *Personal Digital Assistants* - PDAs). Esta tecnologia permite que os usuários tenham acesso permanente à rede, independentemente de sua localização física. Dentro deste contexto, surge uma nova modalidade de comércio eletrônico: o *Mobile-Commerce.*

O *Mobile-Commerce* ou *M-Commerce* é habilidade de negociar bens e serviços em qualquer lugar através de uma rede *wireless* e a Internet (BRIAN, 2003). O *M-Commerce* pode entender-se como a combinação natural que resulta da evolução das comunicações móveis, capazes de transpor progressivamente para o mundo *wireless*, produtos, serviços e aplicações da Internet e levando à criação de outros serviços e produtos que resultarão das características e potencialidades únicas das comunicações *wireless*.

O *M-Commerce* explora a possibilidade de seus usuários executarem operações comerciais em qualquer lugar e a qualquer momento, fornecendo assim grande liberdade aos mesmos. Isso implica no alargamento das fronteiras de acesso à informação, não limitando-se mais aos domínios do escritório ou de casa. É evidente que o *M-Commerce* sofre as conseqüências da estrutura de rede *wireless* existente, mas este é um fator que usuários de telefones celulares já estão acostumados e deve ser melhorado com o avanço da infraestrutura de comunicação. A perda da conexão com a rede *wireless* acaba por definir um limite para o *M-Commerce*, pois aplicações que lidam com riscos de vida não são apropriadas para esta modalidade de comércio eletrônico (NORRIS, 2001).

É de domínio público que a Internet não é um meio seguro para execução de operações que necessitem de privacidade e garantia de integridade dos dados. Dentro deste contexto, o aspecto segurança ganha importância vital para o sucesso do comércio e atividades financeiras.

A comunicação fim-a-fim entre o Cliente e o Servidor de Conteúdos torna-se um fator importante para o sucesso do *M-Commerce*. Uma alternativa para solução do problema de comunicação fim-a-fim sem comprometimento da usabilidade do dispositivo pode ser obtida com a utilização da tecnologia J2ME.

1.1 Objetivos Propostos

Garantir a segurança fim-a-fim entre o dispositivo móvel e o Servidor de Conteúdo é o principal aspecto a ser considerado quando do desenvolvimento de soluções de *M-Commerce*. Sem este preceito básico, qualquer projeto que pretenda utilizar dispositivos móveis para realização de compras e movimentações financeiras enfrentará sérios problemas de credibilidade com o usuário final.

A idéia do protótipo desenvolvido é trazer para o mundo dos celulares os preceitos de segurança já utilizados na comunicação entre equipamentos de médio porte (notebooks e microcomputadores) e Servidores de Conteúdo, entre os quais: autenticação e utilização do protocolo SSL.

Para elaboração desse trabalho, foram pesquisadas as principais normas e dispositivos legais que dessem embasamento teórico a necessidade de estabelecimento de requisitos mínimos de segurança para um sistema de *M-Banking.* Entre estas normas destacamos as seguintes:

- Lei do Sigilo Bancário;
- Resolução 3380 do Banco Central;
- ABNT NBR ISO/IEC 17799:2005: é uma norma da ABNT que serve como referência para criação e implementação de praticas de segurança em TI. Além disso inclui as atividades de estabelecimento de políticas, diretrizes, procedimentos e controles de ambientes de TI.

- ABNT NBR ISO/IEC 27001:2006: é a norma de certificação para Sistemas de Gestão da Segurança da Informação, editada em português em abril de 2006;

Todas as normas a cima abordam de uma maneira geral questões fundamentais sobre sigilo bancário, risco operacional e normas segurança que devem ser observadas por ambientes de TI. O item 10.9 da NBR 17799 fala explicitamente sobre os serviços de comércio eletrônico e suas transações online. Segundo essa norma, devem ser observadas as seguintes diretrizes de segurança em ambientes de TI (ABNT[1], 2005):

- Uso de assinaturas eletrônicas para cada uma das partes envolvidas na transação;

- Todos os aspectos da transação, ou seja, garantido que as credenciais de usuário para todas as partes são válidas e verificadas, a transação permaneça confidencial e a privacidade de todas as partes envolvidas seja mantida;

- Caminho de comunicação entre todas as partes envolvidas é seguro;

- Garantir que o armazenamento dos detalhes da transação está localizado fora de qualquer ambiente publicamente acessível e não retida e exposta em um dispositivo de armazenamento diretamente acessível pela internet;

- Onde uma autoridade confiável é utilizada (para propósitos de emissão e manutenção de assinaturas e/ou certificados digitais), segurança é integrada a todo o processo de gerenciamento de certificados/assinaturas.

Este TCC utilizará como base de estudo a tecnologia J2ME e todos os dispositivos de segurança oferecidos pela mesma para preenchimento dos pré-requisitos de segurança estabelecidos anteriormente.

1.1.1 Objetivo Geral

O objetivo geral deste projeto consiste em demonstrar o potencial da tecnologia J2ME e de suas respectivas facilidades, para desenvolvimento e implementação de sistemas para troca de informações entre dispositivos móveis e Servidores de Conteúdo da Internet com bom nível de segurança.

1.1.2 Objetivos Específicos

São objetivos específicos deste projeto de Trabalho de Conclusão:

- Estudar os aspectos importantes da computação móvel relacionados à aplicação, como: mobilidade, condições ambientais, energia limitada, bem como o relacionamento desta nova tecnologia com a Internet.
- Estudar a arquitetura, o funcionamento e os aspectos relacionados à segurança da tecnologia J2ME;
- Consolidar conhecimentos sobre criptografia, algorítmos de chave pública e privada, certificados, SSL, TLS, HTTPS;

62

[1] Abreviatura de Associação Brasileira de Normas Técnicas

- Consolidar conhecimentos sobre a construção de sistemas baseados na Web;
- Consolidar conhecimentos sobre as tecnologias Java, Banco de Dados e Segurança de Redes.
- Mostrar a implementação de um protótipo de M-Banking desenvolvido pelo autor da monografia, utilizando a tecnologia J2ME

1.2 Estrutura do Trabalho

Este Trabalho de Conclusão está organizado em cinco capítulos. No presente capítulo é feita uma abordagem geral do tema proposto e dos aspectos essenciais para elaboração do mesmo. O segundo capítulo aborda a tecnologia J2ME no contexto das tecnologias Java, conceituação e análise dos principais componentes utilizados no protótipo (*Connected Limited Device Configuration* e *Mobile Information Device Profile*) e exame das características básicas da unidade de desenvolvimento J2ME, as *Midlets*. No terceiro capítulo, é feita uma introdução sobre conceitos de segurança e são mostradas as principais características de segurança fornecidas pela tecnologia J2ME. No quarto capítulo são apresentados os aspectos gerais, a modelagem e os aspectos de segurança envolvidos na construção do protótipo de *M-Banking* desenvolvido pelo autor. No quinto capítulo são apresentas as conclusões e um apanhado geral das dificuldades encontradas, conhecimentos adquiridos e sugestões para trabalhos futuros. Ao final, é apresentada a bibliografia utilizada.

2 JAVA 2 MICRO EDITION

A plataforma Java2 foi divida pela *Sun MicroSystems* em três arquiteturas de desenvolvimento:

- *Java2 Standart Edition* (J2SE), destinada para soluções desenvolvidas para desktops;

- *Java2 Enterprise Edition* (J2EE) para aplicações em ambientes corporativos, e;

- *Java2 Micro Edition* (J2ME), destinada para aplicações em dispositivos móveis, tais como: *Pagers*, PDAs e telefones celulares.

A plataforma J2ME implementa a linguagem Java para dispositivos com poucos recursos de memória e processamento (JANSEN, 2003). A figura 2.1 mostra a disposição das plataformas Java.

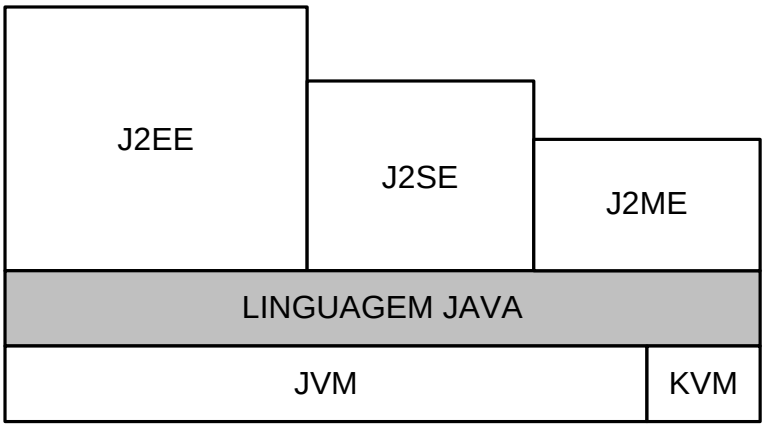

Figura 2.1: Plataforma Java

O J2ME herdou as características mais importantes da plataforma Java, que são:

– *"write once, run anywhere"* – aplicações J2ME podem executar, sem modificações de código, em qualquer dispositivo móvel de uma mesma classe;

– robustez e Segurança – as aplicações Java executam em um ambiente restrito (máquina virtual Java), protegendo os recursos do sistema. Além disso, existem

diversas bibliotecas de criptografia, autenticação e autorização que seguem padrões de mercado. A linguagem Java é robusta, pois os *bytecodes* Java (instruções que podem ser interpretadas por uma máquina virtual Java) passam por verificação, antes de serem executados; e,

– conectividade – o J2ME fornece uma interface simples para aplicações que necessitem ter acesso à rede.

Além dessas vantagens, o J2ME possibilita alta produtividade na construção de sistemas na forma de *Applications Program Interfaces* (APIs) padronizadas e vasta documentação.

2.1 Estrutura do J2ME

Em Keogh (2003), a arquitetura do J2ME está dividida em configurações e *profiles*. Uma configuração é projetada para um tipo específico de dispositivo, tendo como base suas restrições de memória e poder de processamento, ou seja, define um conjunto mínimo de características presentes em uma mesma família de dispositivos. Uma configuração define as facilidades disponíveis em nível de sistema, tais como o conjunto mínimo de bibliotecas Java suportadas e o tipo e as características da máquina virtual presentes no dispositivo. O *profile* especifica o conjunto de características em nível de aplicação para uma determinada classe de dispositivos dentro de uma configuração. Abordaremos no presente TCC a configuração CLDC e o profile MIDP que são específicos para telefones celulares.

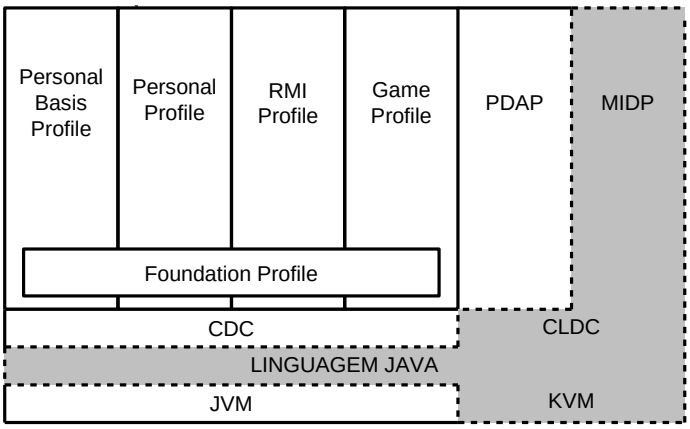

Figura 2.2: Plataforma J2ME

2.1.1 Connected Limited Device Configuration

A configuração CLDC foi desenhada para pequenos dispositivos móveis com uma conexão de rede limitada e memória total disponível entre 160 Kb e 512Kb. Além destes aspectos, Knudsen (2003) menciona ainda os aspectos relacionados a restrições de energia e interface como fatores importantes para desenvolvimento do CLDC. Esta configuração engloba dispositivos como telefones celulares, *pagers*, PDAs e dispositivos de tamanho similar.

A conexão destes dispositivos à rede é normalmente intermitente e no caso de comunicação 3G chegam a 2 Mbps. A CLDC é baseada em uma pequena máquina virtual chamada de KVM. Esta máquina virtual possui um subconjunto de funcionalidades de uma *Java Virtual Machine* (JVM) original (RUUSKANEN, 2008).

2.1.2 Mobile Information Device Profile

O MIDP é um *profile* baseado na configuração CLDC. Utiliza a KVM para executar em dispositivos móveis, sobretudo telefones celulares e alguns tipos de *pagers*. Foi desenvolvida pela JCP e sua especificação pode ser verificada no endereço eletrônico *http://jcp.org/jsr/detail/37.jsp.*

As características mínimas para que um dispositivo móvel possa utilizar a especificação MIDP são as seguintes (SUN, 2008):

- 128 Kb de memória não volátil para a implementação MIDP;
- 32 Kb de memória volátil;
- 8 Kb de memória não volátil para guardar dados persistentes;
- tela com resolução de 96 x 54 *pixels*;
- capacidade de entrada de dados; e,
- conexão de rede (possivelmente intermitente).

Os dispositivos que se enquadram nestas características são os telefones celulares, *pagers* e alguns tipos de PDAs. No presente trabalho será utilizado o profile MIDP 2.0 para desenvolvimento do sistema de *M-Banking*.

Os pacotes de classes Java pertencentes às especificações CLDC 1.1 e MIDP 2.0 estão na tabela 2.1.

Tabela 2.1: Pacotes CLDC e MIDP
Fonte: Knudsen (2003)

Classes CLDC 1.1	Classes MIDP 2.0
Java.lang	Javax.microedition.lcdui
Java.lang.ref	Javax.microedition.lcdui.game
Java.io	Javax.microedition.media
Java.útil	Javax.microedition.media.control
Javax.microedition.io	Javax.microedition.midlet
	Javax.microedition.pki
	Javax.microedition.rms

2.1.3 Midlets e Midlet Suite

Midlet é uma aplicação J2ME projetada para executar em um dispositivo que suporte o *profile* MIDP. Normalmente, os desenvolvedores agrupam *Midlets* que possuem funcionalidades ou características semelhantes em um mesmo pacote chamado de *Midlet Suíte*.

Membros de uma mesma *Midlet Suíte* compartilham recursos do dispositivo móvel hospedeiro e compartilham, também, as mesmas instâncias de classes Java e executam na mesma KVM. Isso significa que se três *Midlets* de uma mesma *Midlet Suíte* instanciam uma mesma classe, somente uma instância deste objeto será criada na KVM. Dados não podem ser compartilhados por *Midlets* que não estejam no mesmo *Midlet Suíte*, pois o nome da *Midlet Suíte* é usado para identificar informações associadas com o *suíte*.

O *Midlet Suíte* é instalado, executado e removido pelo gerente de aplicação que executa nos dispositivos. Cada fabricante pode fornecer este gerente de aplicação. Uma vez instalado, o gerente de aplicação será responsável por dar acesso a cada membro do *suíte* aos recursos do CLDC.

A classe javax.microedition.midlet.MIDLET é a base de todas as aplicações MIDP. Assim como as *aplets* e as *servlets*, as *Midlets* possuem um ciclo de vida bem definido. Este ciclo de vida é controlado por um software chamado *Java Application Manager* (JAM). Quando um usuário executa uma *Midlet*, é o JAM que cria uma instância da classe e executa os métodos.

2.2 Processo de desenvolvimento de uma aplicação MIDP

O processo de desenvolvimento de aplicações J2ME envolve os passos de compilação, pré-verificação, empacotamento, instalação e execução (PIROUMIAN, 2002). As aplicações MIDP são compiladas usando o compilador da plataforma J2SE. Um novo utilitário de pré-verificação examina os arquivos de extensão *class* gerados na compilação que podem ser interpretados da mesma forma pela KVM e pela JVM. Após os passos anteriores a aplicação MIDP é empacotada em dois arquivos: um de extensão JAR e outro, de extensão JAD para que possa ser instalada e, por fim, executada.

Testar aplicações em equipamentos reais aumentam os custos de desenvolvimento, por este motivo os emuladores são recursos importantíssimos para o desenvolvimento de aplicações MIDP. Fornece ao desenvolvedor uma ferramenta de testes e correções de eventuais erros com todas as características do dispositivo real. Importante salientar que os dispositivos reais não possibilitam a realização de testes como nos emuladores, mas são de fundamental importância na homologação da aplicação.

2.3 Arquitetura de Rede do J2ME

O MIDP suporta conexões de rede através do pacote javax.microedition.io. Este pacote fornece suporte básico para protocolos orientados à conexão e não orientados à conexão. O núcleo do pacote de rede do MIDP é o *Generic Connection Framework* (GFC). Ele define um mecanismo genérico para aplicações que querem realizar conexões de rede, podendo utilizar diferentes tipos de conexões que envolvem diferentes tipos de protocolos.

O GCF possibilita que o código da aplicação seja escrito independentemente de um tipo específico de conexão. Esta independência é importante em ambientes pervasivos, onde a natureza da rede pode afetar a disponibilidade dos serviços da aplicação. A classe de conexão "Connector" abstrai os detalhes de requisição e obtenção de diferentes tipos de protocolos de comunicação. Entre os protocolos suportados pelo *profile* MIDP estão: comunicação via porta serial, datagramas, *sockets*, HTTP e HTTPS. O MIDP não suporta protocolos de nível de aplicação como o RMI ou *Common Object Request Broker Architecture* (CORBA). Isso ocorre em função das limitações dos dispositivos móveis que necessitariam de poder de processamento para suportar estes mecanismos de comunicação.

3 SEGURANÇA NO J2ME

Sistemas de segurança existem para proteger objetos que possuam valor, como dinheiro ou propriedade intelectual. Existem sistemas mais seguros e outros menos seguros, mas a segurança perfeita é inalcançável.

O objetivo dos projetistas e desenvolvedores de sistemas de segurança é tornar o custo para descoberta de uma informação maior que o valor da própria informação (KNUDSEN,2008). Neste sentido, tanto as expectativas de clientes quanto de desenvolvedores têm de ser mantidas dentro de limites realistas, isto é, verificar se um ambiente é suficientemente seguro para um determinado propósito.

Existem alguns serviços que fundamentam as tecnologias de segurança (NORRIS, 2001):

- autenticação: é o processo de certificar a identidade tanto do cliente quanto do provedor do serviço;
- privacidade: é a garantia de que os dados que estiverem trafegando entre a origem e o destino da comunicação, não possam ser entendidos ou utilizados por terceiros;
- integridade: é a garantia de que os dados não foram alterados durante o processo de transmissão;
- autorização: é o processo de determinação de que um pessoa específica tem o direito de executar uma ação em particular, em relação a um objeto em particular e em determinada situação; e,
- não-repúdio: é o mecanismo pelo qual nenhuma das partes de uma negociação possa negar que a mesma ocorreu ou que participou dela.

3.1 Introdução à Segurança

Alguns conceitos de segurança serão brevemente abordados, com a finalidade de contextualizarmos os aspectos de segurança do J2ME. Basicamente, estes aspectos são os seguintes: Criptografia, Chave e Cifras, Troca de Chaves, Resumos de Mensagens, Certificados e Autoridades Certificadoras, SSL, TLS e HTTPS. Todas estas tecnologias serão discutidas a seguir.

3.1.1 Criptografia

Criptografia pode ser definida como o processo de codificar informações de maneira que somente a origem e o destino da troca de mensagens possam entende-los. Uma informação sem criptografia chama-se de texto claro, enquanto uma informação criptografada chama-se de texto cifrado. A criptografia baseia-se na matemática e no princípio de que certos problemas matemáticos são extremamente complexos para serem resolvidos e que necessitam de um tempo considerável para sua resolução. Existem diversos algoritmos de criptografia, entre eles pode-se citar: *Data Encryption Standart* (DES), *Triple Data Encryption Standart* (3DES), *Elliptic Curve Cryptography* (ECC), *Rivest Shamir Adelman* (RSA), *International Data Encryption Algorithm* (IDEA), *Rivest Cipher 4* (RC4), *Rivest Cipher 5* (RC5) entre outros.

3.1.2 Chaves e Cifras

A criptografia baseia-se no uso de chaves. A chave é uma informação ou número que, aplicado a um problema matemático, o torna de fácil resolução. Portanto, se a chave for de uso restrito, somente os possuidores da mesma serão capazes de ler o conteúdo da mensagem.

O tamanho da chave utilizada será o fator que determinará a facilidade ou a dificuldade de se "quebrar" um código (AREHART, 2001), quanto maior o tamanho da chave, mais valores possíveis existem para ela assumir.

A cifra é o algoritmo que mantém os dados confidenciais. Basicamente, pode ser considerado como uma equação matemática que transforma um número em texto claro em texto cifrado. Existem dois tipos de cifras: as cifras simétricas e as cifras assimétricas.

Na cifra simétrica, a mesma chave é compartilhada pela origem e pelo destino para criptografar e descriptografar informações. Este tipo de cifra possui bom desempenho, mas existe o problema de troca de chaves com segurança.

As cifras assimétricas trabalham com os conceitos de chave pública e privada. A chave pública pode ser livremente distribuída, enquanto a chave privada é de domínio restrito. A origem encripta a mensagem com a chave pública e somente o possuidor da chave privada será capaz de abri-la.

3.1.3 Troca de Chaves

Como citado anteriormente, apesar do bom desempenho o maior problema dos algoritmos de chave simétrica é a troca de chaves entre origem e destino da comunicação. Para resolver este problema, foram criados algoritmos de troca de chaves. Estes algoritmos permitem a troca de chaves simétricas com maior privacidade. A idéia é utilizar chaves assimétricas para facilitar a troca de um segredo compartilhado, que será utilizado posteriormente para criação de uma chave simétrica compartilhada para criptografar a comunicação de dados. Entre os algoritmos que são utilizados para este tipo de processo existem: *Diffie-Hellman* (DH), RSA e o ECC (AREHART, 2001).

3.1.4 Resumos de Mensagens

Uma função importante do processo de segurança é a garantia de que uma mensagem não tenha sido alterada durante o processo de transmissão. Para atingir este objetivo, foram criados os resumos de mensagens. Os resumos de mensagens são valores criados por funções de *hash*, gerados a partir de porções de uma mensagem ou seja, a função recebe alguns valores de entrada e gera um valor de *hash*, que é dependente do conteúdo da entrada e da função *hash* utilizada. Este valor *hash* é enviado separadamente do conteúdo original da mensagem, caso ocorram alterações nos dados da entrada resultariam em alterações significativas no valor *hash*. Desta maneira, torna-se fácil determinar se os dados foram corrompidos.

Entre os algoritmos que executam esta função de *hash* podem ser citados: *Message Digest 4* (MD4), *Message Digest 5* (MD5) e *Secure Hash Algorithm – 1* (SHA-1) (AREHART, 2001).

3.1.5 Certificados e Autoridades de Certificação

Os certificados foram criados para fornecer autenticação e podem ser aceitos, pois são emitidos e publicados por autoridades reconhecidas, ou seja, a confiança é depositada nessas autoridades. Os certificados contêm informações sobre o sujeito do certificado, a chave pública do mesmo, informações sobre entidade certificadora e assinatura da mesma. Podemos citar as seguintes entidades certificadoras: *AT&T Certificate Services, GTE CyberTrust, KeyWitness International, Thawte Consulting, VeriSign.*

3.1.6 Secure Socket Layer

O *Secure Socket Layer* foi proposto pela *Netscape* em 1997. É um protocolo que se encontra entre camada de transporte a camada de aplicação do modelo *Open Systems Interconnection* (OSI). Permite realizar uma conexão com mais segurança entre o cliente e o servidor.

Entre as funcionalidades do SSL estão à autenticação do cliente e do servidor através do uso de certificados e assinaturas digitais, encriptação de informações e integridade de dados.

Antes de serem verificados os aspectos inerentes aos protocolos que compõe o SSL, é necessário distinguir os conceitos de Sessão SSL e Conexão SSL. Stalings (1999) definiu estes termos da seguinte maneira:

- Conexão SSL: é o transporte oferecido para um tipo de serviço, possui o relacionamento fim-a-fim entre o cliente e o servidor, são passageiras e sempre associadas a uma sessão.

- Sessão SSL: é a ligação entre o cliente e o servidor, e é criada pelo Protocolo de *Handshake* Nas sessões são definidos os parâmetros que serão utilizados no compartilhamento de várias conexões.

Segundo Stalings (1999), em uma aplicação HTTP, entre um cliente e um servidor existem várias conexões e uma sessão entre ambos, apesar de em teoria ser possível estabelecer várias sessões simultâneas.

O SSL não é um protocolo único. Ele é composto por duas camadas de protocolos distintos, na primeira camada o SSL Record Protocol e na segunda camada o SSL *Handshake Protocol*, o SSL *Change Cipher Spec* e o SSL *Alert Protocol*. A figura 3.1 mostra a pilha de protocolos SSL.

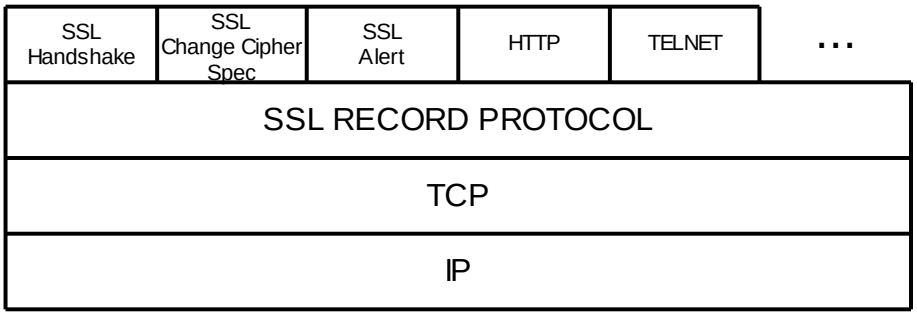

Figura 3.1: Pilha de Protocolos SSL

O SSL *Record Protocol* é responsável pelo encapsulamento das informações dos níveis superiores e a transferência de dados entre o cliente e o servidor. Além disso, pode também fragmentar, comprimir, anexar assinaturas digitais e criptografar informações.

Acima do SSL *Record Protocol* existem os protocolos de *HandShake*, *Change Cipher Spec* e *Alert*. Estes protocolos são responsáveis por estabelecer a ligação entre o cliente e o servidor (seqüência de *handshake*), estabelecer um algoritmo para a sessão e por gerenciar as mensagens de erro.

A sessão SSL é estabelecida através de uma seqüência de *handshake* entre a parte cliente e o servidor. Esta seqüência pode variar dependendo da configuração do servidor. Existem momentos em que são necessárias etapas adicionais para o gerenciamento de informações criptográficas. Em uma situação mais genérica, os passos para estabelecimento de sessão com o SSL são os seguintes:

- cliente e servidor trocam mensagens de saudação (*Hello*) que contém dados aleatórios;

- o servidor envia ao cliente sua chave pública, com um certificado assinado;

- o cliente gera dados aleatórios para usar como um "segredo compartilhado", enviando em seguida estes dados ao servidor, criptografados com a chave pública dele, em uma mensagem denominada *Client Key Exchange*;

- cliente e servidor utilizam uma combinação dos dados aleatórios trocados e dos dados secretos gerados pelo cliente para gerar as chaves criptográficas que serão utilizadas;

- o cliente envia uma mensagem denominada *Change Cypher Spec* para determinar o serviço de criptografia a ser utilizado, e uma mensagem de fim de estabelecimento de conexão (*Finished*); e,

– o servidor responde com o envio de ambos o *Change Cypher Spec* e a mensagem *Finished*.

Ambas as mensagens de fim de estabelecimento de conexão são protegidas pelos parâmetros de criptografia adotados. Deste modo, cliente e servidor podem garantir que estão aptos a se comunicar de forma segura. Não há a preocupação com erros de transmissão, pois o *Transmission Control Protocol* (TCP) fornece um serviço confiável de entrega de dados.

O SSL, como protocolo de transporte seguro, proporciona alguns serviços de segurança necessários:

– confidencialidade - a informação que circula entre o cliente e o servidor que atua na frente do serviço de criptografia, utilizando criptografia de chave simétrica (com uma chave de sessão definida no *handshake*).

– autenticação - as partes que mantêm a comunicação se autenticam mediante certificados baseados em criptografia de chave pública. Isto não ocorre sempre assim, O mais habitual é que seja unicamente o servidor autenticado mediante um certificado digital.

– integridade - a integridade dos dados transmitidos é assegurada usando códigos de integridade (*Message Authentication Code* - MAC) calculados mediante funções de *hash* (SHA ou MD5) (CABRAL, 2002).

3.1.7 Transport Layer Security

O TLS foi desenvolvido pelo IETF, a fim de ser concretizado como um padrão para à Internet. Foi baseado na versão 3.0 do SSL e, em essência, o TLS pode ser considerado como uma versão 3.1 do SSL (STALLIGS, 1999). As principais diferenças entre os dois protocolos estão descritos na tabela 3.1.

Tabela 3.1: Diferenças entre o SSL e o TLS
Fonte: Cabral (2002)

Característica	SSL	TLS
Alerta de erros	Durante a execução do protocolo de *handshake* o servidor deverá esperar pela resposta do cliente. Se o servidor enviar uma mensagem de requisição de certificado, o cliente deverá enviar a mensagem com o certificado ou um alerta *no_certificate*	Uma vez que o servidor envie uma mensagem solicitando o certificado, a resposta deverá ser enviada pelo cliente contendo o certificado
Algoritmos para troca de chaves	Os algoritmos de troca de chaves existentes são RSA, Diffie Hellman e Fortezza Kea.	Não suporta o algoritmo de troca de chaves Fortezza Kea.
Cálculo MAC	No cálculo MAC não se insere o tipo de conteúdo nem a versão do protocolo, portanto estes campos não são protegidos de ataques contra sua integridade.	Os campos tipo de conteúdo e versão do protocolo são protegidos.

Os objetivos propostos através da criação do protocolo TLS são conforme Cabral *apud* Dierks (CABRAL, 2002):

– segurança na criptografia - TLS é utilizado para estabelecer uma conexão cifrada entre as partes;

– interoperabilidade - Programadores independentes podem desenvolver aplicações que utilizem parâmetros de troca de criptografia, sem ter conhecimento de outros códigos;

– flexibilidade - TLS procura oferecer uma estrutura que permita a incorporação de uma nova chave pública e métodos de criptografia conforme necessário. Isto previne contra a necessidade de criar um novo protocolo (arriscando a introdução de novas falhas), e evitar a necessidade de implementar uma nova biblioteca de segurança;

– relativa eficiência - Operações de Criptografia tendem a utilizar muitos recursos da *Central Processing Unit* (CPU). Por esta razão, o protocolo TLS incorporou uma sessão opcional de *caching* para reduzir o número de novas conexões a serem estabelecidas. Este cuidado foi tomado de forma a reduzir a atividade da rede.

3.1.8 Hypertext Transfer Protocol Secure

O protocolo HTTP não é um protocolo seguro. A troca de informações utilizando este protocolo está sujeita a ação de curiosos e pessoas mal intencionadas. O HTTPS é a alternativa mais segura para troca de informações. O HTTPS utiliza o TLS ou SSL para fornecer autenticação e um serviço de criptografia para os protocolos de mais alto nível. Como citado anteriormente, a versão 2.0 do MIDP fornece suporte para o HTTPS.

3.2 Aspectos de Segurança do J2ME

Como mencionado anteriormente, o GCF suporta vários tipos protocolos de comunicação entre eles, o HTTPS. O HTTPS fornece uma conexão segura utilizando o TLS ou SSL na conexão entre o dispositivo móvel e o Servidor de conteúdos. Esta conexão está ilustrada na figura 3.2.

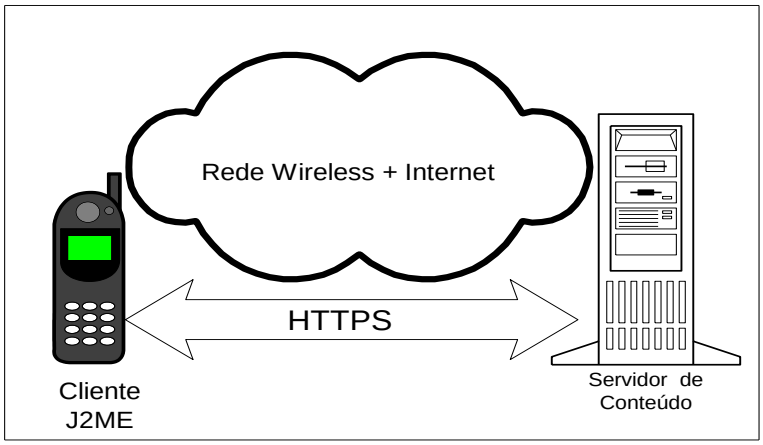

Figura 3.2: Comunicação J2ME

Uma *Midlet* executa em um ambiente confinado (KVM) imune a códigos mal ou maliciosamente escritos. Na pior das hipóteses, este código levará a parada do ambiente Java, permitindo que o restante do dispositivo fique a salvo.

Entretanto, a segurança no MIDP não se restringe ao confinamento do software que está em execução, proporcionado pela máquina Java. O MIDP possui outras características que podem proteger os dispositivos e seus usuários de ataques.

A especificação MIDP define um conjunto de permissões para qualquer tipo de conexão de rede. Para uma aplicação de rede ser acessada, a *Midlet* deve possuir a permissão de rede apropriada.

As permissões definidas pela especificação MIDP 2.0 fazem parte de um conjunto de classes que permitem o estabelecimento de comunicação entre dois nós de uma rede. As classes definidas e o respectivo protocolo de comunicação mais utilizados na arquitetura MIDP estão descritas na tabela 3.2.

Tabela 3.2: Classes de Conexões MIDP

Tipo de Conexão	Classe MIDP
HTTP	javax.microedition.io.Connector.http
HTTPS	javax.microedition.io.Connector.https
Sockets	javax.microedition.io.Connector.socket
SSL	javax.microedition.io.Connector.ssl
Datagrama	javax.microedition.io.Connector.datagram

As *Midlets* não adquirem as permissões de maneira explícita no código, estas permissões são concedidas através dos domínios de proteção. Os domínios de proteção determinam que permissões são concedidas, negadas e as que são de mérito do usuário.

O J2ME *Wireless Toolkit* (ferramenta de desenvolvimento de *Midlets* desenvolvida pela *Sun MicroSystems*), possui quatro domínios de proteção: *Trusted, Untrusted, Minimum e Maximum.*

- os domínios *Trusted* e *Maximum* são equivalentes e todas as permissões são concedidas, ou seja, todas as conexões de rede podem ser realizadas;

- no domínio *Untrusted,* o usuário é sempre consultado sobre a realização de uma conexão; e,

- no domínio *Minimum,* todas as permissões são negadas imediatamente.

Outro aspecto importante da segurança das *Midlets* está relacionado ao *download* de códigos no dispositivo móvel. Baixar códigos pode ser perigoso, especialmente quando a fonte é desconhecida. Mesmo as *Midlets* que executam em um ambiente restrito apresentam alguns riscos. Alguns códigos poderiam fazer ligações não autorizadas que gerariam custos ou poderiam coletar informações do dispositivo para uso indevido por terceiros.

A especificação MIDP sugere que um domínio de proteção seja baseado em assinaturas e certificados (SUN, 2008). A idéia é que o desenvolvedor do *software* obtenha um certificado digital de uma Autoridade Certificadora. A partir desse certificado, o usuário que baixasse software em seu dispositivo estaria seguro da identidade de quem desenvolveu o software que está sendo baixado.

4 PROTÓTIPO DE *M-BANKING*

Até este capítulo foram estudados os aspectos teóricos que fundamentam o protótipo de *M-Banking* proposto. Neste capítulo serão apresentadas as características gerais, a estrutura, o funcionamento e os principais aspectos de segurança envolvidos no desenvolvimento do protótipo concebido pelo autor.

4.1 Características Gerais

O protótipo de *M-Banking* possibilita aos seus usuários acessar as informações de saldo, extrato, transferência de valores e pagamento de títulos bancários através de telefone celular que possua suporte a tecnologia J2ME/MIDP. Utiliza o protocolo HTTPS para comunicação fim-a-fim entre o Cliente e o Servidor de Conteúdo e ainda utiliza algumas técnicas de segurança como: senha de acesso, *message digests* e certificados, para garantia dos preceitos básicos de segurança citados por Norris (2001): autenticação, privacidade, integridade, autorização e não-repúdio.

Em função das limitações dos dispositivos móveis é fundamental que a carga maior de processamento seja realizada pelo Servidor de Conteúdo. Assim sendo foi escolhida a tecnologia de *Servlets* para atendimento das requisições dos dispositivos móveis e realização das interações com o Sistema Gerenciador de Banco de Dados (SGBD) MySQL onde residem os dados dos Clientes.

O objetivo do protótipo é demonstrar a viabilidade técnica de construir sistemas móveis a partir da tecnologia J2ME, com certo grau de segurança e boa usabilidade. Dentro deste contexto, não foram implementadas soluções complexas na parte do Servidor de Conteúdo. Foram construídas soluções relativamente simples, adequadas para o desenvolvimento e testes da aplicação no dispositivo móvel.

4.2 Tecnologias Utilizadas no Projeto

O protótipo de *M-Banking* foi baseado nas seguintes tecnologias:

- no Cliente foi utilizada a tecnologia J2ME, especificamente a configuração CLDC 1.1 e o profile MIDP 2.0;

– no Servidor foram utilizados o servidor TomCat 4.0, *Servlets* (para processamento da requisições dos dispositivos móveis) e um banco de dados relacional MySQL; e,

– como elementos de segurança do projeto foram utilizadas as ferramentas *Java* para geração certificados para Servidores (KEYTOOL), o pacote de classes *Bouncy Castle Cryptography* (fundamental para o processo de autenticação de usuários) e para a comunicação entre o Cliente e o Servidor a utilização do protocolo HTTPS.

Outras tecnologias foram estudadas, mas não foram utilizadas no protótipo. Entre elas as tecnologias XML, *Simple Object Access Protocol* (SOAP) e conceitos de *Web Services*.

4.3 Ferramentas de Desenvolvimento

Como ambiente de desenvolvimento e testes do presente projeto foram utilizadas duas ferramentas da *Sun Microsystems*, o *Forte for Java 4.0 Community Edition (CE)* e o *J2ME Wireless Toolkit 2.0* (WTK20). Estas ferramentas proporcionaram todas as funcionalidades necessárias para o completo desenvolvimento e testes do protótipo proposto.

4.3.1 Forte for Java 4.0 Community Editon

Existem diversos editores que possibilitam a criação de códigos em *Java*, JSP, *Servlets*, HTML e XML. O protótipo foi desenvolvido com a IDE *Forte Java* da *Sun Microsystems*. A figura 4.1 mostra uma visão geral da ferramenta.

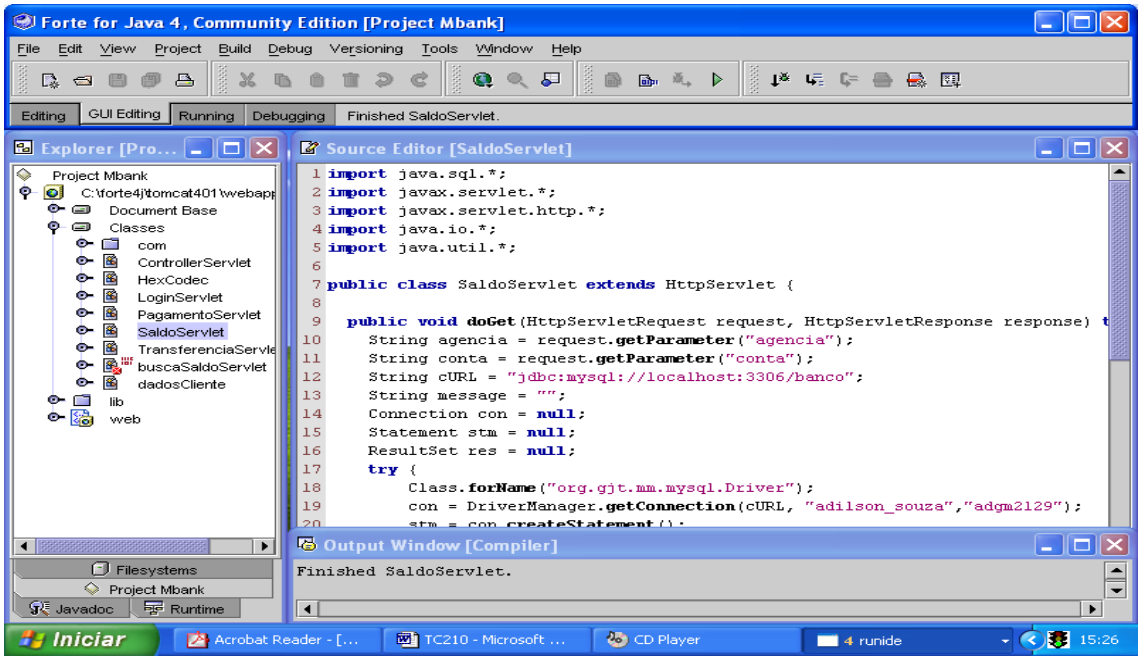

Figura 4.1: Visão da ferramenta *Forte for Java 4.0 CE*

4.3.2 J2ME Wireless Toolkit 2.0

O WTK20 é a ferramenta mais importante do projeto, sem ela não seria possível realizar os testes necessários para implementação do protótipo. Entre outras funções fornece emuladores de dispositivos reais para validação de aplicações, monitores de memória, rede e ferramentas para gerenciamento de certificados. As figuras 4.2 e 4.3 mostram a interface do *software* com o desenvolvedor.

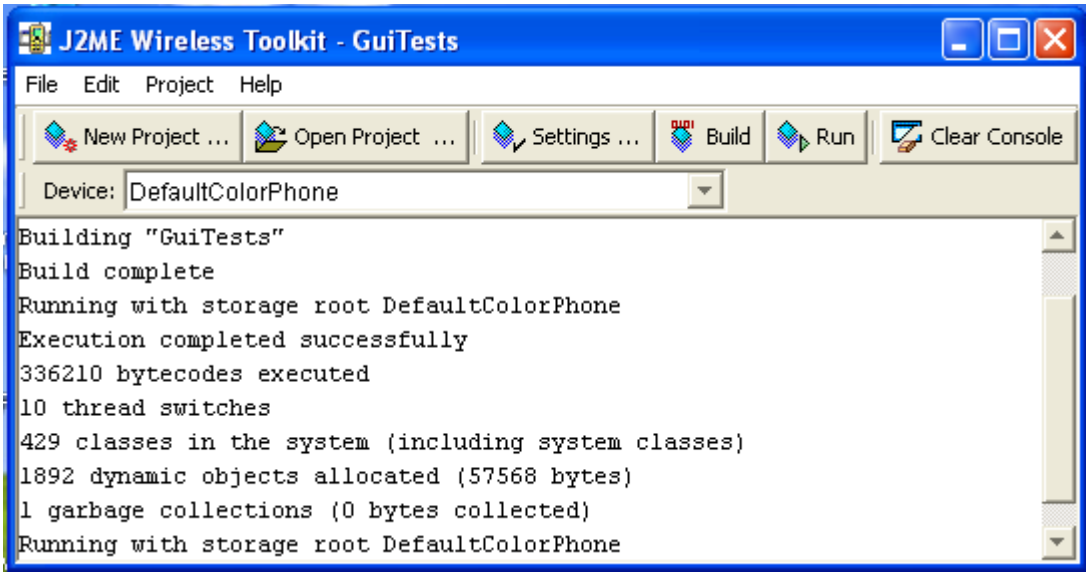

Figura 4.2: *Interface* do WTK20 com o Desenvolvedor

Figura 4.3: Emulador do WTK20

Hoje esta ferramenta já se encontra na versão 2.5 e já vem embutida na IDE Netbeans que é de distribuição gratuíta e liberada para *download* no *site* http://www.javasoft.com.

4.4 Dispositivos e Fabricantes que Suportam a Tecnologia MIDP 2.0

Praticamente todos os grandes fabricantes de telefones celulares (Nokia, Motorola, Sony Ericsson, Samsung) fabricam equipamentos que suportam a tecnologia J2ME. O site JBenchmark (2008) apresenta uma completa lista de fabricante e equipamentos que suportam o J2ME/MIDP 2.0.

4.5 Arquitetura do Protótipo

A figura 4.4 mostra a arquitetura do protótipo de *M-Banking*. O dispositivo móvel com suporte ao J2ME realiza suas interações com o Servidor de Conteúdos através de uma conexão criptografada. Tanto as requisições quanto as respostas para o dispositivo móvel trafegam por uma conexão segura utilizando o protocolo HTTPS.

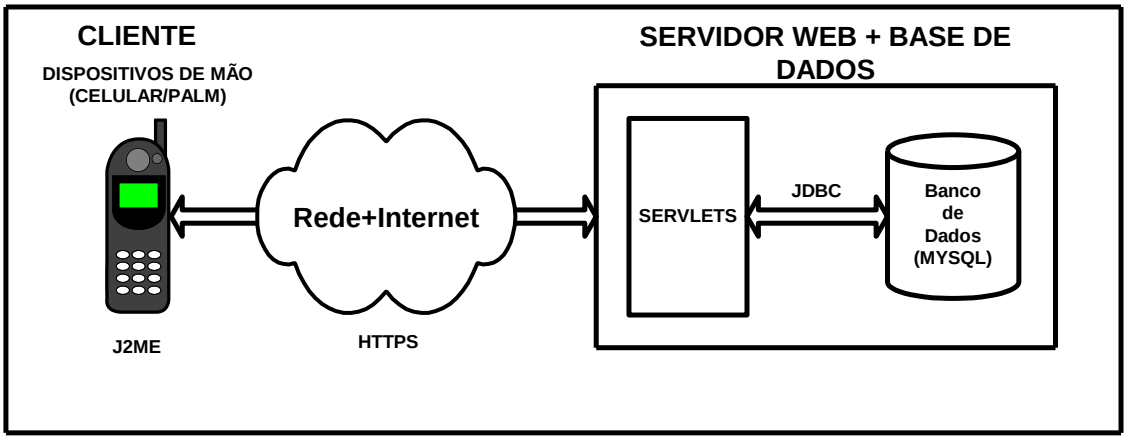

Figura 4.4: Arquitetura do Protótipo de *M-Banking*

Do lado do Servidor as interações entre os *Servlets* e a Base de Dados MySQL é realizada através do *Java DataBase Connectivity* (JDBC).

4.6 Modelagem do Protótipo de *M-Banking*

O protótipo de *M-Banking* pode ser modelado a partir de objetos representados como classes do sistema. Visto que a tecnologia J2ME tem como base a linguagem Java, constata-se que a metodologia Orientada a Objetos é a solução mais indicada para modelar o presente projeto. Através da Orientação a Objetos, é possível desenvolver componentes de *software* reutilizáveis, extensíveis e mais legíveis, devido às características de herança e poliformismo oferecidas por este paradigma.

A fim de expressar os requisitos do sistema em uma notação do paradigma orientado a objeto será utilizada a notação UML. Esta notação, conforme Carvalho *apud* Lopes *et al* (2002), é aplicável para:

- mostrar as fronteiras de um sistema e suas funções principais utilizando atores e casos de uso;

– ilustrar a realização de casos de uso com diagramas de interação;

– representar uma estrutura estática de um sistema utilizando diagramas de classe;

– modelar o comportamento de objetos com diagramas de transição de estado; e,

– revelar a arquitetura de implementação física com diagramas de componentes e de implementação.

A notação UML é uma linguagem para especificação, visualização, construção e documentação de artefatos de sistemas de software, assim como para modelagem de negócios e outros tipos de sistemas. A UML representa uma coletânea das melhores práticas de engenharia que se mostraram vitoriosas na modelagem de sistemas grandes e complexos (CARVALHO, 2002).

Antes de iniciar a modelagem do protótipo de *M-Banking*, serão feitas algumas considerações sobre o contexto do sistema:

– o Cliente poderá possuir várias contas-correntes e várias contas-poupança;

– tanto a conta corrente quanto a poupança deverão manter um histórico de todas as movimentações de crédito e débito; e,

– o Cliente a partir de seu dispositivo móvel poderá executar as funções de consulta de saldo, extrato, pagamento de títulos e transferência de valores para outra conta-corrente.

A seguir são abordados os aspectos relativos os requisitos e modelagem do sistema.

4.6.1 Análise de Requisitos

As funcionalidades do Cliente do protótipo de *M-Banking* são as seguintes:

- autenticação do usuário no sistema a partir da digitação de agência, conta e senha pessoal (já pré-castrada no sistema);

- consulta de saldo de conta-corrente e/ou conta-poupança, informando data/hora das consultas;

- consulta extrato de movimentação de conta-corrente e/ou poupança, com data, número da operação, código de operação e valor;

- pagamento de títulos; e,

- transferência de valores para outras contas-correntes.

4.6.2 Diagrama de *Use-Case*

A partir das funcionalidades descritas na fase de análise de requisitos, pode ser gerado um modelo de *Use-Case* orientado para as ações do Cliente. A figura 4.5. ilustra esta visão.

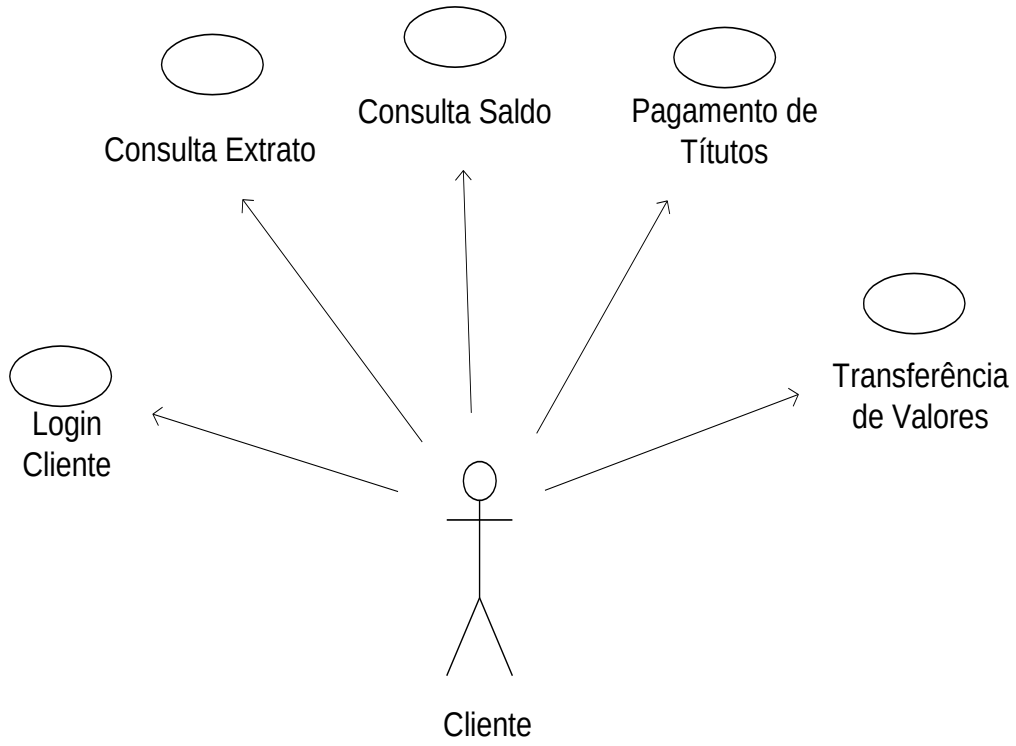

Figura 4.5: Diagrama de *Use-Case* visão Cliente

O diagrama de *Use-Case* ilustrado acima permite que sejam expressadas as funcionalidades e comportamentos pretendidos pelo principal ator do protótipo de *M-Banking.*

4.6.3 Diagrama de Classes

O diagrama de classes representa o modelo da estrutura de um sistema orientado a objetos. A partir do diagrama de *Use-Case*, pode ser elaborado o diagrama de classes do sistema de *M-Banking*. Foram identificadas 11 classes básicas na parte Cliente, que se relacionam conforme o diagrama seguinte.

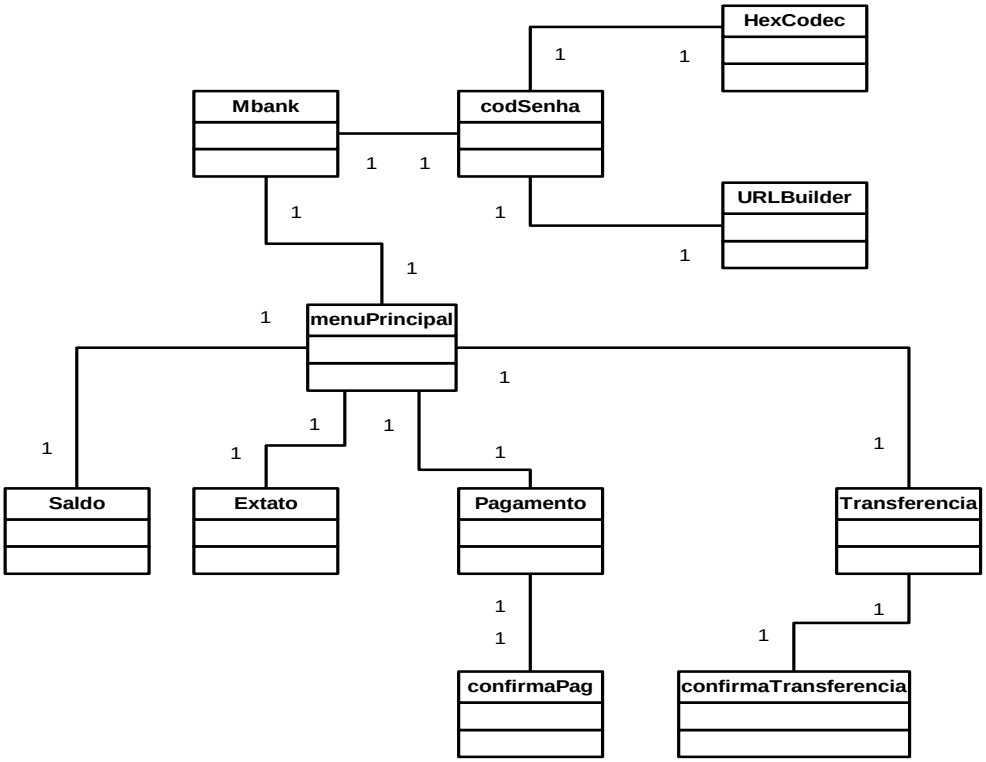

Figura 4.6: Diagrama de Classes *M-Banking*

O diagrama completo das classes desenvolvidas para parte Cliente com seus respectivos atributos e métodos, encontra-se no anexo I deste trabalho.

4.6.4 Diagramas de Seqüência

De posse do diagrama de *Use-Case* e do diagrama classes da etapa de análise do domínio do problema, são traçadas as diversas interações entre as classes do sistema.

Nesta etapa, são utilizados diagramas de seqüência para modelar cada função definida no diagrama de *Use-Case*. A escolha por esta espécie de diagrama possibilita dar mais ênfase a ordem cronológica das diversas interações entre os objetos. Dentro deste contexto a partir deste momento cada uma das funcionalidades do protótipo serão objeto de análise.

4.6.5 Funções definidas no protótipo de M-Banking

Dentro do contexto proposto, inicia-se a partir deste momento a abordagem específica de cada uma das funcionalidades do protótipo.

4.6.5.1 Autenticação do Usuário

Nesta etapa o Cliente realiza sua autenticação no sistema. Fornece o número da conta, número da agência e a sua senha secreta de 6 (seis) dígitos. O diagrama a seguir ilustra esta interação.

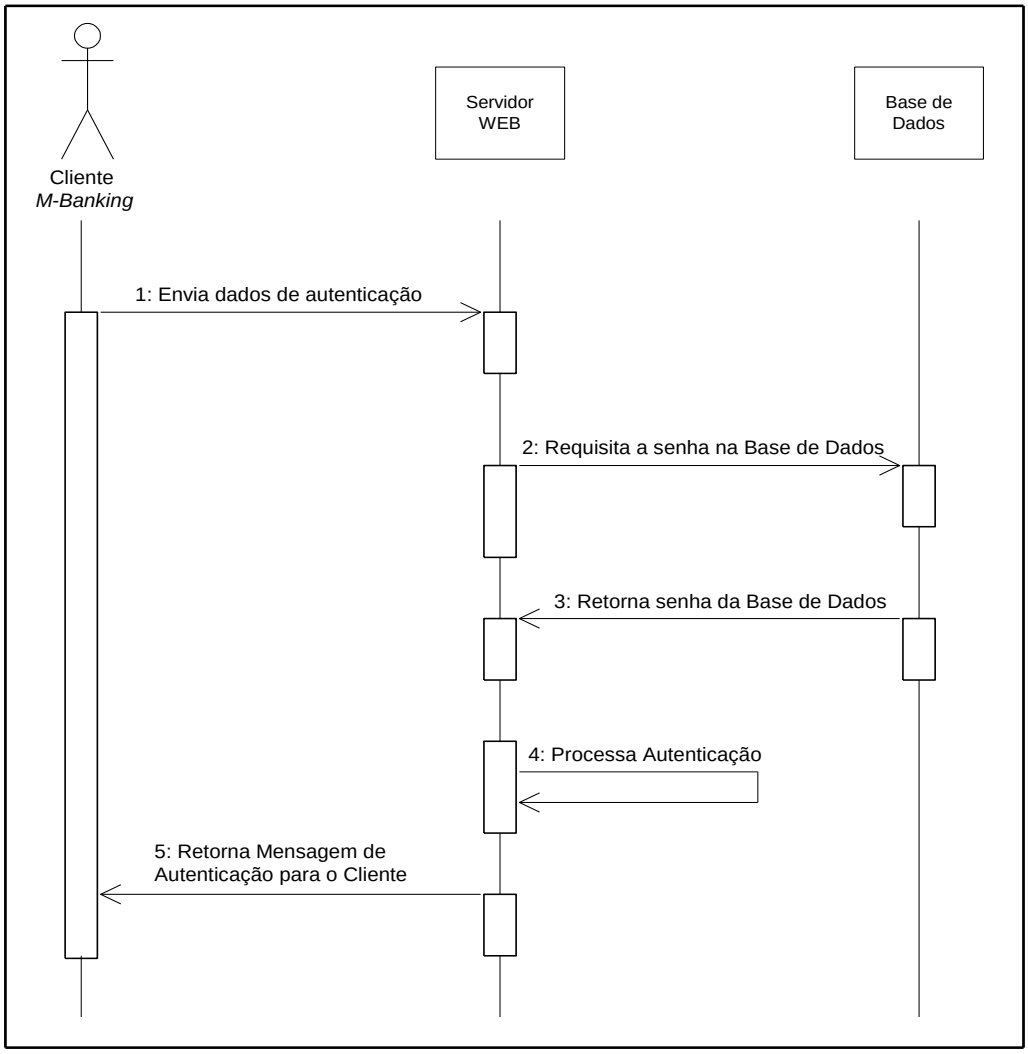

Figura 4.7: Operações de autenticação do Cliente

As operações descritas no diagrama anterior também podem ser mostradas no *Use-Case* textual desta funcionalidade.

Tabela 4.1: *Use-Case* textual do processo de autenticação do Cliente

Operação	Descrição	Figura
1. Seleção Mbank	Cliente seleciona opção de Mbank	4.8
2. Autenticação	Cliente envia dados para autenticação (Conta, agência e senha secreta)	4.9
3. Falha no login	No caso do cliente entrar com algum dado incorreto aparecerá uma mensagem de falha	4.10
4. Login com Sucesso	Se o login ocorrer de maneira correta aparecerá a tela de opções.	4.11

Figura 4.8: Tela seleção do sistema

Figura 4.9: Tela de entrada de dados da função de autenticação

Figura 4.10: Tela login sem sucesso

Figura 4.11: Tela de opções

A partir do sucesso da operação de autenticação do Cliente, ele estará habilitado a executar as operações disponíveis no protótipo de *M-Banking* (Saldo, Extrato, Transferência de Valores e Pagamento de Títulos). Estas opções serão vistas em detalhes a seguir.

4.6.5.2 Consulta de Saldo

Para acessar a opção de Saldo, basta o usuário selecionar a opção no menu do dispositivo móvel. Após selecionada, será feita uma requisição ao Servidor. Este por sua vez enviará para o Cliente a informação solicitada. Estas ações estão ilustradas no diagrama da figura 4.12.

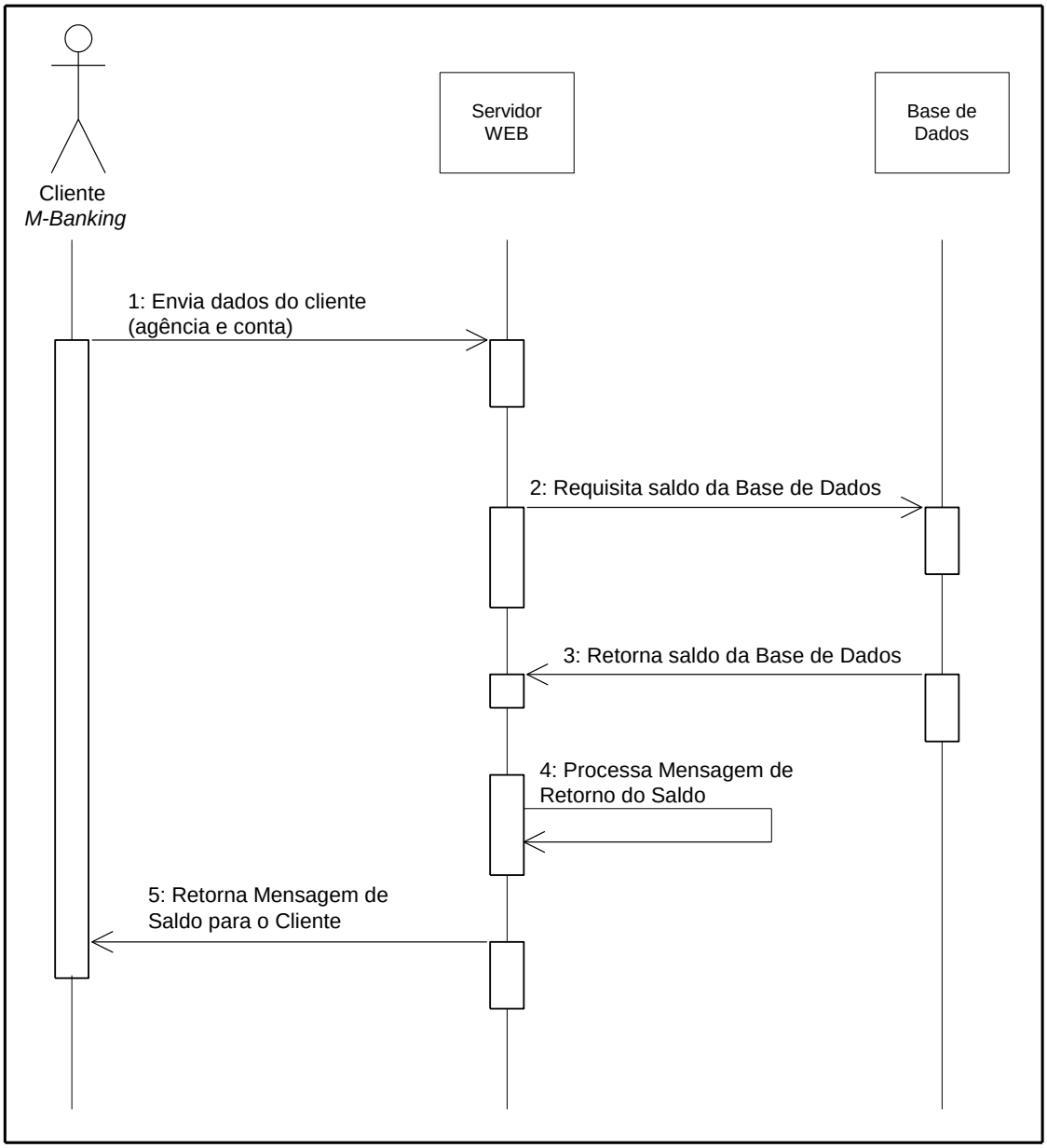

Figura 4.12: Operações para consulta do Saldo

Tabela 4.2: *Use-Case* textual opção de Saldo

Operação	Descrição	Figura
1. Seleção Saldo	Cliente seleciona opção de Saldo	4.13
2. Tela com saldo atualizado	Cliente recebe as informações sobre seus saldo atualizado	4.14

Figura 4.13: Seleção da opção Saldo

Figura 4.14: Tela retorno do Saldo

4.6.5.3 Consulta de Extrato

Para acessar a opção de Extrato, basta o usuário selecionar a opção no menu do dispositivo móvel. Após selecionada, será feita uma requisição ao Servidor. Este por sua vez enviará para o Cliente a informação sobre todas as movimentações realizadas na conta do Cliente em ordem crescente de data. As informações apresentadas aos Clientes são as seguintes:

- Data da realização da operação;

- Número de identificação da operação;

- Código da Operação (D-Débito / C-Crédito); e,

- Valor da operação.

Estas ações estão ilustradas no diagrama da figura 4.15.

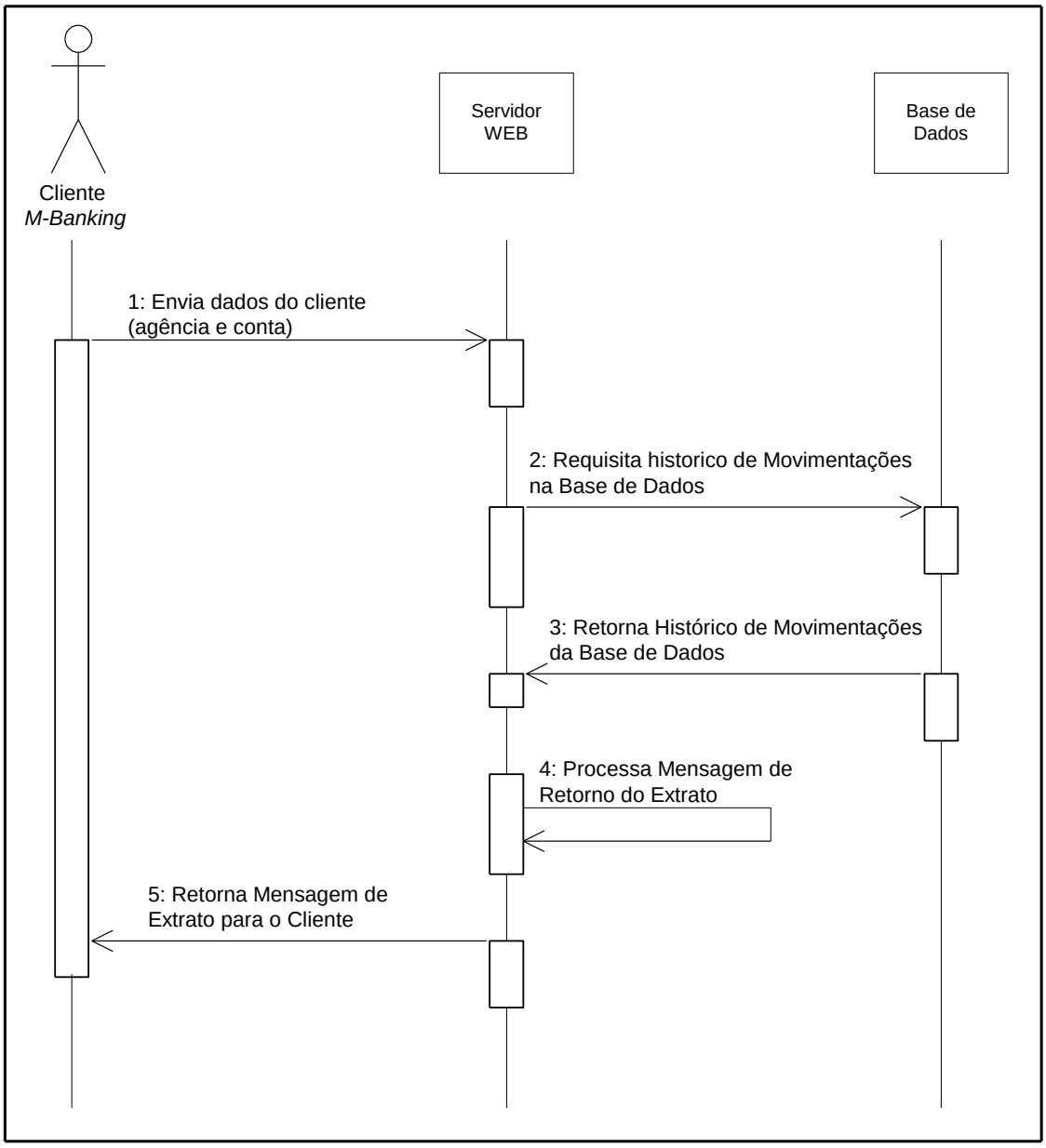

Figura 4.15: Operações para consulta do Extrato

Tabela 4.3: *Use-Case* textual opção de Extrato

Operação	Descrição	Figura
1. Seleção Extrato	Cliente seleciona opção de Extrato	4.16
2. Tela com extrato atualizado	Cliente recebe as informações sobre suas movimentações financeiras	4.17

Figura 4.16: Seleção da opção Extrato

Figura 4.17: Tela retorno do Extrato

Em essência não existe diferença significativa entre os códigos desenvolvidos tanto para a opção de Saldo quanto para a opção de Extrato. A diferença básica ocorre no tipo de seleção realizada na tabela de movimentação no banco de dados MySQL.

4.6.5.4 Transferência de Valores

Para acessar a opção de Transferência de Valores, basta o usuário selecionar a opção no menu do dispositivo móvel. Após selecionada, será mostrada uma nova tela onde o usuário deverá informar o número da agência, conta e valor a ser transferido para o Cliente destino. Após submissão das informações aparecerá uma nova tela com os dados do Cliente origem (número agência, número conta e nome), Cliente destino (número agência, número conta e nome) e o valor a ser transferido. Para realização da operação basta o Cliente confirmá-la.

Em caso de problemas na execução da operação será exibida uma mensagem de erro, caso contrário o Cliente receberá uma mensagem de execução com sucesso da operação. Estas ações estão ilustradas no diagrama da figura 4.18.

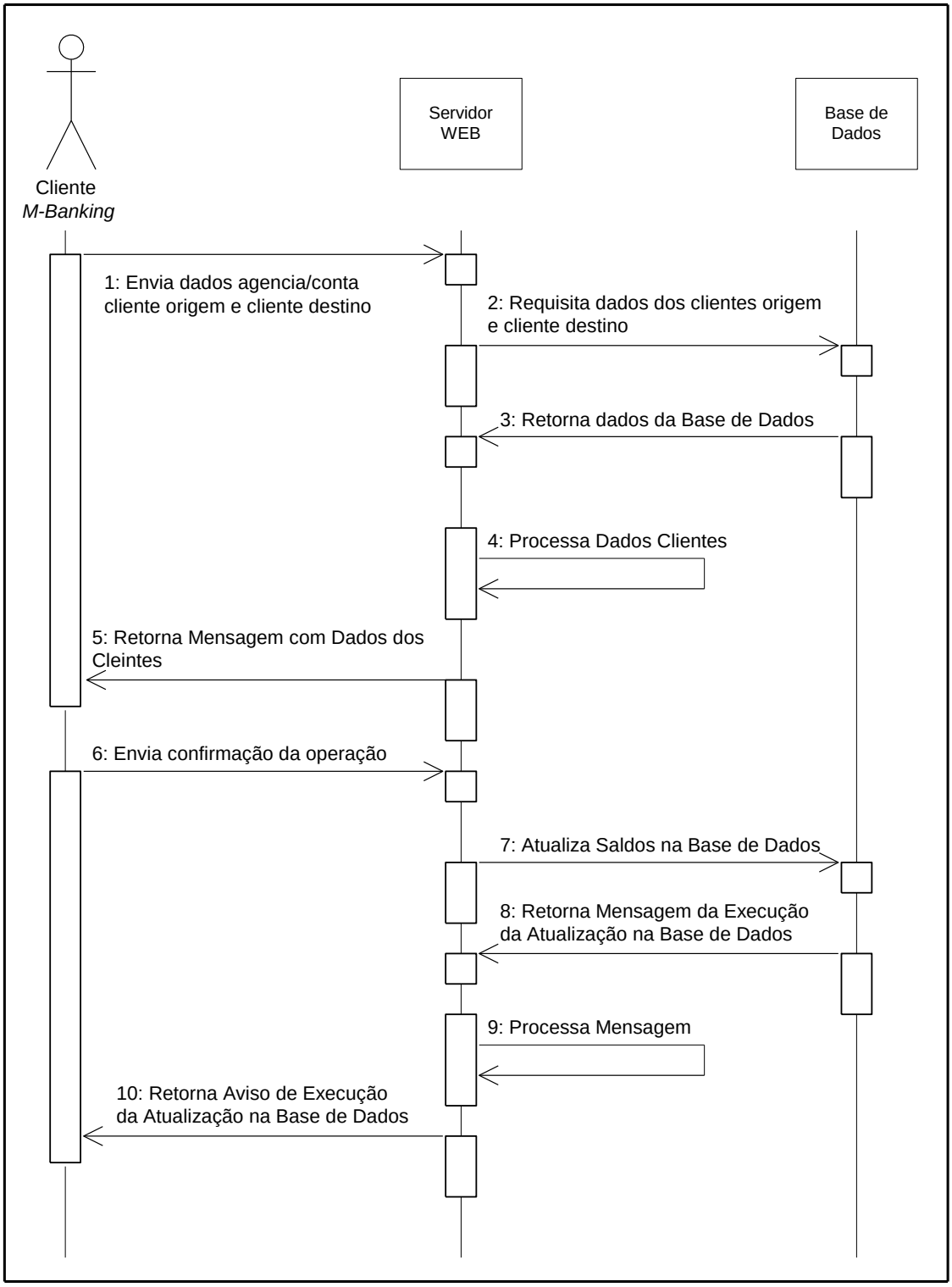

Figura 4.18: Operações para execução de Transferência de Valores

Tabela 4.4: *Use-Case* textual da opção de Transferência

Operação	Descrição	Figura
1. Seleção Transferência	Cliente seleciona opção de transferência	4.19
2. Tela formulário de transferência	Cliente recebe uma tela para preenchimento dos dados da conta destino da operação	4.20
3. Tela de Falha	Em caso de problemas com os dados da conta destino, é emitida uma mensagem informativa para o Cliente	4.21
4. Tela de Confirmação de transferência	Se os dados estiverem corretos aparecerá uma nova tela com os dados do Cliente de origem, Cliente de destino e valor para confirmação	4.22
5. Tele de retorno da transferência	Após executada a transferência é mostrada uma tela informativa do sucesso da operação realizada	4.23

Figura 4.19: Seleção da opção Transferência

Figura 4.20: Tela entrada de dados

Figura 4.21: Tela de Erro

Figura 4.22: Tela de Confirmação

Figura 4.23: Tela de Retorno

Esta operação realiza duas requisições ao Banco de Dados. A primeira verifica a existência da conta de destino da operação e, no caso de sua existência, busca os nomes dos Clientes envolvidos na transação. A segunda realiza a atualização dos saldos de ambos os Clientes.

4.6.5.5 Pagamento de Títulos

Para acessar a opção de Pagamento de Títulos, basta o usuário selecionar a opção no menu do dispositivo móvel. Após selecionada, será mostrada uma nova tela onde o usuário deverá informar o número do título, nome do cedente, data de vencimento e valor do pagamento. Após a submissão das informações será exibida uma nova tela para confirmação dos dados digitados. Para realização da operação basta o Cliente confirmá-la.

Em caso de problemas na execução da operação será emitida uma mensagem de erro, caso contrário o Cliente receberá uma mensagem de execução com sucesso da operação. Estas ações estão ilustradas no diagrama da figura 4.24.

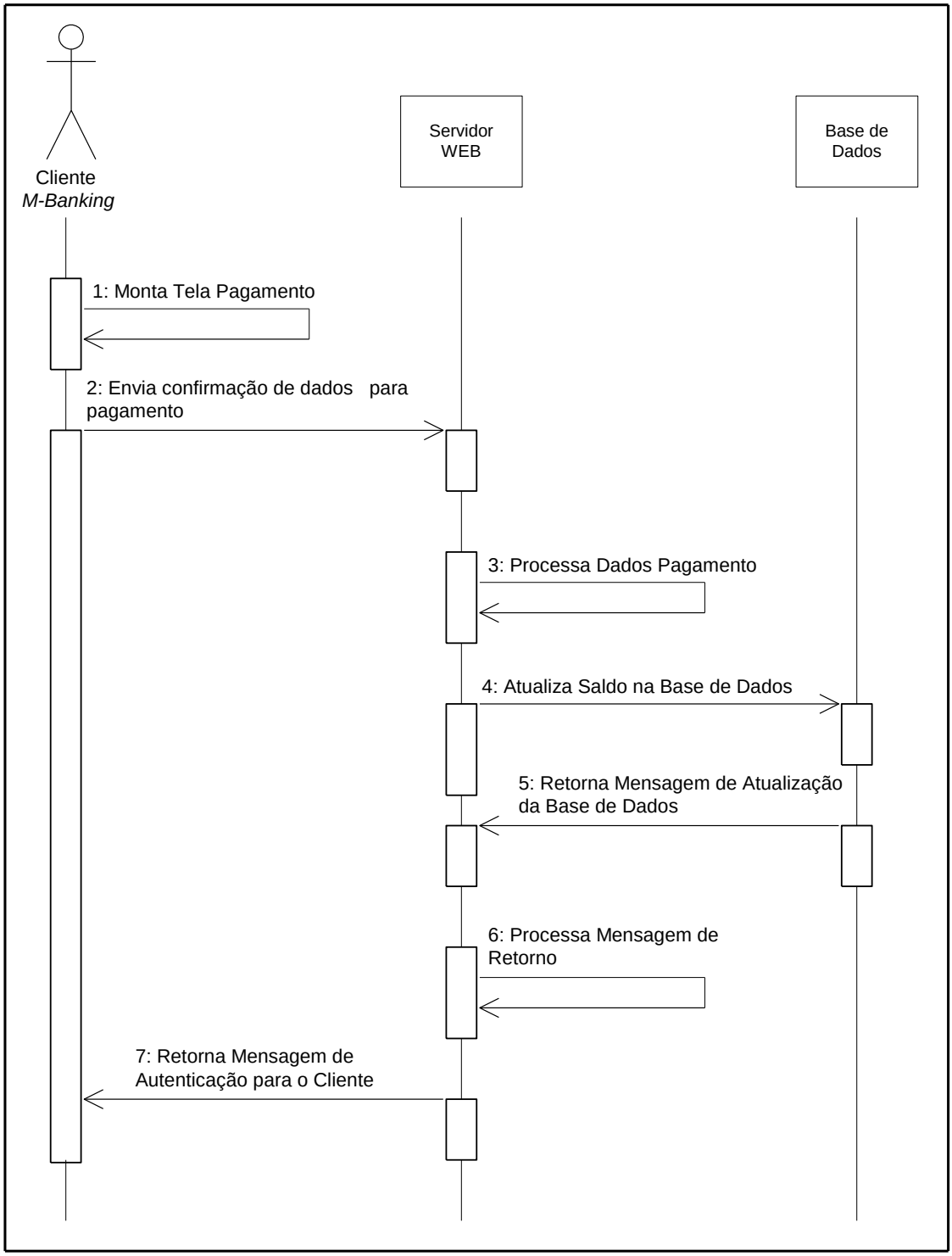

Figura 4.24: Operações para execução de Pagamento de Títulos

Tabela 4.5: *Use-Case* textual da opção de Pagamento

Operação	Descrição	Figura
1. Seleção Pagamento	Cliente seleciona opção de pagamento	4.25
2. Tela formulário de pagamento	Cliente recebe uma tela para preenchimento dos dados sobre o título a ser pago	4.26
3. Tela de Confirmação do pagamento	Aparecerá uma tela com os dados do título a ser pago para confirmação.	4.27
4. Tele de retorno do pagamento	Após executada do pagamento é mostrada uma tela informativa do sucesso da operação realizada	4.28
5. Tela de Falha	Em caso de problemas com pagamento do título será emitida uma mensagem informativa para o Cliente.	4.29

Figura 4.25: Seleção de opção de pagamento

Figura 4.26: Tela de entrada de dados

Figura 4.27: Tela de confirmação de dados do título

Figura 4.28: Tela de retorno do pagamento com sucesso

Figura 4.29: Tela de retorno do pagamento sem sucesso

Esta opção realiza somente um acesso à Base de Dados. Não existe uma rotina do lado do Servidor para tratamento das informações do título a ser pago. É executada somente a atualização do saldo do Cliente.

4.6.6 Diagrama de Implantação

Conforme citado anteriormente, o *profile* MIDP 2.0 permite que sejam estabelecidas conexões seguras entre o Cliente e Servidor de Conteúdos através da utilização do protocolo HTTPS.

A utilização deste protocolo garante a comunicação fim-a-fim entre os dois parceiros de comunicação (Cliente e Servidor). O diagrama de implantação da figura 4.30 ilustra a arquitetura de comunicação entre o dispositivo móvel e o Servidor de Conteúdos, com a utilização do protocolo HTTPS.

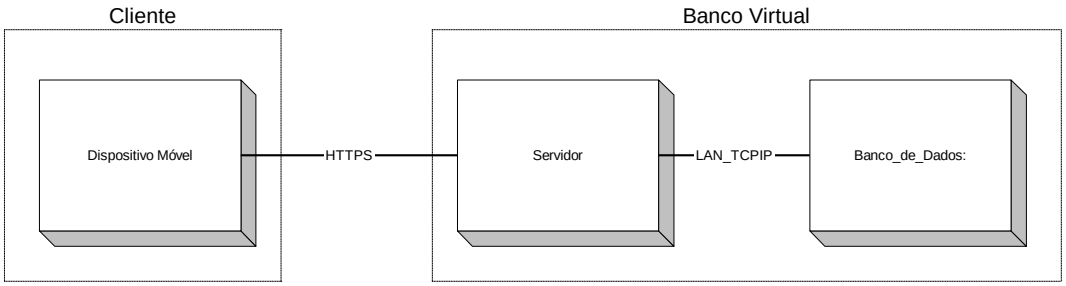

Figura 4.30: Diagrama de implantação do protótipo de *M-Banking*

O diagrama de implantação permite dar uma visão geral dos recursos de *hardware* e a comunicação existente entre os componentes da solução de *M-Banking*. No caso do Banco Virtual do protótipo proposto, tanto o Servidor HTTP quanto a Base de Dados coexistem no mesmo equipamento.

4.7 Aspectos de Segurança do Protótipo de M-Banking

Os aspectos de segurança que dão fundamento a este trabalho serão tratados a partir deste item. Serão abordados os procedimentos para geração de certificados para o Servidor de Conteúdo, o processo de assinatura dos *Midlet Suítes*, os aspectos relacionados à comunicação entre o Cliente e Servidor através da utilização do protocolo HTTPS, a rotina de autenticação do usuário na aplicação, os procedimentos necessários para gerenciamento de sessão HTTP e, por fim, um breve resumo do processo de *download* de aplicativos pela rede *wireless*.

4.7.1 Geração de Certificado para o Servidor

Para realizarmos conexões através do protocolo HTTPS com o Servidor é necessário que este possua um Certificado Digital. O Servidor TomCat utiliza este certificado para habilitar conexões seguras. Não serão abordados os passos para habilitação deste serviço no Servidor, maiores informações sobre este tema podem ser obtidas pelo *site* do projeto Apache http://jakarta.apache.org.

Quanto a criação de um certificado para o Servidor *Web*, foi utilizada uma ferramenta *Java* específica para geração de certificados chamada KEYTOOL. Após a geração do certificado e as devidas configurações no Servidor TomCat será possível estabelecer conexões pelo protocolo HTTPS.

O mesmo certificado gerado para o Servidor deverá ser importado para a aplicação de *M-Banking*, somente assim a aplicação poderá acessar as informações no Servidor. O processo de importação é realizado através de um utilitário específico do WTK20 para gerenciamento de certificados, utilitário este mostrado na figura 4.31.

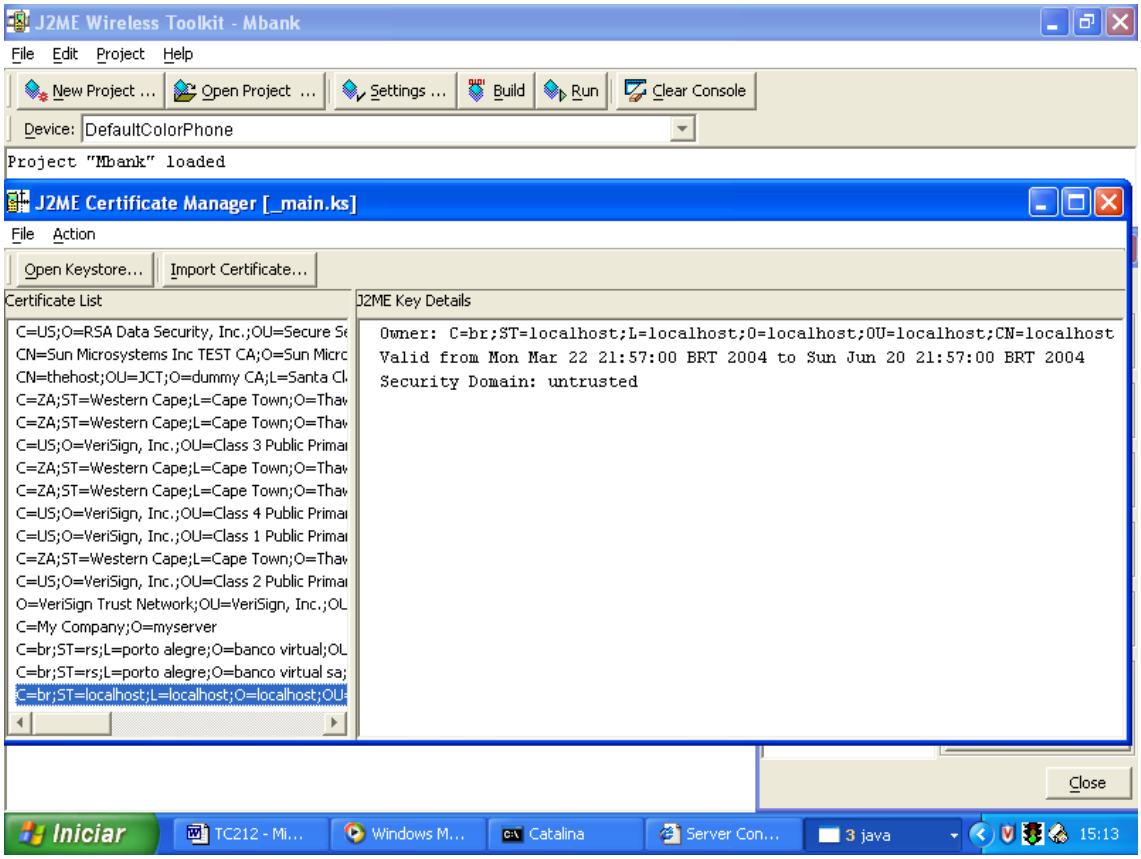

Figura 4.31: Visão do utilitário de gerenciamento de certificados do WTK20

Passada esta etapa já existem as condições necessárias para o protótipo de *M-Banking* acessar as informações no Servidor de Conteúdos.

4.7.2 Protocolo HTTPS

Como citado anteriormente o HTTP não é um protocolo seguro e pode sofrer ataques ou a ação de curiosos ou pessoas mal intencionadas. Neste Trabalho de Conclusão é utilizada uma alternativa mais segura para comunicação com Servidor de Conteúdo, o protocolo HTTPS. Este protocolo é umas das novas características do MIDP 2.0 e provê serviços de autenticação e encriptação de informações.

Estes serviços são realizados pelo SSL, TLS ou por protocolo similar. O GFC torna a atividade de criar conexões HTTPS muito fáceis. O trecho de código da figura 4.32 pertencente à classe que executa a consulta de Saldo do protótipo, demonstra como é feita a conexão HTTPS.

```
63    public void run() {
64        InputStream in = null;
65        OutputStream out = null;
66        HttpsConnection hc = null;
67
68        String url = getAppProperty("saldoMIDlet-URL");
69        System.out.println(url);
70        url = url+"?action=saldo&agencia="+age+"&"+"conta="+cont;
71        System.out.println(url);
72
73        try {
74            hc = (HttpsConnection)Connector.open(url);
75            hc.setRequestMethod(HttpsConnection.GET);
76            System.out.println(hc);
77
78            in = hc.openInputStream();
79            System.out.println(in);
80
81            int tamanho = (int)hc.getLength();
82            byte [] raw = new byte[tamanho];
83            System.out.println(tamanho);
84            in.read(raw);
85            System.out.println(raw);
86            String response = new String(raw);
87            textoVolta.setString(response);
88            System.out.println(response);
89            display.setCurrent(textoVolta);
90            in.close();
91            hc.close();
92        } catch (IOException ioe) {}
93    }
```

Figura 4.32: Exemplo de conexão HTTPS

Para receber dados de um Servidor é necessário passar o *Uniform Resource Locator* (URL) para o método estático *open()* (linha 74). A resposta da conexão realizada com o Servidor é recebida com a utilização do método *openInputStream()* (linha 78). A partir disso pode ser feita à leitura dos bytes recebidos (linhas 81 a 84). Os diversos métodos de conexão podem gerar exceções *Java* e, portanto, é necessário realizar o tratamento das mesmas através dos blocos *try* e *catch.*

4.7.3 Autenticação do Cliente no Sistema

Na plataforma J2SE, a *Sun MicroSystems* fornece suporte à criptografia através do *Java Cryptography Architecture* (JCA) e do *Java Cryptography Extension* (JCE). O problema é que tanto o JCA quanto o JCE são muito robustos para a plataforma J2ME/MIDP. Neste protótipo foi utilizado o pacote de criptografia da *Bouncy Castle*, que além ser um recurso de código aberto oferece uma distribuição específica de seu *software* para o J2ME.

Este pacote criptográfico foi utilizado no protótipo *M-Banking* para resolver o problema de autenticação do Cliente no Servidor. O *message digests* fornece uma maneira de resolver o problema de transmissão de senha. Ao invés de enviar a senha em texto claro são enviados para o Servidor: o número da agência, o número da conta, um número radômico, um valor de *timestamp* e o valor do *message digests*. O valor de

message digests é gerado a partir dos dados informados anteriormente mais o valor da senha informada pelo Cliente. O Servidor de posse das informações recebidas e da senha pré-cadastrada no sistema refaz o cálculo do *message digests*, se existir coincidência entre o valor de *message digests* recebido do Cliente e o valor de *message digests* calculado pelo Servidor o usuário é autenticado no sistema. A figura 4.33 mostra o processo de autenticação por *message digests*.

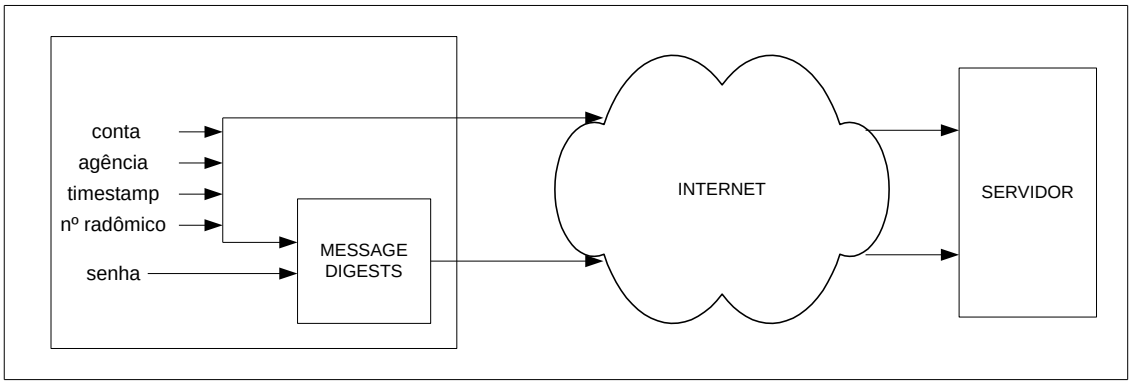

Figura 4.33: Processo de autenticação por *message digests*
Fonte: Knudsen (2003)

Na figura 4.34 ilustrada a classe codSenha responsável por calcular o valor de *message digests* e montar o URL da requisição de autenticação no sistema.

```
15
16 import org.bouncycastle.crypto.Digest;
17 import org.bouncycastle.crypto.digests.SHA1Digest;
18 import java.io.*;
19 import java.util.Random;
20
21 public class codSenha {
22     private Random mRandom;
23
24     public String converte(String url, String ag, String conta, String password) {
25         mRandom = new Random(System.currentTimeMillis());
26         long timestamp = System.currentTimeMillis();
27         long randomNumber = mRandom.nextLong();
28         byte[] agBytes = ag.getBytes();
29         byte[] contaBytes = conta.getBytes();
30         byte[] passwordBytes = password.getBytes();
31         byte[] timestampBytes = getBytes(timestamp);
32         byte[] randomBytes = getBytes(randomNumber);
33
34         Digest digest = new SHA1Digest();
35         digest.update(agBytes, 0, agBytes.length);
36         digest.update(contaBytes, 0, contaBytes.length);
37         digest.update(timestampBytes, 0, timestampBytes.length);
38         digest.update(randomBytes, 0, randomBytes.length);
39         digest.update(passwordBytes, 0, passwordBytes.length);
40         byte[] digestValue = new byte[digest.getDigestSize()];
41         digest.doFinal(digestValue, 0);
42
43         URLBuilder ub = new URLBuilder(url);
44         ub.addParameter("agencia", ag);
45         ub.addParameter("conta", conta);
46         ub.addParameter("timestamp", new String(HexCodec.bytesToHex(timestampBytes)));
47         ub.addParameter("random",new String(HexCodec.bytesToHex(randomBytes)));
48         ub.addParameter("digest",new String(HexCodec.bytesToHex(digestValue)));
49         url = ub.toString();
50         return url;
51     }
```

Figura 4.34: Classe codSenha

Todos os dados informados pelo usuário (agência, conta e senha), mais os valores de *timestamp* e o valor randômico são convertidos para um *array* de *bytes* (linha 25 a 32). Após este processo, os *bytes* resultantes são passados para a função *digest.update* do pacote *Bouncy Castel* (linhas 34 a 39) para posteriormente ser gerado uma valor de digest (linha 41). A classe URLBuilder é utilizada para montar a URL que fará a solicitação de autenticação no Servidor de Conteúdo. Um exemplo de URL gerada pela classe codSenha seria: https://localhost:8443/banco/servlet/LoginServlet?agencia=0001&conta=111111&timestamp=000000fc8a83333f&random=68b530dcfffe3b21&digest=b14cf4f045161d64db8183c22fead1431adfec15.

No Servidor de Conteúdo o processo de autenticação é muito semelhante. A próxima figura ilustra este processo.

```
20
21      String agencia = request.getParameter("agencia");
22      String conta = request.getParameter("conta");
23      String password = lookupPassword(agencia,conta);
24      String timestamp = request.getParameter("timestamp");
25      String randomNumber = request.getParameter("random");
26
27      byte[] agenciaBytes = agencia.getBytes();
28      byte[] contaBytes = conta.getBytes();
29      byte[] timestampBytes = HexCodec.hexToBytes(timestamp);
30      byte[] randomBytes = HexCodec.hexToBytes(randomNumber);
31      byte[] passwordBytes = password.getBytes();
32
33      Digest digest = new SHA1Digest();
34      digest.update(agenciaBytes, 0, agenciaBytes.length);
35      digest.update(contaBytes, 0, contaBytes.length);
36      digest.update(timestampBytes, 0, timestampBytes.length);
37      digest.update(randomBytes, 0, randomBytes.length);
38      digest.update(passwordBytes, 0, passwordBytes.length);
39      byte[] digestValue = new byte[digest.getDigestSize()];
40      digest.doFinal(digestValue, 0);
41
42      String message = "";
43      String clientDigest = request.getParameter("digest");
44      if (isEqual(digestValue, HexCodec.hexToBytes(clientDigest))) {
45        message = "Cliente_OK";
46        session.setAttribute("logado",new String("true"));
47      } else
48        message = "Login sem sucesso.";
49      System.out.println(message);
50
51      response.setContentType("text/plain");
52      response.setContentLength(message.length());
53      PrintWriter out = response.getWriter();
54      out.println(message);
55    }
```

Figura 4.35: *Servlet* LoginServlet

A requisição do dispositivo móvel é recebida pelo Servidor e todos os parâmetros são convertidos para um *array* de *bytes* (linha 21 a 31). Na linha 23 é executado um método que busca da base de dados a senha correspondente à agência e conta informada na requisição. É calculado um valor de *digest* a partir dos dados recebidos e da senha obtida na base de dados (linha 33 a 40). Por último, de posse do valor de *digest* calculado e o valor recebido do Cliente é feita a comparação entre os dois valores (linha 43 a 48). Se os valores forem iguais é liberado o acesso do usuário, caso negativo, é enviada uma mensagem de falha no *login* para o Cliente (linhas 45 e 48).

Para aprimorar ainda mais o processo de autenticação no sistema, Knudsen (2003) sugere que o valor de *timestamp* gerado no momento da autenticação do usuário seja guardado em uma base de dados. Se o valor gerado na atual tentativa for superior

ao valor ao valor guardado o usuário é autenticado e novo valor de timestamp é atualizado na base de dados.

4.7.4 Gerenciamento de Sessão

O HTTP é um protocolo *stateless*, isso significa que cada par de uma solicitação (requisição/resposta) é uma conversação separada. Em algumas situações, como no caso do protótipo de *M-Banking*, é interessante que o Servidor saiba com quem ele está trocando informações. Isso pode ser feito através do gerenciamento de sessão. Do lado do Servidor, uma sessão é somente uma coleção de informações. Quando o Cliente manda uma requisição HTTP para o Servidor, ele inclui um identificador de sessão (ID de sessão) como parte da requisição. O Servidor verifica esta informação e tem condições de identificar o usuário, ou no mínimo verificar o status da sessão indicada.

A maneira mais comum de realizar o gerenciamento de sessão é através dos *cookies*. Os *Web Browsers* executam esta atividade automaticamente. No mundo MIDP não existem *Web Browsers* fazendo este gerenciamento, portanto esta necessidade deve ser implementada no código da aplicação. A figura 4.36 ilustra os passos necessários para realização do processo de armazenamento do Id de sessão.

```
70          try {
71              hc = (HttpsConnection)Connector.open(url);
72              if (mSession != null) {
73                  hc.setRequestProperty("cookie",mSession);
74              }
75              hc.setRequestMethod(HttpsConnection.GET);
76              System.out.println(hc);
77              SecurityInfo si = hc.getSecurityInfo();
78              Certificate c = si.getServerCertificate();
79              String subject = c.getSubject();
80              System.out.println(subject);
81              in = hc.openInputStream();
82
83              String cookie = hc.getHeaderField("Set-cookie");
84              if ( cookie != null) {
85                  int semicolon = cookie.indexOf(';');
86                  mSession = cookie.substring(0,semicolon);
87              }
88
```

Figura 4.36: Gerenciamento de sessão.

A verificação da existência ou não do cookie é realizada entre as linhas 84 e 87, se existir o cookie ele é armazenado na variável mSession. Sempre que receber uma resposta do Servidor, será realizada uma nova verificação por *cookies* (linhas 72 a 74). As informações guardadas na variável mSession no momento do *login* são enviadas para todos os objetos criados, a fim de manter a consistência do processo de gerenciamento de sessão. Isso pode ser observado nos próximos trechos de código.

```
75          } else if (menuselect.equals("Saldo")) {
76                      saldoConta.atualizaAgCon(age,cont,mSession);
77                      saldoConta.startApp();
```

Figura 4.37: Passagem do parâmetro de gerenciamento de sessão mSession.

```
106     public void atualizaAgCon(String ag, String con, String sess) {
107         age = ag;
108         cont = con;
109         mSession = sess;
110     }
111
112 }
```

Figura 4.38: Recepção do parâmetro de gerenciamento de sessão mSession.

Outro aspecto importante no processo de gerenciamento de sessão é o tratamento realizado pelo Servidor de Conteúdo. Existem dois momentos básicos: a inserção da informação de usuário conectado (LoginServlet) e a recuperação e teste desta condição nas demais *Servlets* da aplicação.

Quando o usuário envia para o Servidor de Conteúdo seus dados de autenticação e ocorre uma autenticação bem sucedida do mesmo é criado o cookie da aplicação e é inserido um atributo que informa que o mesmo está conectado para os demais Servlets, este processo pode ser verificado na figura 4.39 (linhas 65 e 66).

```
63      if (isEqual(digestValue, HexCodec.hexToBytes(clientDigest))) {
64        message = "Cliente_OK";
65        HttpSession session = request.getSession(true);
66        session.setAttribute("logado",new String("true"));
67      } else
68        message = "Login sem sucesso.";
```

Figura 4.39: Criação e atribuição de valor ao *cookie* de sessão

Todos os Servlets do protótipo de *M-Banking* realizam a procura e o teste do atributo “logado”, se o valor associado a este atributo for diferente de “*true*” ou não existir uma sessão associada ao usuário será exibida a mensagem “Área não autorizada” e não será possível continuar com a aplicação. Este processo pode ser verificado na figura 4.40.

```
32      if ( session == null) {
33          message = "Área não autorizada!!!";
34          response.setContentType("text/plain");
35          response.setContentLength(message.length());
36          PrintWriter out = response.getWriter();
37          out.println(message);
38      } else {
39          String logado = (String) session.getAttribute("logado");
40          if ( !logado.equals("true") ) {
41              message = "Área não autorizada!!!";
42              response.setContentType("text/plain");
43              response.setContentLength(message.length());
44              PrintWriter out = response.getWriter();
45              out.println(message);
46          }
47      }
```

Figura 4.40: Teste de valor "logado" no *cookie* de gerenciamento de sessão

Das técnicas de gerenciamento de sessão apresentadas por Kurniawan (2002), a técnica de Objetos *Session* é a mais flexível e mais poderosa. Para cada usuário do protótipo é criado um objeto HttpSession. Este objeto ficará sempre associado ao usuário que o criou e, portanto, somente este usuário possuirá acesso as informações constantes nele. Este procedimento possibilita que usuários não possam acessar informações de sessão de outros usuários, garantindo a confidencialidade das informações trocadas entre o Cliente e o Servidor.

4.7.5 Execução do Protótipo em um Dispositivo Real

Como pode ser verificado no item 4.4, os grandes fabricantes de telefones celulares estão desenvolvendo dispositivos com suporte ao *profile* MIDP. Mas o que é necessário para baixar um aplicativo J2ME em um dispositivo real? Existem duas possibilidades de transferir *Midlets Suítes* para o dispositivo móvel: a primeira através da comunicação serial do micro-computador ou através da rede *wireless*. A segunda possibilidade é chamada de *Over the Air Provisioning* (OTA). Esta funcionalidade deve ser proporcionada pelo dispositivo, que deverá possuir mecanismos que permitirão a descoberta e instalação das aplicações. Em alguns casos esta descoberta por ser feita por um *browser* residente no dispositivo, em outros através de uma aplicação escrita especialmente para esta finalidade, mais conhecida com *Discovery Application* (DA). Este processo dá plenos poderes ao usuário para baixar, remover, verificar versões de *software* e ser informado sobre o progresso de *download* de aplicativos. Maiores informações podem ser obtidas pelo *site* http://jcp.org/jsr/detail/37.jsp.

5 CONCLUSÃO

Depois do surgimento do J2SE e do J2EE, o J2ME aparece como a terceira revolução na curta história do *Java*, iniciada em 1995. O mercado de pequenos dispositivos está em franca expansão e o *Java* tem uma característica muito importante que o diferencia de outras tecnologias. No J2ME os desenvolvedores podem escrever um código e o mesmo pode processar em vários dispositivos diferentes sem qualquer tipo de alteração.

Apesar de simples e possível, este protótipo não pretende abranger todas as funcionalidades de um sistema bancário normal, tais como aplicações em fundos de investimento, agenda, solicitação de cartão de débito e talões de cheques. O objetivo aqui é demonstrar que é possível criar uma aplicação de *M-Commerce* robusta e com bom nível de segurança sem a necessidade de se reinventar a roda. Este protótipo simplesmente transportou para o mundo dos dispositivos móveis tecnologias que são amplamente usadas no mundo dos dispositivos com maior capacidade de processamento (Desktops, Notebooks, etc), tais como: utilização de certificados digitais, comunicação HTTPS e desenvolvimento de aplicações orientadas a objeto.

Muitos avanços ocorreram depois que este protótipo foi desenvolvido e podem ser abordados em futuros projetos tais como: XML, SOAP WebServices e os pacotes criptográficos para dispositivos móveis. Uma característica crescente do J2ME é a sua utilização como base para desenvolvimento de *games* com boa jogabilidade para dispositivos de pequeno porte, talvez ai uma sugestão para um novo TCC. Exemplos deste tipo de aplicação podem ser obtidos no *site* http://www.midlet.org. Uma dificuldade encontrada continua ser a pequena quantidade de trabalhos e literatura confiáveis em português que abordem o assunto. Por fim, é sugerida a visita ao *site* da *Sun MicroSystems* (*http://www.javasoft.com*) as pessoas interessadas em adquirir maiores conhecimentos sobre a tecnologia *Java* e seus diversos componentes.

REFERÊNCIAS

ANATEL. **Brasil tem mais de 133 milhões de linhas móveis.** Disponível em: <http://www.anatel.gov.br/Portal/exibirPortalNoticias.do?acao=carregaNoticia&codigo=16383>. Acesso em: 05 set. 2008.

AREHART,C. et al. **Professional WAP** . São Paulo: Makron, 2001. 773p.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 17799:** tecnologia da Informação: técnicas de segurança: código de prática para a gestão de segurança da informação**.** Rio de Janeiro, 2005.

AYRES, R. **Conectando a Empresa à Rede.** Notas de Aula. Disponível em: <http://www.rosimeireayres.com.br/arquivos/aulas/TI/aula2_B2B_B2C_B2E_C2C.pdf> . Acesso em: 10 set. 2008.

BRIAN E.M. et al. **Mobile Commerce - Technology, Theory and Applications**. Hershey, USA: Idea Group, 2003. 276p.

CABRAL, J.L.M.; LEITE, L.M. **Segurança em Transações e Aplicações WAP.** Disponível em: < http://www.wirelessbrasil.org/wirelessbr/colaboradores/cabral_leite/seg_wap_01.html >. Acesso em: 06 jun. 2008.

CARVALHO, C.C. **LearnX:** Desenvolvimento do Módulo Jogo Educacional para o Ambiente de Aprendizagem Cooperativo via *Web*. 2002. 66p. Monografia (Trabalho de Conclusão de Curso) – Curso de Ciência da Computação, Centro Universitário La Salle, Canoas.

ECOMMERCEORG. **Desempenho do Comércio Eletrônico no Brasil**. Disponível em: <http://www.e-commerce.org.br/STATS.htm>. Acesso em: 05 set. 2008.

GUPTA, V. et al. **Wireless Programming with J2ME**. New York, USA: Hungry Minds, 2002. 376p.

INTERNET WORLD STATS. **World Internet Users and Population Stats** Disponível em: <http://www.internetworldstats.com/stats.htm>. Acesso em: 05 set. 2008.

JANSEN, S.B. **Estudo Comparativo das Principais Plataformas de Desenvolvimento para Dispositivos Moveis.** 2003. 49p. Monografia (Trabalho de Conclusão de Curso) – Centro de Informática, Universidade Federal de Pernambuco, Recife.

JBENCHMARK. **Java benchmark for MIDP 2.0 Devices.** Disponível em: <http://www.jbenchmark.com/result.jsp?benchmark=v2>. Acesso em: 05 set. 2008.

KEOGH, J. **J2ME:** The Complete Reference. Berkeley, USA: McGraw-Hill Osborne Media, 2003. 744p.

KNUDSEN, J. **Wireless Java**: Developing with Java 2 Micro Edition. 2nd ed. Berkeley, USA: Apress, 2003. 364.

KNUDSEN, J. **MIPD Application Security 1 Design Concerns and Cryptography.** Disponível em: <http://developers.sun.com/mobility/midp/articles/security1/>. Acesso em: 02 set. 2008.

KURNIAWAN, B. **Java para a Web com Servlets, JSP e EJB**. Rio de Janeiro, Brasil: Ciência Moderna, 2002. 807p.

LESSA, R. **Estudo sobre a Internet Móvel e o M-Commerce**. 2001. 88p. Monografia (Trabalho de Conclusão de Curso) - Faculdade de Informática, Universidade Luterana do Brasil, Gravataí.

NORRIS, M. **Mobile IP Technology for M-Bussines.** Norwood, USA: Artech House, 2001. 291p.

RUUSKANEN, J.P. **Java 2 Plataform Micro Edition.** Disponível em : < http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.9986 >. Acesso em: 10 ago. 2008.

PIROUMIAN, V. **Wireless J2ME:** Platform Programming. Palo Alto, USA: Prentice Hall, 2002. 293p.

STALLIGS, W. **Cryptography and Network Security:** Principles and Pratice. 2nd ed. New Jersey USA: Prentice Hall, 1999 569p.

SUN MICROSYSTEMS. **Mobile Information Device Profile Specification**. Disponível em: <http://www.javasoft.com>. Acesso em: 01 maio 2008.