

1 TECNOLOGIA J2ME

A J2ME, *Java Micro Edition*, é uma tecnologia que permite o desenvolvimento de aplicações Java para dispositivos com poder de processamento, vídeo e memória limitados. Possui uma coleção de APIs, *Application Program Interface*, definida pela comunidade JCP, *Java Community Process*, específicas para dispositivos compactos como Celulares, PDAs, *Personal Digital Assistants*, entre outros. A J2ME trata das necessidades especiais dos dispositivos para consumidor, das quais as edições J2SE, *Java Standard Edition*, e J2EE, *Java Enterprise Edition*, não abrangem (MUCHOW, 2004). A J2SE é a base das três edições voltada principalmente para computadores pessoais. Já a J2EE complementa a J2SE voltada principalmente para servidores (COUTINHO, 2005). Segundo Muchow (2004) as edições de Java são definidas como:

Standard Edition (J2SE): projetada para execução em máquinas simples de computadores pessoais e estações de trabalho. Enterprise Edition (J2EE): com suporte interno para Servlets, JSP e XML, essa edição é destinada a aplicativos baseados no servidor. Micro Edition (J2ME): projetada para dispositivos com memória, vídeo e poder de processamentos limitados (MUCHOW, 2004, p.2).

A JCP especificou a J2ME em dois grupos, conforme as necessidades dos dispositivos, chamados de Configuração sendo denominados CDC, *Connected Device Configuration*, e CLDC, *Connected Limited Device Configuration*. O primeiro para dispositivos com maior capacidade computacional e normalmente fixos como um computador ligado à TV, por exemplo, o segundo para aqueles dispositivos com menor capacidade computacional e normalmente móveis.

Dentro da Configuração existe outra classificação, os Perfis, que objetiva agrupar dispositivos de uma mesma configuração onde as aplicações possam ser portáteis sem perder funcionalidade. MIDP, *Mobile Information Device Profile*, é o perfil destinado a celulares, outros perfis estão sendo especificados pela JCP (GOMES, 2007).

Uma máquina virtual Java específica foi desenvolvida pela Sun para a CLDC chamada KVM, *K Virtual Machine* que manipula considerações especiais destes dispositivos.

Por fim, a J2ME encontra-se hoje em intensa evolução e sua utilização já está mudando e revolucionando o mercado e a forma humana de se viver e relacionar.

1.1 Configurações

Como dito anteriormente, a JCP especificou dois grupos para a J2ME conforme as necessidades dos dispositivos, chamadas Configurações:

Uma configuração define uma plataforma Java para uma ampla variedade de dispositivos. Na verdade, uma configuração define os recursos da linguagem Java e as bibliotecas Java Básicas da JVM para essa configuração em particular (MUCHOW, 2004, p.3).

As necessidades relacionadas são: memória, vídeo, conectividade de rede e poder de processamento. Na Tabela 1.1 são apresentadas as principais diferenças entre os dois tipos de configurações:

Tabela 1.1 Diferenças entre CDC e CLDC (Fonte: GOMES, 2007)

| CDC | CLDC |
|--|---|
| Largura de Banda Persistente e Alta, Redes de Alto Desempenho. | Display Reduzido. |
| Dispositivos com Maior Capacidade Computacional, <i>High-End Consumer Devices</i> . | Conectividade Intermitente. |
| Discos com Grande Capacidade de Armazenamento. | Pouca Capacidade de Armazenamento. |
| Grande Capacidade de Memória. | Pouca Capacidade de Memória. |
| Fonte Permanente de Energia. | Fonte Limitada de Energia (Baterias). |
| <i>Set-Top Boxes</i> de TVs a Cabo (Digitais), Sistemas Automotivos, Eletrodomésticos. | Celulares, Smartphones, Blackbarrys, PDAs, Pagers, etc. |

Vale lembrar que atualmente uma crescente convergência tecnológica está havendo nas indústrias e, devido a essas “antigas” e futuras semelhanças e diferenças muito provavelmente essa classificação poderá ser alterada a qualquer momento (COUTINHO, 2005).

O enfoque desta pesquisa será dado à CLDC, pois se aplica diretamente ao desenvolvimento deste trabalho.

1.1.1 CLDC

A CLDC define uma especificação para uma JVM, *Java Virtual Machine*, um conjunto de classes Java e também uma plataforma padrão mínima para dispositivos pequenos. Portanto, especifica uma configuração mínima em termos de hardware e bibliotecas padrão para o dispositivo (MUCHOW, 2004).

Existem no mercado diversos tipos de hardware e software disponíveis para os dispositivos suportados pela CLDC e, portanto esta deve ser capaz de lidar com toda essa variedade. Para tanto, o único requisito de hardware especificado pela CLDC é a memória. Quanto ao Software, os requisitos incluem a capacidade de o dispositivo executar uma JVM e gerenciar aplicativos Java – selecionar, ativar e remover.

Existem algumas diferenças em relação à J2SE, entre elas: a Matemática de Ponto Flutuante não é suportada na Linguagem J2ME e na JVM, pois é extremamente dispendiosa para o processador desses dispositivos; o método *finalize()*¹ também não é suportado para a linguagem pelo mesmo motivo; o suporte a APIs nativas de outras Linguagens de Programação também foi eliminado na JVM para reduzir os requisitos de memória; quanto ao Tratamento de Erros, a JVM suporta apenas um conjunto pequeno devido a sobrecarga do sistema e também aos sistemas incorporados nos dispositivos; um Carregador de Classes faz-se necessário, este sendo definido e implementado pelo próprio fabricante do dispositivo; a JVM para a CLDC não dispõe de Reflexão², suporte a Thread Groups (as threads são processadas objeto por objeto) e Referências Fracas (um objeto sendo referenciado é candidato à coleta de lixo) (MUCHOW, 2004).

Segurança na troca de informações entre aplicações em redes diferentes é uma das preocupações e também promessas da plataforma Java. Para tanto a verificação de arquivo de classe se faz necessária. O atual gerenciador de segurança do J2SE não pôde ser implementado completamente para a CLDC porque requer muita memória. No entanto, um modelo chamado Caixa de Areia ou *Sandbox* é utilizado. Neste modelo a aplicação é limitada em um ambiente pelas APIs definidas pelos Perfis e Configurações.

Muitas classes foram cortadas, modificadas e especificadas para a CLDC devido às características dos dispositivos. A maioria das bibliotecas são na verdade subconjuntos das bibliotecas originais e aqui se incluem *java.lang.**, *java.util.** e *java.io.**. Especificamente desenvolvida para CLDC temos a *java.microedition.** (PITONI, 2001).

¹ Neste método códigos para fazer a limpeza de recursos alocados podem ser inseridos

² Classes para obter informações sobre a VM

1.2 Perfis

Um perfil fornece bibliotecas ainda mais específicas que as bibliotecas CLDC e, portanto é considerada uma extensão de configuração. Com isso aplicações mais completas são escritas.

Tabela 1.2 Arquitetura do Perfil MID (Fonte: MUCHOW, 2004).



MIDP é o perfil utilizado com a configuração CLDC. Conforme Tabela 1.2, o Perfil está dentro da CLDC, confirmando a afirmação de que um Perfil é uma extensão da CLDC.

O enfoque será dado à MIDP, pois se aplica diretamente ao desenvolvimento deste trabalho.

1.2.1 MIDP

O MIDP estende a CLDC, aumentando e criando alguns requisitos mínimos de hardware e software. O MIDP foi feito para rodar em cima do CLDC. As Tabelas 1.3 e 1.4 listam os requisitos mínimos de Hardware e Software respectivamente.

Tabela 1.3 Requisitos Mínimos MIDP: hardware (Fonte: PITONI, 2001)

| HARDWARE | |
|----------------|---|
| <i>Display</i> | 96 x 54 1bit profundidade 1:1 formato do pixel |
| <i>Input</i> | Teclado de uma Mão Teclado de duas Mãos QWERTY |
| Memória ROM | 128 Kb para MIDP 8 Kb dados de aplicações |
| Memória RAM | 32 Kb para executar o Java |
| Rede | <i>Duplex</i> Sem fio Intermitente Largura de Banda Limitada |
| Som | Mono 8khz |

Tabela 1.4 Requisitos Mínimos MIDP: Software (Fonte: MUCHOW, 2004).

| SOFTWARE |
|--|
| <ul style="list-style-type: none"> • Recursos suficientes para executar a JVM; • Tratamento de Exceções; • Processamento de Interrupções; • Escrita de Elementos Gráficos <i>Bitmap</i> na Tela; • Acesso de Leitura e Escrita à Rede sem Fio; • Mecanismo para Capturar Entrada de um <i>Input Device</i>; • Recursos para Ler e Gravar em Memória não Volátil para Suporte de Dados Persistentes; |

O MIDP também especifica algumas APIs Java além daquelas exigidas pela CLDC. Algumas APIs J2ME são subconjuntos de APIs J2SE e outras são completas, pois são aquelas específicas à J2ME, ou seja, pode-se remover classes e métodos menos essenciais, porém o que sobrar será igual (DOEDERLEIN, 2007).

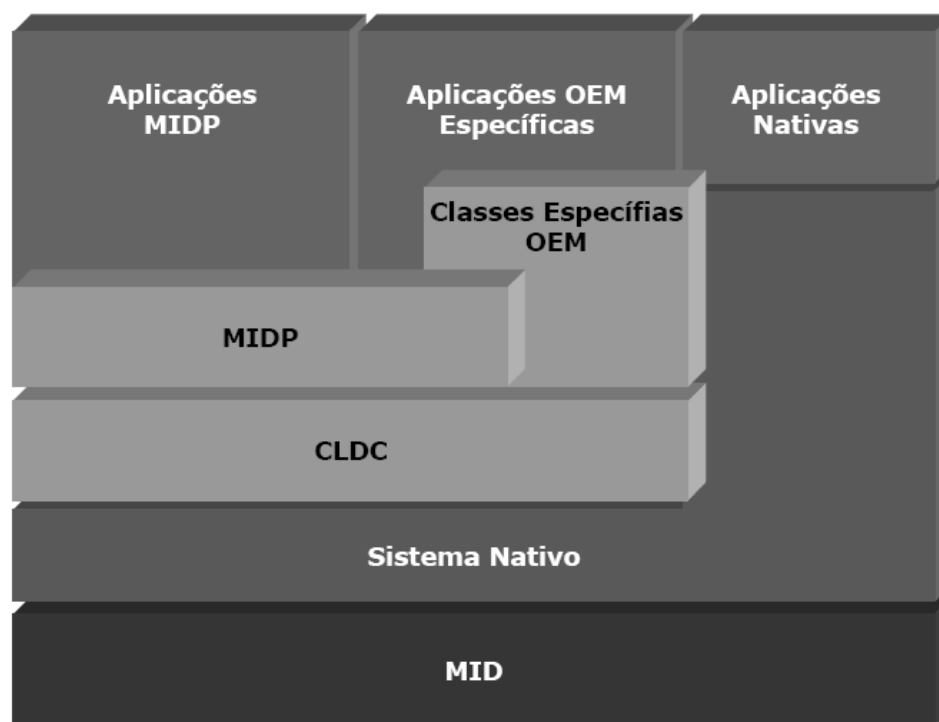


Figura 1.1: Arquitetura MID (Fonte: GALL).

No nível inferior, encontra-se o Hardware, no segundo nível, o Sistema Operacional Nativo do Dispositivo. Os Aplicativos Nativos, no canto superior direito da figura, antes de J2ME, eram os únicos tipos de programas para estes dispositivos, a CLDC é instalada no nível do Sistema Operacional Nativo e é a base do MIDP. Neste Bloco encontra-se a KVM que permitirá às APIs Java de alto nível serem construídas. Conforme a Arquitetura, MIDP tem acesso às suas próprias bibliotecas e também às bibliotecas CLDC. As classes OEM, *Original Equipment Manufacturer*, são fornecidas pelo fabricante, normalmente são não portáteis, são específicas para o dispositivo e rodam em cima da CLDC. Já os Aplicativos OEM específicos podem acessar APIs MIDP e (ou) classes específicas do OEM. Os Aplicativos MIDP em questão são os MIDlet, ou seja, aplicativos Java projetados para serem executados em um dispositivo móvel (MUCHOW, 2004)

Quanto à Rede no MIDP, este por herdar a conectividade do CLDC, pode ser implementado pelo TCP/IP, *Transmission Control Protocol / Internet Protocol*. É possível

também armazenar dados para posterior leitura através do mecanismo RMS, *Record Management System*, que para as MIDlet são o *Record Stores* e os *Records*. As Classes *Timer* e *TimerTask* são usadas para aplicativos que necessitem agendar tarefas (PITONI, 2001).

O software de Gerenciamento de Aplicação deve implementar as funções para instalação, execução, seleção e remoção de MIDlets, sendo assim, os elementos a seguir devem estar disponíveis quando uma MIDlet iniciar pelo software (MUCHOW, 2004):

- Acesso à CLDC e à KVM;
- Acesso às classes definidas pelo MIDP;
- Acesso ao arquivo JAR (*Java Archive*);
- Acesso ao arquivo JAD (*Java Application Descriptor*).

1.2.2 MIDlet

Uma MIDlet, como dito na seção anterior, é um aplicativo Java projetado para ser executado em um dispositivo móvel consistindo de uma ou mais MIDlets empacotadas por um arquivo JAR sendo, uma MIDlet, construída pela classe MIDlet. Desta forma é necessário conhecer o ambiente de desenvolvimento e os fundamentos mais básicos de uma MIDlet, ambos serão apresentados no decorrer desta seção.

O arquivo JAR empacota todos os arquivos e classes Java dentre outras informações necessárias para uma MIDlet ou um conjunto de MIDlets. Um arquivo chamado manifesto descreve o conteúdo do arquivo JAR e está dentro dele próprio. Já um arquivo JAD fornece informações sobre a(s) MIDlet(s) dentro do arquivo JAR e deve estar disponível como parte do conjunto de MIDlets dentro do arquivo JAR (MUCHOW, 2004).

A Tabela 1.5 lista os atributos de um arquivo manifesto. Na primeira coluna estão os atributos, a segunda coluna o objetivo de cada atributo e a última coluna especifica quais atributos são exigidos e quais podem deixar de ser definidos.

Tabela 1.5: Atributos do arquivo manifesto (Fonte: MUCHOW,2004).

| ATRIBUTOS DO ARQUIVO DE MANIFESTO | | |
|-----------------------------------|--|---------|
| ATRIBUTO | OBJETIVO | EXIGIDO |
| MIDlet-Name | Nome do conjunto de MIDlets. | Sim |
| MIDlet-Version | Número de versão da MIDlet. | Sim |
| MIDlet-Vendor | Quem desenvolveu a MIDlet. | Sim |
| MIDlet-< n > | Referência a uma MIDlet específica dentro de um conjunto de MIDlets. Esse atributo contém até três informações: 1. Nome da MIDlet; 2. Ícone dessa MIDlet (opcional); 3. Nome da classe que o gerenciador de aplicativos chamará para carregar essa MIDlet. | Sim |
| MicroEdition-Profile | Qual perfil do JME é exigido pela MIDlet. | Sim |
| MicroEdition-Configuration | Qual configuração do JME é exigida pela MIDlet. | Sim |
| MIDlet-Icon | Ícone usado pelo gerenciador de aplicativos. Mostrado junto da MIDlet-Name no dispositivo. Esse deve ser um arquivo de imagem PNG. | Não |
| MIDlet-Description | Texto descrevendo a MIDlet. | Não |
| MIDlet-URL | URL que pode ter mais informações sobre a MIDlet e/ou sobre o fornecedor. | Não |

Um arquivo JAD deve fornecer informações para o gerenciador de aplicativos sobre o conteúdo de um arquivo JAR, facilitando as decisões e, também deve fornecer um meio de passar parâmetros para as MIDlets sem que o arquivo JAR seja modificado. Na Tabela 1.6 são apresentados os atributos de um arquivo JAD, que podem ser definidos pelo desenvolvedor desde que não se utilize a palavra **MIDlet-** (MUCHOW, 2004).

Tabela 1.6: Atributos do arquivo JAD (Fonte: MUCHOW,2004).

| ATRIBUTOS DO ARQUIVO JAD | | |
|--------------------------|--|---------|
| ATRIBUTO | OBJETIVO | EXIGIDO |
| MIDlet-Name | Nome do conjunto de MIDlets. | Sim |
| MIDlet-Version | Número de versão da MIDlet. | Sim |
| MIDlet-Vendor | Quem desenvolveu a MIDlet. | Sim |
| MIDlet-< n > | Referência a uma MIDlet específica dentro de um conjunto de MIDlets. Esse atributo contém até três informações: 1. Nome da MIDlet; 2. ícone dessa MIDlet (opcional); 3. Nome da classe que o gerenciador de aplicativos chamará para carregar essa MIDlet. | Sim |
| MIDlet-Jar-URL | URL do arquivo JAR. | Sim |
| MIDlet-Jar-Size | O tamanho do arquivo JAR, em bytes. | Sim |
| MIDlet-Data-Size | O número mínimo de bytes exigido para armazenamento de dados persistentes. | Não |
| MIDlet-Description | Texto descrevendo a MIDlet. | Não |
| MIDlet-Delete-Confirm | Mensagens mostradas para um usuário confirmar um pedido de exclusão de um conjunto MIDlet. | Não |
| MIDlet-Install-Notify | URL para receber relatórios de status de instalação. | Não |

Uma observação importante nesta tabela é a duplicidade dos atributos: se os atributos forem duplicados nos arquivos JAD e Manifesto, ou seja, se existem atributos iguais em ambos os arquivos, terá precedência os nomes e valores do arquivo JAD. Os três primeiros atributos da tabela devem ser idênticos nos dois arquivos, caso contrário o arquivo JAR não será carregado (MUCHOW, 2004).

O desenvolvimento de MIDlets por linha de comando exige vários passos: escrita do código java, compilação e verificação prévia, execução, empacotamento, criação dos arquivos JAR e JAD, execução da MIDlet em um simulador e download da MIDlet no dispositivo móvel. Tudo isto torna o desenvolvimento cansativo, principalmente se o projeto está tornado-se complexo e grande. Há também a possibilidade de gerenciar o projeto por pacotes java por linha de comando, gerando ainda mais carga de trabalho. Se tudo estiver organizado, utilizando as versões MIDP e CLDC correspondentes, o software funcionará sem problemas. Para facilitar o desenvolvimento, poupando boa parte desses passos, foram desenvolvidos um Kit e também plugins para IDEs. A *Java Sun Wireless Toolkit 2.5 for CLDC* (WTK) faz praticamente todo o trabalho, ficando para o desenvolvedor apenas a escrita do código java. Eclipse e NetBeans são IDEs que agora possuem plugins para suporte à MIDlets. Uma hierarquia básica de pastas para um projeto pode ser vista na Figura 1.2 (MUCHOW, 2004).

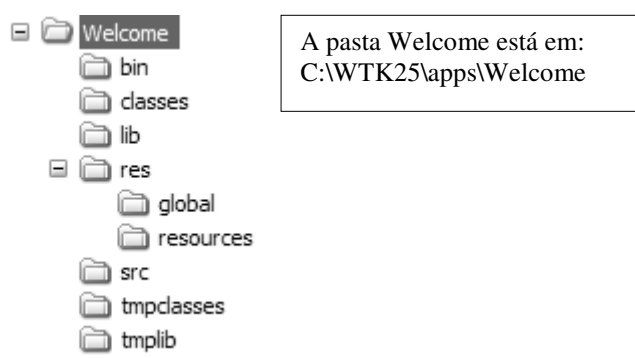


Figura 1.2: Hierarquia básica de um projeto (Fonte: MUCHOW, 2004).

A pasta bin armazena os arquivos de manifesto, JAD e JAR. Na pasta classes encontram-se os arquivos de classes gerados pelo compilador java. Em res ficam todos os arquivos de recursos e em src os arquivos de código-fonte java. Na pasta tmpclasses são armazenados os arquivos de classes verificados previamente. Todas essas pastas são criadas automaticamente pelo WTK (ou pela IDE) (MUCHOW, 2004).

Uma MIDlet possui um ciclo de vida. A Figura 1.3 apresenta os estados de uma MIDlet que são representados como métodos (COUTINHO, 2005).

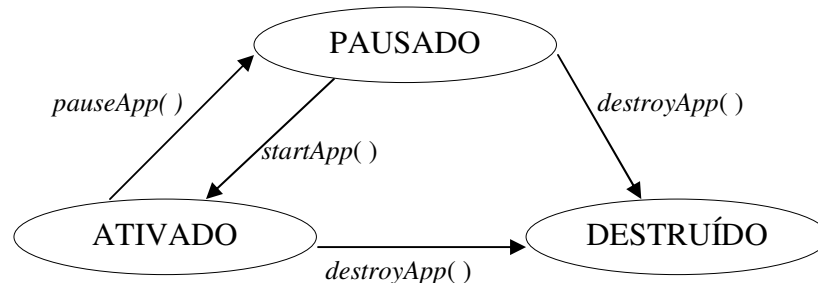


Figura 1.3: Ciclo de Vida MIDlet (Fonte: COUTINHO, 2005).

A comunicação do Gerenciador de Aplicativos com a MIDlet acontece pela Classe MIDlet `javax.microedition.midlet.MIDlet`: a) O método `startApp()` será chamado pelo gerenciador de aplicativos quando uma MIDlet estiver para ser ativada; b) O método `pauseApp()` notifica de que a MIDlet está para pausar; c) O método `destroyApp()` sinaliza que a MIDlet vai desligar (MUCHOW, 2004).

A comunicação da MIDlet com o gerenciador de aplicativos ocorre pela mesma Classe: a) o usuário aperta o botão para sair da aplicação; b) `destroyApp()` limpa todos os recursos; c) `notifyDestroy()` avisa ao gerenciador que pode desligar a MIDlet. O método `notifyPaused()` envia o pedido de pausa para o gerenciador caso a MIDlet queira pausar e `resumeRequest()` avisa o gerenciador que a MIDlet já pode tornar-se ativa novamente (MUCHOW, 2004). Resumindo:

- `startApp()` é a execução da MIDlet;
- `pauseApp()` estado de espera;
- `destroyApp()` fecha a aplicação (CARNIEL, 2005);
- Ativado: indica que a MIDlet está em execução;
- Pausado: a própria aplicação pode pausar-se ou então ser pausada devido, por exemplo, a uma chamada;
- Destruído: a MIDlet libera todos os recursos adquiridos e é desligada pelo gerenciador de aplicativos (MUCHOW, 2004).

Outros dois métodos também importantes dentro da classe MIDlet podem ser usados durante o ciclo de vida de uma MIDlet: `MIDletStateChangeException()` e `MIDletStateChangeException()`, ambos para lançar exceções se ocorrer um erro na mudança de estados. A diferença básica é que o primeiro cria um objeto exceção sem texto e o segundo com texto.

A Tecnologia J2ME está invadindo o mercado e novos profissionais são requisitados. Porém, junto com ela, outras tecnologias devem também evoluir, para que o seu uso seja completo e a informação esteja cada vez mais ao alcance de quem a necessita. A informação disponível a qualquer lugar e a qualquer hora hoje se tornou essencial em muitas áreas, o que não é diferente para este trabalho, que foca a necessidade dos Policiais acessarem informações sobre trânsito de forma mais rápida. Disponibilizar a informação dentro desta tendência é um grande desafio para diversas áreas, porém, os Bancos de Dados são aqueles que já estão sendo revolucionados para tornar a informação acessível. Portanto, o próximo capítulo abordará a nova tecnologia dos Bancos de Dados Móveis que junto com outras novas tecnologias torna possível acessar informações a qualquer hora e em qualquer lugar.