FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS AUGUSTO RIBEIRO DE MARCO

IMPLEMENTAÇÃO E AVALIAÇÃO DE APLICAÇÕES MÓVEIS POR MEIO DE MÉTRICAS DE SOFTWARE

MARÍLIA
2008

CARLOS AUGUSTO RIBEIRO DE MARCO

IMPLEMENTAÇÃO E AVALIAÇÃO DE APLICAÇÕES MÓVEIS POR MEIO DE MÉTRICAS DE SOFTWARE

Trabalho de Curso apresentada ao Curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora:
Profa. Dra. MARIA ISTELA C. MACHADO

MARÍLIA
2008

DE MARCO, Carlos Augusto Ribeiro
Implementação e Avaliação de Aplicações Móveis por Meio de Métricas de Software / Carlos Augusto Ribeiro De Marco; orientadora: Maria Istela Cagnin Machado. SP: [s.n.], 2008.
76 f.

Trabalho de Curso (Graduação em Ciência da Computação) – Curso de Bacharelado, Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, Marília, 2008.

1.Computação Móvel 2.Desenvolvimento de Aplicações Móveis 3.Qualidade de Software 4.Métricas de Software 5.J2ME 6.Android

CDD: 005.14

CARLOS AUGUSTO RIBEIRO DE MARCO

IMPLEMENTAÇÃO E AVALIAÇÃO DE APLICAÇÕES MÓVEIS POR MEIO DE MÉTRICAS DE SOFTWARE

Banca Examinadora da monografia apresentada ao Curso de Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Grau de Bacharel em Ciência da Computação.

Resultado:

ORIENTADORA: _____
Profa. Dra. Maria Istela Cagin Machado.

1º EXAMINADOR: _____

2º EXAMINADOR: _____

Marília, __ de Novembro de 2008.

"O fracasso é o sucesso em processo! Desistir de um sonho é o mesmo que anunciar que você está morrendo. A persistência é a mãe do sucesso. É ela que, mais cedo ou mais tarde, materializa os nossos sonhos."

Marcelo de Almeida

Este trabalho é dedicado aos meus pais, por ter dado a mim a arte de viver.

AGRADECIMENTOS

A minha orientadora Dra. Maria Istela Cagnin Machado por ter me guiado com sabedoria e paciência nos longos caminhos para a realização deste trabalho.

Aos meus pais por me incentivar nesses quatro anos de faculdade e apoiar em todos os momentos difíceis.

A minha namorada pela paciência nos vários momentos que não pude dar o carinho necessário que merece para a realização deste trabalho.

Ao Sr. Adauto e Sra. Lourdes por ter concebido o local em que grande parte do trabalho foi concebido.

Aos amigos conquistados nesses anos de faculdade, especialmente a galera do fundão, no qual construímos uma amizade que com certeza durará muitos anos.

DE MARCO, Carlos Augusto Ribeiro. **Implementação e Avaliação de Aplicações Móveis por Meio de Métricas de Software.** 2008. 76 f. Trabalho de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2008.

RESUMO

A Computação Móvel é um paradigma computacional que tem como objetivo prover ao usuário o acesso permanente a uma rede fixa ou móvel independente de sua posição física. Está no caminho de um amadurecimento e destinada a se tornar o paradigma computacional no futuro. As tecnologias de computação móvel e comunicação sem fio têm sido largamente utilizadas, sendo um mercado que está crescendo rapidamente. Para usar essa tecnologia é necessário que os potenciais usuários tenham aplicações interessantes e um nível de qualidade que satisfaça características de qualidade. Os dispositivos móveis estão sendo fabricados com funcionalidades como *GPS (Global Positioning System)*, câmeras fotográficas digitais, placas de comunicação sem fio multiprotocolos, a fim de oferecer recursos para que sejam implementadas aplicações móveis em um mercado ainda pouco explorado atualmente e muito amplo. Na proporção que aumenta a demanda por sistemas mais complexos, com grande responsabilidade no gerenciamento de informações nas organizações, a qualidade de software torna-se um fator essencial no desenvolvimento de software, isso não é diferente no contexto de aplicações móveis. Para avaliar a qualidade de um software é necessário obter uma medida que quantifique o grau de alcance de uma característica de qualidade. Este trabalho tem como objetivo estudar o paradigma da computação móvel, a definição de tecnologias para o desenvolvimento de aplicações móveis (ou seja, J2ME e Android) e a avaliação de características de qualidade de tais aplicações por meio de um modelo, apresentado na norma NBR 13596, que estabelece métricas capazes de quantificá-la. Adicionalmente é conduzida uma medição nas aplicações móveis desenvolvidas nesta monografia, bem como a análise e interpretação dos resultados obtidos a fim de identificar se as aplicações móveis encontram-se dentro do nível aceito de qualidade.

Palavras-Chave: Computação Móvel, Desenvolvimento de Aplicações Móveis, Qualidade de Software, Métricas de Software, J2ME, Android.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Contexto

A Computação Móvel (LOUREIRO, 2003) é um paradigma que disponibiliza aos usuários a utilização dos dispositivos móveis para acessar redes de comunicação independente da sua localização física. As aplicações desenvolvidas para os dispositivos móveis apresentam recursos limitados, como telas de visualização pequenas, entrada de dados por meio de pequenos teclados, quantidade de memória e processamento reduzidos.

Com o crescimento do desenvolvimento de aplicações para dispositivos móveis, foram estabelecidas Métricas de Software que definem propriedades que devem ser sempre observadas, como alta flexibilidade, reusabilidade e manutebilidade, afim de manter a qualidade e melhorias no processo de desenvolvimento do software (FIGUEIREDO, 2006).

Motivação

Com a ampla concorrência entre as empresas no mercado e a constante evolução tecnológica, surgiu a necessidade do acesso às informações atualizadas no banco de dados das empresas, independente da localização física dos usuários com o objetivo de obter diferenciais para manter a sobrevivência das empresas ou destacar-se no mercado competitivo.

O grande crescimento do uso de dispositivos móveis e a queda no custo do aparelho apresentaram novas oportunidades no desenvolvimento de aplicações móveis. A demanda por aplicações que atendam as necessidades em diversas áreas, requer que as soluções sejam robustas, inovadoras, úteis e acima de tudo com qualidade.

Objetivos

O objetivo do presente trabalho é apresentar as características, estrutura, arquitetura e os principais desafios encontrados na área de computação móvel. Adicionalmente elencar linguagens de programação mais utilizadas ou em ascensão atualmente no desenvolvimento de aplicações móveis, no caso J2ME e Android, apontando suas características, vantagens e desvantagens. Além disso, estudar e apontar métricas de software que sejam adequadas no contexto de aplicações móveis e aplicá-las na avaliação de duas aplicações desenvolvidas com tecnologias distintas a fim de analisar os resultados obtidos do ponto de vista da qualidade de tais aplicações.

Organização do Trabalho

Este trabalho é organizado por capítulos, apresentados da seguinte forma:

No **Capítulo 1**, é discutido Computação Móvel, destacando-se os conceitos, históricos, características dos dispositivos móveis utilizados e os principais protocolos de comunicação.

No **Capítulo 2**, é apresentado o desenvolvimento de aplicações móveis do ponto de vista de implementação, destacando os conceitos, arquitetura, características da linguagem de programação *J2ME (Java 2 Micro Edition)* e da plataforma *Android*.

No **Capítulo 3**, são tratadas métricas de software que podem ser aplicadas em aplicações móveis, apresentando as principais definições deste contexto bem como as principais técnicas utilizadas.

No **Capítulo 4**, é apresentado o desenvolvimento de uma agenda eletrônica, sendo implementada em duas tecnologias distintas, uma em *J2ME* e outra em *Android*. Além disso, neste capítulo são discutidos os resultados obtidos da avaliação realizada nas duas aplicações móveis desenvolvidas por meio das métricas de software apresentadas no Capítulo 3.

Por fim, é apresentado um resumo do trabalho realizado, destacando as contribuições e limitações do mesmo e discutindo sugestões de trabalhos futuros.

CAPÍTULO 1 - COMPUTAÇÃO MÓVEL

Neste capítulo são apresentados o histórico, as características, as vantagens e desvantagens da computação móvel. Adicionalmente, são discutidas as principais tecnologias de comunicação e dispositivos móveis utilizados.

1.1 Histórico

Com os avanços das tecnologias de telecomunicação e de dispositivos de computação portáteis surgiu a Computação Móvel. A Computação Móvel é um novo paradigma no qual está mudando a forma quando as pessoas estão praticando atividades em movimento e/ou não desejam ficar "engessadas" numa rede de comunicação com infra-estrutura fixa (LOUREIRO, 2003).

Na década de 1960 predominava o processamento em lote (*batch*), onde o computador ocupava uma ampla sala, o funcionário preparado processava e recebia seu trabalho sem o contato com o ambiente computacional e utilizava as linguagens de programação *Cobol* e *Fortran*. Depois surgiram os computadores com a introdução dos conceitos de multiprogramação e "*Time Sharing*". Os usuários tinham acesso ao sistema através de terminais remotos e utilizavam as linguagens de programação *PL/I* e Basic. Mais alguns anos depois os computadores pessoais começaram a ser difundidos em larga escala com o desenvolvimento de hardware associado a esse tipo de computador. Os dados começaram a ser apresentados em formas gráficas e passaram a ser utilizadas as linguagens de programação Pascal e C. Por volta dos anos 90, os computadores pessoais começaram a ser utilizados em larga escala em qualquer atividade humana através das redes de computadores, principalmente a rede global conhecida como Internet, novos paradigmas e, consequentemente, as linguagens de programação surgiram (por exemplo, orientação a objetos, orientação a aspectos), os telefones celulares foram popularizados, surgindo o paradigma de Computação Móvel. Este paradigma tem como objetivo permitir que os usuários acessem informações, aplicações e serviços a qualquer lugar e a qualquer momento com acesso permanente a uma rede fixa ou móvel independente da sua localização física ou características de movimento (ROCHA, 2003).

1.2 Conceitos

A Computação Móvel está se tornando uma área madura e destinada a se tornar o paradigma computacional dominante (GIALDI, 2004).

A arquitetura de apoio a Computação Móvel é um modelo composto por componentes fixos e componentes móveis, conforme apresentado na Figura 1 (DUNHAM, 1995). Uma *Máquina Fixa* é um computador na rede fixa que não possui interface de comunicação sem fio. Os componentes fixos na rede são máquinas fixas, também conhecidas como *Estação de Apoio*, que se comunicam entre si através de uma rede fixa de alta velocidade e possuem uma interface de comunicação sem fio. Uma *Unidade Móvel* é um dispositivo ou computador móvel que é capaz de se comunicar com a rede fixa através de uma interface de comunicação sem fio. É através das Estações de Apoio que as Unidades Móveis se comunicam com a rede fixa de alta velocidade. Cada Estação de Apoio possui uma área de cobertura de sinal conhecida como *Célula Sem Fio*, no qual as Unidades Móveis poderão se movimentar podendo, por exemplo, sair de uma Estação de Apoio A1 e entrar em uma Estação de Apoio A2 mantendo a comunicação com a rede fixa.

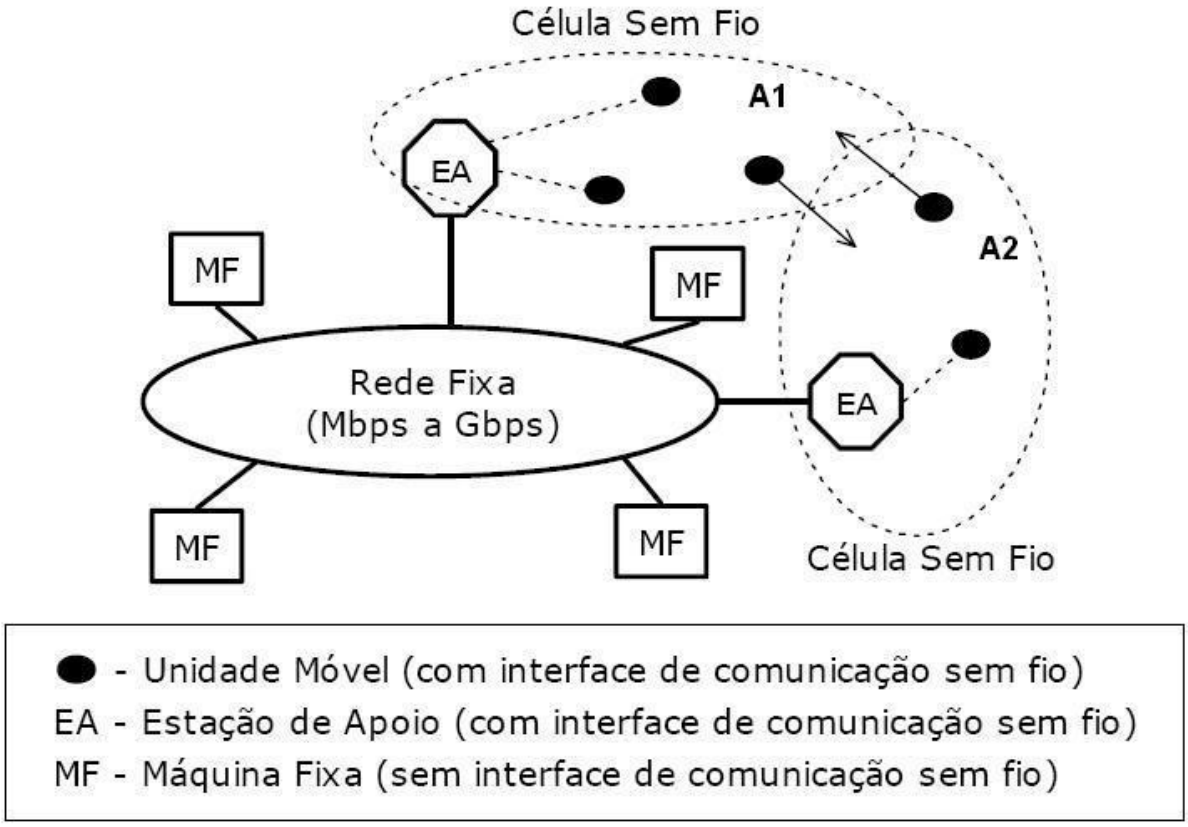

Figura 1 – Uma Arquitetura de Apoio a Computação Móvel (DUNHAM,1995)

As aplicações móveis (sistemas de software no contexto da computação móvel) são muito difíceis de programar, devido a falta de linguagem de programação comum, diferentes normas, modelos de dispositivos móveis (por exemplo, celulares, PDAs (*Personal Digital Assistants*)) e pouca padronização. Construir um sistema móvel é muito complexo, uma vez que requer diferentes decisões de projeto (FRIDAY, 2004).

Com o desenvolvimento de redes sem fio, a partir da necessidade de utilização de dispositivos móveis, surgiram vários problemas no qual foi e é fruto de diversas pesquisas como (MESQUITA, 2007):

- problemas com sinais de rádio finito;
- segurança das informações propagadas no ambiente;
- definição de freqüência única;
- construção de um sistema de largura de banda economicamente viável.

As principais características dos sistemas de software para administrar os requisitos impostos pelas características da Computação Móvel são (GIALDI, 2004):

- capacidade de localizar/endereçar elementos móveis;
- capacidade de perceber mudanças no ambiente de execução;
- capacidade de se auto-configurar;
- capacidade de migrar funcionalidades;
- uso eficiente dos recursos no elemento móvel;
- uso eficiente dos recursos do canal de comunicação sem fio;
- alto grau de tolerância à falhas; e
- uso de mecanismos para autenticação e criptografia de dados.

A principal dificuldade em Computação Móvel na Web não está na mobilidade, mas na gestão dos domínios administrativos que são regras aplicadas entre as redes. No surgimento da Internet podia-se confiar nos endereços IP para comunicação entre os computadores. Atualmente, isso não é verdade, pois existem *Firewall*, que são domínios administrativos que aplicam políticas rígidas entre as redes. A mobilidade requer mais do que uma noção tradicional de autorização para processar ou acessar informações em determinados domínios, ela envolve a autorização tanto de entrar como de sair de determinados domínios. Uma *Unidade Móvel* deve primeiro sair do seu domínio administrativo (obter autorização), em seguida, entrar numa outra área protegida e obter a autorização para acessar as

informações. O acesso às informações é controlado por múltiplos níveis: Computador Local, Rede Local, Rede Metropolitana, Rede Global – Internet. Assim, a aplicação em dispositivos móveis deve ser equipada para navegar nessa hierarquia de domínios administrativos (CARDELLI, 1998).

1.3 Dispositivos Móveis

Com a evolução dos dispositivos móveis e do sistema de comunicação sem fio, os dispositivos ganharam funcionalidades como: *download* de músicas, jogos e vídeos, reprodução de conteúdo multimídia, função de máquina digital de alta resolução, gravação de vídeos, envio de mensagens em formatos textos e gráficos, tanto localmente quanto remotamente através de conexões sem fio, como o *Bluetooth* (Seção 1.4.1) e o *Wi-Fi* (Seção 1.4.4) (PERING, 2006).

Os dispositivos móveis possuem restrições impostas pela mobilidade, que estão definidas a seguir (NEVES, 2005):

1) dispositivos móveis são carentes em recursos em relação a dispositivos fixos – devido ao tamanho, peso, pouca memória e telas reduzidas.

2) a mobilidade é inerentemente mais vulnerável a perigos – pela característica dos dispositivos móveis serem menores, estão sujeitos a sofrerem danos, roubados ou simplesmente perdê-los.

3) a conectividade em sistemas móveis é muito instável com relação ao desempenho e confiabilidade – por ser um dispositivo móvel, o mesmo sofre interferências de diversos tipos na rede de comunicação sem fio, tendo assim um ambiente muito instável.

4) dispositivos móveis dependem de uma fonte de energia finita – por utilizar fonte de energias móveis, como baterias, a computação móvel necessita prever a otimização de recursos, como exemplo, apagar a luz da tela automaticamente após certo período com o dispositivo de celular aberto.

Complementando a última restrição apresentada, o consumo de energia é um grande gargalo para dispositivos móveis em que existe uma crescente pressão dos fabricantes de software para fornecer eficiência energética. Um método utilizado que diminui o consumo de energia consideradamente é a filtragem de dados por meio de uma *MSS (Mobile Support Station)*, a qual é uma estação de suporte móvel que efetua uma fina granularidade na

indexação dos pacotes enviados para os endereços no formato de *multicast*. Assim a *MSS* não envia sinal para o dispositivo móvel permitindo que a CPU permaneça por mais tempo em modo inativo por não estar processamento pacotes que não teriam interesses ao mesmo, conduzindo a uma economia de energia (GUPTA, 1995).

Nas subseções a seguir são apresentadas as características de alguns dispositivos móveis.

1.3.1 PDA – Personal Digital Assistants

São conhecidos como organizadores pessoais, sendo de uma forma genérica uma evolução das agendas eletrônicas, permitindo o desenvolvimento e utilização de novas aplicações, maior flexibilidade na capacidade de processamento, entrada e saída e comunicação sem fio.

Originalmente os PDA´s possuíam 512 Kb de memória e sem presença de um HD (*Hard-Disk*). Com a evolução desses dispositivos, atualmente, possuem um mínimo de 16 Mb de memória, podendo ser expandida por meio de cartões de memórias.

As telas possuem resoluções que variam entre 160x160 a 640x480 *pixels* e dependendo do modelo do aparelho atingem mais de 65000 cores e se comunicam através de Infravermelho ou *BlueTooth* (BLUETOOTH, 2008, PERING, 2006). Reconhecem arquivos textos e diversos tipos de arquivos multimídia.

O protocolo *Bluetooth* é uma tecnologia de comunicação sem fio que tem como característica principal a substituição da tecnologia para cabos de curto alcance. Maiores detalhes sobre este protocolo de comunicação podem ser obtidos na Seção 1.4.1.

O Infravermelho é uma radiação medida por dispositivos que reagem a variação de temperatura provocada pela absorção de infravermelho por uma superfície escurecida. Tem capacidade de transmissão de dados sem fio a velocidade muito baixas, sendo geralmente utilizados para a transmissão de pequenas quantidades de dados entre dispositivos móveis e periféricos, como mouse e teclado sem fio. Além da limitação da velocidade que pode transmitir existe outro fator limitador sendo o físico. Para a transmissão de dados entre dois dispositivos por meio de infravermelho é necessário estar no campo de visão de ambos para que seja enviado e transmitido com sucesso.

O reconhecimento de caracteres na maioria das vezes pode ser através de um tipo de caneta especial, por meio de toques na tela, sendo também possível a confecção de desenhos, constituindo em uma saída de fácil operação (MESQUITA, 2007 e DOCHEV, 2006).

1.3.2 Celular

O celular é o tipo de dispositivo móvel utilizado principalmente para a comunicação verbal por meio da tecnologia *GSM (Global System for Mobile Communications)* e *CDMA (Code Division Multiple Access)*, envio e recebimento de mensagens de texto *SMS (Short Message Service).*

A tecnologia *GSM* tem como característica o seu sinal e os canais de voz são digitais e prover de serviços a baixos custos.

A tecnologia *CDMA* é um método de acesso a canais em sistemas de comunicação, utilizado tanto em telefonia celular quanto para o rastreamento via satélite (GPS) por meio da multiplexagem. A multiplexagem é um método de acesso múltiplo que codifica os dados com um código especial associado a cada canal de comunicação e usa as propriedades construtivas de interferência dos códigos especiais para executa-lá.

Tem como característica a baixa capacidade de memória e transferência de dados. Alguns modelos de celulares possuem o acesso a Internet por meio da tecnologia *WAP (Wireless Application Protocol)* (MESQUITA, 2007) ou *GPRS (General Packet Radio Service)* (CARVALHO, 2001).

A tecnologia *WAP* é um protocolo aberto que permite a comunicação entre aparelhos sem fio e a visualização de *websites* escritos em *WML* (*Wireless Markup Language*) (MESQUITA, 2007) por meio de um navegador. Maiores detalhes sobre este protocolo de comunicação podem ser obtidos na Seção 1.4.2.

A tecnologia *GPRS* é um serviço que permite o envio e o recebimento de informações através da rede telefônica móvel. Maiores detalhes sobre este protocolo de comunicação podem ser obtidos na Seção 1.4.3.

1.3.3 Smart-Phones

Trata-se de um tipo de dispositivo híbrido, que combinam características dos aparelhos de celulares convencionais com as características dos *PDAs*. Possui tamanho inferior a uma *PDA* e superior a um telefone celular. Normalmente não há um tipo de teclado grande.

Os *Smart-Phones* apresentam as características de serem utilizados em várias redes sem fio, estarem sempre conectados a rede de telefonia celular e possuírem um pequeno

teclado facilitando a entrada de dados pelos seus usuários, sincronização dos dados do organizador com um computador pessoal (MESQUITA, 2007).

Os dispositivos móveis de Smart-Phones para a transmissão de dados utilizam as tecnologias *GPRS*, *Bluetooth* e Infravermelho e para a transmissão de voz as redes *GSM* e *CDMA*.

1.4 Principais Protocolos

Os dispositivos móveis para se comunicar com as redes de comunicação fixa, utilizam tecnologias e protocolos específicos para esse meio, nesta seção são apresentados os principais protocolos e suas respectivas características.

1.4.1 Bluetooth

O *Bluetooth* (BLUETOOTH, 2008, PERING, 2006) é um protocolo de comunicação sem fio, que pode conectar dois ou mais dispositivos e tem como características a substituição da tecnologia para cabos de curto alcance, mantendo elevados níveis de segurança, robustez, baixo consumo de energia e custo. Tem como habilidade fundamental a comunicação de dados e voz simultaneamente. Podem conectar dispositivos através de redes sem fio de curto alcance por meio das redes *ad hoc*, conhecidos também como *piconets,* que são redes em que prevêem um dispositivo como mestre e é responsável pela sincronização dos outros dispositivos chamados de escravos. As desvantagens dessa tecnologia devem-se ao fato do seu raio de alcance, aproximadamente de dez metros e o máximo de oito dispositivos que podem se conectar ao mesmo tempo. A tecnologia Bluetooth opera na faixa *ISM (Industrial, Scientific and Medical)*, que são bandas reservadas internacionalmente para o uso comercial de radiofreqüência nas áreas Industrial, Científica e Médica, operando na freqüência entre 2.4GHz e 2.485GHz. É do tipo *Full-Duplex*, que permite transmissão em ambos os sentidos simultaneamente, utilizando multiplexação *spread spectrum*, que se refere a uma técnica avançada de multiplexação ou compartilhamento. A taxa de transmissão de dados varia de 1 Mbps na versão 1.2 e até 3 Mbps na versão 2.0 (BLUETOOTH, 2008, PERING, 2006).

O raio de alcance depende da classe do dispositivo, sendo:

- classe 3: tem um alcance de até 1 metro ou 3 pés;

- classe 2: mais comumente encontrando nos dispositivos móveis, tem um alcance de até 10 metros ou 33 pés;
- classe 1: utilizado em casos industriais, tem um alcance de até 100 metros ou 300 pés.

1.4.2 WAP

A tecnologia *WAP* (SCHMITZ, 2003) é um conjunto de protocolos que define o funcionamento, segurança, transações e outros, permitindo as operadoras, fabricantes e desenvolvedores fazer frente aos requerimentos de flexibilidade e diferenciação que cada vez mais exige o mundo das telecomunicações sem fio. A comunicação entre aparelhos sem fio e a visualização de *websites* é realizado por meio de uma linguagem de marcação denominada *WML* (*Wireless Markup Language*). Os protocolos utilizados minimizam os problemas associados ao uso de protocolos de Internet para transferência de dados sem fio.

O *WML* segue o padrão *XML* (*EXtensible Markup Language*) e foi projetada para o uso dos mais diversos tipos de aplicações com as restrições impostas pelos dispositivos portáteis (MESQUITA, 2007).

A arquitetura é composta de cinco camadas, como mostra a Figura 2, tornando um ambiente escalável e extensivo para o desenvolvimento de aplicações que utilizam comunicações móveis.

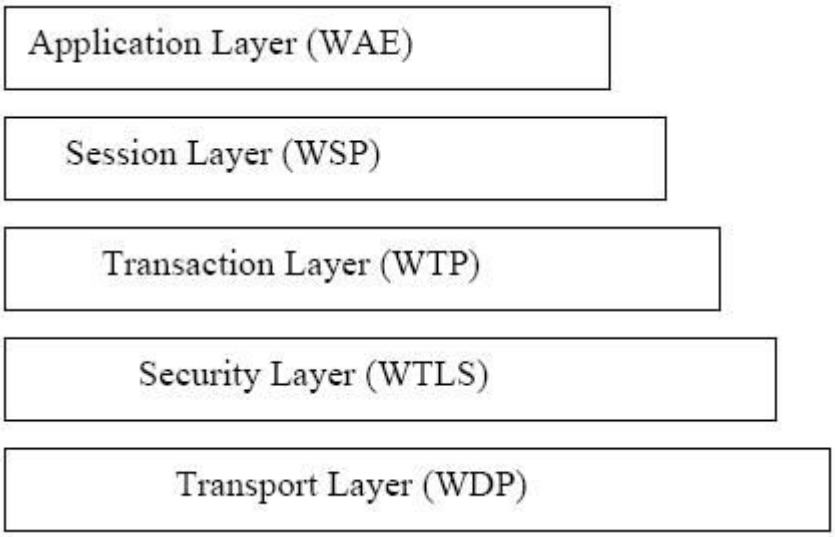

Figura 2 – Pilha de protocolos WAP (SCHIMITZ, 2003)

Com a arquitetura utilizada no protocolo WAP é permitido que outros serviços e aplicações utilizem as funções da pilha de camadas por meio de interfaces. As aplicações externas podem acessar diretamente as camadas de sessão, transação, segurança e transporte, independente dos serviços da portadora fornecidos pela rede.

1.4.3 GPRS

A tecnologia *GPRS (General Packet Radio Service)* (CARVALHO, 2001) é um serviço que permite o envio e recepção de informações através da rede telefônica móvel. Tem como características a alta disponibilidade, taxas de transferências superiores às redes de telecomunicações fixas, eficiência na comutação dos pacotes, compatível com a Internet e tem suporte as tecnologias *TDMA* (*Time Division Multiple Access*) e *GSM* (*Global System for Mobile Communications*) (CARVALHO, 2001).

A arquitetura é composta por cinco camadas que controlam a comunicação entre a estação móvel *MS (Mobile Station)* e a base da estação *BSS (Base Station System),* sendo: *LLC (Logical Link Control), RLC (Radio Link Control)*, *MAC (Medium Access Control)*, *PLL (Physical Link Layer)* e *RFL (Physical RF Layer)* (OLIVEIRA, 2004).

A camada *LLC* tem como funcionalidades o controle de sequências, entrega em ordem, detecção e correção de erros e retransmissão. A camada *RLC* tem como funcionalidades a fragmentação e remontagem dos quadros *LLC* em bloco de dados *RLC* e correção de erros por meio de um mecanismo de retransmissão seletiva desses blocos. A camada *MAC* tem como funcionalidades a multiplexação de vários usuários em um mesmo canal lógico, escalonamento e priorização baseado no *QoS (Quality of Service)* negociado. As camadas *PLL* e *RFL* constituem a camada física e tem como funcionalidades a codificação dos dados, detecção e correção de erros de transmissão no meio físico, modulação/demodulação das ondas físicas (OLIVEIRA, 2004).

1.4.4 Wi-Fi

O protocolo *Wi-Fi (Wireless Fidelity)* (RODRIGUES, 2004) é uma rede de comunicações sem fios que utiliza ondas eletromagnéticas como meio de transmissão de informação, através de um canal que interliga os diferentes equipamentos móveis presentes na

mesma. O volume de informações que uma onda eletromagnética é capaz de transportar está diretamente relacionado com a sua largura de banda.

Uma rede *Wi-Fi (Wireless Fidelity)* é um sistema flexível de comunicações, no qual pode ser implementado como uma extensão às redes cabeadas ou uma rede de comunicação totalmente sem fio, permitindo grande mobilidade aos utilizadores, sem perder a conectividade (RODRIGUES, 2004).

As principais características desse tipo de tecnologia são:

- **mobilidade no acesso às informações e aos recursos na rede:** facilita a implementação de aplicações que requeiram ligação permanente à rede em ambientes que envolvam movimentação do utilizador.
- **aumento de produtividade:** as redes *Wi-Fi* permitem aos seus utilizadores serem mais produtivos, já que possibilitam o acesso à *Internet*, e-mail, bases de dados entre outros, em qualquer lugar;
- **redução de custos:** utilização mais eficiente dos recursos e infra-estruturas;
- **flexibilidade:** é mais fácil adicionar, retirar ou modificar clientes que usem a tecnologia de acesso *wireless*, nomeadamente em infra-estruturas de rede temporárias. Os utilizadores podem ainda trabalhar em vários locais sem necessidade de grandes configurações por parte dos administradores da rede.
- **trabalho colaborativo:** facilidade de acesso a ferramentas compartilhadas, e-mail, *Instant Messaging* e *Groupware* a partir de qualquer localização.

1.5 Considerações Finais

Nesta seção foi apresentada a arquitetura de uma comunicação sem fio entre os dispositivos móveis, apontando as características e as limitações da computação móvel. Em seguida foi apresentando os diversos tipos de dispositivos móveis disponíveis no mercado e apontando as características que o compõem. Finalizando foi apresentando as principais tecnologias utilizadas para a comunicação de dados entre os dispositivos móveis e as redes de comunicações.

Na seção seguinte são abordadas duas tecnologias para o desenvolvimento de aplicações móveis.

CAPÍTULO 2 - DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

2.1 Considerações Iniciais

Essa seção são abordadas as características de duas tecnologias usadas para a implementação de aplicações móveis, sendo a plataforma Java que é amplamente utilizada e a plataforma Android que está em franca ascensão.

2.2 Plataforma J2ME

A linguagem de Programação Java começou como um projeto na SUN que tinha como foco principal a comunicação entre dispositivos computacionais. Um dispositivo que suporta Java significa dizer que o dispositivo possui uma *JVM (Java Virtual Machine),* que é capaz de executar programas em Java. A tecnologia Java é um conjunto de ferramentas e *APIs (Application Programming Interface)* usadas para construir aplicações e *applets* (pequenas aplicações em Java hospedadas dentro de páginas Web) (MORRISON, 2001). A SUN subdividiu o Java em quatro categorias conforme o tipo de aplicação a ser desenvolvida, sendo *J2EE, J2SE, J2ME e Java Card* (SUN, 2008). Salienta-se que a linguagem de programação conhecida como *J2ME (Java 2 Micro Edition)* é uma das edições da plataforma Java, que é de interesse deste trabalho. Na Figura 3 são apresentadas todas as edições da plataforma Java.

Figura 3 – Edições da Plataforma Java (SUN, 2008)

Em 1995 foi lançada uma versão do Java conhecida como *J2SE (Java 2 Standard Edition)*, com a frase *"Write Once, Run Anywhere"* ("Escreva uma vez, execute em qualquer lugar"), que tinha como objetivo desenvolver uma linguagem de programação no qual escreveria apenas uma vez o código e executaria em qualquer plataforma que suportasse uma máquina virtual Java, sendo projetada para aplicações em execução em máquinas simples (MORRISON, 2001, MUCHOW, 2004).

Dois anos depois da introdução da linguagem, foi lançada um nova edição, o *J2EE (Java 2 Enterprise Edition)*, fornecendo suporte para o desenvolvimento de aplicações em nível empresarial, *APIs (Application Programming Interface)* mais complexas, com suporte interno para *Servlets* (TEMPLE, 2004), *JSP (JavaServer Pages)* (TEMPLE, 2004) e *XML (Extensible Markup Language)* (ALMEIDA, 2002) além de conter todos os recursos da edição do *J2SE*.

Servlets são classes Java, desenvolvidas de acordo com uma estrutura definida e instaladas junto a um servidor que implemente um *Servlet Container* (servidor que permite a execução de *Servlets*), que podem tratar requisições recebidas de clientes gerando conteúdo dinâmico. *JSP* é uma tecnologia usada para o desenvolvimento de aplicações para a *Web*, tendo a vantagem de portabilidade de plataformas, permitindo ao desenvolvedor de páginas para Internet produzir aplicações que interajam com banco de dados, tornando-a dinamicamente. O *XML* é uma linguagem idealizada por Jon Bosak, engenheiro da Sun Microsystems, em 1996 e tem como funcionalidade possibilitar ao autor especificar a forma dos dados no documento, além de permitir definições semânticas. Pode conter, simultaneamente, dados e descrição da estrutura do documento, por meio da *DTD (Data Type Definitions)*, que são gramáticas que conferem estrutura ao documento *XML*.

Algum tempo depois, foi lançada uma nova edição, que foi o *J2ME (Java 2 Micro Edition)*, projetada para dispositivos com memória, vídeo e poder de processamento limitados (MUCHOW, 2004). *J2ME* é uma versão simplificada, ou seja, é um subconjunto do *J2SE* que suporta um conjunto mínimo de características que são aplicáveis a dispositivos móveis. Para suportar uma ampla variedade de produtos que se encaixam dentro do escopo do *J2ME*, a SUN introduziu dois conceitos, sendo o de nível de Configuração e o de nível de Perfil, descritos detalhadamente nos Seções 2.2.1 e 2.2.2, respectivamente. Na Figura 4 apresenta-se a relação existente entre tais conjuntos.

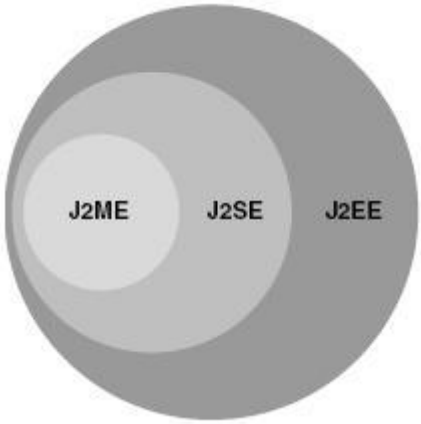

Figura 4 – Relação entre J2ME, J2SE e J2EE (SUN, 2008)

Para completar a família Java também existe o Card Java, que é uma tecnologia que permite que pequenos aplicativos (*applets*) baseados em plataforma Java sejam executados com segurança em *smart cards* e dispositivos similares com limitações de processamento e armazenamento. Essa tecnologia é amplamente utilizada em cartões *SIM (Subscriber Identity Module)*, usados em celulares *GSM* e em cartões de créditos para armazenas informações do assinante, agenda, preferências, serviços contratados com objetivo de autenticação do cliente (MUSHOW, 2004).

2.2.1 Linguagem de Programação J2ME – Nível de Configuração

O nível de Configuração da J2ME está vinculado a uma *JVM (Java Virtual Machine)*, no qual define os recursos da linguagem Java, ou seja, as bibliotecas básicas da *JVM* para o desenvolvimento de aplicativos para rodar em uma variedade de dispositivos. É geralmente baseada nos recursos de memória, vídeo, conectividade de rede e no poder de processamento (MORRISON, 2001, MUCHOW, 2004).

A seguir são apresentadas duas configurações definidas e determinadas para isso:

CDC – Connected Device Configuration

- 512 Kb (mínimo) de memória para executar o Java;
- 256 Kb (mínimo) de memória para alocação de memória em tempo de execução;
- Conectividade de rede, largura de banda possivelmente persistente e alta.

CLDC – Connected Limited Device Configuration

- 128 Kb de memória para executar o Java;

- 32 Kb para alocação de memória em tempo de execução;
- Interface restrita com o usuário;
- Baixo consumo de energia, normalmente alimentado por bateria;
- Conectividade de rede, largura de banda baixa e acesso intermitente;
- Processador de 16-bit;
- Não há suporte para operações matemáticas usando ponto flutuante, datas e literais.
- O método `finalize()` que é chamado quando um objeto é removido da memória para ajudar a limpar quaisquer recursos relacionados não está disponível.
- O *JVM* suportará um conjunto limitado de exceções de tratamento de erros.

No desenvolvimento de uma nova aplicação o nível de configuração define qual o tipo de dispositivo será instalada a aplicação que será utilizada. As aplicações que necessariamente precisarem de mais recursos, como processamento e memória, será utilizado o nível de configuração CDC, que geralmente destina-se a PDA. Caso contrário será utilizado o nível de configuração CLDC, destinado a dispositivos móveis com poder de processamento e memória limitados, geralmente os celulares.

2.2.2 Linguagem de Programação J2ME – Nível de Perfil

Acima do nível de configuração, existe o nível de perfil, que é extensão do nível de configuração. Ele fornece *APIs* específicas para o desenvolvedor escrever aplicações para um tipo particular de dispositivos. Em termos gerais, a configuração define uma família de dispositivos enquanto o perfil isola e especifica um determinado tipo de dispositivo, no qual acrescenta funcionalidades adicionais através de *APIs* associadas ao dispositivo (MORRISON, 2001, MUCHOW, 2004).

Os perfis mais conhecidos são os seguintes: *MIDP (Mobile Information Device Profile)*, *Foundation Profile*, *Personal Basis Profile* e *Personal Profile* que são descritos detalhadamente nas Seções 2.2.2.1, 2.2.2.2, 2.2.2.3 e 2.2.2.4, respectivamente.

2.2.2.1 MIDP

O perfil *MIDP (Mobile Information Device Profile* - Perfil de Dispositivo de Informação Móvel) é baseado no nível de configuração *CLDC* e é o mais amplamente utilizado. Tem como foco o desenvolvimento de aplicações, portanto, disponibiliza os recursos de controle do ciclo de vida de uma aplicação (*MIDLet*), interface com o usuário, som, armazenamento persistente de dados, modelo de sinalização de aplicativos, modelo de segurança, rede e temporizadores (CRUZ, 2005).

MIDLet é uma super-classe do perfil *MIDP*, que controla o ciclo de vida de uma aplicação móvel por meio de métodos específicos que uma *JVM* executa. O ciclo de vida de uma aplicação *MIDLet* é controlada através de métodos que carregam, executam, suspendem e destroem uma aplicação (CRUZ, 2005).

Os MIDLets possuem três diferentes estados: *paused*, *active* e *destroyed*, sendo implementados por três métodos abstratos definidos na classe MIDLet: `startApp`, `pauseApp` e `destroyApp`.

Na figura 5 são apresentados os três métodos abstratos e os três estados de um MIDLet durante o seu ciclo de vida por meio de diagrama de estados.

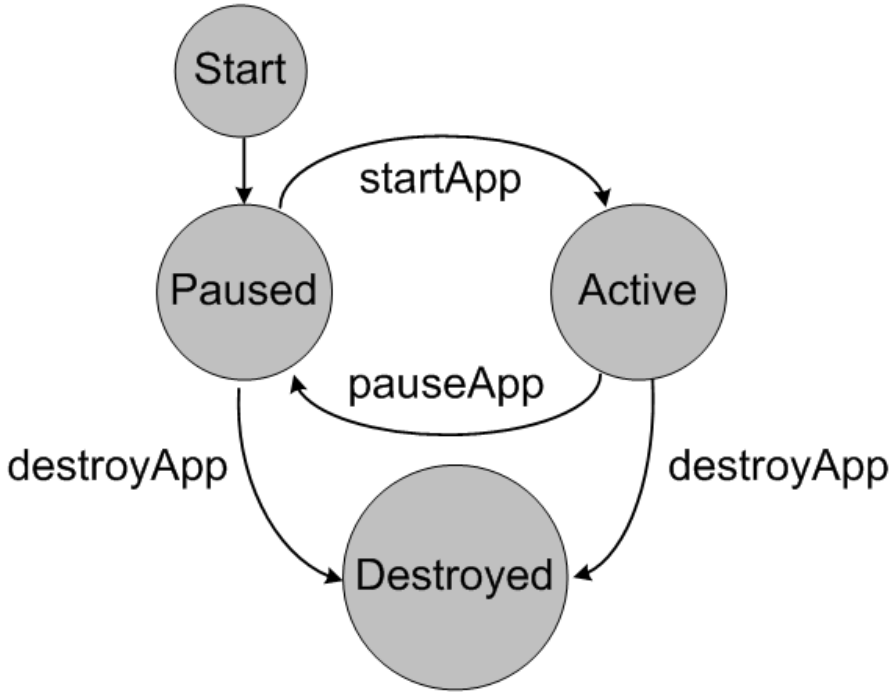

Figura 5 – Ciclo de Vida de um MIDLet (SUN, 2008)

A seguir são apresentados os requisitos mínimos de hardware para a implementação do perfil *MIDP*:

- **Memória**

- 128 Kb de memória não-volátil para as *APIs MIDP*.

- 32 Kb de memória volátil para executar o Java.

- 8 Kb de memória não volátil para armazenar os dados persistentes, como configurações e dados do aplicativo.

- **Entrada de Dados**

- Os requisitos de entrada de dados para dispositivos *MIDP* devem ter no mínimo um tipo de entrada disponível: um teclado ou *touch screen.* O *touch screen* é uma tela que contêm uma película sensível ao toque baseando-se em infravermelho. O mouse não é um requisito de entrada, pois é pouco provável que um dispositivo móvel teria a capacidade de utilizar o mouse.

- **Limitações de Telas**

- Os requisitos para as telas para dispositivos *MIDP* tem como características o tamanho mínimo de 96 pixels de largura e 54 pixels de altura e pelo menos tom de preto e branco.

- **Conectividade de rede sem fio**

- Conexão de rede sem fio de algum tipo no qual é esperado uma comunicação intermitente, como uma conexão dial-up, com largura de banda limitada a 9600 bps.

O sistema operacional nativo executando em um dispositivo móvel pode variar, portanto seguem abaixos os requisitos mínimos de software:

- Um sistema operacional mínimo que possa gerenciar recursos de hardware de baixo nível como processamento de interrupções, tratamento de exceções e agendamento;
- Um mecanismo de leitura e escrita em memória não volátil;
- Suporte mínimo para escrita de elementos gráficos bitmap na tela;
- Um mecanismo para a gestão do ciclo de vida de uma aplicação;
- Um mecanismo para capturar as informações fornecidas pelo usuário através de um teclado ou *touch screen*;
- Leitura e escrita de informações através de acessos a rede de comunicação sem fio;

2.2.2.2 Foundation Profile

O primeiro perfil baseado no nível de configuração *CDC* foi o *Foundation Profile.* É o perfil mais genérico, sendo base para o desenvolvimento de perfis mais sofisticados como o *Personal Profile* e *Personal Basis Profile*.

Foundation Profile, ao contrário de perfil *MIDP*, não fornece classes para a construção de aplicações interativas e é utilizado por dispositivos embutidos que não requer interface gráfica, pois exclui o suporte a interfaces gráficas, como os pacotes `AWT` e `SWING`.

Por outro lado, o *Foundation Profile* inclui suporte para listas de controle, classes e interfaces para interpretar e administrar certificados de segurança e gerar chaves de criptografia, proporcionando um suporte a segurança.

Suporta os protocolos de comunicação *socket* e *HTTP*. Inclui classes para a leitura e escrita de arquivos comprimidos como o *JAR* e *ZIP*.

O arquivo *JAR (Java Archive)*, é um arquivo compactado usado para distribuir um conjunto de classes Java e o arquivo *ZIP* é um tipo de formato de compactação de arquivos de diferentes extensões.

2.2.2.3 Personal Basis Profile

O perfil *Personal Basis* é o de nível intermediário na configuração *CDC* e utiliza todos os recursos do perfil *Foundation,* adicionando três novas características: suporte a interfaces gráficas leves, através de um subconjunto da *API AWT*; e um modelo de aplicação *Xlet* adaptado da *API* Java TV; e comunicação entre *Xlets (Inter-Xlet Communication – IXC)* através de um subconjunto da *API RMI.*

Estes recursos permitem a criação de aplicações interativas com ciclos de vida bem definidos.

O modelo de aplicação Xlet é uma interface que é implementada pela classe principal, no qual fornece métodos abstratos que definem o ciclo de vida da aplicação, conforme apresentado na Figura 6. Cada Xlet deve implementar os seguintes métodos:

- `initXlet`: inicializa uma Xlet e altera para o estado do ciclo de vida “Pausado”;
- `startXlet`: o estado do ciclo de vida é alterado do estado “Pausado” para “Ativo” e sua execução é iniciada;
- `pauseXlet`: o estado do ciclo de vida é alterado para “Pausado”;

- `destroyXlet`: o estado do ciclo de vida é alterado para “Destruído”.

Na figura 6 é apresentado o ciclo de vida de um Xlet através de diagrama de estados.

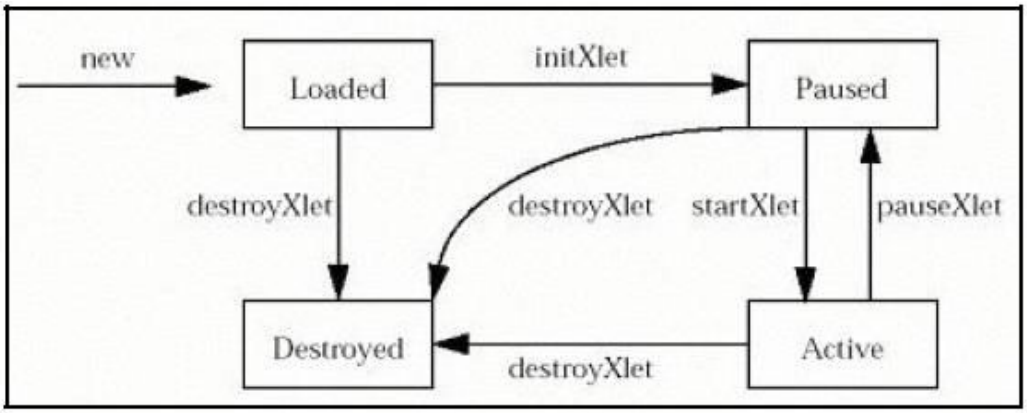

Figura 6 – Ciclo de Vida de um Xlet (SUN, 2008)

A comunicação entre dois ou mais *Xlets* em uma mesma *JVM* tem como objetivo o compartilhamento de objetos e execução de código em um contexto diferente. A comunicação e feita através do *IXC* no qual é baseado em *RMI (Remote Method Invocation* – Chamada Remota de Procedimento) (SUN, 2008), que tem suporte ao mecanismo de desenvolvimento de protocolos de acessos a objetos distribuídos em sistemas remotos, sendo que a comunicação é realizada internamente em uma mesma *JVM* e não entre as *JVMs*.

2.2.2.4 Personal Profile

O perfil *Personal* é baseado no nível de configuração *CDC* e é um superconjunto do perfil *Personal Basis* e adiciona as seguintes características: suporte para a construção de aplicações baseadas em *browser*; subconjunto da *API AWT* mais abrangente que inclui suporte para interfaces gráficas mais pesadas, como consoles de jogos, e eventos; pacote `java.awt.datatransfer` que gerencia o domínio de segurança, oferencendo soluções de segurança de maturidade semelhantes aos de aplicações em plataforma *J2SE*.

2.3 Plataforma Android

A empresa *Google*, em 05 de novembro de 2007, tornou pública a primeira plataforma *Open Source* de desenvolvimento para dispositivos móveis baseada na plataforma Java com

sistema operacional Linux, a qual recebeu o nome de Android, que é a primeira plataforma de desenvolvimento móvel completa, livre e aberta.

A plataforma Android está sendo mantida pela *Open Handset Alliance*, que é um grupo formado por mais de 30 empresas de tecnologias de dispositivos móveis, provedoras de serviços móveis, no qual se uniram para inovar e acelerar o desenvolvimento de aplicações e serviços. A plataforma Android é apoiada por empresas importantes da indústria móvel, como T-Mobile, HTC, Qualcomm, Motorola, Samsung, LG Electronics, Sprint Nextel, Telefônica, China Mobile e outras.

A seguir são apresentados os recursos suportados pela plataforma Android:

- *framework* de aplicação – permite o reuso dos componentes;
- máquina virtual Dalvik (Seção 2.3.1) – otimizada para dispositivos móveis;
- navegador web integrado – baseado no navegador *open source WebKit*;
- gráficos otimizados – através de bibliotecas de gráficos 2D e gráficos 3D baseados na especificação OpenGL ES 1.0;
- SQLite – armazenamento de dados em formato de estruturas relacionais;
- suporte para mídias de áudio, vídeo e imagens nos formatos (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF);
- telefonia GSM;
- Bluetooth, EDGE, 3G e WiFi;
- camera, GPS;
- poderoso ambiente de desenvolvimento, incluindo ferramentas para depuração, analisador de memória e performance, emulador de dispositivo e um *plug-in* para o *IDE* (*Integrated Development Environment* – Ambiente Integrado para Desenvolvimento de Software) Eclipse.

Plataforma Android é nova, sendo que o primeiro dispositivo móvel utilizando a tecnologia Android, batizado de *T-Mobile G1*, foi comercializado nos Estados Unidos em 22 de Setembro de 2008, por 179 dólares, pela *T-Mobile*. O aparelho é todo *touch screen*, teclado *QWERTY*, aplicativos embarcados como *Google Maps* com *StreetView*, *Gmail*, *GTalk*, *Flickr* e *YouTube*. Suporte conexões *3G* e *Wi-Fi*, navegação em *full HTML*, câmera de 3 megapixels e cartão de memória de 1Gb.

A previsão segundo (IDGNOW, 2008) é que o próximo lançamento seja em novembro no Reino Unido e os próximos lançamentos internacionais previstos para 2009.

Empresas multinacionais como Samsung, Motorola e LG, que fazem parte da *OHA (Open Handset Alliance)*, estão aumentando o seu número de desenvolvedores em plataforma Android para lançar seus primeiros dispositivos móveis no mercado utilizando a tecnologia. A plataforma Android apresenta interfaces gráficas arrojadas, GPS, serviços integrados com o Google Maps e também um SGBD (Sistema Gerenciador de Banco de Dados) com nome de SQLite.

2.3.1 Arquitetura

A arquitetura da plataforma Android é dividida em várias camadas sendo: *Applications*, *Application Framework*, *Libraries*, *Android Runtime* e *Linux Kernel*, como mostrado na Figura 7.

Na camada *Applications* está localizada uma lista de aplicações padrões que incluem mapas, navegador, gerenciador de contatos, programa de *SMS*, sendo todas desenvolvidas utilizando a linguagem Java.

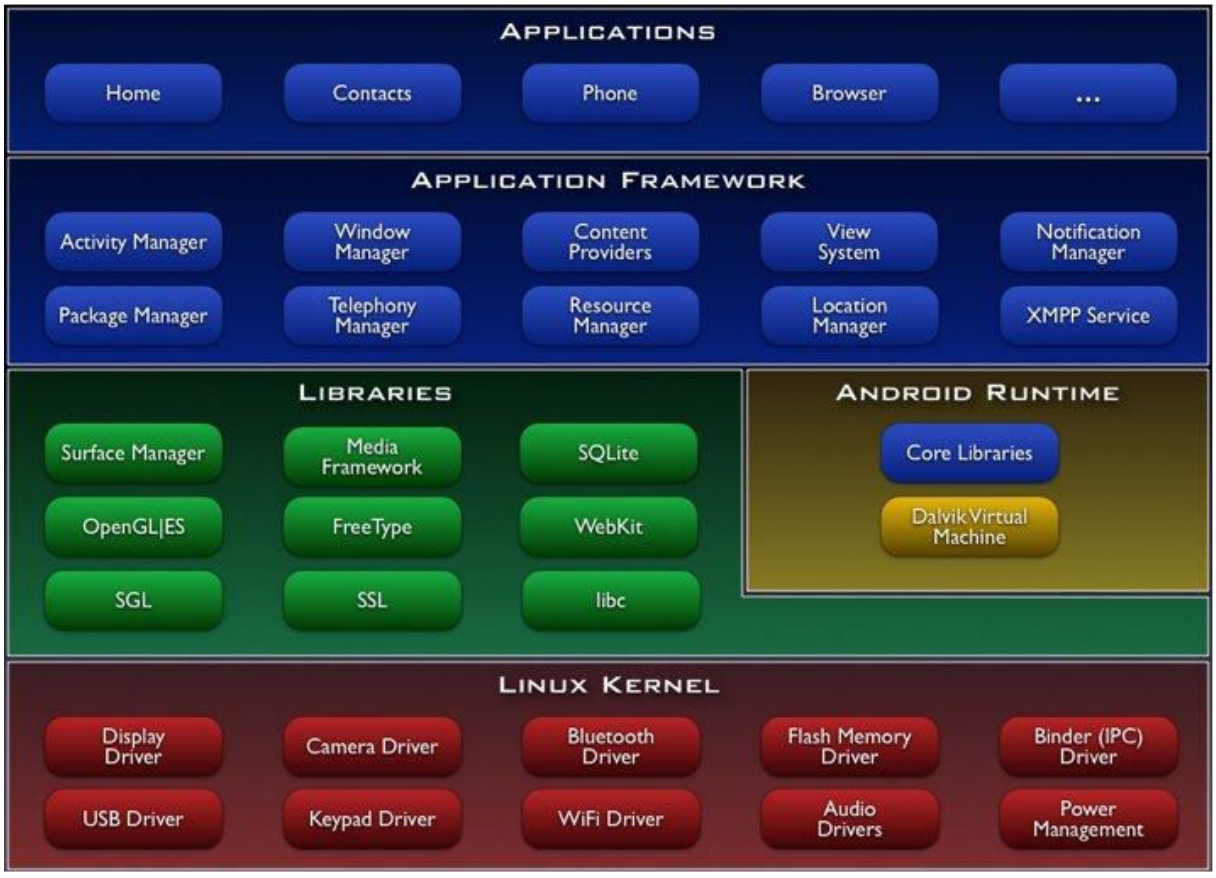

Figura 7 – Arquitetura da Plataforma Android (GOOGLE, 2008)

Na camada *Application Framework*, localizada abaixo da camada *Applications*, estão os componentes que permitirão que novas estruturas sejam utilizadas por futuras aplicações, simplificando a reutilização de código. A seguir estão citados os componentes que fazem parte desta camada:

- Um rico e extensível conjunto de componentes gráficos que pode ser utilizado para construir uma aplicação, bem como listas, caixas de textos, *grids*, botões e também um navegador web embutido;
- Provedores de conteúdo que habilitam as aplicações acessar dados de outras aplicações ou compartilhar seus próprios dados;
- Gerenciador de recursos que provê acesso a recursos não codificados como *strings*, gráficos e arquivos de *layout*;
- Gerenciador de notificação que permite que todas as aplicações exibam mensagens de alerta personalizáveis na barra de status;
- Gerenciador de atividade que gerencia o ciclo de vida das aplicações e permite controlar os recursos previamente alocados.

Na camada *Libraries* é composto por uma coleção de bibliotecas escritas em C/C++ que são utilizadas pela plataforma Android. As bibliotecas são:

- Biblioteca de sistema C – é uma implementação da biblioteca C padrão, otimizada para dispositivos que suportam a plataforma Linux;
- Bibliotecas de mídias – suportam a execução e a gravação da maioria dos formatos de áudio e vídeo, incluindo os formatos *MPEG4*, *H.264*, *MP3*, *AAC*, *AMR*, *JPG* e *PNG*;
- Gerenciador de superfície – gerencia o acesso ao display do dispositivo e camadas de gráficos *2D* e *3D* de múltiplas aplicações;
- LibWebCore – um moderno navegador web;
- *SGL* – uma engine de gráficos 2D;
- *3D* Libraries – uma implementação baseada na especificação OpenGL ES 1.0, a qual utiliza tanto a aceleração de hardware 3D e um avançado e otimizado software para renderização de modelos tridimensionais;
- FreeType – renderização de formatos bitmaps e vetorias de fontes;
- SQLite – uma poderosa e leve engine de banco de dados relacional.

A camada *Runtime*, ambiente de execução, é composta pela máquina virtual Dalvik. Essa máquina virtual foi escrita para que os dispositivos possam suportar múltiplas máquinas virtuais eficientemente. A Dalvik executa arquivos no formato *Dalvik Executable*, com a extensão .dex. O arquivo .dex é uma espécie de bytecode de Java otimizados para a plataforma Android.

A última camada, localizada na base, é denominada *Kernel Linux*, que para o Android é utilizada a versão 2.6, fornecendo serviços do núcleo do sistema como segurança, gerenciamento de memória, gerenciamento de processos, pilhas de redes e modelos de *drivers*. O Kernel é a camada entre o hardware e o software.

Uma aplicação Android é composta pelos componentes: *Activity, Intent Receiver, Service* e *Content Provider*, descritos a seguir:

- *Activity* (Atividade) – é implementada como classe que estende *Activity*, exibindo uma interface para o usuário composta por *Views*;
- *Intent Receiver* – utilizado quando a aplicação precisa ser executada em reação a eventos externos, como uma ligação recebida;
- *Service* – são códigos que permanecem executando sem uma interface gráfica, como um *download* sendo realizado;
- *Content Provider* – provedores de conteúdo e devem ser utilizados quando a aplicação necessita compartilhar dados com outras aplicações.

O ciclo de vida de uma aplicação Android é definido pelos métodos de uma *Activity,* que podem ser sobrescritos para que realizem uma determinada função requerida pela aplicação. Na Figura 8 é apresentado o ciclo de vida de uma aplicação Android.

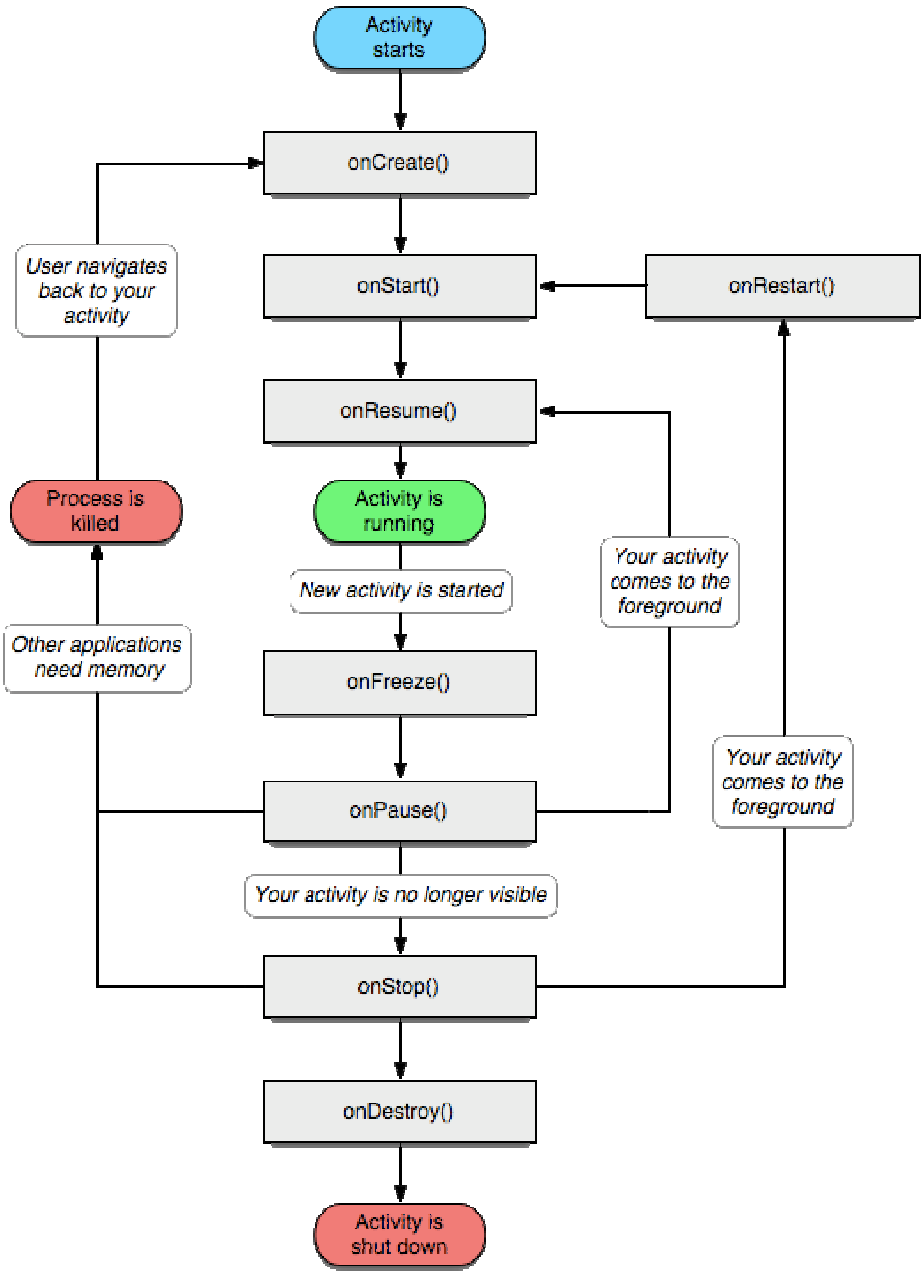

Figura 8 – Ciclo de Vida de uma Aplicação Android (GOOGLE, 2008)

Complementando o modelo de ciclo de vida de uma aplicação Android, na Tabela 1 é apresentado detalhadamente quando é realizada a chamada de um método para a mudança de estado da aplicação.

Tabela 1 – Descrição do Ciclo de Vida de uma Aplicação Android

Método	**Descrição**	**Próximo**
`oncreate()`	Chamado quando a atividade é criada pela primeira vez. Este método proporciona um *bundle* que contém o estado congelado anterior da atividade caso exista.	Sempre é seguido de um `onresume()`.
`onstart()`	Chamado quando uma atividade se torna visível ao usuário.	Seguida por um `onresume()` se a atividade foi criada pela primeira vez, ou `onrestart()` se estiver sendo exibida novamente depois de ter sido parada.
`onrestart()`	Chamado após a atividade ter sido parada, antes de ser resumida novamente.	Seguida pelo `onresume()`.
`onresume()`	Chamado quando a atividade vai começar a interagir com o usuário.	Seguido por `onfreeze()` se outra atividade estiver sendo utilizada antes, ou `onpause()` se esta atividade está sendo encerrada.
`onfreeze()`	Possibilita salvar o estado atual. Quando a atividade foi pausada e outra atividade está sendo resumida para interagir com o usuário. Após pausada, o sistema pode, a qualquer momento, precisar pará-la (ou até mesmo encerrá-la) para poder carregar os recursos para a atividade atualmente ativa na tela. Se isto acontecer, o estado que é fornecido aqui vai ser retornado ao `oncreate()` quando uma nova instância da atividade for iniciada para interagir com o usuário.	Sempre seguida por `onpause()`.
`onpause()`	Chamada quando o sistema está para resumir uma atividade anterior. Isto é geralmente utilizado para submeter alterações não salvas de forma persistente ou outras ocorrências que possam estar consumindo CPU, etc. Implementações deste método devem ser bem rápidas por que a próxima atividade não vai ser resumida até que este método retorne.	Seguida por um `onresume()` se a atividade retorna para a frente ou `onstop()` se ela se torna invisível ao usuário.
`onstop()`	Chamada quando a atividade não está mais visível ao usuário por que outra atividade foi resumida e está cobrindo esta. Isso pode acontecer porque a nova atividade está sendo iniciada, uma existente está sendo trazida para frente desta ou esta está sendo destruída.	Seguida por `onrestart()` se esta atividade está retornando para interagir com o usuário, ou `ondestroy()` se esta atividade está sendo encerrada.

`ondestroy()`	Chamada final de método que é realizada antes que a atividade seja destruída. Isto pode acontecer porque a atividade está encerrando (o método `finish()` foi chamado) ou porque o sistema está destruindo temporariamente esta instância para liberar memória. Estes dois cenários podem ser distinguidos por meio do método `isfinishing()`.	Nada.

Fonte: GOOGLE, 2008

A unidade funcional básica de uma aplicação Android é uma *Activity*, que é um objeto da classe `android.app.Activity`. Uma *Activity* pode fazer muitas coisas, mas não sem a presença de uma tela. Para trabalhar com uma *Activity* é necessário uma interação gráfica com o usuário através da *UI (User Interface)*. É possível trabalhar com *views* e *viewgroups*, que são unidades básicas de controle de interface com o usuário na plataforma Android, descritos a seguir.

2.3.1.1 *Views*

Uma *View* é um objeto da classe `android.view.View` e tem uma estrutura de dados que define as propriedades de *layout* e especifica a área retangular de uma tela. Um objeto `View` controla os eventos de *layout*, desenho, mudança de foco, rolagem e teclas na área de sua representação.

A classe `View` serve como base para a classe de *widgets*, que é um conjunto de subclasses que renderizam elementos interativos na tela no qual define a interface com o usuário. Os elementos disponíveis são *`Text`*, *`EditText`*, *`InputMethod`*, *`MovementMethod`*, *`Button`*, *`RadioButton`*, *`CheckBox`* e *`ScrollView`*.

2.3.1.2 *ViewGroups*

Uma *ViewGroup* é um objeto da classe `android.view.ViewGroup`. É um tipo especial de objeto `View` cuja função é incluir e gerenciar um conjunto subordinado de objetos *`Views`* e *`ViewGroups`*. `ViewGroup` permite adicionar a sua estrutura de *UI (User Interface)* e construir um tela de estrutura complexa com vários elementos e ser endereçada por uma única entidade. Uma *ViewGroup*, por exemplo, pode adicionar a seu nó principal duas *View*,

sendo que a primeira *View* é responsável por listar informações na parte superior e a segunda *View* é responsável por instanciar botões para controlar as ações sobre os registros selecionados.

Na plataforma Android uma *Activity UI* é definida usando uma árvore de nós de *View* e *ViewGroups* como mostrado na Figura 9.

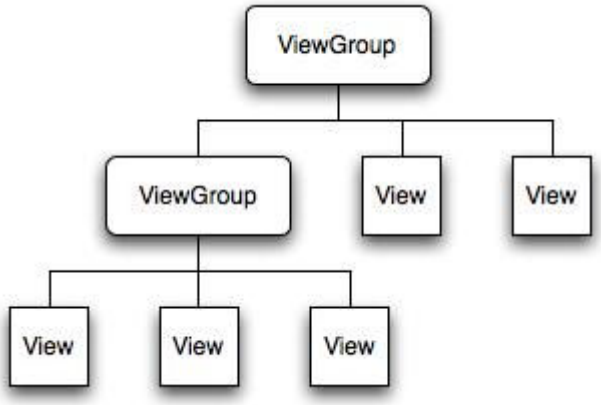

Figura 9 – Estrutura de Árvore de uma *UI* (*User Interface*) (GOOGLE, 2008)

Para renderizar uma estrutura de árvore de *UI (User Interface),* a *Activity* utiliza o método `setContentView()` passando como referência o objeto nó. Caso o objeto referência for um nó *ViewGroup* automaticamente é renderizado todos os nós filhos.

2.4 Considerações Finais

Nesta seção foram apresentadas a estrutura e as características que compõem duas tecnologias, sendo a plataforma J2ME e Android, utilizadas para a implementação de aplicações móveis que serão utilizadas nesse trabalho.

A seção a seguir são apresentadas métricas de software, no contexto de qualidade de software, que serão utilizadas para a avaliação das aplicações móveis desenvolvidas.

CAPÍTULO 3 – QUALIDADE DE SOFTWARE

3.1 Considerações Iniciais

Nesta seção são abordadas as características que compõem a qualidade de software com base na Norma NBR 13596 e apresentado um modelo para avaliar a presença de tais características nas aplicações apontando se as mesmas estão dentro de critérios bem definidos de aceitação.

3.2 Uma Visão sobre Qualidade de Software

As empresas de desenvolvimento de software, nos últimos 15 anos, adotaram novas técnicas, tecnologias e também houve uma maior conscientização da importância do gerenciamento de qualidade de software e da adoção de técnicas específicas para isso. Com isso foi obtido um ganho expressivo na qualidade de software (SOMMERVILLE, 2007).

Segundo Sommerville (2007) para que um novo projeto de software tenha garantia de qualidade (*QA – Quality Assurance*) é necessário definir o processo de como a qualidade de software pode ser atingida e como a organização de desenvolvimento sabe que o software possui o nível de qualidade necessário.

Tal processo está relacionado a seleção e definição de padrões que devem ser aplicados no processo de desenvolvimento de software ou no produto de software.

Sommerville (2007) estabeleceu dois tipos de padrões como parte do processo de garantia de qualidade, sendo:

- *Padrões de produto:* são aplicados ao produto de software em fase de desenvolvimento. São definidos padrões de documentos (estrutura de documentos de requisitos), padrões de codificação (como uma linguagem de programação será utilizada) e padrões de documentação (como um cabeçalho de comentário para uma definição de classe de objeto).
- *Padrões de processo:* são padrões que definem os processos que serão utilizados no desenvolvimento de software. São definidos processos de especificação, projeto e validação.

Os padrões de processo e produto são estritamente relacionados, sendo que os padrões de produto são aplicados à saída do processo de software e os padrões de processo incluem atividades que certificam que os padrões de produto sejam realizados.

Sob esta perspectiva, diferentes razões justificam que padrões de software são importantes (SOMMERVILLE, 2007):

- baseados nas melhores e mais apropriadas práticas para a empresa, ajudando a evitar a repetição de erros cometidos no passado.
- como os padrões englobam as melhores práticas, a garantia de qualidade é assegurada, pois estão sendo utilizados padrões que foram selecionados e usados, constituindo um arcabouço para a implementação do processo de garantia de qualidade.
- os padrões asseguram que todos na empresa adotem as mesmas práticas, tendo como conseqüência o esforço de aprendizado reduzido quando é iniciado um novo trabalho ajudando na continuidade quando o trabalho realizado por uma pessoa é assumido e continuado por outra pessoa.

O desenvolvimento de padrões de software, em geral, é um processo difícil e demorado. Instituições nacionais e internacionais, como a *ANSI (American National Standards Institute), IEEE (Institute of Electrical and Electronic Engineers), ISO (International Organization for Standardization), US DoD (United States Departament of Defense), BSI (British Standards Institution) e NATO (North Atlantic Treath Organization)* têm sido ativas na produção de padrões. Esses padrões são gerais podendo ser utilizados na aplicação em uma variedade de projetos.

Existem diferentes normas para serem utilizadas em diferentes contextos da área de Computação (segurança da informação, teste, ciclo de vida de processos de desenvolvimento, avaliação da maturidade de processos de software, qualidade do produto de software, dentre outros). Neste trabalho, a norma NBR 13596, que é a tradução do texto base da Norma ISO/IEC 9126, está inserida no contexto da qualidade do produto de software, que é de interesse deste trabalho, ou seja, observar a qualidade de aplicações móveis. Assim, esta norma será utilizada neste trabalho como base para a avaliação das aplicações móveis desenvolvidas e é apresentada na próxima seção.

3.3 Norma NBR 13596

A elaboração da Norma NBR 13596 (1996) foi realizada por profissionais especializados de empresas, universidades e centros de pesquisa através da tradução do texto base da Norma ISO/IEC 9126 "*Information Technology – Software Product Quality*". A Comissão de Estudo de Qualidade de Software foi motivada a não criar uma nova norma, pois a integração crescente da economia mundial está uniformizando os conceitos de qualidade, exigindo a utilização de textos normativos comuns a todos os países.

A NBR 13596 define quais são as características que um produto de software deve conter e oferece um modelo para ser efetuada uma avaliação de verificação da presença destas características. Adicionalmente, tal modelo pode ser utilizado nas seguintes situações (NBR 13596, 1996):

- definição dos requisitos de qualidade de um produto de software;
- avaliação da especificação de software para verificar se irá satisfazer aos requisitos de qualidade durante o desenvolvimento;
- descrição de particularidades e atributos do software implementado;
- avaliação do software desenvolvido, antes da entrega;
- avaliação do software desenvolvido, antes da aceitação.

A seguir são apresentadas as características de qualidade que um produto de software deve conter e que devem ser consideradas em cada situação apresentadas anteriormente.

3.3.1 Características de Qualidade de Software

A NBR 13596 define seis características de qualidade de software: Funcionalidade, Usabilidade, Confiabilidade, Eficiência, Manutenibilidade e Portabilidade, conforme mostrado na Tabela 2.

A característica de *Funcionalidade* define os requisitos funcionais que o software ou componentes do software devem executar. A funcionalidade diz respeito à finalidade a que se propõe o produto de software e é, portanto, a principal característica de qualidade para qualquer tipo de software.

Para cada característica é apresentada, de forma informativa, a subdivisão em subcaracterísticas:

- *adequação:* atributos do software que evidenciam a presença de um conjunto de funções e sua apropriação para as tarefas especificadas.
- *acurácia:* atributos do software que evidenciam a geração de resultados ou efeitos corretos ou conforme acordados.
- *interoperabilidade:* atributos do software que evidenciam sua capacidade de interagir com sistemas especificados.
- *conformidade:* atributos do software que fazem com que ele esteja de acordo com as normas, convenções ou regulamentações previstas em leis e descrições similares, relacionadas à aplicação.
- *segurança de acesso:* atributos do software que evidenciam sua capacidade de evitar o acesso não autorizado, acidental ou deliberado, a programas e dados.

Tabela 2 – Características de Qualidade de Software

CARACTERÍSTICAS	DEFINIÇÕES
Funcionalidade	Conjunto de atributos que evidenciam a existência de um conjunto de funções e suas propriedades especificadas. As funções são as que satisfazem as necessidades explícitas ou implícitas.
Confiabilidade	Conjunto de atributos que evidenciam a capacidade do software de manter seu nível de desempenho sob condições estabelecidas durante um período de tempo estabelecido.
Usabilidade	Conjunto de atributos que evidenciam o esforço necessário para se poder utilizar o software, bem como o julgamento individual desse uso, por um conjunto explícito ou implícito de usuários.
Eficiência	Conjunto de atributos que evidenciam o relacionamento entre o nível de desempenho do software e a quantidade de recursos usados, sob condições estabelecidas.
Manutenibilidade	Conjunto de atributos que evidenciam o esforço necessário para fazer modificações especificadas no software.
Portabilidade	Conjunto de atributos que evidenciam a capacidade do software de ser transferido de um ambiente para o outro.

Fonte: NBR 13596, 1996

Os requisitos não funcionais que definem os requisitos de qualidade incluem limitações no produto (segurança, desempenho e confiabilidade) e limitações no processo de desenvolvimento (componentes a serem reutilizados, custos, métodos a serem usados no desenvolvimento).

A *Usabilidade* evidencia o esforço necessário para se utilizar o software, bem como o julgamento individual desse uso, por um conjunto explícito e implícito de usuários. Tem como característica ser um produto de fácil utilização, recordação e aprendizado. A satisfação do usuário quando utiliza o produto e a verificação se o produto desempenha eficientemente uma tarefa para qual foi projetado são fatores importantes a serem avaliados.

A seguir são descritas as subcaracterísticas de Usabilidade:

- *inteligibilidade:* atributos do software que evidenciam o esforço do usuário para reconhecer o conceito lógico e sua aplicabilidade.
- *apreensibilidade:* atributos do software que evidenciam o esforço do usuário para aprender a sua aplicação.
- *operacionalidade:* atributos do software que evidenciam o esforço do usuário para sua operação e controle da operação.

A *Confiabilidade* evidencia a capacidade do software de manter seu nível de desempenho sob condições estabelecidas durante um período de tempo estabelecido. Tem limitações decorrentes de defeitos na especificação de requisitos, projeto ou implementação. As falhas decorrentes desses defeitos dependem de como o produto de software é usado e das opções de programa selecionados

A seguir são descritas as subcaracterísticas de Confiabilidade:

- *maturidade:* atributos do software que evidenciam a frequência de falhas por defeitos no software.
- *tolerância a falhas:* atributos do software que evidenciam sua capacidade de manter um nível de desempenho especificado nos casos de falhas no software ou de violação nas interfaces especificadas.
- *recuperabilidade:* atributos do software que evidenciam sua capacidade de restabelecer seu nível de desempenho e recuperar os dados diretamente afetados, em caso de falha, e em tempo e esforço necessários para tal.

A *Eficiência* evidencia o nível de relacionamento entre o nível de desempenho do software e a quantidade de recursos usados, sob condições estabelecidas. Os recursos podem

incluir outros produtos de software, hardware, materiais, serviços de operação, manutenção ou suporte.

A seguir são descritas as subcaracterísticas de Eficiência:

- *comportamento em relação ao tempo:* atributos do software que evidenciam seu tempo de resposta, tempo de processamento e velocidade na execução de suas funções.
- *comportamento em relação aos recursos:* atributos do software que evidenciam a quantidade de recursos usados e a duração de seu uso na execução de suas funções.

A *Manutenibilidade* evidencia o esforço necessário para fazer modificações especificadas no software. É a facilidade de o programa ser adaptado se o ambiente mudar ou melhorado se o cliente desejar alguma mudança nos requisitos e corrigido se um erro é encontrado.

A seguir são descritas as subcaracterísticas de Manutenibilidade:

- *analisabilidade:* atributos do software que evidenciam o esforço necessário para diagnosticar deficiências ou causas de falhas ou para identificar partes a serem modificadas.
- *modificabilidade:* atributos do software que evidenciam o esforço necessário para modificá-lo, remover seus defeitos ou adaptá-lo a mudanças ambientais.
- *estabilidade:* atributos do software que evidenciam o risco de efeitos inesperados ocasionados por modificações.
- *testabilidade:* atributos do software que evidenciam o esforço necessário para validar o software modificado.

A *Portabilidade* evidencia a capacidade do software de ser transferido de um ambiente para o outro. Por ambiente se entende os casos de ambiente organizacional, de hardware ou de software.

A seguir são descritas as subcaracterísticas de Portabilidade:

- *adaptabilidade:* atributos do software que evidenciam sua capacidade de ser adaptado a ambientes diferentes especificados, sem a necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado.
- *capacidade para ser instalado:* atributos do software que evidenciam o esforço necessário para a sua instalação num ambiente especificado.
- *capacidade para substituir:* atributos do software que evidenciam sua capacidade e esforço necessário para substituir outro software, no ambiente estabelecido para esse outro software.

- *conformidade:* atributos do software que o tornam consonante com padrões ou convenções relacionados à portabilidade.

A Norma NBR 13596 além de definir as características de qualidade do produto de software e orientar quando devem ser utilizadas, também apresenta um modelo para aplicação da qualidade de software que será discutido na próxima seção.

3.3.2 Modelo de Qualidade de Produto de Software

O modelo de qualidade de produto de software da Norma NBR 13596 foi definido com o objetivo de apoiar a aplicação das seis características de qualidade em uma avaliação de produto de software. Em uma avaliação é realizada uma comparação entre o valor medido com um modelo pré-estabelecido. O modelo apresentado é baseado em subcaracterísticas da definição das características de qualidade de software, apresentadas na seção anterior.

Além de definidas as características e subcaracterísticas da qualidade de um produto de software é necessário estabelecer quais serão as métricas, níveis de pontuação e os critérios de aceitação (NBR 13596, 1996). De acordo com a Norma ISO/IEC 14598-1, Métrica, Nível de Pontuação e Critérios de Aceitação são definidos como:

- *métrica:* um método e uma "escala de medição". Por escala entende-se: um conjunto de valores com propriedades definidas e; por medição entende-se: a determinação de um valor (podendo ser um número ou uma categoria) para um atributo de uma entidade.
- *nível de pontuação:* pontuações de uma escala ordinal que são utilizadas para categorizar uma escala de medição.
- *critério de aceitação:* pré-determinação dos níveis de pontuações considerados satisfatórios ou não satisfatórios para um atributo de uma entidade.

A relação existente entre as métricas, nível de pontuação e critérios de aceitação é mostrado na Figura 10. Como as métricas, nível de pontuação e critérios de aceitação podem variar de acordo com a organização ou a aplicação, a Norma NBR 13596 não a define.

Alguns fatores que devem ser considerados em uma avaliação de produto de software são: o grau de importância de cada característica de qualidade e sob qual o ponto de vista do produto de software que está sendo analisado. Essa análise é subjetiva em uma

avaliação de diferentes produtos de software, pois o grau de importância de cada característica de qualidade varia de software para software, portanto é interessante ponderar características de qualidade através de um sistema de pesos proporcionais à sua importância na área de aplicação.

Um exemplo de um sistema de ponderação de pesos por meio de características de qualidade de acordo com a sua área da aplicação é a característica de "Eficiência", que em um produto de software em tempo real tem um peso maior que as outras características.

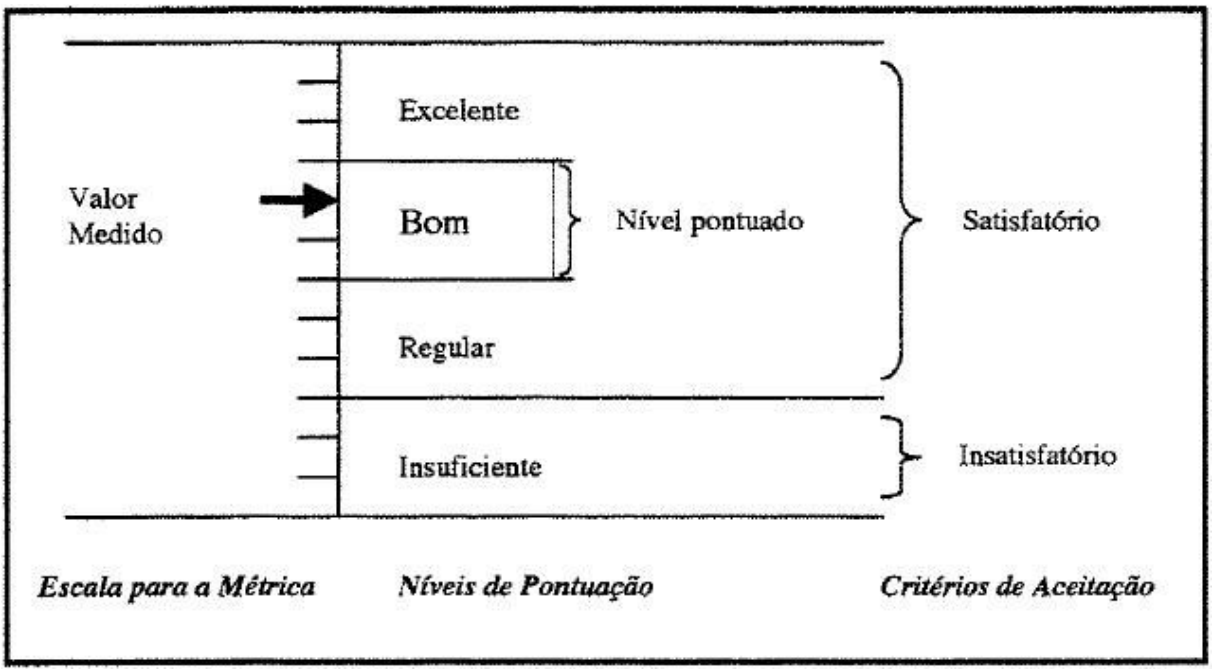

Figura 10 – Valor Medido, Níveis de Pontuação e Critérios de Aceitação (NBR 13596, 1996)

Em uma avaliação de Qualidade de Software, a Norma NBR 13596 aponta três tipos de análises: do usuário, da equipe de desenvolvimento e do gerente.

- o usuário não está interessado nos aspectos internos do software, mas sim no uso. Está interessado em saber se as funções requeridas estão disponíveis, se é confiável, eficiente e fácil de usar.
- a equipe de desenvolvimento está interessada na qualidade em seus aspectos internos e também na qualidade do produto final. O processo de desenvolvimento requer que o usuário e a equipe de desenvolvimento utilizem as mesmas características de qualidade de software, uma vez que elas se aplicam tanto para os requisitos como para a aceitação.

- o Gerente está interessado na qualidade de forma geral e não nas características específicas de qualidade. Tem como responsabilidade a otimização da qualidade dentro das limitações de custos, recursos humanos e prazos tendo uma visão comercial.

A Norma NBR 13596 apresenta um modelo para a avaliação da qualidade de um produto de software com base no desmembramento das seis características de qualidade em dezoito subcaracterísticas. Também lembra a necessidade de se definir para cada caso, as métricas, níveis de pontuação, critérios de aceitação, grau de importância de cada característica através de pesos e o ponto de vista a ser considerado na avaliação.

Por definir um modelo subjetivo, a Norma permite a abertura para a aplicação do modelo nas diversas organizações de software e em diferentes segmentos. A sua versatilidade em suprir diferentes necessidades, permite a aplicação da Norma NBR 13596 em diversos estágios do processo de software em uma organização.

3.4 Considerações Finais

Nesta seção foram apresentadas detalhadamente as características e as subcaracterísticas definidas na Norma NBR 13596. Neste trabalho serão utilizadas as características de Funcionalidade, Usabilidade e Portabilidade para efetuar a avaliação da presença dessas características nas aplicações móveis que serão implementadas na seção seguinte.

CAPÍTULO 4 – ESTUDO DE CASO: DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

4.1 Considerações Iniciais

Neste capítulo será discutido o desenvolvimento e a avaliação, por meio de métricas, de duas aplicações móveis utilizando plataformas diferentes, sendo uma em J2ME e outra em Android. Ambas as aplicações possuem a mesma funcionalidade, relacionada ao domínio de agenda eletrônica.

4.2 Descrição do Estudo de Caso

O domínio de aplicação escolhido para a implementação das aplicações móveis em J2ME e Android é o de agenda eletrônica. Assim, as aplicações devem permitir a inclusão, alteração, remoção e visualização de contatos em uma agenda eletrônica, bem como efetuar ligação para um contato. Na implementação de ambas aplicações são utilizados recursos gráficos de interfaces, fluxo de telas e persistência de dados.

Uma das aplicações é desenvolvida em J2ME e a outra em Android.

A plataforma *J2ME* foi escolhida por ser amplamente utilizada no desenvolvimento de aplicações móveis tendo como principal característica a portabilidade, ou seja, tem a capacidade do software ser transferido de um ambiente para o outro sem a necessidade da aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado (NBR 13596).

No desenvolvimento da aplicação em J2ME é utilizada a *API RMS (Record Management System)* que oferece construtores e métodos para o gerenciamento da persistência dos dados no dispositivo móvel.

A plataforma Android foi escolhida pois é uma linguagem nova e tem como objetivo principal trazer inovação ao espaço da mobilidade e acelerar a evolução na forma como as pessoas acessam a Internet pelo celular.

4.3 Documentação das Aplicações Móveis

A documentação das aplicações foi elaborada utilizando os diagramas de casos de uso e de classes da UML (*Unified Modeling Language*). A UML é uma linguagem para especificação e visualização de sistemas e possui diversos diagramas para representar o sistema nas diversas fases do desenvolvimento, ou seja, desde o levantamento de requisitos até a sua implantação. O diagrama de casos de uso, conhecido também como diagrama de *use cases*, documenta os requisitos funcionais e os atores que interagem com o sistema. O diagrama de classes, mostra as classes, os relacionamentos e as colaborações entre as mesmas a fim de representar o sistema do ponto de vista estrutural.

Na Figura 11 é ilustrado o diagrama de casos de uso da aplicação de agenda eletrônica, sendo que o usuário, representa o ator do sistema (utilizador do sistema) e é responsável por iniciar todos os casos de uso da agenda, ou seja, efetuar ligação a partir de um contato, visualizar dados do contato, excluir contato, alterar dados do contato, cadastrar novo contato e listar todos os contatos.

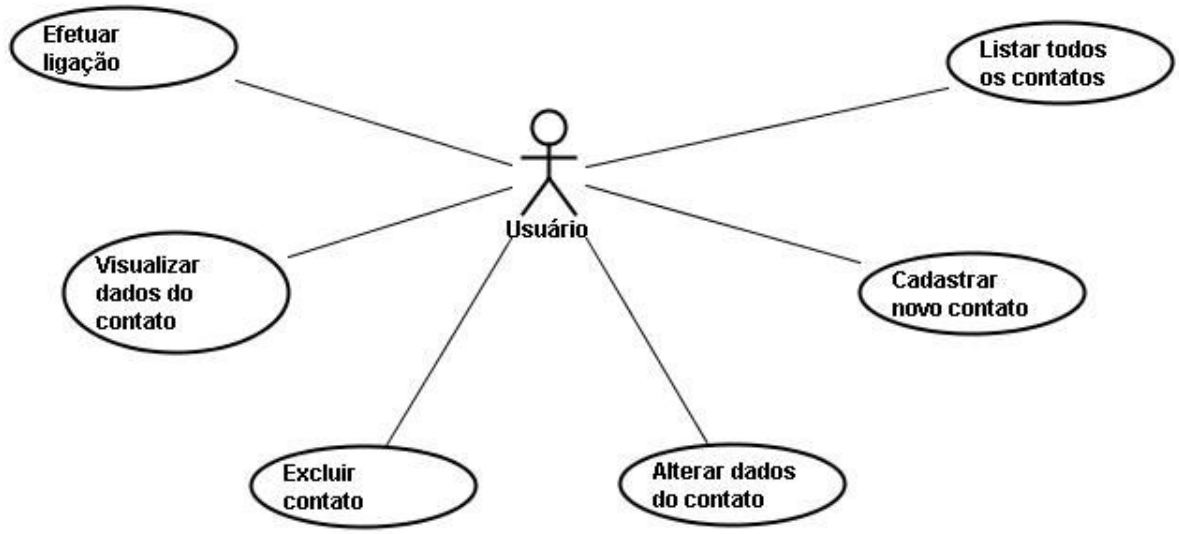

Figura 11 – Diagrama de Caso de Uso da aplicação de Agenda Eletrônica

Para facilitar o entendimento da aplicação de agenda eletrônica, a seguir é apresentada a descrição de cada caso de uso do diagrama apresentado na Figura 11.

UC01 – Listar todos os contatos

Ator: Usuário.

Descrição: Usuário solicita visualização dos contatos e recebe uma lista na tela do dispositivo móvel, para maiores detalhes consulte a Tabela 3.

Tabela 3 – Descrição de Caso de Uso: Listar todos os contatos

Listar todos os contatos
Fluxo principal de eventos 1. O caso de uso inicia o sistema e exibe para o usuário a tela de apresentação da agenda eletrônica. 2. O usuário seleciona a opção de listar todos os contatos. 3. O sistema mostra para o usuário todos os contatos cadastrados na agenda. Fluxo excepcional de eventos 2.1. O usuário pode selecionar a opção de sair do sistema.

UC02 – Cadastrar novo contato

Ator: Usuário.

Descrição: Este caso de uso descreve o procedimento do cadastro de um novo contato, para maiores detalhes consulte a Tabela 4.

Tabela 4 – Descrição de Caso de Uso: Cadastrar novo contato.

Cadastrar novo contato
Fluxo principal de eventos 1. O caso de uso inicia com um novo código de contato gerado automaticamente pelo sistema e todos os campos do cadastro de contato em branco. 2. O usuário deve informar o nome, o telefone residencial, o telefone comercial, o celular, o fax e o e-mail do contato. 3. O usuário deve selecionar a opção gravar para persistir as informações. Fluxo excepcional de eventos 2.1 O usuário pode cancelar a operação a qualquer momento.

UC03 – Alterar dados do contato

Ator: Usuário.

Descrição: Este caso de uso descreve o procedimento de como alterar dados de um contato, para maiores detalhes consulte a Tabela 5.

Tabela 5 – Descrição de Caso de Uso: Alterar dados do contato.

Alterar dados do contato
Fluxo principal de eventos 1. O caso de uso inicia com todos os contatos listados na tela para o usuário. 2. O usuário deve selecionar o contato que deseja alterar as informações. 3. O usuário deve selecionar a opção de "Alterar dados". 4. O usuário deve realizar as alterações necessárias. 5. O usuário deve selecionar a opção gravar para persistir as informações alteradas. Fluxo excepcional de eventos 2.1 ou 3.1 ou 4.1 ou 5.1. O usuário pode cancelar a operação a qualquer momento.

UC04 – Excluir contato

Ator: Usuário.

Descrição: Este caso de uso descreve o procedimento de como excluir um contato, para maiores detalhes consulte a Tabela 6.

Tabela 6 – Descrição de Caso de Uso: Excluir contato.

Excluir contato
Fluxo principal de eventos 1. O caso de uso inicia com todos os contatos listados na tela do usuário. 2. O usuário deve selecionar o contato que deseja excluir da agenda. 3. O usuário deve selecionar a opção de "Excluir contato". 4. O usuário deve confirmar a exclusão do contato. Fluxo excepcional de eventos 2.1 ou 3.1 ou 4.1. O usuário pode cancelar a operação a qualquer momento.

UC05 – Visualizar contato

Ator: Usuário.

Descrição: Este caso de uso descreve o procedimento de como visualizar um contato pelo usuário do dispositivo móvel, para maiores detalhes consulte a Tabela 7.

Tabela 7 – Descrição de Caso de Uso: Visualizar contato.

Visualizar contato
Fluxo principal de eventos 1. O caso de uso inicia com todos os contatos listados na tela do usuário. 2. O usuário deve selecionar o contato que deseja visualizar as informações detalhadamente. 3. O usuário deve selecionar a opção de "Visualizar contato". 4.1 O usuário deve selecionar a opção de "OK" após visualizar as informações detalhadas do contato. Fluxo excepcional de eventos 2.1 ou 3.1 ou 4.1. O usuário pode cancelar a operação a qualquer momento.

UC06 – Efetuar ligação

Ator: Usuário

Descrição: Este caso de uso descreve o procedimento de como efetuar uma ligação através do dispositivo móvel, para maiores detalhes consulte a Tabela 8.

Tabela 8 – Descrição de Caso de Uso: Efetuar ligação

Efetuar ligação
Fluxo principal de eventos 1. O caso de uso inicia com todos os contatos listados na tela do usuário. 2. O usuário deve selecionar o contato que deseja realizar uma ligação. 3. O usuário deve selecionar a opção de "Efetuar ligação". Fluxo excepcional de eventos 2.1 ou 3.1. O usuário pode cancelar a ligação a qualquer momento.

Na Figura 12 é ilustrado o diagrama de classes da aplicação de agenda eletrônica, ou seja, a classe `Contatos`, contendo os atributos e métodos necessários para a inserção, alteração, remoção e visualização de um contato, bem como ligação para um contato.

O atributo `codigo` identifica o código do contato e os atributos *`nome`*, *`telefoneR`*, *`telefoneC`*, *`celular`*, *`fax`* e *`email`* identificam os telefones e email de contato.

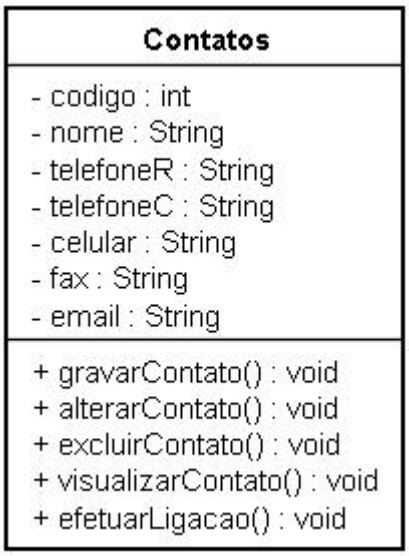

Figura 12 – Diagrama de Classes da aplicação de Agenda Eletrônica

4.4 Implementação da Aplicação Móvel em J2ME

Para iniciar a fase de implementação da aplicação móvel na plataforma *J2ME* (*Java Micro Edition*) foi necessário definir a sua "configuração" e "perfil". A escolha de "configuração" e "perfil" é o que define os recursos de bibliotecas básicas da *JVM* (*Java Virtual Machine*) para o desenvolvimento de aplicativos, para maiores detalhes consulte as Seções 2.2.1 e 2.2.2.

Definiu-se o nível de configuração *CLDC* (*Connected Limited Device Configuration*), por tratar-se de uma aplicação que irá utilizar pouco recurso de memória, não utilizará recurso de conectividade de rede e não realizará operações matemáticas usando ponto flutuante. Quanto ao nível de perfil foi definido utilizar o *MIDP* (*Mobile Information Device Profile*) por apresentar um modelo de controle de ciclo de vida da uma aplicação móvel chamado de *MIDLets*, armazenamento persistente de dados por meio da *API RMS* (*Record Management Store*).

O desenvolvimento da aplicação de agenda eletrônica utilizando a plataforma J2ME inicia-se com a classe principal chamada `Agenda` estendendo a classe MIDlet e implementado a interface `CommandListener` conforme apresentado na Figura 13.

```
...
public class Agenda extends MIDlet implements CommandListener {

    private boolean midletPaused = false;
    private RecordStore bancoDados;
    private RecordEnumeration rEnum;
    private boolean novo = true;
    ...
```

Figura 13 – Classe `Agenda` em J2ME – definição dos atributos

O atributo `bancoDados` da classe `Agenda` é um objeto da classe `javax.microedition.rms.RecordStore`, que oferece métodos para gerenciar a manutenção e recuperação das informações dos contatos cadastrados no banco de dados por meio da *API RMS*. O atributo `rEnum`, objeto da classe `javax.microedition.rms.RecordEnumeration` oferece métodos para realizar a ordenação dos registros sobre os contatos recuperados.

Ao estender a super-classe `MIDlet`, a aplicação (classe `Agenda`) deve implementar três métodos abstratos responsáveis pelo controle do ciclo de vida da aplicação, conforme mostrado a Figura 14.

O método `startApp` é chamado no início da aplicação e também após a saída do estado de pausa. No método `startApp` é implementado uma lógica para saber quando a aplicação está sendo iniciada ou se está saindo do estado de pausa. É importante implementar essa lógica, pois dependendo do estado que a aplicação se encontrava vai definir se irá carregar a janela principal ou se irá carregar as informações que se encontravam anteriormente à pausa da aplicação. Uma pausa na aplicação pode ser causada por diversos fatores, como por exemplo, uma ligação sendo recebida.

O método `initialize` instancia os recursos visuais que são mostrados na tela. O método `startMIDlet` chama a primeira tela do sistema, que trata-se de uma tela de apresentação da aplicação da agenda eletrônica, conforme a Figura 15.

Na tela de apresentação há dois recursos que o usuário pode selecionar, sendo de “Listar” e “Sair”.

Caso o usuário selecione o recurso “Sair”, o sistema irá finalizar e voltará o controle ao sistema operacional correspondente ao dispositivo móvel.

```
...
public void startApp() {
    if (midletPaused) {
        resumeMIDlet();
    } else {
        initialize();
        startMIDlet();
    }
    midletPaused = false;
}

public void pauseApp() {
    midletPaused = true;
}

public void destroyApp(boolean unconditional) {

}
...
```

Figura 14 – Classe `Agenda` em J2ME – métodos do ciclo de vida

Se o usuário selecionar o recurso “Listar”, a aplicação irá recuperar os registros que estão persistidos no banco de dados através da instância de um objeto RMS, chamar uma nova tela e listar na tela do usuário todos os registros cadastrados, conforme esboçado na Figura 16.

Uma vez que os registros de contatos estão listados na tela, o usuário tem as opções de incluir, alterar, excluir e visualizar um contato, além da opção de “Sair”, que retorna à tela de apresentação da aplicação.

Se o usuário selecionar a opção “Incluir”, o sistema irá gerar um novo *ID* utilizando o método `bancoDados.getNextRecordID()` e chamar uma nova tela para que o usuário possa inserir as informações do contato, conforme apresentado na Figura 17.

Com a tela para o registro de um novo contato ativa, o usuário tem duas opções para selecionar sendo: “Gravar” ou “Cancelar”.

Se o usuário selecionar a opção “Gravar” o sistema coleta as informações na tela do dispositivo móvel e atribui a uma variável do tipo `String`, concatenando os diferentes campos entre barras com um número seqüencial, com o objetivo de diferenciar a posição de cada campo e possibilitar a recuperação da informação posteriormente. Depois a informação do contato é convertido a uma variável do tipo `Array` de bytes, que será utilizado como parâmetro para a persistência do registro. Em seguida o sistema através do objeto instanciado da classe `RMS`, realiza a persistência do registro no banco de dados através do método

`bancoDados.addRecord(dados, 0, dados.length)`, conforme apresentado na Figura 17.

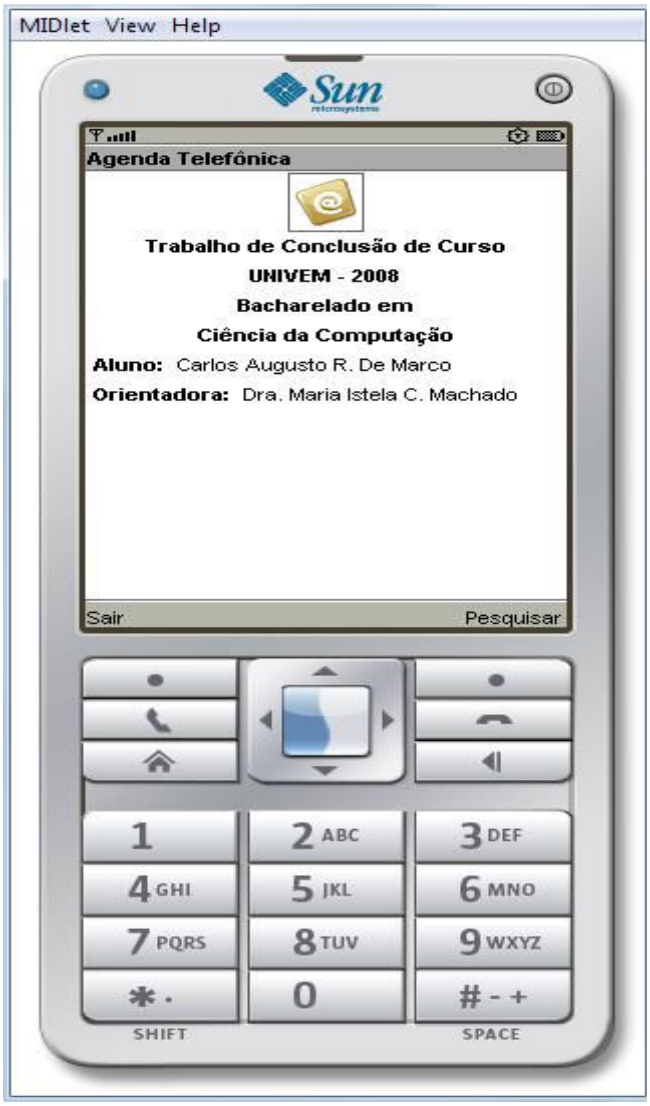

Figura 15 – Tela de Apresentação em J2ME

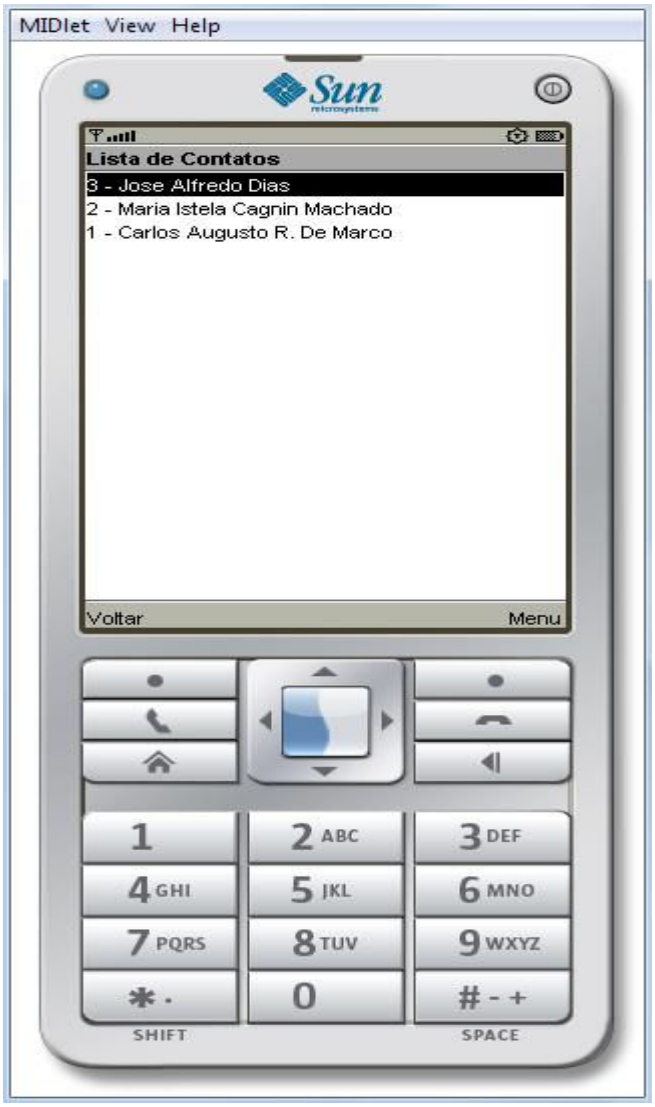

Figura 16 – Listando os Contatos

Se o usuário selecionar a opção “Editar”, o sistema recupera o registro completo através do método `bancoDados.getRecord(ID)`, passando como parâmetro a chave que identifica o registro como sendo único no banco de dados. A informação retorna em um fluxo de bytes, em que é convertida para uma variável do tipo `String` para que possa ser realizado o tratamento adequado. Em seguida a `String` completa contendo todas as informações do contato selecionado é dividida em diversas variáveis `String` que será carregada na tela do usuário em seus respectivos campos em que o usuário poderá editar as informações e depois

persistir os dados através da opção “Gravar”. O processo detalhado acima corresponde ao trecho de código listado na Figura 18.

```
...
try {
    int ID = Integer.parseInt(tfCodigo.getString());

    // Pego as informações no Form
    String registro = tfCodigo.getString() + "1|" +
            tfNome.getString() + "2|" +
            tfTelefoneR.getString() + "3|" +
            tfTelefoneC.getString() + "4|" +
            tfCelular.getString() + "5|" +
            tfFax.getString() + "6|" +
            tfEmail.getString() + "7|";

    // Transformo a String em um tipo de byte
    byte dados[] = registro.getBytes();

    // Se for novo, gravo o registro
    if (novo == true) {
        bancoDados.addRecord(dados, 0, dados.length);

    } // Senão faço a atualização
    else {
        bancoDados.setRecord(ID, dados, 0, dados.length);
    }

    getFmStatus().append("Registro Gravado com Sucesso!");
} catch (Exception e) {
    getFmStatus().append("Erro ao Gravar Registro!");
}
...
```

Figura 17 – Classe `Agenda` J2ME - Persistência de dados utilizando a *API RMS*

Se o usuário selecionar a opção “Deletar”, o sistema remove o registro selecionado através do método `bancoDados.deleteRecord(ID)`, passando como parâmetro a chave que identifica o registro como sendo único no banco de dados.

Na subseção seguinte é feito uma descrição da implementação da aplicação móvel utilizando a plataforma Android.

```
...
byte[] data = bancoDados.getRecord(ID);
String registro = new String(data);

String IDContato = registro.substring(0, registro.indexOf("1|"));
String nome = registro.substring(registro.indexOf("1|") + 2, registro.indexOf("2|"));
String telefoneR = registro.substring(registro.indexOf("2|") + 2, registro.indexOf("3|"));
String telefoneC = registro.substring(registro.indexOf("3|") + 2, registro.indexOf("4|"));
String celular = registro.substring(registro.indexOf("4|") + 2, registro.indexOf("5|"));
String fax = registro.substring(registro.indexOf("5|") + 2, registro.indexOf("6|"));
String email = registro.substring(registro.indexOf("6|") + 2, registro.indexOf("7|"));

getTfCodigo().setString(IDContato);
getTfNome().setString(nome);
getTfTelefoneR().setString(telefoneR);
getTfTelefoneC().setString(telefoneC);
getTfCelular().setString(celular);
getTfFax().setString(fax);
getTfEmail().setString(email);
...
```

Figura 18 – Classe `Agenda` em J2ME- Recuperando registro

4.5 Implementação da Aplicação Móvel em Android

O desenvolvimento da aplicação móvel utilizando a plataforma Android é realizado através da elaboração de arquivos *XML* (*eXtensible Markup Language*) e classes em linguagem de programação Java que estendem a super-classe `Activity`.

A implementação de arquivos *XML* definem as *UIs (User Interface – Interface com o Usuário)*, as configurações que definem a classe que será carregada pelo sistema operacional ao iniciar a aplicação, quais serão os recursos que cada classe terá disponível ao interagir com o sistema operacional e também as referências, geradas automaticamente pelo compilador, entre os recursos criados pelos arquivos *XML*, utilizados para a modelagem da interface, com os recursos que serão manipulados pelas classes que são programadas utilizando a linguagem de programação Java.

Para a persistência de dados é utilizado o banco de dados *SQLite*. É utilizado um objeto da classe `android.database.sqlite.SQLiteDatabase` que expõe métodos para o gerenciamento do banco de dados. A *API* apresenta métodos para criar uma base de dados, recuperar, alterar e excluir informações, executar comandos *SQL (Structured Query Language)* e outras ações comuns para o gerenciamento do banco de dados.

O arquivo *XML* que define as configurações e é utilizado pelo sistema operacional para iniciar a aplicação e define os recursos que cada classe terá disponível para a interação com o sistema operacional é implementado com o nome padrão *AndroidManifest.xml*. Na

Figura 19 é apresentado o arquivo *AndroidManifest.xml* utilizado no desenvolvimento da aplicação móvel.

No arquivo *AndroidManifest.xml são* definidas seis *activity* que correspondem as seis classes implementadas utilizando a linguagem de programação Java estendendo a super-classe `Activity`. Para cada *activity* declarada são atribuídos valores as propriedades "*activity android:name*", "*android:label*" e "*action android:name*". A propriedade "*activity android:name*" define o nome da classe que serão atribuídas as propriedades. A propriedade "*android:label*" define o nome que irá ser mostrado na parte superior quando a *activity* correspondente estiver em execução. A propriedade "*action android:name*" define qual o tipo de ação que cada *activity* terá disponível, sendo que é necessário incluir no mínimo uma *activity* com a propriedade *android.intent.action.MAIN* no desenvolvimento de uma aplicação na plataforma Android, pois essa propriedade define a entrada principal da aplicação.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="android.agenda">
   <application android:icon="@drawable/icon">

       <activity android:name=".Agenda" android:label="@string/app_name">
           <intent-filter>
               <action android:name="android.intent.action.MAIN"/>
               <category android:name="android.intent.category.LAUNCHER"/>
           </intent-filter>
       </activity>

       <activity android:name=".cadastroContato" android:label="@string/app_cadastroContato">
           <intent-filter>
               <action android:name="android.intent.action.VIEW"/>
               <category android:name="android.intent.category.DEFAULT"/>
           </intent-filter>
       </activity>

       <activity android:name=".editarContato" android:label="@string/app_editarContato">
           <intent-filter>
               <action android:name="android.intent.action.VIEW"/>
               <category android:name="android.intent.category.DEFAULT"/>
           </intent-filter>
       </activity>

       <activity android:name=".excluirContato" android:label="@string/app_excluirContato">
           <intent-filter>
               <action android:name="android.intent.action.VIEW"/>
               <category android:name="android.intent.category.DEFAULT"/>
           </intent-filter>
       </activity>

       <activity android:name=".visualizarContato" android:label="@string/app_visualizarContato">
           <intent-filter>
               <action android:name="android.intent.action.VIEW"/>
               <category android:name="android.intent.category.DEFAULT"/>
           </intent-filter>
       </activity>

   </application>
</manifest>
```

Figura 19 – Arquivo de configuração *AndroidManifest.xml*

Para implementar a interface com o usuário é utilizado *XML (eXtensible Markup Language)*, sendo que para cada interface é necessário um arquivo *XML* contendo o layout e os recursos utilizados.

A tela inicial da aplicação móvel foi implementada por meio do arquivo *main.xml* que está apresentado na Figura 20(a) e na Figura 20(b).

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout android:id="@+id/widget30"
               android:layout_width="fill_parent"
               android:layout_height="fill_parent"
               android:orientation="vertical"
               xmlns:android="http://schemas.android.com/apk/res/android">

   <TableLayout android:id="@+id/tabela"
                android:layout_width="fill_parent"
                android:layout_height="364px"
                android:orientation="vertical">

       <ListView android:id="@+id/lista"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content">

       </ListView>
   </TableLayout>
   …
```

Figura 20(a) – Arquivo *main.xml* de implementação de interface

```
   …
   <LinearLayout android:id="@+id/widget65"
                 android:layout_width="fill_parent"
                 android:layout_height="63px">

       <Button android:id="@+id/adicionar"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="Adicionar">
       </Button>

       <Button android:id="@+id/editar"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Editar">
       </Button>

       <Button android:id="@+id/excluir"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Excluir">
       </Button>

       <Button android:id="@+id/visualizar"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Visualizar">
       </Button>
     </LinearLayout>
   </LinearLayout>
```

Figura 20(b) – Arquivo *main.xml* de implementação de interface

Na implementação da interface da tela inicial da aplicação é utilizado o modelo de layout *LinearLayout*, no qual seus filhos são organizados em sequência numa única coluna ou linha. Na interface desenvolvida é utilizada a sequência de filhos em uma única coluna em que é atribuído o valor "vertical" na propriedade "*android:orientation*". É utilizado uma *ListView* em que é apresentado em formato de lista todos os registros de contatos persistidos no banco de dados. Foi inserido quatro *Buttons*, que representam botões, e tem como função disparar os eventos de adicionar, alterar, excluir e visualizar um registro selecionado no *ListView*. Na Figura 21 é apresentada a interface da tela inicial da aplicação.

O usuário tem a funcionalidade de adicionar um novo registro, alterar um registro selecionado, excluir um registro selecionado e visualizar detalhadamente as informações de um registro selecionado.

Se o usuário selecionar a opção "Adicionar", o sistema irá instanciar a *activity* contendo o layout e os métodos necessários para a inclusão de um novo registro através do método `startActivity(intentCadastro)`. O método irá carregar a interface localizada no arquivo *cadastro.xml* e a classe `cadastroContato`.

O método `OnCreate()` é o primeiro método a ser processado quando é ativado a *activity*, sendo aberta uma conexão com o banco de dados através do método `carregarBaseDados()`. Durante a execução desse método é instanciado o objeto `bancoDados` da classe `android.bancodados.banco`, que oferece os recursos necessários para a inclusão, alteração, exclusão e atualização de registros no banco de dados. Assim que o objeto bancoDados é instanciado chama-se o método `bancoDados.abrirConexao()` que realiza a abertura da conexão com o banco de dados.

Em seguida é feito a chamada ao método `proximoID()`, que tem como funcionalidade carregar o próximo *ID* válido para a inclusão de um novo registro. O *ID* corresponde ao campo código da tabela de "Contatos", onde o mesmo é chave primária da tabela, portanto, não podendo ser cadastrado dois registros utilizando o mesmo *ID*. É instanciado um objeto da classe `android.database.Cursor` que recebe o *ID* do último registro válido, assim é incrementado o *ID* e definido o próximo *ID* válido. O código fonte correspondente a essa funcionalidade é visualizado na Figura 22.

Figura 21 – Apresentação da tela inicial da aplicação

O sistema fica em um estado passivo, aguardando a próxima ação do usuário, sendo que o último tem duas opções: 1) informar os dados cadastrais do novo contato e selecionar a opção “Confirmar” ou 2) selecionar a opção “Cancelar” fazendo com que o sistema interrompa toda a operação da inclusão de um novo registro na base de dados.

```
…
private void proximoID() {
    Cursor c = bancoDados.Resultado_SQL("Contatos",
            new String[] { "codigo" }, null, "codigo");
    c.moveToLast();
    int ID = (c.getInt(c.getColumnIndex("codigo")) + 1);
    edCodigo.setText(String.valueOf(ID));
    edCodigo.setEnabled(false);
    edNome.setFocusable(true);
}
…
```

Figura 22 – Método `proximoID()` da classe `cadastroContato`

Caso o usuário selecione a opção “Confirmar” o sistema instancia o objeto da classe `android.agenda.contato` denominado `registro`, carregando os atributos com as informações fornecidas pelo usuário na interface, encapsula as informações em um objeto da classe `android.content.ContentValues` chamado `args` e em seguida chama o método `bancoDados.inserirRegistro()`, passando como parâmetro o nome da tabela e o objeto `args` da classe `android.contenct.ContentValues` contendo as informações dos campos e seus respectivos valores, no qual realiza a persistência dos dados no banco de dados.

A interface do cadastro de um novo contato é visualizada na Figura 23.

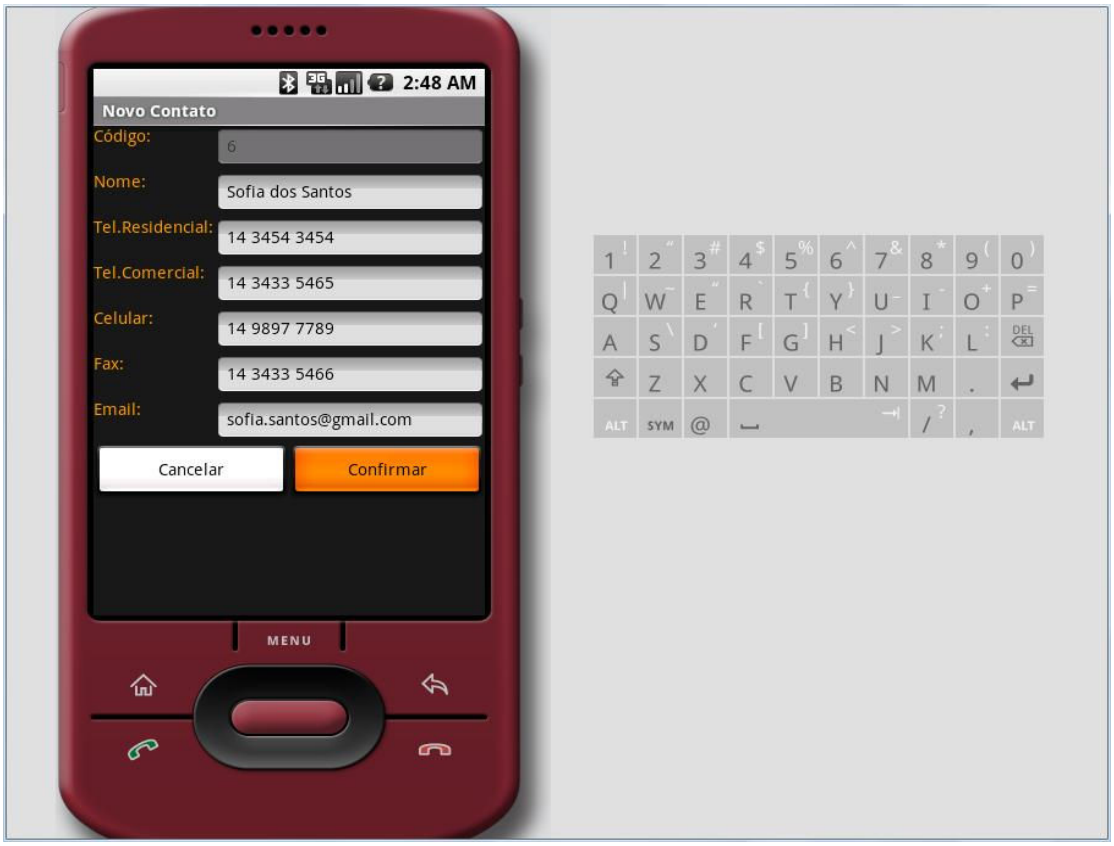

Figura 23 – Cadastro de um novo contato

Na interface principal, se o usuário selecionar a opção “Alterar”, o sistema irá instanciar a *activity* contendo o *layout* e os métodos necessários para a recuperação do registro no banco de dados através do método `startActivity(intentEditar)`. O método irá carregar a interface localizado no arquivo *editar.xml* e a classe `editarContato`.

No método `OnCreate()`, é aberta uma conexão com o banco de dados e é chamado o método `bancoDados.Resultado_SQL()`, passando como parâmetro o nome da tabela, os campos que serão retornados e a condição para realizar a consulta. O resultado da consulta ao

banco de dados retorna um objeto `android.database.Cursor` que permite disponibilizar as informações na interface com o usuário. O código fonte correspondente a tal funcionalidade é visualizado na Figura 24.

```
...
bancoDados = new banco(this);
bancoDados.abrirConexao();

Cursor c = bancoDados.Resultado_SQL("Contatos", new String[] {
        "codigo", "nome", "telResidencial", "telComercial", "celular",
        "fax", "email" }, "codigo = " + String.valueOf(ID), null);
...
```

Figura 24 – Método `Resultado_SQL` da classe `android.bancoDados.banco`

O sistema fica aguardando uma nova ação do usuário: 1) editar as informações cadastrais e selecionar a opção “Confirmar” ou 2) selecionar a opção “Cancelar”.

Caso o usuário edite as informações cadastrais e selecione a opção “Confirmar”, o sistema instancia um objeto da classe `android.agenda.contato` denominado `registro`, carregando seus atributos com as novas informações fornecidas pelo usuário através da interface, encapsula as informações em um objeto da classe `android.content.ContentValues` chamado `args` e em seguida chama o método `bancoDados.atualizarRegistro()`, passando como parâmetro o nome da tabela, o objeto `args` da classe `android.content.ContentValues` contendo as informações dos campos e seus respectivos valores e a condição da atualização, que é o *ID* do registro a ser atualizado. A implementação de tal funcionalidade é apresentada na Figura 25.

Caso o usuário selecione a opção de “Cancelar”, o sistema interrompe o processo de edição do registro e volta para a interface inicial da aplicação.

Na interface principal, se o usuário selecionar a opção “Excluir”, o sistema irá instanciar a *activity* contendo o layout e os métodos necessários para a recuperação e exclusão do registro no banco de dados através do método `startActivity(intentExcluir)`. O método irá carregar a interface especificada no arquivo *excluir.xml* e a classe `excluirContato`.

No método `OnCreate()`, é aberta uma conexão com o banco de dados e chamado o método `bancoDados.Resultado_SQL()`, passando como parâmetro o nome da tabela, os campos que serão retornados e a condição para realizar a consulta. O resultado da consulta é carregado na interface do usuário com todos os campos desabilitados para a edição.

```
…
registro.setCodigo(Integer.valueOf(edCodigo.getText()
        .toString()));
registro.setNome(edNome.getText().toString());
registro.setTelResidencial(edTelResidencial.getText()
        .toString());
registro.setTelComercial(edTelComercial.getText().toString();
registro.setCelular(edTelCelular.getText().toString());
registro.setFax(edFax.getText().toString());
registro.setEmail(edEmail.getText().toString());

ContentValues args = new ContentValues();
args.put("nome", registro.getNome());
args.put("telResidencial", registro.getTelResidencial());
args.put("telComercial", registro.getTelComercial());
args.put("celular", registro.getCelular());
args.put("fax", registro.getFax());
args.put("email", registro.getEmail());

int i = bancoDados.atualizarRegistro("Contatos", args,
        "codigo = " + registro.getCodigo());

if (i == -1) {
    Log.i("[BANCO]", "O registro não foi atualizado!");
} else {
    Log.i("[BANCO]", "Registro atualizado com sucesso!");
}
…
```

Figura 25 – Método `btConfirmar.setOnClickListener` da classe `editarContato`

O sistema fica aguardando uma nova ação do usuário, sendo: 1) selecionar a opção “Confirmar”, em que o sistema irá excluir o registro selecionado ou 2) selecionar a opção “Cancelar”.

Caso o usuário selecione a opção “Confirmar”, o sistema chama o método `bancoDados.deletarRegistro()`, passando como parâmetro o nome da tabela e a condição para a exclusão do registro, sendo que nesse caso a condição é o *ID* do registro selecionado. O código fonte correspondente a remoção do contato é visualizado na Figura 26.

Caso o usuário selecione a opção “Cancelar”, o sistema interrompe o processo de exclusão de registro e volta para a interface inicial da aplicação.

```
…
int i = bancoDados.deletarRegistro("Contatos", "codigo = "
        + registro.getCodigo());

if (i == -1) {
    Log.i("[BANCO]", "O registro não foi deletado!");
} else {
    Log.i("[BANCO]", "Registro deletado com sucesso!");
}
…
```

Figura 26 – Método `btConfirmar.setOnClickListener` da classe `excluirContato`

Na subseção seguinte é realizada uma avaliação das duas aplicações desenvolvidas utilizando as métricas de software descritas no Capítulo 3.

4.6 Avaliação das Aplicações Móveis

Neste trabalho é realizada uma avaliação no contexto de qualidade de produto de software por meio da Norma NBR 13596, apresentada no Capítulo 3, que define as características que um produto de software deve conter e oferece um modelo para ser efetuada a avaliação da verificação da presença dessas características no software. Neste trabalho tal modelo será utilizado para avaliar as aplicações móveis desenvolvidas e apresentadas nas Seções 4.4 e 4.5 utilizando as plataformas *J2ME* e Android, respectivamente.

Para a avaliação das aplicações móveis desenvolvidas neste trabalho foram selecionadas as características de Funcionalidade, Usabilidade e Portabilidade, pois são as características que mais se adéquam ao contexto do caso de uso e das plataformas utilizadas.

Um modelo de qualidade de produto de software, com base na Seção 3.3.2, é proposto neste trabalho com o objetivo de apoiar a avaliação da presença das três características de qualidade de software selecionadas nas aplicações desenvolvidas.

O modelo proposto para a avaliação das aplicações móveis é estabelecido por meio de uma métrica, níveis de pontuação e os critérios de aceitação para cada característica que será avaliada, conforme apresentado na Tabela 9.

Durante a avaliação da presença das características de qualidade nas aplicações móveis desenvolvidas são utilizados dois tipos de análises (Seção 3.3.2): do ponto de vista do usuário e do ponto de vista do desenvolvedor da aplicação.

Para a avaliação é utilizado um questionário com questões relacionadas as características avaliadas e que permite identificar o grau de adequação do sistema a seu

objetivo proposto. Após preenchidos os questionários é realizada a consolidação dos dados, que consiste na obtenção dos valores após a aplicação de funções de ponderação de acordo com a importância de cada critério mediante os objetivos da avaliação.

Tabela 9 – Modelo de classificação

Escala para a métrica	Níveis de pontuação	Critérios de aceitação
3	Excelente	Aceita
2	Bom	Aceitação média
1	Regular	Aceitação baixa
0	Insuficiente	Insatisfatório

O questionário utilizado para coletar as informações das métricas de acordo com a análise dos avaliadores é apresentado em três partes, sendo que cada parte está relacionada a uma característica principal e às suas subcaracterísticas.

O resultado da avaliação de cada característica de qualidade é pontuada por meio da média aritmética das avaliações de suas subcaracterísticas, sendo que a casa decimal acima de seis será considerada na escala de nível superior. O cálculo para a média aritmética é a soma de todos os resultados divido pela quantidade de elementos somados, sendo que a fórmula é apresentada na Figura 25.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Figura 27 – Fórmula para calcular a média aritmética

No questionário inicialmente são apresentadas as questões para a avaliação da característica da qualidade Funcionalidade, baseadas na norma NBR 13596, em que são definidos os requisitos funcionais que o software deve executar. As questões são apresentadas na Tabela 10, em que cada avaliador (usuário) deve definir para cada questão um valor entre 0 e 3.

Tabela 10 – Questionário para avaliação da característica da qualidade Funcionalidade

Subcaracterísticas	Questões	Avaliação
1.1. Adequação	O sistema apresenta a presença de um conjunto de funções e sua apropriação para as tarefas especificadas?	
1.2. Acurácia	O sistema apresenta a geração de resultados ou efeitos corretos ou conforme acordados?	
1.3. Interoperabilidade	O sistema apresenta a capacidade de interagir com os sistemas especificados?	
1.4. Conformidade	O sistema está de acordo com as normas, regulamentações ou convenções relacionadas a aplicação?	
1.5. Segurança de acesso	O sistema apresenta a capacidade de evitar o acesso não autorizado a programas e dados?	

Em seguida são apresentadas as questões, baseadas na norma NBR 13596, para a avaliação da característica da qualidade Usabilidade, a qual mede o esforço necessário para se utilizar o software. Tal questionário deve ser respondido também pelos usuários. As questões são apresentadas na Tabela 11.

Tabela 11 – Questionário para avaliação da característica da qualidade Usabilidade

Subcaracterísticas	Questões	Avaliação
2.1. Inteligibilidade	O sistema apresenta de maneira intuitiva o conceito lógico e sua aplicabilidade?	
2.2. Apreensibilidade	O sistema é fácil para aprender a sua aplicação?	
2.3. Operacionalidade	O sistema é fácil para se operar?	

No último questionário são apresentadas as questões para a avaliação da característica da qualidade Portabilidade, baseadas na norma NBR 13596, o qual mede a capacidade do software de ser transferido de um ambiente para o outro, sendo que ambiente entende-se ser ambiente de hardware, software ou organizacional. Tal questionário deve ser respondido pelos desenvolvedores. As questões são apresentadas na Tabela 12.

Tabela 12 – Questionário para avaliação da característica da qualidade Portabilidade

Subcaracterísticas	Questões	Avaliação
3.1. Adaptabilidade	O sistema apresenta a capacidade de ser adaptado a ambientes diferentes especificados sem a necessidade de aplicação de outras ações?	
3.2. Capacidade para ser instalado	O sistema é fácil para ser instalado em um ambiente especificado?	
3.3. Capacidade para ser substituído	O sistema apresenta dificuldade para ser substituído por outro software?	
3.4. Conformidade	O sistema apresenta padrões ou convenções relacionados a portabilidade?	

Os questionários foram entregues juntamente com o diagrama de casos de uso, que define as funcionalidades das aplicações móveis. Foi preparado o ambiente para que os avaliadores pudessem utilizar o sistema por meio de emuladores e, assim, poder responder os questionários.

Após a aplicação dos questionários, os dados foram tabulados e interpretados. O resultado das avaliações na aplicação móvel desenvolvida na plataforma J2ME está apresentada na Tabela 13, sendo que de acordo com os critérios de aceitação, a aplicação possui aceitação média quanto a funcionalidade, usabilidade e portabilidade.

Analisando o resultado da avaliação efetuada na aplicação móvel utilizando a plataforma J2ME, é possível concluir que:

- a aplicação não está contemplando todas as funcionalidades definidas no diagrama de casos de uso;
- a aplicação não apresenta a capacidade de evitar o acesso não autorizado ao programa e aos dados;
- a plataforma J2ME apresenta um alto nível na capacidade de ser adaptada a ambientes diferentes mas apresenta pouco ou nenhuma dificuldade para ser substituída por outro software do gênero, pois trata-se de uma aplicação simples de ser implementada.

Tabela 13 – Resultado da avaliação efetuada na aplicação móvel em J2ME

	Usuário 1	Usuário 2	Usuário 3	Desenvolvedor	Média
Funcionalidade					1,90
1.1.	2	2	2	1	1,75
1.2.	3	3	3	3	3,00
1.3.	2	2	3	2	2,25
1.4.	2	2	2	2	2,00
1.5.	1	0	1	0	0,50
Usabilidade					1,83
2.1.	2	1	3	2	2,00
2.2.	2	2	2	2	2,00
2.3.	1	2	2	1	1,50
Portabilidade					1,68
3.1.	3	2	3	3	2,75
3.2	2	2	1	3	2,00
3.3.	0	1	1	0	0,50
3.4.	2	1	2	1	1,50

O resultado das avaliações na aplicação móvel desenvolvida na plataforma *Android* está apresentada na Tabela 14, sendo que de acordo com os critérios de aceitação, a aplicação possui aceitação média quanto a funcionalidade e usabilidade e aceitação baixa quanto a portabilidade.

Analisando o resultado da avaliação efetuada na aplicação móvel utilizando a plataforma *Android*, é possível concluir que:

- a aplicação não está contemplando todas as funcionalidades definidas no diagrama de casos de uso;
- a aplicação não apresenta a capacidade de evitar o acesso não autorizado ao programa e os dados;
- a aplicação teve uma ótima aceitação na característica de qualidade de software de usabilidade por apresentar interfaces mais arrojadas e em formato 3D.
- a aplicação apresentou baixa qualidade na subcaracterística de adaptabilidade da característica de portabilidade, especificamente pelo motivo de apenas ser processada na plataforma contendo o sistema operacional Linux para a plataforma *Android*.

Tabela 14 – Resultado da avaliação efetuada na aplicação móvel em *Android*

	Usuário 1	Usuário 2	Usuário 3	Desenvolvedor	Média
Funcionalidade					2,00
1.1.	2	3	2	1	2,00
1.2.	3	2	3	2	2,50
1.3.	2	3	3	3	2,75
1.4.	2	3	2	2	2,25
1.5.	1	0	1	0	0,50
Usabilidade					2,42
2.1.	3	2	3	2	2,50
2.2.	2	3	3	2	2,50
2.3.	3	2	2	2	2,25
Portabilidade					1,31
3.1.	1	0	1	0	0,50
3.2	2	2	2	3	2,25
3.3.	1	1	1	1	1,00
3.4.	2	1	2	1	1,50

As informações coletadas por meio dos questionários em um processo de avaliação realizado por três usuários e um desenvolvedor estão somente na forma de escala de métrica (valor entre 0 e 3), sendo necessário definir os níveis de pontuação através dos resultados obtidos por meio da média aritmética dos avaliadores e posteriormente definir se a aplicação está dentro do nível de critério considerado aceitável.

Na Tabela 15 é apresentado um comparativo entre as duas aplicações onde são fornecidas as informações em que cada característica se enquadra tanto no nível de aceitação quanto ao critério de aceitação.

Com base no comparativo realizado após a avaliação das duas aplicações móveis desenvolvidas em plataformas distintas é possível concluir que:

- a aplicação desenvolvida na plataforma Android apresenta recursos de interface com o usuário mais sofisticada tendo como resultado apresentado uma avaliação superior a da tecnologia J2ME em relação a usabilidade;
- a aplicação desenvolvida na plataforma J2ME oferece a característica de portabilidade superior a da plataforma Android, pois na J2ME é possível processar a aplicação em diversos sistemas operacionais, sendo necessário somente ter instalado

a Máquina Virtual Java, ao contrário da plataforma Android, que tem como pré-requisito o sistema operacional Linux;

- em relação a característica de Funcionalidade as duas aplicações tecnicamente se equilibraram;
- as duas aplicações, segundo o modelo apresentado tiveram como critério de aceitação o nível de aceitação média nas características de Funcionalidade e Usabilidade. Quanto a característica de Portabilidade a aplicação desenvolvida em J2ME teve o nível de aceitação média enquanto a aplicação em Android teve o nível de aceitação baixo.

Tabela 15 – Comparativo entre as avaliações efetuadas nas duas aplicações móveis

Características	J2ME	Android
Funcionalidade	1,90	2,00
- Nível de pontuação	Bom	Bom
- Critério de aceitação	Aceitação média	Aceitação média
Usabilidade	1,83	2,42
- Nível de pontuação	Bom	Bom
- Critério de aceitação	Aceitação média	Aceitação média
Portabilidade	1,68	1,31
- Nível de pontuação	Bom	Regular
- Critério de aceitação	Aceitação média	Aceitação baixa

No capítulo seguinte é apresentada a conclusão do trabalho, ressaltando as suas contribuições e citando as limitações observadas.

CONCLUSÃO

Com o mercado cada dia mais competitivo, as empresas estão buscando soluções inovadoras para obter um diferencial frente aos clientes exigentes. As tecnologias de computação móvel e comunicação sem fio têm sido largamente utilizadas em soluções cada vez mais complexas, e tendo uma grande responsabilidade no gerenciamento de informações das empresas, por isso a qualidade das aplicações móveis desenvolvidas é um fator primordial e deve ser considerada e avaliada durante o desenvolvimento das mesmas.

Este trabalho contribui apresentando as características que compõem uma arquitetura de comunicação sem fio, as principais tecnologias utilizadas para a transmissão de dados por meio de comunicação sem fio, os tipos disponíveis de dispositivos móveis e suas características. Adicionalmente, foram apresentadas duas tecnologias de desenvolvimento de aplicações móveis, sendo uma amplamente utilizada (J2ME) e outra em ascensão (Android), bem como métricas de software no contexto da avaliação da presença de características de qualidade de software através da Norma NBR 13596. Foram discutidas também neste trabalho as implementações de duas aplicações móveis, uma em J2ME e a outra em Android, e as avaliações de tais aplicações aplicando métricas de software.

Durante o desenvolvimento dessa monografia foram encontradas dificuldades no desenvolvimento da aplicação móvel utilizando a tecnologia Android. Tal limitação deve-se ao fato de pouca documentação no contexto do desenvolvimento da aplicação, sendo uma tecnologia nova, mas que tende a convergência da utilização por grandes empresas, como Motorola, LG, Samsung, no desenvolvimento de aplicações em seus dispositivos móveis por tratar-se da primeira plataforma completa, aberta e gratuita para dispositivos móveis, conduzido pela *OHA (Open Handset Alliance)*, grupo formado por mais de 30 empresas de tecnologia e telefonia celular.

Para dar continuidade ao trabalho realizado nesta monografia são sugeridas algumas indicações de trabalho futuro, como a condução de outros estudos de caso de desenvolvimento utilizando sistemas de médio e grande porte, estudo de outras métricas de software definidas na literatura para compor um arcabouço completo de avaliação de aplicações móveis.

REFERÊNCIAS

ALMEIDA, Maurício Barcellos. **Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares.** 2002. Artigo. UFMG - Universidade Federal de Minas Gerais.

BLUETOOTH, Especial Interest Group, 2008. **Specification of the Bluetooth System.** Disponível em <http://www.bluetooth.com/>. Acesso em: 27 março 2008.

CARDELLI, Luca, GORDON, Andrew. **Mobile Ambients.** In: Foundations of Software Science and Computation Structures: First International Conference. Berlin, Germany, 1998.

CARVALHO, Alan. **Tecnologias de Rede para Trainees.** Ed. Book Express, 2001. 128p.

CRUZ, André Luiz Guimarães. **Um framework Java para dispositivos wireless.** 2005. Trabalho de Conclusão de Curso (Bacharelado) – UNIVEM. Marília, São Paulo.

DOCHEV,D., HRISTOV, I.. **Mobile Learning Applications Ubiquitous Characteristics and Technological Solutions**. Bulgarian Academy Of Sciences Cybernetics And Information Technologies, Volume 6, No. 3, Sofia, 2006.

DUNHAM, M. H., HELAL, A. **Mobile computing and databases: Anything New?**. ACM SIGMOD Record. December. 1995.

FIGUEIREDO, E. M. L. **Uma Abordagem Quantitativa para Desenvolvimento de Software Orientado a Aspectos.** 2006. Dissertação (Mestrado) – PUC-RIO, Departamento de Informática. Rio de Janeiro – Rio de Janeiro.

FRIDAY, Adrian, BAKER, Mary, GOYAL, Sachin, ALI, Fahd Al Bin. **Mobile Computing Systems and Applications.** In: 6ºth IEEE WORKSHOP (WMCSA 2004), 2004, English Lake District, UK.

GIALDI, Marcos Vinicius. **Um modelo para Portais Móveis baseado em Middleware Reflexivo e Agentes Móveis.** 2004. Dissertação (Mestrado) – Universidade Estadual de Campinas, Instituto de Computação. Campinas – São Paulo.

GOOGLE. **Official WebSite Android.** Disponível em: http://code.google.com/android/. Acesso em: 01 julho 2008.

GUPTA, Monish, IMIELINSKI, Tomasz, PEYYETI, Sarma. **Energy Efficient Data Filtering and Communication in Mobile Wireless Computing.** In: Second USENIX Symposium on Mobile and Location. 1995

IDGNOW. Now!Digital Business Ltda. **T-Mobile lança G1, primeiro celular do mercado com o Android.** 2008. Disponível em <http://idgnow.uol.com.br/telecom/2008/09/23/t-mobile-lanca-g1-primeiro-celular-do-mercado-com-o-android/. Acesso em: 07 outubro 2008.

JCP, Java Community Process. **JSR-000046 Foundation Profile 1.0b.** Disponível em <http://jcp.org/aboutJava/communityprocess/mrel/jsr046/>. Acesso em: 15 maio 2008.

LOUREIRO, Antonio A. F., SADOK, Djamel F. H., MATEUS, Geraldo R., NOGUEIRA, José Marcos S., KELNER, Judith. **Comunicação sem fio e Computação Móvel: Tecnologias, Desafios e Oportunidades**. Congresso da Sociedade Brasileira de Computação. 2003.

MESQUITA, Paulo Ricardo Batista, BRANCO, Luiz Henrique Castelo, BRANCO, Kalinka Regina Lucas Jaquie Castelo. **Conceitos básicos para desenvolvimento de aplicações de Computação Móvel.** VI Escola Regional de Informática - ERI São Paulo/Oeste, Marília, 2007.

MORRISON, Michael. **Wireless Java with J2ME in 21 Days.** 2001. Sams Teach Yourself. Indianapolis, Indiana – USA.

MUCHOW, John W. **Core J2ME – Tecnologia & MIDP.** 2004. Tradutor: João Eduardo Nóbrega Tortello. São Paulo: Pearson Makron Booksm.

NBR 13596. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Tecnologia de informação – Avaliação de produto de software: Características de qualidade e diretrizes para o seu uso.** Rio de Janeiro, 1996.

NEVES, Junia Maria Martins. **Estudo de Usabilidade em Sistemas Móveis com foco em PDAs.** 2005. Dissertação de Mestrado (Mestrado em Ciência da Computação) – Universidade Estadual de Campinas, Instituto de Computação. Campinas, 2005.

OLIVEIRA, Jeísa P, KAMIENSKI, Carlos A., KELNER, Judith, SADOK, Djamel F. H. **Análise de Desempenho de TCP sobre GPRS em um Ambiente Fim a Fim**. 2004. Centro de Informática – Universidade Federal de Pernambuco.

PERING, Trevor, AGARWAL, Yuvraj, GUPTA, Rajesh, WANT, Roy. **CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces.** In: MobiSys2006 The Fourth International Conference on Mobile Systems, Applications and Services, 2006, Suiça.

ROCHA. Tarcisio da. **Um Sistema de Transações Adaptável para o Ambiente de Comunicação Sem Fio.** 2003. Dissertação de Mestrado (Mestrado em Ciência da Computação) - Universidade Estadual de Campinas, Instituto de Computação. Campinas, 2003.

RODRIGUES, N. G. **Implementação de uma Rede WiFi.** 2004. Congresso Brasileiro de Ciência Computação. Itajaí – Santa Catarina.

SCHMITZ, Richard Salvalaggio. **Monitoramento Remoto de Informações de Pacientes Via Protocolo WAP**. 2003. Trabalho de Conclusão de Curso (Mestrado) – UFSC Universidade Federal de Santa Catarina. Florianópolis, Santa Catarina.

SOMMERVILLE, Ian. **Engenharia de Software.** Tradução: Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edílson de Andrade Barbosa. 8ª Edição. São Paulo: Pearson Addison-Wesley, 2007.

SUN. **Java Technology**. Disponível em:< http://www.sun.com/java/>. Acesso em: 02 outubro 2008.

TEMPLE, André, MELLO, Rodrigo Fernandes de, CALEGARI, Danival Taffarel, SCHIEZARO, Maurício. **Programação Web com JSP, Servlets e J2EE.** ISBN: 85-905209-1-9. 2004.