PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Programa de Pós-Graduação em Informática

FERRAMENTA PARA DESENVOLVIMENTO DE TUTORIAIS VOLTADOS PARA O *M-LEARNING*

Christien Lana Rachid

Belo Horizonte

2011

Christien Lana Rachid

FERRAMENTA PARA DESENVOLVIMENTO DE TUTORIAIS VOLTADOS PARA O *M-LEARNING*

Dissertação apresentada ao Programa de Pós-Graduação em Informática como requisito parcial para qualificação ao Grau de Mestre em Informática pela Pontifícia Universidade Católica de Minas Gerais.

Orientadora: Lucila Ishitani

Belo Horizonte

2011

FICHA CATALOGRÁFICA

Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Rachid, Christien Lana
R119f Ferramenta para desenvolvimento de tutoriais voltados para o m-learning / Christien Lana Rachid. – Belo Horizonte, 2011.
99f. : il.

Orientador: Lucila Ishitani
Dissertação (Mestrado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-graduação em Informática.
Bibliografia.

1. Ensino a distância – Teses.2.Ensino via Web.3.Computação móvel.4.Sistemas de comunicação móvelI.Ishitani, Lucila
II. Pontifícia Universidade Católica de Minas Gerais. IV. Título.

CDU: 681.3.03.06:37

Bibliotecário: Fernando A. Dias – CRB6/1084

PUC Minas
Programa de Pós-graduação em Informática

FOLHA DE APROVAÇÃO

Ferramenta para desenvolvimento de tutoriais voltados para o M-Learning

CHRISTIEN LANA RACHID

Dissertação defendida e aprovada pela seguinte banca examinadora:

Profª. Lucila Ishitani - Orientadora (PUC Minas)
Doutora em Ciência da Computação - UFMG

Prof. Zenilton Kleber Gonçalves do Patrocínio Júnior (PUC Minas)
Doutor em Ciências da Computação - UFMG

Profª. Maria Augusta Vieira Nelson (PUC Minas)
Doutora em Ciência da Computação pela University of Waterloo, U.W., Canadá

Belo Horizonte, 17 de novembro de 2010.

Anel Rodoviário – KM 23,5, Rua Walter Ianni, 255 – São Gabriel / Belo Horizonte – MG / CEP: 31980-110
Fone: (31) 3439-5204 – Fax: (31) 3439-5204 – e-mail: mestradoinfsec@pucminas.br

Dedico este trabalho a alguém que dispensa elogios e superlativos, minha esposa, Ana Amelia. Nada valeria o esforço, se não tivéssemos a quem dedicar as nossas conquistas.

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, a Deus pela minha vida, por todos os bons momentos que Ele me deu o prazer de desfrutar e pela paciência e força que me foram dadas nos inúmeros momentos difíceis.

Agradeço também, à minha família:

- Minha esposa, por ser companheira em todas as horas, também pela paciência, compreensão e cumplicidade;
- Minha mãe, exemplo de dedicação e esforço;
- Meu pai, pelo incentivo;
- Minhas irmãs, Renata e Thais, pela solidariedade;
- Meus sogros, pela ajuda dada nas horas mais difíceis;
- Minha Avó, pelo carinho e pelos almoços descontraídos em sua casa.

As pessoas que contribuiram para a realização deste trabalho:

- Doutora Lucila Ishitani, pela atenção e dedicação na orientação deste trabalho;
- Doutor Zenilton Kleber Gonçalves do Patrocínio Júnior, pelas considerações e sugestões.

RESUMO

Modernas tecnologias e padrões de telecomunicação para a computação móvel no uso do *m-learning* estão surgindo. Hoje, praticamente, qualquer modelo de celular é capaz de lidar com conteúdo multimídia. Em adaptação à essa nova realidade, foi proposta a ferramenta m-tutorial, que auxilia professores na elaboração de materiais educacionais interativos, mais especificamente objetos de aprendizagem voltados para o *m-learning*. Destina-se à construção de tutoriais em pouco tempo e com o mínimo de esforço, utilizando a infraestrutura e a conectividade existentes para dispositivos móveis. Demonstrando a aplicabilidade do m-tutorial, é apresentado um tutorial desenvolvido com os recursos disponíveis na ferramenta. Aspectos de usabilidade são considerados. Uma lista de requisitos de usabilidade foi avaliada por meio das heurísticas de Nielsen, tornando a interface do m-tutorial simples e intuitiva, podendo ser usada por diferentes tipos de usuários. Para avaliar as características e especificações da ferramenta foi utilizado o método de inspeção de software conhecido por *checklist*. Das plataformas avaliadas, foi definido utilizar Java ME por sua portabilidade, por ser de fácil aprendizado, por ser gratuito e por ser projetado para dispositivos com memória, vídeo e poder de processamento limitado. *Frameworks* e bibliotecas para Java ME foram avaliados. Entre os existentes comparados, foram utilizadas as classes da biblioteca Lwuit em conjunto com a tecnologia *Web Service*. Dentre as opções de aplicações móveis o m-tutorial se destaca porque oferece facilidades na criação de objetos de aprendizagem do tipo tutorial para *m-learning* em modo *off-line*. Seu conteúdo é armazenado internamente em uma classe RMS e enviado para um servidor *web* quando possuir uma conexão estável com a Internet. O m-tutorial é baseado em interfaces e classes abstratas. Regras de permissão e de acesso atendem a recursos protegidos do m-tutorial. Os tutoriais são elaborados podendo conter textos, imagens ou vídeos. Muitos dos *softwares* para *desktop* estão sendo adaptados para execução e utilização em dispositivos móveis. Objetos de Aprendizagem também vêm ganhando espaço neste cenário. As características do m-learning favorecem o desenvolvimento e o uso de Objetos de Aprendizagem a qualquer hora, lugar e em diversos dispositivos móveis.

Palavras-chave: *Framework*, bibliotecas, *m-learning*, objetos de aprendizagem.

ABSTRACT

Modern technology and telecommunications standards are developed for mobile computing and m-learning. Today, almost all mobile phone models support multimedia content. Considering this situation, we proposed a tool - m-tutorial - that assists teachers in preparing interactive educational materials, more specifically, learning objects for m-learning. It aims to build tutorials in a short time and with minimal effort using the existing infrastructure and connectivity for mobile devices. For demonstrating the applicability of the m-tutorial it is presented a tutorial developed with available resources in the tool. Usability aspects are considered. A list of usability items was evaluated by the heuristics of Nielsen, making the interface of m-tutorial simple and intuitive, which allows its use by different types of users. To evaluate the characteristics and specification of the tool it was used the method known as software inspection checklist. Among the evaluated platforms, Java ME was chosen for its portability, and for being easy to learn, free and designed for limited memory and video processing devices. Frameworks and libraries for Java ME were evaluated. Among them, we chose to use the LWUIT library classes in conjunction with the Web Service technology. Among the options for mobile applications, m-tutorial stands out because it provides facilities for creating m-learning objects of the type tutorial, in offline mode. Its content is stored internally in a class RMS and sent to a web server when there is a stable connection to the Internet. m-tutorial is based on interfaces and abstract classes. The tutorials are prepared and may contain text, images or videos. Many of the desktop softwares are being adapted for application and use on mobile devices. Learning Objects also have been developed for this scenario. The characteristics of m-learning promote the development and the use of Learning Objects anytime, anywhere and on various mobile devices.

Keywords: Framework, libraries, m-learning, learning objects

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE SIGLAS E ABREVIATURAS

ADL *Advanced Distributed Learning*

ANATEL Agência Nacional de Telecomunicações

API *Application Programming Interface*

AVAM Ambiente Virtual de Aprendizagem com Mobilidade

AWT *Abstract Window Toolkit*

BREW *Binary Runtime Environment for Wireless*

CDC *Connected Device Configuration*

CESTA Coletânea de Entidades de Suporte ao uso de Tecnologia na Aprendizagem

CETIC Centro de Estudos sobre as Tecnologias da Informação e da Comunicação

CLDC *Connected Limited Device Configuration*

CM *Configuration Management*

CMM *Capability Maturity Model*

CRM *Customer Relationship Management*

CSS *Cascading Style Sheets*

EAD Educação a Distância

FTP *File Transfer Protocol*

GUI *Graphic User Interface*

GPL *General public License*

GPLV *General public License Version*

GPRS *General Packet Radio System*

GPS *Global Positioning System*

GSM *Global System for Mobile communication*

HTML *HyperText Markup Language*

HTTP *HyperText Transfer Protocol*

IDE *Integrated Development Environment*

IEEE *Institute of Electrical and Electronics Engineers*

IMS *Instructional Management Systems Project*

IR *Infra Red*

JAVA SE *Java Standard Edition*

JAVA EE *Java Entreprise Edition*

JAVA ME *Java Micro Edition*

JCP *Java Community Process*

JISC *Joint Information Systems Committee*

JSR *Java Specification Requests*

JVM *Java Virtual Machine*

LCDUI *LCD User Interface*

LTSC *Learning Technology Standards Committee*

LWUIT *Lightweight UI Toolkit*

MIDI *Musical Instrument Digital Interface*

MIDP *Mobile Information Device Profile*

MLO *Mobile Learning Object*

MMS *Multimedia Message Service*

MPEG *Moving Picture Experts Group*

MVC *Model-View-Controler*

MWT *Micro Window Toolkit*

OAs Objetos de Aprendizagem

OHA *Open Handset Allince*

PDA *Personal Digital Assistants*

RDF *Resource Description Framework*

RIVED Rede Interativa Virtual de Educação

RIA *Rich Internet Applications*

RFID *Radio Frequency Identification*

RMS *Record Managment System*

RUP *Rational Unified Process*

SGBD Sistema de Gerenciador de Banco de Dados

SGML *Standard Generalized Markup Language*

SMIL *Synchronized Multimedia Integration Language*

SMS *Short Message Service*

SQL *Structured Query Language*

SOA *Service-oriented Architecture*

SOAP *Simple Object Access Protocol*

SVG *Scalable Vector Graphics*

SWT *Standard Widget Toolkit*

TDMA *Time Division Multiple Access*

UDDI *Description, Discovery, and Integration*

UFRGS Universidade Federal do Rio Grande do Sul

UML *Unified Modeling Language*

USP Universidade de São Paulo

VIRTRAM Virtual Training for Mobile Devices

WAP *Wireless Aplication Protocol*

WSDL *Web Service Description Language*

W-PAN *Wireless Personal Area Network*

W-LAN *Wireless Local Area Network*

W-WAN *Wireless Wide Area Network*

WI-FI *Wireless Fidelity*

W3C *World Wide Web Consortium*

XHTML *Extensible HyperText Markup Language*

XML *Extended Mashup Language*

ZCL *Zona de Comercio Livre*

SUMÁRIO

1 INTRODUÇÃO

O Brasil tem vivenciado um crescimento considerável na oferta de cursos por meio da Educação a Distância (EAD). Esses cursos favorecem pessoas que têm dificuldades para frequentar cursos tradicionais que exigem presença em horário e locais determinados para sua realização (PELISSOLI; LOYOLLA, 2004). O aluno faz o treinamento quando pode, não há necessidade de estar no mesmo ambiente que o professor e os demais alunos, não é preciso sacrificar horário de trabalho. É possível rever ou refazer os exercícios quantas vezes quiser.

Dentre as opções de oferta de EAD, há que se considerar o fato de que milhões de pessoas em todo o mundo já se comunicam por meio de tecnologias móveis. Dados da Agência Nacional de Telecomunicações (Anatel)[1] indicam que o Brasil terminou o mês de outubro de 2010 com 189,5 milhões de celulares e uma densidade de 90,6 celular/100 hab. O telefone celular continua sua tendência de crescimento e já caminha para a universalização nos domicílios brasileiros, chegando a 82% dos lares em áreas urbanas e 78% no total do país (CETIC, 2009).

Tecnologias móveis permitem interações sociais relevantes, sensíveis ao contexto e possibilitam conectividade com a Internet (NAISMITH et al., 2004).

Um novo conceito gerado neste contexto é o *m-learning*. A estratégia de *m-learning* é *e-learning* utilizando dispositivos móveis, que são equipamentos como *smartphones*, *palmtops* e telefones celulares (SANTOS et al., 2005). As seguintes limitações caracterizam esses dispositivos: poder de processamento reduzido, tela de tamanho variável (às vezes monocromática), energia limitada (dependente de baterias) e comunicação com taxas de transmissão, geralmente, menores do que as das redes fixas (SVETLANA; YONGLK-YOON, 2009).

Basicamente, o *m-learning* faz uso das tecnologias de redes sem fio, *Bluetooth*, *Wireless Aplication Protocol* (WAP), *General Packet Radio System* (GPRS), *Infra Red* (IR) de recursos fornecidos pela telefonia celular, das linguagens *EXtensible Markup Language* (XML), Java, dos serviços de correio de voz, dos serviços de mensagens curtas (SMS), da capacidade de transmissão de fotos, dos serviços de *e-mail*, *multimedia message service* (MMS) e vídeo sob demanda (KEEGAN, 2002).

Segundo Franciscato e Medina (2009), muitos dos *softwares* para *desktop* estão sendo adaptados para execução e visualização em dispositivos móveis. Objetos de aprendizagem também vêm ganhando espaço neste cenário.

Denomina-se Objeto de Aprendizagem (*Learning Object*) “qualquer entidade, digital

[1] http://www.anatel.gov.br

ou não, que possa ser usada, reutilizada ou referenciada durante a aprendizagem com apoio de tecnologia" (LTSC, 2002). Material multimídia (texto, imagens, som, e vídeo), exercícios, *softwares* educativos, uma apostila são exemplos de objetos de aprendizagem. Entretanto, se há professores que sabem desenvolver materiais digitais educativos, nem todos têm tempo ou conhecimento específico para desenvolvê-los.

Atualmente, existe um tipo de objeto de aprendizagem conhecido como *Mobile Learning Object* (MLO), descrito como entidade independente e personalizável, que permite atividades de aprendizagem em um determinado contexto educacional por meio da tecnologia móvel (FLORES; MORTEO, 2008). Pode ser armazenado no celular e utilizado sem qualquer conexão com a *Internet*.

A partir destas condições torna-se necessário oferecer recursos para facilitar o desenvolvimento de OAs para *m-learning*. Propõe-se, assim, criar objetos de aprendizagem para *m-learning* de forma rápida, que ofereçam condições para que professores desenvolvam OAs em qualquer lugar e a qualquer momento, sem depender da conectividade do dispositivo.

Diversas atividades estão presentes nesta pesquisa, pois a ferramenta foi desenvolvido sobre a plataforma móvel, suas classes foram especificadas, modeladas e avaliadas por técnicas de engenharia de *software*.

Após o estudo inicial das tecnologias envolvidas no desenvolvimento do m-tutorial, foi decidido utilizar a metodologia do *framework Lwuit* para a criação de objetos de aprendizagem para *m-learning*. Esses objetos são criados em modo *off-line* e podem ser enviados para web utilizando à tecnologia *Web Service*.

O m-tutorial foi elaborado seguindo as seis melhores práticas de desenvolvimento de *software*, segundo RUP. Sobre o ciclo de vida iterativo e incremental, a ferramenta foi especificada, suas classes estão modeladas de forma que qualquer desenvolvedor consiga reutilizá-las em outras aplicações. A sua arquitetura se baseia no padrão MVC e seus componentes estão projetados para aplicações distribuídas.

Para validação da ferramenta proposta foi desenvolvido um tutorial como estudo de caso. Segundo Holanda (2009), tutorial pode ser entendido como "um conjunto de instruções que ensinam como fazer, como proceder, etc, utilizado em autoaprendizagem".

A avaliação da ferramenta se realizou por meio das heurísticas de Nielsen, lista de problemas de usabilidade da interface, baseada em práticas definidas por profissionais experientes e especialistas em Interface. É um método que deve ser realizado por avaliadores especialistas e não por usuários comuns (NIELSEN, 1994). Os resultados podem contribuir para o levantamento de aspectos positivos e negativos de *design*.

Ainda serão utilizados métodos de análise estática para verificação e análise, que asse-

gurem que o *software* cumpra com as suas especificações e atenda às necessidades dos usuários. Será necessário considerar variáveis como esforço, produtividade, tempo e custo de desenvolvimento, garantindo a redução do retrabalho e da qualidade do *software*.

Com esse intuito foi proposto a utilização de um *checklist* de inspeção do *software* para avaliar as características e especificações da arquitetura.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho consiste na especificação de uma ferramenta para criação de Objeto de Aprendizagem do tipo tutorial para *m-learning* usando dispositivos móveis como ambiente de desenvolvimento.

1.1.2 Objetivos Específicos

Os objetivos específicos definidos para este trabalho foram:

- Fazer o levantamento do estado da arte na área;
- Fazer uma análise comparativa entre APIs, bibliotecas e *frameworks* para *m-learning*, considerando suas diversas características;
- Especificar características desejáveis para a ferramenta a ser desenvolvida;
- Construir um modelo de classes básicas para desenvolvimento de tutoriais voltados para *m-learning*;
- Avaliar, investigar e selecionar ferramentas, ambientes e linguagens de programação mais adequadas para desenvolvimento da ferramenta;
- Desenvolver uma aplicação para validação da ferramenta no desenvolvimento de tutoriais para dispositivos móveis;
- Avaliar o processo de construção de Objetos de Aprendizagem utilizando um *checklist* de inspeção, verificando se o produto realiza as funções para o qual foi desenvolvido;
- Realizar avaliação da ferramenta entre profissionais experientes e especialistas em interface, utilizando uma lista de princípios (heurísticas) de usabilidade.

1.2 Trabalhos Relacionados

Através de pesquisas realizadas nas bibliotecas digitais, constatou-se que mesmo sendo recente, foram encontrados vários artigos técnico-científicos que apresentam soluções voltadas para o *m-learning*. Alguns dos principais trabalhos sobre o assunto encontram-se aqui organizados.

Projetos usando dispositivos móveis na Educação começaram a ser utilizados pelas pessoas. Segundo Houser, Thornton e Kluge (2002) projetos educacionais são criados inicialmente nas universidades, escolas e programas de treinamento corporativo de ensino a distância. Esses projetos utilizam celulares como parte de um programa de educação que combina Web e dispositivos móveis. Eles mostram que a combinação única de recursos em dispositivos torna-os valiosas ferramentas educacionais, pela facilidade de locomoção e por pessoas carregarem constantemente seus aparelhos celulares. Isso se tornou possível, devido ao surgimento de novas tecnologias que proporcionam uma fácil integração entre as duas plataformas, web e móvel.

No período de 2000 a 2007 Torstein Rekkedal e Aleksander Dye baseados em três projetos europeus "*From e-Learning to m-Learning","Mobile learning - the next generation of learning"*, e *"Incorporating mobile learning into mainstream education*", relatam os benefícios com a utilização de tecnologias móveis na educação a distância.

O primeiro projeto consiste em uma aplicação de aprendizagem móvel para telefones celulares e *Personal Digital Assistants* (PDAs). Os materiais de aprendizagem são desenvolvidos, principalmente, para serem baixados e utilizados em modo *Off-line*. Já o segundo projeto trata de um ambiente multimídia que sempre funciona em modo *on-line* e por isso é destinado a alunos com acesso a redes sem fio. O terceiro projeto consiste em desenvolver conteúdo de cursos para aprendizagem móvel do ensino médio ou nível superior. Um dos principais desafios observados nestes projetos foi relativo ao uso dos dispositivos móveis. Soluções tiveram que ser adaptadas para o tamanho de tela e espaços pequenos. Outro problema encontrado foi a taxa de transferência de dados limitada e o poder de processamento reduzido em dispositivos móveis. Desses projetos foram retiradas informações de como produzir materiais educacionais de qualidade, adaptados a aparelhos com dimensões reduzidas que funcionem tanto em modo *off-line* como no *on-line*.

Pesquisadores dos Estados Unidos (SOLOWAY et al., 2001) desenvolveram vários programas educacionais para PDAs, utilizados no ensino fundamental que permitem que os educadores tenham a liberdade de experimentar o *m-learning*. Esses programas incluem o jogo de perguntas e respostas *Bubble Blasters*[2], simulação de ciência *Cooties*[3] e o editor de mapa

[2]http://download.cnet.com/Bubble-Blasters/3000-18495_4-16840.html
[3]http://www.goknow.com/Products/Cooties/

conceitual *PiCoMap*[4].

Existem outras propostas interessantes sobre *m-learning*. Santos et al. (2005) propuseram um *framework* para a construção de aplicações educacionais para dispositivos móveis, com recursos de realidade virtual, chamado *Virtual Training for Mobile Devices* (VirTraM), que foi utilizado na criação de um museu virtual. O protótipo desenvolvido demonstra interatividade, usabilidade e portabilidade, além de permitir uma experimentação da integração das tecnologias de realidade virtual e computação móvel. Este *framework* falha por não possuir uma interação com a web, e não utilizar recursos tecnológicos como *Global Positioning System* (GPS), Radio *Frequency Identification* (RFID) e *wireless*, tecnologias necessárias para permitir a localização automática no passeio ao museu.

Outro *framework* é o Ambiente Virtual de Aprendizagem com Mobilidade (AVAM), utilizado para o *m-learning*, desenvolvido pela Universidade Federal do Rio Grande do Sul (UFRGS), destinado a organizar os diferentes componentes da aprendizagem com mobilidade, com o propósito de oferecer objetos de aprendizagem significativos e compatíveis com os estilos de aprendizagem preferenciais dos estudantes.

Há duas ferramentas de criação de OAs para *m-learning* disponíveis na web:*Mobile Author Learning*[5] e *MobileSiteGalore*[6]. O *Mobile Author* permite projetar, testar e publicar conteúdo móvel para a utilização em telefones celulares e PDAs. Esta ferramenta executa OAs feitas para qualquer aparelho celular a partir do *desktop*. Já o *MobiSiteGalore* é o pioneiro entre as ferramentas construtores de locais móveis. Com ele, pode-se desenvolver páginas *web* para dispositivos móveis com facilidade. O interessante dessa ferramenta é que ela funciona em qualquer telefone móvel, gratuitamente, sem anúncios. Não requer conhecimentos técnicos e nem *software* para baixar ou instalar, apenas um navegador da web no celular. A deficiência dessa ferramenta está em não permitir a criação de objetos de aprendizagem estando em modo *off-line*.

Fuks et al. (2005) e Fillipo et al. (2006) relatam a adaptação do *software* para *e-learning* AulaNet para dispositivos móveis, chamado AulaNetM, que começou a ser desenvolvido em 2004. “A versão móvel do AulaNet foi adaptada para utilização de fóruns de discussão assíncrona utilizando PDAs". O interessante desse trabalho é como a adaptação para versão móvel encontrou grandes dificuldades impostas pela mobilidade.

Baseados em projetos de sala de aula, Hadzilacos e Tryfona (2005) apresentaram a construção de uma representação multimídia de um sítio arqueológico para aula de história conhecido por Acrópolis. O objetivo do aprendizado consiste em compreender o período histórico por meio da visualização dos edifícios e leitura dos artefatos, obtendo informações dos museus

[4]http://www.goknow.com/Products/PiCoMap/
[5]http://www.abjade.com
[6]http://www.mobisitegalore.com

e vista panorâmica dos locais, fazendo uma descrição verbal dos monumentos arquitetônicos. Os alunos poderão visitar o sítio arqueológico e o museu várias vezes. Trabalhando em dupla podem solicitar orientações a seu professor a qualquer hora. Caso o aluno não reconheça um determinado objeto, pode fazer uma pesquisa na base de dados de seu dispositivo. O interessante desse projeto foi a possibilidade de utilização de aprendizagem coletiva entre professores e alunos.

1.3 Organização da Dissertação

Este texto está organizado da seguinte forma: o Capítulo 2 apresenta conceitos, classificação e categorias sobre *m-learning*, além de mostrar informações importantes sobre objetos de aprendizagem, metadados e objetos de aprendizagem móveis. O Capítulo 3 apresenta as tecnologias utilizadas no desenvolvimento como, tecnologias sem fio, ambientes de desenvolvimento integrado, serviços web, sistemas operacionais e plataformas de desenvolvimento. Um comparativo entre as principais aplicações móveis são avaliados. O Capítulo 4 discute conceitos básicos sobre bibliotecas e *frameworks* existentes, além de mostrar as especificações de algumas APIs. No Capítulo 5 discutiremos usabilidade segundo a abordagem tradicional e do ponto de vista de aplicações para dispositivos móveis, com foco em Internet móvel. Conceitos e métodos de avaliação são apresentados. O Processo de desenvovlvimento de software será discutida no Capítulo 6, metodologia e conceitos são apresentados. No Capítulo 7 será discutida a especificação da ferramenta. São apresentados os diagramas UML do m-tutorial. As principais classes e métodos são demonstrados. Ja no Capítulo 8 o conjunto de testes e resultados obtidos serão descritos, são definidos e analisados por avaliações. As contribuições são apresentadas. Finalmente, no Capítulo 9 o trabalho é encerrado por meio de um resumo dos resultados e sugestões para trabalhos futuros.

2 M-LEARNING

Neste capítulo será abordada a aplicabilidade da computação móvel na aprendizagem. A Seção 2.1 apresenta o conceito de *Mobile Learning*, mostrando uma visão geral sobre computação móvel e apresentando os principais paradigmas educacionais. A Seção 2.2 mostra as características dos equipamentos. A Seção 2.3 apresenta características da aprendizagem móvel. Por fim, a Seção 2.4 apresenta conceitos sobre objetos de Aprendizagem e subseção sobre objetos de aprendizagem móveis.

2.1 Conceitos

Como acontece com muitos paradigmas emergentes, há diversas tentativas para definir *m-learning*. De acordo com Kinshuk et al. (2003),"*m-learning* é definida como a capacidade do uso de dispositivos portáteis para acesso a recursos de aprendizagem". Nah, White e Sussex (2008) consideram que "*m-learning* refere-se ao uso de celular e computador de mão, como PDAs, telefones celulares, *laptops* e *tablet* PCs no ensino e aprendizagem". Cabe destacar que a aprendizagem móvel é automaticamente entendida como *mobile e-learning* e sua história e desenvolvimento tem que ser entendida como uma continuação da aprendizagem convencional.

Segundo o portal da conferência MOLENET (2010), a aprendizagem móvel pode ser amplamente definida, como "a exploração das onipresentes tecnologias de mão, juntamente com as redes de telefonia sem fio e móvel, para facilitar, apoiar, melhorar e ampliar o alcance do ensino e da aprendizagem".

O *m-learning* pode trazer resultados significativos para as instituições de ensino, tais como: baixos custos, flexibilidade de acesso virtual aos materiais de informação, aprendizagem coletiva virtual, entre outros. Os estudos realizados sobre *m-Learning* ainda são muito recentes, acredita-se que em breve teremos a implantação e utilização desta solução nas instituições de ensino públicas ou privadas, assim como, nas grandes corporações, empresas de médio e grande porte na questão do treinamento de sua equipe (MARTINS, 2006).

2.2 Categorias da Aprendizagem Móvel

Segundo Kukulska-Hulme e Traxler (2005) e Traxler e Leach (2006) algumas categorias interessantes de aprendizagem móvel são:

- Aprendizagem móvel guiada pela tecnologia ou por uma tecnologia específica que demonstra viabilidade técnica e possibilidade pedagógica;
- Aprendizagem portátil em escala reduzida - adaptação de soluções da educação a distância convencional para dispositivos portáteis com conexões sem fio;
- Aprendizagem em sala de aula - tecnologias sem fio e móveis são utilizadas como apoio à aprendizagem colaborativa, ligada a outras tecnologias em sala de aula;
- Treinamento móvel e desempenho de apoio - as tecnologias são utilizadas para melhorar a produtividade, oferecendo informações e apoio na hora certa e no contexto de suas prioridades imediatas, funções e deveres;
- Implementação em grande escala - São implantadas tecnologias de comunicações móveis a nível institucional ou departamental para apresentar questões organizacionais, por exemplo, *MELaS*[1] financiadas pelo *Joint Information Systems Committee* (JISC) no Reino Unido;
- Inclusão e diversidade assistida - variadas tecnologias móveis e sem fio estão sendo utilizadas para aumentar a participação dos alunos na educação;
- Ensino a distância, desenvolvimento rural e móvel - tecnologias utilizadas onde obstáculos ambientais e infraestrutura não permite a utilização das tecnologias da educação a distância “convencional".

Naismith et al. (2004) sugerem que as tecnologias móveis podem relacionar-se a seis tipos de aprendizagem ou “categorias de atividade", ou seja, comportamentalista, construtivista, situada, colaborativa, informais e de apoio à coordenação. A aprendizagem móvel pode manifestar-se nas seguintes formas:

- Para atividade do tipo comportamentalista, é o retorno rápido ou elemento de reforço, facilitada por dispositivos móveis;
- Já a atividade construtivista propõe novas idéias ou conceitos tomando como base conhecimentos prévios e atuais. É um conjunto de diretrizes que auxiliam na criação de ambientes educacionais colaborativos capazes de apoiar experiências de aprendizagem atraentes e reflexivas.
- Para atividade situada, os alunos podem ter um dispositivo móvel para fora de um contexto autêntico ou usá-lo movendo-se num contexto de ambiente de conhecimento em um local especialmente equipado, como em um museu;

[1] www.wlv.ac.uk/celt/MELaS

- Para a aprendizagem colaborativa, dispositivos móveis fornecem um meio prático adicional de comunicação e um meio portátil de partilha de informação eletrônica;
- Para a aprendizagem informal, dispositivos móveis acompanham os usuários em suas experiências cotidianas e se tornam uma fonte conveniente de informações;
- O apoio à coordenação de recursos ou de aprendizagem pode ser melhorado por meio da disponibilidade de tecnologias móveis em todos os momentos, como diversos controles.

2.3 Características da Aprendizagem Móvel

Cada usuário emprega seu dispositivo móvel de uma forma diferente. Por exemplo, adolescentes, frequentemente, usam SMS para se comunicar, enquanto profissionais estão mais propensos a utilizar o *e-mail*. Existem diferentes perfis de usuários e papéis no uso de tecnologias móveis (METCALF; MARCO, 2006).

Uma das características de aprendizagem móvel é que ela utiliza diversos dispositivos móveis (KEEGAN, 2005):

- Usados em todos os lugares;
- Considerados de uso pessoal;
- Que sejam baratos e fáceis de usar;
- Utilizados em uma variedade de configurações diferentes.

2.4 Objetos de Aprendizagem

Conforme definido por Tarouco, Fabre e Tamusiunas (2003), objetos de aprendizagem são materiais educacionais com objetivos pedagógicos que servem para apoiar o processo de ensino-aprendizagem. Os seguintes recursos digitais são exemplos de Objetos de Aprendizagem (ANIDO et al., 2002):

- Imagens digitais, vídeos e animações;
- Textos;
- Páginas Web;
- Aplicações educacionais.

Estes objetos de aprendizagem podem ser armazenados em repositórios existentes na internet, como RIVED, OE3, LAB VIRT, CESTA, entre outros. Repositórios são definidos como "coleções de recursos de aprendizagem armazenados em bases de dados ou sistema de arquivos e possuem metadados associados, geralmente disponíveis e pesquisáveis via Web" (MUSA; SILVA; OLIVEIRA, 2004).

Metadados classificam e caracterizam um OA, são utilizados na pesquisa e na identificação do uso de OAs. Metadados para cada OA devem incluir (LTSC, 2002):

- Descrição geral (nome, autores, versão, associações com outros OAs, etc);
- Dados técnicos (tamanho, formato, descrição de instalação, duração, etc);
- Dados educacionais (tipo/grau de interatividade, recurso de aprendizagem, objetivo educacional, etc); e
- Direitos autorais (se tem custo, se há restrições de uso, etc).

2.4.1 Objetos de Aprendizagem Móveis

Para Castillo e Ayala (2007), um objeto de aprendizagem móvel (MLO) é uma entidade de informação digital, interativo, adaptável e reusável nos diferentes contextos, projetado para ser usado em um ambiente de aprendizagem móvel, capaz de suportar aproximações e perspectivas de aprendizagem em diferentes interações. Eles acreditam que objetos de aprendizagem móveis não devem ser apenas uma extensão simples de objetos de aprendizagem, mas uma adaptação às limitações do formato dos dispositivos móveis.

Objetos de aprendizagem móveis devem ser construídos de forma colaborativa, promovido por atividades práticas em um espaço de trabalho compartilhável com anotações ou comentários dos membros da comunidade de aprendizagem (JIUXIN; BO; JUNZHOU, 2008).

Para Brown, Collins e Duguid (1989) um objeto de aprendizagem móvel deve considerar duas perspectivas de interação, de acordo com a aproximação de aprendizagem:

- Permitir que o usuário interaja com a aprendizagem em qualquer lugar e a qualquer hora;
- Simular situação de aprendizagem a fim de permitir que o aluno participe de uma atividade real a qualquer lugar e em qualquer lugar.

3 TECNOLOGIAS DE DESENVOLVIMENTO

Neste capítulo são apresentadas as tecnologias utilizadas no desenvolvimento de aplicações móveis. A Seção 3.1 apresenta as tecnologias sem fio. Na Seção 3.2 são abordados os ambientes de desenvolvimento integrado. A Seção 3.3 apresenta os diversos serviços web. A Seção 3.4 apresenta os sistemas operacionais para dispositivos móveis. A Seção 3.5 apresenta as plataformas para dispositivos móveis e na Seção 3.6, uma comparação entre os sistemas operacionais e plataformas de desenvolvimento para dispositivos móveis.

3.1 Tecnologias Sem Fio

O uso da tecnologia sem fio vem crescendo e com ela as interligações de dispositivos, tanto fixos quanto móveis. Com o crescimento acelerado da utilização dos dispositivos portáteis que necessitam utilizar uma infraestrutura de rede, podendo citar *notebooks* e *handhelds*, surgem as redes sem fios. Existem várias formas de comunicação: *Wireless Personal Area Network* (W-PAN) refere-se às redes sem fios de fraco alcance, é projetada para pequenas distâncias, baixo custo e baixas taxas de transferência; *Wireless Local Area Network* (W-LAN) é uma comunicação de pacote de dados de curto alcance entre estações-base e terminais de usuário e *Wireless Wide Area Network* (W-WAN) são redes com alcance de grandes distâncias geográficas, como uma cidade, um país ou todo o mundo. Dentre algumas tecnologias de comunicação sem fio atualmente disponíveis no Brasil se destacam:

BLUETOOTH é um padrão proposto pela *BluetoothSIG*, voltado para utilização em redes W-PAN, possibilita troca de dados ponto a ponto. O padrão possibilita taxa efetiva de transferência de dados de até 723kbit/s. Possui três classes que especificam a potência do sinal e o alcance, é afetado por obstáculos como paredes e divisórias (OLIVEIRA; PINTO, 2009):
Classe 1, com alcance de até 100m;
Classe 2, com alcance de até 10m;
Classe 3 com alcance de até 10cm.

IEEE 802.11 é um padrão definido pela *Institute of Electrical and Electronics Engineers* (IEEE) voltado para a comunicação em redes W-LAN, sendo também conhecido por "*Ethernet* sem fio". Possibilita conexões ponto a ponto ou a formação de redes com infraestrutura com pontos de acesso à rede fixa. A especificação *Wireless Fidelity* (Wi-fi) possibilita taxas de transferências de 11Mbps a 54Mbps e alcance de até 300m em ambientes abertos,

com degradação progressiva da taxa de transferência (GARDUSSI; SOUZA, 2009).

WAP [1] é um padrão internacional para aplicações que utilizam comunicações de dados digitais sem fio, permite o acesso à Internet através de aparelhos portáteis como celulares e *handhelds* (HANZAN, 2009).

GPRS [2] é um padrão da tecnologia *Global System for Mobile communication* (GSM) voltado para utilização em redes *W-WAN*. Possibilita velocidades de conexão de até 171Kbps. O padrão é baseado na comutação de pacotes, permitindo a cobrança por dados trafegados e uma total integração com a Internet, como a utilização de serviços de navegador, FTP, *e-mails*, dentre outros. O padrão pode ser utilizado em redes celulares GSM e textitTime Division Multiple Access (TDMA) (OLIVEIRA; PINTO, 2009).

3.2 Ambientes de Desenvolvimento Integrado

Nas subseções seguintes são apresentas as IDEs utilizadas na contrução da ferramenta m-tutorial, na IDE *Eclipse* foram modelados os diagramas UML e na IDE *NetBeans* a implementação das classes pertencentes a ferramenta.

3.2.1 Eclipse

Segundo Gonçalves (2007), *Eclipse*[3] é uma IDE de programação escrita completamente em Java, a plataforma pode ser utilizada em estações de desenvolvimento que envolve os sistemas operacionais *Linux*,QNS, OSx e *Windows*. Inicialmente desenvolvida pela IBM, é uma plataforma de código aberto que inclui grandes empresas como *Borland, IBM, Merant, Rational Software, Red Hat, SuSe, QNX Software Systems, Webgain, Hewlett Packard, Fujitso, Oracle e Sybase*. O *Eclipse* em si fornece o ambiente integrado para a execução dos mais variados *plug-ins*. Atualmente, esta poderosa IDE se encontra na versão 3.6, chamada Helios.

3.2.2 Netbeans

NetBeans[4] *Integrated Development Environment* (IDE) é um ambiente gráfico e integrado, de desenvolvimento de *software*, baseado na tecnologia Java, tendo como principal patrocinadora a empresa *Sun Microsystems*. Segundo Emilio e Severo (2005), o ambiente *NetBeans* é composto por um conjunto de ferramentas que permitem a edição, compilação, depuração,

[1] *Wireless Application Protocol*
[2] General *Packet* Radio *Service*
[3] http://www.eclipse.org
[4] http://www.netbeans.org/

documentação e instalação de produtos baseados na tecnologia *Java*. Também é possível estender as capacidades por meio da inclusão de novos módulos, aspecto que torna o ambiente flexível e adaptável às necessidades do desenvolvedor.

3.3 Serviços Web

3.3.1 XML

É um sistema gramatical para construção de linguagens de marcação personalizadas, recomendada pela *World Wide Web Consortium* (W3C) é um consórcio internacional de empresas de tecnologia, que tem como objetivo criar padrões e diretrizes que garantam uma evolução permanente da web. Inspirado de duas fontes distintas: *Standard Generalized Markup Language* (SGML) e *Hypertext Markup Language* (HTML) (CASTRO, 2001).

O tipo de serviço largamente aceito e bem sucedido é o XML *Web Services*.

Segundo Mcgrath (2009) este tipo de serviço possui dois requisitos fundamentais:

- Comunica-se via protocolos *Internet* (normalmente HTTP);
- Envia e recebe dados formatados como documentos XML.

3.3.2 Web Service

Web Service tem a funcionalidade de integrar sistemas, baseando-se em um protocolo conhecido como *Simple Object Access Protocol* (SOAP) e trafegando através da mesma porta de Internet, sem a necessidade de alterar *Proxy* ou qualquer permissão de acesso à rede (PAULON, 2008). Essa tecnologia vem se tornando comum pela sua interoperabilidade e simplicidade de comunicação, além de poder ser implementada utilizando apenas *softwares* gratuitos.

Segundo Pires e Matoso (2005), por meio da tecnologia de Serviços Web, “pode-se encapsular processos de negócios preexistentes, publicá-los como serviços, descobrir dinamicamente serviços publicados e trocar informações que atravessam os limites de uma corporação".

Os *Web Services* surgiram como consequência natural da utilização da Internet. Alguns consideram essa utilização massificada, como um processo que produz a evolução desse meio de comunicação entre pessoas e redes de computadores, o que naturalmente levou à possibilidade de se escrever aplicações e disponibilizá-las ao público em grande escala (ABINADER; LINS, 2006).

Dentre as muitas técnicas utilizadas em aplicações móveis, *Web Service* surgiu com a idéia de resolver os problemas de incompatibilidade entre os protocolos, uso de portas e forma-

tos binários rejeitados por *firewalls*, dentre outros problemas apresentados em *software* projetados com a arquitetura em camada (ROCHA, 2008). Os principais benefícios dos *Web Services* são protocolos baseados em padrão XML de transporte como SOAP e HTTP, permitem a geração automática tanto do código cliente, quanto do código do servidor (SRIRAMA; JARKE; PRINZ, 2008).

A seguir são apresentados alguns protocolos *Web Service* segundo Alonso et al. (2004):

SOAP especifica como representar em XML todas informações necessárias para se executar uma chamada remota. Permite adicionar serviços a um servidor Web e então acessar esses serviços a partir de programas comuns.

WSDL [5] especifica a assinatura das operações que serão implementadas por um objeto remoto. Descreve a localização e a interface de um serviço Web.

UDDI [6] conjunto de especificações para registrar ou pesquisar um serviço Web, via interface Web ou SOAP

As bases para a construção de um *Web Services* são os padrões XML e SOAP. O transporte dos dados é realizado normalmente via protocolo *Hypertext Transfer Protocol* (HTTP) ou HTTPS para conexões seguras (o padrão não determina o protocolo de transporte). Os dados são transferidos no formato XML, encapsulados pelo protocolo SOAP. Abinader e Lins (2006) descrevem porque razões usar *Web Services*:

- Permite utilizar regras de negócio através da rede;
- Conecta aplicações de diferentes fornecedores;
- Utiliza protocolo padronizado (SOAP/WSDL/UDDI);
- Possui baixo custo de comunicação (*Internet*);
- Permite publicação automática (UDDI).

3.3.3 Apache Tomcat

O *Apache Tomcat* é um servidor de aplicações Java para Web, um container Web que implementa as especificações *Java Server Pages* e *Java Servlets*. Desenvolvido desde o início para flexibilidade e desempenho (KURNIAWAN, 2002).

[5] Web Service Description Language
[6] Universal Description, Discovery, and Integration

3.3.4 Apache Axis

O *Apache Axis* é uma implementação *Open Source* cliente/servidor do *Simple Object Access Protocol* SOAP, padrão W3C. É um *framework* que facilita o desenvolvimento de *Web Services* fornecendo APIs para desenvolvimento de *serviços Web* em Java ou C++ (ABINADER; LINS, 2006). Os desenvolvedores podem criar aplicações interoperáveis de computação distribuída usando o *Apache Axis*.

3.4 Sistemas Operacionais

Em relação às tecnologias, existem alguns sistemas operacionais para dispositivos móveis tais como *iPhone OS, Symbian OS, Maemo, Windows Mobile* e *Android*.

3.4.1 iPhone OS

Desenvolvido pela *Apple* para *iPhone e iPod Touch*. Baseado no *Mac OS X*, sua linguagem nativa é o *Objective C* e seu kit de desenvolvimento está disponível apenas para *Mac OS X*. Sua distribuição é exclusivamente através da *iTunes App Store* (SARAIVA, 2009).

3.4.2 Symbian OS

Desenvolvido pela *Symbian LTDA* que era dividida entre *Nokia* (56%), *Sony Ericsson, Panasonic* e *Samsung*. É o líder do mercado (segundo a AdMob, possui cerca de 50% das vendas de *Smartphones*, mas esse número está caindo). Em 2008, a *Nokia* adquiriu todos os direitos da *Symbian*, pretendendo tornar o sistema *Open Source*. Sua Linguagem nativa é o *Symbian C++* e utiliza o kit de desenvolvimento *carbide C++*, baseado no *Eclipse* (POHJOLAINEN, 2006).

3.4.3 Windows Mobile

Desenvolvido pela *Microsoft*, baseado na API Win32 da *Microsoft*. Sua linguagem nativa é o C# e VB e seu kit de desenvolvimento está disponível apenas para *Windows* (POHJOLAINEN, 2006).

3.4.4 Android

Desenvolvido pela *Open Handset Alliance* OHA e lançado em novembro de 2007 pela empresa *Google*, com participação da HTC, *Intel*, *Motorola*, *Qualcomm*, *T-Mobile* e *NVIDIA*.

É um Sistema Operacional aberto baseado em *Linux*. Sua linguagem nativa é o *Java* e seu kit de desenvolvimento é o *Android SDK* que disponibiliza as ferramentas e APIs necessárias para desenvolver aplicações para a plataforma *Android*. Disponível para *Windows*, *Linux* e *Mac OS X* (FRANCISCATO; MEDINA, 2008).

3.5 Plataformas de Desenvolvimento

Em relação às tecnologias, existem algumas plataformas para o desenvolvimento em dispositivos móveis tais como *Java ME, Brew* e *Flash Lite*.

3.5.1 Java ME

Desenvolvida e disponibilizada pela Sun *Microsystems*, em 1994, a linguagem *Java* tornou-se popular desde então. Segundo Johnson (2007) seu sucesso se deve, sobretudo, à portabilidade de código, que permite que um mesmo código-fonte seja usado em diferentes plataformas (*desktops*, *mobile*, servidores, etc.).

Devido ao surgimento da Internet a partir de 1993, novas oportunidades apareceram e a Sun resolveu criar um tipo de linguagem “universal". Para isso, a linguagem *Java* quando compilada gera um *bytecode*, que é lido por um interpretador conhecido como JVM (Java *Virtual Machine*) e pode ser executado em qualquer tipo de sistema computacional para diversos tipos de *hardware*.

A plataforma Java está dividida em: *Java Entreprise Edition*(Java EE), *Java Standard Edition* (Java SE), *Java Micro Edition* (Java ME) e *Java Card*.

O Java ME é uma versão de Java para dispositivos com limitações de memória e processamento, como celulares, PDAs e sistemas embarcados em geral (MUCHOW, 2001).

3.5.2 Brew

Segundo Muller, Frantz e Schreiber (2005), a *Qualcomm* projetou o ambiente de desenvolvimento binário para dispositivos móveis conhecido por *Binary Runtime Environment for Wireless* (BREW), lançado em janeiro de 2001. Neste ambiente, as aplicações são desenvolvidas em C/C++, compiladas e executadas em um chip separado do processador principal do dispositivo.

3.5.3 Flash Lite

A Plataforma *Flash* permite que os desenvolvedores entreguem aplicações que rodem em *desktop* e dispositivos móveis. *Flash Lite* é uma versão leve do *Adobe Flash Player*, projetado para dispositivos móveis. A vantagem do *Flash* sobre as outras tecnologias vem com a maturidade que a plataforma permite o desenvolvimento de *Rich Internet applications* RIAS ou, mais recentemente, para dispositivos móveis. A curva de aprendizado chega a ser desmotivadora para quem quer aprender C/C++ ou *Java ME* em comparação com *Flash Lite* (ANDRADE, 2006). Um dos problemas é a incompatibilidade entre as versões do *Flash*, no caso do programador utilizar recursos somente disponíveis em algumas das versões existentes, o que é comum ocorrer com qualquer *software*.

3.6 Comparando Tecnologias

Dentre as muitas plataformas de desenvolvimento existentes, realizou-se um levantamento comparativo entre as principais plataformas de desenvolvimento móveis, entre elas *Flash Lite*, *Java ME*, *Brew*, *Android* e Sistemas Operacionais como *Windows Mobile, Android, iPhone e Maemo*. Comparou-se a linguagem de programação, a curva de aprendizado, a IDE utilizada, a portabilidade, o custo, documentação e o fabricante.

Os itens avaliados foram importantes para escolha da plataforma e sistema operacional. No item linguagem foi realizado um estudo sobre o tipo de linguagem de programação utilizada. O levantamento do item curva de aprendizado se deu por meio de estudos realizados por Li (2009) em sua pesquisa *Open Source Software for Mobile Phones* onde a classificação do aprendizado se dá pela facilidade encontrada no manuseio da plataforma. O item IDE mostra os ambientes de desenvolvimento utilizados pelas plataformas e sistemas operacionais avaliados. Da portabilidade foi avaliada a possibilidade do *framework* rodar em múltiplas plataformas. O levantamento dos custos para o desenvolvimento do *framework* é importante pois define a forma de distribuição de um *software*. O item Documentação foi avaliado diante do número de documentos encontrados tais como livros, revistas, artigos, blogs e outros disponíveis para o aprendizado. Já o item Empresa Fornecedor foi incorporado ao comparativo com o propósito de informar o nome da empresa responsável por estas tecnologias.

O Quadro 3.6 apresenta as questões consideradas na escolha da plataforma de desenvolvimento. A plataforma *Flash Lite* possui custos variáveis de distribuição, *Symbian* é incompatível com vários aparelhos celulares, o *Android* é recente e utiliza a linguagem Java. Sua portabilidade ainda é desconhecida. Já o Java ME alcança o maior número de micro dispositivos disponíveis no mercado, independente de arquitetura, de código livre, de fácil aprendizado, além de boa portabilidade.

Plataforma e Sistema Operacional							
	Linguagem	**Aprendizado**	**IDE**	**Portabilidade**	**Custo**	**Doc.**	**Empresa Fornecedor**
Symbian	C++	Difícil	*Many Choices,Carbide* C++	Somente *Symbian*	Variável	X	*Nokia*
Android	Java	Médio	*Eclipse*	Somente *Android*	Gratuito	X	*Google*
***Windows* Mobile**	C#,VB	Médio	Visual *Studio*	Somente *Windows Mobile*	Variável	X	*Microsoft*
***Flash* Lite**	*ActionScript*	Fácil	*Macromedia Flash*	Excelente	Variável	X	*Adobe*
iPhone	Objective-C	Médio	Xcode	Somente iPhone	Gratuito, custos de distribuição	X	Apple
Java ME	Java	mÉDIO	*Sun Wireless Toolkit,Eclipse, NetBeans*	Media	Gratuito	X	*Oracle*
Brew	C/C++	Médio	Visual Studio, Visual C++	Media	Gratuito	X	*Qualcomm*
Maemo	C	Médio	Maemo SDK	Domente Maemo	Gratuito	X	*Linux*

Fonte: (LI, 2009)

Plataforma e Sistema Operacional

4 APIS, BIBLIOTECAS E *FRAMEWORKS*

Este capítulo fornece uma visão geral da coleção de APIs e apresenta os principais *frameworks* e bliotecas existentes para a construção de aplicações utilizando dispositivos móveis. A Seção 4.1 apresenta as principais APIs disponíveis para MIDP. A Seção 4.2 apresenta as principais bibliotecas baseadas em Canvas. Na Seção 4.3 serão abordados conceitos sobre *framework*demonstrando os mais utilizados para Java ME A Seção 4.3.1 apresenta a biblioteca *Lwuit*, útil à construção de interfaces gráficas em Java ME.

4.1 APIs

A tecnologia Java possui uma comunidade *Java Community Process* (JCP) conhecida por uma organização internacional de desenvolvedores Java, cujo objetivo é o de desenvolver e rever especificações e implementações. A JCP envolve o uso de *Java Specification Requests* (JSRs), documentos formais que descrevem as especificações propostas e tecnologias da plataforma Java (BARRY, 2003). Um conjunto de rotinas e padrões são estabelecidos e conhecidos por *Application Programming Interface* (API).

As APIs oferecem recursos diversos. São projetadas para rodar em qualquer máquina virtual baseada em Java ME. Algumas das APIs opcionais disponíveis para MIDP são citadas por (JOHNSON, 2007):

JSR 139 (CLDC)- Define o conjunto básico de interfaces de programação de aplicativo e máquina virtual para dispositivos com recursos limitados;

JSR 118 (*Mobile Information Device Profile*)- É uma extensão da API de base fornecida pela configuração CLDC. Permite escrever aplicativos para *download* e serviços de rede;

JSR-234 (*Advanced Mobile Media API*)- Define uma API com funcionalidades multimídias avançadas. Ela permite acesso a recursos multimídias mais sofisticados como suporte avançado de câmera, áudio, rádio e processamento de imagem;

JSR-226 (*Scalable vector graphics*)- Define uma API para desenhar gráficos bidimensionais baseados em gráficos vetoriais (redimensionáveis sem perda de qualidade);

JSR-184 (*3D-graphics*)- Define uma API leve para gráficos 3D interativos. A API é muito importante para criação de interfaces ricas e desenvolvimento de jogos;

JSR-120 (*Text Message*)- Provê acesso à plataforma independente para recursos de comunicação sem fio como serviço de mensagens curtas (SMS) e mensagens multimídia (MMS);

JSR-135 (*MultiMedia*)- Suporta aplicações multimídia em dispositivos habilitados para *Java ME*. Ele foi projetado para funcionar com qualquer protocolo e formato, por exemplo, MP3, MIDI, ou MPEG-4;

JSR-75 (*File Connection*)- Permite acesso e armazenamento em arquivos e acesso a programas nativos do dispositivo;

JSR-82 (*Bluetooth*)- Permite o desenvolvimento de aplicações para acesso a rede, dependendo da disponibilidade da tecnologia *Bluetooth* no dispositivo;

JSR-172 (*web service*)- Define um pacote opcional que permite o acesso padrão do *Java ME* para *web service*.

RMS (*Record Management System*)- é a solução do J2ME para a criação e manipulação de registros oferecido pelo MIDP[1] para persistência de dados. Localizado no pacote javax.microedition.rms, é composto pela classe *RecordStore*, conhecida como armazém de dados.

4.2 Bibliotecas

Bibliotecas são basicamentes arquivos contendo código reutilizável que normalmente pode ser compartilhado entre vários aplicativos, com elas você não precisa escrever o mesmo código várias vezes (HORSTMANN, 2006).

Segundo Horstmann (2006), uma biblioteca é um conjunto de objetos e funções que se concentra em torno de resolver um problema particular ou em torno de uma área específica de desenvolvimento de aplicações, ou seja, acessar banco de dados.

Das diversas bibliotecas existentes para Java ME, as listadas no quadro 4.2 são baseadas em Canvas. Com elas o usuário pode criar Interfaces mais atraentes do que é possível com o alto nível das classes IU (*Form, List, TextBox*, etc). O quadro 4.2 especifica as principais bibliotecas de interface gráfica da plataforma Java ME.

[1] Mobile Information Device Profile

Principais bibliotecas de Interface Gráfica da Plataforma Java ME.		
NOME	**COMENTÁRIOS**	**LICENÇA**
Lwuit	O *Lightweight UI Toolkit(Lwuit)* é um projeto *open source* que ajuda desenvolvedores na criação de aplicativos com *designs* mais sofisticados, oferecendo componentes visuais, e animações para aplicativos Java ME. Ele representa uma alternativa ao *LCDUI (LCD User Interface)* já presente no Java ME. A biblioteca tem apenas 256 KB. Ela oferece aos desenvolvedores, a partir de um custo baixo de espaço de armazenamento, uma alternativa aos componentes LCDUI oferecidos pela plataforma básica do Java ME (BOSSARD, 2008).	GPLv2
LwVCL	É uma biblioteca projetada, para funcionar em qualquer plataforma. Suporta tanto JSE e .NET. Possui uma versão de rascunho para *Java ME MIDP* (GHISI, 2008).	GPL
J2ME Polish	É um conjunto de bibliotecas *open source* e ferramentas que visam auxiliar o programador a lidar com vários aspectos da interface gráfica na criação de aplicações para dispositivos de pequeno porte (ENOUGH, 2010). O *J2ME Polish* possui uma base de dados com mais de 300 dispositivos celulares que podem usar Java ME. Conta também com ferramentas de construção (*build tools*). Essas ferramentas permitem que um único código de aplicação possa ser utilizado em aparelhos de diferentes marcas e modelos (FERNANDES, 2007).	GPL/ COMERCIAL
J4me	É uma biblioteca de código aberto para ajudar a construir aplicações Java ME. J4ME torna o desenvolvimento mais rápido e produtivo e aplicações mais bonitas. No entanto, exige MIDP 2.0 (GHISI, 2008).	Apache License 2.0
Fire	É uma biblioteca que se foca em prover componentes fáceis de serem utilizados por desenvolvedores Java ME. Provê o básico de Java ME (combos, listas, forms, item) mais customizações nesses componentes. Essa biblioteca não depende de aparelho ou tamanho de tela em específico, podendo ser portada para qualquer aparelho (BOSSARD, 2008).	LGPL
MWT	MWT entra em cena para a construção de UIs otimizados para celulares. Inspirado no seus irmãos mais velhos como *AWT, Swing* e *SWT* (LI, 2009).	LGPL

Principais bibliotecas de Interface Gráfica da Plataforma Java ME.

4.3 *Framework*

Segundo Johnson (1997), *framework* é como o esqueleto de uma aplicação, que pode ser customizado por um desenvolvedor de aplicações, ou seja, é uma aplicação semi-completa e reutilizável que, quando especializada, produz aplicações personalizadas. *Frameworks* diferem de outras técnicas de reuso, como a utilização de componentes de *software* e a de padrões de projetos.

Frameworks são implementações feitas em linguagens orientadas a objetos como Java e C++. Componentes de *software* definem unidades reutilizáveis que oferecem serviços através de

interfaces bem definidas. Essas duas tecnologias são complementares de forma que *frameworks* podem ser usados para auxiliar o desenvolvimento de componentes e vice e versa (SZYPERSKI, 1998). Já padrões de projeto descrevem uma solução para o problema. Um *framework* pode possuir vários padrões de projeto, para, o contrário não acontece (BARRETO, 2006).

Para Buschmann et al. (1996), um *framework* é definido como um *software* parcialmente completo projetado para ser instanciado. O *framework* define uma arquitetura para uma família de subsistemas e oferece os construtores básicos para criá-los. Também são explicitados os lugares ou pontos de extensão *(hot-spots)* nos quais devem ser feitas adaptações do código para um funcionamento específico de certos módulos.

Dos *frameworks* existentes para Java ME, os listados no quadro 4.3 são baseados em interfaces gráficas. O quadro 4.3 especifica os principais *frameworks* de interface gráfica da plataforma Java ME.

Principais *frameworks* de interface gráfica da plataforma Java ME.		
NOME	**COMENTÁRIOS**	**LICENÇA**
Kuix	*Kuix* é um *framework* que permite desenvolver facilmente aplicações Java ME. Ele fornece diversos elementos gráficos (botões, campos de texto, listas, menus, etc.) necessários na criação de interfaces gráficas com usuário avançado utilizando XML / CSS abordagem para descrever as telas e as ações do usuário na aplicação (LI, 2009).	GPL
jMobileCore	É um *framework* poderoso para criação de aplicações Java ME. *jMobileCore* fornece suporte para o desenvolvimento compacto e rico de Canvas baseada em GUI. O acesso rápido de dados e comunicações simplifica a criação de aplicações *multithreading midlet* (LI, 2009).	LGPL
Apime	É um *framework* usado para oferecer mais funcionalidade ao celular com Java ME. O núcleo das funcionalidades é interface gráfica, contendo componentes para construir aplicações com a estrutura do Swing de J2SE (LI, 2009).	GPL

Principais *frameworks* de interface gráfica da plataforma Java ME.

As APIs são interfaces expostas por componente, biblioteca é um subprograma ou coleção de subprogramas reutilizadas em aplicações. Ao contrário das bibliotecas, *framework* é quem dita o fluxo de controle da aplicação, chamado de inversão de controle. Ele pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação. Do levantamento feito nas bibliotecas e *frameworks* listados na seção anterior, foi decidido utilizar a biblioteca *Lwuit* por sua facibilidade de aprendizado, pela linguagem de programação Java, por ser gratuito e de boa portabilidade.

4.3.1 Lwuit

Lightweight UI Toolkit (Lwuit) é uma biblioteca que permite criar interfaces gráficas do usuário para dispositivos móveis ou quaisquer outros dispositivos que suportem o perfil MIDP

do Java ME. O *Lwuit* traz diversos recursos capazes de tornar a interface dos aplicativos em Java ME mais agradável, bonita e simples de usar. Vantagens (HILDI, 2009):

- Arquitetura MVC;
- Diversidade de *layouts* e fontes;
- Diversidade de estilos e temas;
- Suporte *touch screen*;
- Animações e transições;
- Multi-plataforma;
- Compatibilidade com SVG[2].

Pensada e baseada na biblioteca *swing* do Java SE, o *Lwuit* possui compatibilidade com MIDP 2.0, CDC e Java SE (KNUDSEN; FISHBEIN; LEVIN, 2009). Ele abstrai códigos RMS para persistência de dados.

A Figura 4.1 ilustra a hierarquia de interface do *framework Lwuit*. Todas as classes deste *framework* derivam de *Component* que é a superclasse abstrata geradora das classes visuais. *Container* é um *Component* responsável por armazenar vários componentes. Já os componentes List e ComboBox são cruciais para aplicações interativas, *TextArea* especifica múltiplas linhas e colunas. Já o objeto *Label* instancia funcionalidades aos componentes *Button*, *RadioButton* e *CheckBox*.

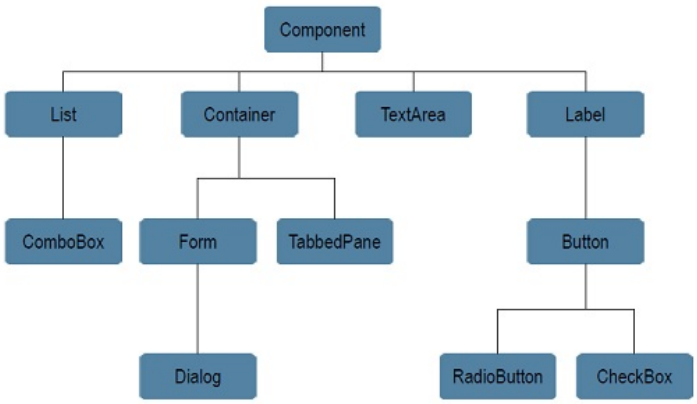


Figura 4.1: Hierarquia *Lwuit*
Fonte: (SUN, 2009)

O m-tutorial foi construído a partir de especializações feitas das classes do *framework Lwuit*. A forma de implementação e arquitetura foram preservadas.

[2]scalable vector graphics

5 USABILIDADE

O objetivo deste capítulo é fornecer o embasamento teórico sobre o ambiente móvel considerado nesta dissertação, informações sobre a usabilidade e as características do método de avaliação de usabilidade escolhido. A Seção 5.1 apresenta os principais conceitos de usabilidade, na Subseção 5.1.1 discute-se a usabilidade em dispositivos móveis, na Seção 5.2 são apresentados métodos de avaliação de usabilidade para dispositivos móveis e, por fim, na Subseção 5.2.1 é apresentado a avaliação heuriística, um método de inspeção de interface.

5.1 Conceito de Usabilidade

A usabilidade de um sistema é a característica que determina a facilidade de manuseio e a rapidez com que ele é aprendido (FERREIRA; LEITE, 2003).

O projeto da interface contribui para compreensão do sistema, serve de apoio para satisfação e controle pelos usuários. É importante conhecer o público-alvo para envolvê-lo no projeto (PREECE; ABRAS; KRICHMAR, 2004).

Segundo Ferreira e Leite (2003), é através da interface com o usuário que a comunicação é estabelecida com o sistema. O projeto da interface deve fazer com que esta interação seja transparente, pois o usuário deve focar sua atenção no trabalho. Uma boa interface satisfaz as necessidades do usuário. Por isso, uma interface deve ser centrada nessas necessidades.

A avaliação de usabilidade, quando aplicada no início do projeto, auxilia para que as alterações sejam realizadas o quanto antes e que os requisitos de usabilidade sejam incorporados ao produto (BETIOL, 2004).

5.1.1 Usabilidade em Dispositivos Móveis

Para realizar a análise de usabilidade em dispositivos móveis é preciso respeitar as diferenças existentes entre eles e o computador de mesa. São várias as diferenças tanto em relação à finalidade, como em seu contexto de uso. Computador de mesa é fixo, está sempre conectado via cabos, possui memória e capacidade de armazenamento praticamente ilimitadas, tela grande e colorida e entrada de dados via *mouse* ou teclado. Os portáteis apesar de serem mais leves e permitirem conexão sem fio, têm pouca memória, pouca ou quase nenhuma capacidade de armazenamento, bateria que deve ser sempre recarregada, conexão ainda lenta e muitas vezes pouco confiável (BETIOL, 2004).

Para solução de usabilidade em sistemas móveis existem problemas para serem enfrentados, como (JOHNSON, 1998):

- Padrão entre os diversos dispositivos, tanto ao seu uso quanto a suas interfaces;
- A inadequação dos atuais modelos de usabilidade,
- As exigências de avaliação dos sistemas móveis;
- As exigências de projeto para usuários móveis, suas tarefas e contextos.

Segundo Betiol (2004), a avaliação de usabilidade para dispositivos móveis dentro do laboratório não fornecem resultados tão satisfatórios quanto as avaliações realizadas próximas à realidade dos usuários, fora dos laboratórios. Conforme Johnson (1998), o laboratório convencional não seria capaz de simular adequadamente aspectos tão importantes como o tempo, e necessidade diária das pessoas.

Esses dispositivos são usados por pessoas em movimento, o que provoca uma capacidade de concentração limitada pelos envolvidos (WEISS, 2002). Levando em consideração esses aspectos e o tamanho reduzido dos aparelhos, vê-se a importância da avaliação de usabilidade para garantir a qualidade no resultado desse projeto.

Deve-se observar ainda algumas diretrizes em *design* de interfaces traçadas para atender às necessidades dos usuários desses dispositivos. Essas não devem ser entendidas como uma receita, mas sim como um conjunto de princípios que auxiliam no desenvolvimento de interfaces (ROCHA; BARANAUSKAS, 2003).

Neves (2005) sumarizou algumas diretrizes para sistemas móveis conforme apresentadas no Quadro 5.1.1. O quadro está dividido em dois grupos de diretrizes: o primeiro grupo, formado pelas características de 1 a 8, corresponde às diretrizes específicas de usabilidade para sistemas móveis e o segundo grupo, formado pelas características restantes (9 a 11), contém diretrizes relacionadas à usabilidade em geral, ao *design* do conteúdo e arquitetura da informação.

Característica	Diretrizes para Sistemas Móveis
1 - Mobilidade	• Fornecer conteúdo relacionado com atividades que se quer ou se necessita fazer enquanto em movimento.
2 - Utilidade	• Contribuir para simplificar a vida do usuário.
3 - Relevância	• Conter apenas a informação que o usuário necessita; • Fornecer conteúdo altamente direcionado: o acesso deve ser simples e o conteúdo de valor para o usuário.
4 - Facilidade de uso	• Fazer com que os usuários não tenham que lembrar itens durante a navegação ou tarefas; • Colocar rótulos nos botões de *hardware* sempre que possível; • Serviço fácil e confortável de usar; • Manter a interação tão óbvia e descomplicada quanto possível *(kiss - "keep it simple and stupid")*.
5 - Foco no usuário	• Envolver o usuário em todo o ciclo de desenvolvimento a fim de aproximar o *design*, o máximo possível, de suas expectativas; • Combinar avaliação teórica e empírica para elaborar novas formas de melhorar a usabilidade; • Priorizar a navegação das tarefas-chave para o usuário; • Os novatos devem ser apresentados aos serviços da forma mais concisa possível; • Não falar de tecnologia - os usuários querem saber das funcionalidades e dos benefícios disponíveis, não estão interessados na tecnologia utilizada.
6 - Personalização	• Reduzir a necessidade de exibir informação por meio da personalização em relação às necessidades do usuário; • Implementar serviços que se tornam disponíveis de acordo com a localização, permitindo ao usuário usar aplicações relacionadas com o lugar onde se encontra.

Característica	***Guideline* para Sistemas Móveis**
7 - Confiabilidade	• Verificar a integridade do conteúdo exibido, os serviços devem ser testados em diferentes dispositivos.
8 - Eficiência do uso	• Usar números como entrada sempre que possível; • Estimar o tamanho da tela e a qualidade da informação que poderá ser exibida no dispositivo alvo; • Minimizar o número de cliques; • Evitar rolagem da tela; • Limitar o escopo de pesquisas e melhorar sua eficiência.
9 - Consistência	• Usar convenções padronizadas para os botões; • Projetar uma navegação consistente com um navegador tradicional; • Ter um botão voltar com mesma função realizada em um navegador tradicional; • Utilizar convenções sugestivas para *links*.
10 - Estrutura da navegação	• Navegação transparente: o usuário precisa saber onde está, de onde veio e como continuar; • Estrutura de navegação clara, simples e rasa; • Minimizar níveis na hierarquia da informação; • Colocar o mínimo possível de *links* na página de entrada; • Reduzir navegação entre páginas ao mínimo possível.
11 - *Design* do conteúdo	• Permitir explorar a formatação do texto ao invés de lê-lo; • Evitar jargões e abreviações arbitrárias; • Confinar todo elemento gráfico nas dimensões da tela; • Oferecer listas de opções (com valores default quando possível); • Projetar pequenos blocos de informação que possam ser visualizados de uma só vez na tela e, em uma carta, estruturar blocos de textos grandes.

Fonte: (Neves, 2005)

Diretrizes para Sistemas Móveis

5.2 Métodos de Avaliação

A avaliação de interfaces não deve ser realizada somente ao final de um processo de *design*, ou somente em uma fase do processo, ou caso exista tempo. É ideal que ocorra durante o ciclo de vida do *design* e seus resultados utilizados para melhorias gradativas da interface (ROCHA; BARANAUSKAS, 2003).

Alguns dos principais objetivos de se realizar avaliação interativas de sistemas são (PRATES; BARBOSA, 2003):

- Identificar as necessidades de usuários ou verificar o entendimento dos projetistas sobre estas necessidades;
- Identificar problemas de interação ou de interface;
- Investigar como uma interface afeta a forma de trabalhar dos usuários;
- Comparar alternativas de projeto de interface;
- Alcançar objetivos quantificáveis em métricas de usabilidade;
- Verificar conformidade com um padrão ou conjunto de heurísticas.

Em diferentes estágios de um processo de *design* são aplicáveis diferentes tipos de avaliação de interfaces. Um exemplo a ser utilizado são os estágios iniciais quando idéias estão sendo exploradas e testadas. Nesses casos, testes informais podem ser suficientes (ROCHA; BARANAUSKAS, 2003). Para avaliar interfaces de usuário, é importante conhecer as características de cada método e conhecer qual deles pode ser aplicado em cada caso nos diferentes tipos de *software* (PRATES; BARBOSA, 2003).

Os métodos de avaliação de interfaces com o usuário podem ser classificados, conforme (ROCHA; BARANAUSKAS, 2003):

- Se usuários reais estão ou não envolvidos;
- Se a interface está ou não implementada.

Dentro deste contexto, são citados os métodos de avaliação de interfaces (ROCHA; BARANAUSKAS, 2003):

- Inspeção de usabilidade - não envolve usuários e pode ser utilizada em qualquer fase do desenvolvimento de um sistema (implementado ou não);

- Teste de usabilidade - método de avaliação centrado no usuário que inclui métodos experimentais ou empíricos, métodos observacionais e técnicas de questionamento;
- Experimentos controlados - experimentos de laboratório que devem ser projetados e controlados. A análise do resultado é feita por meio de dados estatísticos;
- Métodos de avaliação interpretativos - possibilitam aos *designers* um maior entendimento da reação dos usuários. A análise é realizada por meio do uso do sistema no ambiente onde, normalmente, irão utilizá-lo e como o uso desse sistema irá interagir com outras tarefas.

O método de avaliação de interface utilizado para conhecer o grau de usabilidade do *framework* foi o de inspeção de usabilidade, representado por meio da avaliação heurística. Nesse método de avaliação, são os avaliadores que inspecionam ou examinam o ambiente (ROCHA; BARANAUSKAS, 2003). Esses avaliadores podem ser especialistas em usabilidade, consultores de desenvolvimento de *software*, especialistas em um determinado padrão de interface, usuários finais com conhecimento da tarefa ou do contexto de uso (ROCHA; BARANAUSKAS, 2003) e (NASCIMENTO et al., 2007).

5.2.1 Avaliação Heurística

A Avaliação Heurística é um método de inspeção de interface que utiliza como base uma pequena lista de heurísticas conhecidas como princípios reconhecidos de usabilidade (PELISSONI; CARVALHO, 2003).

O método de avaliação heurística deve ser realizado por especialistas. Segundo Rocha e Baranauskas (2003), 3 avaliadores são suficientes. Apesar de serem avaliadores com conhecimento em tecnologia, as preferências pessoais podem afetar o resultado da avaliação. Com mais pessoas é possível melhorar a análise e diagnosticar estas impressões. Nesse caso não são envolvidos usuários. O método apresenta baixo custo em relação à maioria dos métodos de avaliação e é rápido (PRATES; BARBOSA, 2003).

As heurísticas definidas por (NIELSEN, 1994), estão relacionadas no Quadro 5.2.1:

LISTA DE HEURÍSTICAS DE USABILIDADE	
1. Visibilidade do status do sistema	Sistema precisa manter os usuários informados sobre o que está acontecendo, fornecendo um *feedback* adequado dentro de um tempo razoável.
2. Compatibilidade do sistema com o mundo real	Sistema precisa falar a linguagem do usuário, com palavras, frases e conceitos familiares ao usuário, ao invés de termos orientados ao sistema. Seguir convenções do mundo real, fazendo com que a informação apareça numa ordem natural e lógica.
3. Controle do usuário e liberdade	Usuários frequentemente escolhem por engano funções do sistema e precisam ter claras saídas de emergência para sair do estado indesejado sem ter que percorrer um extenso diálogo.
4. Consistência e padrões	Usuários não precisam adivinhar que diferentes palavras, situações ou ações significam a mesma coisa. Seguir convenções de plataforma computacional.
5. Prevenção de erros	Melhor que uma boa mensagem de erro é um *design* cuidadoso o qual previne o erro antes dele acontecer.
6. Reconhecimento ao invés de relembrança	Tornar objetos, ações e opções visíveis. O usuário não deve ter que lembrar informação de uma para outra parte do diálogo. Instruções para uso do sistema devem estar visíveis e facilmente recuperáveis quando necessário.
7. Flexibilidade e eficiência de uso	Usuários novatos se tornam peritos com o uso. Permitir a usuários experientes cortar caminho em ações freqüentes.
8. Estética e *design* minimalista	Diálogos não devem conter informação irrelevante ou raramente necessária. Qualquer unidade de informação extra no diálogo irá competir com unidades relevantes de informação e diminuir sua visibilidade relativa.
9. Ajudar os usuários a reconhecer, diagnosticar e corrigir erros	Mensagens de erro devem ser expressas em linguagem clara (sem códigos) indicando precisamente o problema e construtivamente sugerindo uma solução.
10 - *Help* e documentação	Embora seja melhor um sistema que possa ser usado sem documentação, é necessário prover *help* e documentação. Essas informações devem ser fáceis de encontrar, focalizadas na tarefa do usuário e não muito extensas.

Fonte: (NIELSEN, 1994)

Listas de Heurísticas de Usabilidade

Após identificar as heurísticas violadas, o avaliador deverá definir a localização do problema, o que significa encontrar na interface avaliada onde esse problema está ocorrendo. Localizações segundo Prates e Barbosa (2003):

- Em um único local na interface;
- Em dois ou mais locais na interface, casualmente;
- Na estrutura geral da interface, de forma sistemática;
- Pode ser algo que *não está lá*, ou seja, precisa ser incluído na interface.

Além da localização, a gravidade do problema também deverá ser avaliada, por especialistas com uma combinação dos fatores, conforme Lista de Gravidades Prates e Barbosa (2003):

- Frequência com que o problema ocorre: é um problema comum ou raro?
- Impacto do problema: será fácil ou difícil para os usuários superarem o problema?
- Persistência do problema: é um problema que ocorre apenas uma vez e que os usuários conseguem superar facilmente, ou os usuários serão incomodados pelo problema repetidas vezes?

A gravidade de cada problema deve ser analisada de acordo com a escala disponível abaixo (PRATES; BARBOSA, 2003):

- Não concordo que isto seja um problema (este valor pode resultar da avaliação de um especialista sobre um problema apontado por outro especialista).
- Problema cosmético: não precisa ser consertado a menos que haja tempo extra no projeto.
- Problema pequeno: o conserto deste problema é desejável, mas deve receber baixa prioridade.
- Problema grande: importante de ser consertado; deve receber alta prioridade.
- Catastrófico: é imperativo consertar esse problema antes do lançamento do produto.

6 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Neste capítulo são apresentados conceitos e especificações importantes utilizados na elaboração da documentação do m-tutorial. A Seção 6.1 aborda a metodologia utilizada: RUP e as seis melhores práticas de desenvolvimento de *software* são apresentadas nas subseções seguintes. Já a Seção 6.2 discute conceitos sobre *Checklist* de Inspeção de *Software*. Método de verificação do desenvolvimento de *software*.

6.1 Metodologia Utilizada

Para garantir a produção de *software* de alta qualidade que atenda às necessidades dos usuários, dentro de cronogramas e orçamentos previsíveis, decidiu-se utilizar o Processo Unificado da *Rational* (RUP), por oferecer uma abordagem baseada em disciplinas, atribuindo tarefas e responsabilidades, em uma organização de desenvolvimento de *software*.

As diferentes configurações do RUP possibilitam o suporte de equipes grandes e pequenas, além de técnicas de desenvolvimento menos formais (KRUCHTEN, 2003).

Segundo Kruchten (2003), o aumento da produtividade pode ser obtido pelo uso das seis melhores práticas de desenvolvimento de *software*. Nas subseções seguintes são apresentadas algumas das práticas utilizadas no desenvolvimento do m-tutrial, como, desenvolvimento iterativo e incremental, arquiteturas basedadas em componentes, modelagem visual e verificação contínua da qualidade.

6.1.1 Desenvolvimento Iterativo e Incremental

Em um ciclo de vida iterativo e incremental, o desenvolvimento procede como uma série de iterações que evoluem para o sistema final. O RUP envolve a adoção das seguintes fases baseadas em tempo: concepção, elaboração, construção e transição - cada uma com objetivos específicos. Na fase de concepção em sua primeira iteração, foi levantada uma visão do produto final a partir de um caso de uso básico. Na fase de elaboração primeira iteração, foram planejadas as atividades necessárias, os recursos exigidos e o design da arquitetura a ser desenvolvido. Na fase de construção, um produto completo foi desenvolvido de maneira iterativa e incremental até que estivesse pronto para ser passado aos usuários, já na fase de transição, uma versão beta do sistema foi disponibilizada para testes. No final da transição pode ser iniciado um novo ciclo de desenvolvimento para a evolução do produto, o que envolveria todas as fases

novamente.

6.1.2 Uso de Arquiteturas Baseadas em Componentes

Os componentes são grupos de código coesos, na forma de código fonte ou executável, com interfaces bem definidas e comportamentos que fornecem forte encapsulamento do conteúdo. As arquiteturas baseadas em componentes tendem a reduzir o tamanho efetivo e a complexidade da solução e, portanto, são mais robustas e flexíveis. Um componente representa uma parte física da implementação de um sistema, que inclui código de *software* (fonte, binário ou executável) ou equivalentes (como scripts ou arquivos de comandos).

A arquitetura baseadas em componente do m-tutorial encontra-se especificado na seção diagrama de componentes, onde são mostrados as dependências entre os componentes.

6.1.3 Modelagem Visual (Unified Modeling Language - UML)

Segundo Larman (2007), “a UML é um conjunto de formas de comunicar idéias". É uma maneira de pensar sobre problemas usando modelos organizados em torno de idéias do mundo real. Os modelos são construídos com o uso de notações de *design* gráficas e textuais, semanticamente ricas, para capturar detalhes para o *design* de *software*.

A UML propicia um mecanismo de comunicação para equipes de desenvolvimento, auxiliando na validação de requisitos com o usuário.

Ela é apenas uma linguagem e, portanto, é somente uma parte de um método para desenvolvimento de *software*. É independente de processo, apesar de ser perfeitamente utilizada em processo orientado a casos de usos, centrado na arquitetura, iterativo e incremental (BOGGS; BOGGS, 2002).

A UML não nos indica como devemos desenvolver um *software*, apenas as formas que podem ser utilizadas para representá-lo nos diversos estágios de seu desenvolvimento.

Diagrama de caso de uso - Permite representar as funcionalidades de um sistema e sua interação com o mundo externo. Descreve interações entre o sistema e atores. Um ator é um tipo de usuário ou categoria com papel definido, não faz parte do sistema, mas interage com ele, podendo incluir seres humanos, máquinas, dispositivos ou outros sistemas (CRETU, 1997).

Diagrama de classes - são abstrações de um coletivo de entidades do mundo real. Isto significa que elas representam um modelo comum para um conjunto de entidades semelhantes (STANDZISZ, 2002). O diagrama de classes mostra um conjunto de elementos do modelo declarativos (estáticos), como classes e pacotes, além de seu conteúdo e relaciona-

mentos. Permite a geração automática de código-fonte de programação em determinadas linguagens.

Diagrama de componentes - Um componente representa uma parte física da implementação de um sistema, que inclui código de *software* (fonte, binário ou executável) ou equivalentes (como *scripts* ou arquivos de comandos). Um diagrama de componentes é a relação de todos os itens de *software* que colaboram na confecção do produto final (MATOS, 2002).

Diagrama de implantação - O diagrama de implantação representa como é realizada a distribuição do sistema por meio de nós de *hardware*, infraestrutura de rede, componentes e dependências de *software* e as suas devidas relações de comunicação. Permite avaliar as diferentes situações de implantação do sistema. Exibe a distribuição de componentes do sistema ao longo de recursos que os suportam.

6.1.4 Verificação Contínua da Qualidade

Qualidade é conformidade com requisitos. A verificação contínua da qualidade visa garantir que os requisitos sejam atendidos à medida em que se desenvolvem iterativamente. Aliada ao desenvolvimento iterativo, ela permite eliminar, antecipadamente, inconsistências entre requisitos, projeto e implementação e propicia melhorias no processo de desenvolvimento de *software*.

6.2 Checklist de Inspeção do *Software*

Promover a qualidade de *software* é uma tarefa desafiadora no uso de engenharia de *software*. O termo qualidade é utilizado, internacionalmente, para descrever um processo que garante e demonstra a qualidade dos produtos e serviços oferecidos pelas empresas (KOSCIANSK; SOARES, 2007).

Um sistema de qualidade apropriado deve dar grande ênfase à correção de defeitos e erros para evitar falhas. É muito útil corrigir defeitos e erros logo no início do ciclo de vida de desenvolvimento dos sistemas computadorizados (REZENDE, 2005).

Diversos métodos têm sido propostos para a realização de revisões técnicas e esses métodos se diferenciam principalmente pelo seu grau de formalidade (WEINBERG; FREEDMAN, 1984). Entre as atividades que podem ser utilizadas para verificar a qualidade de um *software* se encontram (MELO; SHULL; TRAVASSOS, 2001):

- Revisões de *software*;
- Testes;

- Padrões e procedimentos formais;
- Controle de mudanças;
- Métricas de *software*;

De acordo com Laitenberger e Debaud (1998), do ponto de vista técnico, uma abordagem para inspeção de *software* é formada pelo tipo do artefato de *software* a ser avaliado, pela forma como a equipe de profissionais será definida para realizar a inspeção, pela técnica de avaliação utilizada e pelo processo de execução seguido.

O principal diferencial de uma abordagem de inspeção está na maneira como ela auxilia o inspetor a encontrar os defeitos. O que determina essa maneira é a técnica de detecção de defeitos utilizada (BARCELOS, 2006). O *checklist* foi a técnica de detecção de defeitos utilizada.

Checklist foi escolhido entre as técnicas existentes por ser mais adequado às características da área de arquitetura de *software* e por permitir minimizar as limitações identificadas nas abordagens de avaliação arquitetural disponíveis na literatura (BARCELOS, 2006). Suas vantagens decorrem da sua simplicidade, facilidade de uso, relação custo-eficácia e apoio para a verificação completa de documentos de *software* (JORGENSEN; MOLOKKEN, 2003).

Com o tempo, *checklists* tornaram-se abrangentes na verificação de desenvolvimento de *software*, associados com as fases do ciclo de vida do *software* (ou seja, listas de verificação de requisitos, *design*, implementação e testes) (BRYKCZYNSKI, 1999).

7 ESPECIFICAÇÃO DO M-TUTORIAL

Neste capítulo são listados os diagramas UML utilizados na elaboração da documentação do m-tutorial. A Seção 7.1 exibe alguns dos diagramas UML elaborados. A subseção 7.1.1 apresenta o diagrama de caso de uso e suas descrições. A subseção 7.1.2 são listadas classes e métodos de maior relevância do m-tutorial. A subseção 7.1.3 apresenta os componentes internos e externos da ferramenta. E a subseção 7.1.4 apresenta o diagrama de implantação do m-tutorial, *hardwares* e componentes são mostrados.

7.1 Diagramas do m-tutorial

O desenvolvimento de uma ferramenta de autoria demanda que desenvolvedores tenham a possibilidade de examinar os requisitos e estudar as funcionalidades a partir de diversos diagramas. Nesta seção são apresentados os modelos UML: Diagrama de caso de uso, Diagrama de classes, Diagrama de componentes e Diagrama de implantação utilizados na elaboração do m-tutorial.

7.1.1 Diagrama de Caso de Uso

Uma rápida descrição de cada ator deve identificar o papel que o ator representa enquanto interagindo com o sistema. O Quadro 7.1.1 descreve as responsabilidades atribuídas aos atores.

Atores	
Professor	Este ator interage por meio de uma interface gráfica, ele consegue criar novos objetos de aprendizagem, por conta própria ou seguindo um assistente.
Aluno	Este ator visualiza conteúdo e faz *download* de arquivos. Ele interage com os OAs criados.
Administrador	Este ator interage por meio de uma página *web*. Ele atualiza os parâmetros do servidor e mantém novos usuários.

Responsabilidades atribuídas aos atores

O Diagrama de caso de uso descreve os requisitos funcionais da ferramenta. Na Figura 7.1 podemos observar o diagrama de caso de uso da ferramenta m-tutorial.

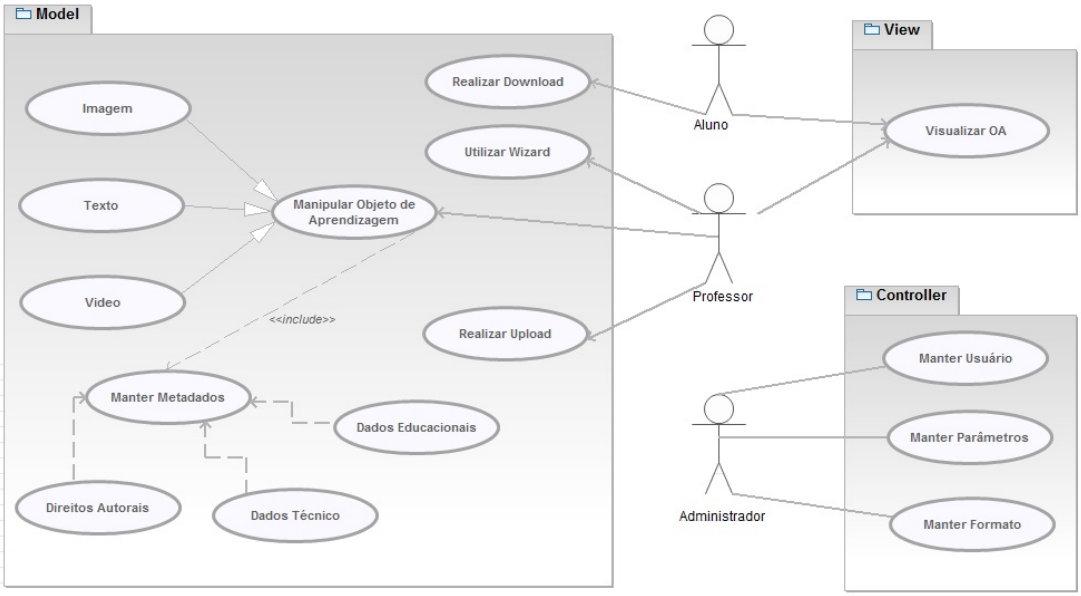


Figura 7.1: Diagrama de Caso de Uso da ferramenta m-tutorial

Os requisitos representados na Figura 7.1 são descritos de forma a apresentar as funcionalidades da ferramenta m-tutorial.

Manipular objetos de Aprendizagem: Este requisito propiciará a criação, a alteração, a exclusão e pesquisa dos objetos de aprendizagem existentes. Esses objetos podem conter textos, imagens ou vídeos.

Manter usuário: o requisito permite manter o perfil de acesso dos usuários e determinar a quantidade de acessos dos usuários.

Visualizar OA: Por este requisito é possível visualizar pelo celular o objeto de aprendizagem.

Manter formato: Este requisito determina o tipo de formato utilizado na importação e exportação de arquivos.

Realizar upload: O requisito citado possibilita ao professor realizar o *upload* do objeto de aprendizagem criado.

Realizar download: O requisito citado possibilita ao aluno fazer o *download* do objeto de aprendizagem em um repositório de dados.

Manter metadados: Através da utilização deste requisito é possível a criação, edição, exclusão e pesquisa por matadados. Oferecendo uma melhor organização e disponibilização dos objetos de aprendizagem em repositórios existentes.

Utilizar *Wizard* : Este requisito permitirá aos usuários do m-tutorial criar objetos de aprendizagem de forma automatizada, seguindo e cumprindo etapas durante a construção do mesmo. Semelhante ao primeiro caso de uso, realizado por meio de um *wizard*.

Manter parâmetro: Este requisito permite ao administrador atualizar parâmetros utilizados pela ferramenta.

7.1.2 Diagrama de Classes

Esta subseção apresenta as principais contribuições dessa ferramenta na forma de um diagrama. São listadas classes e métodos de maior relevância do m-tutorial. Deseja-se com essa ferramenta oferecer a possibilidade das pessoas construírem objetos de aprendizagem do tipo tutorial em qualquer dispositivo móvel independente do lugar, não importando estar conectado.

O Diagrama de classes da Figura 7.2 permite a visualização dos objetos que compõem a estrutura da ferramenta m-tutorial.

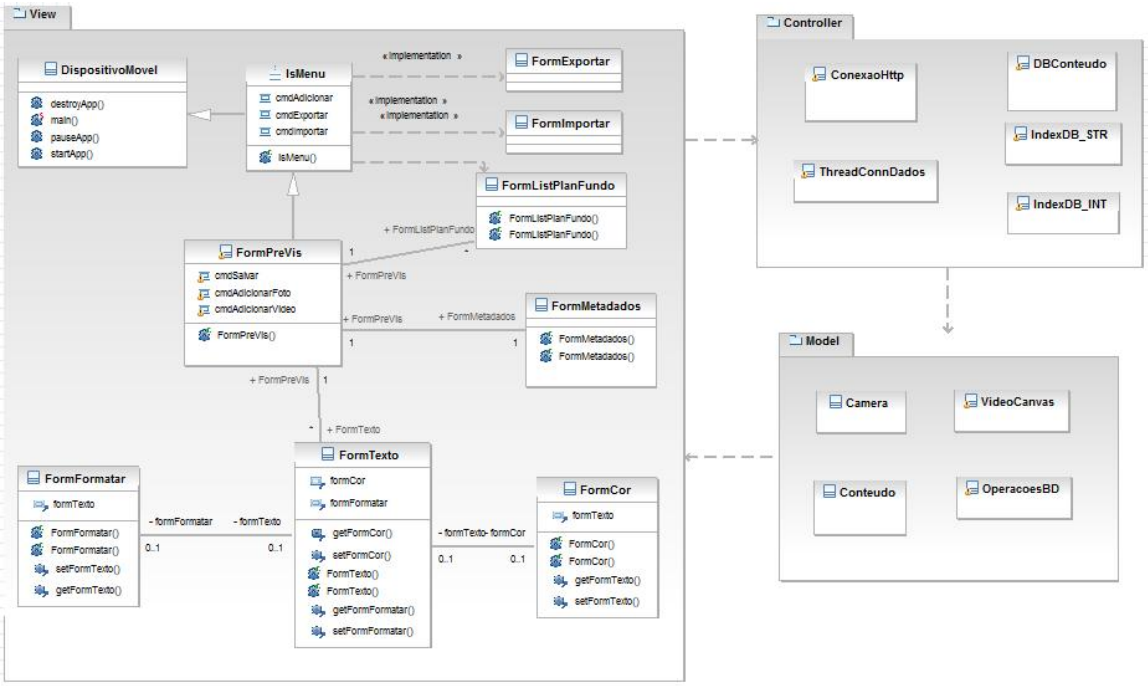


Figura 7.2: Diagrama de classes da ferramenta m-tutorial

A Figura 7.2 mostra um modelo de classes, já validadas pelo *checklist* aplicado na fase de inicialização do ciclo de vida da ferramenta. No pacote *View*, gestos do usuário como cliques ou teclas pressionadas são capturadas. As classes desse pacote estão modeladas de forma que qualquer desenvolvedor consiga reutilizá-las em outras aplicações. A classe principal da ferramenta é chamada de “DispositivoMovel". Por esse objeto é possível navegar em uma lista onde diversos serviços são oferecidos. No pacote *Controller* é apresentado o fluxo da aplicação,

servindo como uma camada intermediária entre a camada *View* e *Model*. Já no pacote *Model* são apresentadas as classes responsáveis por tudo que a aplicação vai fazer, preocupando apenas com o armazenamento, manipulação e geração de dados.

7.1.3 Diagrama de Componentes

O Diagrama de componentes da Figura 7.3 descreve a funcionalidade desempenhada pelos componentes internos e externos da ferramenta. Os componentes internos instanciam as classes concretas com os objetos textos, imagens e vídeos criadas. Já os componentes externos pertencem a biblioteca *Lwuit*.

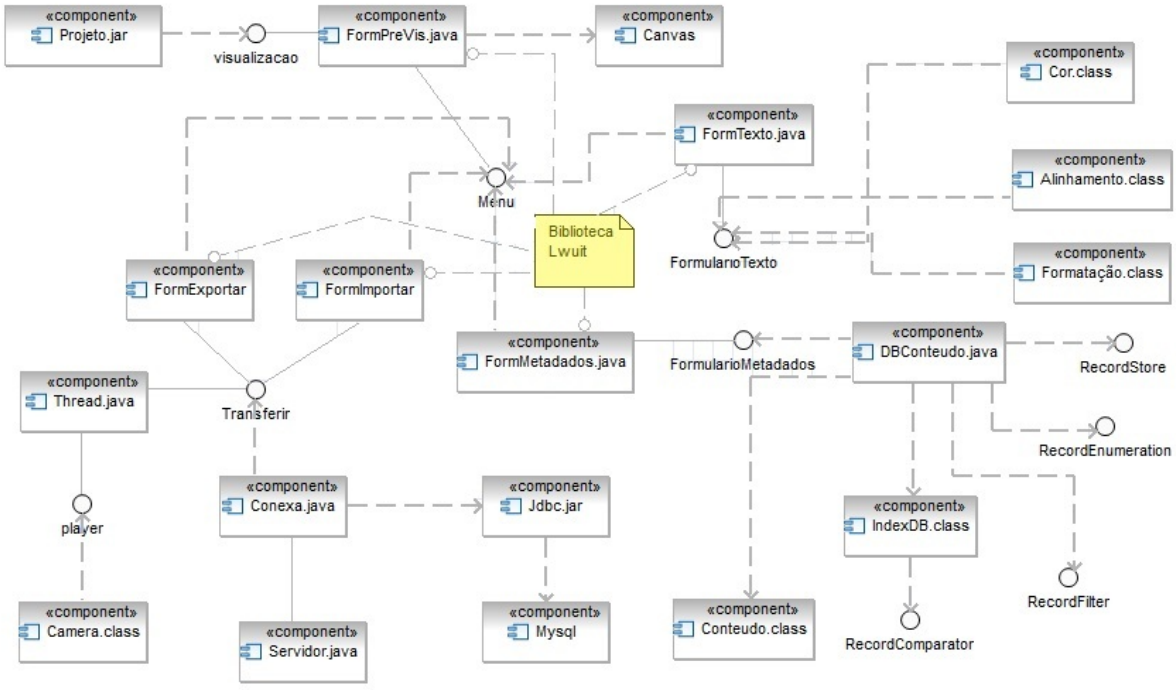


Figura 7.3: Diagrama de Componentes da ferramenta m-tutorial

O Diagrama mostra as dependências entre componentes de *software*. Na Listagem 1 localizada no apêndice B, é apresentado o método getFormPreVis, responsável por listar o conteúdo no formulário de visualização. O componente frmPreVis verifica se existe algum texto ou imagem no formulário de pré-visualização. Caso o formulário esteja vazio, novas imagens e textos serão armazenados para serem pré-visualizados. Caso contrário, o formulário deve ser limpo, excluindo ou armazenando os objetos nele contidos.

O método foto tem a função de iniciar uma *Thread* que executa o método “mostrarCamera", cujo código se encontra na Listagem 5, localizada no apêndice C. O método mostrarCamera verifica se a classe *player* está sendo instanciada por outro método. Se estiver, ele é parado

e fechado. A classe *player* é a responsável por efetuar as ações com os dados multimídia, como apresentar no *display* do celular a câmera do aparelho. Sendo assim, é permitido finalizar outros processos que estejam usando os recursos da câmera do celular. Em seguida um objeto do tipo media*Player* é criado por meio do método *createPlayer* da classe *Manager*. O parâmetro passado no método *createPlayer* define qual a origem dos dados utilizados pelo *player*. Esse parâmetro pode vir da câmera do celular, de um arquivo binário (mp3, mp4, wav), ou de um endereço na web.

Quando o objeto *player* mudar seu estado para “REALIZADO", é iniciada a captura dos dados gerados pela câmera. Em seguida, uma função informa onde serão visualizados os dados gerados pela câmera do dispositivo. Para isso, é instanciado um objeto do tipo *videoControl*. Ele possui a função de estar atualizado constantemente com o conteúdo de vídeo gerado pela câmera digital do celular. Já o componente videoCanvas é o responsável por apresentar o conteúdo recebido pelo *videoControl* no *display* do celular. A Listagem 6, que se encontra no apêndice C, apresenta o método “*initControls*", o mais importante da classe “VideoCanvas".

A classe videoCanvas é uma extensão de Canvas, isso significa que todas as características que essa classe possui, foram herdadas de Canvas. A principal delas é a possibilidade de desenhar pixel a pixel a tela do dispositivo. Esta classe possui um método chamado *initControls*. Esse método recebe por parâmetro os objetos *videoControl* e *player*, os quais são armazenados em variáveis internas da classe, além de atribuir os valores correspondentes à largura e à altura do *display* em *pixels*. A principal importância desse método é definir que a visualização da câmera do dispositivo será em um Canvas. O método *initDisplayMode* recebe o modo de operação e mostra o conteúdo da câmera em um item de um formulário.

Já o método listarConteudo do componente DBConteudo é o responsável no armazenamento dos dados em um RMS. O mecanismo de armazenamento do RMS é implementado como uma coleção de registros onde cada registro é organizado como um array de bytes. A Listagem 2, localizada no apêndice B, apresenta o método responsável por realizar a persistência dos dados. Uma instância da classe *Vector* é inicializada, chamada de vetorConteudo. Esse objeto implementa um array redimensionável de objetos que podem ser acessados via índices. O tamanho desse vetor pode ser aumentado ou reduzido sempre que necessário. Uma nova coleção de registros é gerada pela instanciação da classe *RecordStore*. Fluxos de saída e entrada de dados são inicializados pelas instâncias *strmInBytes* e *strmInDataTypes*. A navegação pelos registros do *RecordStore* são feitas pela classe *RecordEnumeration*. Com essa classe é possível navegar nos registros utilizando um *RecordFilter* para localizar ou um *RecordComparator* para ordenar.

Para a comunicação com o *web service* utilizou-se o protocolo de transferência de dados HTTP construído sobre o protocolo *Transmission Control Protocol* (TCP)/*Internet Protocol* (IP) utilizado na web. A transferência dos objetos de aprendizagem do dispositivo móvel para

um servidor Web se deu por meio da interface transferir, ela instancia um novo formulário que adiciona um componente *textField* e dois do tipo *command*. Por esse formulário é possível importar ou exportar arquivos. A Listagem 3, localizada no apêndice B, apresenta o método utilizado na transferência de arquivos.

Em aplicações que envolvem serviços web, uma requisição *HttpConnection* deve operar numa *thread* separada, caso contrário a aplicação ficará interrompida enquanto sua aplicação estiver se comunicando com o servidor. Por esta *thread* será possível inclusive disparar várias conexões paralelas e controlar de qual conexão se trata. A Listagem 4, localizada no apêndice B, apresenta a *thread* de comunicação de dados. Essa *thread* estabelece uma conexão HTTP com a URL, verificado o status desta conexão. Caso a conexão seja bem sucedida, os resultados são armazenados.

O método setProcessPOST do componente conexão é o responsável por todo o processo de comunicação com o servidor, como por exemplo, controle de erros, timeout, leitura e envio de dados entre outros. No final da requisição, esse método envia para o usuário um aviso de que a requisição terminou. A Listagem **??**, localizada no apêndice B, apresenta o método utilizado na conexão entre o dispositivo móvel e o servidor *web*. Esse método utiliza a classe *StringBuffer* criada para melhorar a velocidade na concatenação de novas *strings*. Um objeto da classe StringBuffer é instanciado quando mais de uma thread são usadas por vez. Ainda é associado um objeto da classe *DataOutputStream*, permitindo que o aplicativo escreva tipos de dados primitivos Java em um fluxo de saída de uma forma portátil e um objeto da classe *DataInputStream*, ao fluxo de entrada de dados do *servlet*. Se a mensagem que chegar ao servidor for “informarDados", o servlet envia informação para o celular utilizando o método *writeUTF*.

As classes e métodos demonstrados nesta seção poderão auxiliar o desenvolvimento de diversas aplicações para dispositivos móveis. Essas classes foram elaboradas com o intuito de serem estendidas em aplicações futuras.

7.1.4 Diagrama de Implantação

A utilização de *frameworks* e bibliotecas em projetos de *software* é uma alternativa bastante comum atualmente. O m-tutorial baseia sua arquitetura em componentes. Com intuito de distribuir as responsabilidades de cada parte da aplicação, escolheu-se utilizar o padrão de projeto *Model-View-Controler* MVC, por resolver problemas de interação entre usuário e sistemas, de forma que a interação em si seja isolada das regras de negócio.

Neste trabalho, o m-tutorial utiliza protocolos baseados em padrões XML que permite a geração automática tanto do código cliente quanto do código do servidor. A comunicação distribuída ocorre por meio de mensagens XML, formatadas e encapsuladas segundo o protocolo

SOAP e transportadas via HTTP.

A conexão do servidor web com um sistema gerenciador de banco de dados realiza-se por meio de um *drive* que trata de todo o processo de comunicação.

A Figura 7.4 ilustra o Diagrama de implantação da ferramenta m-tutorial. Apresenta também como a aplicação cliente pode utilizar-se de periféricos do celular como câmera digital, microfone e futuramente impressora.

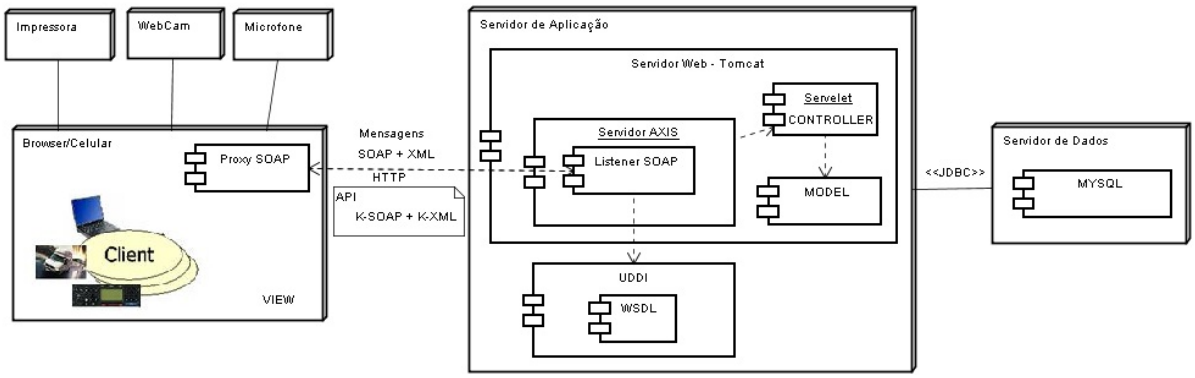


Figura 7.4: Diagrama de Implantação da ferramenta m-tutorial

Para desenvolvimento do *web service* utilizou-se o *Apache Axis* que será executado sobre o *apache Tomcat*. O Axis baseia-se em arquivos *Java Web Service* (JWS) para criar os arquivos de definição WSDL. Para consumir esse *Web Service* instalado no *Apache Axis*, usou-se um aplicativo cliente utilizando a tecnologia Java ME, que se conecta ao servidor utilizando as APIs KXML e KSOAP. Essas APIs permitem aos dispositivos móveis acessarem web services baseados em XML. Esse funcionamento se dá pela codificação, serialização e envio dos métodos e seus argumentos. Esses métodos são recebidos, decodificados e desserializados de maneira transparente pelo servidor.

8 AVALIAÇÃO DA FERRAMENTA PROPOSTA

Neste capítulo, apresentamos os resultados obtidos no desenvolvimento do projeto da dissertação. A Seção 8.1 aborda um estudo de caso utilizado na validação do objeto de aprendizagem criado. A Seção 8.2 apresenta a abordagem proposta para inspecionar documentos arquiteturais através do uso de *checklist*. A Seção 8.3 apresenta à Avaliação Heurística. E a Seção 8.4, um comparativo entre ferramentas para *m-learning*.

8.1 Estudo de Caso

Para desenvolver a aplicação deste projeto foi utilizada a IDE NetBeans 6.7 *Full*, a qual possui o *plug-in Mobility Pack* usado no desenvolvimento visual de aplicações para celular.

A ferramenta implementada foi desenvolvida para criar objeto de aprendizagem de forma dinâmica e interativa, propiciando a criação de tutoriais a qualquer hora, lugar e dispositivo móvel, por exemplo, durante uma viagem, na ida para o trabalho, na fila do banco, entre outros. O aplicativo é composto por formulários intuitivos que seguem uma ordem cronológica para o desenvolvimento de tutoriais. Toda a interação é feita com a utilização das teclas do telefone celular.

Ao iniciar o protótipo, o usuário é direcionado para o formulário principal da ferramenta. Nessa tela será visualizada uma lista com as principais funcionalidades do protótipo, conforme ilustra a Figura 8.1. A mesma funcionalidade na Web é demonstrada no apêndice D.

Figura 8.1: Tela Inicial da ferramenta m-tutorial

Ao escolher a opção criar um novo tutorial pela opção *wizard*, o usuário é encaminhado a uma tela de instruções que oferece orientação para escolha de um plano de fundo. Em seguida é solicitado ao usuário a inserção de um novo texto ou imagem. Caso seja escolhido adicionar texto, um novo formulário é exibido, oferecendo a inserção e edição do texto, com as opções: fonte, estilo, tamanho, sublinhado, cor do texto e alinhamento. Mas se a escolha for pela inserção de uma imagem, o usuário poderá selecionar a imagem desejada em uma lista contida em seu aparelho ou, se o dispositivo possuir câmera, uma nova foto poderá ser tirada e adicionada em seu tutorial. O processo de criação de objetos de aprendizagem pelo modo convencional é semelhante a forma *wizard*. É diferenciado pela forma de o usuário escolher a ordem de criação do tutorial sem a necessidade de uma sequência pré definida.

Após a formatação do texto ou imagem, tendo finalizado o tutorial, o usuário salvará o objeto de aprendizagem. Para finalizar o processo, deverá preencher um formulário de metadados informando os campos: título, autor, área, nível, público, idioma e versão.

Com os campos do formulário-metadados preenchidos, o conteúdo dos campos é adicionado em uma lista armazenada na memória do dispositivo móvel. Se o RMS estiver vazio (primeiro acesso), a lista inicia sem dados. Nos demais acessos serão apresentados os registros armazenados no RMS.

Após apresentada a lista principal, o usuário terá a opção de incluir um novo tutorial por meio da opção adicionar, na qual será chamado um novo formulário. A Figura 8.2 apresenta um exemplo produzido com a ferramenta m-tutorial capturadas a partir do emulador *Wireless Toolkit*.

Figura 8.2: a) Tela de Seleção de Texto ou Imagem. b) Uma Visão da Inserção do Texto. c) Tela de Formatação do Texto. d) Uma Visão da Escolha de Cor do Texto. e) Tela de Alinhamento do Texto. f) Uma visão Final do Tutorial. g) Tela de Inserção dos Metadados.

O produto do m-tutorial em si é um arquivo HTML, pode ser visualizado em qualquer dispositivo móvel. Comparando aos OAs genéricos, o m-tutorial se diferencia pela possibilidade de criar tutoriais adaptáveis a qualquer tamanho de tela e dispositivo.

8.1.1 Testes em Ambiente Real

Os testes em ambiente real da ferramenta m-tutorial aconteceu em duas etapas. Na primeira etapa o tutorial foi elaborado por meio do aparelho celular F029. o celular utilizado possui uma grande diversidade de funções e um design moderno. Seu sistema operacional é o *Symbian OS*. Ele permite a executação de inúmeros aplicativos e tudo mais simultaneamente graças a sua qualidade e componentes, além de oferecer acesso a qualquer tipo de conteúdo ou aplicativos com apenas um toque na tela. A Figura 8.3 ilustra a criação de tutoriais fora do simulador.

Figura 8.3: a) Tela inicializar o m-tutorial. b) Tela Principal com a função adicionar objeto de aprendizagem. c) Tela informativa plano de fundo.d) Uma Visão da Escolha de plano de fundo. e) Tela de escolha de texto, imagem ou salvar. f) Uma visão Final do Tutorial.

Na segunda etapa dos testes, o m-tutorial na web foi avaliado pelo *Smartphone Nokia* E61. Este aparelho foi escolhido por possuir tela colorida de matriz ativa em formato paisagem (320 x 240 píxels), WLAN integrado (IEEE 802.11g, 802.11e, 802.11i), integração com diversos navegadores, serviços multimídia, entre outros. Na Figura 8.4 é apresentada a página inicial com a funcionalidade de criação de objetos de aprendizagem, podendo escolher a inserção de um plano de fundo, texto, imagem ou vídeo. A opção de *upload, download* e pesquisa de objetos de aprendizagem também é demonstrado.

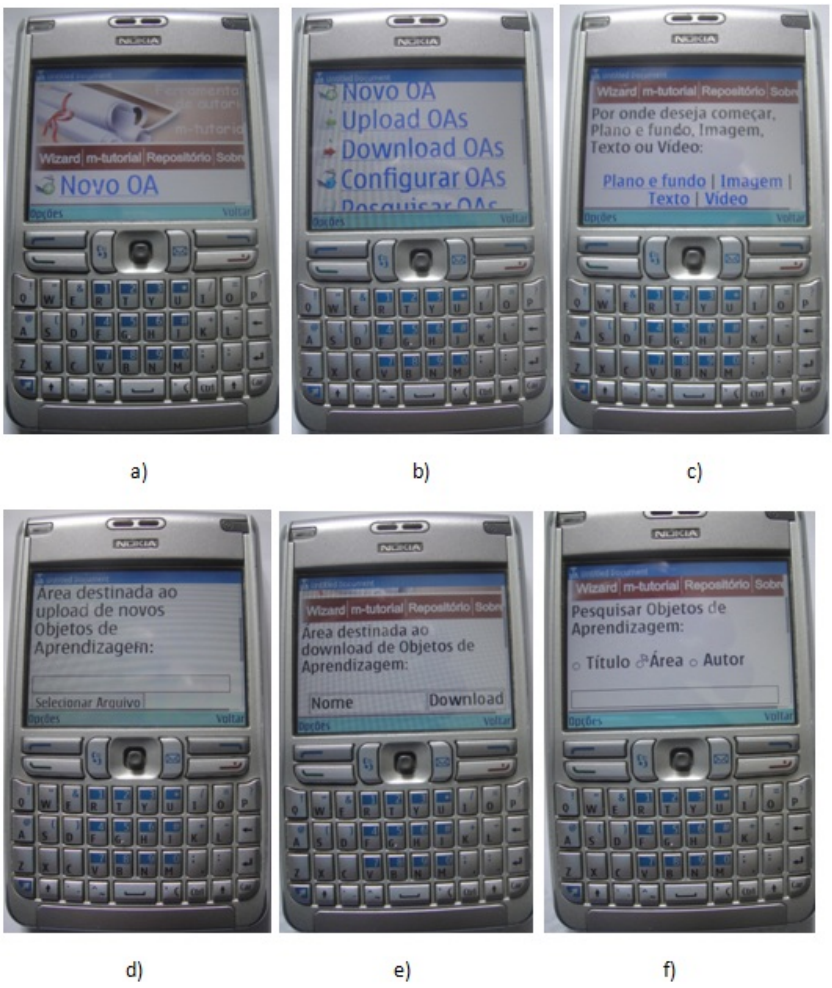


Figura 8.4: a) Tela inicial da ferramenta m-tutorial. b) Tela inicial da ferramenta m-tutorial. c) Tela de seleção de plano de fundo, texto, imagem ou vídeo. d) Tela de *upload* de objetos de aprendizagem. e) Tela de *download* de objetos de aprendizagem. f) Pesquisar objetos de aprendizagem.

8.2 Análise dos Dados do *checklist*

Com base em informações obtidas a partir de experiências educativas oferecidas em extensões dos cursos de engenharia de *software* das universidades do Texas e Nevada nos Estados Unidos, nos anos de 2004 a 2005, (COOPER; LIDDLE; DASCALU, 2005) e de acordo com o *checklist* de inspeção de pré-teste utilizado por Silva e Crespo (2008) elaborou-se uma lista de verificação para avaliar as características específicas dos documentos arquiteturais, representadas por meio da notação UML. O *checklist* elaborado para inspeção é composto por 38 itens de avaliação que visam verificar a inspeção de documentos arquiteturais representados por diagramas UML. Com a forma de avaliar a clareza dos itens, as opções “Sim” e “Não” representam os itens aplicáveis e não aplicáveis. Os itens aplicáveis são aqueles compatíveis com os requisitos da ferramenta. Já os não aplicáveis, são itens nos quais a conformidade com os requisitos sofreram alteração. Além destas opções, o *checklist* disponibiliza uma terceira opção, “NA - Não Avaliado", que deve ser marcada caso o participante identifique que o item não pode ser

usado para avaliar o documento arquitetural que lhe fora passado. O Quadro 8.2 exibe a lista de verificação da avaliação do plano de garantia da qualidade de *software* aplicado.

Itens de avaliação da consistência das representações entre os diagramas				
Nº	Modelo de caso de uso	SIM	NÃO	NA
1	Cada caso de uso representa um requisito funcional?			
2	Existe rastreabilidade entre requisitos identificados e necessidades?			
3	Cada requisito funcional possui uma especificação detalhada?			
4	As especificações comtemplam os fluxos básicos, alternativos e exceção?			
5	As especificações contemplam pré-requisitos e pós-condições?			
6	Existe um modelo de casos de uso para cada subsistema identificado?			
7	Todos os casos de uso estão adequadamente descritos?			
8	Todo caso de uso gráfico possui uma descrição textual?			
9	Todos os atores estão adequadamente representados?			
10	Cada caso uso é iniciado por um ou mais atores?			
11	A multiplicidade entre os casos de uso é descrito (opcional)?			
12	Os atores representam pessoas, *hardware* ou sistemas externos?			
13	Entrada de dados do usuário ou a saída do sistema estão claramente descritos?			
Nº	Modelo de classes	SIM	NÃO	NA
14	Todas as classes possuem nome e descrição adequados?			
15	Todos as categorias de requisitos não funcionais foram levantadas?			
16	Cada requisito não funcional possui uma especificação detalhada?			
17	Todas as classes do modelo foram implementadas?			
18	Todos os métodos de cada classe foram implementados?			
19	Todos os atributos de cada classe foram implementados?			
Nº	Modelo de componentes	SIM	NÃO	NA
20	Os pacotes agrupam componentes com as mesmas características?			
21	Cada componente agrupa classes de única camada: usuário, negócios, dados?			
22	Todas as dependências de componentes foram estabelecidas?			
Nº	Modelo de Arquitetura	SIM	NÃO	NA
23	O software e hardware necessários são especificados?			
24	Foram especificadas de forma apropriada as funcionalidades de interação entre hardware, software com o sistema?			
25	Existem parâmetros que modificam a funcionalidade da aplicação?			
26	A ferramenta possui independência do banco de dados?			
27	A ferramenta possui independência do sistema operacional?			
28	Os mecanismos arquiteturais (padrões) estão definidos claramente no design de forma que sejam reutilizáveis e compreensíveis??			

Itens de avaliação da consistência das representações entre os diagramas				
Nº	Detalhes de design de UI	SIM	NÃO	NA
29	Ilustrações ou animações são usadas para completar as explicações do texto?			
30	As telas são compatíveis com o padrão do ambiente?			
31	As denominações dos títulos estão de acordo com o que eles representam?			
32	Os títulos dos menus são distintos entre si?			
33	O sistema possui comandos para desfazer e refazer?			
34	sistema emite sinais sonoros quando ocorrem problemas?			
35	Os usuários têm a possibilidade de criar novos comandos no sistema?			
36	O usuário pode terminar um processo a qualquer instante?			
37	O sistema fornece *feedback* para todas as ações do usuário?			
38	Os títulos de telas, janelas e caixas de diálogo estão no alto, centrados ou justificados à esquerda?			

Checklist de Inspeção do *Software*

8.2.1 Resultados Obtidos

Durante o planejamento desse estudo, definiu-se avaliar os requisitos funcionais e não funcionais dos diagramas UML que compõem essa ferramenta. Elaborou-se um *checklist* de inspeção com base nas representações dos diagramas. Detalhes de *design* e consistência das representações foram inspecionadas. A seguir são descritos os resultados obtidos para cada um desses questionamentos.

Visto que o estudo foi realizado em um ambiente acadêmico, os inspetores escolhidos foram estudantes de pós-graduação com alguma experiência em projeto arquitetural. Contudo, devido ao interesse no assunto, o número de inspetores se limitou a três indivíduos.

Por não estar no objetivo do estudo observar os participantes durante a execução da inspeção, a lista de verificação foi conduzida em apenas uma sessão, ou seja, os participantes realizaram a inspeção no mesmo tempo e no mesmo lugar. Para cada participante, foi distribuído o *checklist* de avaliação, o documento arquitetural, o documento de requisitos e acesso ao protótipo para avaliação dos detalhes de *design*.

Se um participante tiver identificado um item aplicável como não aplicável ou vice-versa, ou então se o item identificar um defeito, mas ele não tiver marcado de tal forma, considera-se que o participante não entendeu corretamente o propósito do item, sendo considerado como item não avaliado. O Gráfico 8.2.1 mostra os itens não avaliados pelos participantes.

Itens Não Avaliados pelos Participantes

Valendo-se dos dados coletados, identificou-se, para cada item, a quantidade de participantes que atribuíram erroneamente valores para eles. Os itens 11, 21, 32 não foram compreendidos por pelo menos 2 dos participantes e por isso sofreram melhorias. Já os itens 2, 17 e 22 foram marcados por apresentar problemas, como falta de clareza ou não serem, realmente, aplicáveis a uma avaliação arquitetural. Os itens duvidosos passaram por um maior nível de detalhamento e atualizações foram necessárias.

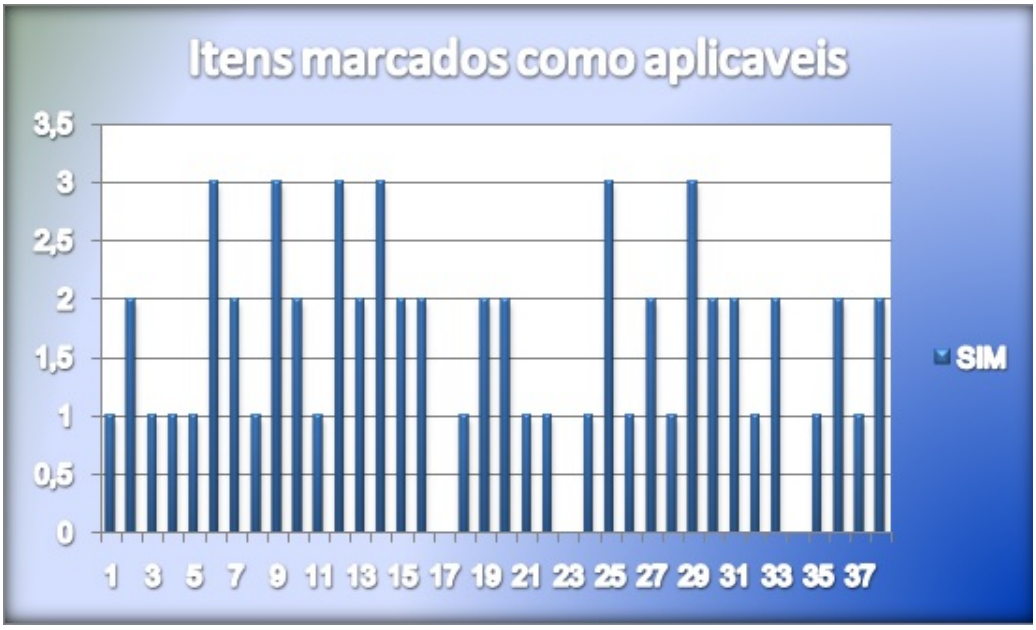
Dos dados levantados, o Gráfico 8.2.1 apresenta os itens marcados como não aplicáveis aos requisitos dos diagramas.

Itens Marcados como Não Aplicáveis pelos Participantes

Nos itens marcados como não aplicáveis, os números 23 e 34 são de baixa relevância para funcionalidade da ferramenta, por isso ainda não haviam sido tratados no momento da inspeção. Os itens 1, 3, 4, 5, 17, 18, 26, 35 e 37 passaram novamente por uma verificação e sofreram as modificações informadas pelos participantes. Já os itens 7, 10, 13, 15, 16, 18,19, 20, 22, 27, 30, 31, 33, 36 e 38 foram avaliados como itens aplicáveis pela maioria dos avaliadores. Os resultados foram significativos e por meio dessa inspeção a qualidade dos diagramas melhorou, erros e inconsistências nos diagramas foram corrigidos.

Verificou-se um outro fato importante ao analisar os resultados com relação à detecção de defeitos, é que a maioria dos diagramas já se encontravam compatíveis aos requisitos funcionais e não funcionais da ferramenta.

O Gráfico 8.2.1 apresenta os itens marcados como aplicáveis pelos participantes.



Itens Marcados como Aplicáveis pelos Participantes

Os itens 6, 9, 12, 14, 25 e 29 foram marcados por todos os participantes como itens aplicaveis, nenhuma inconsistência na avaliação foi encontrada. Já os itens 2, 7, 10, 13, 15, 16, 19, 20, 27, 30, 31, 33, 36 e 37 foram marcados como aplicáveis por pelo menos 2 dos participantes, portanto, sem inconsistências importantes. Já os demais itens foram validados como itens não aplicados, tendo que passar por uma restruturação.

Contudo, o objetivo dessa inspeção é obter algum indício sobre a clareza dos itens, para que sejam melhorados, e os resultados obtidos por essa caracterização permitem a identificação desses indícios. O Quadro 8.2.1 apresenta o resultado dos itens do *checklist* referente aos quadros 8 e 9, 34% dos itens apresentaram como não aplicáveis as especificações da ferramenta. Grande parte dos itens foram reavaliados e readequados a conformidade do m-tutorial. A mai-

oria dos itens, que representa 66%, apresentaram-se aplicáveis as especificações. Nesses itens não foram encontrados erros nos modelos UML e detalhes de *design*.

Item	Identifica Defeitos	Item	Identifica Defeitos
1	SIM	20	NÃO
2	NÃO	21	SIM
3	SIM	22	NÃO
4	SIM	23	SIM
5	SIM	24	NÃO
6	NÃO	25	NÃO
7	NÃO	26	SIM
8	NÃO	27	NÃO
9	NÃO	28	NÃO
10	NÃO	29	NÃO
11	SIM	30	NÃO
12	NÃO	31	NÃO
13	NÃO	32	NÃO
14	NÃO	33	NÃO
15	NÃO	34	SIM
16	NÃO	35	SIM
17	SIM	36	NÃO
18	SIM	37	SIM
19	NÃO	38	NÃO

Resultados dos itens do *checklist*

8.3 Comparativo entre Ferramentas para *m-learning*

Com o propósito de fazer um levantamento entre as principais ferramentas para dispositivos móveis, foram estabelecidos critérios para a execução da comparação entre ferramentas para *m-learning*. Os critérios adotados são: nível de reutilização, quantidade de documentação disponível, boa usabilidade, se seus componentes são modulares, segurança, eficiência e custos. Nas principais características que um *framework* deve possuir. De acordo com os requisitos levantados na construção do m-tutorial, decidiu-se fazer também um comparativo entre as ferramentas para *m-learning* no contexto educacional.

As ferramentas escolhidas para este comparativo são: m-tutorial, *Mobile Author* e *mobiSiteGalore*. Os Quadros 8.3 e 8.3 apresentam os resultados alcançados com a comparação. A justificativa se encontra nas próximas subseções.

QUADRO COMPARATIVO DAS CARACTERISTICAS ENTRE AS FERRAMENTAS							
Framework	Reusável	Boa Documentação	Boa Usabilidade	Modularidade	Seguro	Eficiência	Custos
m-tutorial	Reusável	Bem documentado	Boa usabilidade	Modular	Seguro	Eficiente	Gratuito
Mobile Author	Não Reusável	Bem documentado	Boa usabilidade	Modular	Seguro	Eficiente	Pago
mobiSiteGalore	Não Reusável	Bem documentado	Boa usabilidade	Modular	Seguro	Eficiente	Pago

Quadro Comparativo das Caracteristicas entre as Ferramentas

QUADRO COMPARATIVO ENTRE AS FERRAMENTAS NO CONTEXTO EDUCACIONAL			
Framework	Mobilidade da Aprendizagem	Aprendizagem Colaborativa	Recursos dos Dispositivos
m-tutorial	Boa Mobilidade	Nada Colaborativo	Utiliza
Mobile Author	Boa Mobilidade	Nada Colaborativo	Não Utiliza
mobiSiteGalore	Boa Mobilidade	Nada Colaborativo	Não Utiliza

Quadro Comparativo entre as Ferramentas no Contexto Educacional

8.3.1 Reutilização

No comparativo feito entre as três ferramentas pode-se perceber que o m-tutorial foi elaborado pela instanciação de classes concretas por meio dos objetos textos, imagens e vídeos criados. Ele permite a geração de código seguindo uma arquitetura de referência. Já o *Mobile Author* e *mobiSiteGalore* são aplicações proprietárias, suas classes são protegidas, por isso não foi possível avaliar esse item.

8.3.2 Boa Documentação

Uma boa documentação é fundamental para o sucesso de uma ferramenta. Realizou-se um levantamento da documentação oferecida pelas ferramentas comparadas. O m-tutorial mais uma vez se destacou. Sua documentação faz parte do protótipo, embasamento teórico e modelos UML são oferecidos a desenvolvedores de aplicações. A ferramenta *mobiSiteGalore* baseia sua documentação voltada para o usuário, oferece suporte técnico disponível a toda a hora por meio de e-mail. Nenhuma documentação de classes é oferecida. Já o *Mobile Author* possui uma documentação mais detalhada que o *mobiSiteGalore*, como formulário de retorno, além de um programa de treinamento. Mas também sua documentação é voltada para interface com o usuário.

8.3.3 Boa Usabilidade

As ferramentas avaliadas são voltados para criação de objetos de aprendizagem para dispositivos móveis. A facilidade de uso foi o destaque das ferramentas. Todas as três ferramentas

foram testadas na prática. Para utilizar algum das ferramentas avaliadas não é necessário nenhum conhecimento técnico, habilidades de programação ou experiência em *web design*. Com *mobiSiteGalore* pode-se escolher entre uma variedade de templates atraentes para seu site móvel. Uma vez escolhido, é possível customizar completamente as cores, fontes e *layout* da página do seu *web* site em minutos. No *Mobile Author* a criação de OAs são feitas com poucos cliques, permite-se que criadores de conteúdo desenvolvam de forma fácil e eficaz conteúdos *m-learning* para uma ampla variedade de necessidades de aprendizagem. Já o m-tutorial foi projetado com intuito de possibilitar a construção de OAs do tipo tutorial para dispositivos móveis, esses OAs são criados no próprio aparelho celular de forma simples, depois podem ser exportados para um servidor *web* por meio do requisito upload.

8.3.4 Modularidade

A modularidade de uma ferramenta se baseia na combinação de componentes. Um módulo, normalmente, utiliza os serviços de outros módulos não sendo, portanto, considerado um sistema independente. Verificou-se que todos as três ferramentas possuem características modulares. O *Mobile Author*, por exemplo, possibilita adicionar módulos como imagens, som mp3, vídeo, texto, listas numeradas e listas com marcadores. A nova versão do *mobiSiteGalore* permite incorporar em seu *web* site vídeos do *YouTube, Google Maps*, imagens, áudio e arquivos. Já o m-tutorial combina módulos manipuláveis com tarefas específicas como, por exemplo, os objetos imagens, áudio e vídeo. São manipulados pelo requisito manipular objetos de aprendizagem, mas realizam operações específicas.

8.3.5 Segurança

Para que ferramentas sejam consideradas seguras, são projetadas de modo a garantir a segurança de quem programa e, principalmente, de quem usa o que foi feito a partir delas. O m-tutorial possui classes de regras de negocio como ponte para a camada de banco de dados e permissões de acesso a código que representam o acesso a um recurso protegido ou a capacidade de executar uma operação protegida. Só código com confiança suficiente pode atender a recursos protegidos do sistema. O *mobiSiteGalore* foi testado e certificado para ser extremamente confiável. Todos os processos, desenvolvimento e suporte são certificados para serem totalmente compatíveis com as normas de qualidade ISO 9001:2000. Já o *Mobile Author* em nenhum momento apresenta informações sobre segurança, suas classes são de uso particular, imposibilitando alguns testes.

8.3.6 Eficiência

Para medir a eficiência das ferramentas levou-se em conta o tempo gasto no processo de criação do objeto de aprendizagem. No *mobiSiteGalore*, o tempo médio de 54 minutos é o suficiente para criar e visualizar um site Wap em um celular, o objeto de aprendizagem é suportado por qualquer aparelho com um navegador móvel (MOBISITEGALORE, 2010). No m-tutorial, o tempo de criação de um objeto de aprendizagem varia de acordo com a capacidade de armazenamento dos dispositivos móveis e das necessidades do usuário. Os tutoriais são elaborados podendo conter textos, imagens ou vídeos. O tempo médio na criação de um tutorial varia. Um tutorial simples gasta uma média de 8 minutos até que possa ser visualizado em um dispositivo móvel. Já o *Mobile Author* permite criar *Checklist*, questionários, testes e enquetes na média de 10 minutos, dependendo do tamanho do objeto criado.

8.3.7 Custos

A ferramenta m-tutorial foi desenvolvida por meio da licença "*general public license version* 2"(GPLv2), no qual a obrigação de restringir a liberdade de outros implica não poder distribuir o programa. A ferramenta *mobiSiteGalore* não oferece uma versão *Premium* ou paga. Todos os recursos liberados até agora são de uso gratuito. Porém, nenhum código ou API para interagir com a ferramenta estão disponíveis. Já a versão gratuita do *Mobile Author* inclui apenas ferramentas de criação de conteúdo. Para adquirir a versão completa é necessário o pagamento de uma licença comercial.

8.3.8 Mobilidade na Aprendizagem

Todas as ferramentas comparadas são excelentes para utilização da mobilidade com aprendizagem. O m-tutorial proporciona ao usuário a criação de objeto de aprendizado do tipo tutorial estando em modo *off-line* a qualquer hora em qualquer dispositivo móvel. Seu conteúdo é armazenado internamente em uma classe RMS e enviada para um servidor web quando possuir uma conexão estável. No *Mobile Author*, objetos de aprendizagem com recursos multimídia são criados a partir de um PC, podendo ser executados em qualquer tipo de aparelho móvel. Já *mobiSiteGalore* é uma aplicação baseada em *web service*. Atua na construção de objetos de aprendizagem do tipo *web* site. Esse objeto de aprendizado pode ser criado a partir de um celular, com a objeção de ter que sempre estar conectado para conseguir criar um objeto.

8.3.9 Aprendizagem Colaborativa

Aprendizagem colaborativa é a situação na qual duas ou mais pessoas aprendem ou tentam aprender, conjuntamente, alguma coisa (IRALA; TORRES, 2004). Das três ferramentas

avaliadas, nenhuma delas se enquadram nesse contexto de aprendizagem.

8.3.10 Recursos dos Dispositivos Móveis

Este item validou se alguma das ferramentas comparadas se utilizam dos recursos oferecidos pelos aparelhos celular, como filmar, gravar áudio, tirar fotos, transferir dados por *blueto-oth* entre outros. O m-tutorial utiliza um método capaz de manipular a câmera dos dispositivos móveis, possibilitando incluir recursos como imagem e vídeo em seus tutoriais. Já as outras duas ferramentas também utilizam de recursos multimídias, porém não possibilitam utilizar recursos do celular na criação de objeto de aprendizagem.

8.4 Avaliação Heurística

Na subseção 5.2.1 especificou-se o método de inspeção de usabilidade representado por meio da avaliação heurística, utilizada para avaliar o m-tutorial.

A avaliação heurística foi realizada por dois estudantes do mestrado de ciência da computação sendo um da Universidade Federal de Viçosa e outro da PUC Minas e um professor da disciplina de Interface Usuário Máquinas da Universidade Presidente Antônio Carlos.

A proposta da avaliação heurística foi inspecionar o protótipo do *framework* m-tutorial realizada por meio de sessões curtas de avaliação individual, nas quais cada inspetor recebeu as seguintes informações:

- Inspecionar somente a opção no menu principal “adicionar";
- Julgar a conformidade da interface de acordo com o conjunto de princípios (“heurísticas") de usabilidade disponibilizados em uma planilha;
- Anotar os problemas encontrados e sua localização;
- Julgar a gravidade destes problemas;
- Gerar um relatório individual com o resultado de sua avaliação.

8.4.1 Principais Resultados da Avaliação Heurística.

O Quadro 8.4.1 apresenta os principais resultados da Avaliação Heurística. Separados por problemas, heurística violada, gravidade e contribuição:

Problemas	Heurística Violada	Gravidade	Contribuição
Não existe a possibilidade do usuário voltar ao menu principal.	Controle do usuário e liberdade.	Grande	Permitir ao usuário realizar escolhas diferentes do que realmente queria.
O usuário terá dificuldades de compreender que para digitar as informações precisa clicar em T9.	Compatibilidade do sistema com o mundo real.	Catastrófico	Permitir que o usuário não desista de utilizar o sistema por não entender como digitar as informações.
O usuário não consegue cancelar uma imagem adicionada.	Controle do usuário e liberdade.	Grande	Permitir ao usuário a possibilidade de cancelar uma imagem já adicionada.

Principais Resultados Avaliação Heurística

Os resultados obtidos com a avaliação são visualizados nas figuras 10, 11, 12, 13, 14 e 15 e descritos nos quadros 12, 13, 14, 15, 16 e 17.

Figura 8.5: Tela Plano de Fundo da ferramenta m-tutorial

Problema: Caso o usuário queira voltar para o menu principal e escolher uma outra opção não existe a possibilidade.

Heurística Violada: Controle do usuário e liberdade.

Explicação: Em alguns casos acontece do usuário realizar uma escolha por engano no sistema, existe a necessidade de retornar ao menu e fazer uma nova opção.

Localização 1 - Em um único local na interface.

Gravidade 3 - Problema grande. É comum que o usuário realize escolhas diferentes do que realmente queria, para voltar ao menu principal ele terá que sair do programa para fazer uma nova opção.

Problema: Não foi atribuída nenhuma funcionalidade para a opção listar.

Heurística Violada: Estética e design minimalista.

Explicação: Informações irrelevantes podem tirar a concentração para o que de fato é funcional no sistema, nesse caso a opção para listar os planos de fundo são as setas.

Localização 2 - Em dois ou mais locais na interface, casualmente.

Gravidade 2 - Problema pequeno. Este diálogo pode confundir o usuário que insistirá na opção.

Avaliação Heurística da Tela de Plano de Fundo

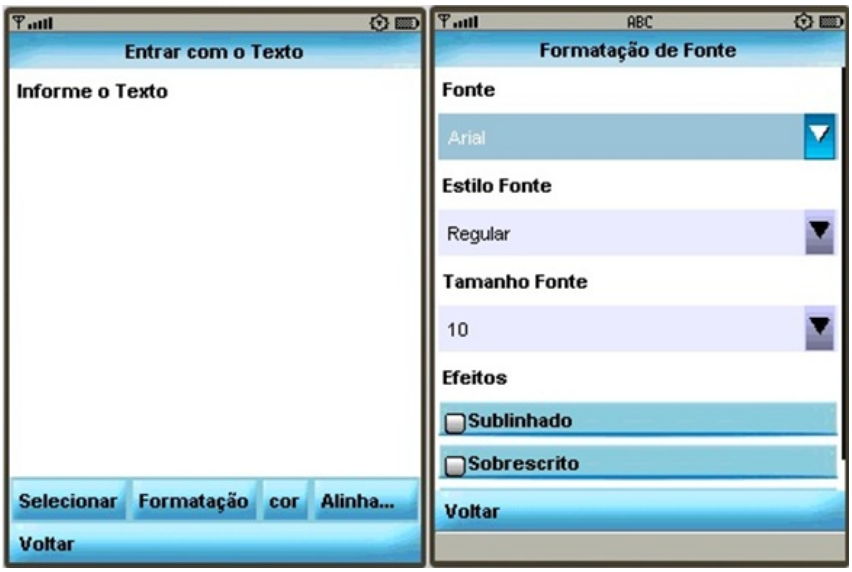


Figura 8.6: Tela Entrada e Formatação de Textos da ferramenta m-tutorial

Problema: Caso o usuário queira salvar as alterações, não existe a opção “Salvar".

Heurística Violada: Consistência e padrões.

Explicação: No desenvolvimento de um sistema é importante seguir padrões já utilizados em outros para que o usuário não fique perdido, sem saber o que fazer.

Localização 2 - Em dois ou mais locais na interface, casualmente.

Gravidade 2 - Problema pequeno. Como na sua maioria os usuários não possuem experiência em informática, seguir padrões existentes facilita a compreensão.

Avaliação Heurística da Tela de Entrada e Formatação de Textos

Figura 8.7: Tela Alinhar Textos da ferramenta m-tutorial

Problema: Para o usuário que tem interesse em justificar seu texto, a opção não foi disponibilizada.

Heurística Violada: Consistência e padrões.

Explicação: É comum que exista “justificar"entre as opções de formatação de texto.

Localização 4 - Precisa ser incluído na interface.

Gravidade 0 - Este caso não é necessariamente um erro e sim uma utilidade que os usuários geralmente encontram em outros sistemas.

Avaliação Heurística da Tela Alinhar Texto

Figura 8.8: Tela Visualizar Imagens e Textos da ferramenta m-tutorial

Problema: O usuário não consegue cancelar uma imagem adicionada.

Heurística Violada: Controle do usuário e liberdade.

Explicação: Frequentemente o usuário faz opções que deseja alterar depois.

Localização 1 - Em um único local na interface.

Gravidade 3 - Problema grande. O usuário deverá ter a possibilidade de cancelar uma imagem já adicionada.

Avaliação Heurística da Tela de Visualização

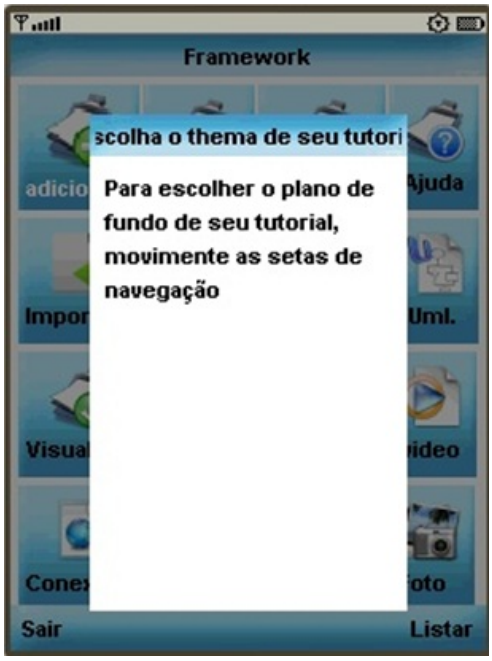


Figura 8.9: Tela Selecionar Plano de Fundo da ferramenta m-tutorial

Problema: Neste caso o usuário deve clicar nas setas enquanto esta tela estiver aberta, ele precisa aguardar a abertura do plano de fundo.
Heurística Violada: Prevenção de erros.
Explicação: Ao fim da mensagem o usuário deve ser informado para aguardar a tela de plano de fundo ser aberta.
Localização 4 - Precisa ser incluído na interface.
Gravidade 1 - Com esta informação o usuário será informado do funcionamento da interface.

Avaliação Heurística da Tela Selecionar Plano de Fundo

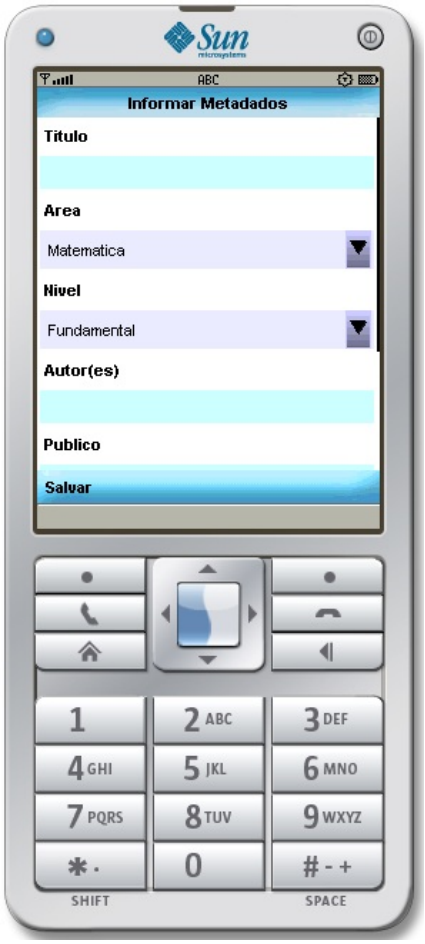


Figura 8.10: Tela Informar Metadados da ferramenta m-tutorial

Problema: O usuário terá dificuldades de compreender que para digitar as informações precisa clicar em T9.
Heurística Violada: Compatibilidade do sistema com o mundo real.
Explicação: T9 não tem ligação com o mundo real o que dificultará o entendimento.
Localização 3 - Na estrutura geral da interface, de forma sistemática.
Gravidade 4 - Catastrófico. O usuário poderá desistir de utilizar o sistema por não entender como digitar as informações.

Avaliação Heurística da Tela Informar Metadados

9 CONCLUSÃO

Para estimular o uso de objetos de aprendizagem voltados para *m-learning*, é necessário fornecer recursos aos professores para o desenvolvimento desses objetos. Neste trabalho, propôs-se a ferramenta m-tutorial com as seguintes características: que objetos de aprendizagem sejam acessados de lugares remotos, que seja fácil publicar material de apoio e que OAs sejam criados mesmo estando em modo *off-line*.

Para melhor visualização do que pode ser construído com a ferramenta, apresentou-se um tutorial construído a partir das funcionalidades do m-tutorial. A facilidade de desenvolvimento está em interfaces intuitivas e bem documentadas de forma que o usuário com conhecimento básico de informática consiga interagir e criar OAs de forma rápida.

Esta dissertação envolveu o desenvolvimento de aplicações para dispositivos móveis utilizando de tecnologias como web Service, XML, Java ME, entre outros. Os dispositivos móveis estão se transformando em importantes ferramentas para educação e treinamento, com esse intuito desenvolveu-se o m-tutorial seguindo as técnicas de engenharia de *software* para que sejam reaproveitadas em futuras aplicações.

Diversas tecnologias foram avaliadas, uma maior dedicação foi dada ao contexto de ferramentas para *m-learning*. Levantou-se informações sobre bibliotecas e *frameworks* para Java ME. Esse levantamento propiciou a escolha da biblioteca *Lwuit* entre os diversas existentes. Estabeleceu-se rotinas e padrões pelas APIs e listou-se métodos de avaliação de *frameworks*.

Desenvolveu-se o m-tutorial utilizando a metodologia RUP e as seis melhores práticas de desenvolvimento de *software* sobre os diagramas de caso de uso, classes, componentes e implantação desenhou-se o m-tutorial. Sua especificação propicia aos desenvolvedores um maior entendimento dos componentes da ferramenta.

Elaborou-se um estudo de caso para validação da ferramenta no desenvolvimento de tutoriais para dispositivos móveis. A criação de tutorial por meio da utilização de textos e imagens é demonstrada na figura 8 localizada na seção estudo de caso do capítulo 7.

A verificação contínua da qualidade é dada pela utilização de um *checklist* de inspeção de *software*. Os diagramas UML foram inspecionados por profissionais com um certo grau de experiência em engenharia de *software*. As inconsistências encontradas em seus diagramas foram resolvidas na fase inicial do ciclo de vida do m-tutorial.

Realizou-se um comparativo entre três ferramentas para *m-learning*. Suas classes foram

especificadas e implementadas seguindo características que um *framework* deve possuir para ser reusável, extensível, seguro, eficiente e completo. Avaliou-se também aspectos educacionais, mobilidade da aprendizagem, recursos dos dispositivos móveis e aprendizagem colaborativa. Os critérios avaliados no comparativo realizado ajudaram desenvolvedores de *m-learning* a contruirem novos aplicativos para dispositivos móveis.

A inspeção de usabilidade do m-tutorial foi representada por meio da avaliação heurística, os resultados foram analisados e alguns problemas resolvidos. Entre os problemas encontrados, voltar ao menu principal e escolher uma nova opção foram implementados em todas as telas. Erro no menu foi revisto, verificou-se que no período da avaliação por engano o valor *default* passado pela IDE estava sendo utilizado. Com a necessidade de poder excluir uma imagem, um método foi incorporado a ferramenta. Outros problemas foram resolvidos, como a opção de justificar um texto e prevenir erros. No contexto geral, após a análise dos resultados da avaliação do m-tutorial, a interface foi adaptada às necessidades do usuário.

A partir dos resultados apresentados, pode-se perceber que o m-tutorial, mesmo com algumas limitações, permite aos professores a elaboração de materiais pedagógicos interativos e dinâmicos, sem a utilização de uma conectividade existente para dispositivos móveis.

Como trabalhos futuros, propomos a construção de componentes adicionais, como um componente que permita criação de páginas web para dispositivos móveis, feitas no próprio aparelho, além de possibilitar ao usuário desenvolver tutoriais em português, espanhol e inglês. Essa ferramenta se propõe a utilizar o maior número possível de recursos que um celular pode oferecer, como os diferentes recursos de transferência de arquivos, formas de acesso e meios de armazenamento de dados.

Já que o m-tutorial baseia sua arquitetura no *Lwuit*, deseja-se utilizar as classes da ferramenta m-tutorial e adaptá-las para uso da TV digital oferecendo aos professores uma forma de criar objetos de aprendizagem do tipo tutorial pelo celular e visualizá-las na TV. Suportado no Ginga-J da TV digital brasileira, o *Lwuit* TV é bastante semelhante ao *Lwuit* no celular, mas tem que trabalhar agora com resoluções de tela maior e proporções diferentes para TV.

Tutoriais criados no formato de áudio ou vídeo são os grandes desafios do m-tutorial. Deseja-se oferecer classes que possibilitem ao conteudista elaborar objetos de aprendizagem em diversas extensões. Um tutorial criado em áudio ou vídeo e armazenado no próprio aparelho poderá ser enviado para um repositório de dados ofertado na Web quando possuir conectividade com a Internet.

Outras formas de avaliação como CMMI, ISO 9000-3, projeto *SPICE*, modelo PSP entre outros poderão ser utilizados para melhoria dos processos do m-tutorial. Estimativas de esforços, custos, recursos, cronogramas e características de qualidade poderão ser relacionadas ao m-tutorial.

Esta pesquisa mostrou como uma ferramenta pode se beneficiar pelo uso do padrão MVC. Mostrou-se também uma forma de avaliar a ferramenta que leva em conta tanto aspectos técnicos quanto não-técnicos. Espera-se que esta pesquisa possa auxiliar arquitetos de *software* e usuário conteudista na construção de novas aplicações para *m-learning*.

REFERÊNCIAS

ABINADER, J. A.; LINS, R. D. ***Web Services em Java***. São Paulo, Brasil: Brasport, 2006. ISBN 85-7452-249-X.

ALONSO, G. et al. ***Web Services: Concepts, Architecture and Applications***. 2004. Springer Verlag.

ANDRADE, F. ***Uma visão geral sobre: J2ME x BREW x FLASH LITE***. 2006. Http://www.felipeandrade.org.

ANIDO, L. E. et al. **Educational metadata and brokerage for learning resources**. *Comput. Educ.*, Elsevier Science Ltd., Oxford, UK, UK, v. 38, n. 4, p. 351–374, 2002. ISSN 0360-1315.

BARCELOS, R. F. **Uma Abordagem para Inspeção de Documentos Arquiteturais Baseada em Checklist**. In: . http://www.dsc.upe.br/˜mlc/artigos/requisitos/dissertacao/_RafaelBarcelos.pdf: Universidade Federal do Rio de Janeiro, COPPE, 2006.

BARRETO, C. G. **Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente Aula Net**. In: . http://www2.dbd.puc-rio.br/pergamum/tesesabertas/041082306pretextual.pdf: Universidade Católica do Rio de Janeiro, 2006.

BARRY, D. K. ***Web Services and Service-Oriented Architectures: The Savvy Manager's Guide***. San Fransisco, EUA: Morgan Kaufmann Publishers, 2003. ISBN 1558609067.

BETIOL, A. H. **Avaliação de Usabilidade para os computadores de mão: um estudo comparativo entre três abordagens para ensaios de interação**. In: . http://www.tede.ufsc.br/teses/PEPS4025.pdf: Universidade Federal de Santa Catarina, 2004.

BOGGS, W.; BOGGS, M. ***Mastering: UML com rational rose 2002 - Bíblia***. Rio de Janeiro, Brasil: Alta Books, 2002.

BOSSARD, A. ***Evaluation: GUI libraries for J2ME***. 2008. Http://newsofthefuture.net/index.php?/archives/33-Evaluation-GUI-libraries-for-J2ME.html.

BROWN, J. S.; COLLINS, A.; DUGUID, P. **ARMOLEO: An Architecture for Mobile Learning Objects**. *IEEE Multidisciplinary Engineering Education Magazine*, Washington, USA, v. 18, n. 1, p. 32–42, 1989.

BRYKCZYNSKI, B. **A survey of software inspection checklists**. *SIGSOFT Softw. Eng. Notes*, ACM, New York, NY, USA, v. 24, n. 1, p. 82, 1999. ISSN 0163-5948.

BUSCHMANN, F. et al. ***Pattern-Oriented Software Architect A System of Patterns***. West Sussex PO19 IUD, England: John Wiley e Sons, 1996.

CASTILLO, S.; AYALA, G. **ARMOLEO: An Architecture for Mobile Learning Objects**. *IEEE Multidisciplinary Engineering Education Magazine*, IEEE Computer Society, Los Alamitos, CA, USA, v. 2, p. 1–4, 2007.

CASTRO, E. ***XML para a world wide web***. Rio de Janeiro, Brasil: Campus, 2001. ISBN 853520784.

CETIC. ***Uso das Tecnologias da Informação e da Comunicação (TIC)***. 2009. Http://www.cetic.br/tic/2005/.

COOPER, K.; LIDDLE, S.; DASCALU, S. ***Experiences Using Defect Checklists in Software Engineering Education***. 2005.

CRETU, A. **Use Case Software Development and Testing Using Operational Profiles**. In: . [S.l.: s.n.], 1997.

EMILIO, C.; SEVERO, P. ***NetBeans IDE para desenvolvedores que utilizam a tecnologia Java***. Rio de Janeiro, Brasil: Brasport, 2005.

ENOUGH. ***J2me Polish***. 2010.

FERNANDES, J. F. L. ***Desenvolvendo interfaces gráficas para aplicações JavaMe***. 2007. WebMobile.

FERREIRA, S. B. L.; LEITE, J. C. S. do P. **Avaliação da Usabilidade em Sistemas de Informação: o Caso do Sistema Submarino**. v. 7, n. 2, p. 115–136, 2003.

FLORES, R. C.; MORTEO, G. L. **A Model for Collaborative Learning Objects Based on Mobile Devices**. In: *ENC '08: Proceedings of the 2008 Mexican International Conference on Computer Science*. Washington, DC, USA: IEEE Computer Society, 2008. p. 89–95. ISBN 978-0-7695-3439-8.

FRANCISCATO, F. T.; MEDINA, R. D. **M-Learning e Android: um novo paradigma?** In: . Porto Alegre, RS, BRASIL: Revista Novas Tecnologias na Educação, 2008. v. 6.

FRANCISCATO, F. T.; MEDINA, R. D. **Sistema de Gerenciamento de Objetos de Aprendizagem para dispositivos Móveis**. Novas Tecnologias na Educação, CINTED-UFRGS, RS, BRASIL, v. 7, n. 1, 2009.

GARDUSSI, B. G.; SOUZA, W. C. ***Sistema de Prontuário Médico com Serviços em Palmotop***. 2009.

GHISI, B. ***Making your ME application looks even better***. 2008. Http://weblogs.java.net/blog/brunogh/archive/2008/01/making_your_me.html.

GONÇALVES, E. ***Eclipse IDE: dicas e truques***. Rio de Janeiro, Brasil: Ciência Moderna, 2007.

HADZILACOS, T.; TRYFONA, N. **Constructive m-Learning Environments**. In: *ICALT '05: Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*. Washington, DC, USA: IEEE Computer Society, 2005. p. 271–273. ISBN 0-7695-2338-2.

HANZAN, G. C. ***SuperWaba:The Litebase Companion Version 2.0***. 2009.

HILDI. ***Cine Mobits no site do LWUIT da Sun***. 2009. Http://www.mobits.com.br/2009/5/7/cine-mobits-no-site-do-lwuit-da-sun.

HOLANDA, A. B. de. ***Novo Aurélio século XXI: o dicionário da língua portuguesa***. Rio de Janeiro, BRASIL: Nova Fronteira, 2009.

HORSTMANN, C. ***Padrões e Projetos Orientados a Objetos***. São Paulo, SP, Brasil: Bookman, 2006. ISBN 0471744875.

HOUSER, C.; THORNTON, P.; KLUGE, D. **Mobile Learning: Cell Phones and PDAs for Education**. *Computers in Education, International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 1149, 2002.

IRALA, E. A. F.; TORRES, P. L. ***O uso do AMANDA como ferramenta de apoio a uma proposta de aprendizagem colaborativa para a língua inglesa***. 2004.

JIUXIN, C.; BO, M.; JUNZHOU, L. **The Self-Adaptive Framework of Learning Object Based on Context**. In: *CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2008. p. 941–944. ISBN 978-0-7695-3336-0.

JOHNSON, P. ***Usability and Mobility: Interactions on the Move***. 1998. First Workshop on Human Computer Interaction with Mobile Devices.

JOHNSON, R. E. **Frameworks = (components + patterns)**. *Commun. ACM*, ACM, New York, NY, USA, v. 40, n. 10, p. 39–42, 1997. ISSN 0001-0782.

JOHNSON, T. ***Desenvolvendo Aplicações com J2me***. São Paulo, Brasil: Novatec., 2007. ISBN 978-85-7522-143-3.

JORGENSEN, M.; MOLOKKEN, K. **A Preliminary Checklist for Software Cost Management**. In: *Proceedings of the 3 rd IEEE International Conference on Quality Software QSIC 03*. [S.l.: s.n.], 2003. p. 134–140.

KEEGAN, D. **The Future of Learning: From eLearning to mLearning**. FernUniversitat, ZIFF, Postfach 940, D - 58084 Hagen, Germany. Fax: 49 2331 880637; Web site: http://www.fernuni-hagen.de/ZIFF., fERN uNIV., Hagen, GERMANY, p. 173, 2002.

KEEGAN, D. **The Incorporation of Mobile Learning into Mainstream Education and Training**. In: *Proccedings of mLearning2005-4th World Conference on m-learning, Cape Town*. [S.l.: s.n.], 2005.

KINSHUK et al. **Mobile technologies in support of distance learning**. *Asian Journal of Distance Education*, v. 1, n. 1, p. 60–68, 2003.

KNUDSEN, J.; FISHBEIN, C.; LEVIN, A. ***Lightweight UI Toolkit Developers Guide***. 2009. Http://www.sun.com.

KOSCIANSK, A.; SOARES, M. dos S. ***Qualidade de software: Aprenda as Metodologias e Técnicas mais Modernas para o Desenvolvimento***. São Paulo, Brasil: Novatec, 2007. ISBN 9788575221129.

KRUCHTEN, P. ***The Rational Unified Process: An Introduction***. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0321197704.

KUKULSKA-HULME, A.; TRAXLER, J. **Mobile teaching and learning**. In: . [S.l.: s.n.], 2005. p. 25–44. Kukulska-Hulme, Traxler 2005 - Mobile teaching and learning js, 17.05.2007.

KURNIAWAN, B. ***Java para Web com servlets, JSP e EJB***. Rio de Janeiro, Brasil: Ciência Moderna Ltda, 2002. ISBN 8573932104.

LAITENBERGER, O.; DEBAUD, J. marc. **An Encompassing Life-Cycle Centric Survey of Software Inspection**. *Journal of Systems and Software*, v. 50, p. 5–31, 1998.

LARMAN, C. ***Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo***. Porto Alegre, Brasil: Bookman, 2007. ISBN 0-13-148906-2.

LI, W. ***Open Source Software for Mobile Phones***. 2009. Http://opensource.ngphone.com/.

LTSC. ***Draft Standard for Learning Object Metadata. IEEE 1484.12.1-2002***. 2002. Http://ltsc.ieee.org.

MARTINS. ***M-learning: Aprendizado Móvel na nova Economia Digital***. 2006. Http://pt.shvoong.com/internet-and-technologies/473562-learning-aprendizado-móvel-na-nova/.

MATOS, A. V. de. ***UML: Prático e descomplicado***. São Paulo, Brasil: Érica Ltda, 2002. ISBN 8571948135.

MCGRATH, S. ***XML Aplicações Práticas***. Rio de Janeiro, Brasil: Campus, 2009. ISBN 8535204083.

MELO, W.; SHULL, F.; TRAVASSOS, G. ***Software Review Guidelines***. 2001. COPPE/UFRJ, Relatório Técnico.

METCALF, D. S.; MARCO, J. M. D. ***mLearning: mobile learning and performance in the palm of your hand***. Amherst, Ma, Canada: HRD Press, Inc, 2006. ISBN 0874259061.

MOBISITEGALORE. ***mobile website builder***. 2010.

MOLENET. ***Mobile Learning Network***. 2010. Http://www.molenet.org.uk/.

MUCHOW, J. W. ***Core J2ME Technology e MIDP***. 2001. Editora Prentice Hall PTR.

MULLER, L. F.; FRANTZ, G. J.; SCHREIBER, J. N. C. ***Qualcomm Brew X Sun J2ME Um comparativo entre soluções para desenvolvimento de jogos em dispositivos móveis***. 2005. Http://www.dcc.unesc.net/sulcomp/05/Art036SulComp2005.pdf.

MUSA, D. L.; SILVA, L. M.; OLIVEIRA, J. P. M. de. **Sharing Learner Profile through an Ontology and Web Services**. In: *5th International Workshop on Management of Information on the Web, Zaragoza, Spain*. [S.l.: s.n.], 2004. p. 2.

NAH, K. c.; WHITE, P.; SUSSEX, R. **The potential of using a mobile phone to access the internet for learning efl listening skills within a korean context**. *ReCALL*, Cambridge University Press, New York, NY, USA, v. 20, n. 3, p. 331–347, 2008. ISSN 0958-3440.

NAISMITH, L. et al. **Literature review in mobile technologies and learning**. *FutureLab Report*, v. 11, 2004.

NASCIMENTO, M. R. do et al. **Sistema Especialista para a Automatização do Processo de Inspeção de Conformidade de Produtos de Software ao Padrão ISO 9241**. *FutureLab Report*, v. 31, p. 59–06, 2007.

NEVES, J. M. M. ***Estudo de usabilidade em sistemas moveis com foco em PDAs***. Tese (Doutorado), Universidade Estadual de Campinas . Instituto de Computação, 2005.

NIELSEN, J. **Usability inspection methods**. In: *CHI '94: Conference companion on Human factors in computing systems*. New York, NY, USA: ACM, 1994. p. 413–414. ISBN 0-89791-651-4.

OLIVEIRA, W. P.; PINTO, W. ***Desenvolvendo Aplicações Móveis com SuperWaba***. 2009.

PAULON, R. **Web Services e J2ME de ponta a ponta**. In: . Rio de Janeiro, RJ, BRASIL: Web mobile, 2008. v. 16.

PELISSOLI, L.; LOYOLLA, W. **Aprendizado Móvel (m-learning): Dispositivos e Cenários**. In: *11º Congresso Internacional de Educação a Distância*. Salvador, BA, BRASIL: [s.n.], 2004.

PELISSONI, C. G.; CARVALHO, J. O. F. de. ***Uma Proposta de Metodologia para o Ensino da Disciplina Interação Humano-Computador em cursos de Computação e Informática***. Tese (Doutorado), 2003.

PIRES, P.; MATOSO, M. ***Tecnologia de Serviços Web***. 2005. Mini-curso SBES.

POHJOLAINEN, J. ***Mobile Platform Overview***. 2006. Tempere University of Applied Sciences.

PRATES, R. O.; BARBOSA, S. D. J. **Avaliação de Interfaces de Usuário - Conceitos e Métodos**. In: ____. [S.l.]: Anais do XXIII Congresso Nacional da Sociedade Brasileira de Computação. XXII Jornadas de Atualização em Informática (JAI), 2003. cap. Avaliação de Interfaces de Usuário - Conceitos e Métodos.

PREECE, J.; ABRAS, C.; KRICHMAR, D. M. **Designing and evaluating online communities; research speaks to emerging practice**. *Int. J. Web Based Communities*, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, v. 1, n. 1, p. 2–18, 2004. ISSN 1477-8394.

REZENDE, D. A. ***Engenharia de software e sistemas de informação***. Rio de Janeiro, Brasil: Brasport livros e Multímidia Ltda, 2005. ISBN 8574522155.

ROCHA, D. **Utilizando web Wervice em aplicações móveis**. In: . Rio de Janeiro, RJ, BRASIL: Web mObile, 2008. v. 16.

ROCHA, H. V. da; BARANAUSKAS, M. C. C. ***Design e avaliação de interfaces humano-computador***. Campinas, SP, Brasil: NIED/UNICAMP, 2003. ISBN 85-7452-249-X.

SANTOS, R. et al. **museuM: Uma Aplicação de m-Learning com Realidade Virtual**. In: *11º Congresso Internacional de Educação a Distância*. São Leopoldo: Unisinos, RS, BRASIL: [s.n.], 2005.

SARAIVA, E. G. ***Java ME no Mercado Mobile***. 2009. Http://www.erisvaldojunior.com/.

SILVA, O. J. da; CRESPO, A. N. ***Aplicação de um Checklist de Pré-Teste***. 2008.

SOLOWAY, E. et al. **Log on education: Handheld devices are ready-at-hand**. *Commun. ACM*, ACM, New York, NY, USA, v. 44, n. 6, p. 15–20, 2001. ISSN 0001-0782.

SRIRAMA, S.; JARKE, M.; PRINZ, W. **MWSMF: A Mediation Framework Realizing Scalable Mobile Web Service Provisioning**. *1st International ICST Conference on Mobile Wireless Middleware, Operating Systems and Applications*, ICST, May 2008.

STANDZISZ, P. C. ***Projeto de Software Utilizando a UML***. [S.l.]: UTFPR, 2002.

SUN, M. ***class Camera***. 2004.

SUN, M. ***Developers Guide, Lightweight UI Toolkit***. 2009.

SVETLANA, K.; YONGLK-YOON. **Adaptation e-learning contents in mobile environment**. In: *ICIS '09: Proceedings of the 2nd International Conference on Interaction Sciences*. New York, NY, USA: ACM, 2009. p. 474–479. ISBN 978-1-60558-710-3.

SZYPERSKI, C. ***Component Software: Beyond Object-Oriented Programming***. New York, NY: ACM Press and Addison-Wesley, 1998.

TAROUCO, L. M. R.; FABRE, M.-C. J. M.; TAMUSIUNAS, F. R. **Reusabilidade de Objetos educacionais. In: RENOTE**. *Revista Eletrônica de Novas Tecnologias na Educação*, UFRGS, Porto Alegre, Brasil, v. 1, n. 1, p. 39–42, 2003.

TRAXLER, J.; LEACH, J. **Innovative and Sustainable Mobile Learning in Africa**. *Wireless and Mobile Technologies in Education, IEEE International Workshop on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 98–102, 2006.

WEINBERG, G.; FREEDMAN, D. **Reviews, walkthroughs, and inspections**. *IEEE Transactions on Software Engineering SE-10,*, n. 1, p. 68–72, 1984. Disponível em: <http://www.ece.cmu.edu/ẽce749/papers/weinberg84_inspections.pdf>.

WEISS, S. ***Handheld Usability***. West Sussex PO19 IUD, England: England: John Wiley e Sons Ltd, 2002.

APÊNDICE A – INFORMAÇÕES PARA AVALIAÇÃO HEURÍSTICA

Avaliação Heurística	
Avaliador:	
Problema:	
Heurística Violada:	
Explicação:	
Gravidade:	

Planilha para Avaliação Heurística

APÊNDICE B – LISTA DE CÓDIGOS DO M-TUTORIAL

Algorithm 1: Pré-Visualização

```
1:  public String getFormPreVis()
2:  {
3:  form = new Form("Pre-Visualização"');
4:  String ret="True";
5:  form.show();
6:  if(!Imagem.equals()
7:  {
8:  labe1= new TextArea(texto);
9:  form.addComponent(label1); }
10: if(!Imagem.equals(){
11: Image imagem, imgVideo;
12: try {
13: imgVideo=Image.creatImage(Image);
14: image=Image.creatImage(Image);
15: Label ImageLabel=new Label(imagem);
16: Label videoLabel=new Label(video);
17: form.addComponent(ImagemLabel, videoLabel);
18: }
19: cath (IOException ex){
20: ex.printStackTrace();
21: }
22: }
23: return ret; }
```

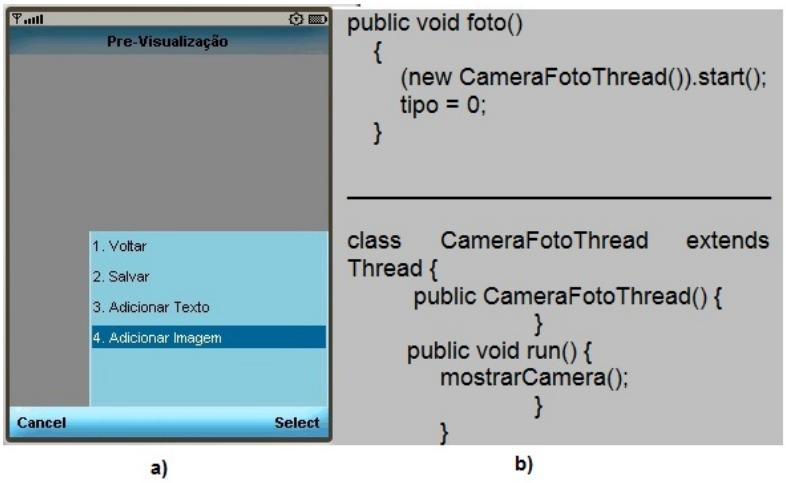


Figura B.1: a) Interface Adicionar Imagem, b)Método Foto() e Classe CameraFotoThread

O método listarConteudo pertence à classe conteudo

Algorithm 2: Método listarConteudo

```
1:  public Vector getListarConteudo() throws RecordStoreException {
2:  Vector vetorConteudo = new Vector();
3:  RecordEnumeration re = null;
4:  ByteArrayInputStream strmInBytes = null;
5:  DataInputStream strmInDataTypes = null;
6:  RecordStore rs = null;
7:  try {
8:  rs = openRecStore(RMS_CONTEUDO);
9:  if (rs.getNumRecords() > 0) {
10: re = initEnumerationSTR(rs);
11: while (re.hasNextElement()) {
12: int id = re.nextRecordId();
13: byte[] recData = rs.getRecord(id);
14: //byte[] recData = re.nextRecord();
15: if (recData != null) {
16: strmInBytes = new ByteArrayInputStream(recData);
17: strmInDataTypes = new DataInputStream(strmInBytes);
18: Conteudo conteudo = new Conteudo();
19: conteudo.setRecordId(id);
20: conteudo.setTitulo(strmInDataTypes.readUTF());
21: conteudo.setArea(strmInDataTypes.readUTF());
22: conteudo.setAutores(strmInDataTypes.readUTF());
23: conteudo.setPublicoAlvo(strmInDataTypes.readUTF());
24: conteudo.setLingua(strmInDataTypes.readUTF());
25: conteudo.setNivel(strmInDataTypes.readUTF());
26: conteudo.setVersao(strmInDataTypes.readUTF());
27: conteudo.setFundo(strmInDataTypes.readUTF());
28: conteudo.setTexto(strmInDataTypes.readUTF());
29: conteudo.setImagem(strmInDataTypes.readUTF());
30: vetorConteudo.addElement(conteudo);
31: strmInDataTypes.close();
32: strmInBytes.close();
33: } }
34: re.destroy();
35: closeRecStore(rs);
36: } else {
37: rs.closeRecordStore();
38: this.deleteRecStore(RMS_CONTEUDO); }
39: } catch (Exception e) {
40: System.out.println("reading RecordStore '"+ RMS_CONTEUDO + "'.");
41: e.printStackTrace();
42: throw new RecordStoreException(e.getMessage());
43: } return (vetorConteudo);
44: }
```

O método FormTransferir pertence à classe FormPreVis:

Algorithm 3: Método Transferir Arquivos

```
1:  private void FormTransferir() {
2:  tela = new Form("HTTPSender"');
3:  cmdEnviar = new command("Enviar", Command.ITEM, 0);
4:  tela.addCommand(cmdEnviar);
5:  tela.addCommand(cmdReceber);
6:  tela.setCommandListener(this);
7:  tela.setCommandListener(this);
8:  }
```

O método setProssPOST pertence à classe ConexaoHttp:

Algorithm 4: Método setProssPOST

```
1:  public String setProcessPOST(String dados, HttpConnection conn) {
2:  HttpConnection hc = null;
3:  String strRetorno = ;
4:  StringBuffer sb = new StringBuffer();
5:  try {
6:  hc = conn;
7:  DataOutputStream oStrm = (DataOutputStream) hc.openDataOutputStream();
8:  oStrm.writeUTF(dados);
9:  oStrm.flush();
10:  oStrm.close();
11:  DataInputStream iStrm = (DataInputStream) hc.openDataInputStream();
12:  int ch;
13:  sb = new StringBuffer();
14:  while ((ch = iStrm.read()) != -1) {
15:  sb.append((char) ch); }
16:  strRetorno = sb.toString().trim();
17:  iStrm.close();
18:  if(strRetorno == null) {
19:  strRetorno = "Erro de conexão com o web-server - NULL"; }
20:  } catch (IOException e) {
21:  strRetorno = e.getMessage();
22:  catch (SecurityException e) {
23:  strRetorno = e.getMessage();
24:  } finally {
25:  } return strRetorno;
26:  }
27:  }
```

APÊNDICE C – LISTA DE CÓDIGOS UTILIZADOS PELO M-TUTORIAL

O método mostrarCamera pertence à classe Camera:

Algorithm 5: Método Mostrar Câmera (SUN, 2004)

```
1:  private void showCameraFoto() {
2:  try {
3:  liberaRecursos();
4:  player = Manager.createPlayer("capture://video");
5:  player.realize();
6:  player.addPlayerListener(this);
7:  videoControl = (VideoControl)player.getControl("VideoControl");
8:  aVideoCanvas = new VideoCanvas();
9:  aVideoCanvas.initControls(videoControl, player);
10: aVideoCanvas.addCommand(cmCapturar);
11: aVideoCanvas.setCommandListener(this);
12: switchDisplayable(null, aVideoCanvas);
13: // getDisplay().setCurrent(aVideoCanvas);
14: player.start();
15: contentType = player.getContentType();
16: }
17: catch (IOException ioe)
18: {
19: System.out.println("ioe: "+ioe.getMessage());
20: }
21: catch (MediaException me)
22: {
23: System.out.println("me: "+me.getMessage());
24: }
25: }
```

O método initControls pertence à classe VideoCanvas:

Algorithm 6: método initControls

```
1:  public void initControls(VideoControl videoControl, Player player) {
2:  int width = getWidth();
3:  int height = getHeight();
4:  this.vc = videoControl;
5:  this.plr = player;
6:  videoControl.initDisplayMode(VideoControl.USE_DIRECT_VIDEO, this);
7:  videoControl.initDisplayMode(VideoControl.USE_GUI_PRIMITIVE, this);
8:  try {
9:  videoControl.setDisplayLocation(2, 2);
10:  videoControl.setDisplaySize(width - 4, height - 4);
11:  }
12:  catch (MediaException me) {
13:  try {
14:  videoControl.setDisplayFullScreen(true); }
15:  catch (MediaException me2) {
16:  }
17:  }
18:  videoControl.setVisible(true);
19:  }
```

APÊNDICE D – M-TUTORIAL NA WEB

Figura D.1: a)Página inicial do m-tutorial na *Web*. b)Página novo tutorial. c) Página de pesquisa por objetos de aprendizagem exitente. d) Página destinada ao *upload* de novos objetos de aprendizagem. e) Página de *download* do arquivo de instalação do m-tutorial.