UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

CAMPUS LUIZ MENEGHEL

ROBERTA DO PRADO SOARES

SISTEMA DE TRANSFERÊNCIA DE ARQUIVOS PARA DISPOSITIVOS MÓVEIS BASEADO EM JAVA, UTILIZANDO A TECNOLOGIA ANDROID

BANDEIRANTES

2013

ROBERTA DO PRADO SOARES

SISTEMA DE TRANSFERÊNCIA DE ARQUIVOS PARA DISPOSITIVOS MÓVEIS BASEADO EM JAVA UTILIZANDO A TECNOLOGIA ANDROID

Trabalho de Conclusão de Curso apresentado à Universidade Estadual do Norte do Paraná – Campus Luiz Meneghel - como requisito parcial para a obtenção de grau de Bacharelado e Licenciatura no curso de Sistemas de Informação.

Orientador: Estevan Braz Brandt Costa

BANDEIRANTES

2013

ROBERTA DO PRADO SOARES

SISTEMA DE TRANSFERÊNCIA DE ARQUIVOS PARA DISPOSITIVOS MÓVEIS BASEADO EM JAVA UTILIZANDO A TECNOLOGIA ANDROID

Trabalho de Conclusão de Curso apresentado à Universidade Estadual do Norte do Paraná – Campus Luiz Meneghel - como requisito parcial para a obtenção de grau de Bacharelado e Licenciatura no curso de Sistemas de Informação, com nota final igual a _____, conferida pela Banca Avaliadora formada pelos professores:

Estevan Braz Brandt Costa
Universidade Estadual do Norte do Paraná-CLM

Carlos Eduardo Ribeiro
Universidade Estadual do Norte do Paraná-CLM

Glauco Carlos Silva
Universidade Estadual do Norte do Paraná-CLM

Bandeirantes, _____ de _____ de 2013.

Dedico este trabalho primeiramente a Deus, por ter me dado o dom da vida e a graça de poder estudar. Em maneira especial, dedico à todos àqueles que me ajudaram, colaboram e acreditaram na conclusão deste trabalho.

AGRADECIMENTOS

Agradeço à Deus, incondicionalmente, pelo dom da vida. Aos meus pais, Suzimara do Prado Soares e Roberto Ferreira Soares, por terem me dado a vida e me educado, participando ativamente de todo o caminho percorrido até aqui, por terem acreditado em mim e lutado comigo por todos esse anos. Em especial, ao meu irmão Murilo do Prado Soares, por toda sinceridade, transparência e felicidade que uma criança pode proporcionar. Sem dúvida alguma, vocês são as pessoas mais importantes em minha vida.

De uma forma especial, agradeço a todos os professores do curso de Sistemas de Informação, por todo o conhecimento que nos foi passado durante esses quatro anos de faculdade. Agradeço principalmente ao Estevan Braz Brandt Costa, meu professor orientador que, me apoiou e me ajudou para realização deste trabalho.

Não posso deixar de agradecer à toda XVII Turma, pela batalha conjunta e por todos esses anos de conhecimento, amizade e companheirismo, sem dúvidas a melhor turma da faculdade, pela qual tenho maior respeito e admiração e que ficará para sempre guardada na lembrança. Agradeço pela minha dupla Jhony Petterson e Felipe Eleutério, por todos os trabalhos, apresentações, brincadeiras e puxões de orelhas. Uma lembrança em especial para Jaini Domingues e Felipe Rezende pela ajuda e amizade quando eu mais precisei no final de 2012.

Agradeço pelo amor do meu namorado Ricardo Cecílio, por sempre estar ao meu lado e me incentivar. Obrigada pelo companheirismo, respeito, amor e por me permitir fazer parte de sua vida. Às minhas amigas: loira, Keily Carvalho; pequena, Andressa Mangolin e cupida, Marieli Marchione pela amizade e sinceridade durante esse anos. À minha amiga, irmã, agora comadre Majorí Maziero pela amizade e companheirismo, e ao seu marido Saulo Nascimento, sem dúvidas são amigos mais que especiais. Aos meninos da van, pela alegria que traziam aos meus dias. E, à todos aqueles que um dia fizeram parte da minha vida de uma forma significativa e que colabolaram pela realização deste trabalho.

Minha eterna gratidão à todos.

"O maior erro que se pode cometer

é fazer as coisas mais complicadas

do que elas realmente são."

(Um estudo em Vermelho-Sherlock Holmes)

RESUMO

Este trabalho apresenta a especificação de um sistema de controle de acesso de arquivos para dispositivos móveis utilizando *Web Services*. Apresenta um estudo sobre os *Web Services*, linguagem Java, plataforma Android, dispositivos móveis e transferência de arquivos. É mostrado como a plataforma Android vinculada ao Java possui recursos para desenvolvimento em *smartphones* e como a comunicação do dispositivo móveis com o servidor pode ser feita de maneira simples.

Palavras-chave: Dispositivos móveis. *Web Service*. Java. Android.

ABSTRACT

The present work shows the specification of a system control of file access for mobile devices using Web Services. It also presents a study on Web Services, Java, Android, mobile devices and transfer files. The search has showed how the Android platform has linked to Java one have developed features on smartphones and how the mobile communication device with the server can be done in a simple way.

Keywords: Mobile device. *Web Service*. Java. Android.

LISTA DE FIGURAS

LISTA DE SIGLAS

APIs - A*pplication Programming Interface*

AVD – *Android Virtual Device*

GPS - *Global Positioning System*

HTML - *HyperText Markup Language*

HTTP - *HyperText Transfer Protocol*

HTTPS - *Hypertext Transfer Protocol Secure*

IP - *Internet Protocol*

J2ME - *Java Plataform Micro Edition*

JDBC - *Java Database Connectivity*

JVM - *Java Virtual Machine*

RMI - *Remote Method Invocation*

SDK - *Software Development Kit*

SMS - *Safety Management System*

SOAP - *Simple Object Access Protocol*

SOs – Sistemas Operacionais

TCP - *Transmission Control Protocol*

UML - *Unified Modeling Language*

URL – *Uniform Resource Locator*

VM – Virtual Machine

WDSL - *Web Services Description Language*

XML - *Extensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

Atualmente, os tempos são marcados pela mobilidade e uso crescente de dispositivos móveis (SIQUEIRA,2011). A incorporação de *softwares* e aplicativos de todo o tipo, tem transformado o celular em um verdadeiro computador de bolso e capaz de fazer tudo, os chamados *smartphones*.

Estes aparelhos assumem grandes vantagens e disponibilidade de serviços, com funções que antes eram exclusivas de computadores *desktops*, para atender as necessidades dos usuários e também para acompanhar a constante evolução deste setor da tecnologia. As principais razões para os consumidores optarem por *smartphones* é a portabilidade e ser mais funcional que um telefone celular comum.

Para acompanhar o aumento da popularidade desses aparelhos, surgem diversas demandas da área de sistemas e aplicativos para dispositivos móveis, para que se possa tirar o máximo proveito dos recursos (LECHETA,2010). Há no mercado um setor destinado somente ao desenvolvimento de *software* para tais dispositivos.

Segundo Burégio (2003), as aplicações variam entre os mais diversos tipos, desde uma simples aplicação *stand-alone* até os mais recentes *Web Services*. Vale ressaltar que, mesmo com os mais diversos aplicativos, os meios de trocas de informação entre dispositivos ainda são precários, dentre eles, pode-se citar os mais comuns *Bluetooth*, *Wi-Fi*, tecnologias *2G* e *3G*, mas em determinados casos esses meios retardam o tempo de troca de informação, assim para atender a demanda dos dispositivos móveis e essa comunicação entre os mesmos, as empresas teriam de aplicar um grande investimento para o desenvolvimento e aplicações com redes *Wi-Fi*.

Em algumas empresas, o número de dispositivos móveis é maior que o número de computadores *desktops* (SIQUEIRA,2011). Assim, muitas dessas empresas investem na compra ou desenvolvimento de

aplicativos para facilitar e agregar valor no dia-a-dia do seu funcionário(LECHETA,2010).

Baseando-se essas informações, surgiu então a ideia de explorar área de desenvolvimento para dispositivos móveis e criar uma ferramenta para transferência de arquivos. A ferramenta que será desenvolvida é uma aplicação cliente para um dispositivo móvel com S.O. Android e um *Web Service* Java que, se conectará à essa aplicação e realizará todas as operações para o fluxo das transferências de arquivos.

1.1 FORMULAÇÃO DO PROBLEMA

Como integrar a tecnologia Android, presente nos *smartphones*, com o uso de *Web Services* para transferência de arquivos de maneira eficiente?

1.2 OBJETIVO GERAL

O objetivo deste trabalho é desenvolver uma aplicação que, com o auxílio de *Web Service* através da internet, seja capaz de fazer o envio e recebimento de arquivos entre um dispositivo móvel e um servidor de arquivos.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos do trabalho são:

a) criar um módulo que irá ser executado no servidor, no caso será o *Web Service,* para então fazer a gerência do sistema, ele será capaz de ler e gravar arquivos em um banco de dados e disponibilizá-los;

b) criar outro módulo para ser executado no dispositivo móvel, com Android, que terá as funcionalidades do sistema, ou seja a aplicação será capaz de consumir o serviço web previamente descrito e;

c) realizar a transferência de arquivos do dispositivo móvel para o servidor, utilizando o *Web Service*.

1.4 JUSTIFICATIVA

A presente pesquisa justifica-se pela necessidade em oferecer uma alternativa para transferir arquivos de um dispositivo móvel para um *Web Service* de uma maneira mais eficaz.

O mercado de celulares cresce a cada dia e os usuários buscam sempre aparelhos que apresentam cada vez mais recursos. Sendo assim, a plataforma Android é a resposta exata para isso, pois os dispositivos com tal tecnologia estão cada vez mais acessíveis e apresentam uma abundância em aplicativos e serviços. Além disso, oferece um ambiente de desenvolvimento bastante poderoso, com todos os recursos necessários para a criação de uma aplicação cliente e fazer com que a mesma se comunique com um *Web Service*.

Assim como o avanço da tecnologia Android, os *Web Services* também vêm ganhando destaque, quando o assunto é componente distribuído. Utilizar um *Web Service* é absolutamente viável pois eles são baseados em padrões abertos e podem ser usados em qualquer plataforma. Fica claro observar então, que o *Web Service* pode ser acessado por meio de programas de computadores ou até mesmo de dispositivos móveis.

Criar uma aplicação Android que se comunique com o *Web Service* é uma maneira de acoplar duas tecnologias consideradas novas e provar que é possível meios mais ágeis e acessíveis para se ter acesso à arquivos. A aplicação que executa no celular pode estar conectada ou online, enquanto sincroniza e transfere informações diretamente de um servidor, que nada mais é que um *Web Service* implementado e executando.

1.5 METODOLOGIA

Essa pesquisa é classificada como exploratória na medida em que contribui para aumentar os conhecimentos sobre o tema. A pesquisa será executada por meio de revisão de literatura sobre como transferir arquivos

entre dispositivos móveis baseado em Java para Android, seguido de um estudo empírico, o qual se verifica a eficiência desse aplicativo.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os assuntos e conceitos que serão abordados no presente trabalho. Por primeiro, é abordado o tema dispositivos móveis, posteriormente o Java e o Java ME. Após, será tratado o Android e sua relação com Java, seguindo por *Web Services* e os protocolos de transferência de arquivos.

2.1 DISPOSITIVOS MÓVEIS

Segundo Netto (s.d.), mobilidade é o termo utilizado para identificar dispositivos que podem ser operados a distância ou sem fio. Dispositivos que podem ser desde um simples “bip”, até os mais modernos *pockets*, ou até mesmo *smartphones*. Sendo assim, nos últimos anos, têm-se passado grande transformação tecnológica em meio a sociedade, o que acarreta também uma grande mudança de conceitos. Paradigmas que antes impediam que uma nova tecnologia chegasse ao mercado, estão sendo quebrados, e assim sendo inovado cada dia mais.

Outro conceito, segundo Reza B'Far (2005), Sistemas Computacionais Móveis são sistemas que podem facilmente ser movidos fisicamente ou cujas capacidades podem ser utilizadas enquanto eles estão sendo movidos. Como estes sistemas preveem tal mobilidade, eles normalmente oferecem recursos e características que não se encontra em sistemas comuns. Para efeito de classificação, combina-se que, para ser “móvel” mesmo, um dispositivo/sistema deve oferecer a possibilidade de acesso imediato e com o usuário em movimento.

Dentro do conceito de dispositivos móveis supracitado, segundo Mendonça (s.d.), pode-se considerar que a computação móvel começou em meados de 1992, com a introdução no mercado de um *handheld* chamado Newton, pela Apple. Este modelo não teve muita repercussão, mas é considerado o início dos dispositivos móveis.

Em 1996, a U.S. Robotics lançou o (*Palm*) Pilot 1000 e 5000, dispositivos que tiveram uma grande aceitação no mercado e lançaram as bases de toda uma plataforma de "*Palms*" que chegaram a atingir 80% do mercado mundial e existem até hoje. Também em 1996, começaram a surgir dispositivos com o Windows CE 1.0, da Microsoft.

A partir do Sistema Operacional Pocket PC 2000, embutido em dispositivos como o HP Jornada e o Compaq Ipaq, esta plataforma ganhou aceitação do mercado e começou a crescer. Correndo por fora, a empresa Symbiam foi formada em 1998 por alguns dos maiores fabricantes de celulares do mundo e a PSION, e entregou ao mercado o sistema operacional Symbiam, que roda na maioria dos *smartphones* e handhelds da Nokia. (Mendonça, s.d.)

De acordo com Pekus Consultoria e Desenvolvimento (2002), P*ocket PCs* e *Palms* são categorias de dispositivos móveis frequentemente utilizados em processos de computação móvel. Muito mais do que assistentes pessoais ou agendas eletrônicas, estes dispositivos são computadores que podem ser facilmente levados a qualquer lugar, criados para atender profissionais em movimento que necessitem de rapidez, facilidade e segurança no acesso a informações corporativas e pessoais.

Conforme Schaefer (2004, p. 20), as grandes inovações trazidas pelas tecnologias sem fio fizeram com que a indústria deste setor tenha tido um crescimento expressivo nos últimos anos, tornando-se uma das mais eficientes e rápidas áreas tecnológicas do mundo, permitindo que as pessoas comuniquem-se de forma barata e fácil, sem ficarem presas aos seus telefones ou computadores de mesa. É apresentado na figura 1, os dispositivos móveis de acordo com sua história e evolução.

Figura 1- Avanço dos dispositivos móveis

Sendo assim, atualmente o mercado está se centralizando em recursos para dispositivos móveis, criando equipamentos que concentram funções de palmtops, celular, câmera fotográfica, GPS etc., além de oferecerem excelente performance, grande capacidade de armazenamento e inúmeras possibilidades de comunicação.

2.1.1 Benefícios

De acordo com Mendonça (s.d.), a vantagem mais trivial da mobilidade é, logicamente, a possibilidade de acessar dados em qualquer lugar e a qualquer momento. Com sistemas móveis bem planejados, é possível:

a) reduzir custos de comunicação;
b) reduzir custos de entrada/processamento de dados;
c) otimizar o tempo e;
d) aumentar seu faturamento.

Não seria possível enumerar todas as vantagens de se utilizar sistemas móveis no dia-a-dia de uma pessoa ou empresa. Além disso, cada tipo de solução oferece conjuntos de vantagens diferentes.

2.1.2 *Smartphones*

Segundo Barros (2011), *smartphone* é, em tradução literal, "um telefone inteligente". E não há melhor maneira de definir este tipo de produto. Ele é a evolução do celular. A capacidade de realizar e receber chamadas é “apenas um detalhe” para este aparelho, que permite uma infinidade de possibilidades. Os *smartphones* são híbridos entre celulares e computadores.

Não têm o *hardware* potente de um PC, mas também não são tão simples quanto um telefone. Afinal, eles englobam algumas das principais tecnologias de comunicação em somente um local: internet, GPS, e-mail, SMS, mensageiro instantâneo e aplicativos para muitos fins. Para muitos, é como ter o mundo ao alcance de um simples toque. É apresentado na figura 2, alguns dos diversos modelos de *smartphones.*

Figura 2 - *Smartphones*

Conforme Miranda (2004), o maior objetivo de um *smartphone* é ser um dispositivo móvel pequeno e permitir um tempo mínimo de uso sem recarga de bateria. Devido a isso, o processador não pode possuir um *clock* muito elevado, pois significará maior consumo de energia.

Os *smartphones* modernos estão muito mais próximos (em termos de tecnologia) de um *netbook* do que de um celular mais antigo. Assim como os computadores, precisam de um sistemas operacional para funcionar, pois possuem *softwares*, que são responsáveis por administrar todos os programas (no caso, aplicativos) e funções.

O sistemas operacional de um *smartphone* é a interação entre o usuário e o aparelho, ele permite que seja possível utilizar diversos aplicativos multimídia ao mesmo tempo, sem ter que fechar um ou outro, como em um computador. Assim, o sistema operacional oferece uma função multitarefa, permitindo baixar aplicativos úteis e de entretenimento, conforme a necessidade do usuário.

Na sessão seguinte, segue o sistema operacional Android, o qual será utilizado no desenvolvimento deste trabalho.

2.2 ANDROID

Segundo Ableson (2009), Android é um ambiente operacional completo baseado no *kernel* Linux® V2.6. Inicialmente, o destino de implementação para o Android era a arena do telefone móvel, incluindo *smartphones* e dispositivos de baixo custo. Entretanto, a variedade completa do Android de serviços de computação e o suporte totalmente funcional têm potencial para ir além do mercado de telefones móveis. O Android pode ser útil para outras plataformas e aplicativos.

A plataforma Android é o produto do *Open Handset Alliance*, um grupo de organizações que colaboram para a construção de um telefone móvel melhor. O grupo, liderado pelo Google, inclui operadores de telefonia móvel, fabricantes de aparelhos portáteis, fabricantes de componentes, provedores de plataformas e soluções de *software* e empresas de marketing. (Ableson, 2009)

A partir de um ponto de vista de desenvolvimento de software, o Android fica bem ao centro do mundo do *software* livre. O primeiro telefone portátil com capacidade para Android no mercado foi o dispositivo G1 fabricado pela HTC e fornecido pela T-Mobile (IBM). O dispositivo se tornou disponível após quase um ano de especulações, quando as únicas ferramentas de desenvolvimento de *software* disponíveis eram alguns *releases* do SDK em constante aprimoramento. Conforme a data de *release* do G1 se aproximava, a equipe do Android liberou o SDK V1.0 e os aplicativos começaram a aparecer para a nova plataforma.

Com a variedade de recursos do Android, seria fácil confundi-lo com um sistema operacional *desktop*. O subsistema inclui:

a) Janelas;
b) Visualizações e;
c) *Widgets* para a exibição de elementos comuns como caixas de edição, listas e listas suspensas.

Historicamente, duas áreas onde aplicações móveis lutaram para acompanhar suas contrapartes de desktop são gráfico/mídia e métodos de armazenamento de dados. O Android aborda o desafio dos gráficos com suporte integrado para gráficos em 2-D e 3-D, incluindo a biblioteca OpenGL. O

peso do armazenamento de dados é amenizado porque a plataforma Android inclui o banco de dados SQLite de *software* livre popular. É mostrado na figura 3 , uma visualização simplificada das camadas do *software* Android.

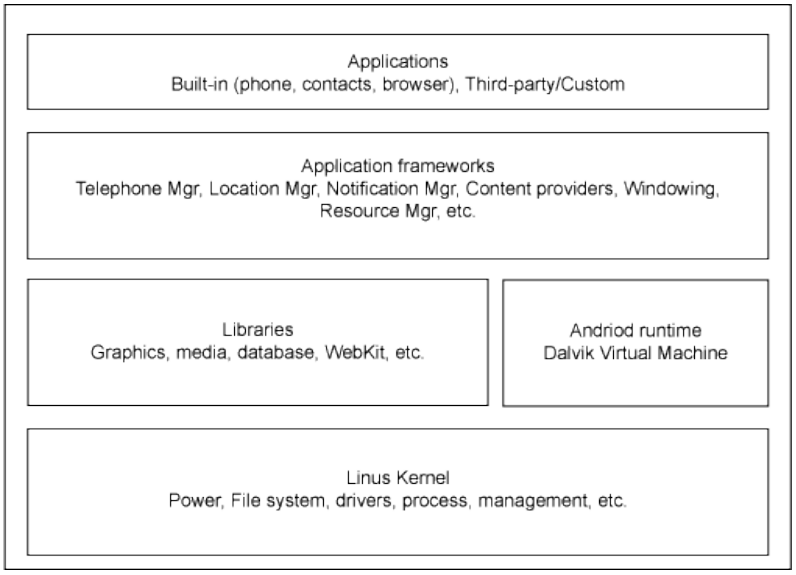


Figura 3 - Camadas de Software, Android

Conforme mencionado, o Android é executado sobre um *kernel* Linux. Os aplicativos Android são gravados na linguagem de programação Java e são executados em uma máquina virtual (VM). É importante observar que a VM não é uma JVM, como pode esperar, mas é uma *Dalvik Virtual Machine*, uma tecnologia de *software* livre. Cada aplicativo Android é executado em uma instância da *Dalvik VM*, que, por sua vez, reside em um processo gerenciado por *kernel* Linux, conforme mostrado na figura 4 abaixo.

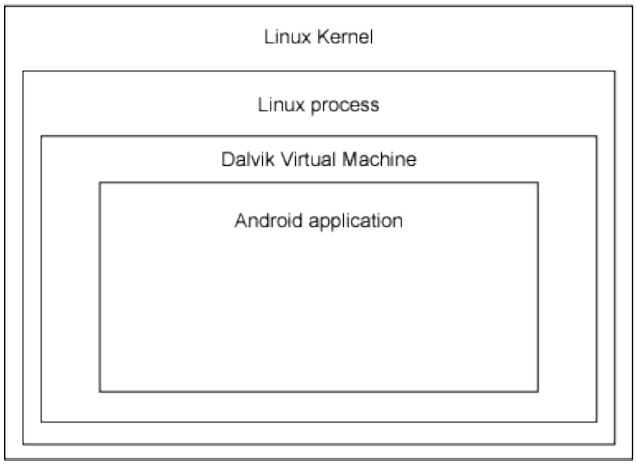


Figura 4 - Linux Kernel

Uma aplicação Android consiste de uma ou mais das seguintes classificações:

a) **Atividades:** Um aplicativo que tem uma UI visível é implementado com uma atividade. Quando um usuário seleciona uma aplicação a partir da tela inicial ou lançador de aplicativos, uma atividade é iniciada.
b) **Serviços:** Um serviço deve ser usado para qualquer aplicação que precisa persistir durante um longo período de tempo, tal como um monitor de rede ou atualização de verificação de aplicação.
c) **Os provedores de conteúdo:** O trabalho de um provedor de conteúdo é gerenciar o acesso a dados persistentes, como um banco de dados SQLite. Se a sua aplicação é muito simples, você não pode necessariamente criar um provedor de conteúdo. Se você está construindo uma aplicação maior, ou um que torna os dados disponíveis para múltiplas atividades ou aplicações, um provedor de conteúdo é o meio de acessar seus dados.
d) **Receptores de radiodifusão:** Um aplicativo Android pode ser lançado para processar um elemento de dados ou responder a um evento, como o recebimento de uma mensagem de texto.

Dessa forma, aplicações Android, são desenvolvidas em linguagem em Java, pois, é a plataforma ideal para implementar aplicativos baseados na Internet e software para dispositivos móveis que se comunicam por rede.

2.3 JAVA

Segundo a Easy Java Magazine 1 (2010), a Plataforma Java é um dos ambientes de desenvolvimento de aplicações mais utilizados no mundo hoje. Desde as primeiras novidades – como *JavaBeans, JDBC, Applets, AWT/Swing, RMI* – passando pelas inovações na linguagem (*Annotations, Generics*) e chegando até as tecnologias *Enterprise Edition* (EJB, *Web*

*Service*s, JSF), Java vem constantemente mudando a forma de como construir *softwares*, criando muitas oportunidades para o desenvolvedor.

Segundo Moraes, em 1991, na Sun Microsystems, foi iniciado o *Green Project*, o berço do Java uma linguagem de programação orientada a objetos. Os mentores do projeto eram Patrick Naughton, Mike Sheridan, e James Gosling. O objetivo do projeto não era a criação de uma nova linguagem de programação, mas antecipar e planejar a “próxima onda” do mundo digital.

Eles acreditavam que em algum tempo haveria uma convergência dos computadores com os equipamentos e eletrodomésticos comumente usados pelas pessoas no seu dia-a-dia. A tecnologia Java tinha sido projetada para se mover por meio das redes de dispositivos heterogêneos, redes como a Internet.

Agora aplicações poderiam ser executadas dentro dos *browsers* nos *Applets* Java e tudo seria disponibilizado pela Internet instantaneamente. Foi o estático HTML dos *browsers* que promoveu a rápida disseminação da dinâmica tecnologia Java.

Desde seu lançamento, em maio de 1995, a plataforma Java foi adotada mais rapidamente do que qualquer outra linguagem de programação na história da computação. Em 2003 Java atingiu a marca de 4 milhões de desenvolvedores em todo mundo. Java continuou e continua crescendo e hoje é com certeza um padrão para o mercado oferecendo qualidade, performance e segurança ainda sem nenhum competidor a altura. Java tornou-se popular pelo seu uso na Internet e hoje possui seu ambiente de execução presente em *web browsers*, *mainframes*, SOs, celulares, *palmtops* e cartões inteligentes, entre outros.

2.3.1 J2ME (Java 2 Micro Edition)

Segundo o site da Oracle, a tecnologia Java ME foi originalmente criada para lidar com as limitações associadas com a criação de aplicativos para dispositivos pequenos. Para este efeito, a Oracle definiu os princípios básicos para a tecnologia Java ME caber em um ambiente limitado e torná-lo possível para criar aplicativos Java executados em pequenos dispositivos com memória limitada, *display* e capacidade de potência.

A plataforma Java ME é uma coleção de tecnologias e especificações que podem ser combinados para criar um ambiente de execução Java completo, especificamente para atender às exigências de um dispositivo específico ou mercado. Isso oferece uma flexibilidade e coexistência para todos os usuários, de forma transparente coopera para oferecer a experiência mais atraente para o usuário final (Oracle).

Ainda de acordo com a Oracle, a tecnologia Java ME é baseada em três elementos:

a) uma configuração oferece o conjunto mais básico de bibliotecas e capacidades de máquinas virtuais para uma ampla gama de dispositivos,
b) um perfil e um conjunto de APIs que suportam uma faixa mais estreita de dispositivos e;
c) um pacote opcional que é um conjunto de tecnologias específicas de API.

2.3.2 Android x Java ME

Segundo Corrêa, o desenvolvimento de aplicações no Android foi focado na reusabilidade de código, isto é, uma aplicação pode utilizar componentes desenvolvido por outra aplicação. Esta abordagem tem problemas de implementação na prática, pois ainda não existe meio de se *linkar* este componente como uma parte integrante da sua aplicação. Desta forma, para cada componente de outra aplicação que seja utilizado, vai haver uma instância de máquina virtual para executar este componentes. Isso limita a reusabilidade, pois o uso de diversos componentes pode levar a um *overhead* de processamento muito grande.

No J2ME, o foco é o mesmo para as outras tecnologias: portabilidade. Levando em consideração a variedade de dispositivos que o J2Me procura atender, e é bem mais variado que a plataforma PC, perde-se em alguns pontos para garantir a portabilidade. A JSME não assume nada sobre o sistema que ela está operando, e alguns para alguns tipo de *hardware*,

não é fornecida a máquina virtual uma informação a respeito deste. Por exemplo, não existem métodos consistentes e dedicados para descobrir se o dispositivo sobre o qual uma aplicação está rodando tem teclado QWERTY.

Corrêa, ainda cita que é de se esperar que um sistema operacional com ambiente de execução dedicado consiga lidar melhor com vários problemas que uma máquina virtual embutida sobre um sistema operacional (*a priori*) genérico. Pelo que foi estudado e visto na prática, o Android tem uma série de vantagens sobre o J2ME. O sistema operacional fica entre o *hardware* e a máquina virtual, e de forma análoga ao que tem-se com o Java para *desktop*, ganha-se em portabilidade mas perde-se em diversos outros pontos. No contexto de dispositivos móveis com capacidade de processamento limitado, o ônus da portabilidade pode não ser compensatório.

De acordo com Ogligari (2011), entretanto, conclui que entre o que foi citado acima não existe uma resposta mágica, não existe um padrão e uma plataforma a ser seguida, tudo vai depender do que o desenvolvedor estiver criando ou do que o cliente está exigindo, pois por exemplo: se o cliente exige que o seu aplicativo funcione na grande maioria dos dispositivos presentes no mercado hoje, aconselha-se o uso de Java ME, já se o cliente quer passar uma imagem de pioneirismo e está sempre atento ao mercado e seus números, Android.

2.4 *WEB SERVICES*

Segundo i-Web, um *Web Service* é um componente, ou unidade lógica de aplicação, acessível por meio de protocolos padrões de Internet. Como componentes, esses serviços possuem uma funcionalidade que pode ser reutilizada sem a preocupação de como é implementada. O modo de acesso é diferente de alguns modelos anteriores.

Conforme Scribner e Stiver (2002, p. 10), *Web Services* podem ser descritos como qualquer funcionalidade que está acessível pela internet, geralmente utilizando uma ou mais mensagens XML no protocolo de comunicação. *Web Services* usa o conceito de operação, que representa a associação de uma requisição para zero ou mais respostas. Quando estas

operações são combinadas para satisfazer algum propósito específico, tem-se uma interface.

A base para a construção de um *Web Service* são os padrões SOAP e XML. O transporte é feito totalmente via *Hypertext Transfer Protocol* (HTTP), através do formato XML e encapsulado no protocolo SOAP. Por meio da linguagem WSDL, os métodos do *Web Service* são descritos.

2.4.1 Arquitetura de um *Web Service*

A arquitetura de *Web Service*s se baseia na interação de três entidades: provedor do serviço (*service provider),* cliente do serviço (*service requestor*) e servidor de registro (*service registry*). De uma forma geral, as interações são para publicação, busca e execução de operações. É ilustrado na figura 5, estas operações, os componentes envolvidos e suas interações.

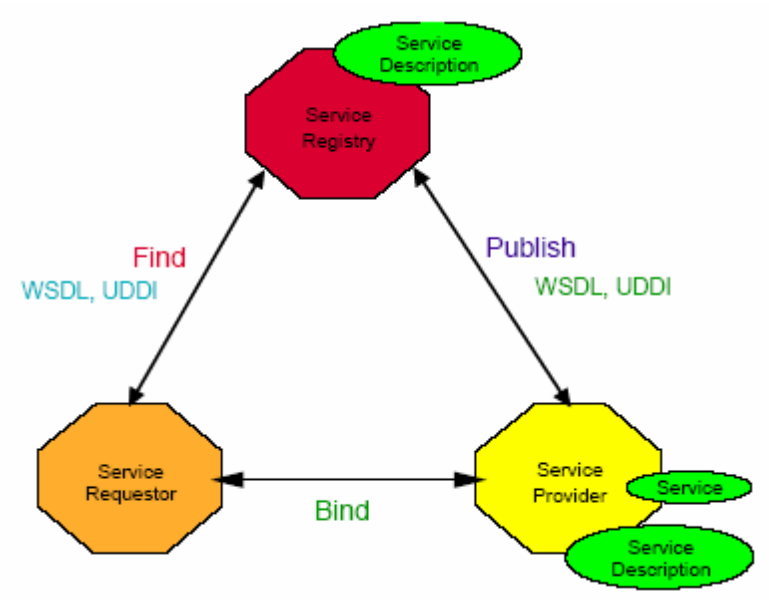


Figura 5 - Arquitetura de um *Web Service*

O provedor do serviço *(service provider)* representa a plataforma que hospeda o *Web Service* permitindo que os clientes acessem o serviço. O cliente do serviço *(service requestor)* é a aplicação que está

procurando, invocando ou iniciando uma interação com o *Web Service*. O cliente do serviço pode ser uma pessoa acessando por meio de um browser ou uma aplicação realizando uma invocação aos métodos descritos na interface do *Web Service*. O *service registry* representa os servidores de registro e busca de *Web Services* baseados em arquivos de descrição de serviços que foram publicados pelos *service providers*. Os clientes (*service requestor*) buscam por serviços nos servidores de registro e recuperam informações referentes a interface de comunicação para os *Web Services* durante a fase de desenvolvimento ou durante a execução do cliente, denominados *static binding* e *dinamic binding*, respectivamente.

Todo esse processo faz uso do protocolo HTTP para se comunicar. Como normalmente os *firewalls* não bloqueiam esta porta, não existem restrições para o seu funcionamento em redes internas ou internet.

A comunicação e troca de mensagens é possível por causa do pacote de camadas de um *Web Service*. Ela é implementada por meio de cinco tipos de tecnologias, organizadas em camadas definidas uma sobre a outra. É mostrado na figura 6, esta estrutura em camadas.

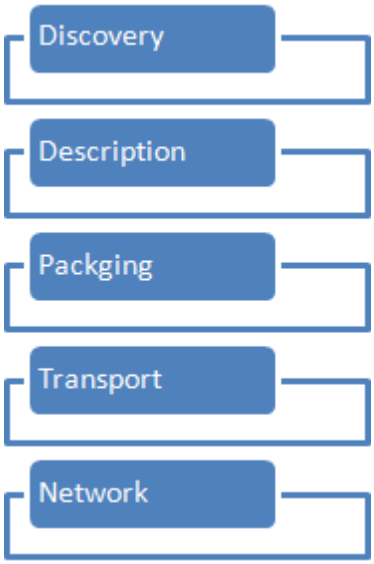


Figura 6 - Camadas do *Web Service*

As camadas *packging, description e discovery* da pilha de tecnologias do *Web Service* são essenciais para o funcionamento do modelo de três entidades (*service provider, service registry e service consumer*).

Como cada camada da pilha do *Web Service* se dirige a um problema em separado, a implementação destas camadas também é independente. Ou seja, quando uma nova camada tiver que ser criada ou alterada, não haverá necessidade de alteração nas outras camadas.

A primeira camada da pilha (*discovery*), fornece o mecanismo que um consumidor de *Web Service* necessita para buscar a interface e descrições do provedor de serviço. A especificação usada e reconhecida para a criação desta camada é a *Universal Description, Discovery, and Integration* (UDDI). UDDI é uma especificação técnica que tem como objetivo descrever, integrar e descobrir *Web Services* (SNELL; TIDWELL; KULCHENKO, 2002, p. 7).

Assim como as demais tecnologias do *Web Service*, o UDDI é baseado em XML, a qual fornece uma plataforma de dados neutra e permite descrever relações hierárquicas (RECKZIEGEL, 2006).

A segunda camada da pilha de tecnologias *Web Service* (*description*), é a camada responsável por descrever o que o *Web Service* pode oferecer ao cliente. De tal maneira que o cliente do serviço possa fazer uso do serviço (SNELL; TIDWELL; KULCHENKO, 2002, p. 7). É nesta camada que se descreve os serviços externos, ou interfaces que são oferecidos por um determinado *Web Service*. Para prover estas informações é usada uma linguagem especificada em XML, a WSDL. Neste XML, entre vários elementos, estão definidos os tipos de dados, parâmetros de entrada e saída de um serviço e protocolo usado para comunicação (RECKZIEGEL, 2006).

Para que os dados trafeguem pela rede, eles devem ser —empacotados em um formato que todos entendam, ou seja, um formato padrão. A terceira camada (*packaging*) é a camada responsável por este empacotamento de dados, onde o SOAP é o padrão mais comum de empacotamento, baseado em XML (SNELL; TIDWELL; KULCHENKO, 2002, p. 8).

A camada de transporte (*transport*) tem como objetivo mover dados entre dois ou mais locais em uma rede. Nela incluem várias tecnologias que permitem a comunicação entre aplicações. Entre estas tecnologias estão protocolos como *Transmission Control Protocol* (TCP), HTTP, *Simple Mail Transfer Protocol* (SMTP) entre outros. O *Web Service* pode ser construído

sobre quase todos os protocolos de transporte (SNELL; TIDWELL; KULCHENKO, 2002, p. 8, tradução própria).

A camada de rede (*network*) da pilha de tecnologias do *Web Services* é exatamente a mesma do modelo de rede de TCT/IP. É sobre esta camada que os protocolos de transporte trafegam.

2.4.2 SOAP

SOAP é um protocolo para troca de mensagens em ambiente distribuído, baseado em definições XML. Utilizado para acessar *Web Services*, seu papel básico é de definir mecanismos que expressem o modelo semântico das aplicações, assim como o modelo em que os dados são decodificados. Esse protocolo empacota as chamadas e retornos aos métodos dos *Web Services*, sendo utilizado principalmente sobre HTTP (RECKZIEGEL, 2006).

Conforme Seely (2002, p. 43), o SOAP é um mecanismo de comunicação entre as aplicações em uma plataforma descentralizada e distribuída. A especificação do SOAP é dividida em quatro partes principais:

a) SOAP *envelope*: define uma plataforma para descrição do que há na mensagem e como processá-la. Ela guarda todos os dados da mensagem e é a única parte do protocolo que é obrigatória;

b) SOAP *enconding rules*: define um mecanismo de serialização que pode ser usado para a troca de instâncias de tipos definidos pela aplicação;

c) SOAP RPC *style:* define uma convenção que pode ser usada para representar chamadas e respostas remotas a procedimentos, nada mais que a dupla requisição/resposta, que não é obrigatória;

d) link SOAP-HTTP: define um protocolo que liga o SOA e o HTTP. Ele descreve como mensagens SOAP são transmitidas via HTTP.

2.4.3 WSDL

Web Service Definition Language define um sistema para descrição de serviços. Por meio desta linguagem são descritos os serviços e interfaces oferecidos por uma determinada aplicação, assim como a sua

localização. Como outras tecnologias para *Web Services*, sua especificação também é baseada no XML (RECKZIEGEL, 2006).

Em um documento WSDL existem elementos que formam sua especificação. Estes elementos servem para definir os parâmetros de um determinado serviço.

Para Reckziegel (2006), os elementos deste documento estão definidos como:

a) *types* (tipos): container de definição de tipos de dados;

b) *message* (mensagem): define parâmetros de entrada e saída de um serviço;

c) *operation* (operações): são as definições dos métodos, relação entre parâmetros de entrada e saída;

d) *port type* (tipo de porta): descreve o agrupamento lógico das operações, ou seja, das definições dos métodos;

e) *binding* (vínculo): especifica o protocolo e formato de dado a ser usado para um dado *port type*;

f) *service* (serviço): define a localização real do serviço. É uma coleção que pode conter várias portas, e cada uma é especificada para um tipo de vínculo.

Como em qualquer XML, o WSDL também precisa de um elemento raiz. Este elemento chama-se *definitions.* O WSDL utiliza *namespaces* para aumentar a reutilização dos elementos/componentes definidos em seu documento. Estes *namespaces* são espaços para nomes definidos no interior de um XML, o que permite sua unicidade de nomes.

3. DESENVOLVIMENTO

O sistema deste trabalho é formado por dois módulos, um aplicativo e um *Web Service*. O aplicativo é executado no dispositivo móvel e faz a interação do sistema com o usuário. Por meio deste é que o usuário fará todas as operações no sistema. O *Web Service* é executado no servidor e é responsável por executar as requisições do aplicativo cliente. Toda comunicação entre o aplicativo e o servidor é realizada por meio do *Web Service*, pelo protocolo SOAP.

O *Web Service* foi desenvolvido na linguagem Java, utilizando o Java SDK na versão 6, disponível no ambiente de desenvolvimento *Netbeans 7.2.1.* O Java SDK é composto por dois pacotes de software o JDK (*Java Development Kit*) e o JRE (*Java Runtime Enviroment*), ambos contém ferramentas necessárias para o desenvolvimento e execução de aplicações feitas em Java.

A aplicação Android foi desenvolvida, também, utilizando a linguagem Java, juntamente com alguns comandos de HTML, disponíveis na plataforma *Eclipse SDK 4.2*, utilizando-se o Android SDK. Além disso foi necessária a instalação de um *Plug-in* ADT, que faz a integração do Eclipse SDK ao Android SDK e um emulador de celulares para rodar e testar as aplicações desenvolvida.

Este trabalho fez uso de UML (*Unified Modeling Language)* para elaboração da documentação resumida e análise do projeto dos sistemas *Web Service* e aplicativo Android.

Os testes realizados foram feitos por meio do emulador do Eclipse e de um *smartphone*, o que será melhor detalhado na sessão 4.

Na sequência será apresentada a especificação do trabalho.

3.1 Especificação

Nesta seção serão apresentadas as especificações do sistema apresentado. Para especificação dos requisitos foram utilizadas as técnicas da UML através da ferramenta *Astah Community*, para a construção dos diagramas de atividades, diagramas de casos de uso e suas descrições, diagramas de classes e diagramas de sequências

3.1.1 DIAGRAMAS DE ATIVIDADES

O Diagrama de Atividades foi desenvolvido para facilitar a visualização de determinadas situações do sistema.

É apresentado na figura 7, a seguir, o Diagrama de Atividade do caso de uso Enviar Arquivos:

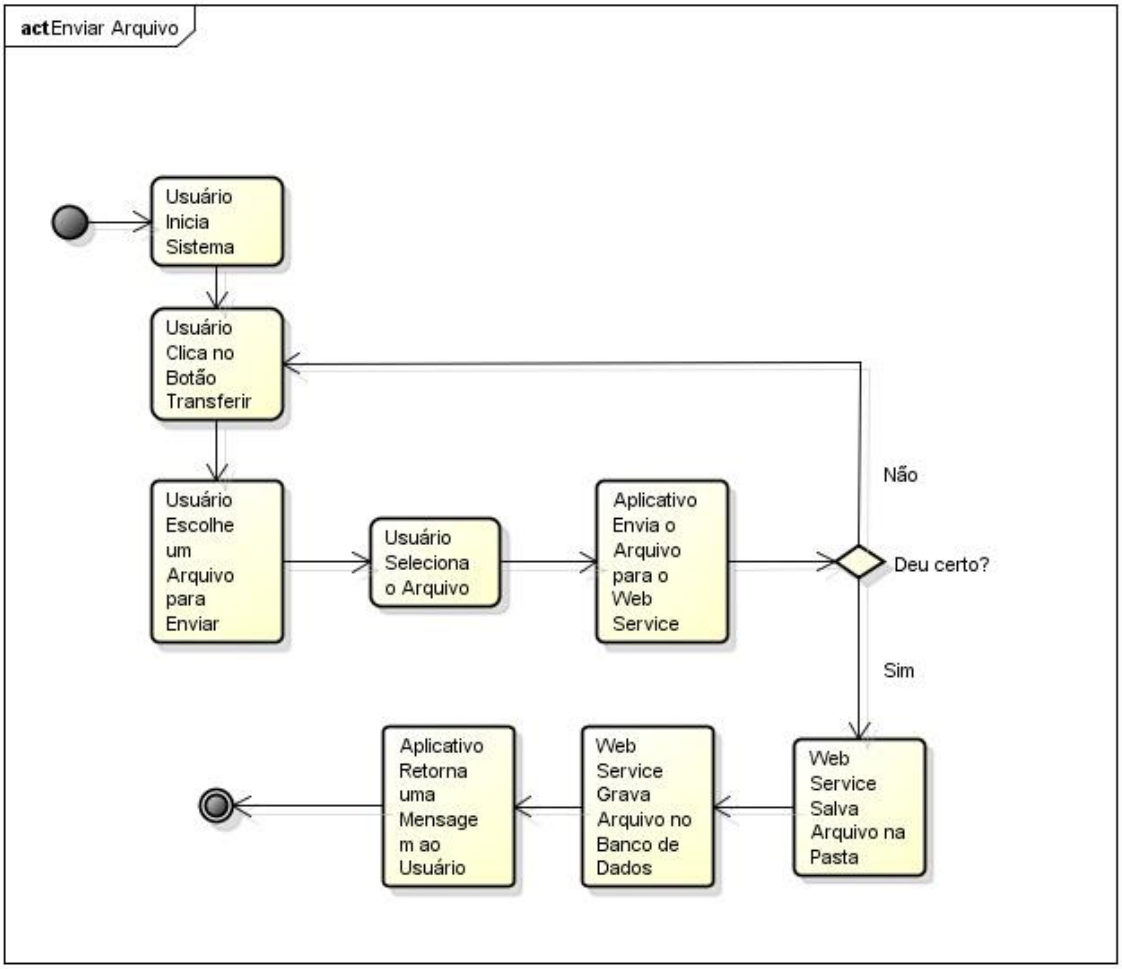


Figura 7 - Diagrama de Atividade - Enviar Arquivo

No Diagrama apresentado, o usuário ao iniciar o sistema, tem a opção de clicar no botão Transferir. O sistema mostra na tela todos os arquivos disponíveis no cartão de memória e que podem ser enviados para o servidor. O usuário escolhe o arquivo e seleciona. Automaticamente o sistema envia o arquivo selecionado para o *Web Service*, que, no caso, funciona como um servidor. Se o envio der certo, o *Web Service* salva o arquivo na pasta do servidor e grava no banco de dados, retornando uma mensagem, na tela do dispositivo, ao usuário informando que a operação deu certo. Se não der certo, retorna a mensagem informando que deu errado e o usuário volta a fazer todos os passos anteriores.

O Diagrama apresentado na figura 8, mostra o processo de retornar um arquivo, conforme o caso de uso Retornar Arquivo:

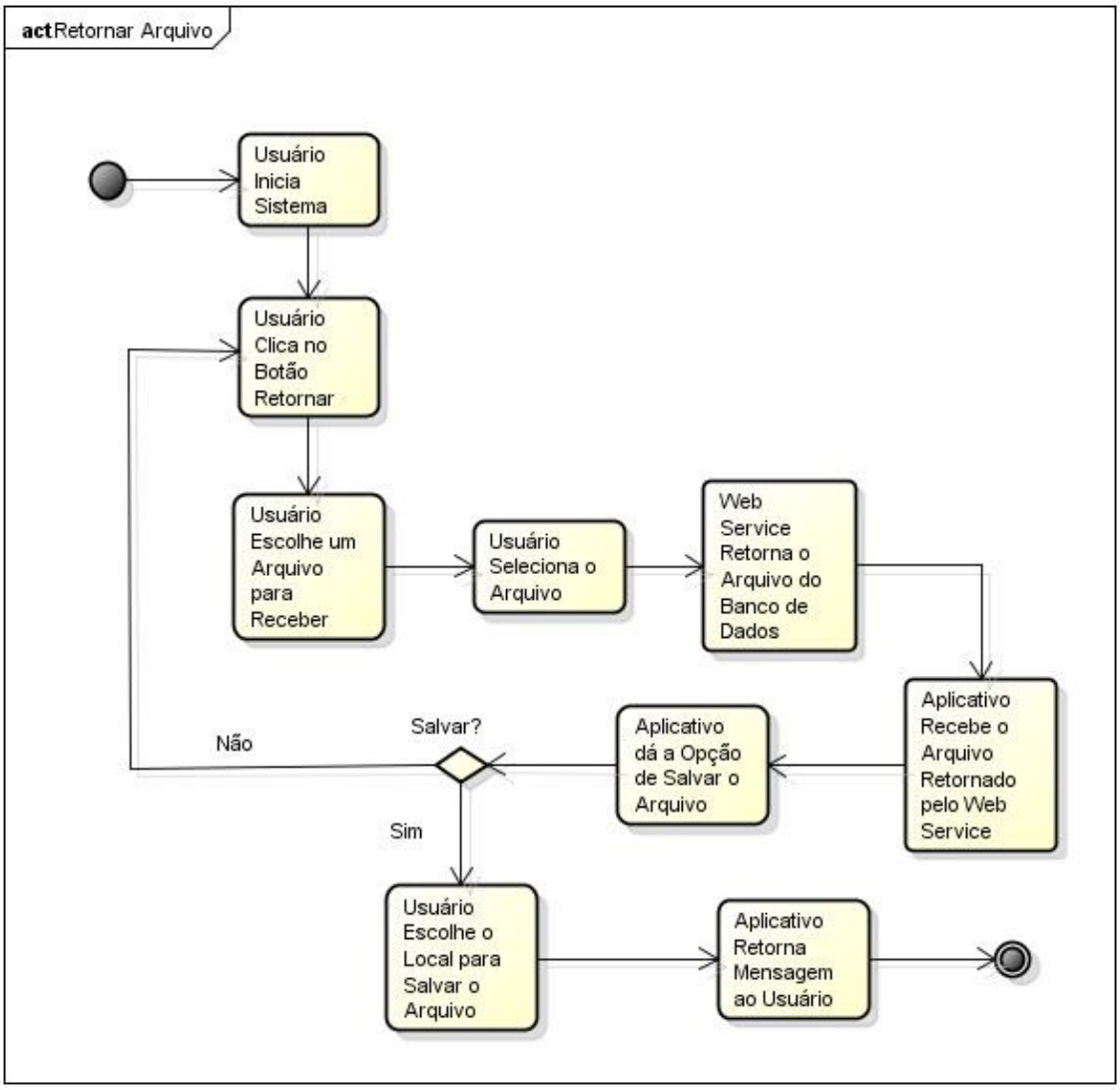


Figura 8 - Diagrama de Atividade - Retornar Arquivo

No Diagrama apresentado, o usuário ao iniciar o sistema, tem a opção de clicar no botão Retornar. O sistema retorna uma lista com todos os arquivos que foram enviados para o *Web Service,* salvos e gravados no banco de dados. Assim, o usuário pode escolher e selecionar um arquivo para retornar para o dispositivo. O Web Service então, retorna o arquivo selecionado do banco de dados para o dispositivo dando a opção de salvar o arquivo novamente. Se o usuário desejar pode salvar o arquivo no local desejado, e então o sistema retorna uma mensagem na tela do dispositivo informando que a operação deu certo e que o arquivo foi salvo. Se o usuário não quiser salvar ou escolheu um arquivo errado para retornar, ele pode voltar a executar todos os passos anteriores.

3.1.2 DIAGRAMAS DE CASOS DE USO

São demonstrados os principais Casos de Uso do usuário do sistema, do servidor *Web Service* e da aplicação cliente.

O Usuário possui três Casos de Uso, conforme é mostrado na figura 9, abaixo:

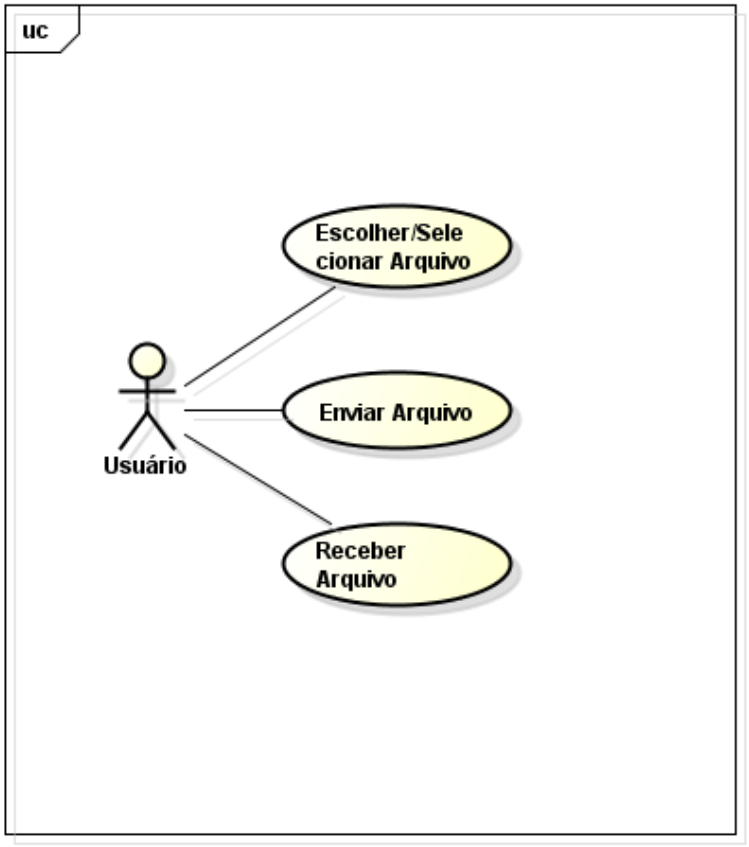


Figura 9 - Diagrama de Caso de Uso - Usuário

a) Escolher/Selecionar Arquivo: responsável pela escolha e seleção do arquivo para ser enviado.
b) Enviar Arquivo: responsável pelas operações de envio automático do arquivo no momento em que é selecionado.
c) Receber Arquivo: responsável pelas operações de receber de volta no dispositivo um arquivo que foi escolhido pelo usuário.

FLUXO PRINCIPAL:

1. O usuário entra no sistema e escolhe qual opção deseja executar.
2. Ao escolher a opção de “Transferir Arquivo”, aparecerá uma tela contendo todos os arquivos disponíveis para envio, e então o usuário seleciona qual arquivo deseja enviar. [FS1] [FE1] [RN1]
3. Selecionando a opção de “Retornar Arquivo”, aparecerá uma outra tela contendo todos os arquivos que foram enviados para o servidor, assim, o usuário poderá escolher qual arquivo deseja retornar e então escolher onde deseja abri-lo. [FS2] [FE2] [RN2]

FLUXO SECUNDÁRIO:

FS1. Caso não seja encontrado nenhum arquivo para envio, o usuário precisa verificar se há algum cartão de memória contento arquivos.

FS2. Caso não seja encontrado nenhum arquivo para retornar, o usuário precisa verificar se já foram enviados arquivos para o servidor.

FLUXO DE EXCEÇÃO:

FE1. Se não for possível enviar o arquivo e houver um erro o sistema informa para o usuário.

FE2. Se não for possível retornar o arquivo e houver erro o sistema informa para o usuário.

REGRAS DE NEGÓCIO:

RN1. O usuário só poderá transferir arquivos se houver arquivos disponíveis para envio e se o servidor estiver em execução.

RN2. O usuário só poderá retornar arquivos se houver arquivos disponíveis no banco de dados e se o servidor estiver em execução.

O Servidor possui quatro Casos de Uso, conforme é apresentado na figura 10, abaixo:

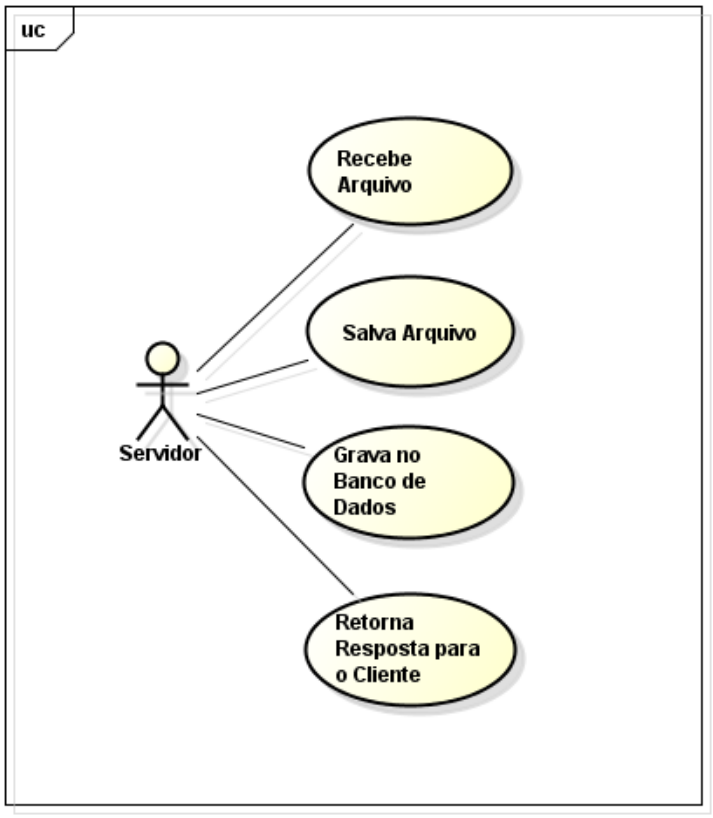


Figura 10 - Diagrama de Caso de Uso - Servidor

a) Receber Arquivo: responsável por receber as requisições feitas pelo cliente, junto com o arquivo.
b) Salvar Arquivo: responsável por executar as operações para salvar o arquivo em determinada pasta do servidor.
c) Gravar no Banco de Dados: responsável por executar as operações que gravam o arquivo, após ser salvo, no banco de dados Postgre.

d) Retornar resposta para o Cliente: responsável por executar as requisições recebidas e retornar uma mensagem para cliente, após a execução, proveniente da operação que foi realizada.

FLUXO PRINCIPAL:

1. O Servidor é conectado. [FE1]
2. O Servidor recebe as opções de requisição de receber um arquivo. [FS1] [FE2]
3. O Servidor salva o arquivo recebido na pasta do sistema e grava automaticamente no banco de dados. [FS2] [RN1] [FE3]
4. O Servidor envia uma resposta para o cliente informando se a operação deu certo ou não. [RN2]

FLUXO SECUNDÁRIO:

FS1. Caso o servidor não receba as requisições e/ou o arquivo é necessário verificar se o arquivo é válido para envio e verificar a conexão do servidor.

FS2. Caso o servidor não consiga gravar o arquivo nas devidas pastas, é preciso verificar a conexão e ver se há permissão para acessar tais pastas.

FLUXO DE EXCEÇÃO:

FE1. Se o Servidor não estiver conectado, não é possível fazer a transferência, é necessário tentar uma nova conexão e verificar se há conexão com a Internet disponível.

FE2. O Servidor só recebe as requisições se estiver conectado.

FE3. Só é possível receber, salvar e gravar o arquivo se o servidor estiver conectado corretamente.

REGRAS DE NEGÓCIO:

RN1. O Servidor só pode salvar e gravar automaticamente um arquivo, após receber a requisição com a escolha do mesmo pelo usuário, e se houver conexão.

RN2. O Servidor só envia respostas depois que terminar de executar a operação desejada pelo cliente.

A Aplicação Cliente possui quatro Casos de Uso, conforme é apresentado na figura 11, abaixo:

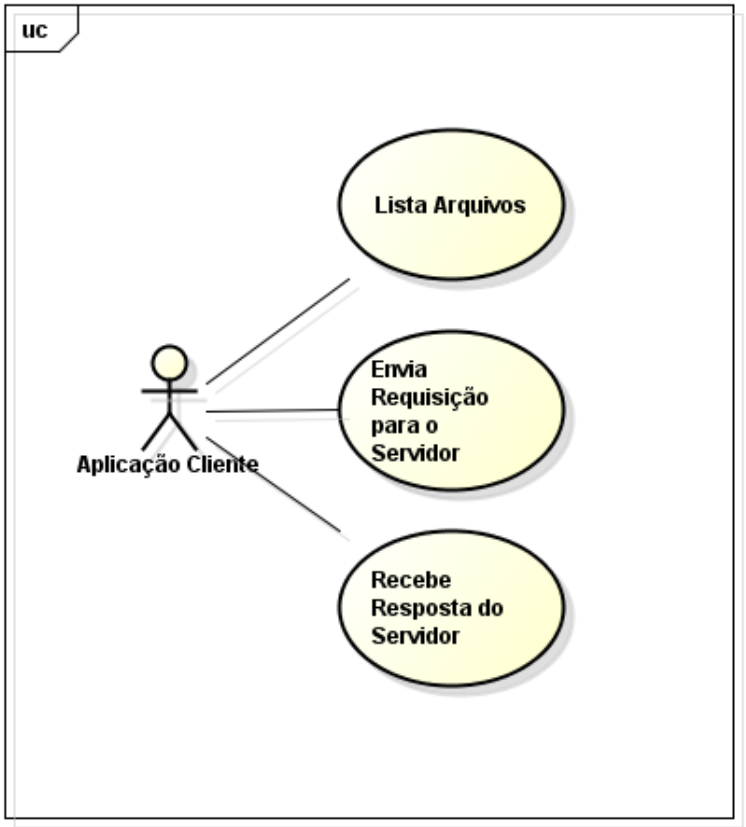


Figura 11 - Diagrama de Caso de Uso - Aplicação Cliente

a) Listar Arquivos: responsável por executar a operação que lista todos os arquivos existentes no cartão de memória do dispositivo, e no banco de dados.
b) Enviar Requisição para o Servidor: responsável por processar as solicitações feitas pelo usuário, e se comunicar com o servidor.

c) Receber Resposta do Servidor: responsável por processar as respostas vindas do servidor, exibindo uma mensagem de confirmação ou erro das operações.

FLUXO PRINCIPAL:

1. A aplicação lista os arquivos assim que o usuário escolher qual a opção deseja executar, transferir ou retornar. [FS1] [RN1] [FE1]
2. Conforme a opção escolhida pelo cliente, a aplicação envia as requisições para o servidor. [FS2] [RN2]
3. O Servidor se estiver conectado corretamente, executa as requisições e retorna a resposta para a aplicação. [FS3] [RN3]

FLUXO SECUNDÁRIO:

FS1. Caso o usuário escolha a opção para transferir, a aplicação lista os arquivos disponíveis no cartão de memória do dispositivo que esteja pronto para enviar. Caso o usuário escolha a opção de retornar, a aplicação lista os arquivos que já foram enviados e estão salvos, gravados e disponíveis para retornar.
FS2. Caso a aplicação não envie as requisições para o servidor, é necessário verificar se há uma conexão válida entre ambos.
FS3. Caso o servidor não retorne a resposta para a aplicação, é necessário verificar se há uma conexão, ou se o arquivo que foi escolhido é válido.

FLUXO DE EXCEÇÃO:

FE1. Se a aplicação não listar os arquivos, é preciso verificar se há arquivos disponíveis para envio ou retorno.

REGRAS DE NEGÓCIO:

RN1. A aplicação só lista os arquivos se o usuário escolher qual das opções deseja executar, transferência ou retorno.

RN2. A aplicação só envia as requisições para o servidor, se este estiver conectado e se o usuário escolher uma opção.
RN3. O servidor só retorna a resposta para a aplicação se estiver conectado e se executado as requisições conforme a opção escolhida pelo cliente.

3.1.3 DIAGRAMAS DE CLASSES

O projeto foi desenvolvido baseado na orientação objeto, utilizando o Diagrama de Classes UML como linguagem de modelagem.

O Diagrama de Classe apresentado na figura 12, abaixo, mostra as classes existentes no projeto do *Web Service*:

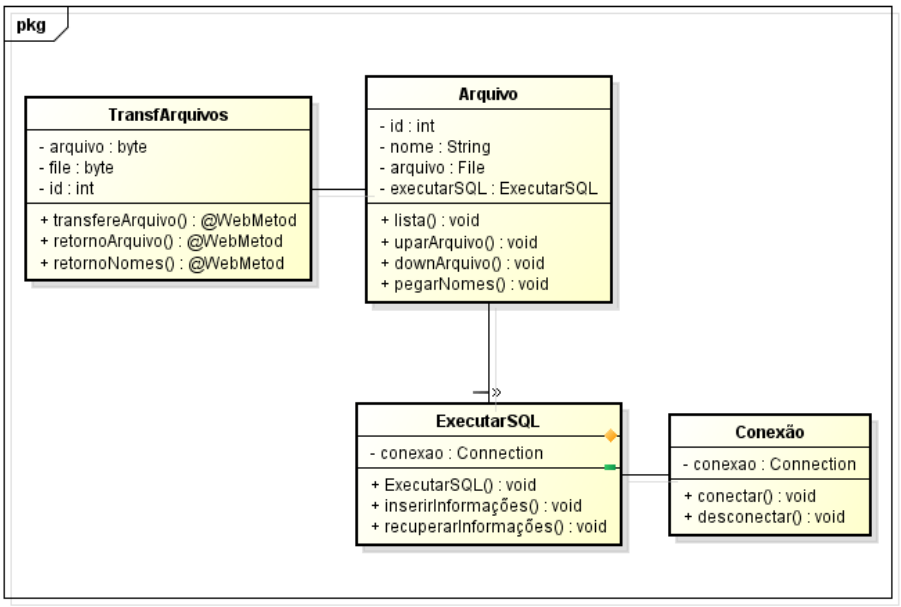


Figura 12 - Diagrama de Classes - Web Service

O Diagrama de Classe apresentado na figura 13, abaixo, mostra as classes existentes no projeto do Aplicativo Android:

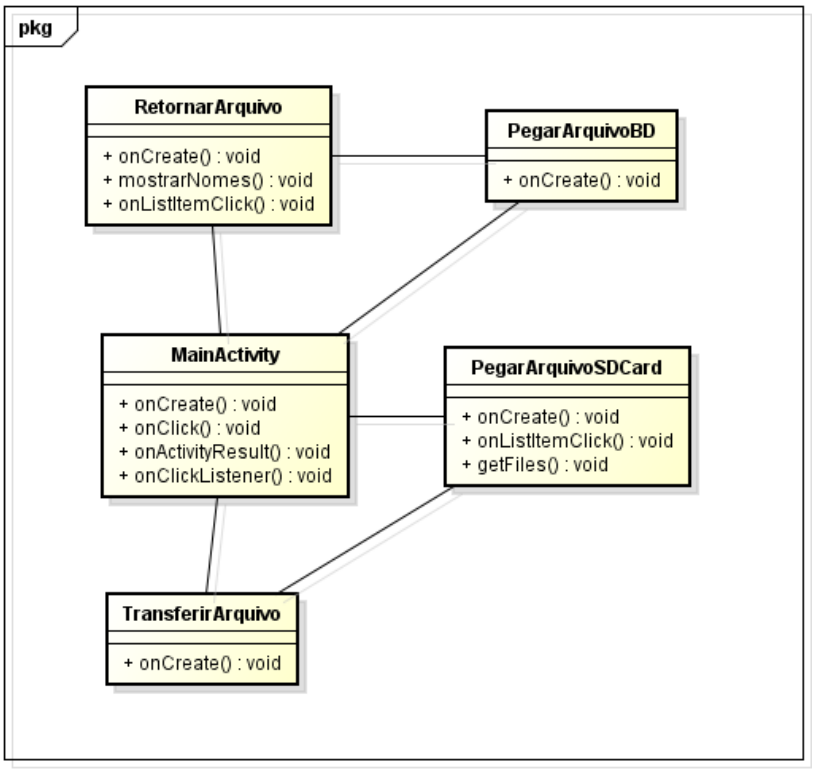


Figura 13 - Diagrama de Classes - Aplicação Android

3.1.4 DIAGRAMAS DE SEQUENCIA

O Diagrama de Sequência é um diagrama usado pela UML e representa a sequência dos processos e as mensagens transmitidas entre os objetos.

É apresentado na figura 14, o Diagrama de Sequência do caso de uso Enviar Arquivos:

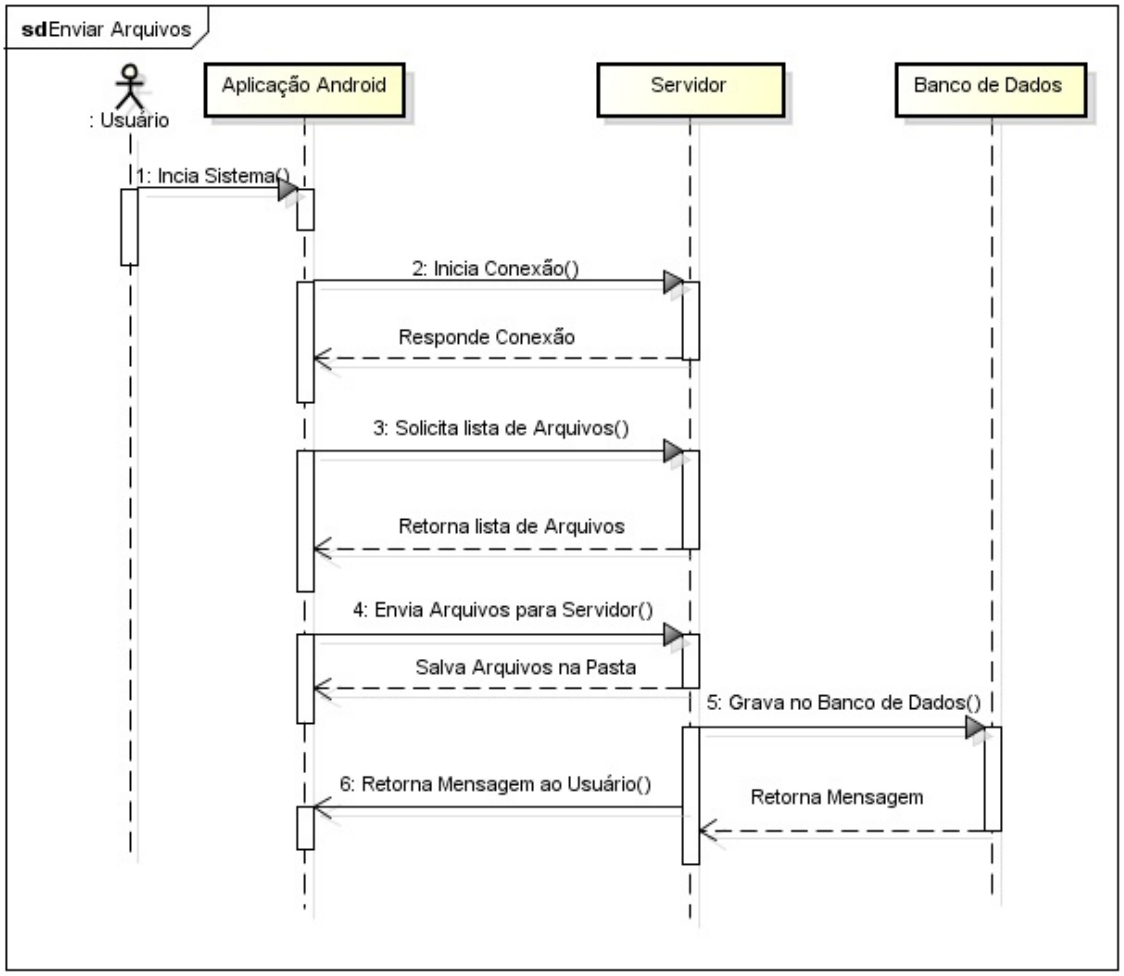


Figura 14 - Diagrama de Sequência - Enviar Arquivos

É apresentado na figura 15, a seguir, o Diagrama de Sequência do caso de uso Retornar Arquivos:

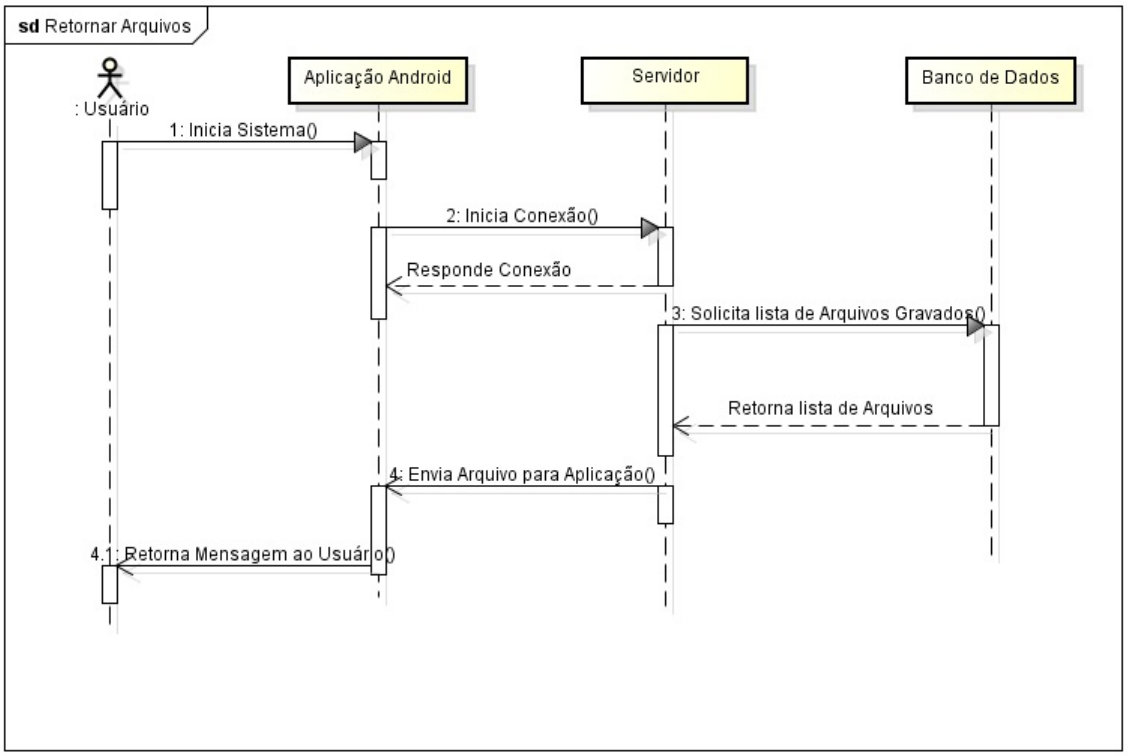


Figura 15 - Diagrama de Sequência - Retornar Arquivos

3.1.5 MODELO DE ENTIDADE RELACIONAL (MER)

O MER tem como objetivo representar de forma visual as estruturas de dados de um banco de dados. Para desenhar o MER deste sistema foi utilizado a ferramenta DB *Designer*.

Como banco de dados, é utilizado o *PostgreSQL*, na versão 9.1. È apresentado na figura 16, o MER da estrutura do *Web Service*:

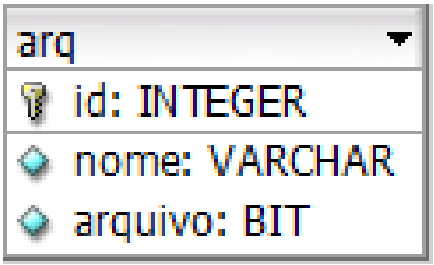


Figura 16 - MER

Na tabela arq são armazenadas as informações de todos os arquivos que são enviados para o servidor, como a id do arquivo e o seu nome e o próprio arquivo fica gravado na coluna arquivo. Por meio desta tabela é possível que o usuário consiga retornar uma lista com todos os arquivos que estão gravados e disponíveis para retornar no dispositivo.

4. RESULTADOS OBTIDOS

Este trabalho atingiu o seu objetivo proposto. O sistema é capaz de fazer a transferência de arquivos entre o dispositivo móvel e o servidor e também de retornar um arquivo do servidor para o dispositivo. O sistema executa as requisições por meio da escolha do usuário e registra no banco de dados o que foi transferido.

O desempenho do sistema se mostrou satisfatório, pois não há demora para carregar a aplicação e nem para carregar a árvore de arquivos e pastas disponíveis no cartão de memória e no banco de dados, além disso todos os arquivos foram enviados e retornados sem perda nenhuma de informações.

A aplicação foi testada no emulador do Android *platform* 2.3.3, *API Level* 10, com as configurações de *device* 3.3” WQVGA (240x400:lpdi), *memory ram* 512, *internal storage* 500 MB e *SD card* 500 MB, que foi criado no *Edit Android Virtual Devic*e (AVD) disponível na plataforma de desenvolvimento *Eclipse.* Além disso, também foi testada no *smartphone Samsung Galaxy Y*, Android 2.3. Os testes foram feitos com arquivos de diferentes formatos: .txt, .doc, .pdf e .jpg, sendo casa um deles nos tamanhos:

- entre 1 e 30 KB;
- entre 30 e 60KB;
- entre 60 e 100 KB;
- entre 100 e 500 KB;

- entre 500 KB e 1MB e,
- entre 1 e 3 MB.

A seguir será apresentada as tabelas, contento o resultados dos testes realizados com a aplicação.

EMULADOR	Tipos de Arquivos/Transferir				Tipos de Arquivos/Retornar			
	TXT	PDF	DOC	JPG	TXT	PDF	DOC	JPG
Tamanhos								
entre 1 e 30KB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 30 e 60KB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 60 e 100K	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 100 e 500KB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 500KB e 1MB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 1 e 3MB	Falha	Falha	Falha	Falha	/	/	/	/

Figura 17 - Teste de Envio e Retorno no Emulador

Conforme é apresentado na figura 17, o teste realizado no emulador, se mostrou satisfatório apenas com arquivos até 1MB de diferentes formatos. Os arquivos foram enviados e retornados sem perder nenhuma informação. Para cada formato e tamanho foram realizados três testes em diferentes momentos, todos apresentaram sucesso na execução, apesar da demora com relação ao envio de arquivos acima de 500KB. Como pode-se notar, arquivos acima de 1MB o emulador não suporta e consequentemente não faz o envio e nem o retorno dos arquivos.

SMARTPHONE	Tipos de Arquivos/Transferir				Tipos de Arquivos/Retornar			
	TXT	PDF	DOC	JPG	TXT	PDF	DOC	JPG
Tamanhos								
entre 1 e 30KB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 30 e 60KB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 60 e 100KB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 100 e 500KB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 500KB e 1MB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado
entre 1 e 3MB	Enviado	Enviado	Enviado	Enviado	Retornado	Retornado	Retornado	Retornado

Figura 18 - Teste de Envio de Retorno no Smartphone

Conforme é apresentado na figura 18, o teste realizado no *smartphone*, se mostrou mais eficiente e satisfatório do que o teste realizado no emulador. Foram enviados e retornados, os arquivos, sem perder nenhuma informação. Para cada formato e tamanho, também foram realizados três testes em diferentes momentos e todos apresentaram sucesso na execução. A

vantagem do teste no *smartphone* é que ele suporta arquivos acima de 1MB e faz o envio e o retorno sem problema algum, além disso o tempo de transferência é menor do que o do emulador, mas isso não é o foco do teste, é apenas para conhecimento e diferenciação dos testes.

Com base nessas informações, será mostrado, a seguir, o funcionamento e estrutura do *Web Service* e do aplicativo Android.

4.1 Estrutura do *Web Service*

O servidor foi criado como um *Web Service* na ferramenta NetBeans 7.1 e foi publicado por meio da plataforma Apache Tomcat 7.0. Na figura 19, abaixo, é mostrada a pasta principal do Servidor.

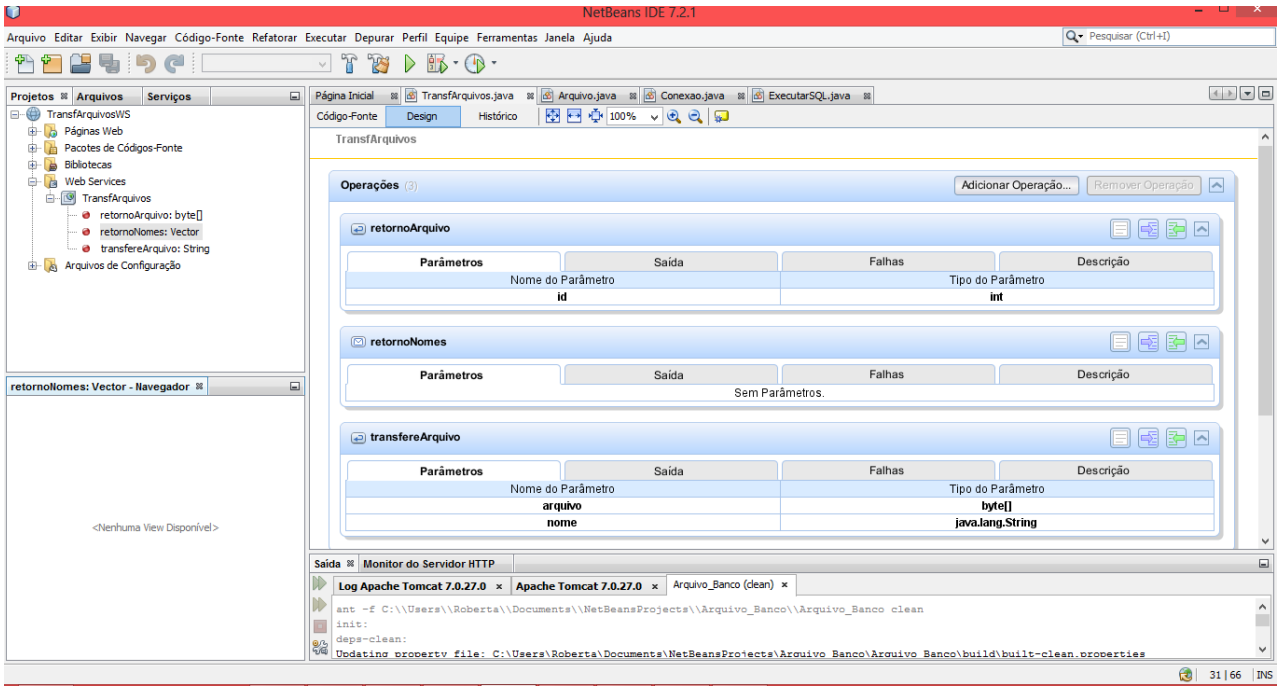


Figura 19 - Estrutura do Web Service

Dentro do pacote “org.me.transferir”, existe o servidor com nome de “TransfArquivos”, o qual declara três métodos que executam as principais operações e serviços do *Web Service*, estes são: “retornoArquivo” , “retornoNomes” e “transfereArquivo”. Na figura 20, é mostrada a principal parte do código com o conteúdo que executa tais operações.

```
1  package org.me.transferir;
   import ...
14
15 @WebService(serviceName = "TransfArquivos")
16 public class TransfArquivos {
17
18     @WebMethod(operationName = "transfereArquivo")
19     public String ja(@WebParam(name = "arquivo") byte[] arquivo,@WebParam
20             (name = "nome") String nome ) {
21         try {
22             File file = new File(nome);
23             BufferedOutputStream bos = null;
24             bos = new BufferedOutputStream(new FileOutputStream(file));
25             bos.write(arquivo);
26             bos.close();
27             Arquivo arq = new Arquivo();
28             arq.setArquivo(new File(nome));
29             arq.setNome(nome);
30             arq.uparArquivo();
31             return "Arquivo enviado com êxito!";
32          }  catch (Exception ex) {
33             return ex.getMessage(); }}
34
35     @WebMethod(operationName = "retornoArquivo")
36     public byte[] operação(@WebParam(name = "id") int id) {
37         Arquivo arq = new Arquivo();
38         arq.setId(id);
39         byte[] Bytes = arq.downArquivo();
40         return Bytes;  }
41
42     @WebMethod(operationName = "retornoNomes")
43     public java.util.Vector retornoNomes() {
44         Arquivo arq = new Arquivo();
45         Vector nomes = arq.pegarNomes();
46         return nomes; }}
```

Figura 20 - Código das Operações do Web Service

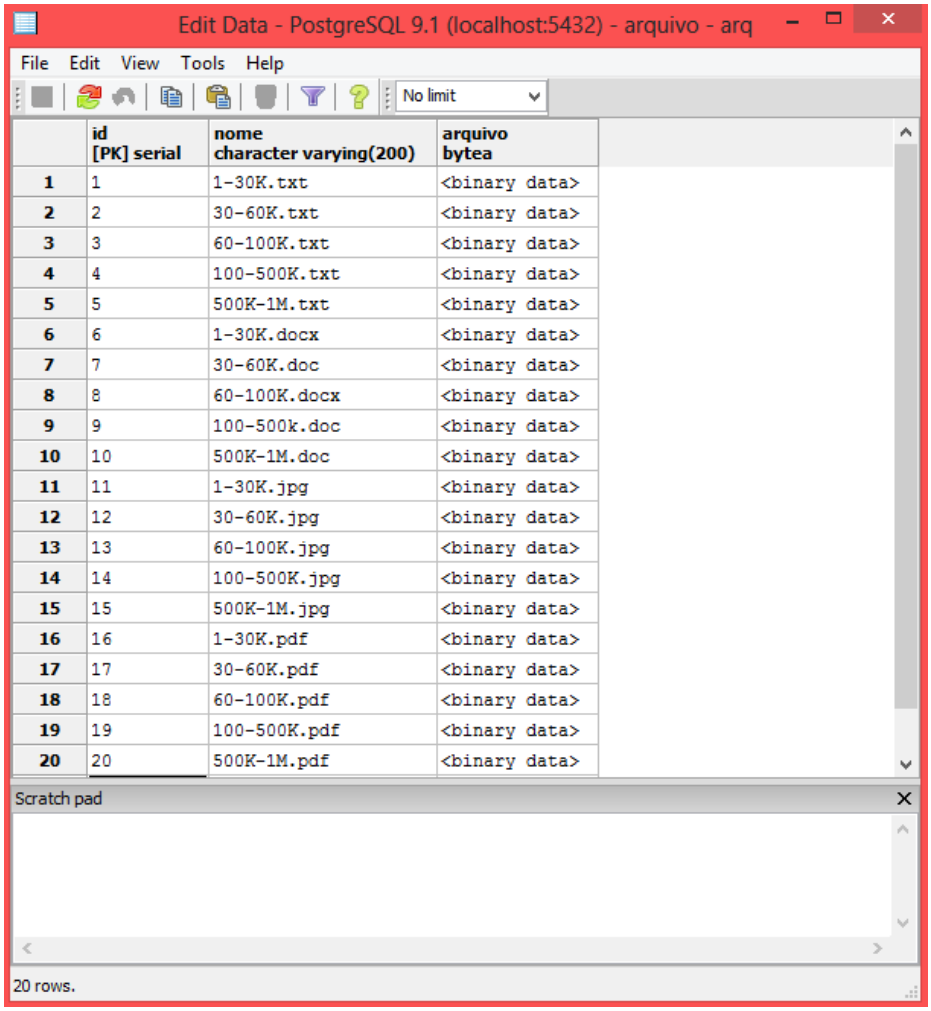
Conforme a figura, os métodos são anotados com *@WebMethod* que indica ao servidor de aplicação que este método será exposto para os cliente como um serviço que poderá ser consumido. Os parâmetros dos métodos também são anotados com *@WebParam(name="")*, porque algumas aplicações, como o Android, não são capazes de encontrar os parâmetros se não estiverem com tais anotações.

Seguindo a figura, na linha 18 é criado primeiro método do *Web Service*, "transfereArquivo" que é responsável por transferir arquivos. Dentro deste método, na linha 22 é criado um nome para o arquivo que será recebido pelo servidor, seguido pela linha 24, onde é criado o próprio arquivo e por meio da linha 25 é gravado os *bytes* do arquivo na pasta do servidor. Na linha 27 é recebido o arquivo e na linha 30, o método acessa a classe Arquivo,

conforme mostrada na figura 12, e executa o método “uparArquivo” que é responsável por executar as operações de transferência e então acessar o banco de dados.

Para retornar os arquivos são utilizados os métodos “retornoArquivo” e “retornoNomes”. Por meio da linha 39 é acessada a classe Arquivo do sistema e o método “downArquivo”, o qual executa as operações de retornar o arquivo do servidor e do banco de dados para o dispositivo móvel. Com o método “retornoNomes” da linha 42, é acessada novamente a classe Arquivo e executado o método “pegarNomes”, o qual é responsável por retornar um vetor contendo os nomes e os id’s dos arquivos disponíveis para retorno e mostrar para o servidor, conforme a linha 46.

Os arquivos que são transferidos ficam são gravados automaticamente no banco de dados Postgre. Na figura 21, é mostrada onde os arquivos transferidos ficam salvos.



	id [PK] serial	nome character varying(200)	arquivo bytea
1	1	1-30K.txt	<binary data>
2	2	30-60K.txt	<binary data>
3	3	60-100K.txt	<binary data>
4	4	100-500K.txt	<binary data>
5	5	500K-1M.txt	<binary data>
6	6	1-30K.docx	<binary data>
7	7	30-60K.doc	<binary data>
8	8	60-100K.docx	<binary data>
9	9	100-500k.doc	<binary data>
10	10	500K-1M.doc	<binary data>
11	11	1-30K.jpg	<binary data>
12	12	30-60K.jpg	<binary data>
13	13	60-100K.jpg	<binary data>
14	14	100-500K.jpg	<binary data>
15	15	500K-1M.jpg	<binary data>
16	16	1-30K.pdf	<binary data>
17	17	30-60K.pdf	<binary data>
18	18	60-100K.pdf	<binary data>
19	19	100-500K.pdf	<binary data>
20	20	500K-1M.pdf	<binary data>

Figura 21 - Banco de Dados

4.2 Aplicativo Android

A tela apresentada ao abrir o aplicativo, tem em sua interface dois botões “Transferir” e “Retornar”, os quais são os responsáveis por todas as funcionalidades do sistema. Na figura 22, é apresentada essa interface inicial do aplicativo, executando no emulador Android.

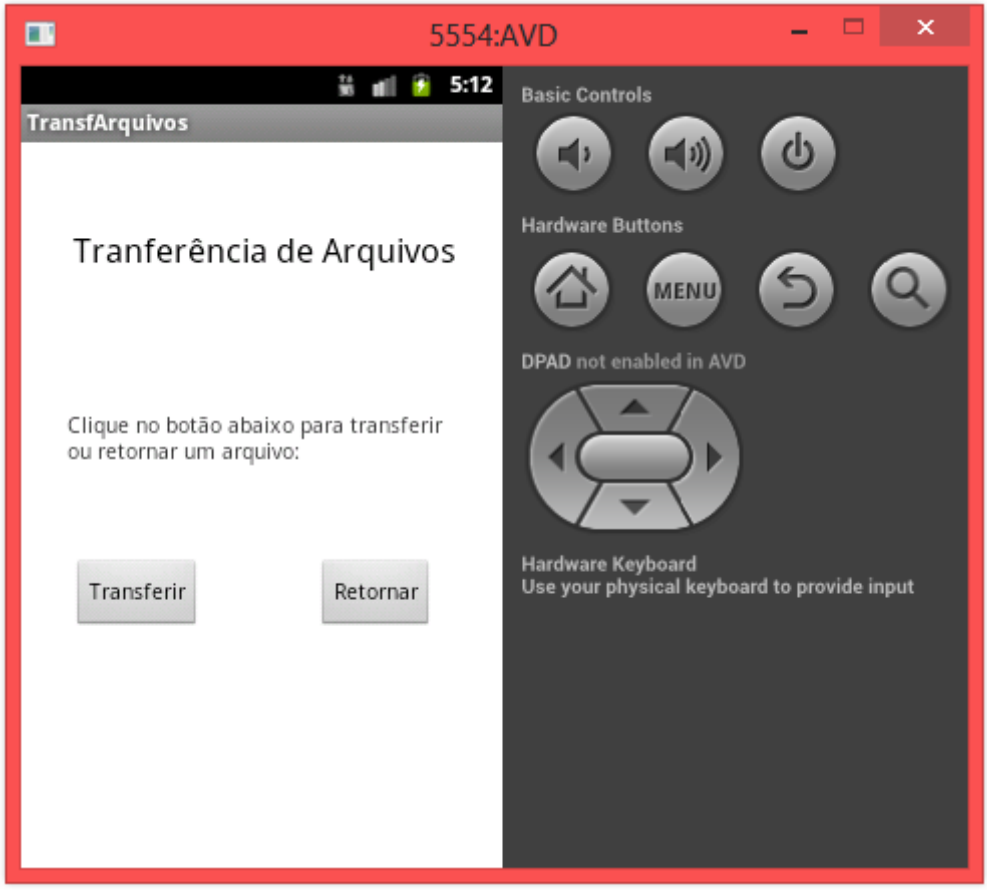


Figura 22 - Tela Principal do Sistema

Selecionando o botão “Transferir”, o usuário poderá escolher qual arquivo deseja enviar para o servidor. Isso acontece, pois todos os arquivos que estão guardados no cartão de memória do dispositivo, aparecem listados em uma nova tela, permitindo que o usuário selecione qual deseja enviar. Na figura 23, é mostrada a tela onde são listadas as pastas contendo os arquivos disponíveis para envio.

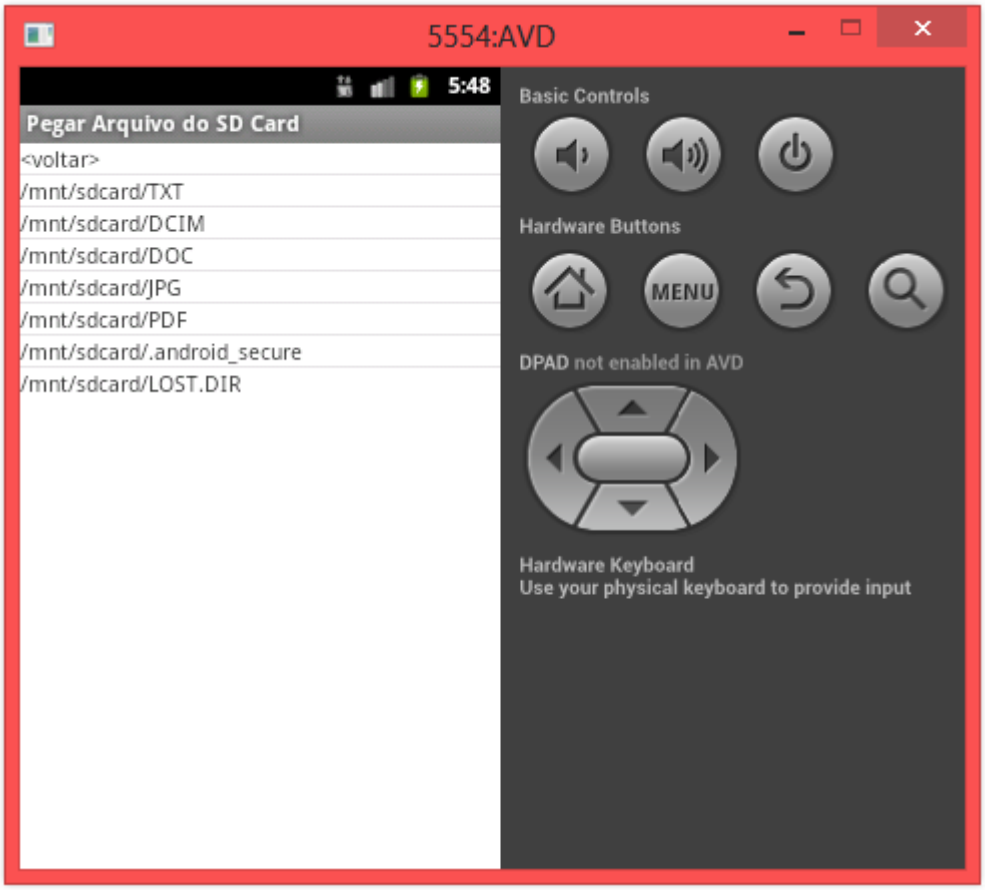


Figura 23 - Tela com Lista das Pastas que contém arquivos para envio

A outra opção que o usuário possui, é selecionar o botão "Retornar", assim ele poderá escolher qual arquivo, que já foi enviado para servidor, deseja voltar para seus documentos no dispositivo. Isso é possível pois ao enviar um arquivo para o servidor, ele é gravado automaticamente no banco de dados, possibilitando o aplicativo acessar diretamente o banco para recuperar esses arquivos. Ao clicar no botão, aparecem listados em uma nova tela todos os arquivos que já foram enviados e estão disponíveis para retorno. Na figura 24, é mostrada a tela onde são listados os arquivos disponíveis para retornar para o dispositivo.

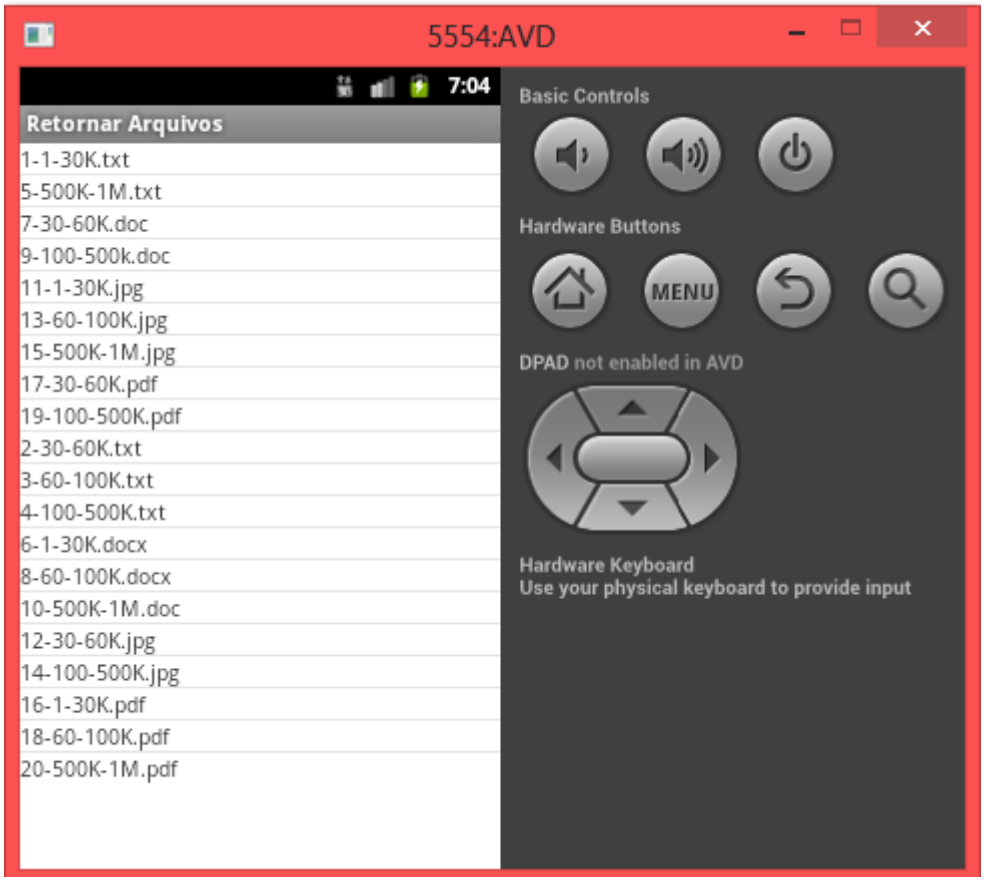


Figura 24 - Tela que lista os arquivos disponíveis para retorno

Após o término de qualquer uma das operações, o aplicativo retorna uma mensagem ao usuário informando se não deu certo, ou não, a operação que executou. É possível retornar a qualquer momento para tela inicial, se o usuário quiser executar as duas operações seguidamente.

Para que todas essas funcionalidades sejam executadas corretamente, são necessários alguns códigos. Na figura 25, a seguir, é apresentado o arquivo *AndroidManifest.xml*, onde está declarada todas as *Activities* e permissões que serão iniciadas com o aplicativo

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.transfarquivos"
4      android:versionCode="1"
5      android:versionName="1.0" >
6
7      <uses-sdk
8          android:minSdkVersion="8"
9          android:targetSdkVersion="17" />
10     <uses-permission android:name="android.permission.INTERNET"/>
11     <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
12     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
13     <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
14     <application
15         android:allowBackup="true"
16         android:icon="@drawable/ic_launcher"
17         android:label="@string/app_name"
18         android:theme="@style/AppTheme" >
19         <activity
20             android:name="com.example.transfarquivos.MainActivity"
21             android:label="@string/app_name" >
22             <intent-filter>
23                 <action android:name="android.intent.action.MAIN" />
24                 <category android:name="android.intent.category.LAUNCHER" />
25             </intent-filter>
26         </activity>
27         <activity android:label="Pegar Arquivo do SD Card" android:name=".PegarArquivoSDCard"></activity>
28         <activity android:name=".Tela2" android:label="Tela2"></activity>
29          <activity android:name=".RetornarArquivo" android:label="Retornar Arquivos"></activity>
30         <activity android:name=".TransferirArquivo" android:label="Transferir Arquivos"></activity>
31          <activity android:name=".ListarArquivos" android:label="Listar Arquivos"></activity>
32          <activity android:name=".PegarArquivoBD" android:label="Pegar Arquivos do BD"></activity>
33   </application> </manifest>
```

Figura 25 - Android Manifest

Conforme a figura, nas linhas 10 à 13, são mostradas as declarações das permissões, por meio de uses-permission, que são utilizadas para o funcionamento correto do aplicativo. E das linhas 27 à 32 são mostradas as *Activies* que são iniciadas com o sistema e que executam as suas principais funções.

A classe TransferirArquivos executa as operações de conexão com o servidor para transferir os arquivos que são escolhidos pelo usuário. Na figura 26, é mostrada o início da classe.

```
1  package com.example.transfarquivos;
2
3  import java.io.File;
24
25 @SuppressLint("SdCardPath")
26 public class TransferirArquivo extends Activity {
27     private String METHOD_NAME = "transfereArquivo";
28     private String NAMESPACE = "http://transferir.me.org/";
29     private String SOAP_ACTION = "http://transferir.me.org/transfereArquivo";
30     private static final String URL = "http://192.168.1.100:8084/TransfArquivosWS/TransfArquivos?wsdl";
31
32     @Override
33     public void onCreate(Bundle savedInstanceState) {
34         super.onCreate(savedInstanceState);
35         setContentView(R.layout.activity_transferir_arquivo);
36
37         Intent it = getIntent();
38         Bundle params = null;
39         String msg = "";
```

Figura 26 - Trecho da Classe TransferirArquivo

Por meio das linhas 27 à 30, é feita a comunicação com o Web Service, utilizando os comandos “*METHOD_NAME*” , “*NAMESPACE*” , “*SOAP_ACTION*” e “*URL*”, onde contém as informações de acesso do Web Service. O comando setContentView lê o arquivo *layout* activity_transferir_arquivo e exibe esta *Activty* na tela.

A conexão com o *Web Service* é feita conforme é mostrado no código, da classe TransferirArquivos, na figura 27, abaixo.

```
65
66              SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
67              PropertyInfo pi = new PropertyInfo();
68              pi.setName("arquivo");
69              pi.setValue(byt);
70              pi.setType(MarshalBase64.BYTE_ARRAY_CLASS);
71              request.addProperty(pi);
72
73              pi = new PropertyInfo();
74              pi.setName("nome");
75              pi.setValue(f.getName());
76              request.addProperty(pi);
77
78              SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
79                      SoapEnvelope.VER11);
80              MarshalBase64 marshal = new MarshalBase64();
81              marshal.register(envelope);
82              envelope.dotNet = false;
83              envelope.setOutputSoapObject(request);
84              HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
85              androidHttpTransport.call(SOAP_ACTION, envelope);
86              Object result = envelope.getResponse();
87              System.out.println("Result : " + result.toString());
88              ((TextView) findViewById(R.id.txtAddition)).setText("Addition : "
89                      + result.toString());
90          } catch (Exception E) {
91              E.printStackTrace();
92              ((TextView) findViewById(R.id.txtAddition)).setText("ERROR:"
93                      + E.getClass().getName() + ":" + E.getMessage());
94          }
```

Figura 27 - Código de conexão com Web Service

Na linha 66 é inicializado o protocolo SOAP, o qual representará o encapsulamento da requisição que será feita ao *Web Service*, recebe como parâmetro uma *String* representando *namespace* referente ao *Web Service* e o nome da operação. Nas linhas 78 e 79 é criado o objeto envelope que é a abstração de um envelope SOAP e passa como parâmetro a versão do protocolo SOAP que irá utilizar. O comando *MarshalBase64* na linha 80 indica que foi usada a codificação de 64 bits. Nas linhas 82 e 83 é chamado

o método para encapsular o *request* como corpo do envelope SOAP a ser enviado. Na linha 84 é instanciado um objeto *HttpTransporte* e passa como parâmetro a URL para acessar o *Web Service*. Por fim na linha 85 é chamado o *Web Service* por meio do método *call* e do envelope que já foi construído.

Para pegar os arquivos que estão salvos no cartão de memória do dispositivo móvel, é necessário que estejam disponíveis alguns arquivos, conforme foi explicado na figura 11 . É preciso que a aplicação retorne uma lista contendo estes arquivos e isso é possível conforme é mostrado na figura 28 abaixo, por meio dos métodos onListItemClick e getFiles disponíveis na classe PegarArquivoSDCard.

```
21      @Override
22      public void onCreate(Bundle icicle) {
23          super.onCreate(icicle);
24          setContentView(R.layout.activity_pegararquivo_sdcard);
25          getFiles(new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/").listFiles());
26      }
27      @Override
28      protected void onListItemClick(ListView l, View v, int position, long id){
29          int selectedRow = (int)id;
30          if(selectedRow == 0){
31              getFiles(new File("/").listFiles());
32          }else{
33              File file = new File(items.get(selectedRow));
34              if(file.isDirectory()){
35                  getFiles(file.listFiles());
36              }else{
37
38                  Intent it = new Intent();
39                  it.putExtra("arquivo", file.getAbsolutePath());
40
41                  setResult(1, it);
42                  finish();
43               }}}
44      private void getFiles(File[] files){
45          items = new ArrayList<String>();
46          items.add(getString(R.string.goto_root));
47          for(File file : files){
48              items.add(file.getPath());
49          }
50          ArrayAdapter<String> fileList = new ArrayAdapter<String>(this,R.layout.activity_listar_arquivos, items);
51          setListAdapter(fileList);
52      }
```

Figura 28 - Código de pegar os arquivos do SDCard

O processo para retornar arquivo segue os mesmos passos do processo de transferir arquivo. A conexão com o servidor ocorre da mesma maneira, o que pode ser verificado na figura 26. O que ocorre de diferente, é que ao invés de acessar o cartão de memória do dispositivo, o sistema acessa o banco de dados para retornar uma lista contendo os arquivos que já foram enviados e estão disponíveis para retorno. Na figura 29, é mostrado o código, da classe RetornarArquivo, que executa esse processo, nas linhas 71 à 74 ele acessa um vetor de nomes que foi criado no servidor e nas linhas 86 e 87 ele

retorna a lista contendo os id's e os nomes dos arquivos que estão disponíveis para retornar.

```
71    private void mostrarNomes(Vector nomes){
72        items = new ArrayList<String>();
73        for(Object obj:nomes){
74            items.add(obj.toString());
75        }
76        ArrayAdapter<String> fileList = new ArrayAdapter<String>(this,R.layout.activity_listar_arquivos, items);
77        setListAdapter(fileList);
78    }
79       @Override
80        protected void onListItemClick(ListView l, View v, int position, long id){
81
82         int selectedRow = (int)id;
83            if(selectedRow == 0){
84                }else{
85                    Intent it = new Intent();
86                    String retorno = items.get(selectedRow);
87                    it.putExtra("idArquivo", retorno);
88                    setResult(1, it);
89                    finish();
```

Figura 29 - Código da Classe RetornarArquivo

O retorno dos nomes exatos dos arquivos, após a escolha e retorno para o dispositivo, é feito por meio do seguinte código das linhas 60 à 62, da classe PegarArquivoBD, conforme é mostrado na figura 30.

```
60              byte[] byt = (byte[])envelope.getResponse();
61              String fileName = palavras[1];
62              File f = new File("mnt/sdcard/" +fileName);
```

Figura 30 - Pegar Arquivo do Banco de Dados

5. CONCLUSÕES

Por meio deste projeto, conclui-se que com auxílio das ferramentas descritas e das pesquisas realizadas é possível integrar diversas tecnologias de sistemas de informação.

Foi possível observar que o desenvolvimento para dispositivos móveis está cada vez mais acessível e não houve dificuldades para a criação do projeto nesta área.

A utilização de *Web Service* como solução para disponibilizar serviços foi fundamental. Com esta tecnologia é possível a comunicação de uma aplicação com qualquer outro sistema que seja capaz de se conectar ao *Web Service*. No caso criado, foi possível encontrar diversas vantagens de sua utilização, como por exemplo, a lógica de sua aplicação que assemelha-se à qualquer programa comum; sua acessibilidade que pode ser de qualquer computador e principalmente por se basear em padrões abertos e ser independente de plataformas, podendo ser implementado em qualquer linguagem.

Desenvolver uma aplicação, de transferência de arquivos, para dispositivos móveis que se comunique com *Web Service* é uma nova alternativa para os usuários gravarem e recuperarem os seus arquivos de uma maneira mais simples e eficaz. Sendo assim, por meio desta computação distribuída e com o uso de duas tecnologias consideradas novas, Android e *Web Service*, os usuário que possuírem a aplicação em seu dispositivo poderão ter acesso a seus arquivos à partir de qualquer lugar devido à facilidade de centralização de dados que o sistema possui.

Com a análise da aplicação que foi desenvolvida, pode-se concluir que o envio e retorno de arquivos, de qualquer formato e tamanho, por meio de um *smartphone* para o *Web Service* é totalmente possível, e não há perca alguma de informação entre ambos.

Para um trabalho mais completo, tem-se as seguintes sugestões e modificações para implementar em trabalhos futuros:

a) implementar a interface visual do aplicativo no dispositivo móvel;
b) implementar as opções de cadastro de usuário com validação de login e senha no banco de dados;
c) implementar operações de renomear, excluir, bloquear ou desbloquear e alterar os arquivos que forem enviados e retornados;
d) implementar a segurança do sistema durante a comunicação com os *Web Service*, com uso de certificados digitais e;
e) implementar uma rotina para fazer o tratamento de envio de recebimento de arquivos grande.

REFERÊNCIAS BIBLIOGRÁFICAS:

_____. Serviços na Web (*Web Services*) Teses Abertas, PUC – Rio de Janeiro. Disponível em: <http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0220942_04_cap_03.pdf>. Acesso em: 13 de maio de 2012.

ABLESON, Frank. Introdução ao desenvolvimento Android. 2009. Disponível em: <http://www.ibm.com/developerworks/opensource/library/os-android-devel/index.html>. Acesso em: 20 de abril de 2012.

BARROS, Thiago. O que é Smartphone e para que serve. 2011. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2011/12/o-que-e-smartphone-e-para-que-serve.html>. Acesso em: 05 de maio de 2012.

BURÉGIO, Vanilson A. A. Desenvolvimento de aplicações para dispositivos móveis com .NET**.** 2003**.** 69 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife. Disponível em: <http://www.cin.ufpe.br/~tg/2003-1/vaab.PDF>. Acesso em: 20 de junho de 2012.

CAELUM. FJ-21: Java para Desenvolvimento Web. Apostilas gratuitas. Disponível em http://www.caelum.com.br/download/caelum-java-web-fj21.pdf. Acesso em: 20 de junho de 2012.

CORREA, Fábio Arreguy. Android x Java ME. Disponível em: <http://saloon.inf.ufrgs.br/twiki/viewfile/Disciplinas/INF01008/TF09FabioCamargoAdroidxJ2ME?rev=1;filename=AndroidXJ2ME.pdf>. Acesso em: 12 de maio de 2012

DEITEL, Paul. DEITEL, Harvey. Java, como programar, 8ª edição. São Paulo, SP: Pearson Prentice Hall, 2010. ISBN: 978-85-7605-563-1.

FURLAN, Paula. Mobilidade impulsiona consumo de smartphones e tablets. São Paulo, 2012. Disponível em: <http://consumidormoderno.uol.com.br/na-pele-do-consumidor/mobilidade-impulsiona-consumo-de-smartphones-e-tablets> Acesso em: 20 de abril de 2012.

GOOGLE. Ksoap2 Android. Disponível em: http://code.google.com/p/ksoap2-android/ . Acesso em: 20 de junho de 2012.

GUIMARÃES, Camila . As empresas detectam vantagens no celular que também é computador. São Paulo,2005. Disponível em: <http://www.netmarkt.com.br/noticia2004/2306.html> . Acesso em: 13 de abril de 2012

I-WEB. Web Services. São Paulo, 2003. Disponível em: <http://www.iweb.com.br/iweb/pdfs/20031008-webservices-01.pdf>. Acesso em: 12 de maio de 2012.

LECHETA, Ricardo R. Google Android, Aprenda a criar aplicações para dispositivos móveis com Android SDK, 2ª edição. São Paulo, SP: Novatec, 2010. ISBN: 978-85-7522-244-7.

MENDONÇA, Aderval. Mobilidade em Análise. Disponível em: <http://www.devmedia.com.br/mobilidade-em-analise/3309>. Acesso em: 20 de abril de 2012.

MIRANDA, Luiz Henrique. Introdução ao mundo móvel. Goiás, [2004]. Disponível em: <http://www.devgoiania.net/artigos.aspx>. Acesso em: 20 abril 2012.

MICHELS, Fernando. Sistema de Transferência de Arquivos para Dispositivos Móveis Baseados em Web Services. Universidade Regional de Blumenau, Centro de Ciências Exatas e Naturais – Ciência da Computação. Blumenau, 2010.

MORAES, Marcelo. A História do Surgimento da Linguagem Java. Disponível em: <http://www.marcelomoraes.com.br/conteudo/marcelo/Java/historia_Java.pdf>. Acesso em: 05 de maio de 2012

NETTO, Max Mossiman. Mobilidade e Dispositivos Móveis. Disponível em: <http://www.linhadecodigo.com.br/artigo/206/mobilidade-e-dispositivos-moveis.aspx> Acesso em: 20 de abril de 2012

OGLIARI, Ricardo. AndroixxJava ME, o que é possível com a nova plataforma. 2011. Disponível em: <http://itweb.com.br/blogs/Android-x-Java-me-o-que-e-possivel-com-a-nova-plataforma-fisl-12/>. Acesso em: 12 de maio de 2012.

ORACLE, About Java ME. Disponível em: <http://www.oracle.com/technetwork/Java/Javame/about-Java-me-395899.html>. Acesso em: 12 de maio de 2012

RASMUSSEM, Bruna. Pesquisa Mundial do IDG mostra crescimento rápido do uso de smartphones. 2011. Disponível em <http://www.tecmundo.com.br/tablet/12288-pesquisa-mundial-do-idg-mostra-crescimento-rapido-do-uso-de-smartphones.htm> Acesso em: 20 de abril de 2012

RECKZIEGEL, Mauricio. Entendendo os web services. São Paulo, 2006. Disponível em: <http://www.imasters.com.br/artigo/4245/webservices/entendendo_os_webservices>. Acesso em: 14 de maio de 2012.

SCRIMGER, Rob; LASSALLE, Paul; PAHIHAR, Mirdula. Tcp/ip - a Biblia, 2002.

SEELY, Scott. SOAP: cross platform Web Services development using XML. New Jersey: Prentice Hall, 2002.

SIQUEIRA, Ethevaldo. Política e Tecnologia no Mundo Digital, A revolução da Mobilidade. Disponível em: <http://blogs.estadao.com.br/ethevaldo-siqueira/2011/03/12/a-revolucao-da-mobilidade-3-0/>. Acesso em: 20 de abril de 2012

SNELL, James; TIDWELL, Doug; KULCHENKO, Pavel. Programing web services with SOAP. Sebastopol: O'Reilly & Associates, 2002.

SOUZA, Bruno; NARDON, Fabiane,; REHEM, Serge. A História da Tecnologia Java. Easey Java Magazine. Edição 1, ano 1. p. 17 – 28. Disponível em: <http://www.devmedia.com.br/a-historia-da-tecnologia-Java-easy-Java-magazine-1/18446>. Acesso em: 15 de maio de 2012.

TOMÉ, Marco Sandoval. Integração de Sistemas Desktop com Dispositivos Móveis Utilizando Web Services. Universidade Tecnológica Federal do Paraná, Tecnologia em Análise e Desenvolvimento de Sistemas. Medianeira, 2011.