



Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis - IMESA

CELSO AUGUSTO BIANCHINI

**DESENVOLVIMENTO DE APLICAÇÕES PARA TABLETS
UTILIZANDO TECNOLOGIA ANDROID COM INTEGRAÇÃO WEB
SERVICE E A PLATAFORMA .NET**

ASSIS
2012

CELSO AUGUSTO BIANCHINI

**DESENVOLVIMENTO DE APLICAÇÕES PARA TABLETS
UTILIZANDO TECNOLOGIA ANDROID COM INTEGRAÇÃO WEB
SERVICE E A PLATAFORMA .NET**

Trabalho de Conclusão de
Curso apresentado ao
Instituto Municipal de Ensino
Superior de Assis, como
requisito do Curso de
Graduação.

Orientador: Drº Almir Rogério Camolesi

Área de Concentração: Informática.

ASSIS
2012

FICHA CATALOGRÁFICA

BIANCHINI, Celso Augusto

Desenvolvimento de aplicações para *Tablets* utilizando tecnologia *Android* com integração Web Service e a Plataforma .Net / Celso Augusto Bianchini. Fundação Educacional do Município de Assis – FEMA – Assis, 2012.

60p.

Orientador: Drº Almir Rogério Camolesi

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA

1. Google *Android* 2. Web Services 3. MySQL.

CDD: 001.6
Biblioteca da FEMA

**DESENVOLVIMENTO DE APLICAÇÕES PARA TABLETS
UTILIZANDO TECNOLOGIA ANDROID COM INTEGRAÇÃO WEB
SERVICE E A PLATAFORMA .NET**

CELSO AUGUSTO BIANCHINI

Trabalho de Conclusão de Curso
apresentado ao Instituto Municipal
de Ensino Superior de Assis,
como requisito do Curso de
Graduação, analisado pela
seguinte comissão examinadora:

Orientador: Drº Almir Rogério Camolesi

Analisadora: Espª Diomara Martins Reigato Barros

Assis
2012

DEDICATÓRIA

Dedico este trabalho à minha família, minha esposa Rosangela, amigos e todas as pessoas que acreditaram em meus sonhos e anseios, apoiando-me com força necessária para que pudesse realizá-los.

AGRADECIMENTOS

Primeiramente a Deus, é a ele que dirijo minha maior gratidão, pois sem Ele, nada seria possível. Pois Ele é o motivo da minha existência e a minha fortaleza e meu consolador nas horas mais difíceis.

A minha esposa Rosangela pelo seu amor, compreensão, ajuda e companheirismo em todos os momentos, deixando os meus dias mais felizes e belos.

Aos meus familiares, Aurora Roque Bianchini e João Bianchini Filho a aos meus irmãos, por estarem sempre ao meu lado, apoiando-me e conduzindo-me durante minha caminhada.

Ao meu orientador Dr^o Almir Rogério Camolesi, pela orientação, não somente durante este trabalho, mas por toda a minha vida acadêmica.

E a todos que colaboraram direta ou indiretamente na execução deste trabalho.

RESUMO

O objetivo deste trabalho foi pesquisar e compreender os conceitos da arquitetura orientada a serviços e as ferramentas empregadas para o uso de Web Services na linguagem de programação .NET, assim como o desenvolvimento de um aplicativo baseado na plataforma Windows da Microsoft, com o uso da linguagem de programação C# e também foi desenvolvido outro aplicativo baseado na plataforma Google *Android*, capazes de gerar e consumir serviços disponíveis na Web.

Foi elaborado um estudo de caso sobre a análise de uma implementação de um sistema proposto no levantamento dos requisitos, ao qual este sistema é específico para gestão de custos para Área Agrícola. Foram selecionadas algumas tabelas para a elaboração de um protótipo para a realização do estudo das pesquisas apresentadas neste trabalho.

Com a finalidade de aplicar os conceitos adquiridos na fase de pesquisas teóricas deste trabalho, foi desenvolvido na linguagem de programação C#, um protótipo para desktop de algumas tabelas, para fazer a comunicação com uma aplicação que foi desenvolvida na plataforma *Android* para o dispositivo móvel *Tablet*, utilizando da tecnologia Web Service e a linguagem Java para fazer a integração da comunicação entre as duas tecnologias desenvolvidas.

Ao final da etapa de implementação, foi possível testar e validar os aplicativos desenvolvidos, integrando-os como parte deste trabalho acadêmico, com a finalidade de expor os resultados alcançados e contribuir com novas pesquisas relacionadas ao assunto.

Palavras-chave: Google *Android*, MySQL, .NET, JSON, *Tablet*, Web Service.

ABSTRACT

The objective of this study was to investigate and understand the concepts of service-oriented architecture and the tools employed for the use of Web Services in the programming language. NET, as well as the development of an application based on Microsoft's Windows platform, using the C # programming language and also other application was developed based on the Google *Android* platform, able to generate and consume Web services available.

A study case on the analysis of an implementation of a proposed system requirements gathering, which is specific to this system cost management for Agricultural Area. Some tables were selected for the development of a prototype for the study of the research presented in this paper.

In order to apply the concepts acquired in the phase of theoretical research of this paper, was developed in the programming language C #, a prototype desktop of some tables, to communicate with an application that was developed on the *Android* platform for mobile device *Tablet* using Web Service technology and the Java language to integrate communication between the two technologies developed.

At the end of the implementation stage, it was possible to test and validate the applications developed by integrating them as part of academic work, with the aim of exposing the achievements and contribute new research on the subject.

Keywords: Google *Android*, MySQL, .NET, JSON, *Tablet*, Web Service.

LISTA DE ILUSTRAÇÕES

Figura 01 - Dispositivos Móveis.....	22
Figura 02 – Imagem de um <i>Tablet</i> CSL Spice Mi700 DroidPad.....	24
Figura 03 - Componentes do Sistema Operacional <i>Android</i>	26
Figura 04 – <i>Android</i> SDK Manager.....	32
Figura 05 - Ilustra a simulação do <i>Tablet</i> no emulador do <i>Android</i> 3.0.....	34
Figura 06 - Estrutura do .NET Framework.....	35
Figura 07 - Código fonte em C# para compilação em linguagem de máquina....	37
Figura 08 - Cenário de Funcionamento de um Web Services.....	41
Figura 09 - Elementos de uma mensagem SOAP.....	42
Figura 10 - Funcionamento do Registro UDDI.....	45
Figura 11 - Pilhas de protocolos que compõem um Web Services.....	45
Figura 12 - Cenário detalhado de padrões Web Services.....	46
Figura 13 - Mostra a modelagem do funcionamento dos dois sistemas.....	53
Figura 14 - Mapa Mental demonstrando uma visão geral do sistema.....	54
Figura 15 - Visão Geral dos Casos de Usos do Sistema.....	56
Figura 16 - Visão Geral do Diagrama de Entidade Relacionamento (DER).....	57
Figura 17 - Diagrama de Entidade Relacionamento que será desenvolvido.....	58
Figura 18 - Casos de Usos utilizados no desenvolvimento dos protótipos.....	60
Figura 19 - UC1 – Diagrama de Use Case – Manter Usuários.....	61
Figura 20 - UC2 – Diagrama de Use Case – Manter Item.....	62
Figura 21 - UC3 – Diagrama de Use Case – Manter Equipamento.....	63
Figura 22 - UC4 – Diagrama de Use Case – Manter Localização Área.....	64
Figura 23 - UC5 – Diagrama de Use Case – Manter Movimentação de Tarefas	65
Figura 24 - UC6 – Diagrama de Use Case – Manter Movimentação de Ordem	66
de Serviço.....	
Figura 25 - UC7 – Diagrama de Use Case – Manter Movimentação de Itens	67
de OS.....	
Figura 26 - Diagrama de Classes do Pacote Modelo – VO (Value Objects).....	69
Figura 27 - Diagrama de Classes do Pacote DAL.....	70
Figura 28 - Diagrama de Classes do Pacote BLL.....	71
Figura 29 - Diagrama de Atividades de todo o processo dos protótipos.....	72

Figura 30 - Diagrama de seqüência – manter ordem de serviço.....	73
Figura 31 - Diagrama de seqüência – manter itens da ordem de serviço.....	74
Figura 32 - Diagrama de seqüência – manter tarefas realizadas.....	75
Figura 33 - Diagrama de casos de usos do módulo desktop.....	77
Figura 34 - Diagrama de casos de usos do módulo móbile (<i>Tablet</i>).....	78
Figura 35 - Processo de comunicação Web Service entre C# e <i>Android</i>	80
Figura 36 - Sincronização das tabelas do SQLite para o MySQL.....	81
Figura 37 - Sincronização das tabelas do MySQL para o SQLite.....	82
Figura 38 - Tela de autenticação do Login e Senha do Desktop.....	83
Figura 39 - Tela do menu de opções do módulo desktop.....	84
Figura 40 - Tela de cadastros de opções do módulo desktop.....	84
Figura 41 - Interface da tela de cadastros dos usuários do sistema.....	85
Figura 42 - Organização em pacotes.....	86
Figura 43 - Tela de autenticação do Login e Senha do <i>Tablet</i>	87
Figura 44 - Tela do menu do aplicativo móvel (<i>Tablet</i>).....	88
Figura 45 - Tela das opções do menu do <i>Tablet</i>	88
Figura 46 - Tela de cadastro das ordens de serviços no <i>Tablet</i>	89
Figura 47 - Tela de lançamento dos itens da Ordem de Serviço (<i>Tablet</i>).....	89
Figura 48 - Tela de controle de tarefas dos equipamentos agrícolas (<i>Tablet</i>)....	90
Figura 49 - Tela de sincronização do <i>Tablet</i>	90
Figura 50 - Mostra a organização em pacotes do <i>Tablet</i>	91
Figura 51 - Organização de pacotes – imagens e telas do <i>Tablet</i>	92
Figura 52 - Organização de pacotes – Web Service em Java.....	93
Figura 53 – Pacote para consumir Web Service em Java.....	100
Figura 54 - Mapa Mental demonstrando uma visão geral do sistema.....	112
Figura 55 - Visão Geral dos Casos de Usos do Sistema.....	113
Figura 56 - Visão Geral do Diagrama de Entidade Relacionamento (DER).....	114
Figura 57 - Diagrama de Classes do Pacote DAL.....	115
Figura 58 - Diagrama de Classes do Pacote BLL.....	116
Figura 59 - Diagrama de Atividades de todo o processo dos protótipos.....	117

LISTA DE TABELAS

Tabela 1 - Principais APIs do <i>Android</i>	26
Tabela 2 - Instalando o Android Development Tools no Eclipse.....	33
Tabela 3 - Manter Usuário.....	61
Tabela 4 - Manter Item.....	62
Tabela 5 - Manter Equipamento.....	63
Tabela 6 - Manter Localização Área.....	64
Tabela 7 - Manter Movimentação de Tarefas.....	65
Tabela 8 - Manter Movimentação de Ordem de Serviço.....	66
Tabela 9 - Manter Movimentação de Itens da Ordem de Serviço.....	67

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming

HTTP - HyperText Transport Protocol

SDK - Software Development Kit

SOA - Arquitetura Orientada a Serviços

SOAP – Simple Object Access Protocol

SQL - Structure Query Language

UC - Use Case

UDDI – Universal Description Discovery and Integration.

UML - Unified Modeling Language

XML - Extensible Markup Language (Linguagem de Marcação Extensível)

WSDL - Web Services Description Language.

OS – Ordem de Serviço.

SUMÁRIO

1 – INTRODUÇÃO.....	16
1.1 OBJETIVOS.....	18
1.2 JUSTIFICATIVA.....	18
1.3 PUBLICO ALVO.....	18
1.4 ESTRUTURA DO TRABALHO.....	19
 2 – FUNDAMENTAÇÃO TEÓRICA.....	20
2.1 TIPOS DE DISPOSITIVOS MÓVEIS.....	21
2.1.1 Vantagens dos Dispositivos Móveis.....	22
2.1.2 <i>Tablets</i> um Dispositivo Móvel em Crescimento.....	23
2.2 <i>ANDROID</i>	24
2.2.1 Arquitetura do <i>Android</i>	25
2.2.2 Aplicativos.....	27
2.2.3 Framework dos Aplicativos.....	27
2.2.4 Bibliotecas.....	27
2.2.5 <i>Android</i> Runtime.....	28
2.2.6 Kernel do Linux.....	29
2.2.7 SQLite.....	29
2.2.8 Ambiente de Desenvolvimento do <i>Android</i>	30
2.2.8.1 Requisitos Mínimos para a Utilização do <i>Android</i>	30
2.2.8.2 Instalação do <i>Android</i>	31
2.3 PLATAFORMA .NET.....	34
2.3.1 A Linguagem C#.....	36
2.3.2 Vantagens da Utilização do .NET FRAMEWORK.....	37
2.4 O BANCO DE DADOS MYSQL.....	38
2.5 WEB SERVICES.....	39
2.5.1 Esquema de funcionamento de um Web Service.....	39
2.5.2 SOAP (Simple Object Access Protocol).....	41
2.5.3 WSDL - Web Services Description Language.....	43
2.5.4 UDDI - Universal Description, Discovery and Integration.....	43
2.5.5 Modelo de dados JSON.....	46

2.5.6 Comparação entre JSON e XML.....	47
2.5.7 Utilização da Arquitetura REST no desenvolvimento de aplicações Web.....	49
2.5.7.1 Definição do Rest.....	50
3 - ANÁLISE E ESPECIFICAÇÃO DO PROTÓTIPO.....	52
3.1 DESCRIÇÃO DO PROBLEMA.....	52
3.2 MAPA MENTAL DO SISTEMA.....	53
3.3 LISTA DE EVENTOS DO SISTEMA.....	54
3.4 CASOS DE USOS DO SISTEMA.....	55
3.5 DIAGRAMA DE ENTIDADE E RELACIONAMENTO (DER).....	56
3.6 LISTA DE EVENTOS DO DESENVOLVIMENTO DOS PROTÓTIPOS.....	58
3.7 CASOS DE USOS.....	59
3.8 ESPECIFICAÇÃO DOS CASOS DE USOS.....	60
3.9 DIAGRAMA DE CLASSES.....	68
3.10 DIAGRAMA DE ATIVIDADES.....	72
3.11 DIAGRAMA DE SEQUÊNCIA.....	73
3.11.1 Manter Ordem de Serviço.	73
3.11.2 Manter Itens da Ordem de Serviço.....	74
3.11.3 Manter Tarefas Realizadas.....	75
3.12 MÓDULO DESKTOP – (BACK-END).....	76
3.12.1 Casos de Usos do Módulo Desktop.....	76
3.13 MÓDULO MÓBILE - (FRONT-END).....	77
3.13.1 Casos de Usos do Módulo MóBILE.....	78
3.14 MÓDULO DE INTEGRAÇÃO (WEB SERVICE).....	79
3.14.1 Processo de mapeamento das tabelas dos bancos de dados... ..	80
4 - IMPLEMENTAÇÃO DO SISTEMA.....	83
4.1 APLICATIVO DESKTOP (C#).....	83
4.1.1 Organização dos pacotes e classes da aplicação Desktop.....	85
4.2 APLICATIVO MÓBILE (TABLET).....	87
4.2.1 Organização dos pacotes e classes da aplicação Tablet.....	91

4.3 DESENVOLVIMENTO DO WEB SERVICE.....	93
4.3.1 Organização dos pacotes do Web Service em Java.....	93
4.3.2 Organização dos pacotes no <i>Android</i> para consumir Web Service em Java.....	100
5 - CONCLUSÃO.....	108
6 REFERENCIAS.....	109
7 ANEXOS.....	112

1 - INTRODUÇÃO

A utilização de dispositivos móveis vem se tornando, cada vez mais presente em nosso cotidiano e nos ambientes empresariais. E com a disponibilidade de recursos oferecidos, por inúmeras oportunidades de exploração de Software, voltada ao ambiente empresarial, estão em constante crescimento. Pela característica de permitir a mobilidade na comunicação, as redes sem fio constituem a melhor alternativa de conexão neste setor (LAGO; SILVA, 2006).

A computação utilizada para dispositivos móveis é caracterizada pela utilização de computadores portáteis com capacidade de comunicação, fortemente associada com a mobilidade de hardware, software e comunicação (ITO, 2008).

Segundo Ito (2008), as tecnologias de computação móvel abrangem computadores móveis comumente designados por *PDA's*, *Handhelds*, *Smartphone*, *Tablets* e outros dispositivos que refletem a comunicação entre os computadores móveis e celulares. Cada tipo de dispositivo móvel possui um sistema operacional integrado que realiza todo o seu funcionamento e processo de comunicação. Por exemplo: os *PocketPCs*¹ utilizam o sistema operacional Windows Móvel e seus aplicativos para a plataforma Windows.

Existem vários sistemas operacionais disponíveis para os dispositivos móveis, por exemplo: (Nokia – *Symbian*²; Google – *Android*³; Apple – *iOS*⁴; Samsung – *Bada OS*⁵, Microsoft - *Windows Phone*⁶),

Neste contexto, surgiram os computadores de mão mais conhecidos como *Tablets*. O *Tablet*⁷ é um produto inovador que chegou ao mercado recentemente para inovar o perfil da informática. O aparelho possui uma estrutura fina, compacta e pode ser levado para qualquer lugar de forma prática e viável, um exemplo de sua portabilidade. Pelo fato do *Tablet* ser uma novidade no mercado tecnológico, há ainda muitas pessoas que desconhecem as funções desse equipamento.

Dado o fator mobilidade e conectividade oferecida por um *Tablet*, podemos utilizar ferramentas de desenvolvimento, criadas para suportar e facilitar o uso dos

¹ <http://www.palmbrasil.com.br/windows-mobile>

² <http://www.symbian.com>

³ <http://www.android.com/>

⁴ <http://www.apple.com/br/>

⁵ <http://www.samsung.com/br/>

⁶ <http://www.microsoft.com/mobile>

⁷ <http://www.tablet.com.br/o-que-e-um-tablet-saiba-em-detalhes/>

recursos de hardware. Uma das ferramentas utilizadas no desenvolvimento de aplicativos é o *Android*, baseado no sistema operacional Linux e com um ambiente de desenvolvimento flexível e poderoso. O *Android* é uma plataforma que oferece as ferramentas necessárias para o desenvolvimento de aplicativos para os dispositivos móveis, utiliza a linguagem Java⁸, com suporte para diversos serviços e hardware, além de possuir um sistema de código aberto e livre, sendo que as ferramentas de desenvolvimento do aplicativo são gratuitas e disponibilizadas sem nenhum custo. (SILVA, 2009).

As aplicações móveis geralmente devem ser integradas com outras tecnologias e ser capaz de absorver técnicas de comunicação com outros dispositivos, enfim, é um instrumento poderoso para desenvolvedores interessados em integração e praticidade (ITO, 2008).

Temos várias ferramentas que fazem o serviço de sincronização das informações de diferentes sistemas, ou seja, temos a opção de usar as seguintes tecnologias (Java Business Integration⁹ (JBI), Enterprise Service Bus¹⁰ (ESB), Web Service¹¹, etc.).

Para o desenvolvimento de aplicativos móveis que possam acessar, armazenar, processar informações de qualquer lugar, com a facilidade de sincronização das informações, pode-se utilizar a tecnologia Web Service como um intermediador de comunicação entre diversos tipos de sistemas (HANSEN; PINTO, 2003).

Sendo assim, a Plataforma .NET apresenta funcionalidades capazes de preencher as novas necessidades de desenvolvimento. É uma plataforma capaz de integrar serviços, garantir portabilidade à solução, tem suporte a XML (Extensible Markup Language) que facilita a comunicação com a web, com uma capacidade de atender as tecnologias de comunicação com os dispositivos móveis, ou seja, é uma ferramenta poderosa para desenvolvedores interessados em integração e praticidade (GONÇALVES, 2007).

⁸ www.oracle.com.br

⁹ <http://www.javabeat.net/articles/60-service-oriented-java-business-integration-1.html>

¹⁰ <http://www.javabeat.net/articles/60-service-oriented-java-business-integration-1.html>

¹¹ <http://www.webservices.org/>

1.1 - OBJETIVOS

O objetivo do presente trabalho é apresentar os conceitos de desenvolvimento de aplicações, utilizando as novas tecnologias de programação voltada para aplicativos móveis e a sua comunicação com outras tecnologias, a fim de auxiliar novas pesquisas e incentivar a utilização destes recursos nos ambientes empresariais.

O trabalho também deve mostrar como funciona a integração e a comunicação de um sistema desenvolvido na plataforma *Android*, que executa em um *Tablet* e um sistema desenvolvido para Desktop na Plataforma .NET (C Sharp). A tecnologia utilizada para fazer o papel do intermediador entre a comunicação dos dois sistemas será o Web Service.

Portanto, esse trabalho tem como objetivo mostrar a utilização e a integração destas tecnologias e abrir novas possibilidades para futuras pesquisas de sistemas integrados.

1.2 - JUSTIFICATIVA

A valorização dos *Tablets* como um produto de alta tecnologia, tem despertado um grande interesse nos ambientes empresariais. À falta de conhecimento aprofundado na utilização de tecnologias, que vem sendo utilizadas no desenvolvimento de aplicativos comerciais específicos para os *Tablets* e a comunicação com outros aplicativos.

Sendo assim, este trabalho vem demonstrar a integração da tecnologia *Android* com a Plataforma .NET usando a linguagem (C Sharp) e o Web Service como intermediador na comunicação dos dois sistemas.

1.3 - PÚBLICO ALVO

Conforme pesquisas realizadas, pode-se observar que a tecnologia *Android*¹² está em constante crescimento no mercado global, desta forma os consumidores estão optando para a compra de *tablets* com o sistema operacional

¹² <http://www.opovo.com.br/app/opovo/tendencias/2012/01/27/noticiasjornaltendencias.2774148/android-ganha-mais-espaco-no-mercado-global.shtml>

Android ao invés de notebooks e computadores de mesa. Desta forma, observa-se o interesse comercial das empresas que estão a cada dia modernizando os seus processos, sempre buscando inovações em novas tecnologias, com objetivo de estarem cada vez mais competitivas no mercado.

1.4 - ESTRUTURA DO TRABALHO

Este trabalho foi dividido em capítulos que serão explicados a seguir:

No primeiro capítulo foi apresentado à contextualização e a justificativa para o desenvolvimento da proposta do trabalho.

A seguir, no segundo capítulo serão abordados os conceitos de fundamentação teórica das tecnologias utilizadas para o estudo de um protótipo de um sistema.

No terceiro capítulo apresenta as etapas de análise e especificações do protótipo de uma parte de um sistema para controle de frotas agrícolas contemplando o levantamento de requisitos, lista de eventos, os casos de usos e suas especificações e os principais diagramas da UML (classes, seqüências e atividade).

A implementação de um protótipo do sistema, exibindo um detalhamento sobre a aplicação desenvolvida assim como a distribuição das camadas, organização de pacotes e a interface criada para interagir com o usuário final será apresentado no Capítulo 4.

No final, o quinto capítulo será apresentado à conclusão e trabalhos futuros.

2 – FUNDAÇÃO TEÓRICA

Neste capítulo vamos abordar as tecnologias e ferramentas utilizadas no desenvolvimento de um protótipo de um sistema para *Tablet* e C#.

Atualmente, o alvoroço do mercado está voltado para os dispositivos móveis, que vem se destacando no mundo da Informática. Dentre os vários tipos existentes no mercado, podemos citar os *Tablets*, uma espécie de prancheta digital que permite rodar um sistema operacional que possui diversas funções de um computador pessoal e que, principalmente, permite a leitura de livros, revistas e jornais, um dos aspectos mais explorados pelos novos modelos. Além dos seus recursos, como acessar a internet, rodar aplicativos, ouvir música, assistir vídeos e jogar games, porém de uma forma mais intuitiva, por meio da tecnologia Touchscreen, onde o usuário pode manipular o conteúdo com o toque dos próprios dedos diretamente na tela (Natansohn, Cunha 2010).

O *Android* é uma plataforma utilizada em alguns dos modelos de *Tablets*, é baseado no Sistema Operacional (*Android* SO) Linux, possui diversos componentes, com várias disponibilidades de bibliotecas e interface gráfica, que também disponibiliza ferramentas para a criação de aplicativos (Lecheta, 2009).

Muitas vezes é necessário, que haja a integração de soluções, para garantir que o produto final atenda os preceitos definidos na fase de Concepção do projeto, e ainda, que seja possível poupar os investimentos realizados previamente em sistemas legados utilizados até então, buscando um novo modelo de desenvolvimento, desta forma utilizamos a Plataforma .NET para alcançarmos o objetivo (Lima, Edwin 2002).

De acordo com Lima, Edwin (2002), a plataforma .NET é composta por um Framework, que possui Bibliotecas Unificadas e fornece um ambiente de programação orientado a objetos consistente em um ambiente de execução segura do código, além de um ambiente de execução que elimina os problemas de desempenho dos ambientes interpretados ou com scripts.

O .NET é uma plataforma de desenvolvimento avançada produzida pela Microsoft, que apresenta como vantagem ser multi-plataforma e multi-linguagem, oferecendo suporte a uma variedade de linguagens de programação, tanto as disponibilizadas pela própria Microsoft, como C#, VB.NET, C++.NET, J#, bem como

outras linguagens oferecidas por outras empresas como COBOL, Pascal, Eiffel, Fortran, Perl e Python oferecidas pela Fujitsu (Lima, Edwin 2002).

O MySQL é um servidor de banco de dados SQL multiusuário multithreaded. A SQL é a linguagem de banco de dados mais popular no mundo, é uma linguagem padronizada que torna fácil o armazenamento e acesso de informações.¹³

Os Web Services apresentam uma estrutura arquitetural que permite a comunicação entre vários tipos de aplicações. Um serviço pode ser invocado remotamente ou ser utilizado para compor um novo serviço juntamente com outros serviços. A tecnologia de Web Services está baseada em linguagem (XML), onde permite invocar ou reutilizar um serviço sem a necessidade de conhecer a plataforma ou linguagem de programação usada na sua construção (Bray, 2000).

O JSON (JavaScript Object Notation) é uma estrutura de dados em JavaScript. Foi criada a partir do padrão ECMA-262 que por sua vez possui convenções muito semelhante às linguagens como C, C#, C++, Java, JavaScript, Perl, Python e várias outras. Desta forma, as características do JSON se torna o formato ideal para troca de informações independente da linguagem, principalmente para tráfego de informações em ambientes heterogêneos via HTTP (WEBBER; PARASTATIDIS, 2010).

Para realizar a troca de informações entre o sistema do *Tablet* e o banco de dados no servidor será escolhida a tecnologia JSON para troca de mensagens.

2.1 - TIPOS DE DISPOSITIVOS MÓVEIS.

Os dispositivos móveis estão cada vez mais populares e agregando funções que vão muito além de uma simples agenda eletrônica ou de um meio de comunicação, como os celulares. Esses dispositivos estão se tornando pequenos computadores que realizam tarefas complexas com um poder de processamento que está em constante desenvolvimento (Filgueiras, 2009).

Com o surgimento dos dispositivos móveis, eles passaram a ser computadores que podem ser facilmente levados a qualquer lugar, criados para atender profissionais e pessoas em movimento que necessitam de rapidez,

¹³ AB, MySQL. Guia Completo do MySQL. Disponível em: <http://www.apostilando.com/sessao.php?cod=26>
Acesso em: 22 de Outubro de 2012.

facilidade e segurança no acesso a informações organizacionais e pessoais. Além disso, as grandes inovações trazidas pela tecnologia wireless fizeram com que a indústria deste setor tenha um crescimento explosivo nos últimos anos, tornando-se uma das mais eficientes e rápidas áreas tecnológicas do mundo, permitindo que as pessoas comuniquem-se de forma rápida e fácil sem ficarem presas aos seus telefones ou computadores de mesa (Laudon, 1999).

Podemos encontrar dezenas de modelos diferentes no mercado e cada um deles pode ter um destes sistemas operacionais: Windows Mobile, iPhone OS, Blackberry, Symbian, Palm OS, alguma distribuição Linux ou o *Android* (Laudon, 1999). Na figura 1, apresenta uma variedade de dispositivos móveis.



Figura 1 – Dispositivos Móveis.

Fonte: <http://computeiros-agpt.blogspot.com.br/>

2.1.1 - Vantagens dos Dispositivos Móveis.

De acordo com Schaefer (2004), os dispositivos móveis, do ponto de vista empresarial, são ótimos geradores de informação, podendo ser utilizado na automatização do processo, até nas coletas de informações estratégicas, pois com suas reduzidas dimensões podem ser transportados e estar presentes em todas as situações em que um profissional pode atuar.

Vamos destacar algumas vantagens dos dispositivos móveis em relação aos computadores (Schaefer, 2004).

- Tamanho: bastante reduzido e muito mais leve do que os PCs, podendo ser transportados de forma muito mais prática;
- Fácil manuseio: os dispositivos móveis possuem uma interface gráfica simples de manusear se comparado aos computadores;
- Consumo de energia: por serem menores e mais econômicos gastam menos energia que os computadores visto que o tempo de recarga é menor;
- Custos operacionais: como os dispositivos móveis são mais compactos e possuem atividades específicas, estes aparelhos não possuem alguns periféricos internos, como discos rígidos e discos flexíveis, diminuindo consideravelmente os custos com a manutenção;

Na atualidade que vivemos hoje, observamos que as pessoas estão cada vez mais dependentes das informações disponíveis na Internet, o que antes poderia ser feito apenas via terminal remoto, agora pode ser acessado via dispositivo móvel. Com o surgimento da tecnologia sem fio disponibiliza ao usuário, facilita a possibilidade de obter informações que lhes sejam úteis, a qualquer momento ou qualquer lugar (Pelissoli, 2004).

A mobilidade é outra característica que deve ser levada em consideração. A capacidade de poder continuar uma comunicação e manter o envio de dados constante mesmo quando em movimento pode ser considerada uma das melhores vantagens de um dispositivo móvel (Pelissoli, 2004).

2.1.2 – *Tablets* um Dispositivo Móvel em Crescimento.

Os *Tablets*, assim como os *Smartphones*, evoluíram muito com o passar do tempo, resultado de muita pesquisa e tecnologia aplicada aos aparelhos. A dificuldade encontrada antes na aceitação do consumidor foi por causa dos preços elevados e das limitações técnicas. Depois do lançamento do iPhone 3G e, posteriormente, do iPad, que até o final do ano de seu lançamento chegou a dominar 84% do mercado, que os aparelhos passaram a despertar o interesse do consumidor (WEBDIG, 2011).

Atualmente os *Tablets* juntamente com os *Smartphones* são considerados um mercado em crescimento no setor de tecnologia. Grandes empresas estão investindo no segmento de dispositivos móveis, assim como a Intel, maior fabricante de processadores do mundo, que “anunciou que está desenvolvendo uma arquitetura de processadores competitiva para *Tablets*” (G1, 2011).

A figura 2 abaixo mostra a foto de um *Tablet* CSL Spice Mi700 DroidPad com *Android*.



Figura 2 – Imagem de um *Tablet* CSL Spice Mi700 DroidPad.

Fonte: <http://targethd.net/2010/10/01/tablet-csl-spice-mi700-droidpad-chega-com-android-froyo-e-uma-aprencia-muito-conhecida/>

2.2 – ANDROID.

O *Android* é gratuito, de código aberto e baseado em um Kernel Linux. Juntamente com o *Android*, é disponibilizado um SDK para a construção de aplicativos em Java e um plugin para o Eclipse IDE. Java é uma linguagem de programação sólida e bem aceita no mercado e o Eclipse é a IDE mais utilizada no mundo para o desenvolvimento nessa linguagem (Filgueiras, 2009).

Ele possui um middleware (programa de computador que faz a mediação entre outros softwares) e algumas aplicações básicas que garantem o seu funcionamento.

Conforme Filgueiras (2009), segue abaixo os recursos oferecidos pelo *Android*:

- Framework de aplicação que possibilita a reutilização e substituição dos componentes.
- Máquina virtual otimizada para dispositivos móveis.
- Navegador integrado baseado no WebKit, cujo código fonte é aberto.
- Biblioteca gráfica 2D. Gráficos 3D baseados na especificação OpenGL ES 1.0 (com aceleração de hardware opcional).
- SQLite para armazenamento estruturado de dados.
- Suporte para os formatos de áudio, vídeo e imagem mais utilizados (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- Telefonia GSM (dependente do hardware).
- Bluetooth, EDGE, 3G e WiFi (dependente do hardware).
- Câmera, GPS, bússola e acelerômetro (dependente do hardware)
- Ambiente de desenvolvimento rico incluindo um emulador de dispositivo, ferramentas para debug, memória e desempenho e um plugin para o Eclipse IDE.

2.2.1 - Arquitetura do *Android*.

O Sistema Operacional *Android* possui alguns componentes principais conforme mostra na figura 3.

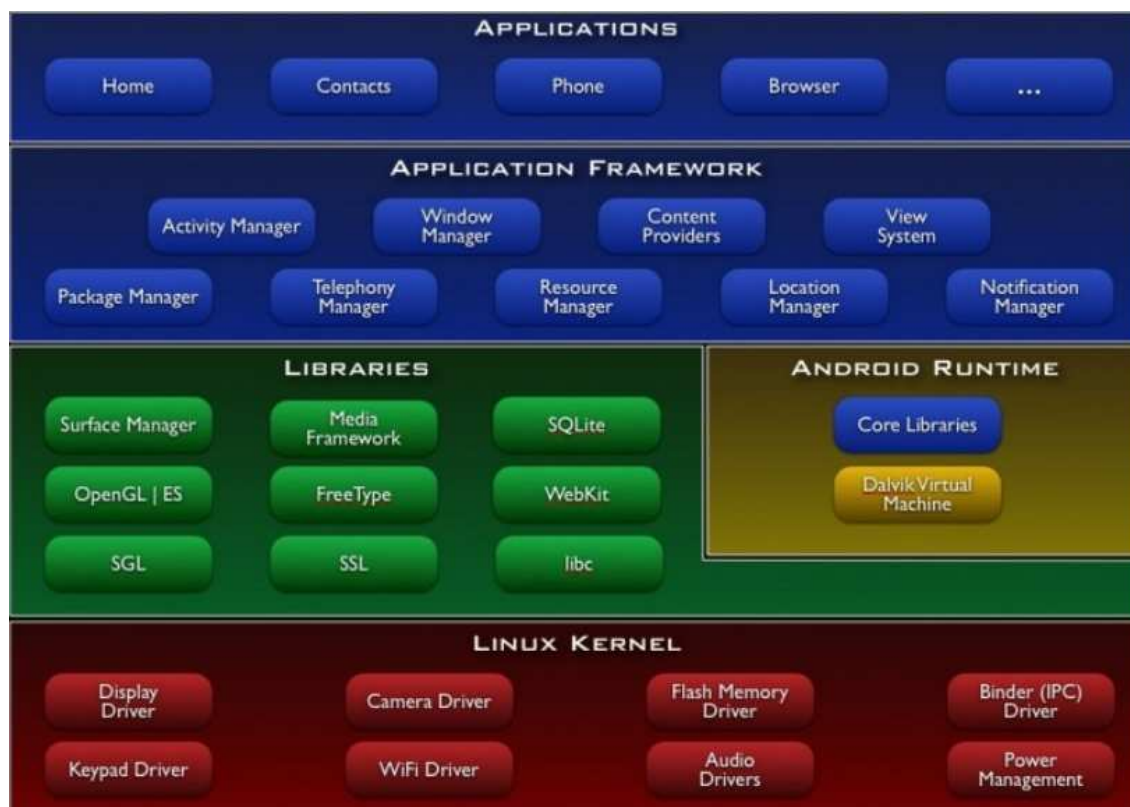


Figura 3. Componentes do Sistema Operacional *Android*.

Pacote	Descrição
android.util	Contém várias classes utilitárias (classes de containers, utilitários XML)
android.os	Contém serviços referentes ao sistema operacional, passagem de parâmetros e comunicação entre processos.
android.graphics	Pacote principal dos recursos gráficos.
android.text android.text.method android.text.style android.text.util	Suporte para um conjunto de ferramentas de processamento de texto, suporte ao formato de texto rico (RTF), métodos de entradas, etc.
android.database	Contém APIs para comunicação com o banco de dados SQLite
android.content	APIs de acesso a dados no dispositivo, como as aplicações instaladas e seus recursos.
android.view	O pacote principal que contém os principais componentes de interface gráfica.
android.widget	Contém widgets prontos (botões, listas, gerenciadores de layout, etc) para serem utilizados nas aplicações
android.app	APIs de alto-nível referentes ao modelo da aplicação. É implementada por meio de Activities (Atividades)
android.provider	Contém várias APIs para padrões de provedores de conteúdos (content providers)
android.telephony	APIs para interagir com funcionalidades de telefonia
android.webkit	Inclui várias APIs para conteúdos de context web, bem como um navegador embutido que pode ser utilizado por qualquer aplicação.

Tabela 1. Principais APIs do *Android*

2.2.2 – Aplicativos.

O *Android* já possui um conjunto de aplicativos, incluindo: cliente de email, programa para envio de SMS, calendário, mapas, navegador, agenda de contatos entre outros. Todos foram feitos utilizando a linguagem de programação Java (Tanenbaum, 2003).

2.2.3 – Framework dos Aplicativos.

Os desenvolvedores têm acesso às mesmas API's utilizadas pelas aplicações citadas acima. A arquitetura foi projetada para simplificar a reutilização de componentes, permitindo até a substituição de um aplicativo pelo usuário (Tanenbaum, 2003).

Conforme Tanenbaum (2009) existem conjuntos de serviços e sistemas que são utilizados ao escrever uma aplicação para o *Android*. Alguns desses componentes incluem:

- ✓ Um rico e extenso conjunto de *Views* que auxiliam a criação de uma aplicação possuindo listas, tabelas, caixas de texto, botões e até navegadores.
- ✓ *Content Providers* que permitem o compartilhamento de dados entre as aplicações.
- ✓ *Resource Manager* para acessar recursos como figuras e outros arquivos.
- ✓ *Notification Manager* que permite a exibição de alertas personalizados na barra de status.
- ✓ *Activity Manager* que gerencia o ciclo de vida das aplicações e permite a navegação entre elas.

2.2.4 – Bibliotecas.

Durante o desenvolvimento de programas, existe rotinas básicas que muitas vezes são comuns em muitos softwares, como abrir arquivos, realizar cálculos aritméticos, que não são o foco principal do desenvolvedor (Filgueiras, 2009).

Para se evitar a reescrita dos códigos de funções e rotinas que se repetem ou possuem altas complexidades, existem as bibliotecas. As bibliotecas possuem

códigos e dados que auxiliam na execução de serviços e permite a separação de partes do programa, ou seja, a divisão em módulos.

Segundo Filgueiras (2009), o *Android* possui um conjunto de bibliotecas, disponíveis para a criação de seus aplicativos:

- ✚ Biblioteca C do sistema: derivada da biblioteca padrão do sistema operacional BSD modificada para dispositivos baseados em Linux.
- ✚ Bibliotecas Multimídia: baseada na OpenCORE da PacketVideo. Suportam execução e gravação dos formatos de áudio e vídeo mais populares, assim como arquivos de imagem estática. Por exemplo: MPEG4, H.264, MP3, AAC, AMR, JPG e PNG.
- ✚ Surface Manager: Gráficos 2D e 3D a partir de várias aplicações.
- ✚ LibWebCore: um engine moderno para navegadores que gerencia tanto o navegador principal do *Android* quanto um navegador dentro de uma aplicação.
- ✚ SGL: Engine para gráficos.
- ✚ Bibliotecas 3D: uma implementação baseada nas APIs do OpenGL ES 1.0. Elas usam tanto a aceleração de hardware quando disponível, quanto um renderizador 3D otimizado por software.
- ✚ FreeType: renderização de bitmaps e fontes.
- ✚ SQLite: engine para bancos de dados relacionais. Disponível para todas as aplicações.

2.2.5 - *Android* Runtime.

No *Android*, toda aplicação é executada em um processo próprio. Isso é possível, pois cada aplicativo roda em uma instância própria da máquina virtual Dalvik. O Dalvik foi escrito de forma que várias instâncias da máquina virtual seja executada em um mesmo dispositivo de maneira eficiente (Tanenbaum, 2003).

A máquina virtual Dalvik executa arquivos do formato Dalvik Executable (.dex) que nada mais é do que classes Java compiladas e otimizadas para um consumo baixo de memória. Além disso, ela invoca o Kernel do Linux para as funcionalidades como *threading* e gerenciamento de memória de baixo nível (Filgueiras, 2009).

2.2.6 – Kernel do Linux.

Os dispositivos móveis, assim como os computadores e outros dispositivos eletrônicos, possuem arquiteturas distintas entre si. Para que um sistema possa operar com diferentes tipos de hardware, é necessário a utilização dos sistemas operacionais, assim como os PC's e os Notebook's (Tanenbaum, 2003).

O sistema operacional deve ser capaz de gerenciar o processador, memória e outros dispositivos de entrada e saída, além de fornecer, suporte aos programas de usuários, uma facilidade de interação com o hardware (Filgueiras, 2009).

Para a utilização dos dispositivos moveis, é necessário o desenvolvimento de softwares que não seja apenas para um único modelo de celular, mas para um determinado sistema operacional em especifico, ficando este responsável por gerenciar as particularidades de cada dispositivo (Tanenbaum, 2003).

O *Android* utiliza o Kernel do Linux 2.6, que é responsável pelos serviços de segurança, gerenciamento de memória, processos, rede e drivers, este último componente é muito importante, pois garante que o desenvolvedor não precisará se preocupar em como acessar ou gerenciar dispositivos específicos, produzindo assim uma abstração entre o hardware e o software (Filgueiras, 2009).

2.2.7 – SQLite.

O SQLite é uma biblioteca que implementa mecanismo de manipulação de banco de dados SQL. O código para o SQLite é de domínio público e, portanto, é livre para ser utilizada para qualquer fim, comercial ou privado. SQLite é atualmente encontrado em mais aplicações do que podemos contar, incluindo vários projetos de alto perfil¹⁴.

Funciona como um mecanismo de banco de dados SQL embutido. Ao contrário da maioria das demais bases de dados SQL, não tem um servidor separado, lê e escreve diretamente para arquivos em disco ordinário. Fornece um completo banco de dados SQL com várias tabelas, índices, triggers e views, tudo isso contido em um único arquivo em disco (Tanenbaum, 2003).

¹⁴ <http://www.sqlite.org/index.html>

O uso do SQLite¹⁵ é recomendado onde a simplicidade da administração, implementação e manutenção, são mais importantes que incontáveis recursos que o SGBDs mais voltados para aplicações complexas que o implementam.

A base do código é feita por uma equipe internacional de desenvolvedores que trabalham em tempo integral para o projeto. Os desenvolvedores continuam a expandir as capacidades do SQLite para melhorar o seu desempenho e confiabilidade (Hipp, 2008).

Por se tratar de um sistema leve e que exige muito pouco em relação a espaço em disco e poder de processamento, o SQLite é muito indicado para sistemas embarcados em dispositivos móveis (Tanenbaum, 2003).

2.2.8 - Ambiente de Desenvolvimento do *Android*.

Na seção abaixo, descreve como baixar e instalar o SDK do *Android*, o *plugin* ADT e configurar o ambiente de desenvolvimento tornando possível a criação de aplicações na plataforma *Android* (Filgueiras, 2009).

2.2.8.1 - Requisitos Mínimos para a Utilização do *Android*.

Segue abaixo uma lista completa do que pode ser utilizado para começar a construir aplicativos para o *Android* (Filgueiras, 2009).

Os seguintes sistemas operacionais são oficialmente suportados:

- ✓ Windows XP ou Vista
- ✓ Mac OS X 10.4.8 ou superior (x86)
- ✓ Linux (Ubuntu Dapper Drake).

O ambiente de desenvolvimento Eclipse é fortemente recomendado por possuir componentes que se integram ao *Android* e facilitam o desenvolvimento. O Eclipse deve atender alguns requisitos para que essa integração seja possível. Sendo assim, outros ambientes podem ser utilizados para o desenvolvimento (Filgueiras, 2009).

Abaixo, consta uma lista dos requisitos de cada um dos possíveis ambientes:

¹⁵ <http://www.sqlite.org/>

❖ Eclipse IDE

- ⇒ Eclipse 3.3 (Europa) ou 3.4 (Ganymede)
 - ✚ Plugin Eclipse JDT (incluso na grande maioria dos pacotes do Eclipse)
 - ✚ WST (opcional)
- ⇒ JDK 5 ou JDK 6 (JRE não é suficiente)
- ⇒ Plugin *Android Development Tools*(opcional)
- ⇒ Não é compatível com o compilador GNU para Java (gcj)

❖ Outros ambientes de desenvolvimento ou IDEs

- ⇒ JDK 5 ou JDK 6 (JRE não é suficiente)
- ⇒ Apache Ant 1.6.5 ou superior para Linux e Mac, 1.7 ou superior para Windows
- ⇒ Não é compatível com o compilador GNU para Java (gcj)

2.2.8.2 – Instalação do *Android*.

Antes de instalar, é necessário baixar o SDK através do site <http://developer.android.com/sdk/index.html>. O processo de instalação inclui apenas dois passos básicos: descompactar o conteúdo do arquivo baixado em um local do computador e adicionar o caminho da pasta tools à variável de ambiente PATH do sistema operacional. Isso permite que algumas ferramentas de linhas de comando, sejam utilizadas sem a necessidade de digitar o caminho completo de onde estão localizadas¹⁶.

Na figura 4, mostra como funciona o SDK Manager.

¹⁶ <http://developer.android.com/sdk/index.html> - Instalação do Android – Acesso em 18 de Outubro 2012.

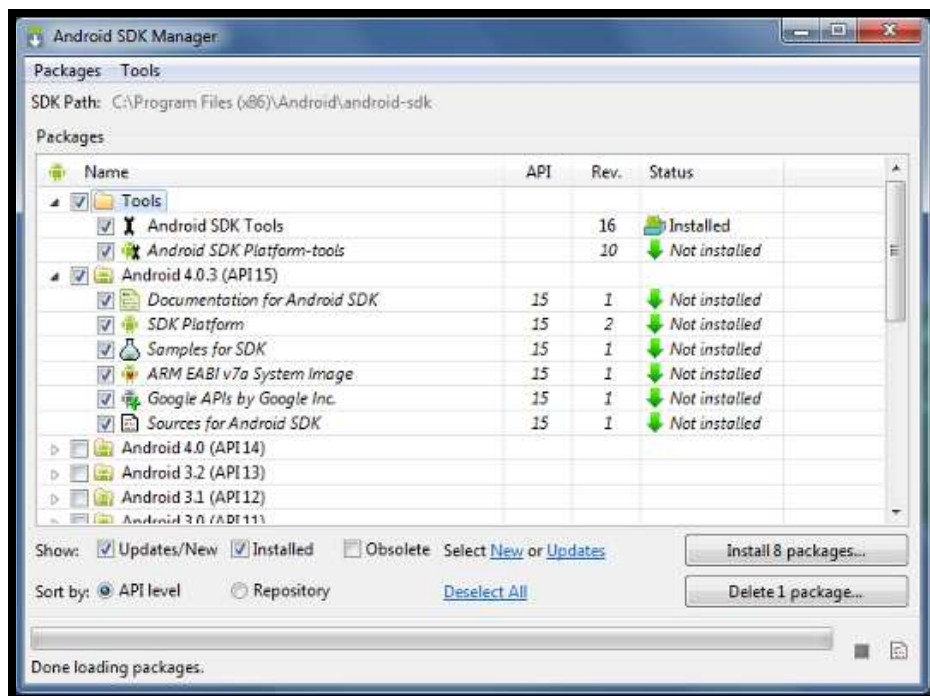


Figura 4 – Android SDK Manager

Ao utilizar o Eclipse como ambiente de desenvolvimento, é possível instalar um *plugin* chamado *Android Development Tools*. Esse *plugin* permite a integração do Eclipse com as ferramentas e projetos do *Android*, facilitando a criação, execução e busca por erros nas aplicações (Filgueiras, 2009).

Os passos descritos na tabela 2 indicam como baixar e instalar o *plugin* ADT de acordo com a versão do Eclipse escolhida.

Eclipse 3.3 (Europa)	Eclipse 3.4 (Ganymede)
<ol style="list-style-type: none"> 1. Abra o Eclipse e selecione Help> Software Updates> Find and install... 2. Na próxima janela, selecione Search for new features to install clique em Next. 3. Clique em New Remote Site. 4. Na próxima janela, digite um nome para o site remoto (ex. <i>Android Plugin</i>) e digite a URL: 	<ol style="list-style-type: none"> 1. Abra o Eclipse e selecione Help > Software Updates... 2. Na próxima janela, clique na aba Available Software. 3. Clique em Add Site... 4. Digite em Location: https://dl-ssl.google.com/android/eclipse 5. Volte para a aba Available Software, selecione Developer Tools e clique

<p>https://dl-ssl.google.com/android/eclipse</p> <p>Clique em OK.</p> <p>5. O novo site deve estar adicionado à lista de busca e selecionado. Clique em Finish.</p> <p>6. Na janela Search Results, selecione <i>Android</i> Plugin > Developer Tools. Isso irá selecionar as ferramentas <i>Android</i> Developer Toolse <i>Android</i> Editors. Essa última é opcional, porém recomendada. Caso escolha instalá-la, é necessário o pluginWST conforme mencionado nos requisitos mínimos necessários. Clique em Next.</p> <p>7. Selecione Accept terms of the license agreement clique em Next.</p> <p>8. Clique em Finish.</p> <p>9. O plugin ADT não é assinado. Aceite a instalação clicando em Install All.</p> <p>10. Reinicie o Eclipse</p>	<p>em Install...</p> <p>6. Na janela seguinte, os itens “<i>Android</i> Developer Tools” e “<i>Android</i> Editors” devem estar selecionados. Esse último é opcional, porém recomendado. Caso escolha instalá-lo, é necessário o plugin WST conforme mencionado nos requisitos mínimos necessários. Clique em Finish.</p> <p>7. Reinicie o Eclipse.</p>
---	---

Tabela 2 – Instalando o *Android* Development Tools no Eclipse.

Ao abrir o Eclipse novamente, é necessário atualizar o diretório do SDK nas preferências:

1. Selecione Window > Preferences... para abrir o painel de preferências.
2. Selecione *Android* na lista à esquerda.
3. Clique em Browse... e localize o diretório onde foi instalado o SDK.
4. Clique em Apply e depois em OK.

Apos a instalação do plugin é possível iniciar o emulador dentro do Eclipse, e instalando a aplicação automaticamente e com o debug do Eclipse integrado é possível depurar o código-fonte como qualquer outra aplicação Java.

A figura 5 ilustra o emulador do *Android* com um *Tablet*.



Figura 5 – Ilustra a simulação do *Tablet* no emulador do *Android 3.0*.

2.3 - PLATAFORMA .NET

A plataforma .NET pode ser classificada como uma referência de modelo de desenvolvimento, foi desenvolvida pela Microsoft, que visa a implementação de software independente de linguagem, plataforma e dispositivo. Um dos principais objetivos desse modelo é permitir a integração entre aplicações através da troca de informações pela internet (BURÉGIO, 2003).

A plataforma .NET é considerada o coração da estratégia .NET. Possui uma estrutura gerencia e executa vários aplicativos, como por exemplo: Web Services, contém uma biblioteca de classes denominada Framework Class Library (FCL), garante a segurança e fornece muitos outros recursos de programação. A FCL contém uma variedade de componentes reutilizáveis, evitando o problema de criar novos componentes, facilitando a vida dos programadores (DEITEL et al, 2003).

É considerada multi-plataforma por apresentar um conceito similar à tecnologia JAVA. Todo código desenvolvido, ao ser compilado, é interpretado e depurado, já contendo as verificações de lógica é transformado em uma linguagem intermediária chamada MSIL (Microsoft Intermediate Language). Essa linguagem

intermediária somente é entendida pelo CLR (Common Language Runtime) que seria semelhante à JVM (Java Virtual Machine). A interpretação é feita quando o código escrito é compilado, dessa forma, o papel da CLR é transformar o código MSIL na linguagem nativa da máquina em questão. Assim sendo, toda aplicação construída no .NET Framework pode ser executada em todas as plataformas que tem CLR's desenvolvidas (DEITEL et al, 2003).

A plataforma .NET também é considerada como multi-linguagem, pois qualquer linguagem que seja compatível com a plataforma de desenvolvimento .NET pode ser utilizada, ou seja, se um determinado compilador de linguagem segue as especificações da CLS (Common Language Specification), ela é compatível com .NET, e gera códigos MSIL compatíveis com a CLR (BURÉGIO, 2003).

A figura 6 apresenta uma visão geral do funcionamento da estrutura da plataforma .NET Framework.

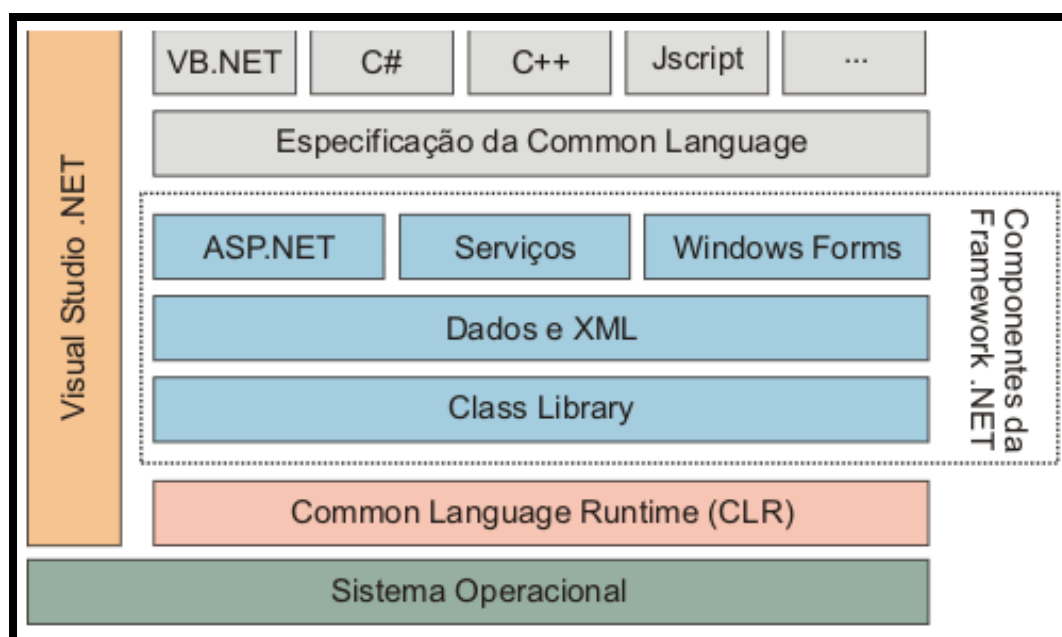


Figura 6 – Estrutura do .NET Framework.

Fonte: <http://www.linhadecodigo.com.br/artigo/458/arquitetura-de-aplicacoes-distribuidas-utilizando-net.aspx>

A plataforma .NET oferece uma variedade de funcionalidades para realização de diversas tarefas, porém, centradas na orientação a objetos e em grande integração com a internet. Dentre os recursos oferecidos, estão os Web

Services, que também são oferecidos por outras plataformas, por exemplo o Java (Santos, 2003).

2.3.1 – A Linguagem C#.

C# (Csharp) é uma linguagem de programação orientada a objetos, foi criada pela Microsoft e faz parte da sua plataforma .Net. A construção da linguagem C# baseou-se na linguagem C++ e Java. Originalmente desenvolvida por um pequeno time conduzido por dois engenheiros da Microsoft, Anders Hejlsberg e Scott Wiltamuth. Hejlsberg é também conhecido pela criação do Turbo Pascal, uma linguagem popular para programação PC, e por liderar o time que arquitetou o Borland Delphi, um dos primeiros vitoriosos ambientes de desenvolvimento integrados (Integrated Development Environments, IDEs) para programação cliente/servidor. Foi criada praticamente do zero para funcionar na nova plataforma, sem preocupações de compatibilidade com código de legado. O compilador C# foi o primeiro a ser desenvolvido. A maior parte das classes do .NET Framework foram desenvolvidas em C# (EDUCACAO, 2008).

A sintaxe do C# é altamente expressiva, mas ela também é simples e fácil de aprender. É instantaneamente reconhecida por qualquer pessoa familiarizada com C, C++ ou Java. Os desenvolvedores que sabem qualquer uma dessas linguagens são geralmente capazes de começar a trabalhar de forma produtiva com C# dentro de um tempo muito curto. Sua sintaxe do C# simplifica muita das complexidades do C++ e fornecem recursos poderosos, como tipos de valor nulo, enumerações, delegados, expressões lambda e acesso direto a memória, que não são encontrados no Java (Zemel, 2009).

O C# é uma linguagem orientada a objetos e suporta os conceitos de encapsulamento, herança e polimorfismo. Todas as variáveis e métodos, incluindo o método principal (Main), o ponto de execução de uma aplicação, é encapsulado em definições de classes. Uma classe derivada pode herdar diretamente somente de uma classe pai, mas pode herdar de qualquer quantidade de interfaces. Métodos da classe derivada que substituem métodos virtuais de uma classe pai exigem a utilização da palavra-chave override como forma de evitar a redefinição acidental (MSDN, 2012).

A figura 7 apresenta o processo de compilação da linguagem no visual C#.

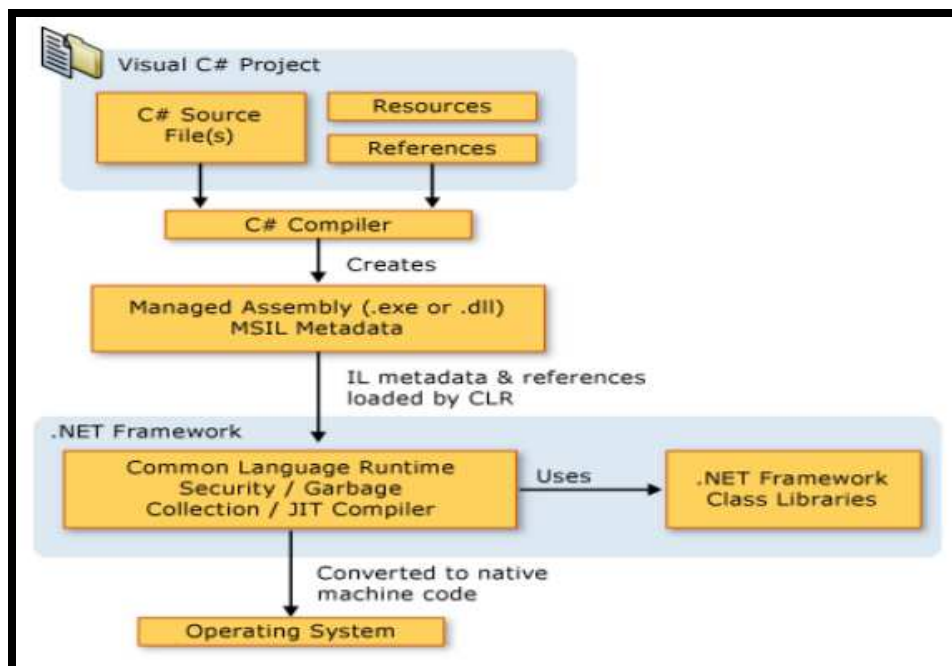


Figura 7 – Código fonte em C# para compilação em linguagem de máquina.

2.3.2 - Vantagens da Utilização do .NET FRAMEWORK.

De acordo com Venturi (2005), existem inúmeras vantagens de utilizar o .NET Framework, vamos relacionar abaixo algumas dessas vantagens:

- a) A independência da utilização de uma linguagem específica, sendo que a plataforma .NET suporta várias linguagens de programação, que podem ser utilizadas no desenvolvimento de uma infinidade de aplicações;
- b) O sistema de tipos único – Desta forma o .NET Framework possui um sistema de tipos único que pode ser utilizado por todas as linguagens compatíveis com .NET. Todos os elementos do sistema de tipos são tratados como objetos, o que permite a sua utilização em qualquer linguagem que suporte ao modelo a orientação a objetos;
- c) O modelo de aplicação unificado de acordo com as funcionalidades do .NET Framework que estão disponíveis para qualquer linguagem compatível com o .NET. Essa característica permite, que um mesmo trecho de código possa ser utilizado por aplicações desktop, Web ou até Web Services;
- d) Suporte aos principais padrões Web – Para o desenvolvimento de aplicações Web com .NET é facilitado pelo grande suporte, que o mesmo possui aos

padrões tecnológicos utilizados atualmente na internet. Como exemplos pode-se citar HTML, XML, SOAP, Web Services entre outros;

e) A facilidade de interação entre linguagens – Os objetos criados em diversas linguagens podem comunicar-se entre si e até mesmo usar uma classe criada em outra linguagem. Como exemplo pode-se definir uma classe no Visual Basic .NET e usar no C#, derivar uma classe original ou chamar um método na classe criada.

2.4 - O BANCO DE DADOS MYSQL.

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo (MILANI, 2006).

O MySQL foi criado na Suécia por dois suecos e um finlandês: David Axmark, Allan Larsson e Michael "Monty" Widenius, que têm trabalhado juntos desde a década de 1980. Hoje seu desenvolvimento e manutenção empregam aproximadamente 400 profissionais no mundo inteiro, e mais de mil contribuem testando o software, integrando-o a outros produtos, e escrevendo a respeito dele (Duarte, 2007).

O MySQL possui licenças gratuitas, comerciais, profissionais e outros, a diferença entre elas fica por conta de recursos, ou seja, na versão gratuita possuímos o banco de dados completo, porém a versão “cluster” possui alguns recursos como o controle transacional que não possuímos na versão gratuita. Atualmente o MySQL é mantido pela empresa Oracle Corporation (MILANI, 2006).

O MySQL tem alto poder de execução e de armazenamento. Dependendo da plataforma onde a ferramenta será usada, suas tabelas poderão armazenar espaços extraordinários, ficando limitadas somente ao tamanho máximo de arquivos com que a plataforma pode manipular. No caso de tabelas do tipo InnoDB, cujo o armazenamento pode ser realizado por um ou mais arquivos separados, é possível armazenar até 65.536 TB (terabytes). Quando se trata de expressões SQL, o SGBD suporta execuções de scripts SQL com até 61 uniões de tabelas (joins), e em se tratando de velocidade de execução, e pode ser enquadrado entre os mais velozes, se não o mais veloz, justamente por ter sido um dos motivos que levou seus

programadores a desenvolvê-lo, baseado em tecnologias que permitiram tal fato. (MILANI, 2006).

Entre os usuários do banco de dados MySQL estão: NASA, Friendster, Banco Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S. Army, U.S. Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems, Google, etc¹⁷.

2.5 - WEB SERVICES.

Web Services são aplicações padronizadas que podem ser descritas, publicadas e invocadas sobre uma rede, geralmente a Web. Ou seja, é uma interface que descreve uma coleção de operações que são acessíveis pela rede através de mensagens em formato XML. Permitem uma integração de serviços de maneira rápida e eficiente (Hansen, 2003).

Um Web Service é um componente de software, para a utilização de serviços Web, independente de implementação e plataforma. É descrito utilizando uma linguagem de descrição de serviços, publicado em um registro e descoberto através de um mecanismo padrão. Pode também ser invocado a partir de uma Application Program Interface (API) através da rede e ser utilizado juntamente com outros serviços (Kreger, 2001).

Os Web Services são considerados aplicações modulares e auto-descritivas, acessíveis através de um URL, independentes das plataformas de desenvolvimento e que permitem a interação entre aplicações sem intervenção humana, apresentam-se como a solução para os atuais problemas de integração entre diversos tipos de aplicações (Lopes; Ramalho, 2004).

2.5.1 - Esquema de funcionamento de um Web Service.

De acordo com Lopes e Ramalho (2004), um Web Service possui um ciclo de vida que compreende em quatro estados distintos: Publicação, Descoberta, Descrição e Invocação.

¹⁷ <http://pt.wikipedia.org/wiki/MySQL>

Publicação: Ocorre através do qual o fornecedor do Web Service reconhece a existência do seu serviço, efetuando o registro do mesmo no repositório de Web Services (UDDI).

Descoberta: Ocorre através do qual uma aplicação cliente busca conhecimento da existência do Web Service pretendido pesquisando num repositório UDDI.

Descrição: Processo pelo qual um Web Service expõe a sua API (documento WSDL); desta maneira uma aplicação cliente tem acesso a toda a interface do Web Service, onde se encontram descritas todas as funcionalidades por ele disponibilizadas, assim como os tipos de mensagens que permitem acessar as funcionalidades.

Invocação: Processo pelo qual cliente e servidor interagem, através do envio de mensagens de input e de eventual recepção de mensagem de output.

A cada um dos estados citados anteriormente, são nomeados conforme relação abaixo:

Invocação - Protocolo de comunicação (SOAP – Simple Object Access Protocol);

Descrição - Linguagem para a descrição dos seus serviços e tipo dos dados utilizados (WSDL - Web Services Description Language);

Publicação e Descoberta - mecanismo de publicação e busca dos serviços (UDDI - Universal Description, Discovery and Integration).

A Figura 8 mostra o cenário de funcionamento de um Web Services e as tecnologias envolvidas.

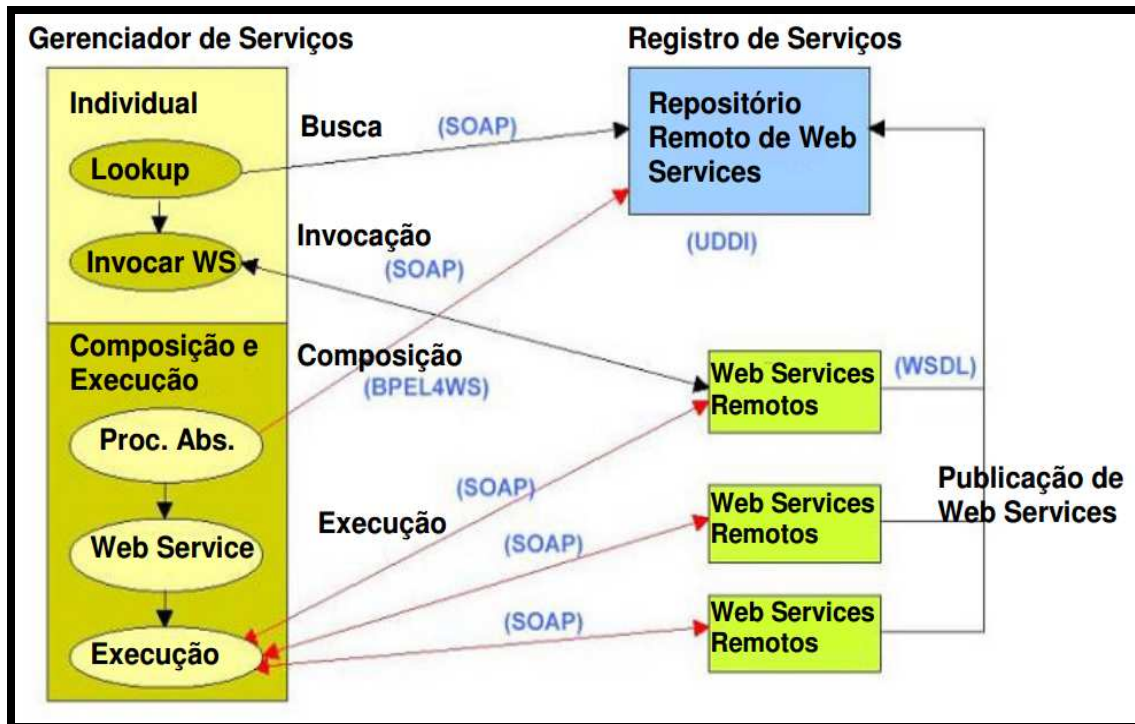


Figura 8: Cenário de Funcionamento de um Web Services.

Conforme Lopes, Ramalho (2004), a configuração dos quatros estados citados acima, permite construir o ciclo de vida de uma Web Service, conforme processos mencionados abaixo:

- O fornecedor constrói o serviço utilizando a linguagem de programação que preferir;
- Estabelece a interface/assinatura do serviço que foi definida em WSDL;
- O fornecedor deve registrar o serviço no UDDI;
- O utilizador (aplicação ao cliente) realiza a pesquisa num repositório UDDI e encontra o serviço;
- A aplicação cliente estabelece a ligação com Web Service e estabelece a troca de informações com o fornecedor, via mensagens SOAP.

2.5.2 – SOAP (Simple Object Access Protocol).

SOAP originou-se da idéia de um mecanismo de RPC (Remote Procedure Calls) baseado em XML originalmente proposto por Dave Winer em 1998. SOAP é uma especificação da W3C proposta por organizações como Userland®, Ariba®, Microsoft®, IBM®, Compaq®, HP®, Lotus®, SAP® entre outras. É um protocolo

elaborado para facilitar a chamada remota de funções via Internet, permitindo que dois programas se comuniquem de uma maneira muito semelhante às páginas Web (Becker; Sobral; Claro).

A comunicação entre clientes e servidores é feita através do protocolo SOAP. Dessa maneira, as RPC (Remote Procedure Calls - Chamadas Remotas de Procedimento) são codificadas em XML e operam sobre HTTP. Com isso, podem-se acessar os serviços de um objeto localizado em outro local da rede, através de uma chamada local a este objeto. (Becker; Sobral; Claro).

A especificação SOAP (definida pela W3C) define as seguintes informações, como necessárias em toda chamada de RPC:

- + A URI, equivalente à URL, do HTTP do objeto alvo;
- + Nome do método;
- + Os parâmetros do método (requisição ou resposta);
- + Uma assinatura do método opcional;
- + Um cabeçalho opcional.

Uma mensagem SOAP consiste basicamente em uma estrutura com os elementos apresentados na figura 9.

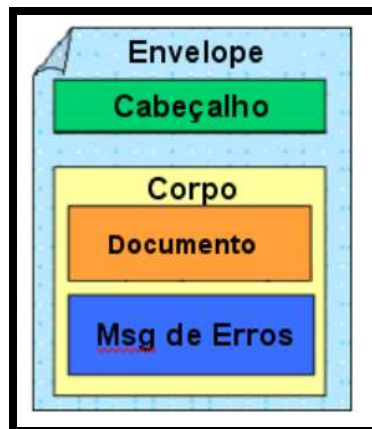


Figura 9: Elementos de uma mensagem SOAP.

Envelope: Toda mensagem SOAP deve ter um. É o elemento raiz do documento XML. O Envelope pode conter declarações de *namespaces* e também atributos adicionais, como o que define o estilo de codificação, que define como os dados são representados no documento XML (Basci, 2009).

Cabeçalho (Header): É um cabeçalho opcional. Ele carrega informações adicionais, como por exemplo, se a mensagem deve ser processada por um

determinado nó intermediário. Quando utilizado, o cabeçalho deve ser o primeiro elemento do Envelope (Becker; Sobral; Claro).

Corpo: O corpo é um elemento obrigatório e contém o *payload*, ou seja, a informação a ser transportada para o seu destino final (Basci, 2009).

2.5.3 - WSDL - Web Services Description Language.

A descrição de um Web Service se dá através de um arquivo WSDL, cujo formato está em XML. Este arquivo tem como objetivo descrever a interface do serviço, ou seja, os métodos que são disponibilizados pelo Web Service, parâmetros recebidos, a resposta enviada e ainda o processo de comunicação com o servidor SOAP. É através do WSDL que o UDDI (Universal Description, Discovery and Integration) descreve os detalhes de localização e chamada ao Web (Basci, 2009).

O WSDL é extensível para permitir a descrição de serviços e suas mensagens, independentemente dos formatos de mensagem e dos protocolos de rede que sejam usados (Becker; Sobral; Claro).

Esse padrão possui cinco elementos importantes, divididos em duas camadas: abstrata e concreta. A definição abstrata é composta por tipos, mensagens e port (portas). Os tipos de dados são aqueles que são enviados e recebidos dentro da mensagem. Depois de definidos os tipos, são definidos os dados que serão enviados pelo sistema através da definição de mensagem. Já port trata-se de uma série de operações relacionadas à mensagem, como entrada, saída e tratamento de exceções (Becker; Sobral; Claro).

2.5.4 UDDI - Universal Description, Discovery and Integration.

O UDDI é um grande cadastro global. São as páginas amarelas dos Web Services. Como as páginas amarelas tradicionais, pode-se procurar por uma companhia que ofereça os serviços necessários, ler sobre o serviço oferecido ou contatar alguém para obter mais informações (Becker; Sobral; Claro).

De acordo com Marcílio (2006), um diretório UDDI é um arquivo XML que descreve o negócio e os serviços. O registro possui três partes:

- ✓ Páginas brancas: descrevem a companhia: nome, endereço e contatos;
- ✓ Páginas amarelas: incluem as categorias, baseada em taxonomias padrões.
- ✓ Páginas verdes: descrevem a interface para o serviço, em nível de detalhe suficiente para se escrever uma aplicação que use o Web Service.

A maneira como os serviços são definidos no documento UDDI é chamado *Type Model* ou *tModel*. Em muitos casos, o *tModel* contém o arquivo WSDL que descreve a interface SOAP de um Web Service (Becker; Sobral; Claro).

O diretório UDDI também inclui várias maneiras de procurar os serviços. Por exemplo, pode-se procurar por fornecedores de um serviço em uma localização geográfica específica ou por negócios de um tipo específico. Os provedores de serviço registram ou publicam seus Web Services nesse diretório e um mecanismo de descoberta localiza o Web Service para um cliente, mediante uma pesquisa. A pesquisa pode ser por tipo de serviço ou por fornecedor (Marcílio, 2006).

O Web Services é uma tecnologia que se constitui de outras tecnologias e padrões que estão sendo desenvolvidas há pouco tempo. Estes padrões especificam meios de troca de mensagens entre serviços (SOAP), busca de serviços em registros (UDDI) e configuração dinâmica de clientes baseadas em descrições de serviços (WSDL) (Lopes; Ramalho, 2004).

A Figura 10 mostra o funcionamento de um registro UDDI dentro do Web Service.

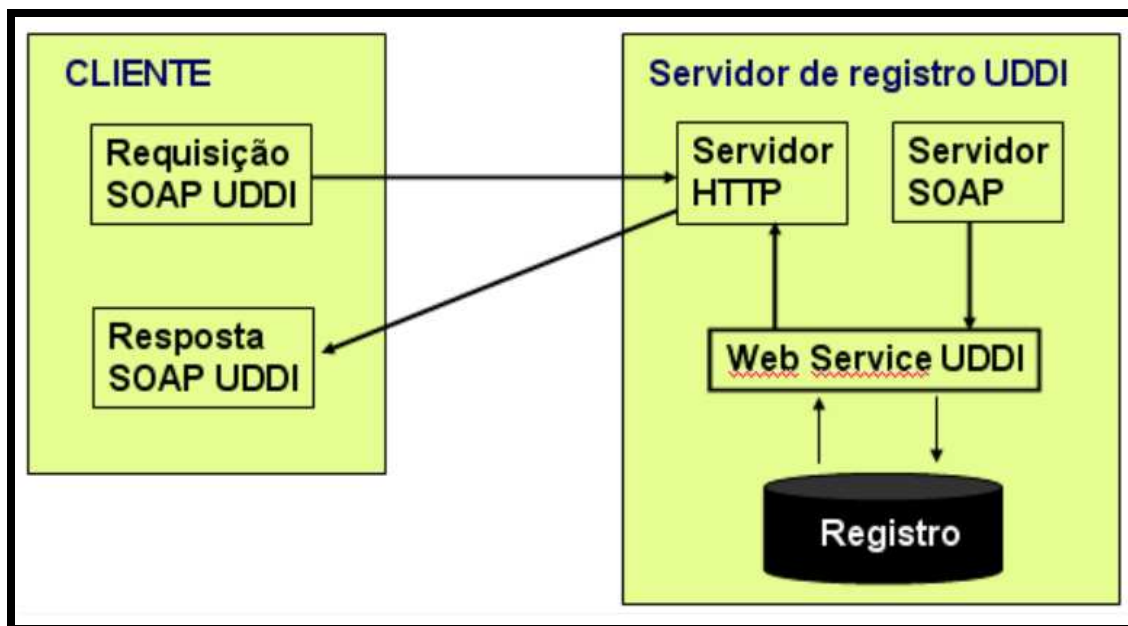


Figura 10 – Funcionamento do Registro UDDI.

A figura 11 representa a interoperabilidade e padrões na qual são desenvolvidos Web Services.

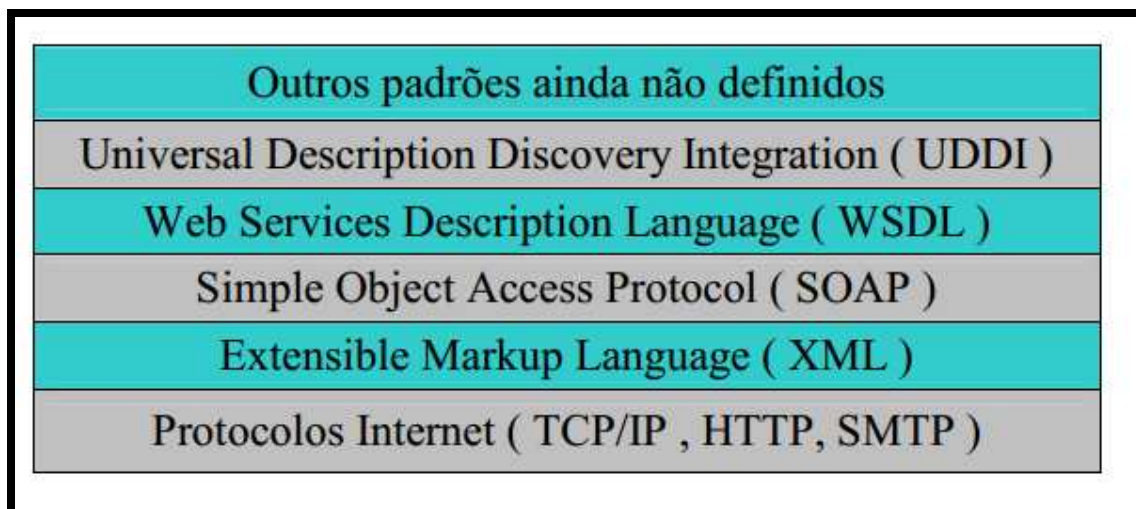


Figura 11: Pilhas de protocolos que compõem um Web Services.

A Figura 12 relacionada abaixo, mostra o cenário de padrões para Web Services de uma forma mais detalhada, desde os componentes dos protocolos de comunicação SOAP (TCP/IP, HTTP e XML), os protocolos de definição, busca e divulgação (WSDL e UDDI) e as linguagens de composição no nível mais elevado

da hierarquia, que são responsáveis pela segurança, a qualidade de serviço (QoS) e gerenciamento dos Web Services.



Figura 12: Cenário detalhado de padrões Web Services.

2.5.5 - Modelo de dados JSON.

Uma boa explicação em português do modelo de dados JSON encontra-se no site JSON.org.

JSON está constituído em duas estruturas:

Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object*, *record*, *struct*, dicionário, *hash table*, *keyed list*, ou *arrays* associativos.

Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência¹⁸.

O primeiro elemento sintático de um documento JSON pode ser o caractere { para denotar o início de um dicionário, ou o caractere [para indicar o início de uma lista¹⁹.

Estas duas estruturas podem ser alinhadas arbitrariamente, sendo assim, árvores ou hierarquias podem ser representadas. Além de poderem ser dicionários ou listas, os valores internos podem ser tipos simples como números, strings ou as constantes true, false e null.

¹⁸ Introdução ao JSON. Disponível em: <<http://www.json.org/json-pt.html>>. Acesso em 17 Outubro 2012.

¹⁹ CROCKFORD 2006b. “A JSON text is a serialized object or array”. Tradução: “Um texto JSON é um objeto ou um array serializado”

Os números seguem a sintaxe de JavaScript, podendo representar inteiros ou ponto flutuante.

As strings são cadeias de caracteres Unicode representando textos de tamanho arbitrário em qualquer idioma, codificadas em UTF-8 ou ASCII puro.

2.5.6 - Comparação entre JSON e XML.

Tanto XML quando JSON são padrões internacionais com fundamentos sólidos. XML é formalizado em recomendações do W3C²⁰, e é baseado na SGML (Standard Generalized Markup Language), que é normatizada como ISO-8879²¹. JSON é formalizado em no RFC-4627 e é baseado em um subconjunto da sintaxe de JavaScript, que é normatizada como ECMA-262 e como ISO/IEC 16262 (CROCKFORD, 2006).

XML é um padrão mais antigo e ambicioso, tem maior aceitação no mercado, pretendendo ser um formato de intercâmbio que atenda as necessidades de troca de documentos semi estruturados complexos e variados, bem como o intercâmbio de dados em forma de registros relativamente mais simples, estruturados e uniformes (Santos, 2003).

JSON tem um objetivo mais modesto, de servir apenas como padrão para a transmissão de registros. Seu uso se disseminou com a arquitetura de aplicações AJAX, como formato de transporte de dados entre servidores Web e navegadores (ironicamente, o X de AJAX se refere a XML, mas hoje o formato JSON é o preferido neste tipo de aplicação, por ser mais compacto e de interpretação mais fácil no navegador, graças ao uso da sintaxe de JavaScript) (Santos, 2003).

JSON nasceu como uma alternativa mais simples para substituir o uso de XML, mas não em qualquer tipo de aplicação, e sim num subconjunto delas, ou seja, para cada caso, deve-se analisar qual será a melhor solução. Antes de apresentar mais formalmente o formato JSON, vamos apresentá-lo informalmente por meio de comparações com XML, a fim de deixar mais claro os problemas que JSON tenta solucionar.

²⁰ Recomendações disponíveis em <<http://www.w3.org/XML/Core/>>. Acesso em 17 Outubro 2012.

²¹ ISO 8879:1986 Information processing — Text and office systems — Standard Generalized Markup Language (SGML).

Por ter um objetivo mais limitado, JSON é mais simples do que XML. Por exemplo, apenas a codificação de caracteres UTF-8 é aceita, e esta é a codificação recomendada pelo W3C para uso geral²².

Em XML é possível especificar uma codificação de caracteres arbitrária. Do ponto de vista estrutural, JSON evita duas complexidades de XML que são úteis na marcação de documentos narrativos, mas não na transmissão de registros de bases de dados: a dualidade entre conteúdos e atributos, e o conteúdo misto. Vamos exemplificar cada uma destas questões (CROCKFORD, 2006).

Conforme Santos (2003), exemplifica a qualidade entre conteúdos e atributos, suponha que temos um registro de aluno com campos, “nome” e “número de matrícula”. Duas variantes possíveis em XML são:

```
<aluno>
  <matricula>123456</matricula>
  <nome>Fulano de Tal</nome>
</aluno>
```

ou então:

```
<aluno matricula="123456">
  <nome>Fulano de Tal</nome>
</aluno>
```

Há outras possibilidades.

Em JSON, a representação natural é uma só:

```
{“aluno” : {“matricula” : “123456”,
           “nome” : “Fulano de Tal”} }
```

A idéia de atributos em XML foi herdada de SGML, uma linguagem criada para a marcação de documentos para publicação eletrônica. No contexto de publicação, um atributo é útil para associar ao texto metadados ou parâmetros que não fazem parte do fluxo principal de texto, como por exemplo, associar um estilo visual a um parágrafo, ou uma URI a uma referência, enfim, informações que não aparecerão para o ser humano que ler o documento (CROCKFORD, 2006).

²² “The examples above show declarations for UTF-8 encoded content. This is likely to be the best choice of encoding for most purposes, but it is not the only possibility.” Disponível em: <<http://www.w3.org/International/O-charset>>, acesso em 17 Outubro 2012. Ver também Recommended list of Doctype declarations<<http://www.w3.org/QA/2002/04/valid-dtd-list.html>>, acesso em 17 Outubro 2012.

Porém, ao usar XML como formato para representação de registros, a escolha entre representar um dado como conteúdo ou atributo provoca dúvidas e discussões recorrentes (Santos, 2003).

JSON não implementa o conceito de atributos, evitando esse dilema.

Outra complexidade estrutural do XML que JSON evita completamente é o chamado conteúdo misto. Eis um exemplo de XML com conteúdo misto:

```
<p>O uso de XML para dados organizados em <strong>registros</strong>
é um mero acidente.</p>
```

No trecho acima, temos um nodo `<p>` com três nodos filhos:

- a) um nodo texto com o conteúdo “O uso de XML para dados organizados em ”;
- b) um elemento `` com o conteúdo “registros”;
- c) um nodo texto com o conteúdo “ é um mero acidente.”

No formato JSON simplesmente não existe conteúdo misto. Caso fosse necessário representar o texto acima em JSON, teríamos que recorrer à criação de novos itens, seguindo alguma convenção. Mas o caso é que JSON foi explicitamente concebido para representar registros que são fundamentalmente associações entre nomes e valores, e é por isso que não possuem uma forma de representar valores anônimos desassociados, como os nodos do texto do XML²³.

2.5.7 - Utilização da Arquitetura REST no desenvolvimento de aplicações Web.

Os pequenos aplicativos embutidos nos dispositivos moveis estão cada vez mais utilizando os recurso de comunicação a sistemas Web. A maioria desses aplicativos é extremamente dependente de dados para serem úteis e esses dados podem vir dos mais variados lugares, um exemplo, é a plataforma *Android* que disponibiliza ao desenvolvedor um banco de dados SQLite, onde ele poderá criar suas tabelas, armazenar e buscar dados, mas cada vez mais esses dados vêm de serviços Web (RICHARDSON; RUBY, 2007).

A Web é uma plataforma “orientada a recursos”. Um Recurso pode ser definido como qualquer coisa que é exposta a Web através de um identificador e

²³ XML: Extensible Markup Languagein: Network Dictionary. Disponível em <http://www.networkdictionary.com/software/xml.php>. Acesso em 17 Outubro 2012. “XML oferece a possibilidade de representar as estruturas de dados mais gerais da ciência da computação: registros, listas e árvores.”. Nossa tradução.

que possamos manipular, ou seja, invocar um método de leitura ou escrita. (WEBBER; PARASTATIDIS, 2010).

Desde o surgimento do REST (Transferência de Estado Representacional) tornou-se um termo, mais utilizado na arquitetura de software de sistemas Web que está cada vez mais integrando em suas aplicações. Esta arquitetura foi proposta pelo Dr. Roy T. Fielding em 2000 e desde então vem sendo adotada em vários sistemas de grande porte como Twitter, Facebook, Flickr e todas as APIs de serviços públicos do Google como Google Agenda, Google Health, Google Data e Google Maps (WEBBER; PARASTATIDIS, 2010).

2.5.7.1 - Definição do Rest.

O protocolo HTTP, é considerado como servidores em HTTP, ou seja, é um protocolo simples e não apresenta muitos recursos. Em sua primeira versão ele apresentou endereçamento e *statelessness*: duas características de seu projeto que tornou um avanço perante seus rivais e que ainda o mantém estável mesmo nos mega-sites de hoje (RICHARDSON; RUBY, 2007).

Em sistemas que utilizam esta arquitetura são sistemas Clientes-Servidores comuns, entretanto suas requisições e respostas são construídas ao redor da transferência da representação de recursos. Recursos são os blocos fundamentais de sistemas baseados na web, ao ponto que a Web é considerada como “orientada a recursos” (WEBBER; PARASTATIDIS, 2010). A abstração chave de informação do REST são os recursos que ele oferece. Qualquer informação que pode ser nomeada pode ser um recurso: documentos, imagens, informações sobre o tempo, uma pessoa, e assim sucessivamente (FIELDING, 2000).

De acordo com Fielding (2000), a arquitetura REST está fundamentada sob o protocolo HTTP e os seus devidos métodos. A maioria dos sistemas de hoje em dia, segue a seguinte regra dos métodos HTTP, concebidos para simular operações de CRUD:

GET:Leitura;

POST:Escrita;

PUT:Alteração ou Atualização;

DELETE:Remoção.

Estas aplicações novas estão utilizando os princípios REST a fim de obterem um bom nível de escalabilidade em seu projeto. Normalmente essas aplicações disponibilizam a seus usuários uma API dos métodos REST disponíveis, ou seja, os seus métodos de entradas e os métodos de saídas (RICHARDSON; RUBY, 2007).

3 – ANÁLISE E ESPECIFICAÇÃO DO PROTÓTIPO

Neste capítulo, serão apresentados o problema e a sua modelagem.

3.1 – DESCRIÇÃO DO PROBLEMA.

Desenvolver um protótipo de um Sistema de Gerenciamento de Custos de Equipamentos Agrícolas, ao qual no item 3.2 será apresentada toda a idéia geral de como funcionaria este sistema, destacando-se os pontos a serem implementados para efeito de pesquisa de integração de Software das tecnologias abordadas neste trabalho. Através deste sistema é controlado o que é gasto por cada tipo de Equipamento Agrícola, ou seja, por exemplo, considerando para um determinado modelo de trator “XX” da marca “Y” e frota “Z”, trabalhando em uma determinada área e realizando um determinado tipo de serviço, este sistema tem como objetivo fazer o controle de todos os custos envolvidos, na operação e manutenção deste equipamento. Levando em consideração o envolvimento dos custos, desde o abastecimento de diesel, as revisões periódicas, as reformas de pós-safra e todas as manutenções envolvidas com este equipamento, desta forma, o sistema vai realizar os cálculos de custo geral por cada hora trabalhada deste equipamento, ou os custos por hora de uma determinada área trabalhada por aquele equipamento, e em seguida fornecendo qualquer tipo de relatório envolvendo os custos destes equipamentos.

O desenvolvimento dos dois protótipos será da seguinte forma. Será desenvolvido duas aplicações, sendo uma aplicação para o Sistema Operacional *Android* que utiliza o dispositivo móvel conhecido como *Tablet* que tem acesso ao seu banco de dados interno no *Android* que é o SQLite, que irá demonstrar uma movimentação entre as tabelas cadastradas neste banco. A outra aplicação será desenvolvida no Sistema Operacional Windows, ao qual com o uso da plataforma .NET e seu ambiente de programação de alto nível, será desenvolvido uma aplicação em C#, que será o nosso módulo Desktop que irá conectar-se com o Banco de Dados MySQL que está instalado na mesma máquina que o C#. No banco de dados MySQL será desenvolvido as mesmas tabelas que estão no banco de dados SQLite do *Android*. Por fim, será utilizada a tecnologia Web Service que fará a integração da comunicação entre os dois bancos de dados utilizados.

A figura 13, ilustrada abaixo, mostra a modelagem do funcionamento dos dois sistemas, ou seja, como será a nossa arquitetura a ser desenvolvida para os dois protótipos mencionados acima.

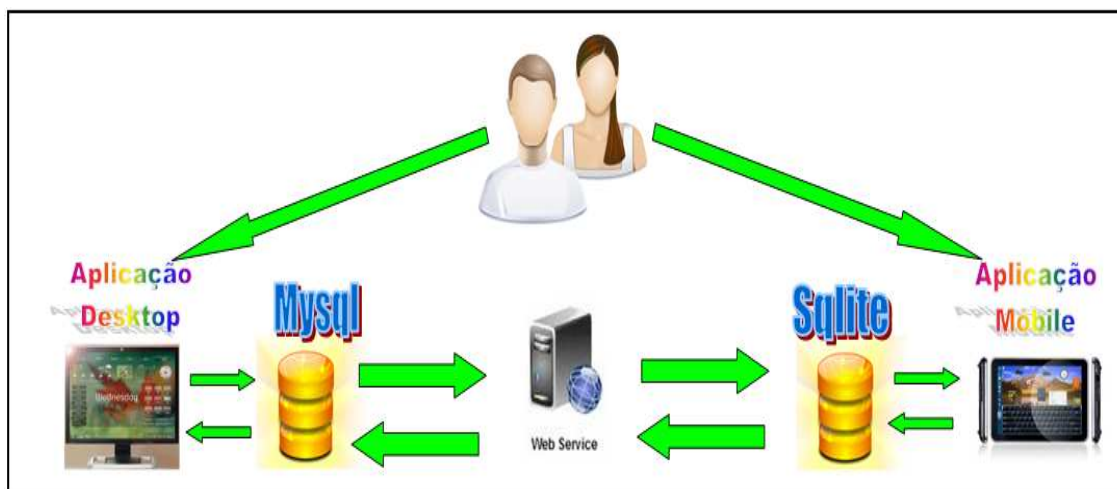


Figura 13 – Mostra a modelagem do funcionamento dos dois sistemas.

3.2 – MAPA MENTAL DO SISTEMA.

Nesta etapa do trabalho, foi utilizado o mapa mental, que é considerado um padrão mundial para a criação, gerenciamento e comunicação de idéias. Através dele conseguimos organizar as idéias de conhecimento por meio de uma visualização intuitiva e amigável, possuindo grande versatilidade visual.

Nos mapas mentais, inicia-se com uma idéia central do problema, onde todos os ramos do mapa significam uma decomposição da idéia principal em idéias relacionadas, demonstrando em um modelo visual e amigável para a sua interpretação.

Na Figura 14, está um mapa mental demonstrando uma visão geral de como seria a implementação do Sistema de Gerenciamento de Custos de Equipamentos Agrícolas, mas para efeito de pesquisas deste trabalho, foi destacado no mapa mental com o símbolo do Pingüim 🐧 quais os casos de usos que serão desenvolvidos tanto no C# como no *Tablet*.



Figura 14 – Mapa Mental demonstrando uma visão geral do sistema.

OBS: A visualização desta imagem encontra-se no Anexo, pag. 112.

3.3 – LISTA DE EVENTOS DO SISTEMA.

Para a realização da modelagem do comportamento dos sistemas baseados em orientação a objetos, são determinados quais os eventos que acontecem no sistema. Abaixo estão mencionados os principais eventos relacionados com as entidades que interagem com o sistema, destacando-se em azul o que será desenvolvido nas duas aplicações, sendo uma aplicação para Desktop e outra para o dispositivo móvel (*Tablet*).

1. Cadastrar Veículos
2. Cadastrar Abastecimentos
3. Cadastrar Estados
4. Cadastrar Cidades
5. Cadastrar Cargos

6. Cadastrar Funcionários
7. Cadastrar Fornecedores
- 8. Cadastrar Itens**
9. Cadastrar Tarefas das Áreas
10. Cadastrar Marcas
11. Cadastrar Modelos
- 12. Cadastrar Equipamentos**
- 13. Cadastrar Localização das Áreas**
14. Cadastrar Tipos de Manutenção
15. Cadastrar Sistemas de Lubrificação
16. Cadastrar Sub-Sistemas de Lubrificação
- 17. Cadastrar Usuários**
18. Cadastrar Tabelas dos Usuários
19. Inserir Lançamento dos Abastecimentos
- 20. Inserir Lançamento das Tarefas dos Equipamentos**
- 21. Inserir Lançamento das Ordens de Serviços**
- 22. Inserir Lançamento dos Itens das Ordens de Serviços**
23. Inserir Lançamento de KM nas Ordens de Serviços
24. Inserir Lançamento de MO nas Ordens de Serviços
25. Inserir Lançamento das Lubrificações do Equipamentos
26. Emitir Relatório dos Custos dos Equipamentos
27. Emitir Relatório das Ordens de Serviços
28. Emitir Relatório das Tarefas dos Equipamentos
29. Emitir Relatório de Lubrificação dos Equipamentos
30. Emitir Relatório dos Abastecimentos

3.4 – CASOS DE USOS DO SISTEMA.

Um caso de uso serve para especificar uma seqüência de ações que um sistema realiza e produz para um determinado ator em particular.

Segue abaixo na Figura 15, uma visão geral dos casos de usos de como seria o Sistema de Gerenciamento de Custos de Equipamentos Agrícolas, ao qual foi realizado o levantamento dos requisitos. Nesta figura estão destacados em vermelho os setes casos de usos que serão desenvolvidos nas duas aplicações,

sendo que os sete casos de usos com o (CRUD) serão desenvolvidos para Desktop e três casos de usos com o (CRUD) de movimentações das tabelas serão desenvolvidos no *Tablet* ao qual será demonstrado mais adiante neste trabalho.

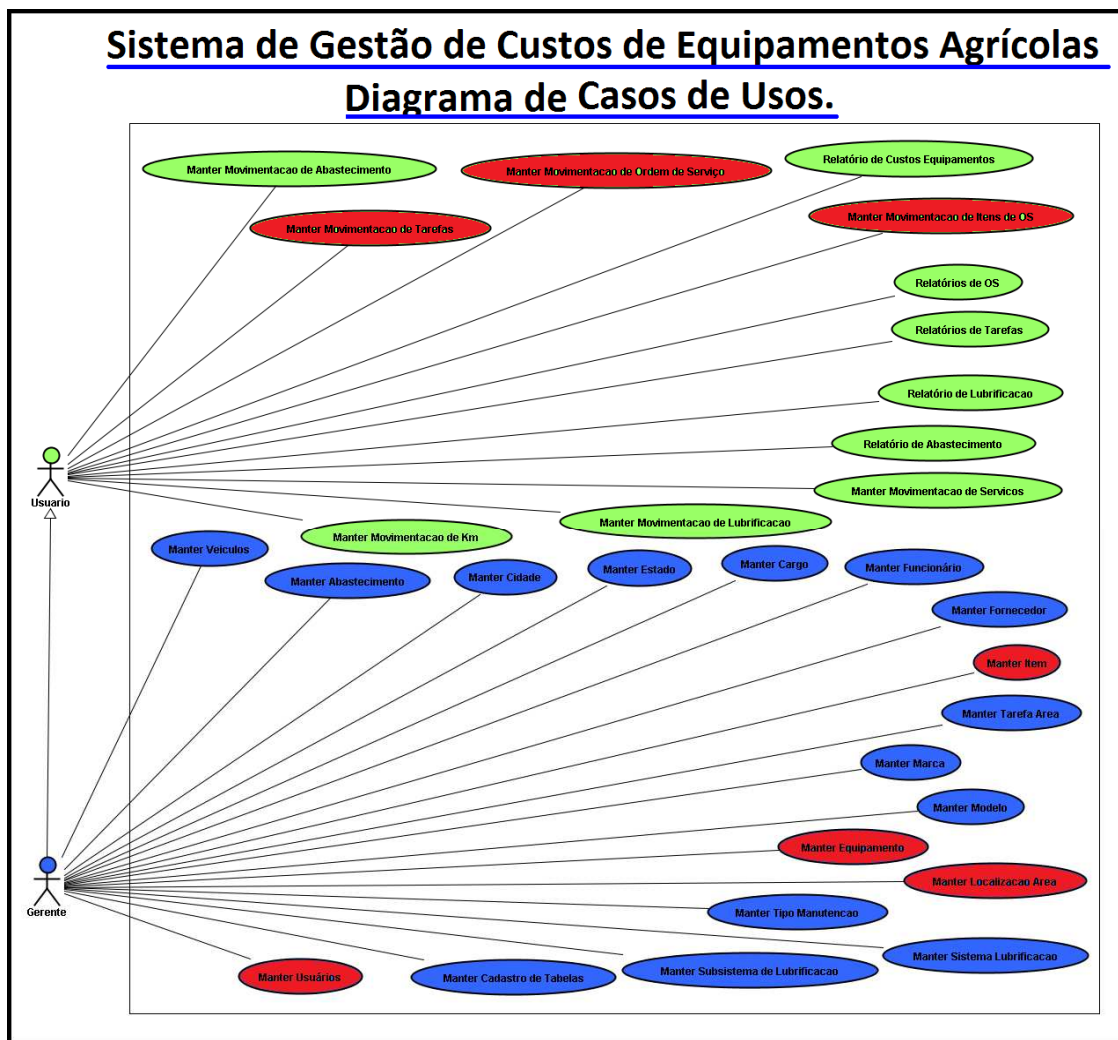


Figura 15 – Visão Geral dos Casos de Usos do Sistema.

OBS: A visualização desta imagem encontra-se no Anexo, pag. 113.

3.5 – DIAGRAMA DE ENTIDADE E RELACIONAMENTO (DER).

O Diagrama de Entidade e Relacionamento é baseado na percepção do mundo real, que consiste em um conjunto de objetos básicos chamados de entidades, que são relacionamentos uns com os outros na busca da solução de um problema.

Será apresentado neste trabalho, como será o processo de sincronismo destas tabelas dos dois bancos de dados.

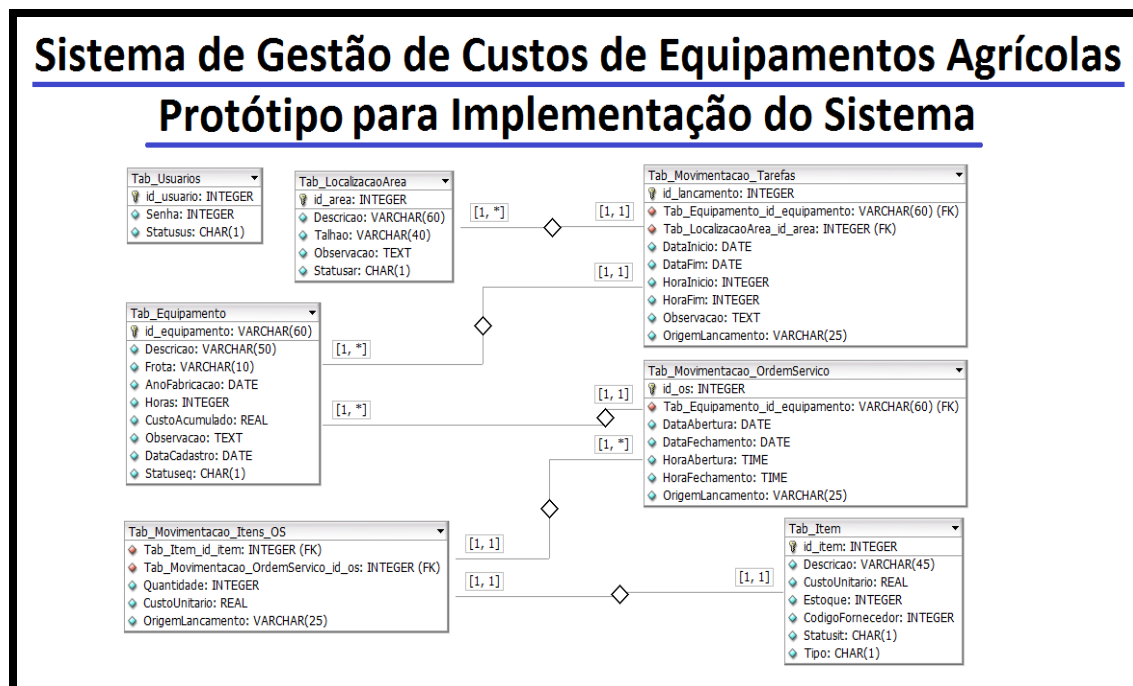


Figura 17 – Diagrama de Entidade Relacionamento que será desenvolvido.

3.6 – LISTA DE EVENTOS DO DESENVOLVIMENTO DOS PROTÓTIPOS.

Para modelagem do comportamento do sistema baseado em orientação a objetos, determinam-se quais os eventos que acontecem. Eventos fazem com que os sistemas realizam várias ações, ou seja, é sobre eles que aplicamos o (CRUD). A seguir são descritos os eventos relacionados com o desenvolvimento dos dois protótipos.

1. Cadastrar Itens
2. Cadastrar Equipamentos
3. Cadastrar Localização das Áreas
4. Cadastrar Usuários
5. Inserir Lançamento das Tarefas dos Equipamentos
6. Inserir Lançamento das Ordens de Serviços
7. Inserir Lançamento dos Itens das Ordens de Serviços

3.7 – CASOS DE USOS.

Na figura 18, serão apresentados os casos de usos utilizados no processo de desenvolvimento dos dois protótipos.

- ✓ **Manter Movimentação de Tarefas:** Responsável por todas as operações relacionadas às tarefas que cada equipamento irá fazer.
- ✓ **Manter Movimentação de Ordens de Serviços:** Responsável por todas as operações relacionadas ao processo de Ordens de Serviços de cada equipamento.
- ✓ **Manter Movimentação de Itens de Ordens de Serviços:** Responsável por todas as operações relacionadas aos Itens que serão lançados nas Ordens de Serviços.
- ✓ **Manter Usuários:** Responsável por todas as operações relacionadas aos usuários do sistema.
- ✓ **Manter Item:** Responsável por todas as operações relacionadas os itens que serão lançados nas Ordens de Serviços.
- ✓ **Manter Equipamento:** Responsável por todas as operações relacionadas aos equipamentos que serão controlados.
- ✓ **Manter Localização de Área:** Responsável por todas as operações relacionadas à localização que cada equipamento irá trabalhar.

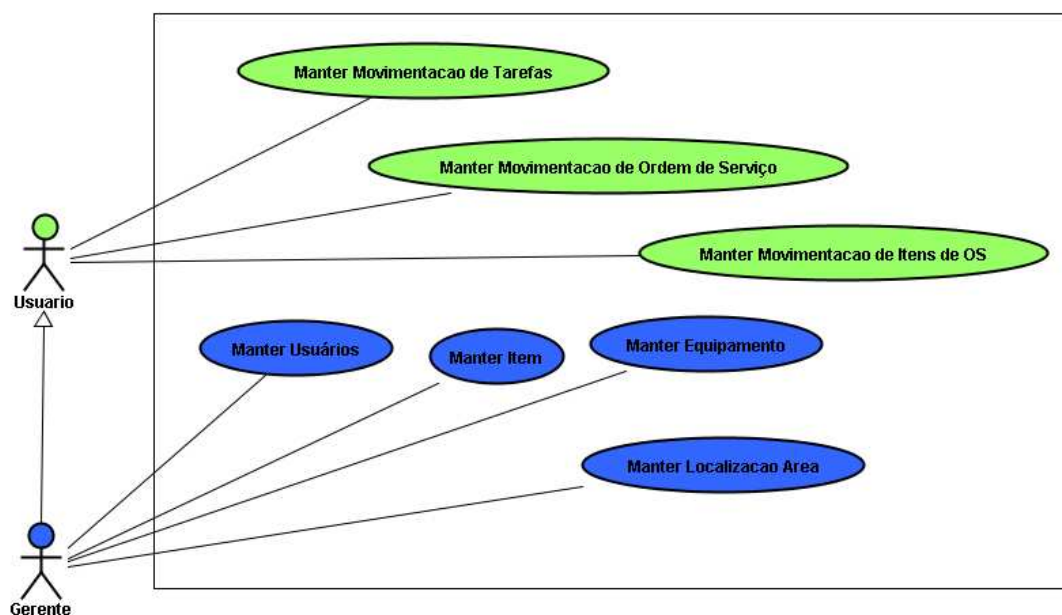


Figura 18 – Casos de Usos utilizados no desenvolvimento dos protótipos.

3.8 – ESPECIFICAÇÃO DOS CASOS DE USOS.

Conforme, foram relacionados acima quais são os casos de usos a serem utilizados nos dois protótipos que será desenvolvido, abaixo serão realizadas as especificações de cada casa de uso, ou seja, como será o seu comportamento dentro dos protótipos.

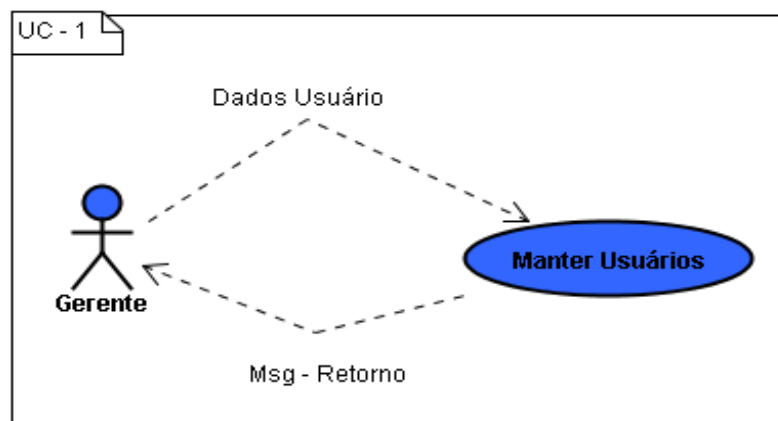


Figura 19 – UC1 – Diagrama de Use Case – Manter Usuários.

Nome da Use Case 1	Manter Usuário
Ator(es)	Gerente
Pré-Condição	O gerente deverá estar autenticado no sistema.
Cenário Principal	1- O sistema solicita os dados necessários para o cadastro do usuário. 2- O gerente informa os dados de acordo com os campos a serem preenchidos na tela. 3- O gerente seleciona a opção Cadastrar 4- O sistema emite a mensagem “Usuário cadastrado com sucesso”. 5- O sistema cadastra o usuário no banco de dados.
Cenário Alternativo	O gerente poderá cancelar o processo durante o cadastro.
Casos de Teste	3.1- O sistema verifica se os campos foram preenchidos corretamente. 3.2- O sistema não confirma o cadastro e emite uma Mensagem de erro. 3.3- O sistema cancela a operação e limpa a tela.

Tabela 3 - Manter Usuário.

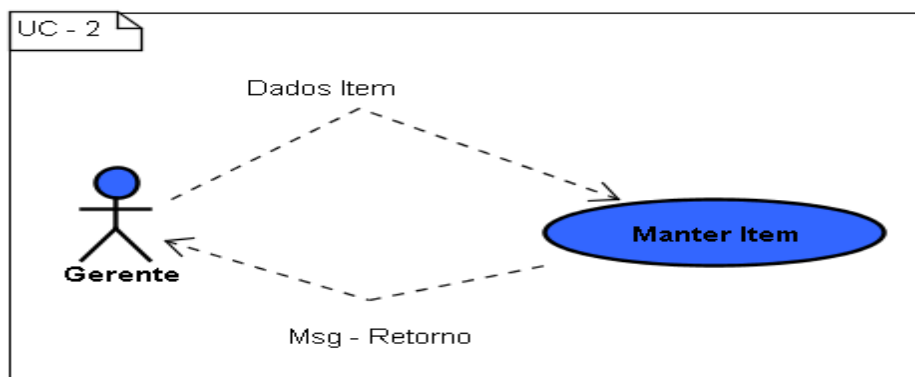


Figura 20 – UC2 – Diagrama de Use Case – Manter Item.

Nome da Use Case 2	Manter Item.
Ator(es)	Gerente
Pré-Condição	O gerente deverá estar autenticado no sistema.
Cenário Principal	1- O sistema solicita os dados necessários para o cadastro do Item. 2- O gerente informa os dados de acordo com os campos a serem preenchidos na tela. 3- O gerente seleciona a opção Cadastrar 4- O sistema emite a mensagem “Item cadastrado com sucesso”. 5- O sistema cadastra o Item no banco de dados.
Cenário Alternativo	O gerente poderá cancelar o processo durante o cadastro.
Casos de Teste	3.1- O sistema verifica se os campos foram preenchidos corretamente. 3.2- O sistema não confirma o cadastro e emite uma Mensagem de erro. 3.3- O sistema cancela a operação e limpa a tela.

Tabela 4 - Manter Item.

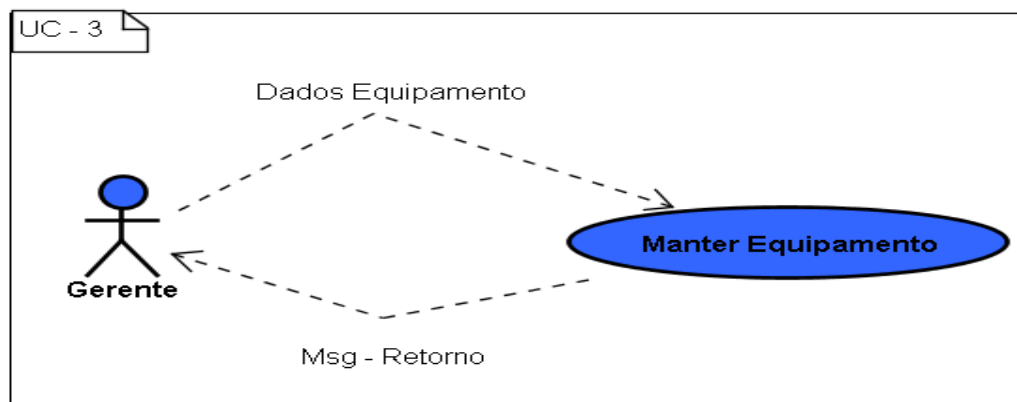


Figura 21 – UC3 – Diagrama de Use Case – Manter Equipamento.

Nome da Use Case 3	Manter Equipamento.
Ator(es)	Gerente
Pré-Condição	O gerente deverá estar autenticado no sistema.
Cenário Principal	<p>1- O sistema solicita os dados necessários para o cadastro do Equipamento.</p> <p>2- O gerente informa os dados de acordo com os campos a serem preenchidos na tela.</p> <p>3- O gerente seleciona a opção Cadastrar</p> <p>4- O sistema emite a mensagem “Equipamento cadastrado com sucesso”.</p> <p>5- O sistema cadastra o Equipamento no banco de dados.</p>
Cenário Alternativo	O gerente poderá cancelar o processo durante o cadastro.
Casos de Teste	<p>3.1- O sistema verifica se os campos foram preenchidos corretamente.</p> <p>3.2- O sistema não confirma o cadastro e emite uma Mensagem de erro.</p> <p>3.3- O sistema cancela a operação e limpa a tela.</p>

Tabela 5 - Manter Equipamento.

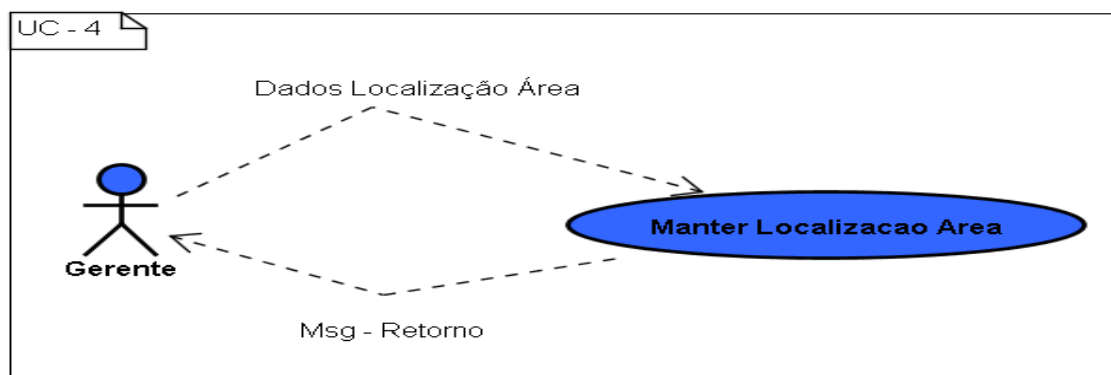


Figura 22 – UC4 – Diagrama de Use Case – Manter Localização Área.

Nome da Use Case 4	Manter Localização Área.
Ator(es)	Gerente
Pré-Condição	O gerente deverá estar autenticado no sistema.
Cenário Principal	1- O sistema solicita os dados necessários para o cadastro da Localização da Área. 2- O gerente informa os dados de acordo com os campos a serem preenchidos na tela. 3- O gerente seleciona a opção Cadastrar 4- O sistema emite a mensagem “Área cadastrada com sucesso”. 5- O sistema cadastra a Área no banco de dados.
Cenário Alternativo	O gerente poderá cancelar o processo durante o cadastro.
Casos de Teste	3.1- O sistema verifica se os campos foram preenchidos corretamente. 3.2- O sistema não confirma o cadastro e emite uma Mensagem de erro. 3.3- O sistema cancela a operação e limpa a tela.

Tabela 6 - Manter Localização Área.

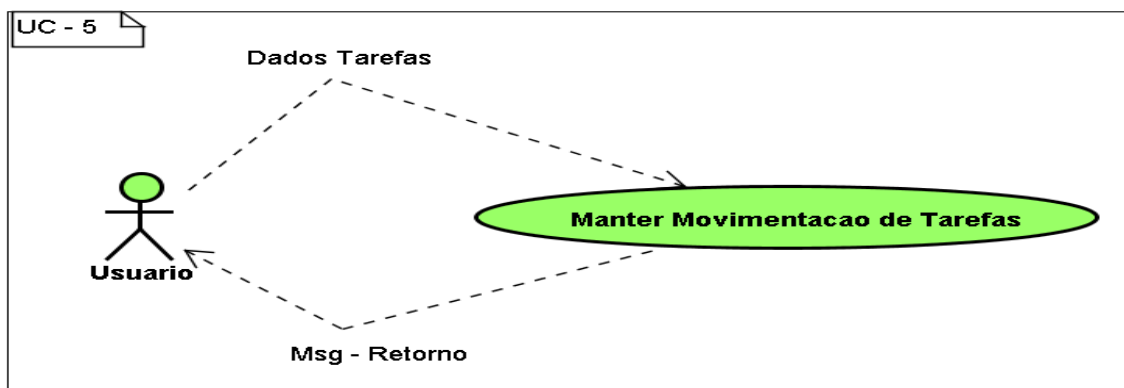


Figura 23 – UC5 – Diagrama de Use Case – Manter Movimentação de Tarefas.

Nome da Use Case 5	Manter Movimentação de Tarefas.
Ator(es)	Usuário
Pré-Condição	O Usuário deverá estar autenticado no sistema.
Cenário Principal	<p>1- O usuário seleciona a opção Consultar:</p> <ul style="list-style-type: none"> • Áreas disponíveis. • Equipamentos disponíveis. <p>2. O sistema faz a busca e verifica as Áreas disponíveis e os Equipamentos disponíveis.</p> <p>3. O sistema solicita os dados necessários para o cadastro de Movimentação de Tarefas.</p> <p>4- O usuário informa os dados de acordo com os campos a serem preenchidos na tela.</p> <p>5- O usuário seleciona a opção Cadastrar</p> <p>6- O sistema emite a mensagem “Movimentação de Tarefas cadastrada com sucesso”.</p> <p>7- O sistema cadastra a Movimentação de Tarefas no banco de dados.</p>
Cenário Alternativo	O usuário poderá cancelar o processo durante o cadastro.
Casos de Teste	<p>5.1- O sistema verifica se os campos foram preenchidos corretamente.</p> <p>5.2- O sistema não confirma o cadastro e emite uma Mensagem de erro.</p> <p>5.3- O sistema cancela a operação e limpa a tela.</p>

Tabela 7 - Manter Movimentação de Tarefas.

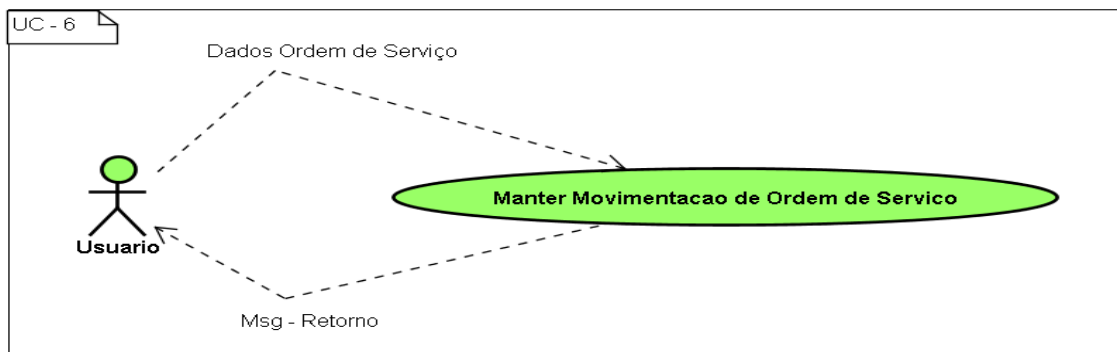


Figura 24 – UC6 – Diagrama de Use Case – Manter Movimentação de Ordem de Serviço.

Nome da Use Case 6	Manter Movimentação de Ordem de Serviço.
Ator(es)	Usuário
Pré-Condição	O Usuário deverá estar autenticado no sistema.
Cenário Principal	<ol style="list-style-type: none"> 1- O usuário seleciona a opção Consultar: <ul style="list-style-type: none"> • Equipamentos disponíveis. 2. O sistema faz a busca e verifica os Equipamentos disponíveis. 3. O sistema solicita os dados necessários para o cadastro de Movimentação de Ordem de Serviço. 4- O usuário informa os dados de acordo com os campos a serem preenchidos na tela. 5- O usuário seleciona a opção Cadastrar 6- O sistema emite a mensagem “Movimentação de Ordem de Serviço cadastrada com sucesso”. 7- O sistema cadastra a Movimentação de Ordem de Serviço no banco de dados.
Cenário Alternativo	O usuário poderá cancelar o processo durante o cadastro.
Casos de Teste	<ol style="list-style-type: none"> 5.1- O sistema verifica se os campos foram preenchidos corretamente. 5.2- O sistema não confirma o cadastro e emite uma Mensagem de erro. 5.3- O sistema cancela a operação e limpa a tela.

Tabela 8 - Manter Movimentação de Ordem de Serviço.

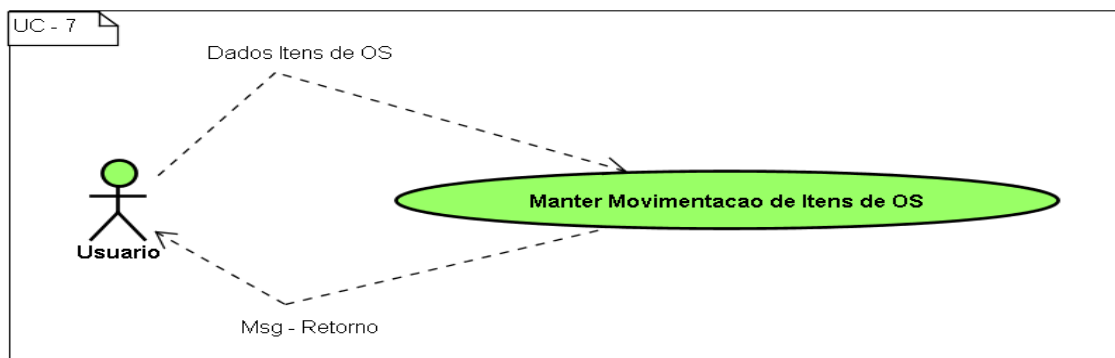


Figura 25 – UC7 – Diagrama de Use Case – Manter Movimentação de Itens de OS.

Nome da Use Case 7	Manter Movimentação de Itens de OS.
Ator(es)	Usuário
Pré-Condição	O Usuário deverá estar autenticado no sistema.
Cenário Principal	<p>1- O usuário seleciona a opção Consultar:</p> <ul style="list-style-type: none"> • Ordens de Serviços disponíveis. • Itens disponíveis. <p>2. O sistema faz a busca e verifica as Ordens de Serviços disponíveis e os Itens disponíveis.</p> <p>3. O sistema solicita os dados necessários para o cadastro de Movimentação de Itens de OS.</p> <p>4- O usuário informa os dados de acordo com os campos a serem preenchidos na tela.</p> <p>5- O usuário seleciona a opção Cadastrar</p> <p>6- O sistema emite a mensagem “Movimentação de Itens de OS cadastrada com sucesso”.</p> <p>7- O sistema cadastra a Movimentação de Itens de OS no banco de dados.</p>
Cenário Alternativo	O usuário poderá cancelar o processo durante o cadastro.
Casos de Teste	<p>5.1- O sistema verifica se os campos foram preenchidos corretamente.</p> <p>5.2- O sistema não confirma o cadastro e emite uma Mensagem de erro.</p> <p>5.3- O sistema cancela a operação e limpa a tela.</p>

Tabela 9 - Manter Movimentação de Itens da Ordem de Serviço.

3.9 – DIAGRAMA DE CLASSES.

O desenvolvimento do protótipo do Sistema foi desenvolvido em camadas que foram definidas da seguinte maneira: Modelos que conterá os atributos dos campos de cada tabela. A camada DAL (Data Access Layer) conterá os métodos de manipulação das tabelas no banco de dados e a camada BLL (Business Logic Layer) onde será implementado todas as regras de negócios para o aplicativo.

A figura 26 mostra o diagrama de classe do pacote Modelo VO.

- **Modelos – VO (Value Objects).**
 - ❖ **UsuarioVO:** classe responsável por instanciar os usuários do sistema.
 - ❖ **LocalizaçãoÁreaVO:** classe responsável por instanciar a localização das áreas do sistema.
 - ❖ **ItemVO:** classe responsável por instanciar os itens do sistema.
 - ❖ **ItensdaOSVO:** classe responsável por instanciar os itens das OS do sistema.
 - ❖ **TarefasRealizadasVO:** classe responsável por instanciar as tarefas do sistema.
 - ❖ **EquipamentoVO:** classe responsável por instanciar os equipamentos do sistema.
 - ❖ **OrdemdeServicoVO:** classe responsável por instanciar as ordens de serviço do sistema.

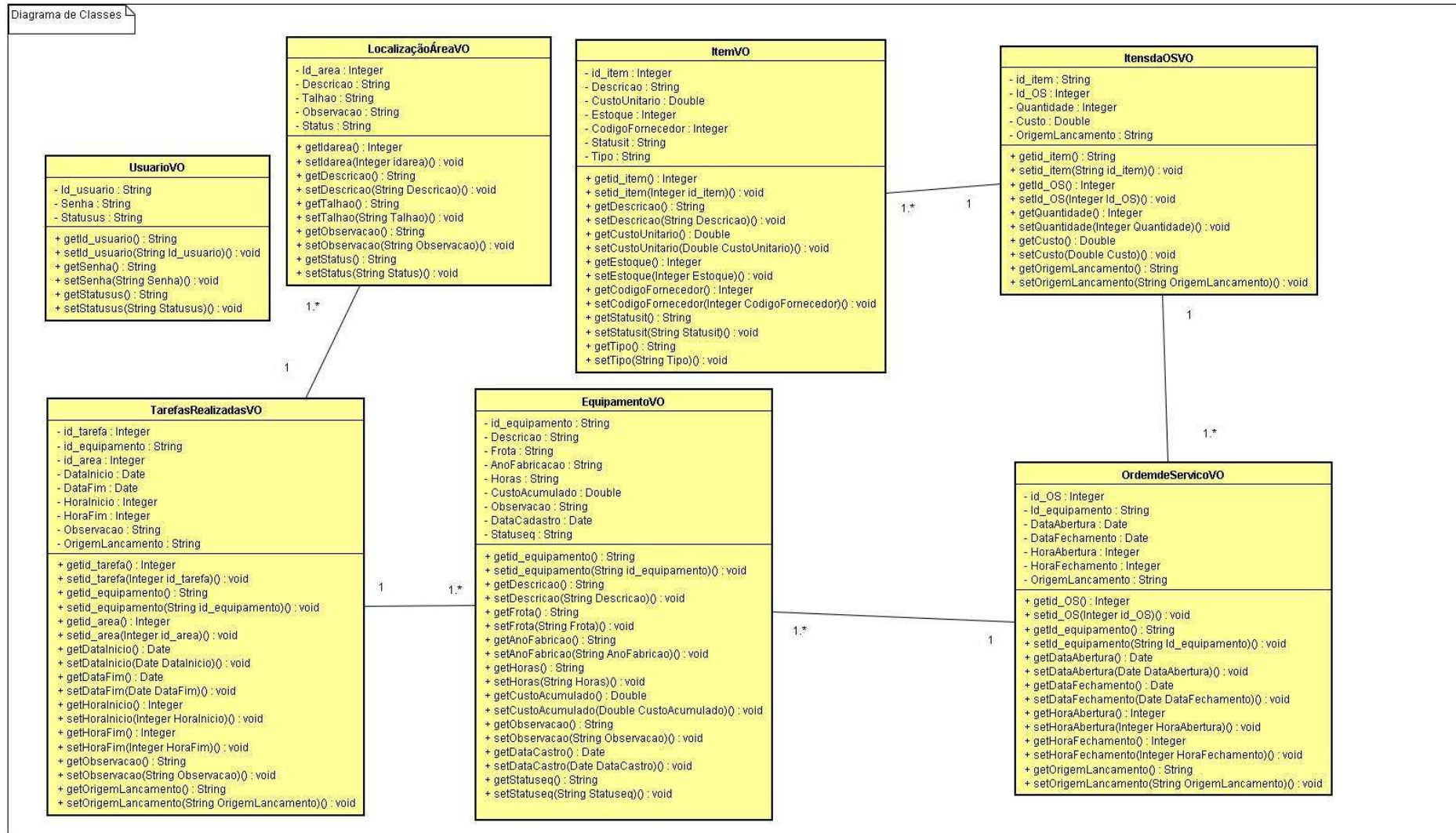


Figura 26 – Diagrama de Classes do Pacote Modelo – VO (Value Objects).

A figura 27 mostra o diagrama de classe do pacote DAL.

- **DAL**

- ❖ **UsuarioDAL**: classe responsável por armazenar os usuários no banco.
- ❖ **LocalizaçãoÁreaDAL**: classe responsável por armazenar a localização das áreas no banco.
- ❖ **ItemDAL**: classe responsável por armazenar os itens no banco.
- ❖ **ItensdaOSDAL**: classe responsável por armazenar os itens das OS no banco.
- ❖ **TarefasRealizadasDAL**: classe responsável por armazenar as tarefas realizadas no banco.
- ❖ **EquipamentoDAL**: classe responsável por armazenar os equipamentos no banco.
- ❖ **OrdemdeServicoDAL**: classe responsável por armazenar as ordens de serviço no banco.
- ❖ **Conexão**: classe genérica de acesso entre o banco de dados e a cama DAL.

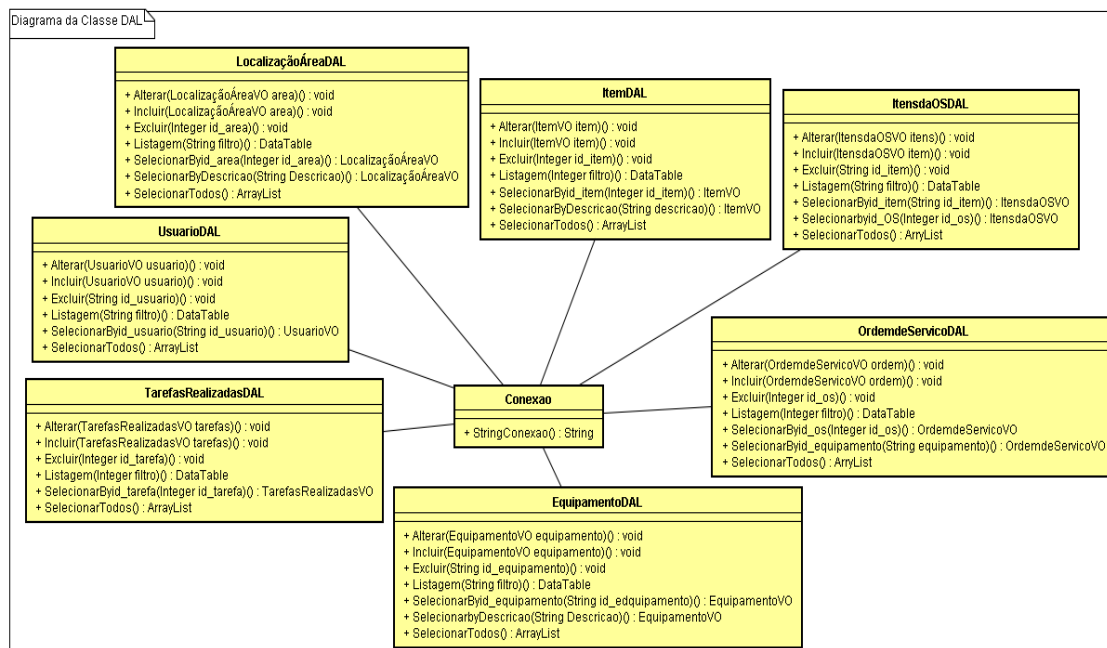


Figura 27 – Diagrama de Classes do Pacote DAL.

OBS: A visualização desta imagem encontra-se no Anexo, pag. 115.

A figura 28 mostra o diagrama de classe do pacote BLL.

- **BLL**

- ❖ **UsuarioBLL**: classe responsável pelas regras de negocio para os usuários.
- ❖ **LocalizaçãoÁreaBLL**: classe responsável pelas regras de negocio da localização das áreas.
- ❖ **ItemBLL**: classe responsável pelas regras de negocio dos itens.
- ❖ **ItensdaOSBLL**: classe responsável pelas regras de negocio dos itens das OS.
- ❖ **TarefasRealizadasBLL**: classe responsável pelas regras de negocio das tarefas realizadas.
- ❖ **EquipamentoBLL**: classe responsável pelas regras de negocio dos equipamentos.
- ❖ **OrdemdeServicoBLL**: classe responsável pelas regras de negocio das ordens de serviço.

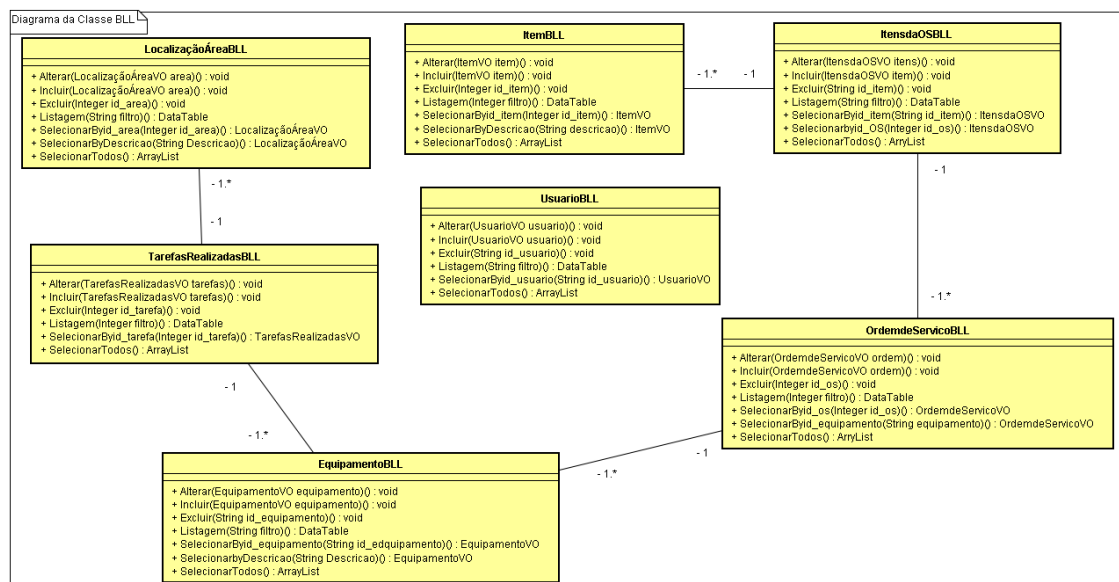


Figura 28 – Diagrama de Classes do Pacote BLL.

OBS: A visualização desta imagem encontra-se no Anexo, pag. 116.

3.10 – DIAGRAMA DE ATIVIDADES.

O diagrama de atividades representa o processo, ou fluxo das informações entre as varias atividades existentes no sistema. Na figura 29, é apresentado o diagrama de atividades do aplicativo desktop e do dispositivo móvel *Tablet*. O diagrama mostra todos os processos que um usuário pode realizar nos dois protótipos que foram desenvolvidos.

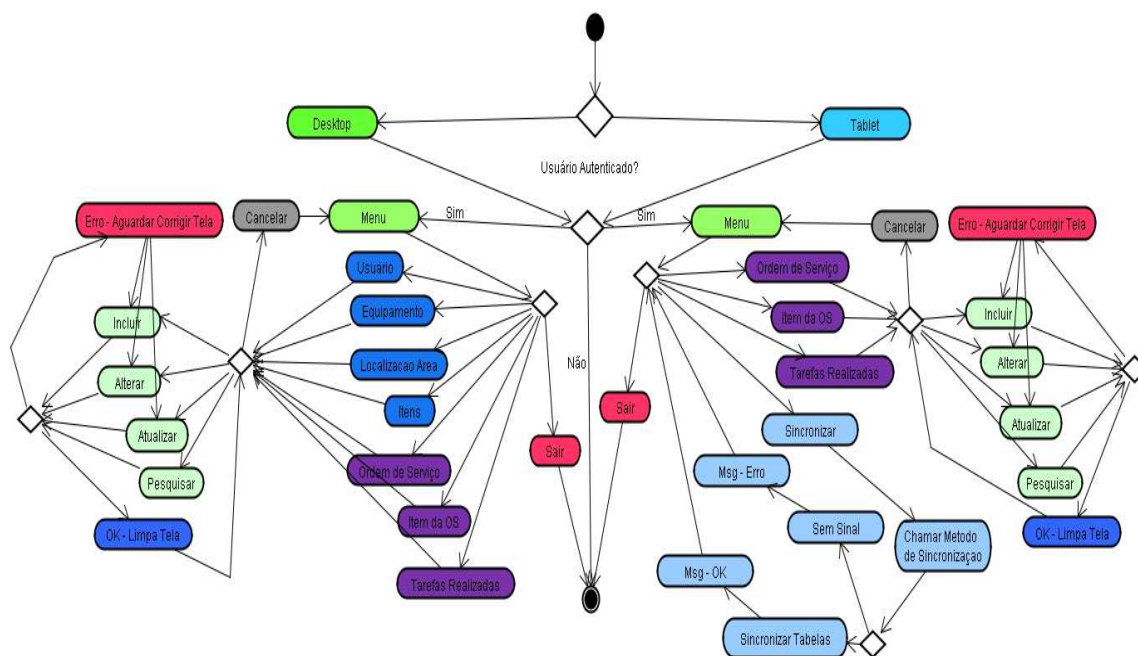


Figura 29 – Diagrama de Atividades de todo o processo dos protótipos.

OBS: A visualização desta imagem encontra-se no Anexo, pag. 117.

3.11 – DIAGRAMA DE SEQÜÊNCIA.

O diagrama de seqüência representa a seqüência das ações ocorridas em um conjunto de classes, demonstrando como ocorre a troca de mensagens entre elas. Será utilizado um diagrama para cada caso de uso especificado: manter ordem de serviço, manter itens da ordem de serviço e manter tarefas realizadas.

3.11.1 – Manter Ordem de Serviço.

A figura 30 mostra o diagrama de seqüência com as ações que podem ser executadas neste caso de uso: inserir, alterar, excluir e pesquisar ordem de serviço.

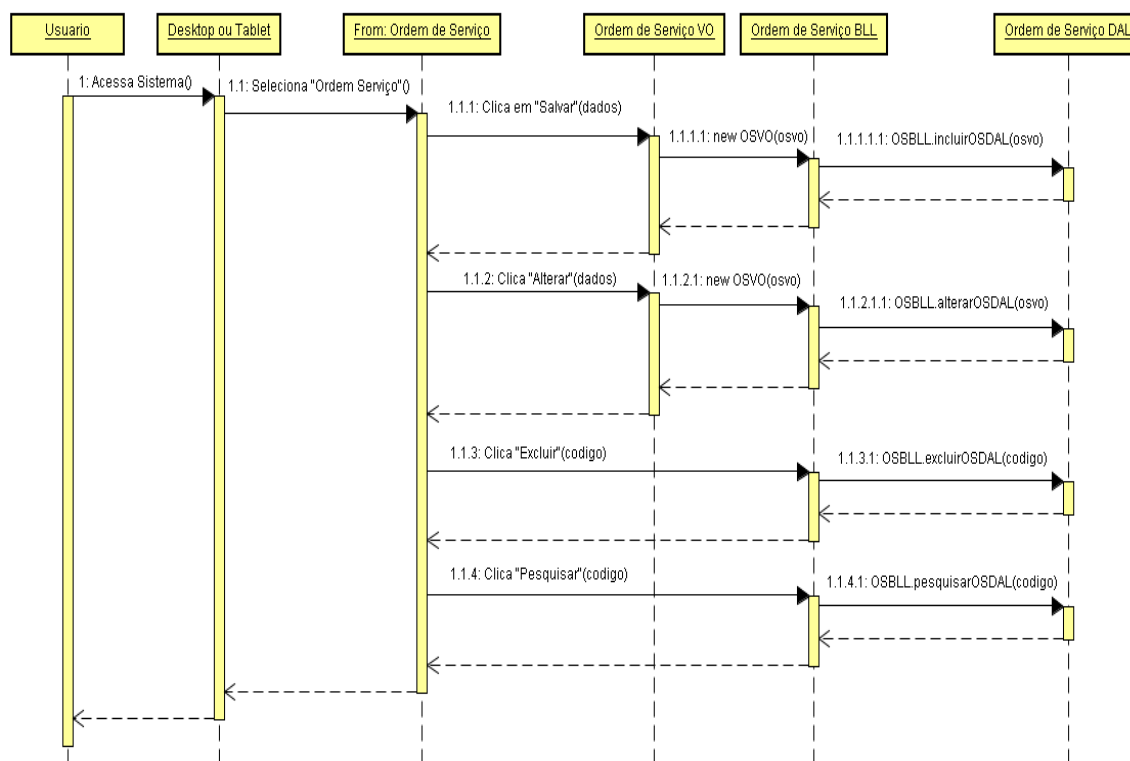


Figura 30 – Diagrama de seqüência – manter ordem de serviço.

3.11.2 – Manter Itens da Ordem de Serviço.

A figura 31 mostra o diagrama de seqüência com as ações que podem ser executadas neste caso de uso: inserir, alterar, excluir e pesquisar itens da ordem de serviço.

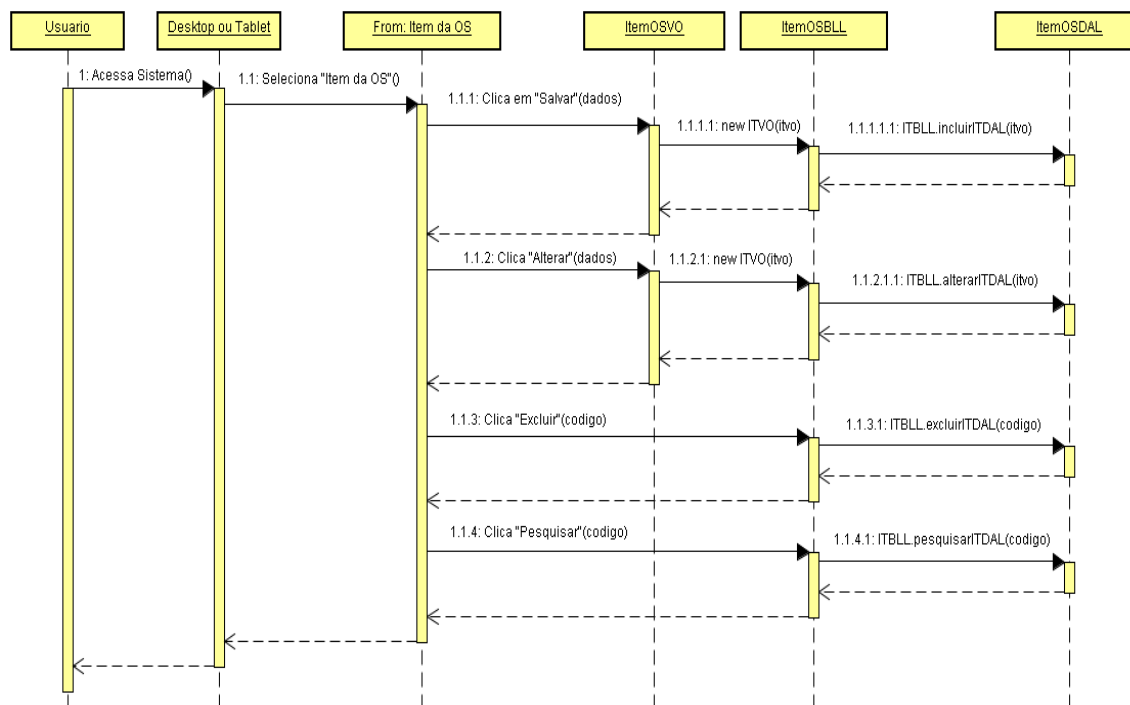


Figura 31 – Diagrama de seqüência – manter itens da ordem de serviço.

3.11.3 – Manter Tarefas Realizadas.

A figura 32 mostra o diagrama de seqüência com as ações que podem ser executadas neste caso de uso: inserir, alterar, excluir e pesquisar tarefas realizadas.

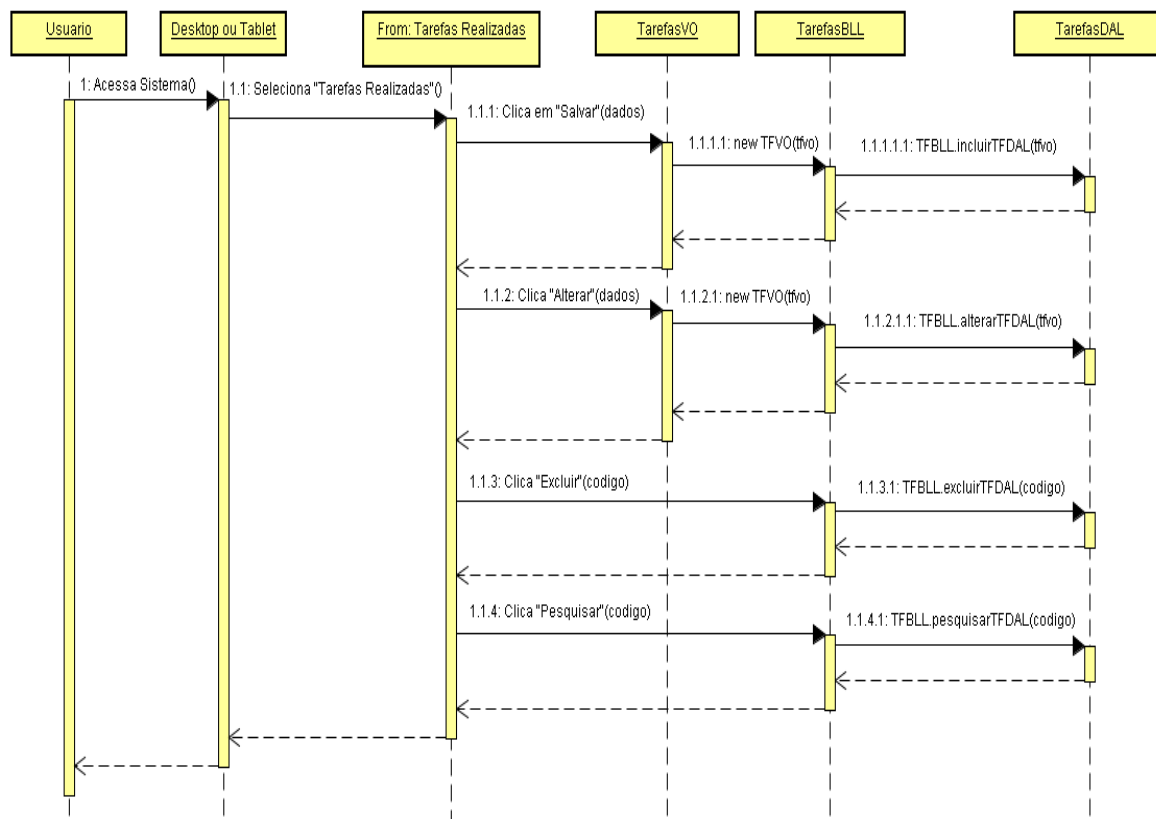


Figura 32 – Diagrama de seqüência – manter tarefas realizadas.

3.12 – MÓDULO DESKTOP – (BACK-END).

O módulo Desktop ou retaguarda, mais conhecido como *Back-end*, que será desenvolvido em C#, tem por objetivo, fazer todo o papel das funções disponíveis do sistema.

Através do *Back-end*, o usuário poderá fazer todos os cadastros, movimentações e emitir os relatórios do sistema.

O módulo Desktop é o responsável por toda a movimentação do sistema, ou seja, é o coração do sistema de gestão da empresa. É considerado o módulo principal dos processos de informatização do setor agrícola da empresa.

As tecnologias utilizadas para no desenvolvimento do módulo Desktop, são as seguintes:

- ❖ **Plataforma:** Microsoft .Net
- ❖ **Linguagem de Programação:** C#
- ❖ **Ferramenta Desenvolvimento:** Microsoft Visual Studio 2010.

Com o objetivo de melhorar os processos de informatização da empresa, e auxiliar o módulo *Back-end*, será desenvolvido um módulo *Front-end* que será implementado no dispositivo móvel conhecido como *Tablet* que será apresentado neste trabalho.

3.12.1 - Casos de Usos do Módulo Desktop.

Na figura 33, vamos apresentar quais serão os casos de usos que compõem o desenvolvimento do protótipo do modulo desktop.

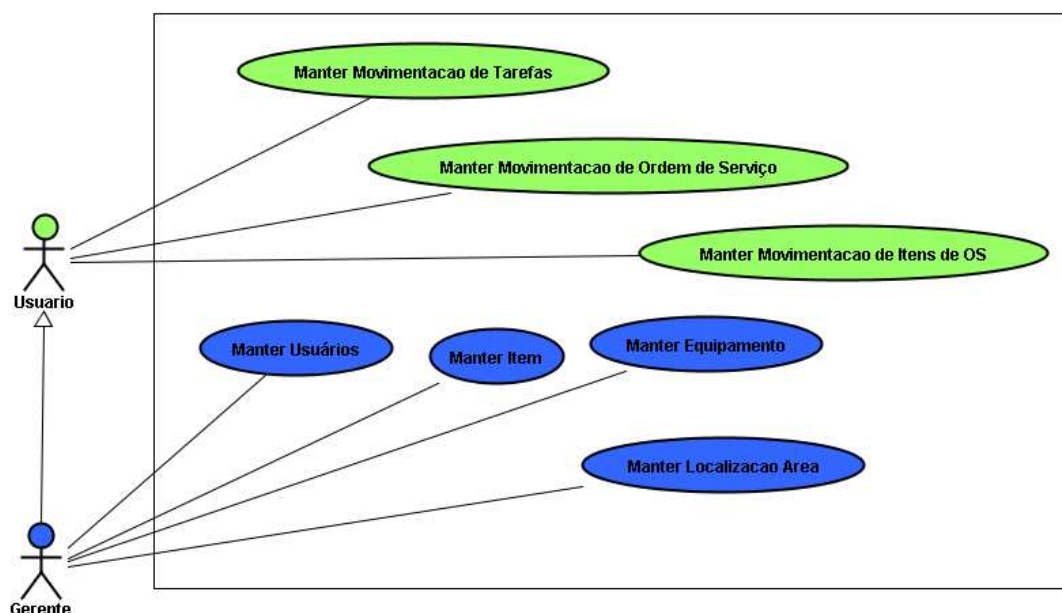


Figura 33 – Diagrama de casos de usos do módulo desktop.

No desenvolvimento do módulo Desktop, serão utilizados os setes casos de usos apresentados na figura acima, ou seja, todos eles contemplarão a implementação do CRUD (Create, Read, Update and Delete).

Com relação ao detalhamento dos casos de usos do módulo Desktop, foram apresentados dos tópicos 3.6 até 3.11 deste trabalho.

3.13 – MÓDULO MÓBILE - (FRONT-END).

O módulo móvel, ou aplicação mais conhecida como *Front-end* tem por objetivo, auxiliar no campo ou nas fazendas, o acesso aos usuários para realizarem os apontamentos das suas atividades diárias envolvidas com a utilização dos equipamentos agrícolas.

A vantagem da utilização do dispositivo móvel (*Tablet*) no campo, é devido ao recurso disponível pelo *Android*, um banco de dados interno conhecido como SQLite que permite o armazenamento das informações igual a um banco de dados comum. Para o lançamento das atividades realizadas no campo o *Tablet*, não precisará de conexão com a Internet, pois afinal tudo será armazenado no seu banco de dados para depois sincronizar via Web Service com o banco de dados do servidor da empresa.

O módulo móvel é uma forma de auxiliar no desenvolvimento dos processos, pois afinal todo o lançamento era realizado somente no Desktop. Sendo assim, este processo era inviável para a empresa, por que estava gerando um atraso nos lançamentos das atividades ocorridas no campo com os equipamentos agrícolas. Com o uso do *Tablet*, fica mais rápido os lançamentos, pois são realizados durante o dia pelos próprios líderes de campo, tratorista e mecânicos e depois com o uso da Internet é só realizar a sincronização dos lançamentos e tudo estará no servidor da empresa disponível para a análise das gerencias.

As tecnologias utilizadas para no desenvolvimento do módulo móvel, são as seguintes:

- ❖ **Plataforma:** Linux – Google *Android*.
- ❖ **Linguagem de Programação:** Java e XML.
- ❖ **Ferramenta Desenvolvimento:** Eclipse e o Emulador SDK.
- ❖ **Dispositivo Móvel:** *Tablet* com S.O *Android*.

3.13.1 - Casos de Usos do Módulo Móvel.

Na figura 34, vamos apresentar quais serão os casos de usos que compõem o desenvolvimento do protótipo do módulo móvel.

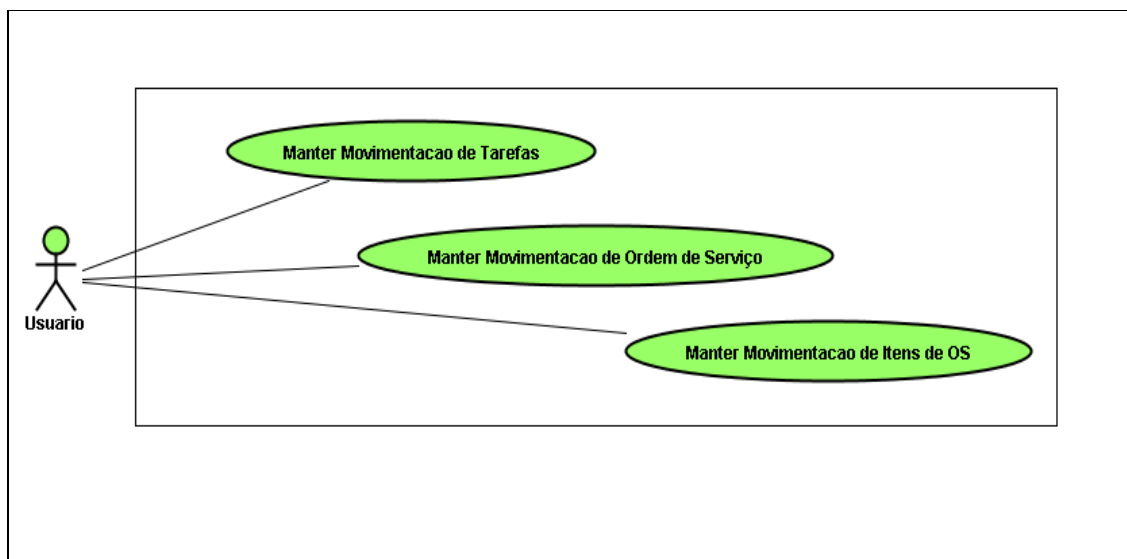


Figura 34 – Diagrama de casos de usos do módulo móvel (*Tablet*).

No desenvolvimento do módulo *Front-end*, serão utilizados os três casos de usos apresentados na figura acima, ou seja, todos eles contemplarão a implementação do CRUD (Create, Read, Update and Delete).

Com relação ao detalhamento dos casos de usos do módulo móvel, foram apresentados dos tópicos 3.6 até 3.11 deste trabalho.

3.14 – MÓDULO DE INTEGRAÇÃO (WEB SERVICE).

O módulo de integração tem por objetivo, fazer a comunicação entre os dois bancos de dados, que serão utilizados entre as duas aplicações.

Na aplicação *Back-end* (Desktop) será utilizado o banco de dados mais conhecido como o MySQL.

Na aplicação *Front-end* que utilizaremos um dispositivo móvel conhecido como *Tablet*, será utilizado o banco de dados nativo do *Android* que é o SQLite.

Será desenvolvida uma aplicação, conhecida como Web Service, que será o responsável pelo processo de integração das comunicações ou troca de informações entre os dois bancos de dados apresentados. O Web Service que será desenvolvido nesta pesquisa é o modelo REST (Transferência de Estado Representacional) que é um padrão de comunicação de serviços web, ao qual utiliza as seguintes formas de “encapsular” as informações (XML, JSON, YAML, etc) que realiza a transferência das informações através do protocolo HTTP.

A tecnologia utilizada para o desenvolvimento do Web Service responsável pela troca de informações entre os bancos de dados, serão as seguintes:

- ❖ **Plataforma de Desenvolvimento:** Java
- ❖ **Modelo Web Service:** REST com o uso do JSON.
- ❖ **Protocolo de comunicação:** HTTP.
- ❖ **Servidor Web:** Apache Tomcat/7.032
- ❖ **Ferramenta Desenvolvimento:** Eclipse.

Na figura 35, mostra o exemplo do processo de comunicação dos dois protótipos desenvolvidos em C# e *Android* comunicando-se via Web Service.

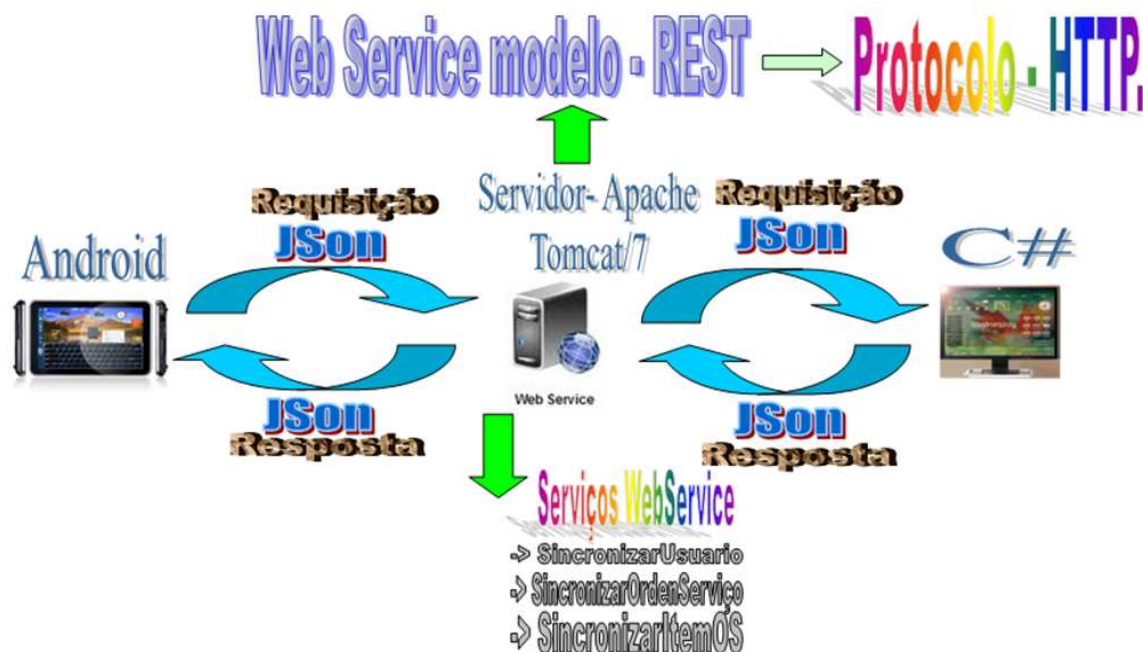


Figura 35 – Processo de comunicação Web Service entre C# e Android.

3.14.1 – Processo de mapeamento das tabelas dos bancos de dados.

Na figura 36, mostra quais serão as tabelas do banco de dados SQLite do *Android* que serão sincronizadas para o banco de dados MySQL do servidor, ou seja, as três tabelas destacadas em vermelho estarão ajudando a alimentar as informações no banco de dados principal que está instalado no servidor da empresa ao qual é o MySQL.

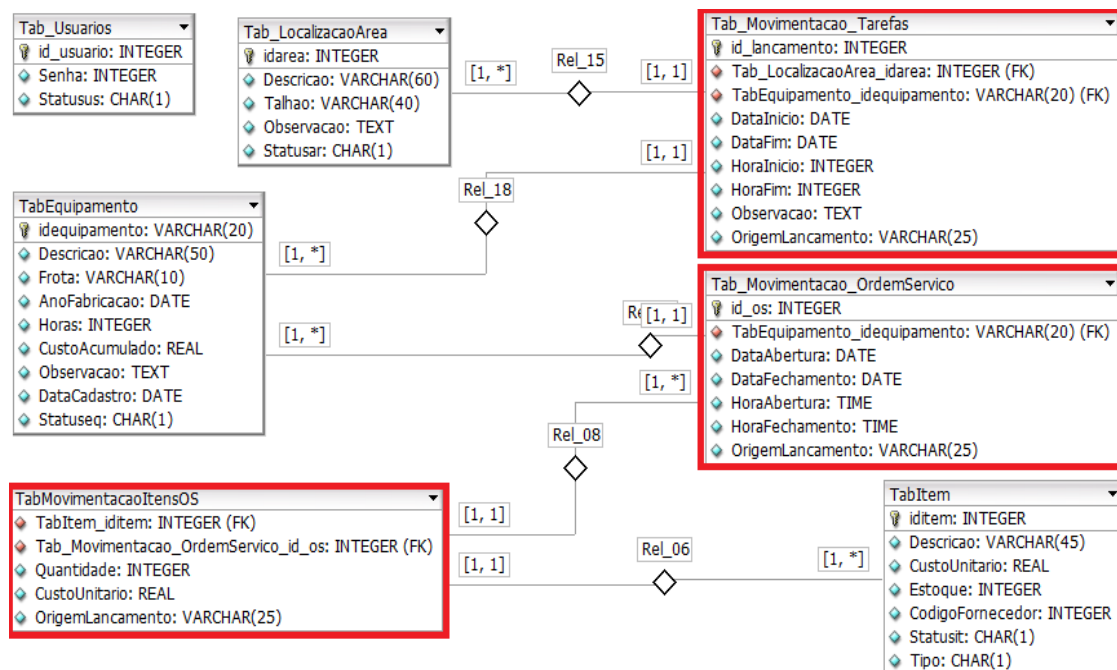


Figura 36 – Sincronização das tabelas do SQLite para o MySQL.

Na figura 37, mostra quais serão as tabelas do banco de dados MySQL do servidor que serão sincronizadas para o banco de dados SQLite do *Android*, ou seja, as quatro tabelas destacadas em vermelho serão as tabelas de bases de informações disponíveis no *Tablet* para os usuários realizarem os lançamentos de suas tarefas, sendo assim, estas tabelas destacadas em vermelho só receberão informações das tabelas do banco de dados MySQL com o uso do Web Service.

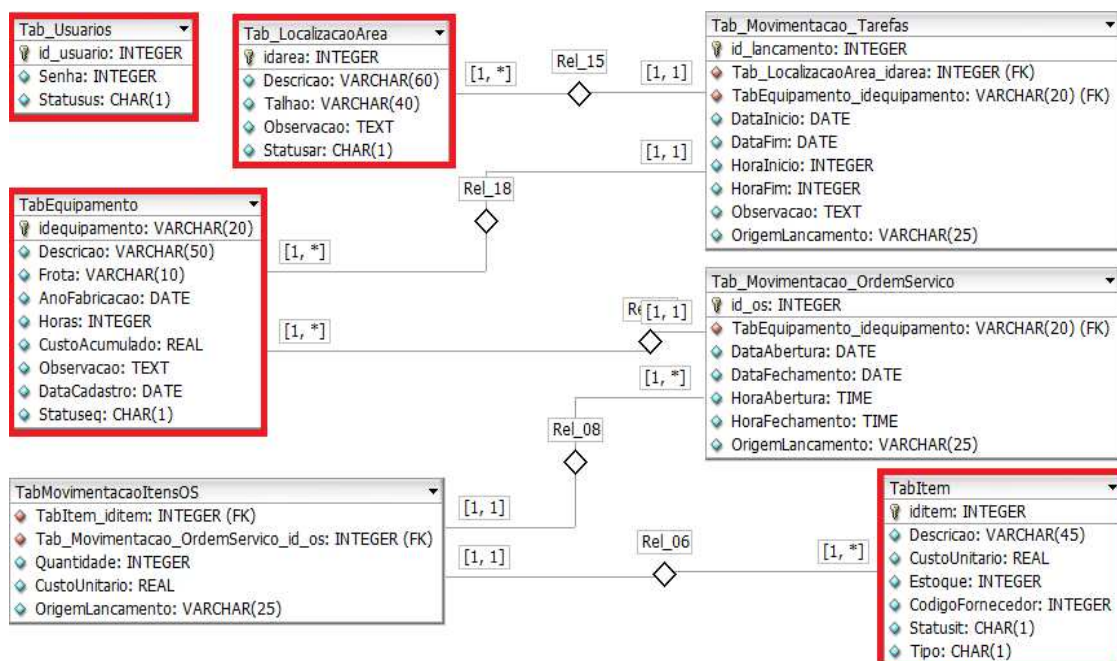


Figura 37 – Sincronização das tabelas do MySQL para o SQLite.

4 – IMPLEMENTAÇÃO DO SISTEMA.

Para a implementação do aplicativo Desktop foi utilizado o ambiente de programação Visual Studio .NET 2010.

Para a implementação do aplicativo móvel (*Tablet*) foi utilizado o ambiente de programação Eclipse INDIGO versão 3.7.2 com o *Android* Virtual Device Manager (AVD), Target: *Android* 3.0 (API level 11), SKIN: WXGA.

4.1 APLICATIVO DESKTOP (C#).

Neste tópico vamos apresentar os processos de implementação do módulo Desktop.

Na figura 38, apresenta a interface inicial do aplicativo para o desktop, sendo obrigatório informar um nome de usuário e uma senha para acessar as funcionalidades do sistema.



Figura 38 – Tela de autenticação do Login e Senha do Desktop.

Na figura 39, apresenta a tela de menu do aplicativo desktop, onde o usuário poderá selecionar as opções disponíveis no sistema.

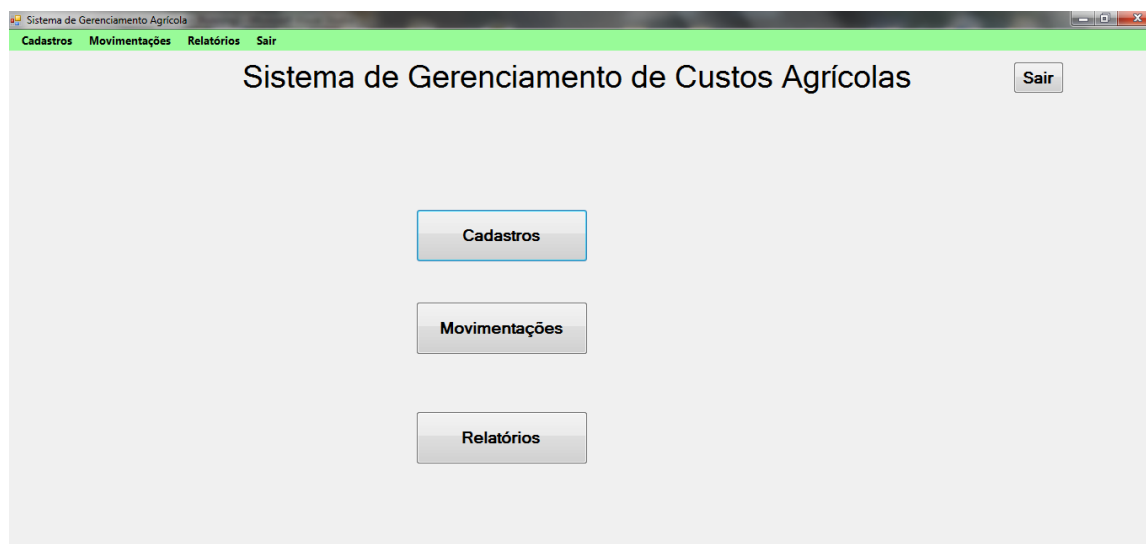


Figura 39 – Tela do menu de opções do módulo desktop.

Na figura 40, apresenta a tela de cadastros do aplicativo desktop, onde o usuário poderá selecionar as opções disponíveis no sistema.

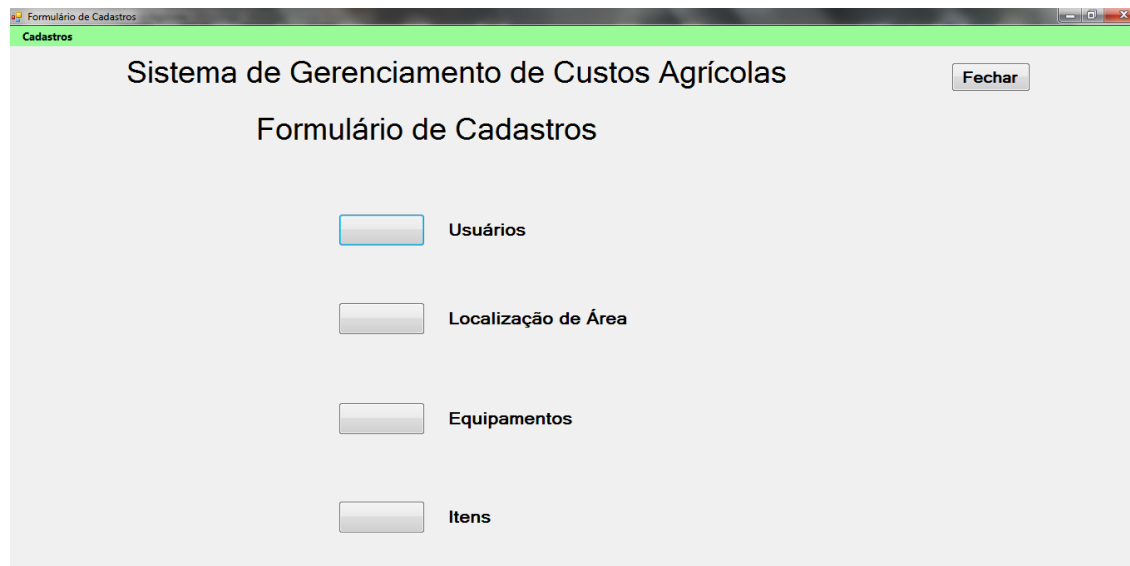


Figura 40 – Tela de cadastros de opções do módulo desktop.

Na figura 41, apresenta a interface da tela de cadastro de usuários do sistema, com as funções de Incluir, Excluir, Alterar e Pesquisar.

Sistema de Gerenciamento de Custos Agrícolas Fechar

Pesquisar.:

ID Usuário:

Senha:

Status: Ativo

	id_usuario	Senha	Status
▶	Carlos	1	Ativo
	Celso	12	Ativo
	Jose	456	Ativo
*			

Figura 41 – Interface da tela de cadastros dos usuários do sistema.

4.1.1 Organização dos pacotes e classes da aplicação Desktop.

Para uma melhor organização, o sistema foi organizado em pacotes. A figura 42 apresenta os pacotes: BLL, DAL, Model, Conexão, Visão.

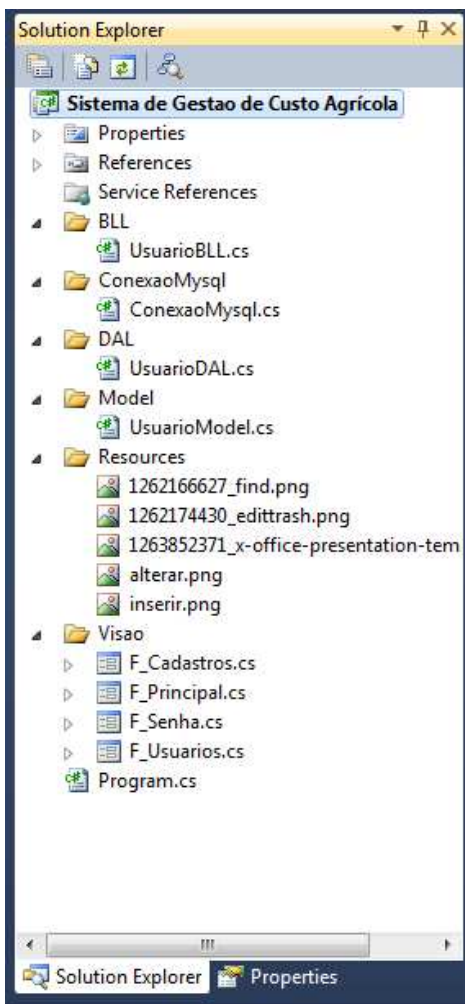


Figura 42 – Organização em pacotes.

Pacote BLL: É nele que ficam as classes de modelagem do projeto, são as classes que fazem a persistência dos dados com o banco de dados.

Pacote ConexãoMySQL: É nele que fica a classe que guarda o usuário e senha do banco.

Pacote DAL: É nele que fica os métodos que realizam o acesso ao banco de dados, como Gravar, Alterar, Excluir, Pesquisar e Listar.

Pacote Model: É nele que ficam todas as classes utilizadas no sistema, aos quais as classes contem os atributos e os métodos de acessos a elas.

Pacote Resources: É nele que ficam todas as imagens do sistema.

Pacote Visão: É nele que ficam todas as telas de interface do sistema.

4.2 APLICATIVO MÓBILE (TABLET).

Neste tópico vamos apresentar os processos de implementação do módulo móvel (*Tablet*).

Na figura 43, apresenta a interface inicial do aplicativo móvel (*Tablet*), sendo obrigatório informar um nome de usuário e uma senha para acessar as funcionalidades do sistema.

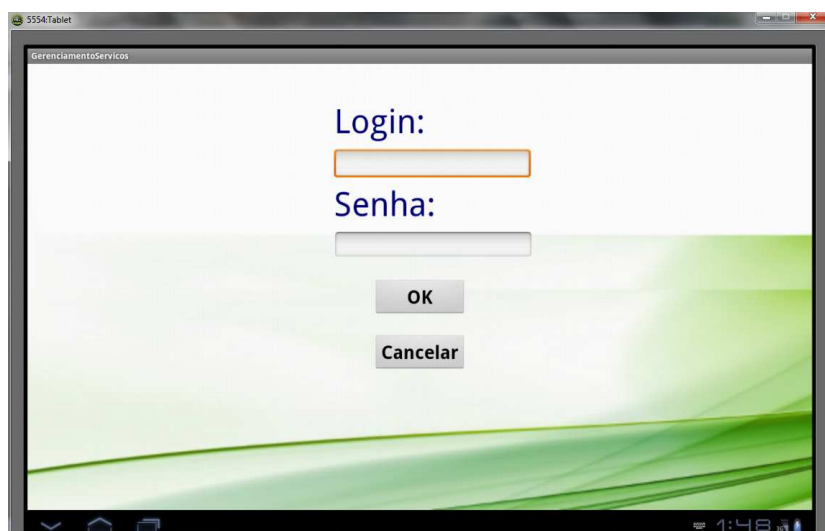


Figura 43 – Tela de autenticação do Login e Senha do *Tablet*.

Nas figuras 44 e 45, apresentam as telas de menu do aplicativo móvel (*Tablet*), onde o usuário poderá selecionar as opções disponíveis no sistema.



Figura 44 – Tela do menu do aplicativo móvel (*Tablet*).



Figura 45 – Tela das opções do menu do *Tablet*.

Na figura 46, apresenta a interface da tela de cadastro de ordens de serviços do sistema, com as funções de Incluir, Excluir, Alterar e Pesquisar.

5554:Tablet

GerenciamentoServicos

Controle de Ordens de Serviços

Ordem de Serviço.: **PesquisarOS**

Equipamento.: **E**

Data Abertura.:

Data Fechamento.:

Hora Abertura.:

Hora Fechamento.: OrigemLancamento.:

Lancar Pecas.:

Salvar **Pesquisar** **Editar** **Excluir**

Usuario: q **Sair**

2:15

Figura 46 – Tela de cadastro das ordens de serviços no *Tablet*.

Na figura 47, apresenta a interface da tela de cadastro de itens da ordem de serviço do sistema, com as funções de Incluir, Excluir, Alterar e Pesquisar.

5554:Tablet

GerenciamentoServicos

Lançamento de Itens da OS.:

IDlançamento.: Ordem de Serviço.: **Pesquisar OS**

Codigo do Item.: **Pesquisar Item**

Descrição..:

Quantidade.:

Custo Unitário..:

Custo Total.:

OrigemLancamento.: q

Salvar **Pesquisar** **Editar** **Excluir**

Usuario: q **Sair**

2:20

Figura 47 – Tela de lançamento dos itens da Ordem de Serviço (*Tablet*).

Na figura 48, apresenta a interface da tela de controle de tarefas dos equipamentos, com as funções de Incluir, Excluir, Alterar e Pesquisar.

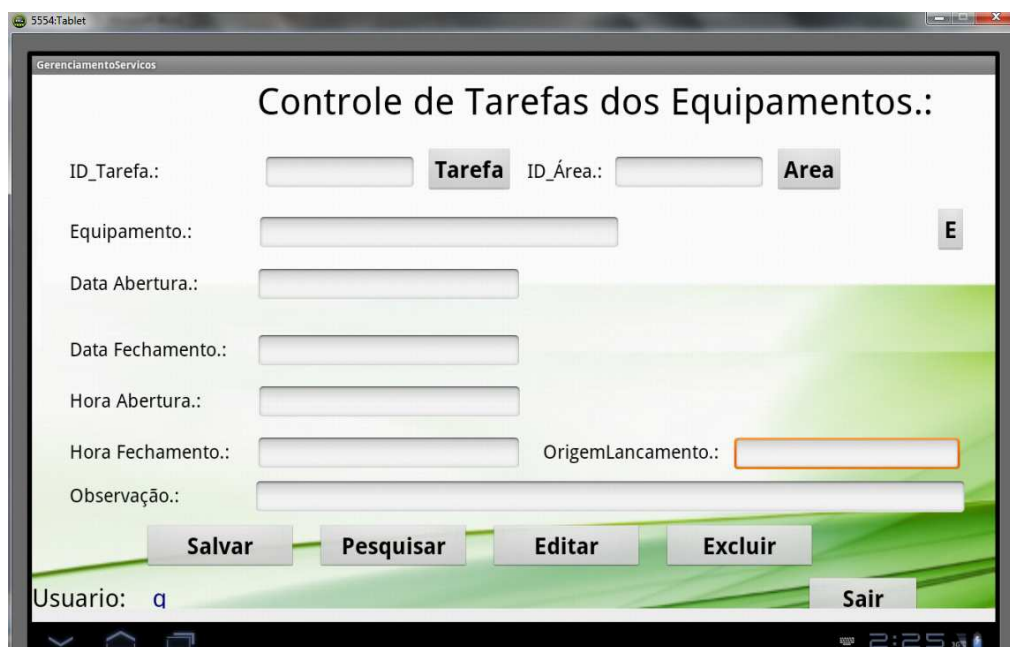


Figura 48 – Tela de controle de tarefas dos equipamentos agrícolas (*Tablet*).

Na figura 49, apresenta a interface da tela de sincronização das tabelas do banco de dados SQLite com o banco de dados MySQL.

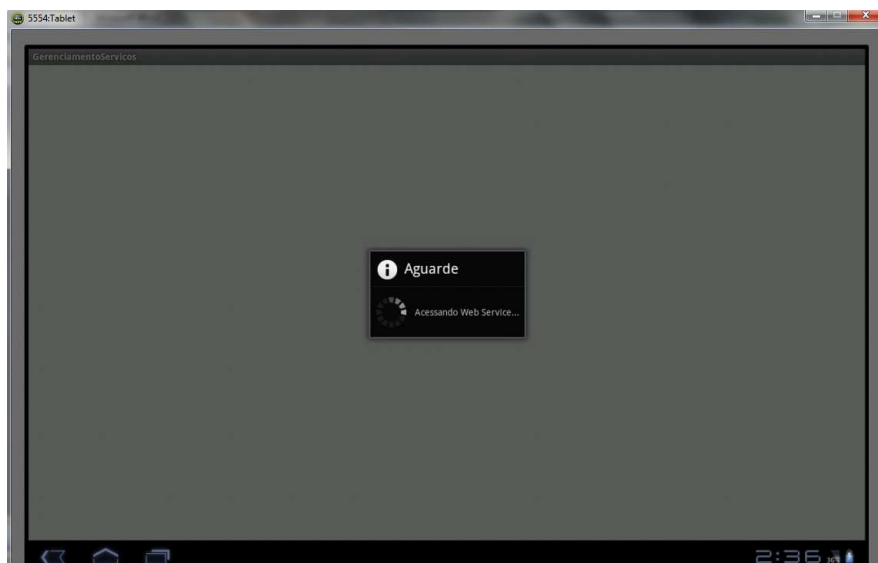


Figura 49 – Tela de sincronização do *Tablet*.

4.2.1 Organização dos pacotes e classes da aplicação *Tablet*.

Para uma melhor organização, o sistema foi organizado em pacotes. A figura 50 apresenta os pacotes: BLL, DAL, DB, Model.

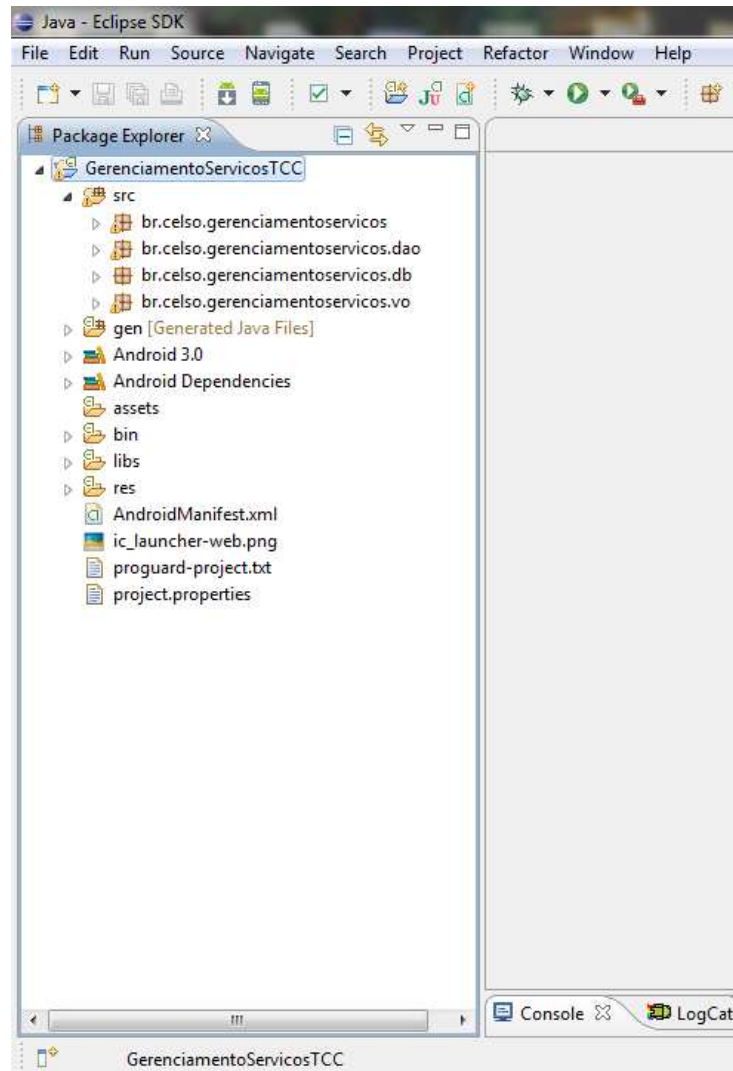


Figura 50 – Mostra a organização em pacotes do *Tablet*.

Pacote “br.celso.gerenciamentoservicos”: É nele que ficam as classes BLL, ou seja, as classes que fazem a regra de negocio e realizam a persistência dos dados com o banco de dados.

Pacote “br.celso.gerenciamentoservicos.dao”: É nele que fica os métodos que realizam o acesso ao banco de dados, como Gravar, Alterar, Excluir, Pesquisar e Listar.

Pacote “br.celso.gerenciamentoservicos.db”: É nele que ficam as classes de interface que são responsável pela criação das tabelas no banco de dados, ou seja, neste pacote contém todas as classes com o seus atributos e o método de create table para geração de cada tabela do banco de dados.

Pacote “br.celso.gerenciamentoservicos.vo”: É nele que ficam todas as classes utilizadas no sistema, ao quais as classes contém os atributos e os métodos de acessos a elas.

Para uma melhor organização, o sistema foi organizado em pacotes. A figura 51 apresenta os pacotes: Drawable-hdpi e Layout.

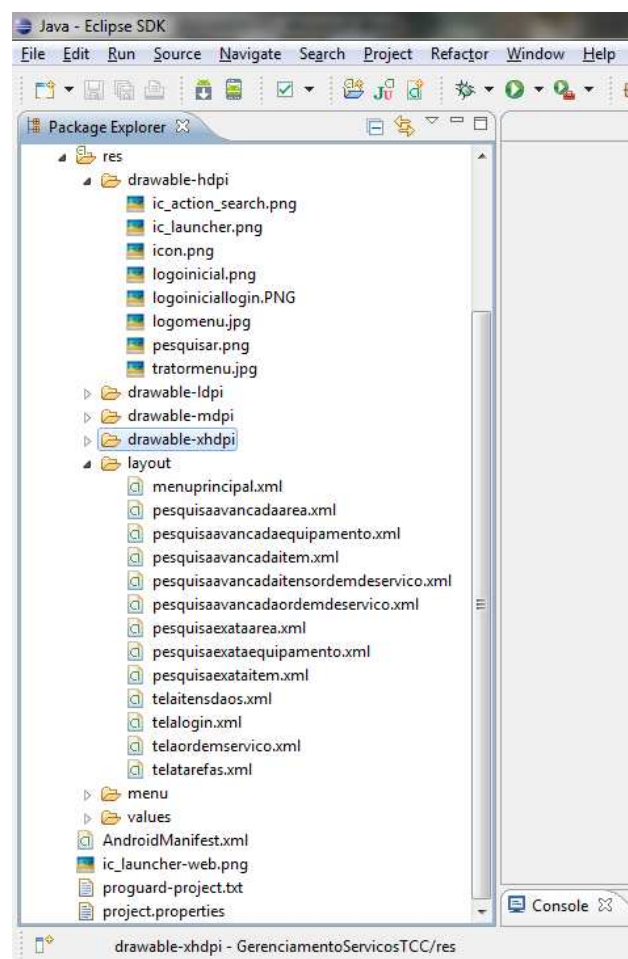


Figura 51 – Organização de pacotes – imagens e telas do *Tablet*.

Pacote Drawable-hdpi: É nele que ficam todas as imagens do sistema.

Pacote Layout: É nele que ficam todas as telas de interface do sistema.

4.3 DESENVOLVIMENTO DO WEB SERVICE.

Neste tópico vamos apresentar os processos de desenvolvimento do módulo Web Service.

4.3.1 Organização dos pacotes do Web Service em Java.

Para uma melhor organização o Web Service em Java foi organizado em pacotes.

A figura 52 apresenta os pacotes: SGA.Conexão, SGA.DAO, SGA.Resources e SGA.VO.

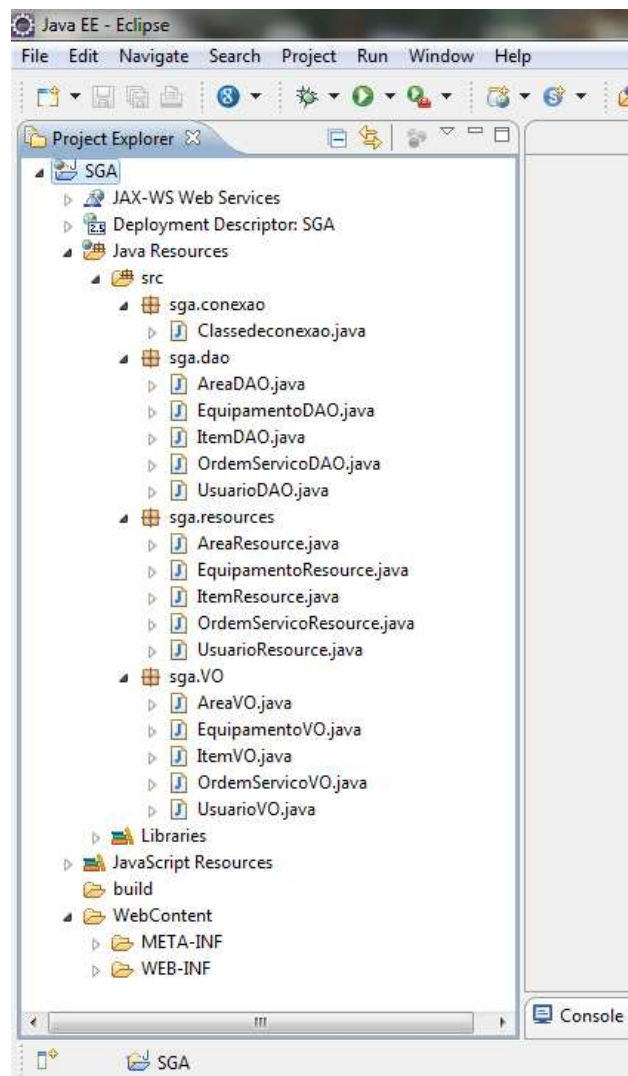


Figura 52 - Organização de pacotes – Web Service em Java.

Pacote “SGA.CONEXAO”: É nele que fica a classe responsável por realizar a conexão com o banco de dados MySQL. Nesta classe que fica os métodos de conexão e desconexão e execução da SQL de acesso aos dados no banco de dados.

Abaixo mostra o código fonte da classe responsável por realizar a conexão com o banco de dados.

```
package sga.conexao;
import java.sql.*;
public class Classedeconexao {
String driver = "com.mysql.jdbc.Driver";
String url = "jdbc:mysql://localhost/servicos";
String usuario = "root";
String senha = "informarsenha";
    private Connection conexao;
    public Statement statement;
    public ResultSet resultset;

    public boolean conecta()
    {
        boolean result = true;
        try {
            Class.forName(driver);
            conexao = DriverManager.getConnection(url, usuario, senha);
            System.out.println("conectou legal com o MySQL");

        } catch (ClassNotFoundException Driver) {
            System.out.println("Driver não localizado: " + Driver);
            result = false;
        } catch (SQLException Fonte) {
            System.out.println("Deu erro na conexão " +
                "com a fonte de dados: " + Fonte);
            result = false;
        }
        return result;
    }

    public void desconecta() {
        try {
            conexao.close();
            System.out.println("banco fechado");
        } catch (SQLException fecha) {
            System.out.println("Não foi possível " +
                "fechar o banco de dados: " + fecha);
        }
    }
}
```

```

    public void executeSQL(String sql) {
        try {
            statement = conexao.createStatement(
                ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
            resultSet = statement.executeQuery(sql);
        } catch (SQLException sqlex) {
            System.out.println("Não foi possível " +
                "executar o comando sql," + sqlex + ", o sql passado foi " +
sql);
        }
    }

    public void InsertSQL(String sql) {
        try {
            statement = conexao.createStatement();
            statement.executeUpdate(sql);
        } catch (SQLException sqlex) {
            System.out.println("Não foi possível " +
                "executar o comando sql," + sqlex + ", o sql passado foi " +
sql);
        }
    }
}

```

Pacote “SGA.DAO”: É nele que fica as classes DAO responsável por realizar os métodos de manipulação dos objetos no banco de dados. Para cada classe de Objetos, precisamos de uma classe DAO para realizar a manipulação dos dados no banco de dados.

Abaixo mostra o código fonte da classe “**OrdemServicoDAO**” responsável por realizar a manipulação dos dados da Ordem de Serviço no banco de dados.

```

package sga.dao;
import java.sql.SQLException;
import java.lang.String;
import java.util.ArrayList;
import java.util.List;
import sga.conexao.Classedeconexao;
import sga.VO.OrdemServicoVO;

public class OrdemServicoDAO {

    Classedeconexao conusuario = null;
    String ordenacao = "id_os";

    public List<OrdemServicoVO> selecionarOrdensServicos(){

        List<OrdemServicoVO> ordemservico = new ArrayList<OrdemServicoVO>();
        conusuario = new Classedeconexao();
        conusuario.conecta();
        conusuario.executeSQL("Select * from tab_movimentacao_ordemservico order by "
+ ordenacao);
    }
}

```

```

    try {
        while (conusuario.resultset.next()) {
            OrdemServicoVO os_vo = new OrdemServicoVO();
            os_vo.setIds(conusuario.resultset.getString("id_os"));

            os_vo.setIdequipamento(conusuario.resultset.getString("TabEquipamento_idequi
pamento"));

            os_vo.setDataabertura(conusuario.resultset.getString("DataAbertura"));

            os_vo.setDatafechamento(conusuario.resultset.getString("DataFechamento"));

            os_vo.setHoraabertura(conusuario.resultset.getString("HoraAbertura"));

            os_vo.setHorafechamento(conusuario.resultset.getString("HoraFechamento"));

            os_vo.setOrigemlancamento(conusuario.resultset.getString("OrigemLancamento")
);
            ordemservico.add(os_vo);
        }
        conusuario.desconecta();

    } catch (SQLException erro) {
        System.out.println("Nao localizou dados " + erro);
    }

    return ordemservico;
}

public String lista_OrdemServico(ArrayList<OrdemServicoVO>
lista_OrdemServico_Json)
{
    System.out.println("ola celso - passei aqui.");

    for(OrdemServicoVO osvo : lista_OrdemServico_Json)
    {
        System.out.println("OS: " + osvo.getIds());
        System.out.println("Equipamento: " + osvo.getIdequipamento());
        System.out.println("DataAbertura: " + osvo.getDataabertura());
        System.out.println("DataFechamento: " + osvo.getDatafechamento());
        System.out.println("HoraAbertura: " + osvo.getHoraabertura());
        System.out.println("HoraFechamento: " + osvo.getHorafechamento());
        System.out.println("OrigemLancamento: " +
osvo.getOrigemlancamento());
        System.out.println("Agora é só gravar no banco de dados. -
Celso.\n\n");

        if(osvo.getOrigemlancamento().toString().equals("Tablet") ||
osvo.getOrigemlancamento().toString().equals("Editar Tablet"))
        {
            inserir(osvo);
        }
        if(osvo.getOrigemlancamento().toString().equals("Editar Desktop") )
        {
            Update(osvo);
        }
    }
}

```



```

        return "Lista de Usuarios inserida com sucesso";
    }

    public void inserir(OrdemServicoVO vo){
        vo.setOrigemlancamento("Desktop");

        String sql = "INSERT INTO tab_movimentacao_ordemservico
        (TabEquipamento_idequipamento, DataAbertura, DataFechamento, HoraAbertura,
        HoraFechamento, OrigemLancamento) VALUES ( "+ "'" + vo.getIdequipamento()+"'"+ " ",
        "+" + vo.getDataabertura()+"'"+ " ", " "+ "'" + vo.getDatafechamento()+"'"+ " ", " "+
        "'" + vo.getHoraabertura()+"'"+ " ", " "+ "'" + vo.getHorafechamento()+"'"+ " ", " "+
        "'" + vo.getOrigemlancamento()+"'"+ " )" ";

        conusuario = new Classedeconexao();
        conusuario.conecta();
        conusuario.InsertSQL(sql);
        conusuario.desconecta();
    }

    public void Update(OrdemServicoVO vo){

        String id = vo.getIdos().toString();
        int idos = Integer.parseInt(id);
        vo.setOrigemlancamento("Desktop");

        String sql = "update tab_movimentacao_ordemservico set
        TabEquipamento_idequipamento = "+ "'" + vo.getIdequipamento()+"'"+ " ", " +
        "DataAbertura = "+ "'" + vo.getDataabertura()+"'"+ " ", " + "DataFechamento =
        "+ "'" + vo.getDatafechamento()+"'"+ " ", " + "HoraAbertura =
        "+ "'" + vo.getHoraabertura()+"'"+ " ", " + "HoraFechamento =
        "+ "'" + vo.getHorafechamento()+"'"+ " ", " + "OrigemLancamento =
        "+ "'" + vo.getOrigemlancamento()+"'"+ " where id_os = " + idos;

        conusuario = new Classedeconexao();
        conusuario.conecta();
        conusuario.InsertSQL(sql);
        conusuario.desconecta();
    }
}

```

Pacote “SGA.VO”: É nele que fica as classes VO responsável por estabelecer os atributos da classe e os métodos assessores dos atributos. Para cada classe de Objetos, precisamos de uma classe VO para realizar a manipulação dos atributos dos objetos.

Abaixo mostra o código fonte da classe “**OrdemServicoVO**” responsável por realizar a manipulação dos atributos da Ordem de Serviço.

```

package sga.VO;

import java.io.Serializable;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class UsuarioVO implements Serializable{

    private static final long serialVersionUID = -6296966208133669986L;
    private String idusuario;
    private String senha;
    private String statusus;

    public String getIdusuario() {
        return idusuario;
    }
    public void setIdusuario(String idusuario) {
        this.idusuario = idusuario;
    }
    public String getSenha() {
        return senha;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }
    public String getStatusus() {
        return statusus;
    }
    public void setStatusus(String statusus) {
        this.statusus = statusus;
    }

    public UsuarioVO(){
    }

    public UsuarioVO(String idusuario, String senha, String statusus){
        this.idusuario = idusuario;
        this.senha = senha;
        this.statusus = statusus;
    }
}

```

Pacote “SGA.Resources”: É nele que fica as classes Resources responsável por estabelecer os métodos GET e POST para o envio e busca dos dados na URL do Web Service. Para cada classe de Objetos, precisamos de uma classe Resources para realizar a manipulação dos métodos GET e POST.

Abaixo mostra o código fonte da classe “**OrdemServiçoResource**” responsável por realizar a manipulação dos métodos GET e POST da Ordem de Serviço.

```

package sga.resources;

import java.util.ArrayList;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import sga.VO.OrdemServicoVO;
import sga.dao.OrdemServicoDAO;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonParser;

@Path("/ordens")
public class OrdemServicoResource {
    @GET
    @Path("/GET_ORDENS")
    @Produces("application/json; charset=UTF-8")
    public String selecionar_Ordens() {

        OrdemServicoDAO OS_DAO = new OrdemServicoDAO();

        return new Gson().toJson(OS_DAO.selecionarOrdensServicos());
    }

    @POST
    @Path("/POST_ORDENS")
    @Produces("application/json")
    @Consumes("application/json")
    public String Inserir_OrdensServicos(String lists_OS_Json) {

        Gson gson = new Gson();

        ArrayList<OrdemServicoVO> lista_OS = new ArrayList<OrdemServicoVO>();
        JsonParser parser = new JsonParser();
        JsonArray array = parser.parse(lists_OS_Json).getAsJsonArray();

        for (int i = 0; i < array.size(); i++) {
            lista_OS.add(gson.fromJson(array.get(i), OrdemServicoVO.class));
        }

        OrdemServicoDAO OS_DAO = new OrdemServicoDAO();

        return OS_DAO.lista_OrdemServico(lista_OS);
    }
}

```

4.3.2 Organização dos pacotes no *Android* para consumir Web Service em Java.

Para uma melhor organização, o pacote de conexão com Web Service em Java foi organizado da seguinte forma.

A figura 53 apresenta o pacote: “br.celso.gerenciamentoservicos.WS”

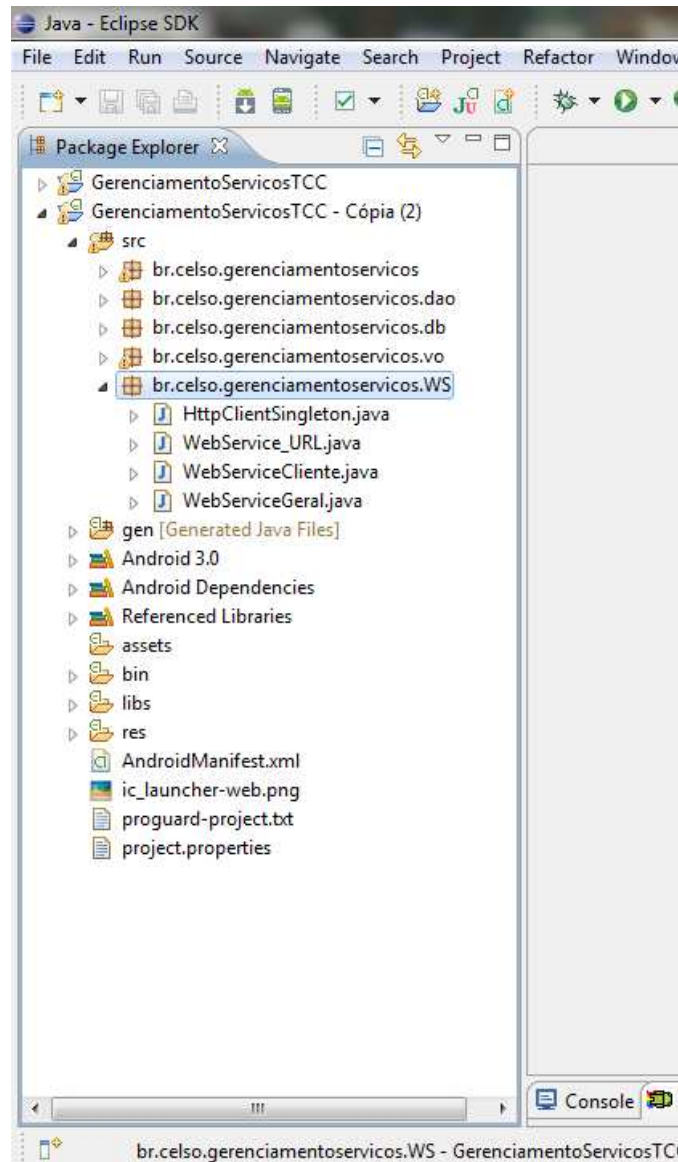


Figura 53 – Pacote para consumir Web Service em Java.

Classe “HttpClientSingleton”: É responsável por estabelecer o tempo de conexão com o Servidor de Internet. Caso ultrapassar o tempo determinado de busca no servidor nesta classe, é automaticamente interrompido para não deixar que a aplicação *Android* venha travar.

Abaixo mostra o código fonte da classe “**HttpClientSingleton**”.

```
package br.celso.gerenciamentoservicos.WS;

import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;

public class HttpClientSingleton {

    private static final int JSON_CONNECTION_TIMEOUT = 3000;
    private static final int JSON_SOCKET_TIMEOUT = 5000;
    private static HttpClientSingleton instance;
    private HttpParams httpParameters ;
    private DefaultHttpClient httpclient;

    private void setTimeOut(HttpParams params){
        HttpConnectionParams.setConnectionTimeout(params,
JSON_CONNECTION_TIMEOUT);
        HttpConnectionParams.setSoTimeout(params, JSON_SOCKET_TIMEOUT);
    }

    private HttpClientSingleton() {
        httpParameters = new BasicHttpParams();
        setTimeOut(httpParameters);
        httpclient = new DefaultHttpClient(httpParameters);
    }

    public static DefaultHttpClient getHttpClientInstace(){
        if(instance==null)
            instance = new HttpClientSingleton();
        return instance.httpclient;
    }
}
```

Classe “WebServiceCliente”: É responsável por executar os métodos GET e POST para qualquer transmissão de dados no Web Service. Esta classe é utilizada para a transmissão dos dados de qualquer classe VO do *Android*.

Abaixo mostra o código fonte da classe “**WebServiceCliente**”.

```
package br.celso.gerenciamentoservicos.WS;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URI;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import android.util.Log;

public class WebServiceCliente {

    public final String[] get(String url) {

        String[] result = new String[2];
        HttpGet httpget = new HttpGet(url);
        HttpResponse response;

        try {
            response = HttpClientSingleton.getHttpClientInstace().execute(httpget);
            HttpEntity entity = response.getEntity();

            if (entity != null) {
                result[0] = String.valueOf(response.getStatusLine().getStatusCode());
                InputStream instream = entity.getContent();
                result[1] = toString(instream);
                instream.close();
                Log.i("get", "Result from post JsonPost : " + result[0] + " : " +
result[1]);
            }
        } catch (Exception e) {
            Log.e("NGVL", "Falha ao acessar Web service", e);
            result[0] = "0";
            result[1] = "Falha de rede!";
        }
        return result;
    }
}
```

```

public final String[] post(String url, String json) {
    String[] result = new String[2];
    try {

        HttpPost httpPost = new HttpPost(new URI(url));
        httpPost.setHeader("Content-type", "application/json");
        StringEntity sEntity = new StringEntity(json, "UTF-8");
        httpPost.setEntity(sEntity);

        HttpResponse response;
        response = HttpClientSingleton.getHttpClientInstace().execute(httpPost);
        HttpEntity entity = response.getEntity();

        if (entity != null) {
            result[0] = String.valueOf(response.getStatusLine().getStatusCode());
            InputStream instream = entity.getContent();
            result[1] = toString(instream);
            instream.close();
            Log.d("post", "Result from post JsonPost : " + result[0] + " : " +
result[1]);
        }

    } catch (Exception e) {
        Log.e("NGVL", "Falha ao acessar Web service", e);
        result[0] = "0";
        result[1] = "Falha de rede!";
    }
    return result;
}

private String toString(InputStream is) throws IOException {

    byte[] bytes = new byte[1024];
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    int lidos;
    while ((lidos = is.read(bytes)) > 0) {
        baos.write(bytes, 0, lidos);
    }
    return new String(baos.toByteArray());
}
}

```

Classe “WebService_URL”: É responsável por guardar os endereços das URL’s de conexão com o Servidor Apache para executar os métodos GET e POST das classes VO que foram desenvolvidas.

Abaixo mostra o código fonte da classe **“WebService_URL”**.

```
package br.celso.gerenciamentoservicos.WS;

public class WebService_URL {

    private static final String IP = "10.0.2.2";
    private static final String PORTA = "8080";

    private static final String URL_WS_USUARIO_GET = "http://" + IP + ":" +
PORTA + "/SGA/usuarios/GET_USUARIOS";
    private static final String URL_WS_USUARIO_POST = "http://" + IP + ":" +
PORTA + "/SGA/usuarios/POST_USUARIOS";
    private static final String URL_WS_EQUIPAMENTO_GET = "http://" + IP + ":" +
PORTA + "/SGA/equipamentos/GET_EQUIPAMENTOS";
    private static final String URL_WS_ITEM_GET = "http://" + IP + ":" + PORTA
+ "/SGA/itens/GET_ITEM";
    private static final String URL_WS_AREA_GET = "http://" + IP + ":" + PORTA
+ "/SGA/areas/GET_AREA";
    private static final String URL_WS_ORDEM_GET = "http://" + IP + ":" + PORTA
+ "/SGA/ordens/GET_ORDENS";
    private static final String URL_WS_ORDEM_POST = "http://" + IP + ":" +
PORTA + "/SGA/ordens/POST_ORDENS";

    public String getUrlWsOrdemGet() {
        return URL_WS_ORDEM_GET;
    }
    public String getUrlWsOrdemPost() {
        return URL_WS_ORDEM_POST;
    }
    public String getUrlWsAreaGet() {
        return URL_WS_AREA_GET;
    }
    public String getUrlWsItemGet() {
        return URL_WS_ITEM_GET;
    }
    public String getUrlWsUsuarioGet() {
        return URL_WS_USUARIO_GET;
    }
    public String getUrlWsUsuarioPost() {
        return URL_WS_USUARIO_POST;
    }
    public String getUrlWsEquipamentoGet() {
        return URL_WS_EQUIPAMENTO_GET;
    }
}
```


Classe “WebServiceGeral”: É responsável por instanciar e executar os métodos GET e POST da classe **“WebServiceCliente”** para cada classe VO.

Abaixo mostra o código fonte da classe **“WebServiceGeral”**.

```
package br.celso.gerenciamentoservicos.WS;

import java.util.ArrayList;
import br.celso.gerenciamentoservicos.dao.OrdemServicoDAO;
import br.celso.gerenciamentoservicos.vo.OrdemServicoVO;
import android.app.Activity;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonParser;

public class WebServiceGeral extends Activity {

    private ProgressDialog dialog;
    String result = null;
    String Mensagem = null;
    WebService_URL url = new WebService_URL();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // setContentView(R.layout.main);

        new TarefaWS().execute();
    }

    class TarefaWS extends AsyncTask<Void, Void, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            dialog = ProgressDialog.show(WebServiceGeral.this, "Aguarde",
                "Acessando Web Service....SGA.");
        }

        @Override
        protected String doInBackground(Void... params) {

            try {
                PUT_OrdemServico();
                GET_OrdemServico();
                result = "OK";
                return result;
            } catch (Exception e) {
                Log.e("Celso:", " erro: Exception e: " + e);
                result = "erro";
                return result;
            }
        }
    }
}
```

```

@Override
protected void onPostExecute(String resultt) {
    resultt = result;
    super.onPostExecute(resultt);
    if (result == null) {
        Toast.makeText(WebServiceGeral.this,
            "Erro ao Acessar o Web Service...",
            Toast.LENGTH_LONG)
            .show();
    }
    dialog.dismiss();
    finish();
}

public void GET_OrdemServico() {
    String[] resposta = new
WebServiceCliente().get(url.getUrlWsOrdemGet());
    try {
        if (resposta[0].equals("200")) {
            Gson gson = new Gson();
            ArrayList<OrdemServicoVO> lista_Ordens = new
ArrayList<OrdemServicoVO>();
            JsonParser parser = new JsonParser();
            JSONArray js_array =
parser.parse(resposta[1]).getAsJsonArray();
            Log.e("celso:", "resposta[0]: " + resposta[0]);
            Log.e("celso:", "resposta[1]: " + resposta[1]);
            Log.e("celso:", "js_array: " + js_array);

            for (int i = 0; i < js_array.size(); i++) {
                Log.e("celso:", "js_array.get(i): " +
js_array.get(i));
                lista_Ordens.add(gson.fromJson(js_array.get(i),
OrdemServicoVO.class));
            }

            Log.e("celso:", "lista_Ordens: " + lista_Ordens);

            OrdemServicoDAO osdao = new OrdemServicoDAO(this);
            osdao.LimparTabela();

            for (OrdemServicoVO osVO : lista_Ordens) {
                osdao.inserir(osVO);
            }
        }
    } catch (Exception e) {
    }
}

```

```

    public String PUT_OrdemServico() throws Exception {

        OrdemServicoDAO osdao = new OrdemServicoDAO(this);

        ArrayList<OrdemServicoVO> lista_OS_vo = (ArrayList<OrdemServicoVO>)
osdao
            .selecionarOrdensServicos();

        Log.e("celso:", "osdao.selecionarOrdensServicos(): " + lista_OS_vo);
        Gson gson = new Gson();
        String clienteJSON = gson.toJson(lista_OS_vo);
        Log.e("celso:", "String clienteJSON = gson.toJson(lista_OS_vo);"
            + clienteJSON);

        String[] resposta = new WebServiceCliente().post(
            url.getUrlWsOrdemPost(), clienteJSON);
        if (resposta[0].equals("200")) {

            Log.e("celso:", "String PUT_OrdemServico(): " + resposta[1]);
            return resposta[1];
        } else {
            throw new Exception(resposta[1]);
        }
    }
}

```

5 – CONCLUSÃO.

A área de ciências da computação está crescendo cada vez mais no Brasil e no mundo. Com o uso de softwares integrados e específicos para uma determinada área em particular, tem contribuído para esse aumento, devido à necessidade de maior controle e levantamento de dados de processos das empresas.

O desenvolvimento deste trabalho foi dividido em duas fases: a primeira fase consistiu em adquirir conhecimento dos métodos e tecnologias envolvidas para o desenvolvimento dos aplicativos, enquanto a segunda fase, consistiu na implementação dos aplicativos, visando o resultado na prática tudo o que foi estudado.

A proposta de aprender novas tecnologias, principalmente àquelas pouco difundidas no meio acadêmico foi desafiador. Foram encontradas algumas dificuldades no início, mas o resultado obtido foi bastante satisfatório. Alguns conhecimentos adquiridos já estão sendo utilizados profissionalmente.

As pesquisas realizadas sobre as ferramentas, frameworks, tecnologias e arquiteturas envolvidas para o desenvolvimento dos aplicativos, foram às principais contribuições deste trabalho. Contribui de forma direta para os alunos, como material de estudo e para empresas que desejam adotar o uso destes padrões de projeto em suas aplicações, melhorando a qualidade e a produtividade.

Uma contribuição importante desta pesquisa é incentivar outras pessoas a desenvolver aplicativos com estas tecnologias, ou seja, os conhecimentos apresentados neste trabalho serão fundamentais para a elaboração de outros projetos de pesquisa no futuro.

6 – REFERENCIAS BIBLIOGRÁFICAS.

BASCI, D., M. S. **Data complexity metrics for xml web services**. Advances in Electrical and Computer Engineering, v. 9, p. 9 - 15, 2009.

BECKER, A. K.; SOBRAL, B. J.; CIARO, B. D. **Web Servicese XML - Um Novo Paradigma da Computação Distribuída** - Universidade Federal de Santa Catarina.

BRAY, T.; PAOLI, J.; SPERBERG-McQueen, C. M. and MALER, E. **Extensível Markup Language (XML) 1.0**. Second Edition. W3C Recommendation 6 October 2000 – Site: <http://www.w3.org/TR/2000/REC-xml-20001006> Acesso em 22 de Outubro, 2012.

BURÉGIO, V. A. A. **Desenvolvimento de aplicações para dispositivos móveis com .NET**. Recife/PE: UFP, 2003. 69 p. Dissertação (Trabalho de Conclusão de Curso) - Universidade Federal de Pernambuco, 2003.

CROCKFORD, DOUGLAS (2006). **JSON: The Fat-Free Alternative to XML**. JSON.ORG, 2006. Disponível em: <<http://www.json.org/fatfree.html> . Acesso em 22 Outubro, 2012.

DEITEL, H. M. et al. **C# Como programar**. Tradução João Eduardo Nóbrega Tortello. São Paulo: Pearson Education do Brasil, 2003.

DUARTE, E. MySQL. **SQL Magazine**, Rio de Janeiro, n. 37, p. 42, 2007.

EDUCACAO, PORTAL: **História e características da linguagem C#**. Disponível em: <<http://www.portaleducacao.com.br/informatica/artigos/6137/historia-e-caracteristicas-da-linguagem-c>>. Acesso em 19 de Outubro, 2012.

FIELDING, ROY; Architectural Styles and the Design of Network-based Software Architectures. Dissertação de Doutorado, University of California, 2000.

FILGUEIRAS, B. G. **3DViewer: Visualizador de Imagens Tridimensionais para o Sistema Operacional Android** - Universidade Federal do Rio de Janeiro, 2009.

G1, Intel investe em tablets, mas diz que é precipitado falar em 'era pós-PC'. Disponível em: <<http://g1.globo.com/tecnologia/noticia/2011/08/intel-investe-em-tablets-mas-diz-que-e-precipitado-falar-em-era-pos-pc.html> > Acesso em 05 de Outubro, 2012.

GONÇALVES, I. G. L. **Proposta de utilização da linguagem C# (C Sharp) para desenvolvimento de um Sistema de Apoio Administrativo sobre a Plataforma .NET**. Montes Claros/MG: UEMC, 2007. 63p. Monografia como requisito para conclusão do curso de Sistemas de Informação - Universidade Estadual de Montes Claros, 2007.

HANSEN, R. P. **GlueScript: Uma linguagem Específica de Domínio para composição de Web Services.** (Dissertação de Mestrado) - Programa Interdisciplinar de Pós-Graduação PIPCA - Universidade do Vale do Rio dos Sinos, 2003.

HANSEN, R. P.; PINTO, S. C. S. C. **XIV Simpósio Brasileiro de Informática na Educação – NCE - IM / UFRJ.** Construindo Ambientes de Educação Baseada na Web Através de Web Services Educacionais, p.61–70, 2003.

HIPP, WYRICK COMPANY, I. (2008). **About Sqlite.** Disponível em: <<http://www.sqlite.org/about.html>>. Acesso em 09 de Outubro, 2012.

ITO, G. C. **Uma arquitetura para geração de interfaces adaptativas para dispositivos móveis.** São José dos Campos/SP: INPE, 2008. 216p. Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada - Instituto Nacional de Pesquisas Espaciais, 2008.

KREGER, H. **Web Services Conceptual Architecture.** IBM Software Group, 2001.

LAGO, A. B.; SILVA, W. F. **XXI Congresso de Iniciação Científica e Tecnológica em Engenharia.** Coleta de dados do Sistema SIG@Livre utilizando dispositivos móveis, 2006.

LAUDON, K. C.; LAUDON, J. P. **Sistemas de informação.** Rio de Janeiro: LTC, 1999.

LECHETA, R. R. (2009). **Google Android - Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** Editora Novatec, 1º edição.

LIMA, E. **C# e .Net para desenvolvedores Eugênio Reis.** Rio de Janeiro, 2002.

LOPES, F. J. C.; RAMALHO, C. J. **Web Services: Metodologias de Desenvolvimento,** 2004.

MARCÍLIO, LUIZ. D. **Análise de linguagens de composição para Web Services – 2006 –** p.76 - Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática, Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

MILANI, A. **MySQL - Guia do Programador.** 1º ed. São Paulo: Novatec, 2006. p. 400 .

MSDN: **Introdução à linguagem C# e ao Framework .NET.** Disponível em: <[http://msdn.microsoft.com/pt-br/library/z1zx9t92\(v=VS.90\).aspx](http://msdn.microsoft.com/pt-br/library/z1zx9t92(v=VS.90).aspx)>. Acesso em 02 de Outubro, 2012.

NATANSOHN, G.; CUNHA, R. **Revistas Brasileiras Online:** plataformas móveis, Brasília, n. p.1-18, 2010.

PELISSOL, L.; LOYOLLA, W. **Aprendizado Móvel (M-learning): Dispositivos e Cenários**. São Paulo: 2004. Disponível em: <<http://www.abed.org.br/congresso2004/por/htm/074-TC-C2.ht>>. Acesso em 08 de Outubro, 2012.

RICHARDSON, LEONARD; RUBY Sam. **RESTful web services**. O'Reilly Media, 2007.

SANTOS, MICHAEL S. **Utilização de web services na plataforma .NET para a criação de um aplicativo visualizador de notícias para dispositivos móveis**. 2003. p.89 Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro Universitário Luterano de Palmas, Palmas.

SCHAEFER, C. **Protótipo de aplicativo para transmissão de dados a partir de dispositivos móveis aplicados a uma empresa de transporte**. Blumenau/SC: URB, 2004. 53p. Dissertação (Trabalho de Conclusão de Curso) – Universidade Regional de Blumenau, 2004.

SILVA, L. E. P. **Utilização da plataforma Android no desenvolvimento de um aplicativo para o cálculo do Balanço Hídrico Climatológico**. Dourados/MS: UEMS, 2009. 39p. Monografia da disciplina Projeto Final de Curso II para a obtenção do título de Bacharel em Sistemas de Informação – Universidade Estadual do Mato Grosso do Sul, 2009.

TANENBAUM, A. S. (2003). **Sistemas Operacionais Modernos**. Editora Pearson, 2ª Edição. São Paulo.

VENTURI, ELI – **Protótipo de um sistema para controle e monitoração residencial através de dispositivos móveis utilizando a plataforma .NET**. 2005 – p.69 - Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

WEBBER, JIM; PARASTATIDIS, Savas; ROBINSON Ian. **REST in Practice: Hypermedia and Systems Architecture**. O'Reilly Media, 2010.

WEBDIG, Web dig blog – **10 tentativas, na linha do tempo, a evolução dos Tablets!** – 2011- Disponível em: <<http://www.webdig.com.br/7309/linha-tempo-evolucao-tablets/>> Acesso em 05 de Outubro, 2012.

ZEMEL, T. **O que é um framework: definição e benefícios de se usar frameworks**. Disponível em: <<http://codeigniterbrasil.com/passos-iniciais/o-que-e-um-framework-definicao-e-beneficios-de-se-usar-frameworks/>>. Acesso em 03 de Outubro, 2012.

7 - ANEXOS

Mapa mental do Sistema de Gerenciamento de Custos de Equipamentos Agrícolas.

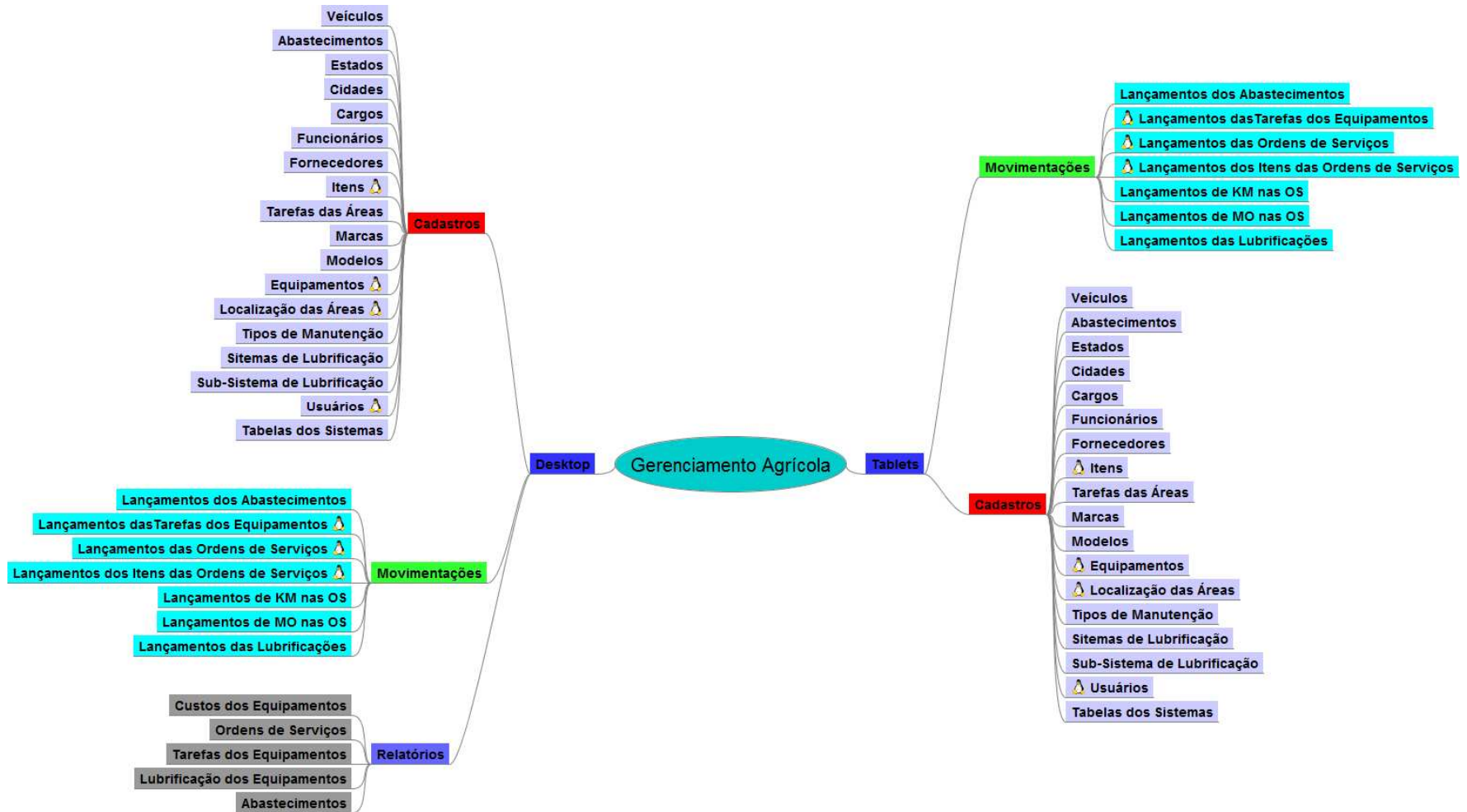


Figura 54 – Mapa Mental demonstrando uma visão geral do sistema.

Na figura 55, visão geral dos casos de usos de como seria o Sistema de Gerenciamento de Custos de Equipamentos Agrícolas.

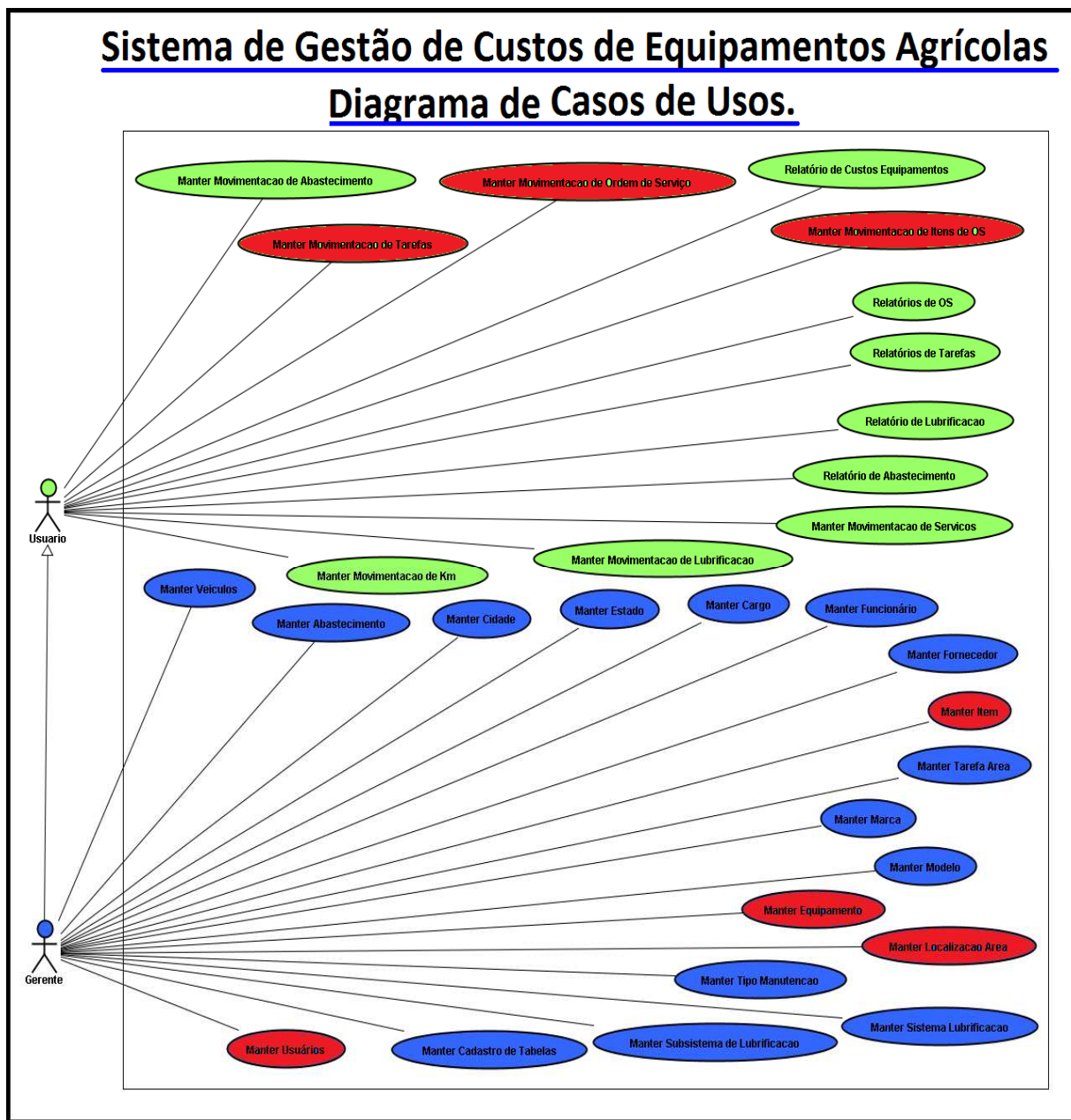


Figura 55 – Visão Geral dos Casos de Usos do Sistema.

Na figura 56, foi elaborado o Diagrama de Entidade e Relacionamento (DER) do Sistema de Gerenciamento de Custos de Equipamentos Agrícolas

Sistema de Gestão de Custos de Equipamentos Agrícolas

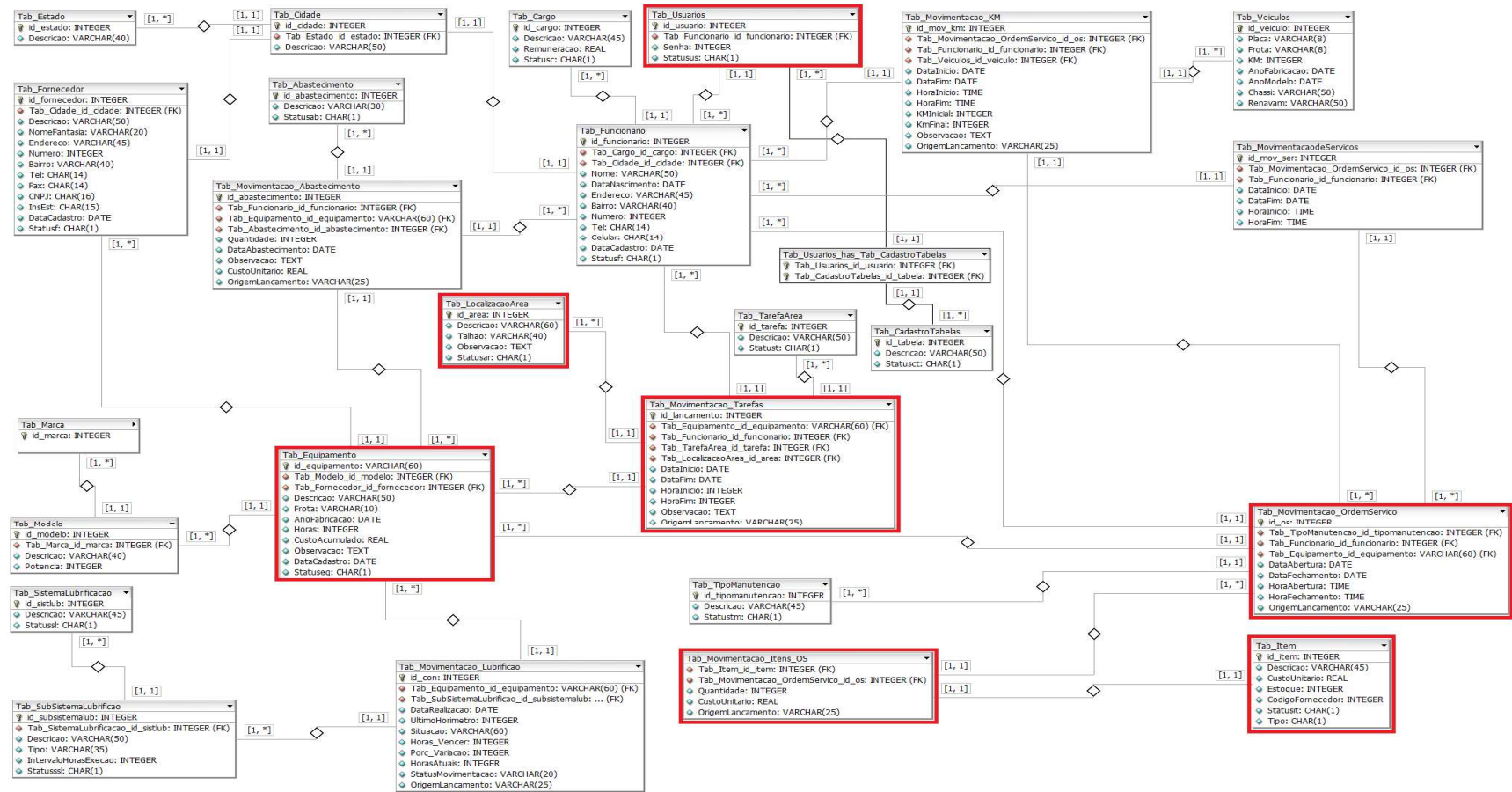


Figura 56 – Visão Geral do Diagrama de Entidade Relacionamento (DER).

A figura 57 mostra o diagrama de classe do pacote DAL.

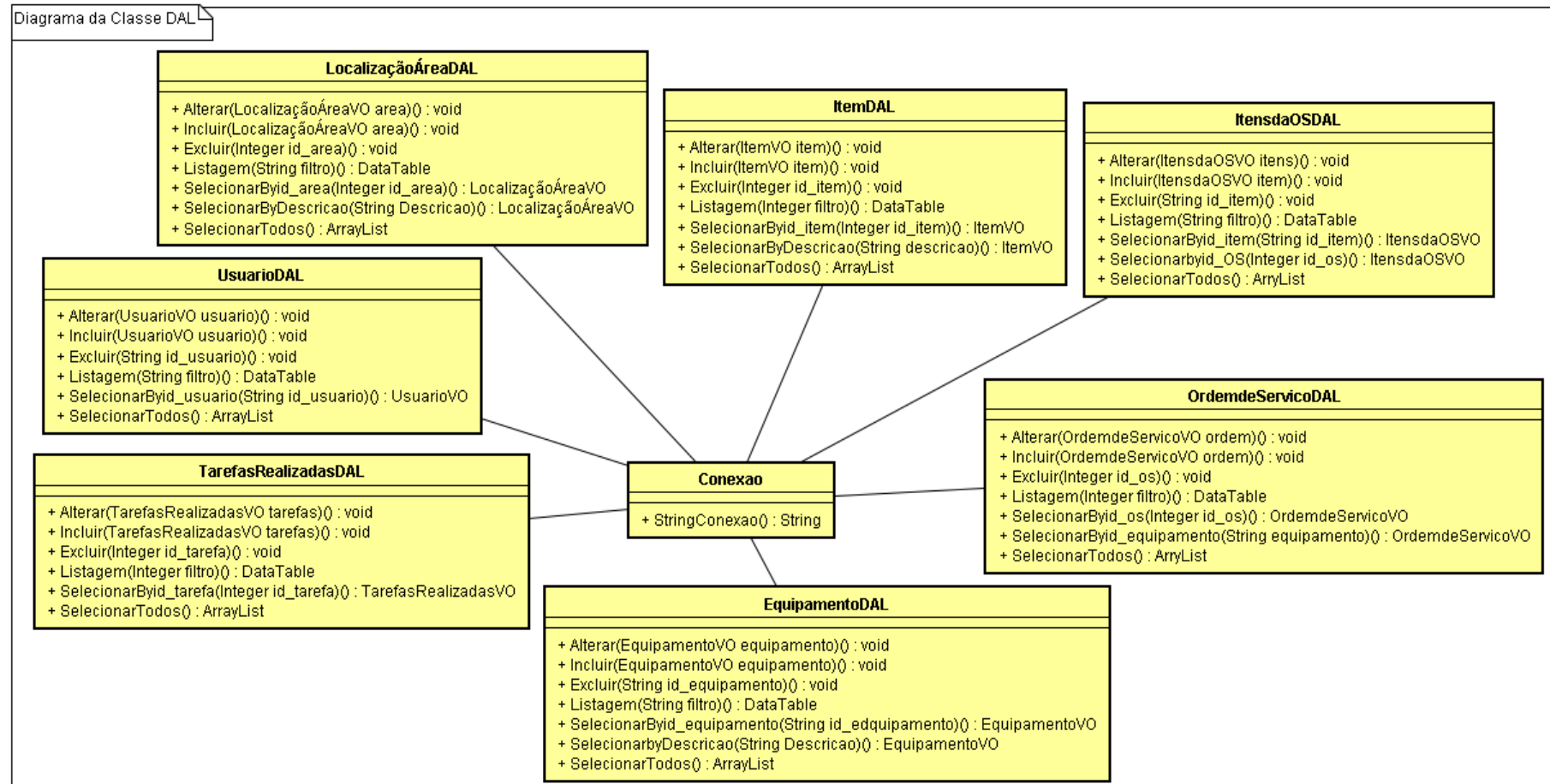


Figura 57 – Diagrama de Classes do Pacote DAL.

A figura 58 mostra o diagrama de classe do pacote BLL.

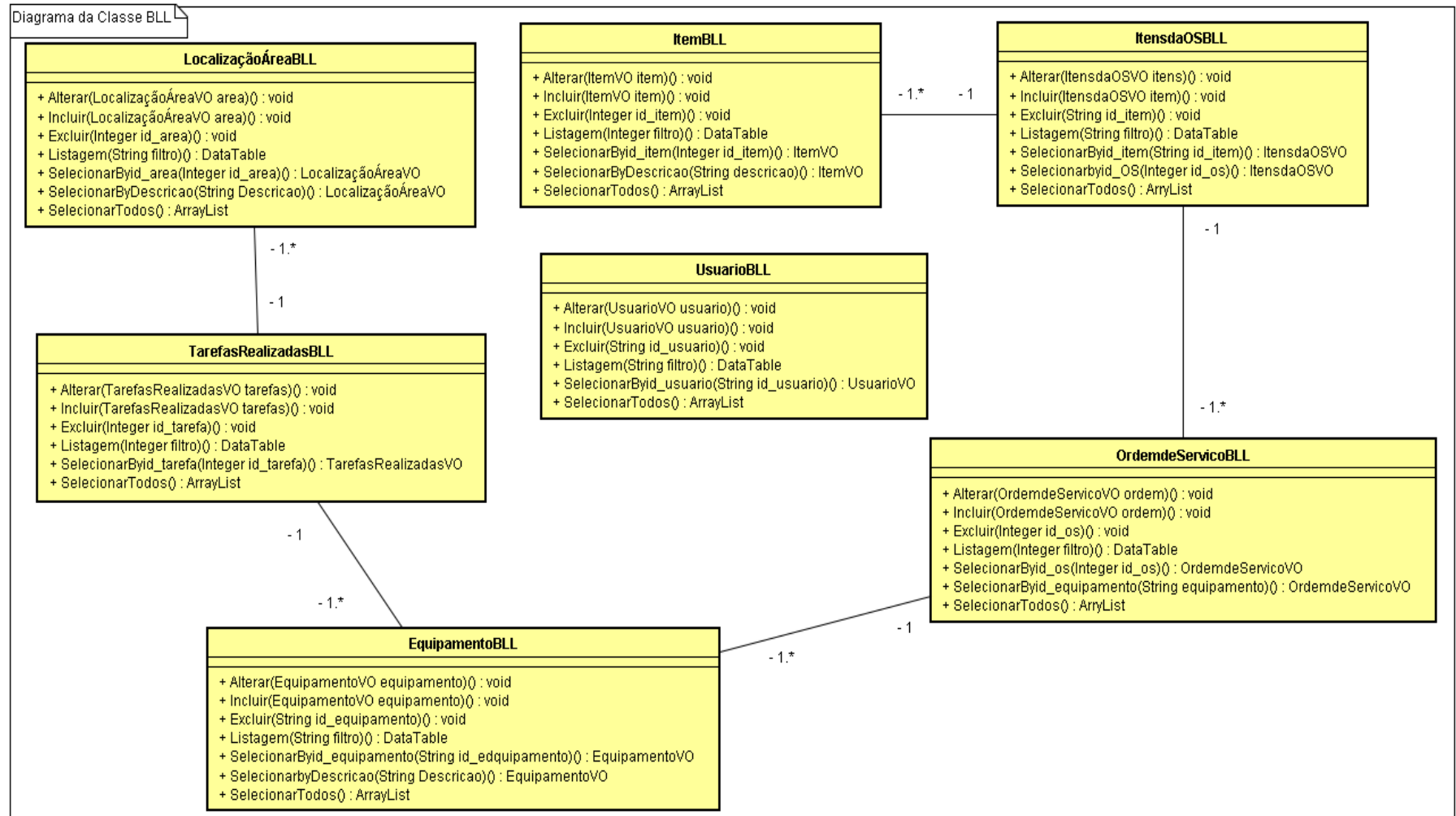


Figura 58 – Diagrama de Classes do Pacote BLL.

Na figura 59, é apresentado o diagrama de atividades do aplicativo desktop e do dispositivo móvel *Tablet*.

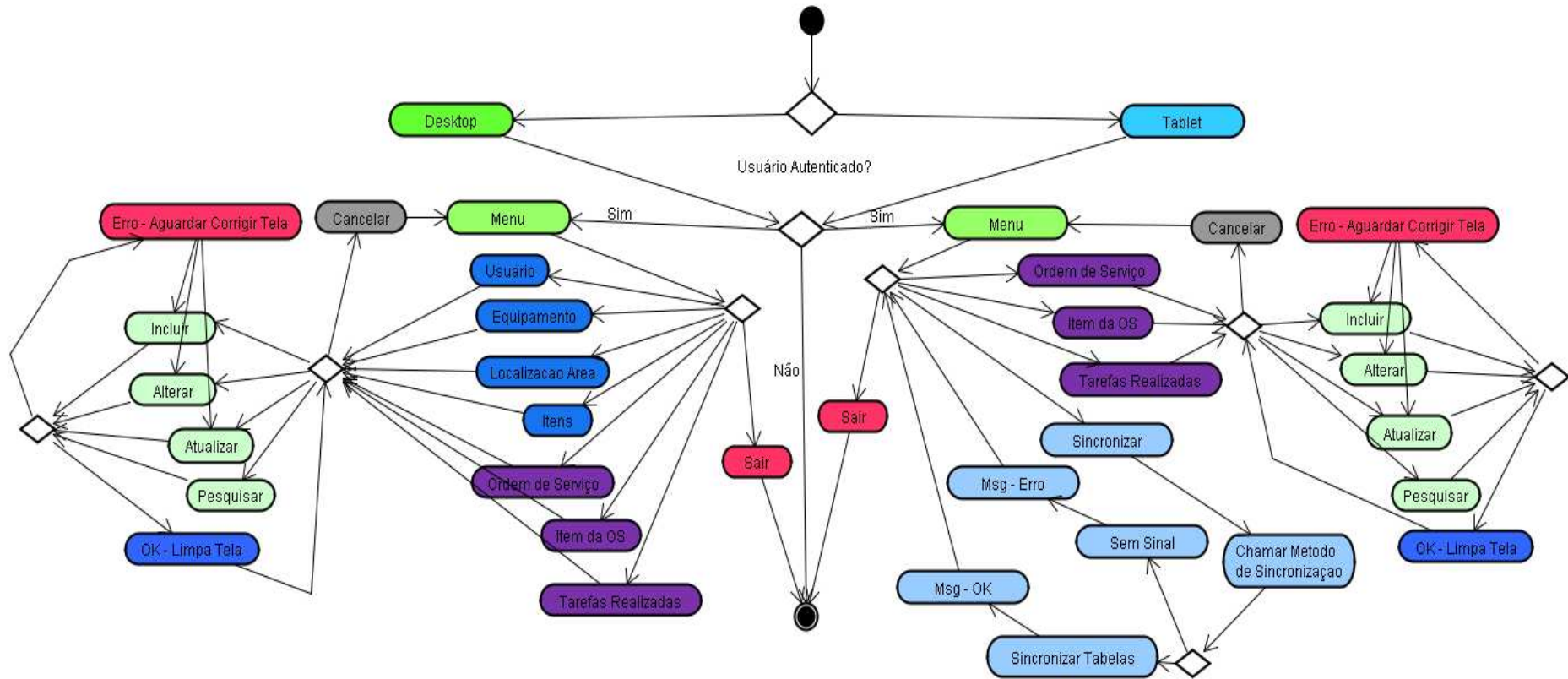


Figura 59 – Diagrama de Atividades de todo o processo dos protótipos.