Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

KLEBER APARECIDO DE SOUZA DOMINGOS

IMPLEMENTAÇÃO DE PROJETO EM GOOGLE ANDROID PARA GESTÃO DE TRATOS CULTURAIS

Assis
2014

KLEBER APARECIDO DE SOUZA DOMINGOS

IMPLEMENTAÇÃO DE PROJETO EM GOOGLE ANDROID PARA GESTÃO DE TRATOS CULTURAIS

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

Orientador: Prof. Esp. Guilherme de Cleva Farto

Área de Concentração: Informática

Assis
2014

FICHA CATALOGRÁFICA

DOMINGOS, Kleber Aparecido de Souza
Implementação de projeto em *Google Android* para gestão de tratos culturais / Kleber Aparecido de Souza Domingos. Fundação Educacional do Município de Assis – FEMA – Assis, 2014.
74p.

Orientador: Prof. Esp. Guilherme de Cleva Farto
Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA

1. Google Android **2**. Java **3**. PostgreSQL

CDD: 001.61
Biblioteca da FEMA

IMPLEMENTAÇÃO DE PROJETO EM GOOGLE ANDROID PARA GESTÃO DE TRATOS CULTURAIS

KLEBER APARECIDO DE SOUZA DOMINGOS

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: Prof. Esp. Guilherme de Cleva Farto

Analisador (1): Prof. Dr. Alex Sandro Romeo de Souza Poletto

Assis
2014

DEDICATÓRIA

Dedico este trabalho à Deus primeiramente, pois tem
me dado forças mesmo através das dificuldades
e a minha família que sempre me apoiou,
mesmo com a minha ausência.

AGRADECIMENTOS

Primeiramente a Deus, pois sem Ele, nada seria possível e minha família que sempre me apoio até mesmo dando tão pouca atenção em determinados momentos.
A esta universidade, seu corpo docente, direção e administração que oportunizaram avançar na minha vida profissional e pessoal.
Ao meu orientador Prof. Guilherme, pelo suporte nesse desafio proposto, correções e incentivos, mesmo na hora de quase desistir.
Aos meus pais, pelo amor, incentivo e apoio incondicional, minha querida esposa que sempre está dando forçar e me ajuda a vencer desafios e superar barreira a 15 anos da nossa vida junto.
A minhas filhas Heloisa e Laura que sempre entenderam as horas que não estava presente ou a falta de atenção.
E todos que diretamente ou indiretamente contribuíram para chegar até aqui nessa etapa da minha vida e a vencer, muito obrigado.

“A persistência é o menor caminho para o êxito.”

Charles Chaplin (1889-1977)

RESUMO

O objetivo geral desse trabalho é a pesquisa e compreender a estrutura de recursos oferecidos pelas tecnologias já utilizadas pelo usuário e somente adaptando-as de acordo com as suas necessidades, por se tratar de dois recursos que o usuário já possui um aparelho de celular e internet.

O sistema auxiliará um melhor acompanhamento das operações agrícola desde o preparo da terra até a entrega do produto colhido.

O sistema *Web* desenvolvido com JFS, *Primefaces* com uma base de dados em *PostgreSQL* vindo proporcionar ao usuário os cadastros que serão utilizados na mobilidade, apontamentos, relatórios, planejamentos de programação de operações futuras, localização dos dispositivos através do GPS, envios de mensagens para se comunicar com um determinando dispositivo que obtiver o módulo instalado, vindo assim ajudar nas tomadas de decisões.

O sistema mobile, o usuário realizará os apontamentos, após finalizados serão enviados para tabelas temporárias, validados e processados automaticamente no sistema *Web*, ficando somente os apontamentos com inconsistências para serem corrigidos e reprocessados. Reaproveitando o sistema de localização através do GPS que enviará a posição atual do dispositivo, recebimento e envio de mensagens mantendo a comunicação com o usuário da *Web Service*.

Palavras-chave: *Java*; XML; Orientação a Objetos; *Google Android; PostgreSQL*.

ABSTRACT

The aim of this study is to research and understand the structure of resources offered by technology already used by the user and only adapting them according to their needs, they are two features that the user already has a cell phone and internet .

The system will help better monitoring of agricultural operations from land preparation to delivery of the harvested product.

The Web system developed with JFS, Primefaces with a database in PostgreSQL coming provide the user the registers to be used for mobility, notes, reports, schedules programming for future operations, location of the device using GPS, sending messages to communicate with a device determining who obtains the module installed, so come help in decision making.

The mobile system, the user will hold the notes once finalized will be sent to temporary, automatically validated and processed in the Web system tables, leaving only the notes with inconsistencies to be corrected and reprocessed. Reusing the system location via GPS that sends the current position of the device, receiving and sending messages maintaining communication with the user of the Web Service.

Keywords: Java, XML, Object Oriented, Google Android, PostgreSQL.

LISTA DE ILUSTRAÇÕES

Figura 1 – Logotipo *Google Android*.....21
Figura 2 – A relação entre o Android e o hardware.....22
Figura 3 – Emulador Android.....24
Figura 4 – Estrutura do Google *Android*.....25
Figura 5 – Compilação do Código. 33
Figura 6 – Compilação do Código *Java*. 33
Figura 7 – Elementos de uma aplicação JSF. 36
Figura 8 – Camada de Persistência. 37
Figura 9 – Módulos do *PrimeFaces*.....38
Figura 10 – Mapa Mental do Sistema.....43
Figura 11 – Diagrama de Caso de Uso - Administrador. 46
Figura 12 – Diagrama de Caso de Uso comum – Adm. e Usuário – “Manter”. 47
Figura 13 – Diagrama de Caso de Uso comum – Adm. e Usuário – “Apontar”. 48
Figura 14 – Diagrama de Caso de Uso comum – Adm. e Usuário – “Emitir”. 48
Figura 15 – Diagrama de Caso de Uso – *Mobile* – Administrador.....49
Figura 16 – Diagrama de Caso de Uso comum – *Mobile* Pragas – Adm. e Usuário – “Realizar”.....49
Figura 17 – Diagrama de Atividades do UC – *Web* – Apto. – Infestação de Pragas. 50
Figura 18 – Diagrama de Sequência do UC –*Web* – Apto. – Infestação de Pragas. 51
Figura 19 – Diagrama de Classe – *Mobile* – Apto. – Infestação de Pragas. 52
Figura 20 – DER - Sistema Web. 53
Figura 21 – DER – Aplicativo *Mobile* – Apontamento de Operações Agrícola. 54
Figura 22 – DER – Aplicativo *Mobile* – Apontamento de Plantio. 54
Figura 23 – DER – Aplicativo *Mobile* – Apontamento de Infestação de Pragas. 55
Figura 24 – DER – Aplicativo *Mobile* – Apto. de Aplicações de Produtos Agrícola. ..55
Figura 25 – DER – Aplicativo *Mobile* – Apontamento de Colheita.....56
Figura 26 – DER – Aplicativo *Mobile* – Apontamento de Entrega. 56
Figura 27 – Estrutura Analítica do Projeto. 58
Figura 28 – Sequenciamento das Atividades. 59
Figura 29 – Estrutura em Camadas. 61
Figura 30 – Tela de Login - Web. 61
Figura 31 – Tela de Cadastros - Web. 62
Figura 32 – Tela de Apontamentos Temporários – Web. 63
Figura 33 – Tela de Apontamentos – Web. 63
Figura 34 – Tela de Planejamentos. 64
Figura 35 – Tela de Localização e Envio de Mensagens. 65
Figura 36 – Tela de Login – Google Android. 66
Figura 37 – Tela de Menu Principal – Google Android. 66
Figura 38 – Tela de Menu Apontamentos – Google Android. 67
Figura 39 – Tela de Apontamentos de Operações – Google Android. 68
Figura 40 – Tela de Mensagens – Google Android. 68

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS 14
1.2 JUSTIFICATIVAS 15
1.3 PÚBLICO ALVO 15
1.4 MOTIVAÇÃO 16
1.4 ESTRUTURA DO TRABALHO 16
2 TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO 17
2.1 TECNOLOGIA GOOGLE ANDROID 18
2.1.1 Introdução Ao Google Android 18
2.1.2 *Google Android* e *Open Handset Alliance* 19
2.1.3 Sistema Operacional Linux 21
2.1.4 Máquina Virtual *Dalvik* 22
2.1.5 Desenvolvimento de Aplicações *Google Android* 23
2.1.6 *Android* SDK 24
2.2 PLATAFORMA GOOGLE ANDROID 25
2.2.1 Camada de Aplicações 25
2.2.2 Camada *Framework* 26
2.2.3 Camada Biblioteca 27
2.2.4 Camada *Runtime* 28
2.2.5 Camada *Linux Kernel* 28
2.3 TECNOLOGIA *JAVA* 28
2.3.1 Histórico da Linguagem 28
2.3.2 Introdução à *Java* 29
2.3.3 Característica-Chaves do *Java* 30
2.3.4 Máquina Virtual Java 32
2.3.5 Os *Bytescodes Java* 34
2.3.6 *Java Enterprise Edition* 34
2.3.7 *Javaserver Faces* 35
2.3.8 *Java Persistence* 36
2.3.9 *Framework Primefaces* 38
2.4 BANCO DE DADOS *POSTGRESQL* 39

2.4.1 Histórico do Banco de Dados Postgresql 39
2.4.2 Introdução ao Banco de Dados Postgresql 40
2.4.3 Principais Características do Banco de Dados Postgresql 40
6 ANÁLISE E ESPECIFICAÇÃO DO SISTEMA 43
3.1 MAPA MENTAL 43
3.2 LISTA DE REQUISITOS 44
3.3 DIAGRAMA E ESPECIFICAÇÃO DE CASO DE USO 45
3.3.1 Caso de Uso – Sistema Web 45
3.3.2 Caso de Uso – Aplicativos Mobile 49
3.4 DIAGRAMA DE ATIVIDADE 50
3.5 DIAGRAMA DE SEQUENCIA 51
3.6 DIAGRAMA DE CLASSE 51
3.7 MODELO – ENTIDADE-RELACIONAMENTO 53
4 ESTRUTURA DO PROJETO 57
4.1 ESTRUTURA ANALITICA DO PROJETO 57
4.2 SEQUENCIAMENTO DAS ATIVIDADES 58
5 IMPLEMENTAÇÃO DO APLICATIVO 60
5.1 APLICATIVO WEB 60
5.1.1 Tela Login 61
5.1.2 Tela de Cadastros 62
5.1.3 Tela de Apontamentos Temporários 62
5.1.4 Tela de Apontamentos 63
5.1.5 Tela Programação e Planejamento 64
5.1.6 Tela Localização e Envio Mensagem 64
5.2 APLICAÇÃO GOOGLE ANDROID 65
5.2.1 Tela Login *65*
5.2.2 Tela Menu Principal 66
5.2.3 Tela Menu de Apontamentos 67
5.2.4 Tela de Apontamento de Operações 67
5.2.5 Tela Mensagens 68
6 CONCLUSÃO 69
REFERÊNCIAS BIBLIOGRÁFICAS 70

1 – INTRODUÇÃO

Com o grande crescimento no Mercado de Agronegócio, vem também a necessidade de novos desenvolvimentos para o auxílio para tomada de decisão e conseguir o máximo de produtividade. Tendo vista a adaptação dessas tecnologia, para que possam ser utilizadas. Porem há um grande desafio, na busca por tecnologias de baixo custo e ao mesmo tempo, ajudar nas tomadas de decisões, que poderá obter a melhor produtividade e assim dando um grande salto no mercado competitivo.

Sendo assim se torna necessário um estudo mais profundo nas tecnologias que o mercado oferece com o custo baixo, que podem ser readaptada para a necessidade do usuário. Hoje em dia aqueles que estão no ramo de agronegócio (produtor) já tem essas ferramentas juntamente com eles, porém bastam ser adaptadas para a suas necessidades, sendo uma delas seriam o celular e a internet.

Um desses recursos que poderão ajudar na adaptação dessas ferramentas seria um Sistema Operacional desenvolvido pelo Andy Rubin em outubro de 2003 o *Android*, utilizados em celulares, porem se tratavam de projetos secretos. Em Agosto de 2005 a Google anuncia a compra da *Android* Inc., dando início assim os primeiros passos no ramo de mobilidades e mudando o conceito de celulares. Após 3 anos são lançados os primeiros celulares com o seu sistema operacional trazendo várias novas ferramentas avançadas para o usuário daquela época.

Se tratando de um sistema *open-source* e com grande visão da *Google* com o seu crescimento no mercado competitivo de mobilidade, são disponibilizados para desenvolvedores ferramentas que possam ajudar a divulgação de novos aplicativos para o seu sistema no mercado, e consequentemente levando a uma grande quantidade de aplicativos desenvolvidos e disponibilizados para usuários de várias categorias e com necessidades diferentes, como: mapas, aplicativos com banco de dados integrado, GPS entre outros. O *Android* está na sua versão 4.3 *Jelly Bean* passando por várias melhorias e novidades.

Outra recurso disponível para ajudar no desenvolvimento dessa adaptação seria a linguagem programação em *Java*. Sendo uma linguagem orientada a objetos, que ajuda o desenvolvedor atuar em várias plataformas *Desktop*, *Web*, Mobilidades entre outros. Sendo também utilizado para o desenvolvimento dos aplicativos no sistema operacional *Android*. Outra vantagem dessa ferramenta que pode ser utilizada em diferentes sistemas operacionais sem a necessidade de rescrever o seu código no sistema operacional que será utilizado, só havendo a necessidade da instalação de sua máquina virtual.

Com estudos nas necessidades do mercado de agronegócio, o usuário com baixo custo utilizando as ferramentas que já obtém, com a adaptação dessas mesma, podem ajudar a tomar determinadas decisões que poderão fazer a diferença e colocar um nível mais alto de competividade do mercado.

1.1 OBJETIVOS

O objetivo geral desse trabalho é a pesquisa e compreender a estrutura de recursos oferecidos pelas tecnologias já utilizadas pelo usuário e somente adaptando-as de acordo com as suas necessidades, por se tratar de dois recursos que o usuário já possui um aparelho de celular e internet. A partir desse estudo será possível realizar um melhor comparativo entre os pontos positivos e negativos e a o custo final para o usuário.

Pretende adquirir conhecimento necessário para o desenvolvimento dos aplicativos necessário que auxiliaram o usuário nas tomadas de decisões e gerenciamento de processos.

Sendo os objetivos específicos:

- O sistema operacional *Android* com estudo em *Java Android* para um aplicativo de apontamento, consultas;
- E as tecnologias *Java* EE, JFS, *PrimeFaces*, *Jasper Report*, para um sistema Web de validações de dados apontados, consultas e relatórios.
- Pesquisa de caso de uso:
  - Levantamentos de requisitos necessários;

- Desenvolver uma arquitetura do ambiente;
- Implementação do sistema;
- Teste e homologação;
- Descrever os resultados obtidos.

1.2 JUSTIFICATIVAS

Com a grande necessidade de recursos de baixo custo para obter grandes lucros cada vez mais no mercado tão competitivo, não é diferente no mercado do agronegócio, pois tem grandes desafios para alcançar altos níveis de produção. Outro grande motivo também de se tratar de um mercado necessitado de tecnologias que ajudem desde do pequeno ao grande produtor sem muitas dificuldades para obterem as metas desejadas.

Com finalidades de atrair esse mercado carente e por se tratar de reaproveitar recursos tecnológicos que já são sendo utilizados pelo produtores no seu dia a dia, isso ao um estudo mais profundos nessas tecnologia para obter um projeto de baixo custo com gestão de tomadas de decisões.

Sendo esses recursos o celular com Sistema Operacional *Android* e internet. Tratando-se de um sistema operacional *open-source*, que pode obter-se os recursos ao máximo do aparelho e do sistema operacional para desenvolvimento de *software* com auxílio de banco de dados integrado no próprio sistema *Android* e da internet para execução de uma aplicação Web de gestão e relatórios juntamente com um banco de dados *PostgreSQL*.

1.3 PÚBLICO ALVO

O sistema a ser desenvolvido neste trabalho buscará atender as necessidades do pequeno ao grande agricultor facilitando tomadas de decisões tendo a levar para uma produtividade maior com baixo custo.

Por se tratar de uma área que cresce a cada dia mais a necessidade de novas tecnologias para auxiliar nas tomadas de decisões e gestão operacional, serão adaptadas tecnologias já utilizadas no dia a dia para evitar custo elevado no projeto.

1.4 MOTIVAÇÃO

A principal motivação o conhecimento nas tecnologias que serão utilizadas nesse projeto, pois de se tratar de tecnologias que estão aquecendo atualmente o mercado econômico na atualidade e também adquirir muito mais conhecimento em um banco de dados robusto e *open-source*. Com esse conhecimento adquirido venha me ajudar a sair na frente no mercado de trabalho.

1.5 ESTRUTURA DO TRABALHO

Este trabalho está estruturado nas seguintes partes:

- **Capítulo 1 – Introdução,** ”Este capítulo trás um breve apanhado sobre o projeto proposto, objetivo do projeto, justificativa, publico alvo, motivação e estrutura do trabalho.”
- **Capítulo 2 – Tecnologia e Ferramentas de Desenvolvimentos,** ”Este capítulo tem um detalhamento das tecnologias *Google Android*, Java e banco de Dados *PostgreSQL* que serão utilizadas no desenvolvimento do sistema.”
- **Capítulo 3 – Análise e Especificação do Sistema,** ”Este capítulo serão comentados os principais diagramas levantados que ajudarão nos requisitos de desenvolvimento do projeto.”
- **Capítulo 4 – Estrutura do Projeto,** “Este capítulo veremos os diagramas de estrutura analítica, sequenciamento das atividades, que foram utilizados para o acompanhamento e desenvolvimento do projeto.
- **Capítulo 5 – Conclusão,** ”Este capítulo nos trás as considerações finais do projeto.”
- **Referências Bibliográficas**

2 – TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

O intuito deste capítulo é apresentar as ferramentas e tecnologias que serão empregadas para o desenvolvimento do sistema. A plataforma *Google Android*, foi escolhida, pois por se tratar de uma plataforma *open-source* e também por ser utilizadas em diversos aparelhos celulares e outras mobilidades de fácil acesso à qualquer usuário. Torna-se o projeto de baixo custo, sendo um atrativo para o grande mercado competitivo. Outro fato que levou a utilização dessa plataforma, que hoje em dia o mercado coorporativo está extremamente exigente, pois quanto mais rápido obtiver as informações estando *online*, poderão evitar prejuízos, com as informações utilizadas nas tomadas de decisões.

A tecnologia *Java* foi escolhida, por se tratar de uma tecnologia orientada a objetos de alto nível, multiplataformas trabalhando em *Virtual Machine* – JVM (Máquina Virtual *Java*), assim evitando a incompatibilidade que ocorre muito em outras tecnologias. Utilizando as IDE (*Integrated Development Environment*) que serão adotados para o desenvolvimento do sistema são o *Eclipse* e o *Netbeans*, são de fácil acoplamento as novas bibliotecas, componentes, frameworks e diversas API (*Application Programming Inteface*).

Desta forma utilizando a tecnologia *JavaServer* para desenvolvimento com o *Framework PrimeFaces* para a camada visual fazendo a interação com o usuário, torna-se possível a implementação de aplicações *Web*, sendo capazes de utilizar em qualquer navegador de internet e dispositivos *Mobile*. Juntamente com essas tecnologias para a persistência e manipulação de dados será utilizado o banco de dados *PostgreSQL*, por se tratar de um SGDB *open-source*, robusto e confiável, tão quanto o *Oracle*.

Sendo utilizado o servidor de aplicação *Apache Tomcat* é um contêiner *Java* e um servidor *Web* ao mesmo tempo, tendo suporte às tecnologias *JavaServelet* e *JavaServer*, embora seja robusto o suficiente para ser utilizado em ambientes de produção.

2.1 – TECNOLOGIA *GOOGLE ANDROID*

O *Android* se trata de uma plataforma de código-aberto criado inicialmente pela Google e mantida pela *Open Handset Alliance* (OHA), com o objetivo de desenvolvimentos de aplicativos para dispositivos móveis. No momento se tornando uma revolução no desenvolvimento de aplicações. Constitui o seu sistema operacional baseado no *Kernel* do *Linux* e diversos aplicativos, com uma rica interface gráfica, com navegador de internet, suporte à multimídia, banco de dados *Mobile*, integração com a API do *Google Maps* e muito outras características.

O desenvolvimento e integração de seus aplicativos utiliza-se a linguagem de programação *Java* e um IDE ou ambiente de desenvolvimento de alto nível.

2.1.1 – Introdução ao *Google Android*

Com o grande crescimento do mercado de mobilidade, estudos mostram que, atualmente já existem cerca de seis bilhões de pessoas que possuem aparelhos celulares e *tablets*, vindo assim se torna um mercado bem promissor, não restando dúvidas que inovações são inevitáveis nessa área.

O *Android* foi desenvolvido pelo Andy Rubin em outubro de 2003, utilizados em celulares, porem se tratavam de projetos secretos. Em Agosto de 2005 a *Google* anuncia a compra da *Android* Inc., dando início assim os primeiros passos no ramo de mobilidades e mudando o conceito no mercado de aparelhos móveis.

Com essa motivação e visão de mercado a *Google* lança uma plataforma de código-aberto juntamente com a parceria com dezenas de empresas do ramo da telefonia móvel, como a *Motorola, Samsung, Sony Ericsson* e muito outras, dando-se a origem à *Open Handset Alliance* (OHA), sendo a primeira plataforma totalmente livre para aplicações móveis, já visando o futuro da mobilidade, abrindo caminho para novos desenvolvedores que ajudaram a tornar o seu sistema um grande sucesso, disponibilizando um kit de desenvolvimento o *Android* SDK e APIs, necessárias para o desenvolvimentos de aplicações para sua plataforma, utilizando a linguagem *Java* (PEREIRA; SILVA,2009), (Meier,2012) e (LECHETA,2009).

O *Android* se trata de uma plataforma completa, escrita para dispositivos móveis tendo o seu código totalmente aberto para desenvolvimentos de novos aplicativos para mobilidade, tendo um sistema operacional, *middleware*, aplicativos e interface do usuário muito amigável (PEREIRA; SILVA,2009).

Sendo desenvolvido com base no sistema operacional Linux, é composta por um conjunto de ferramentas que atua em toda a fase de desenvolvimento do projeto, desde a execução até a criação de softwares específicos, no entanto seus componentes do SO são escrito em C ou C++, já os aplicativos para o usuário são em *Java*. Tendo uma característica essa plataforma, não existindo diferença entre aplicativos integrados e os aplicativos desenvolvidos pelo SDK, sendo assim pode-se utilizar todos os recursos disponíveis no dispositivo (Ableson; Sen; King; Ortiz, 2012).

Para os fabricantes de celulares, obtiveram uma grande vantagem, por ter um sistema operacional sem ter que pagar por ele e ainda customizar de acordo com as suas necessidades, sem precisar compartilhar as alterações, pois se trata de um sistema *open-source* (LECHETA,2009).

2.1.2 – Google Android e Open Handset Alliance

Formado por empresas líderes do mercado na telefonia de celulares liderados pela Google, sendo essas empresas *Motorola, Samsung, Sony Ericsson* e muitas outras.

Esse grupo é formado atualmente por 84 empresa que se uniram para acelerar a inovação em móveis e oferecer aos consumidores uma tecnologia mais rica e barata.

No site da OHA (*OPEN HANDSET ALLIANCE*, 2014) existe uma ótima descrição do que seria essa aliança. O texto, traduzido do inglês, enfatiza:

> Hoje, existem 1,5 bilhão de aparelhos de televisão em uso em todo o mundo e um bilhão de pessoas têm um telefone celular, tornando o aparelho um dos produtos de consumo mais bem-sucedidos do mundo. Dessa forma, construir um aparelho celular superior melhoraria a vida de inúmeras pessoas de todo o mundo. A *Open Handset Alliance* é um grupo formado por empresas líderes em tecnologias moveis que compartilham essa visão para mudar a experiência móvel de todos os consumidores [...].

Assim o objetivo do grupo é definir uma plataforma única e aberta para celulares para deixar os consumidores mais satisfeitos com o produto final. Outro objetivo principal dessa aliança é criar uma plataforma moderna e flexível para o desenvolvimento de aplicações coorporativas. O resultado dessa união de empresas criou-se o *Android* (LECHETA,2009).

O *Android* é a nova plataforma de desenvolvimento para aplicativos móveis, com o seu sistema operacional baseado em Linux, uma interface visual amigável, sendo utilizado para desenvolvimentos de seu aplicativos a linguagem *Java* e o seu SO tem uma capacidade de segurança muito forte (LECHETA,2009) e (PEREIRA; SILVA,2009).

A *Open Handset Alliance* consiste em todas as estruturas envolvidas no processo da telefonia móvel:

- OPERADORAS DE CELULAR: responsável pela conexão e o serviço para o usuário final;
- FABRICANTES DE APARELHOS: responsáveis pela criação do hardware;
- EMPRESAS DE SEMICONDUTORES: fazem os chips dos aparelhos celulares;
- EMPRESA DE SOFTWARE: desenvolvem os softwares que serão executados no Android;

- EMPRESAS DE COMERCIALIZAÇÃO: são as empresas responsáveis pela divulgação, marketing e comercialização dos produtos para o usuário.

A intenção é construir em conjunto com o sistema operacional de código aberto, qualquer software instalado no aparelho terá o mesmo privilégio que, hoje apenas aplicativos criados ou licenciados pelo fabricante tem. Sendo assim, será possível total integração e customização entre sistema e aparelhos (PEREIRA; SILVA,2009).

A Figura 01 ilustra o logotipo escolhido para o *Android*.

Figura 1 – Logotipo *Google Android*

2.1.3 – Sistema Operacional Linux

O sistema operacional do *Android* foi baseado no *kernel* 2.6 *Linux* e é responsável por gerenciar a memória, os processos, threads e a segurança dos arquivos e pastas, além de redes e drives, o *kernel* atua também como uma camada de abstração entre hardware e o resto da pilha de software. Com uma interface de usuário rica, aplicativos de usuário, bibliotecas de códigos, frameworks de aplicativo, suporte a multimídia, até as funcionalidades de telefone e muito mais (LECHETA,2009), (PEREIRA; SILVA,2009) e (Ableson; Sen; King; Ortiz, 2012).

Caso necessário, o próprio sistema operacional decide em encerrar alguns processos para liberar mais memória e recursos e até reiniciar o processo posterior se necessário (LECHETA,2009).

Figura 02 ilustra a relação entre o *Android* e o *hardware* em que ele é executado.

Figura 2 – A relação entre o Android e o hardware

2.1.4 – Máquina Virtual Dalvik

Como sabemos a linguagem *Java* utilizada para construção de aplicações para o *Android* e o seu sistema operacional não existem uma máquina virtual *Java* (JVM), na verdade temos uma máquina virtual otimizada para execução em dispositivos móveis chamada de *Dalvik* (LECHETA,2009).

O *Dalvik* é uma máquina virtual com melhor desempenho, maior integração com a nova geração de hardware e projetada para executar várias máquinas virtuais paralelamente, sendo projetada para funcionar em sistema de baixa frequência de CPU, pouca memória RAM disponível e sistema operacional sem espaço de *Swap* (PEREIRA; SILVA,2009).

O *Dalvik* executa arquivo .dex (*Dalvik Executable*), que são classes de *Java* convertida para a máquina virtual através da ferramenta DX, distribuída juntamente

com o SDK do *Android*, baseia-se no *kernel Linux* para funcionalidades subjacentes como encadeamento e a gestão de baixo nível de memória. Os arquivos .dex e outros recursos como imagens são compactados dentro de um único arquivo com a extensão .apk (*Android Package File*), representa a aplicação final, pronta para ser distribuída e instalada (PEREIRA; SILVA,2009) e (LECHETA,2009).

As coisas importantes a saber sobre o *Dalvik* VM são aplicativos *Android* são executados dentro dela e que ela depende do *kernel* para serviços como gerenciamentos de processos, memórias e sistema de arquivos (Ableson; Sen; King; Ortiz, 2012).

2.1.5 - Desenvolvimento De Aplicações Google Android

O *Android* é a primeira plataforma para aplicações em móveis completamente livre e código aberto, sendo isso a grande vantagem para a sua evolução, uma vez que diversos programadores do mundo poderão contribuir para melhorar a plataforma (LECHETA,2009).

Por se tratar da linguagem *Java* para o desenvolvimento de aplicações para o *Android*, se tornou possível esse grande crescimento da plataforma, usando seu ambiente de desenvolvimento preferido como *Eclipse* ou *Netbeans*.

Sendo o ambiente preferido pela *Google*, o *Eclipse* tem um *plug-in* chamado ADT (*Android Development Tools*) para facilitar o desenvolvimento, testes e a compilação do projeto.

Utilizando o *plug-in* ADT é possível executar o emulador do *Android* diretamente do *Eclipse*, sendo possível utilizar todos os seus recursos como o debug passo a passo, que funciona perfeitamente, visualizando logs e simulando envio de uma mensagem SMS ou de uma ligação telefônica, além da capacidade de visualizar e enviar arquivos para o emulador, executar o *garbage collector*, visualizar a memória *heap,* etc (LECHETA,2009).

2.1.6 – *Android* SDK

O *Android* SDK é o software utilizado para desenvolver aplicações no *Android*, que tem um emulador para simular o celular, ferramentas utilitárias e uma API completa para a linguagem *Java*, com todas as classes necessárias para desenvolver as aplicações. Existe um *plug-in* para o *Eclipse* que visa justamente integrar o ambiente de desenvolvimento *Java* como o emulador (LECHETA,2009).

A Figura 03 ilustra o emulador do *Android*.

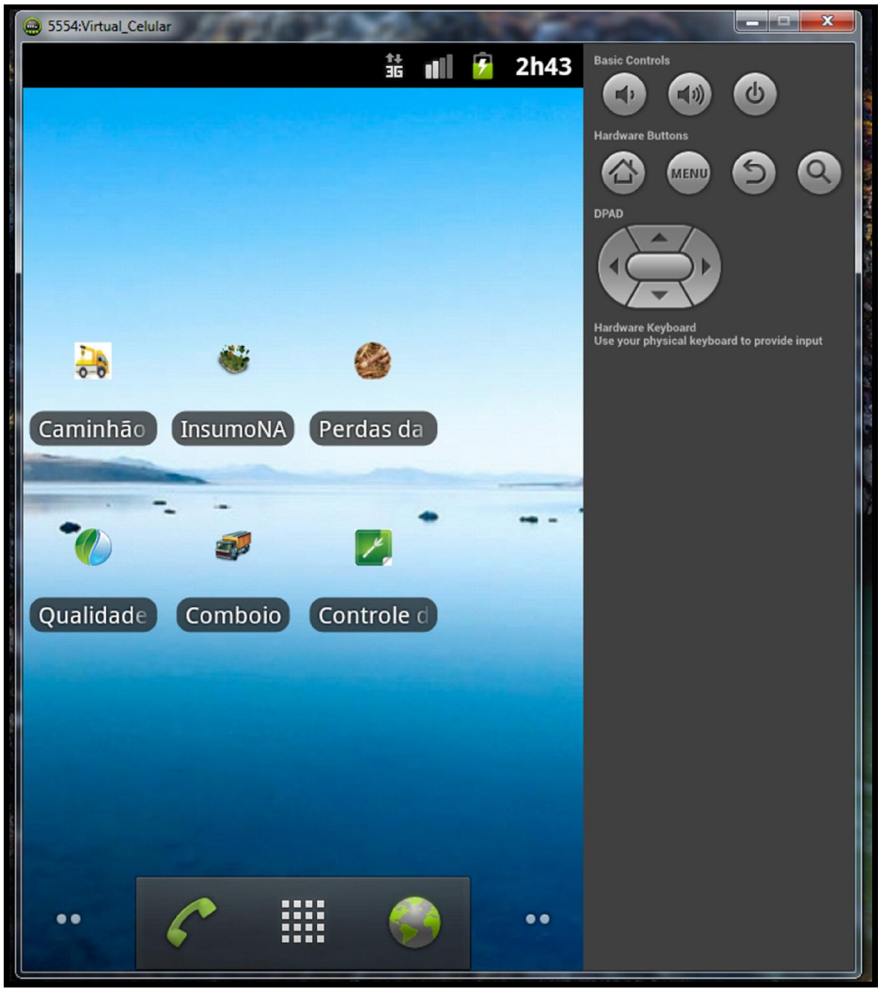

Figura 3 – Emulador Android

2.2 – PLATAFORMA GOOGLE ANDROID

A Figura 04, podemos ver como é a estrutura do Google *Android*.

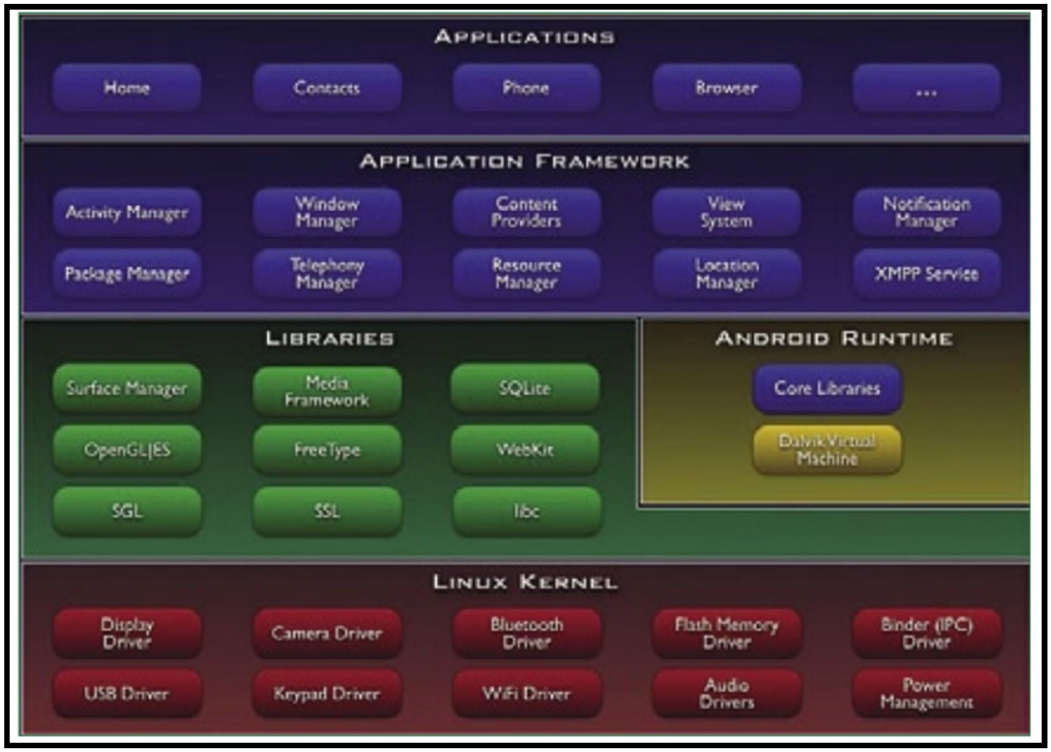

Figura 4 – Estrutura do Google *Android (In: Linux Magazine, 43)*

2.2.1 – Camada de Aplicações

Acima de todas as outras camadas está a de aplicativos, na qual se encontram todos os aplicativos fundamentais (escritos em *Java*) do *Android*, um cliente de e-mail, mapas, navegadores, calendários, programas de SMS, gerenciador de contatos, agendas, entre outros que serão desenvolvidos pela comunidade (PEREIRA; SILVA,2009).

2.2.2 – Camada Frameworks

Na camada do *framework* (*Application Framework*), encontramos todas as APIs e os recursos utilizados pelos aplicativos, como classes visuais, que incluem listas, grades, caixas de textos, botões e até um navegador web embutido, *View System* (componentes utilizados na construção de aplicativos), provedor de conteúdo (*Content Provider*), que possibilita que a aplicação possa acessar informações de outra aplicações, ou até mesmo compartilharem as suas informações, possibilitando a troca de informação entre aplicativos e gerenciador de localização (GPS, e *Cell* ID). Dentre os principais elementos desta camada temos:

- *ACTIVITY MANAGER*: gerencia o ciclo de vida de todas as *activities*, quando iniciar e quando terminá-las, possibilitando o deslocamento de uma *activity* para outra e assim por diante;
- *PACKAGE MANAGER*: é utilizada pelo *activity manager* para ler as informações dos APK's (Pacotes de arquivos do *Android*). A *package manager* se comunica com o resto do sistema e diz quais os pacotes estão sendo utilizados no dispositivo e quais são capacidades deste pacotes;
- *WINDOWS MANAGER*: basicamente, gerencia as apresentações de janelas, qual a janela estará ativa e assim por diante;
- CONTENT PROVIDERS: possibilita o compartilhamento de dados entre os aparelhos, possibilitando a troca de informações entre os aplicativos;
- *VIEW SYSTEM:* disponibiliza todo o tratamento gráfico para a aplicação, como botões, layouts e frames.

Todos os conjuntos de componentes disponíveis na plataforma *Android* possuem pleno acesso às APIs utilizadas pelo núcleo da plataforma. Esta arquitetura foi projetada para simplificar a reutilização de componentes, qualquer aplicação pode publicar as suas capacidades e qualquer outra aplicação candidata pode, então, fazer o uso dessas capacidades (sujeito a restrições de segurança aplicada pelo quadro) (PEREIRA; SILVA,2009).

2.2.3 – Camada de Bibliotecas

Essa camada carrega consigo um conjunto de bibliotecas C/C++ utilizadas pelo sistema, incluídas nesse conjunto da biblioteca C padrão (Libc), e possui ainda, também aquelas bibliotecas das áreas multimídia, visualização de camadas 2D e 3D, funções para navegadores web, funções para gráficos, funções de aceleração de hardware, renderização 3D, fontes bitmap e vetorizadas e funções de acesso ao banco *SQLite*. Todos esses recursos estão disponíveis no *framework* para o desenvolvimento de aplicativos.

- *WEBKIT:* o mesmo mecanismo de código-fonte aberto por trás dos navegadores Safari do Mac e Safari *Mobile* do *IPhone*, tornou se padrão para as maiorias das plataformas móveis;
- *SQLite:* uma poderosa *engine* de banco de dados relacional, implementada em C, leve e embutida, com suporte à base de dados acima de 2 *terabaytes*. Implementa a maioria do SQL92*;*
- *SGL:* responsável pelos gráficos 2D;
- *SURFACE MANAGER:* fornece o acesso aos subsistemas de exibição, como as camadas de aplicações 2D e 3D;
- *MEDIA LIBRARIES:* suporte para os mais populares formatos de áudio e vídeo, e também *hardware* e *software* de *plug-ins* de *codec*;
- *LIBWEBCORE:* um *web browser engine* utilizando tanto no *Android Browser* quanto para exibição web;
- *3D LIBRARIES:* uma implementação baseada nas APIs do *OpenGL* ES 1.0 APIs; as bibliotecas utilizam aceleração 3D via *hardware* (quando disponível) ou o *software* de renderização 3D altamente otimizado incluído no *Android* (PEREIRA; SILVA,2009).

2.2.4 – Camada Runtime

Pacotes básicos *Java* para um ambiente quase completo de programação e a máquina virtual *Dalvik*, que emprega os serviços do *kernel Linux* (Ableson; Sen; King; Ortiz, 2012).

2.2.5 – Camada Linux Kernel

Utiliza a versão 2.6 do *kernel* do *Linux* para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, pilha de protocolos de rede e modelo de drives. O *kernel* também atua como uma camada de abstração entre *hardware* e o resto da pilha de *software*. Um acessório interessante é o *Binder* (IPC), que é responsável por obter e enviar para aplicação requerente a interface de serviço da aplicação requerida, possibilitando a comunicação Inter-processos (IPC)

2.3 – TECNOLOGIA *JAVA*

Java é uma linguagem de programação orientada a objetos, utilizada em multiplataformas, tendo sua primeira versão oficial no ano de 1995, desde então vem sofrendo muitos aperfeiçoamentos e adaptações para evitar novos problemas da programação moderna.

2.3.1 – Histórico da Linguagem

Java é uma linguagem de programação de uso geral desenvolvida pela Sun Microsystems, empresa americana fundada em 1982. Extremamente rica e poderosa, a linguagem *Java* alcançou grande expressividade no cenário de programação, tornando-se uma das principais soluções nessa área. Hoje, *Java* se

apresenta como uma plataforma de desenvolvimento completa, sento utilizada na produção de programas corporativos, jogos on-line, de processamento cientifico, de programas educacionais, de aplicações multimídia e de programas em rede, apenas para citar alguns exemplos (COSTA,2008).

A primeira versão oficial da linguagem foi lançada em 1995. Desde então houve muitos aperfeiçoamentos e adaptações, o que vem culminando em uma linguagem de programação madura e adaptada aos novos problemas da programação moderna (COSTA,2008).

Uma razão para a popularidade de *Java* é a sua independência de plataforma. Isto significa que o mesmo programa compilado pode executar em qualquer computador. Esta independência torna o *Java* diferente da maioria das outras linguagens de programação, que exigem diferentes compiladores para diferentes sistemas operacionais (Hubbard,2004).

O interpretador da linguagem está embutido em um software chamado JVM (*Java Virtual Machine*). Como há uma JVM especifica para cada plataforma, um mesmo programa compilado em *Java* pode ser executado em plataformas diferentes, como *Windows* e *Linux* (COSTA,2008).

2.3.2 – Introdução à *Java*

A programação orientada a objetos (OOP) é uma metodologia de desenvolvimento de software em que um programa é percebido como um grupo de objetos que trabalham juntos. Os objetos são criados com modelos, chamados classes, e contém os dados e as instruções necessária para usar esses dados. *Java* é completamente orientada a objetos (CADENHEAD; LEMAY,2005).

Neutralidade de plataforma é a capacidade de um programa executar sem modificações em diferentes ambientes de computação. Os programas *Java* são compilados para um formato chamado *bytecode*, que é executado por qualquer sistema operacional, software ou dispositivo com um interpretador *Java*. Você pode criar um programa *Java* em uma máquina *Windows* XP, que será executado em um

servidor *Web Linux*, *Apple* Mac usando OS X e assistente digital *Palm*. Desde que uma plataforma tenha um interpretador *Java*, ela pode executar o *bytecode* (CADENHEAD; LEMAY,2005).

Mas o que é exatamente o *Java*? Mais que uma linguagem de programação, *Java* é uma completa plataforma de desenvolvimento e execução. Esta plataforma é composta de três pilares: máquina virtual *Java* (JVM), um grande conjunto de APIs e a linguagem *Java* (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPPAT; KUNG,2012).

2.3.3 – Características-Chave do *Java*

A Plataforma *Java* possui muitas característica-chave desde sua criação. Alguns deles serão apresentados nesse tópico.

- SIMPLES: a sintaxe do *Java* é, de fato uma versão limpa da sintaxe do C++. Não há necessidade de arquivos de cabeçalho, aritméticos de ponteiro (ou mesmo uma sintaxe de ponteiro), estruturas, uniões, sobrecarga de operadores, classes básicas virtuais e assim por diante.
- ORIENTADO A OBJETOS: a orientação a objetos provou seu valor nos últimos 30 anos e é inconcebível que uma linguagem de programação moderna não utilize. Ba verdade, os recursos orientados a objetos do *Java* e o C++ são a herança múltipla, que o *Java* substituiu pelo conceito mais simples das interfaces, e o modelo de metaclasse do *Java*.
- COMPATIBILIDADE COM REDES: achamos que a capacidade de rede do *Java* são fortes e fáceis de usar. Qualquer um que tentou fazer programação para Internet utilizando outra linguagem vai adorar a maneira como o Java simplifica tarefas insuportáveis de invocação de método remoto permite a comunicação entre objetos distribuídos.
- ROBUSTEZ: esse recurso também é muito útil. O compilador do *Java* detecta vários problemas que, em outras linguagens, só viriam à tona em tempo de execução. Quando ao segundo ponto, qualquer um que passou horas

procurando corrupção de memória causada por um bug de ponteiro ficará satisfeito com esse recurso do *Java*.

- SEGURO: desde do início, o *Java* foi projetado para tornar certos tipos de ataques impossíveis, como estouro de memória, corrupção de memória e leitura ou gravação de arquivos sem permissão.
- NEUTRO EM REALAÇÃO ARQUITETURA: o compilador gera um formato de arquivo de objeto neutro em relação à arquitetura - o código compilado é executável em muitos processadores, dada a presença do sistema em tempo de execução do *Java*. O compilador *Java* faz isso gerando instruções *bytecode* que não tem nada a ver com uma arquitetura de computação especifica. Em vez disso, elas são projetadas para ser fáceis interpretadas em qualquer máquina e instantaneamente convertidas em código de máquina nativo.
- PORTÁVEL: diferentemente do C e C++, não há nenhum aspecto" dependente de implementação" da especificação. Os tamanhos dos tipos de dados primitivos estão especificados, assim como o comportamento da aritmética neles. Ter um tamanho fixo de tipos de numéricos elimina dor de cabeça básica de alocação de portas. Dados binários são armazenados e transmitidos em um formato fixo, elimina a confusão em relação aos ordenamento de bytes. *Strings* são salvas em um formato Unicode padrão.
- INTERPRETADO: o interpretador do Java pode executar *bytecodes* Java diretamente em qualquer máquina em que o interpretador for portado. Como a vinculação é um processo mais incremental e leve, o desenvolvimento pode ser muito mais rápido e exploratório. Hoje, os *bytecodes* são convertidos de código de máquina pelo compilador *just-in-time*.
- MÚLTIPLOS THREADS: os benefícios do multithreading são melhor responsividade interativa e comportamento em tempo real.
- DINÂMICO: de vários modos, o *Java* é uma linguagem mais dinâmica do que o C ou C++. Ele foi projetado para adaptar-se a um ambiente em evolução. As bibliotecas podem adicionar livremente novos métodos e variáveis de

instancia sem nenhum efeito seus clientes. No *Java*, descobrir informações sobre tipos em tempo de execução é simples (HORSTMANN; CORNELL,2010).

2.3.4 – Máquina Virtual Java

O conceito de máquina virtual não é exclusividade do *Java*. As *Hardware Virtual Machines*, como *VMWares*, *Parallels* ou *VirtualBox*, também muito famosas, são usadas para rodar vários sistemas operacionais ao mesmo tempo, utilizando o recurso de virtualização que abstrai o hardware da máquina. Já a JVM, *Java Virtual Machine* (Máquina Virtual *Java*), é uma *Application Virtual Machine*, que abstrais não só a camada de hardware como a comunicação com o Sistema Operacional.

Ao usar o conceito de máquina virtual, o *Java* elimina o problema da portabilidade do código executável. Em vez de gerar um executável especifico, como "*Linux* PPC" ou "*Windows* i386", o compilador *Java* gera um executável para uma máquina genérica, virtual, não física, a JVM.

JVM é uma máquina completa que roda em cima da real. Ela possui suas próprias instruções de máquina (*assembly*) e suas APIs. Seu papel é executar as instrução de máquina genéricas no Sistema Operacional e no hardware especifico sob o qual estiver rodando (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPPAT; KUNG,2012).

As figuras abaixo mostra a diferença entre os códigos da linguagem C e Java.

A Figura 05, podemos ver o processo de compilação de uma aplicação em C.

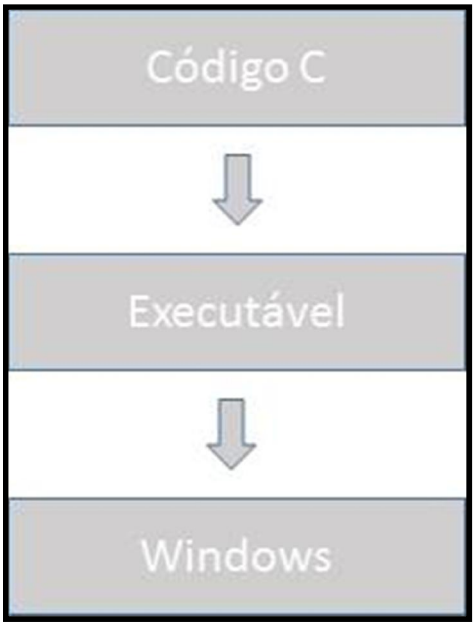

Figura 5 – Compilação do Código C (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPPAT; KUNG,2012).

A Figura 06, podemos ver o processo de compilação de uma aplicação em *Java*.

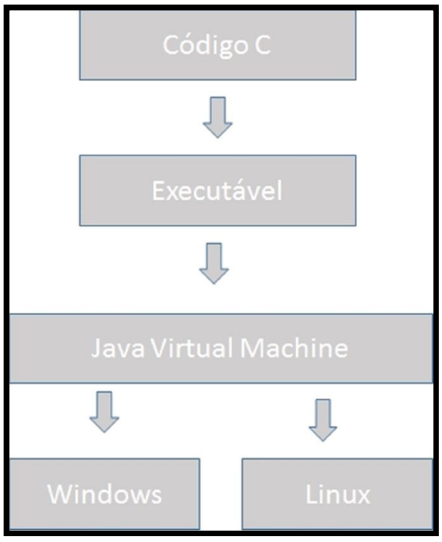

Figura 6 – Compilação do Código *Java* (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPPAT; KUNG, 2012).

2.3.5 – Os *Bytecodes Java*

A JVM é, então, o ponto-chave para a portabilidade e a performance da plataforma Java. Essas instruções de máquina virtual são equivalentes aos mnemônicos (comandos) do *assembly*, com diferença de seu alvo é uma máquina virtual. O nome *bytecode* vem do fato de que cada *opcode* (instrução) tem o tamanho de um *byte* e, portanto, a JVM tem capacidade de rodar até 256 *bytecodes* diferentes (embora o *Java* 7 possua apenas 205). Isto faz dela uma máquina teoricamente simples de se implementar e de rodar até em dispositivos com pouca capacidade (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPPAT; KUNG,2012).

2.3.6 – *Java Enterprise Edition*

Como o próprio *Bryan Basham* disse em seu prefácio, o *Java* EE surgiu a partir das necessidades das aplicações corporativas, visando mover grande parte da complexidade acidental para as camadas de infraestrutura do *framework*, tornando as aplicações mais leves e, ao mesmo tempo, mais robustas. Devida a esta característica, ele se estabeleceu com um padrão de fato para aplicações corporativas. Embora exista concorrência, principalmente do mundo “LAMP” (*Linux*-*Apache*-MySQL-PHP), ainda é a plataforma mais estável, robusta e flexível para qualquer tipo de aplicação.

O *Java Enterprise Edition* traz muitas novidades, sendo algumas:

- *PROFILES* (perfis): subconjuntos de tecnologias *Java* EE, voltados para fins específicos. O *framework* vem com o *Web Profile*, voltado para criação de *Websites* leves e funcionais.
- *SERVLET* 3.0: nova versão da especificação que faz uso intensivo de anotações, dispensando o famigerado arquivo “*web*.xml”.

- *RESTFUL WEB SERVICES*: a API JAX-RS 1.1 permite a criação de *web Services* usando o estilo arquitetural *REST*, que é mais simples de usar que o SOAP.
- EJB 3.1 E EJB *LITE*: nova versão da especificação com vários novidades, como processamento assíncrono, dispensa de *interfaces* e facilidade de desenvolvimento.
- JSF 2.0: desde que foi lançado, o JSF vem revolucionando a maneira de desenvolver *Websites* em *Java*. Agora, com *Facelets* incorporado, torna-se mais fácil de usar.
- CDI: *Contexts and Dependency Injection for Java* EE. Esta tecnologia permite desacoplar implementações de abstrações, deixando a cargo do Container a tarefa de inserir automaticamente os recursos necessitados pela aplicação.
- *BEAN VALIDATION*: em vez de "poluir" suas classes de modelo com código de validação, você pode definir metadados (anotações ou XML) para descrever as regras de validação de entidades;
- *WAR PACKAGING*: não é mais necessário criar um EAR só para distribuir um *Website*. Agora podemos empacotar vários elementos dentro de um arquivo WAR (SAMPAIO,2011).

2.3.7 – *Javasever Faces*

JAVASERVER FACES, ou JSF, é a tecnologia recomendada para o uso na camada *web*. Ela provê inúmeros benefícios, como o desacoplamento de apresentação e lógica de negócios, além de um modelo de componentes muito interessante.

O JSF é muito mais do que um "organizador" de sites como o *Struts*. Ele torna o desenvolvimento web uma atividade realmente profissional, ao invés de artesanal.

De acordo com site da *Oracle* (*java*.sun.com), o JSF é:

- Um modelo de programação *web* orientados a eventos;

- Um conjunto de APIs para representação de componentes visuais e gerenciamento de seu estado, com manipulação de eventos e validação de entrada de dados;
- Um conjunto de APIs para definição e controle de navegação de páginas;
- Um conjunto de tags para uso em páginas JSP ou XHTML (*Facelets*) para representar os componentes visuais (SAMPAIO,2011).

A Figura 07, podemos ver os elementos de uma aplicação JSF.

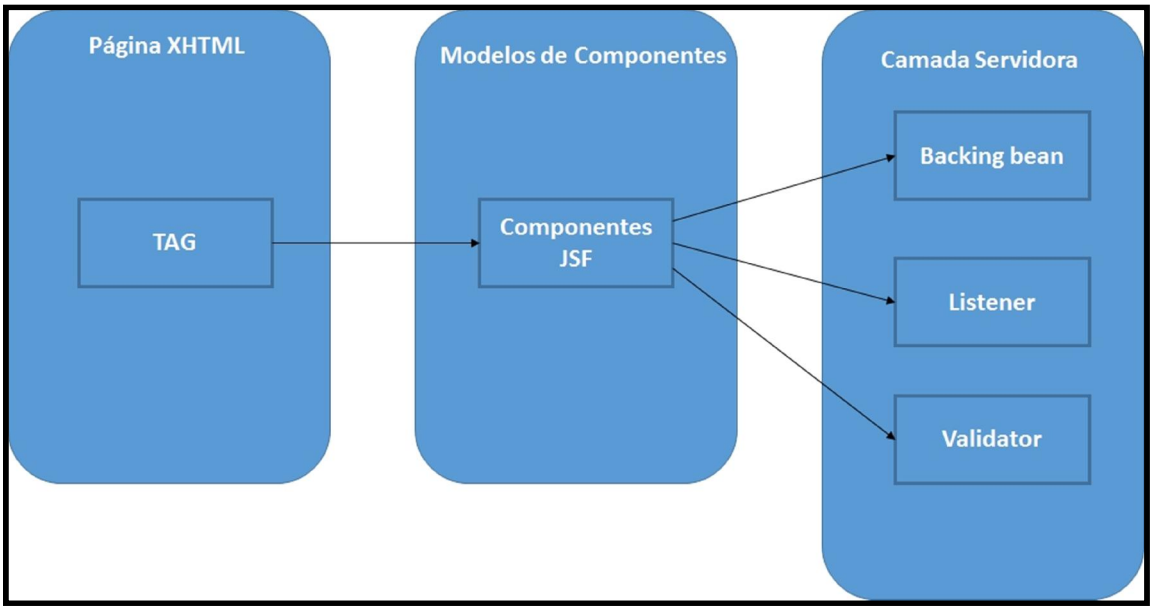

Figura 7 – Elementos de uma aplicação JSF (SAMAPIO, 2011).

2.3.8 – *Java Persistence*

A API de persistência do *Java* serve para desacoplar as aplicações dos meios de persistência, como bancos de dados relacionais, por exemplo, provendo uma camada uniforme que utiliza o padrão "*Domain Model*" (J2EE *patterns*).

Todo mapeamento objeto/relacional é feito de maneira simples e transparente para o desenvolvedor, ficando este apenas como o trabalho de lidar com as entidades envolvidas.

O *Java Persistence* é o *Entity*, um POJO (*Plain Old Java Object*) com algumas anotações. Ele pode ser utilizado por aplicações *Java* EE, *Java* SE e *Java* ME sem problemas, enquanto que o *Entity Beans* do EJB antigo só poderia ser utilizado por clientes J2EE (SAMPAIO,2011).

O JPA é um framework utilizado na camada de persistência para o desenvolver ter uma maior produtividade, com impacto principal num modo para controlarmos a persistência dentro Java. Pela primeira vez, nós desenvolvedores temos um modo "padrão" para mapear nossos objetos para o do Banco de Dados. Persistência é uma abstração de alto-nível sobre JDBC (DEVMEDIA,2014).

A Figura 08, Camada de persistência.

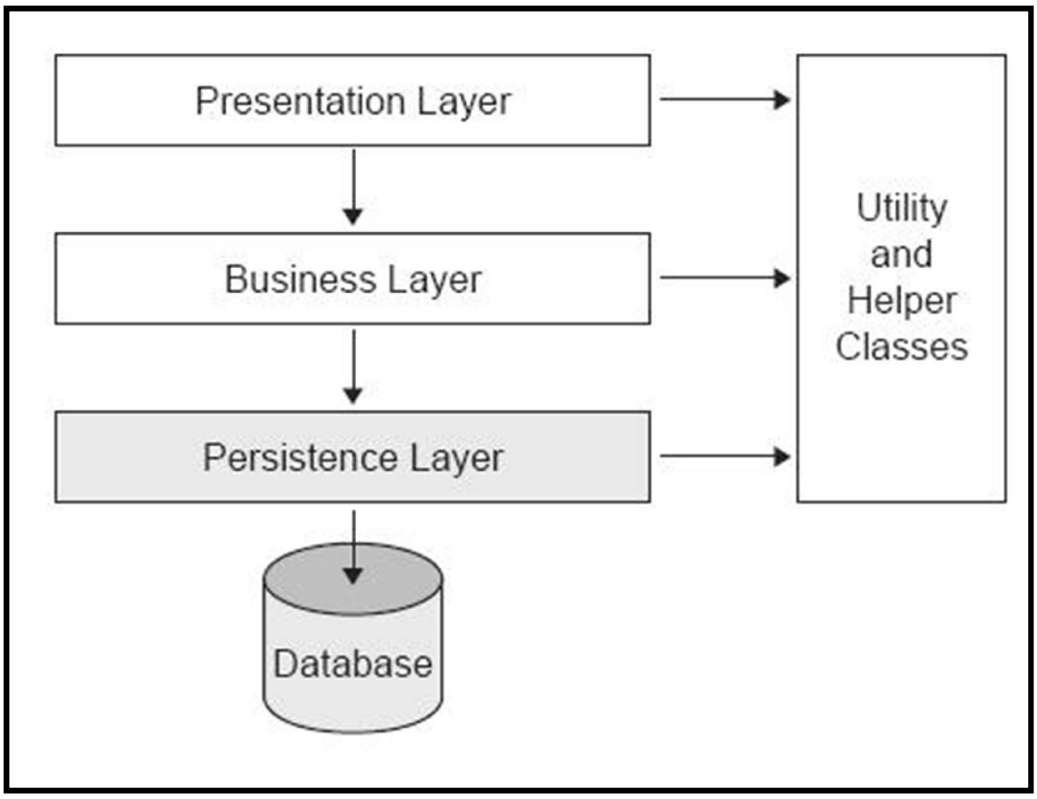

Figura 8 – Camada de Persistência (DEVMEDIA, 2014).

A nova *Java Persistence Specitication* define mapeamento entre o objeto *Java* e o banco de dados utilizando ORM, de forma que *Beans* de entidade podem ser portados facilmente entre um fabricante a outro (DEVMEDIA,2014).

2.3.9 – *Framework Primefaces*

PrimeFaces fornece um arsenal de componentes de interface rica em recursos com um alto grau de usabilidade, sofisticação, flexibilidade e interatividade. Estas características tornam o *PrimeFaces* uma tecnologia poderosa para desenvolvedores JSF que oferecem vantagens significativas sobre os componentes JSF padrão (HLAVATS,2013).

O *Primefaces* permite que sejam adicionados componentes ao conjunto mínimo por meio de uma especificação *JavaServer Faces* (JSF). Ele está organizado em três módulos: UI *Components*, *Optimus* e *Faces Trace*.

A Figura 09, Módulos do *PrimeFaces*.

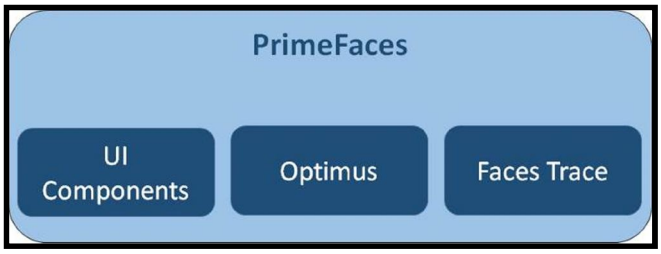

Figura 9 – Módulos do *PrimeFaces* (SANTOS,2013).

Ui Components compreende os componentes que as funcionalidades encapsulada de Ajax, Javascript e gráficos animados.

Optimus oferece soluções para facilitar desenvolvimento com JSF removendo o fardo da adoção de declaração XML e contém componentes de extensão de segurança.

Faces Trace é encarregado de melhorar a rastreabilidade das aplicações baseadas em JSF e coletar informações de métricas de desempenho. (SANTOS,2013).

2.4 – BANCO DE DADOS POSTGRESQL

O *PostgreSQL* é um *open-souce*, cliente/servidor, banco de dados relacional. *PostgreSQL* oferece uma combinação única de característica que comparam bem com os principais banco de dados comerciais, como Sysbase, *Oracle* e DB2. Umas das principais vantagens para o *PostgreSQL* é que ele é *open source*, você pode ver o código fonte do *PostgreSQL* (DOUGLAS,2005).

PostgreSQL não é propriedade de uma única empresa, é desenvolvido, mantido e fixado por um grupo de desenvolvedores voluntários ao redor do mundo. Você não terá que pagar para comprar o *PostgreSQL*, porem você encontrar fontes comerciais para suporte técnico (MOMJIAN,2001).

2.4.1 – Histórico do Banco de Dados Postgresql

PostgreSQL é um banco de dados relacional com uma longa história. No final de 1970, a Universidade da Califórnia em *Berrkeley* iniciou o desenvolvimento do ancestral banco de dados relacional do *PostgreSQL* conhecido como *Ingres*. Por volta de 1986, Michael *Stonebraker* da UC *Berkeley* liderou uma equipe que adicionou recursos orientados a objetos para o núcleo do *Ingres*, a nova versão ficou conhecida como *Postgres*, sendo comercializada pela empresa *Illustra*, que se tornou parte da *Informix Corporation*, Andrew Yu e Jolly Chen adicionaram suportes SQL para o *Postgres* no meado dos anos 90.

As versões anteriores tinham usado uma linguagem de consulta diferente, especifica do *Postgres* conhecido como *Postquel*. Em 1996, muitos recursos novos foram adicionados, incluindo o modelo MVCC de transação, mais aderência ao padrão

SQL92 e muito mais desempenho e melhorias, pois sua vez veio assumir o novo nome *PostgreSQL* (DOUGLAS,2005).

2.4.2 – Introdução ao Banco de Dados *Postgresql*

PostgreSQL certamente tem uma reputação, ele é conhecido por ter um rico conjunto de recursos e versões de software muito estáveis, posição firme da sua configuração padrão leve e elogiado por sua segurança (SMITH,2010).

Um dos benefícios mais claros do *PostgreSQL* é que ele é *open source*, o que significa que você tem uma licença para instalar, utilizar e distribuir, sem taxas de alguém ou royalties. Em cima disso, o *PostgreSQL* é bem conhecido como um banco de dados que fica por longos períodos sem requerer pouca ou nenhuma manutenção em muitos casos.

PostgreSQL tem as seguintes características principais como: excelente cumprimento SQL *Standards* até o SQL *Server* 2008; arquitetura cliente-servidor, projeto altamente concorrente onde os leitores e escritores não bloqueiam uns aos outros; altamente configurável e extensível para muitos tipos de aplicações; excelente escalabilidade e desempenho com amplos recursos de ajuste.

O *PostgreSQL* se concentra nos seguintes objetivos: software robusto e de alta qualidade com manutenção de código; baixa manutenção de administração; SQL compatíveis com os padrões, interoperabilidade e compatibilidade; desempenho, segurança e alta disponibilidade (SIMON; KROSING,2010).

2.4.3 – Principais Características do Banco de Dados Postgresql

PostgreSQL se tem beneficiado por sua longa história, sendo hoje um dos servidores de banco de dados mais avançados. Tendo algumas características em sua distribuição padrão:

- OBJETO-RELACIONAL: cada tabela define uma classe. *PostgreSQL* implementa herança entre tabelas, funções e operadores polimórficos;
- COMPATIBILIDADE: com normas do *PostgreSQL*, sintaxe implementa a maioria do padrão SQL92 e muitas características do SQL99;
- OPEN SOURCE: Uma equipe internacional de desenvolvedores mantém *PostgreSQL*, definido pelo menos desde 1996. É um produto que pode ser usado de forma produtiva em qualquer linguagem natural, apenas não em Inglês;
- TRANSAÇÃO: o processamento de dados do PostgreSQL protege e coordenar vários usuários simultâneos através de um processamento de transações completo. Baseado em MVCC (Multi Versão de Controle de Concorrência), fornecendo um melhor desempenho do que se encontraria com os produtos que coordenam vários usuários;
- INTEGRIDADE RELACIONAL: sendo apoiado por chave estrangeira e primaria em relações entre tabelas, bem como gatilhos. As regras de negócios podem ser expressas no banco de dados, em vez de depender de uma ferramenta externa;
- MULTI-PROCEDURAL: Triggers e outros procedimentos podem ser escrito em qualquer uma das várias linguagens procedurais. Sendo o código do lado do servidor escrito em PL/PGSQL, uma linguagem procedural semelhante ao da *Oracle* PL/SQL, também podendo ser desenvolvido em Tcl, Perl, mesmo *bash* (código fonte do *Linux*/*Unix Shell*);
- APIs MULTI-CLIENTE: suporta o desenvolvimento de aplicações cliente em muitas ferramentas diferentes entre elas C, C++, ODBC, *Perl*, PHP, Tcl, *Python* entre outras;
- TIPOS DE DADOS EXCLUSIVOS: fornece uma variedade de tipos de dados, além dos dados numéricos de costume, tem também, tipos de dados *Boolean*, geométricos, tipos de dados especificamente para lidar com endereços de rede;

- EXTENSIBILIDADE: uma das características mais importantes do *PostgreSQL* é que ele pode ser prorrogado. Você pode adicionar novos tipos de dados, novas funções e operadores, e até mesmo novas linguagens procedurais e cliente (DOUGLAS,2005).

3 – ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

Neste capítulo serão apresentadas as ferramentas utilizadas para analise, especificações de requisitos funcionais e a modelagem de diagramas para melhor entendimento do sistema.

3.1 – MAPA MENTAL

Os mapas mentais têm como principal objetivo manter coisas organizadas e destacam palavras-chave de uma ideia principal que, a partir de seu refinamento, acrescentam novos tópicos em forma de ramificações (KOLIFRATH, 2013).

A Figura 10 ilustra o mapa mental referente às funcionalidades e responsabilidades da aplicação a ser desenvolvida neste trabalho de conclusão de curso:

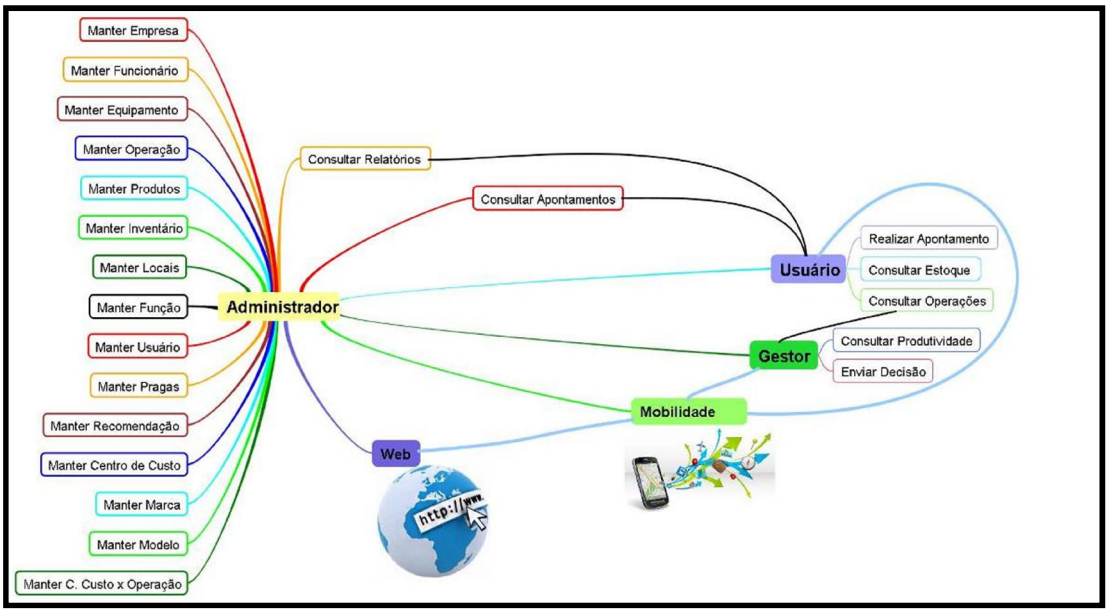

Figura 10 – Mapa Mental do Sistema.

O mapa mental apresentado fornece uma análise lógica sobre as funcionalidades do sistema.

3.2 – LISTA DE REQUISITOS

O sistema *Web* mencionado, se desta os seguintes requisitos:

- Cadastrar Empresa;
- Cadastrar funcionários;
- Cadastrar equipamentos;
- Cadastrar operações;
- Cadastrar produtos;
- Cadastrar inventários;
- Cadastrar locais;
- Cadastrar funções;
- Cadastrar pragas;
- Cadastrar recomendações;
- Cadastrar centro de custo;
- Cadastrar marca;
- Cadastrar modelo;
- Cadastrar centro de custo X operação;
- Realizar apontamentos de pragas, aplicação, plantio, colheita e entrega;
- Realizar consultas de apontamentos em geral;
- Gerar gráficos de apontamentos;
- Emitir relatórios de apontamentos, tomadas de decisão e outros.

O aplicativos *Mobile* mencionado, se desta os seguintes requisitos:

- Manter Configuração;
- Manter Usuário;
- Realizar Importação;
- Realizar apontamentos de pragas, aplicação, plantio, colheita e entrega;
- Realizar consultas de apontamentos em geral.

3.3 – DIAGRAMA E ESPECIFICAÇÃO DE CASO DE USO

Os casos de uso são uma técnica para captar os requisitos funcionais de um sistema. Eles servem para descrever as interações típicas entre os usuários de um sistema e o próprio sistema, fornecendo uma narrativa sobre como o sistema é utilizado. Os casos de uso são reconhecidos com uma parte importante da UML (FOWLER, 2005).

3.3.1 – Caso de Uso Sistema Web

Para melhor descrever foram elaborados uns diagramas de caso de uso do sistema web como parte da documentação.

A Figura 11 apresenta o Diagrama de Caso de Uso para entidade “Administrador”:

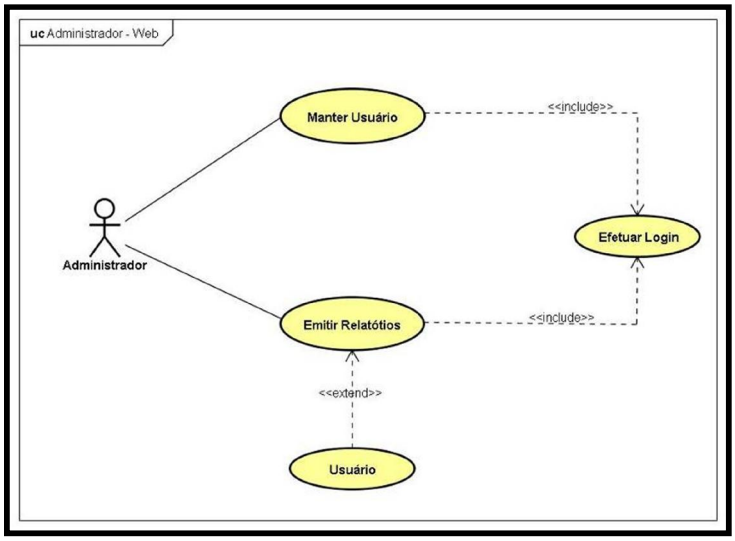

Figura 11 – Diagrama de Caso de Uso - Administrador.

A Figura 12 apresenta o Diagrama de Caso de Uso comum para as entidades “Administrador” e “Usuário” com a representação Manter:

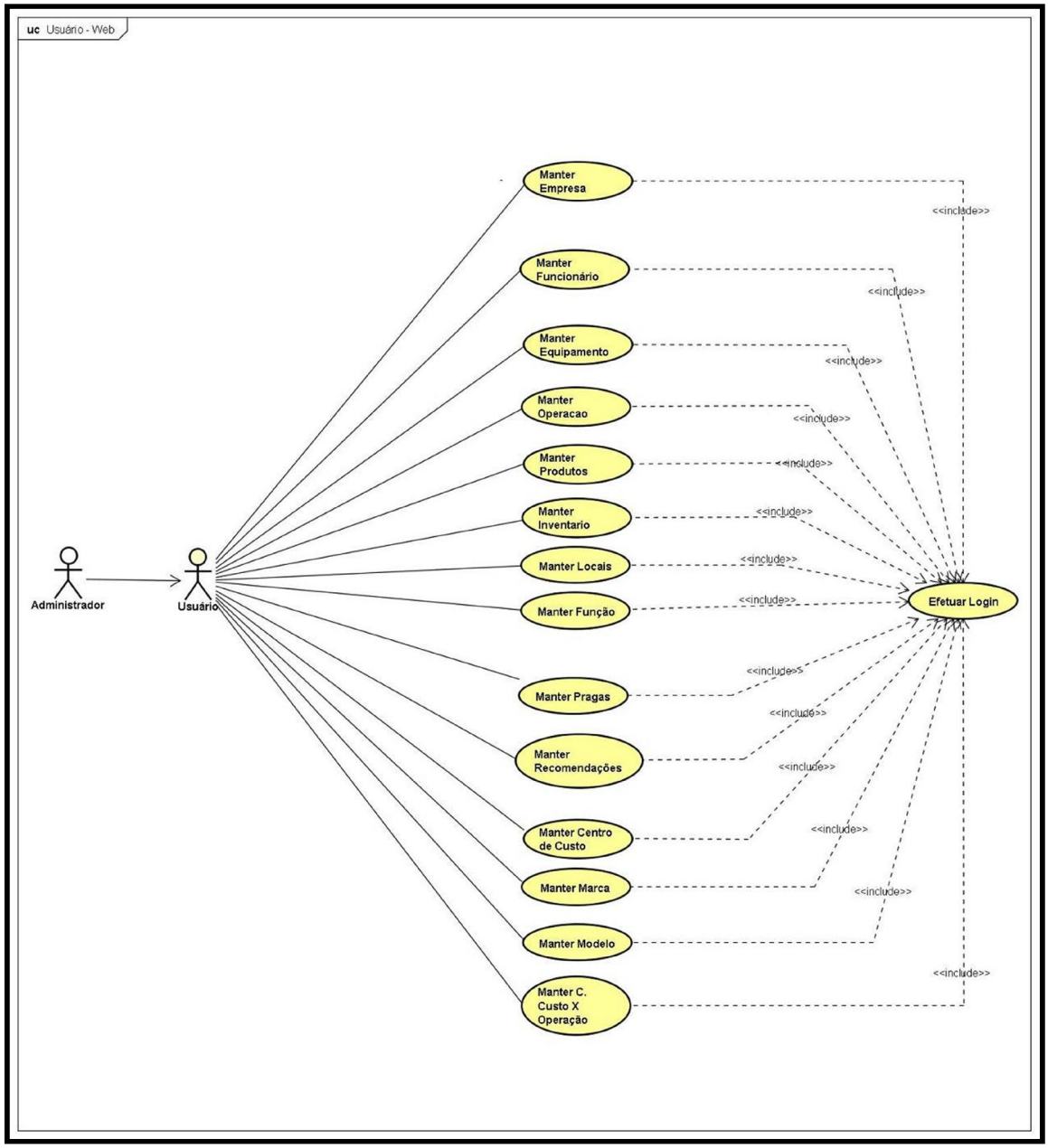

Figura 12 – Diagrama de Caso de Uso comum – Adm. e Usuário – "Manter".

A Figura 13 apresenta o Diagrama de Caso de Uso comum para as entidades "Administrador" e "Usuário" com a representação Apontar:

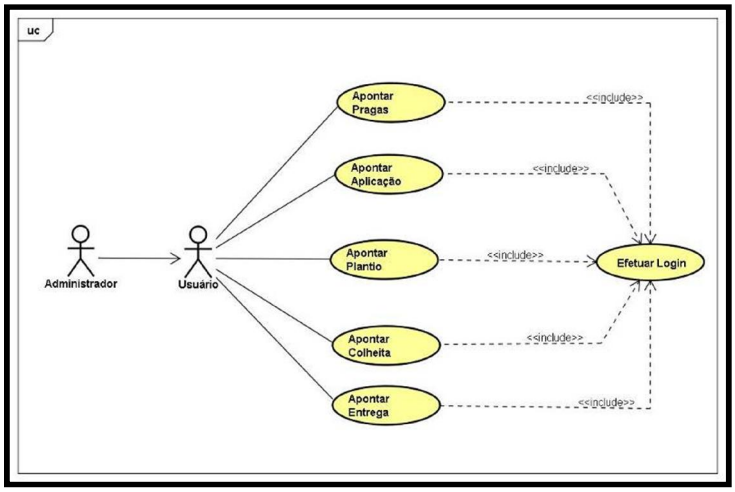

Figura 13 – Diagrama de Caso de Uso comum – Adm. e Usuário – "Apontar".

A Figura 14 apresenta o Diagrama de Caso de Uso comum para as entidades "Administrador" e "Usuário" com a representação Emitir:

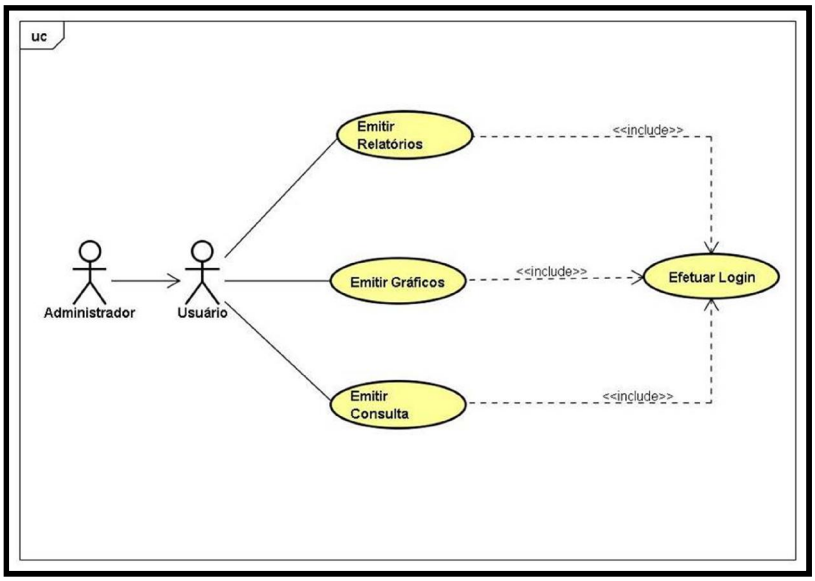

Figura 14 – Diagrama de Caso de Uso comum – Administrador e Usuário – "Emitir".

3.3.2 – Caso de Uso - Aplicativos *Mobile*

Os casos de usos abaixo especificam os aplicativos da mobilidade: Operação, Plantio, Infestação de Pragas, Aplicação de Produtos, Colheita, Entrega.

A Figura 15 apresenta o Diagrama de Caso de Uso da entidade "*Mobile* – Administrador":

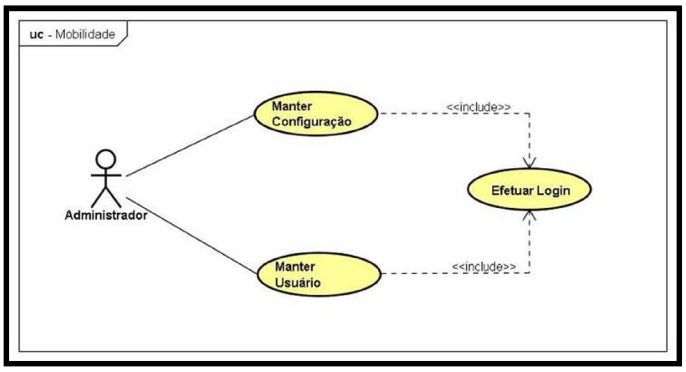

Figura 15 – Diagrama de Caso de Uso – *Mobile* – Administrador.

A Figura 16 apresenta o Diagrama de Caso de Uso da entidade de uso comum "Mobilidade – Administrador" e "*Mobile* – Usuário":

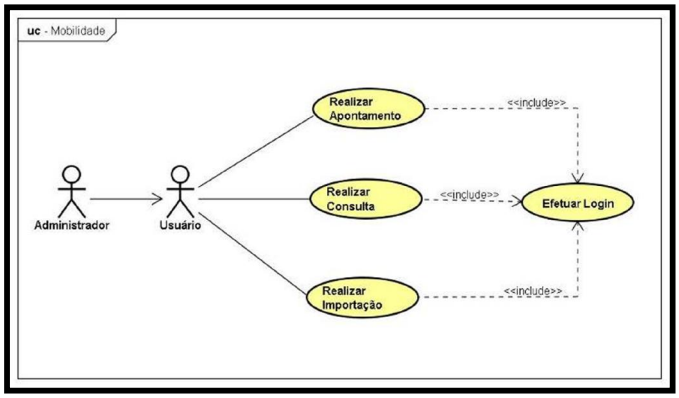

Figura 16 – Diagrama de Caso de Uso comum – *Mobile* Pragas – Adm. e Usuário – "Realizar".

3.4 – DIAGRAMA DE ATIVIDADE

Um diagrama de atividade UML complementa o caso de uso através de uma representação gráfica do fluxo de interação em um cenário especifico. Similar ao fluxograma, um diagrama de sistema, setas para representar o fluxo através do sistema, losangos de decisão para representar uma decisão com ramificação (cada seta saindo do losango é identificada e as linhas horizontais cheias indicam as atividades paralelas que estão ocorrendo) (PRESSMAN, 2011).

A Figura 17 apresenta o diagrama de atividades para o caso de uso do Sistema Web do Apontamento Infestação de Pragas – Administrador - Usuário:

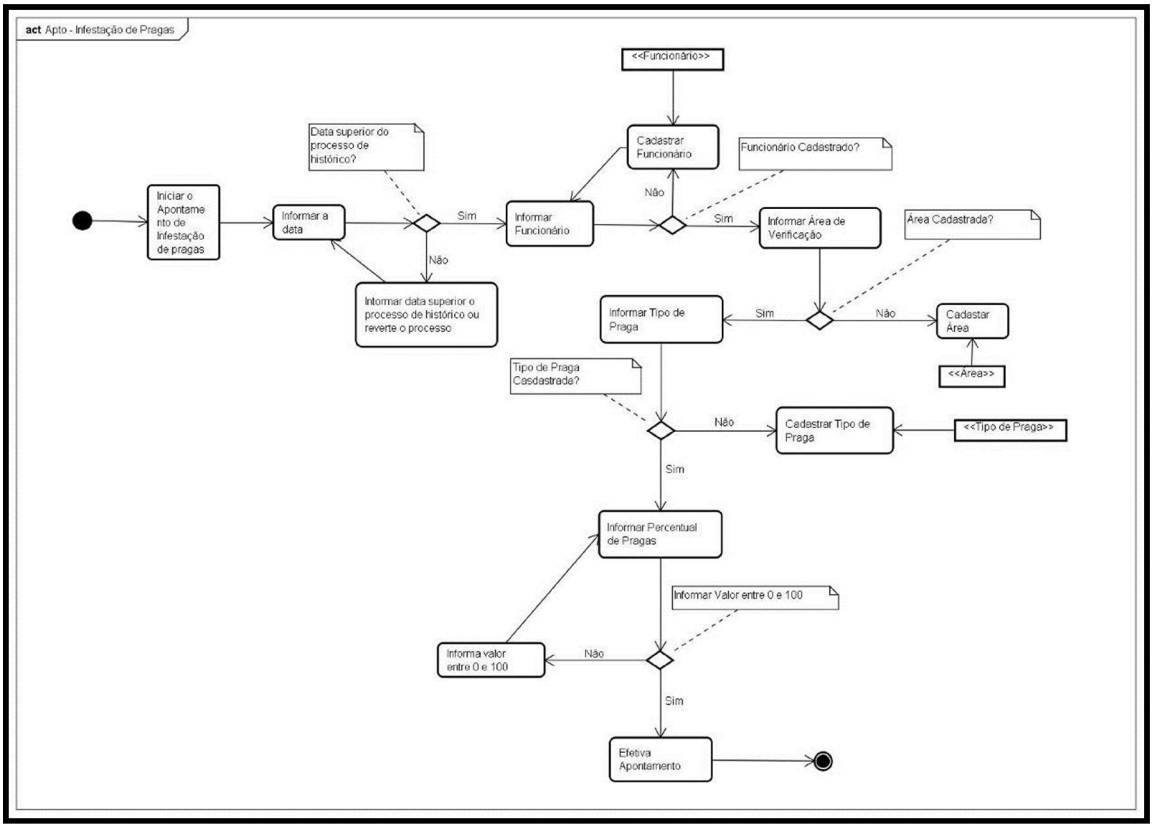

Figura 17 – Diagrama de Atividades do UC – Sistema *Web* – Apto. – Infestação de Pragas.

3.5 – DIAGRAMA DE SEQUENCIA

O diagrama de sequência mostra a troca de mensagens entre as classes na realização de uma caso de uso. É utilizado para mostrar como um cenário especifico de um caso de uso será implementado. O diagrama de sequência dá ênfase em mostrar as trocas de mensagens no decorrer do tempo (MARTINS, 2010).

A Figura 18 apresenta o diagrama de sequência para o caso de uso do Sistema Web do Apontamento Infestação de Pragas – Administrador - Usuário:

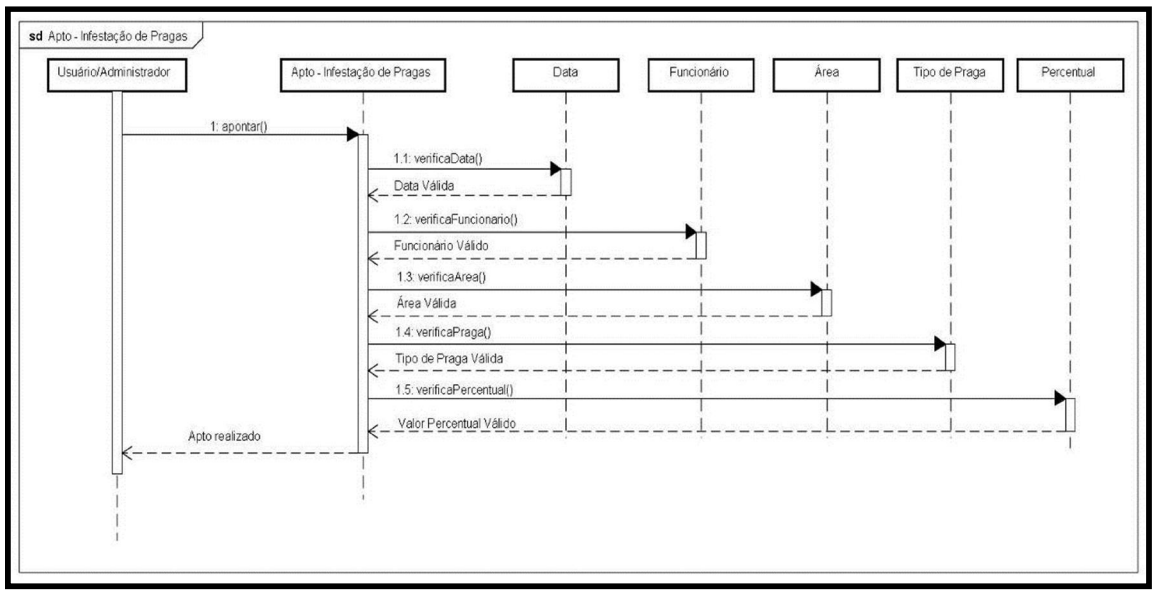

Figura 18 – Diagrama de Sequência do UC – Sistema *Web* – Apto. – Infestação de Pragas.

3.6 – DIAGRAMA DE CLASSE

Os diagramas de classe são os diagramas encontrado com maior frequência na modelagem de sistema orientados a objetos. Um diagrama de classe mostra um conjunto de classes, interfaces e colaboração e seus relacionamentos (BOOCH; RUMBAUGH; JACOBSON, 2006).

A Figura 19 apresenta o diagrama de classe do Aplicativo *Mobile* do Apontamento Infestação de Pragas – Administrador - Usuário:

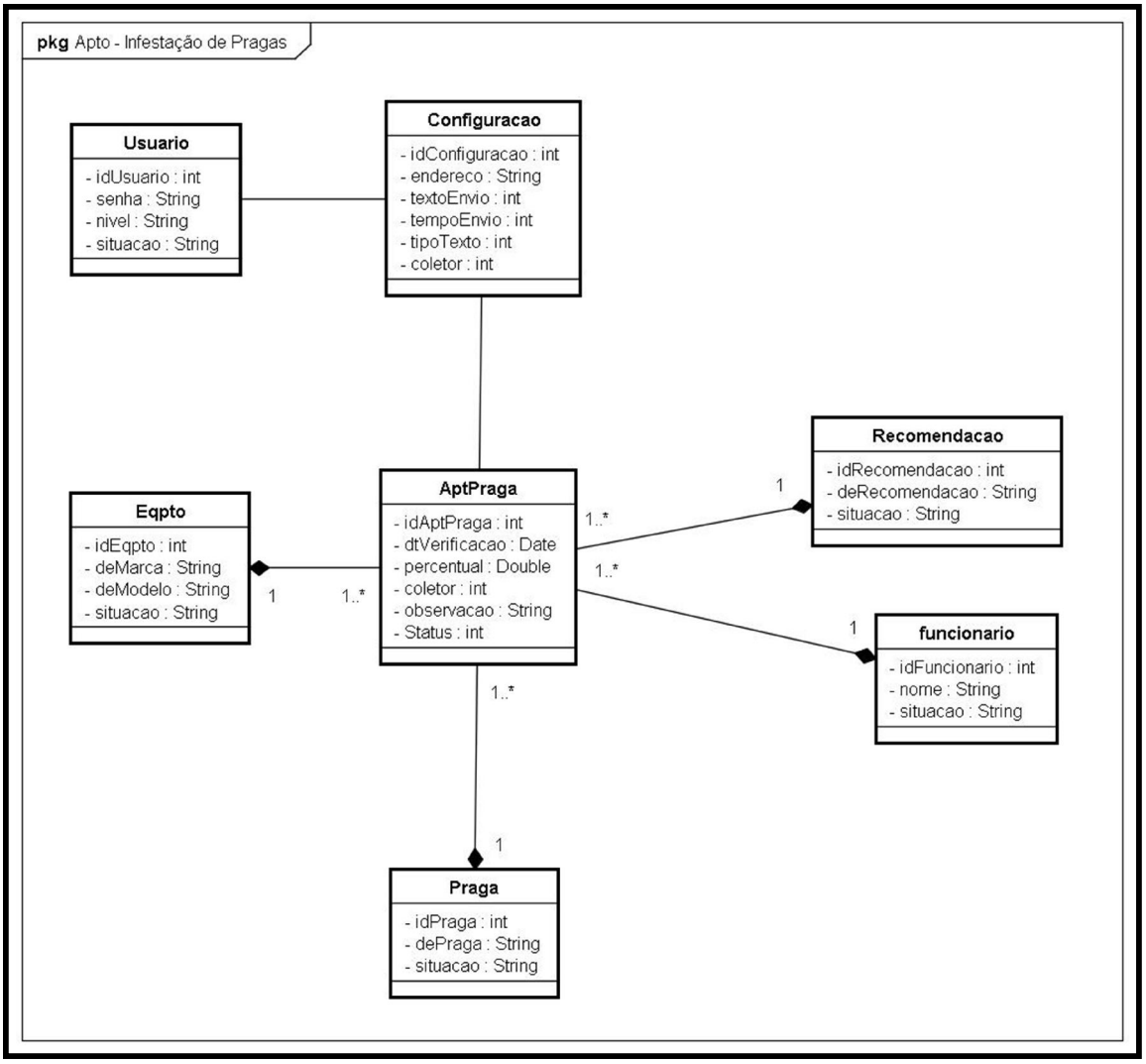

Figura 19 – Diagrama de Classe – Aplicativo *Mobile* – Apto. – Infestação de Pragas.

3.7 – MODELO – ENTIDADE-RELACIONAMENTO

A notação fundamental para a modelagem de dados conceitual é o Diagrama Entidade-Relacionamento. O principal propósito do DER é representar os objetos de dados e suas relações, sendo que cada entidade termina representada pelo menos por uma tabela de dados, e normalmente expressa um depósito de Dados do DFD (REZENDE, 2005).

A Figura 20 apresenta o Diagrama Entidade-Relacionamento do Sistema *Web*:

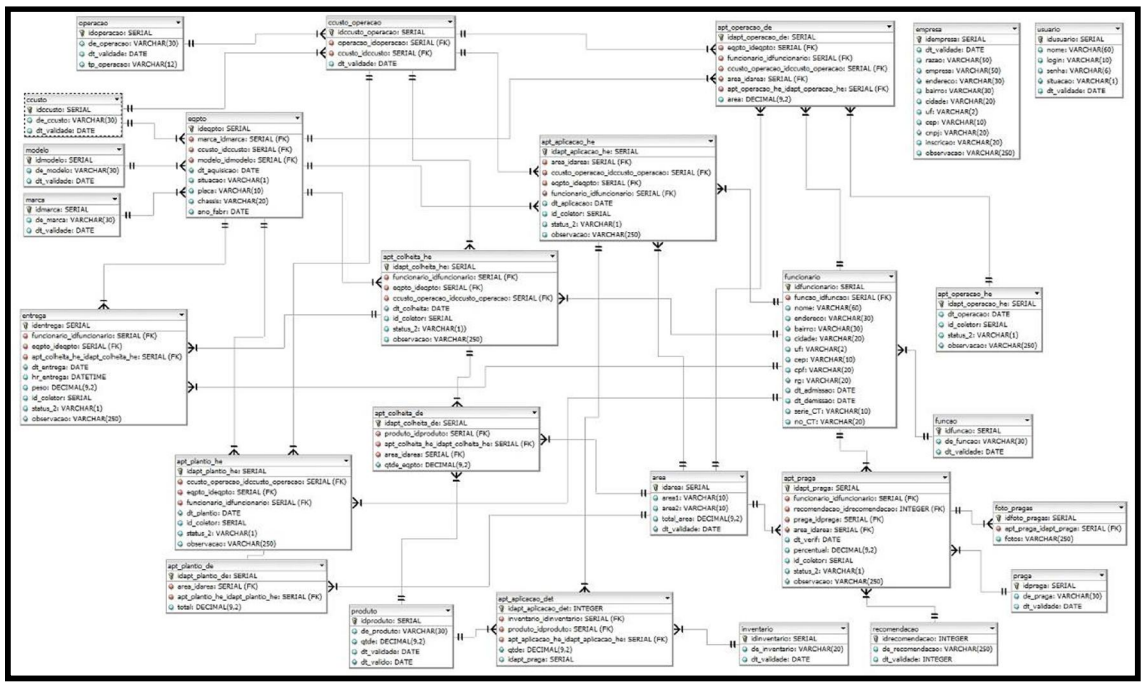

Figura 20 – DER - Sistema Web.

A Figura 21 apresenta o Diagrama Entidade-Relacionamento do Aplicativo *Mobile* – Apontamento de Operações Agrícola:

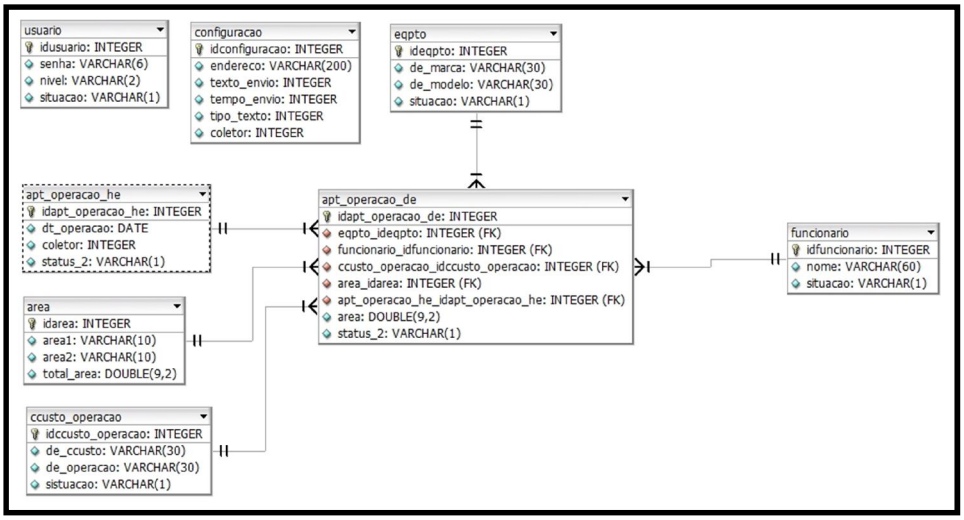

Figura 21 – DER – Aplicativo *Mobile* – Apontamento de Operações Agrícola.

A Figura 22 apresenta o Diagrama Entidade-Relacionamento do Aplicativo *Mobile* – Apontamento de Plantio:

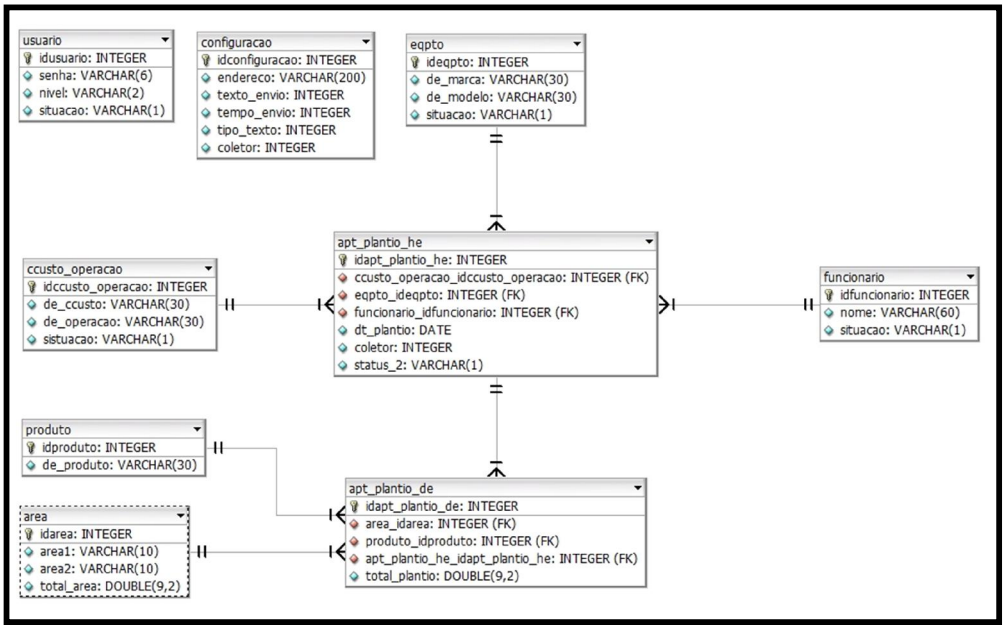

Figura 22 – DER – Aplicativo *Mobile* – Apontamento de Plantio.

A Figura 23 apresenta o Diagrama Entidade-Relacionamento do Aplicativo *Mobile* – Apontamento de Infestação de Pragas:

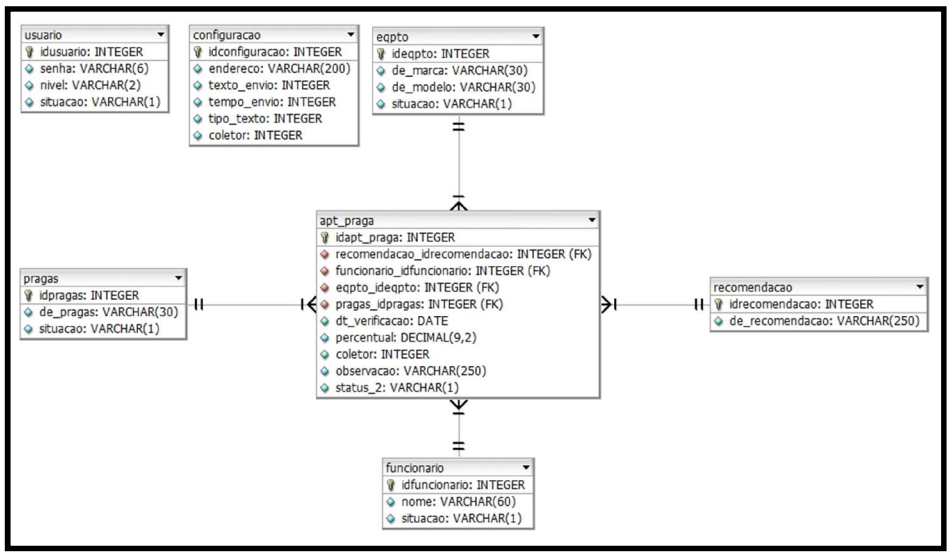

Figura 23 – DER – Aplicativo *Mobile* – Apontamento de Infestação de Pragas.

A Figura 24 apresenta o Diagrama Entidade-Relacionamento do Aplicativo *Mobile* – Apontamento de Aplicações de Produtos Agrícola:

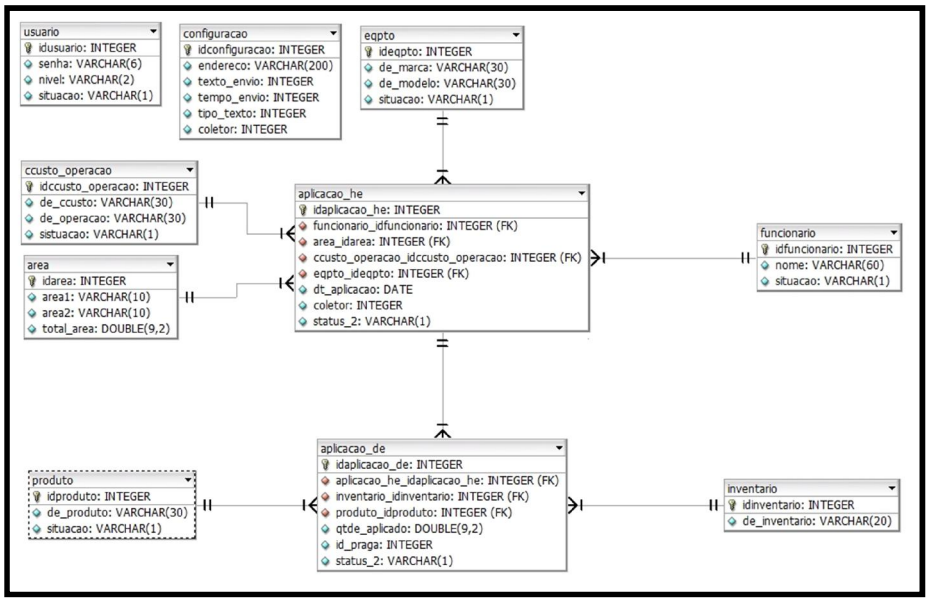

Figura 24 – DER – Aplicativo *Mobile* – Apto. de Aplicações de Produtos Agrícola.

A Figura 25 apresenta o Diagrama Entidade-Relacionamento do Aplicativo *Mobile* – Apontamento de Colheita:

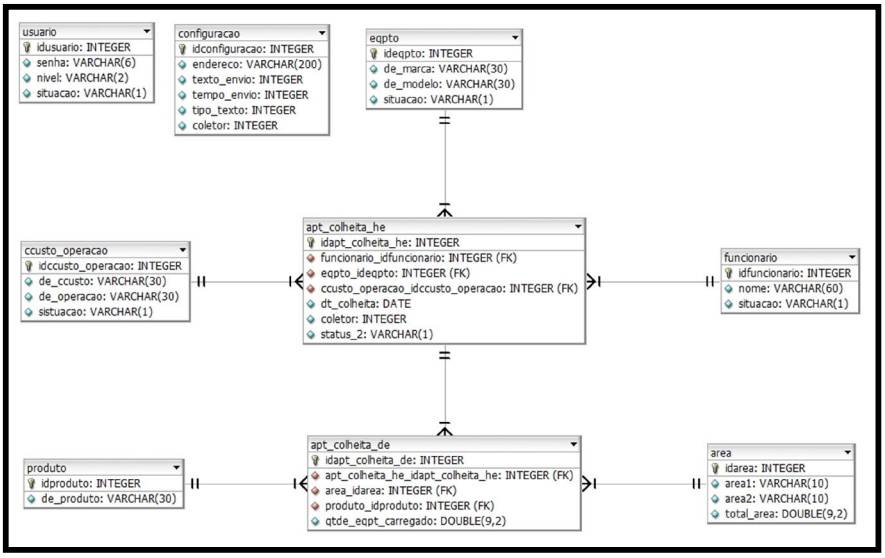

Figura 25 – DER – Aplicativo *Mobile* – Apontamento de Colheita.

A Figura 26 apresenta o Diagrama Entidade-Relacionamento do Aplicativo *Mobile* – Apontamento de Entrega:

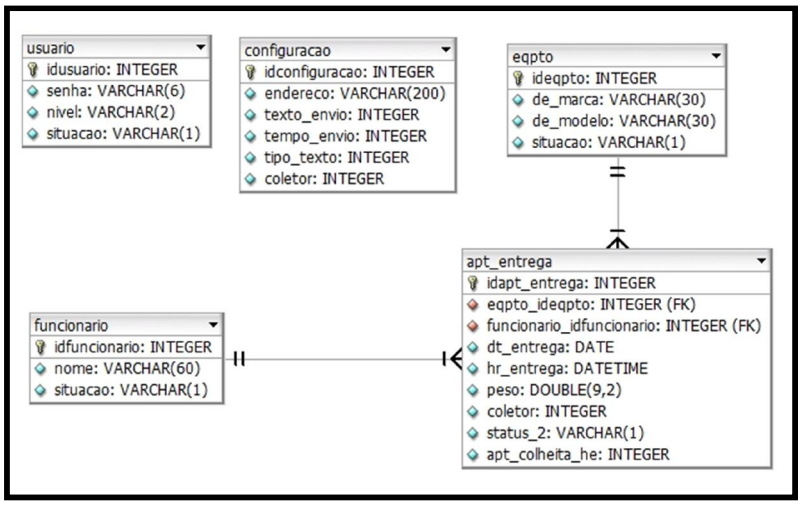

Figura 26 – DER – Aplicativo *Mobile* – Apontamento de Entrega.

4 – ESTRUTURA DO PROJETO

Nesse capitulo será apresentada a estrutura do projeto adotada para o desenvolvimento do trabalho de conclusão de curso, a qual consiste em fases e etapas. Será utilizado o diagrama WBS (*Work Breakdown Structure)* que é definido como o processo de subdivisão hierárquica do trabalho em componentes para gerenciamento mais fácil, tendo como objetivo principal identificar elementos terminais, como produtos, serviços e resultados a serem realizados em um projeto.

4.1 – ESTRUTURA ANALITICA DO PROJETO

A EAP, também conhecida como WBS, é a ferramenta de gerenciamento do escopo do projeto. Cada nível descendente do projeto representa um aumento no nível de detalhamento do projeto, como se fosse um organograma (hierárquico). O detalhamento pode ser realizado até no nível desejado, apresentado dados genéricos ou detalhados. O detalhamento mais usual é até o pacote de trabalho (*work package*) (VARGAS, 2007).

A Figura 27 apresenta a Estrutura Analítica do Projeto:

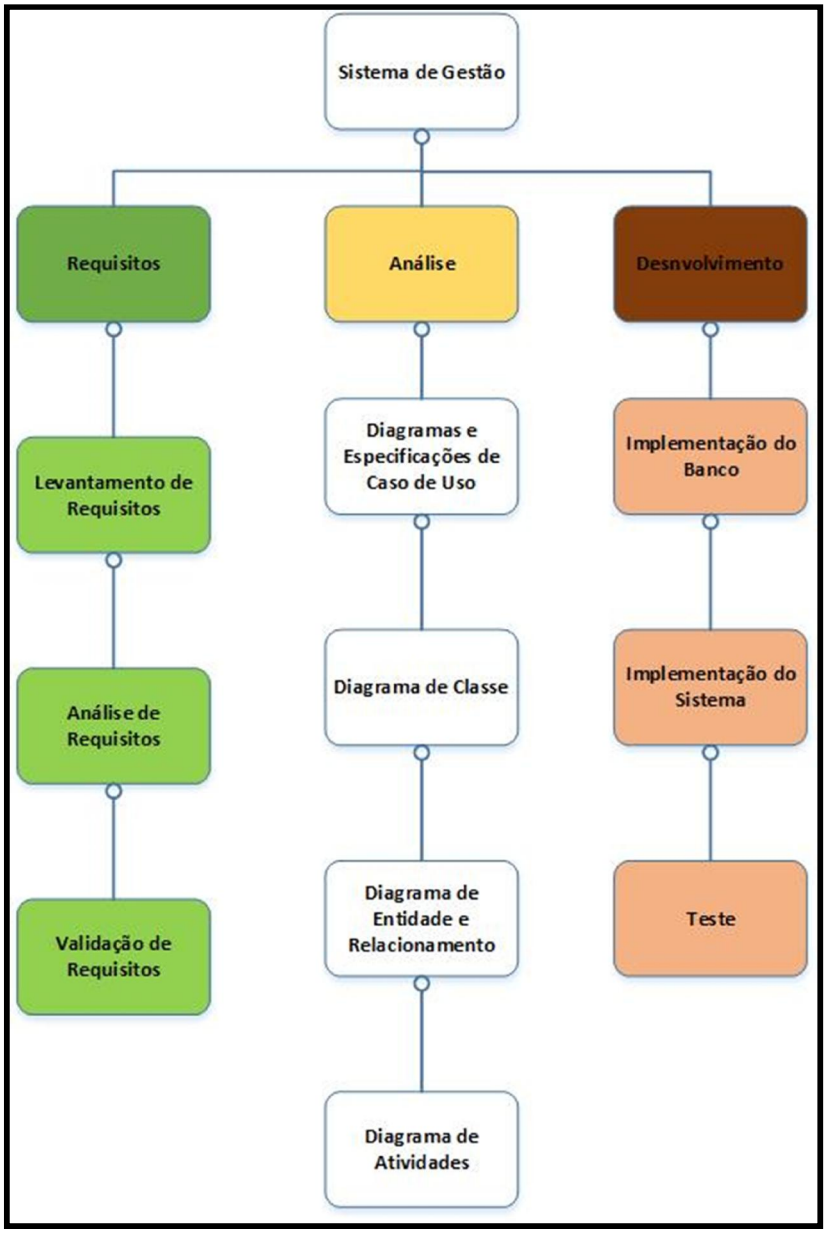

Figura 27 – Estrutura Analítica do Projeto.

.

4.2 – SEQUENCIAMENTO DAS ATIVIDADES

O diagrama de sequenciamento de atividades ilustra o tempo de duração para a realização de cada atividade do projeto em desenvolvimento, tendo como objetivo definir de forma lógica de execução das tarefas.

A Figura 28 apresenta o Sequenciamento de Atividades:

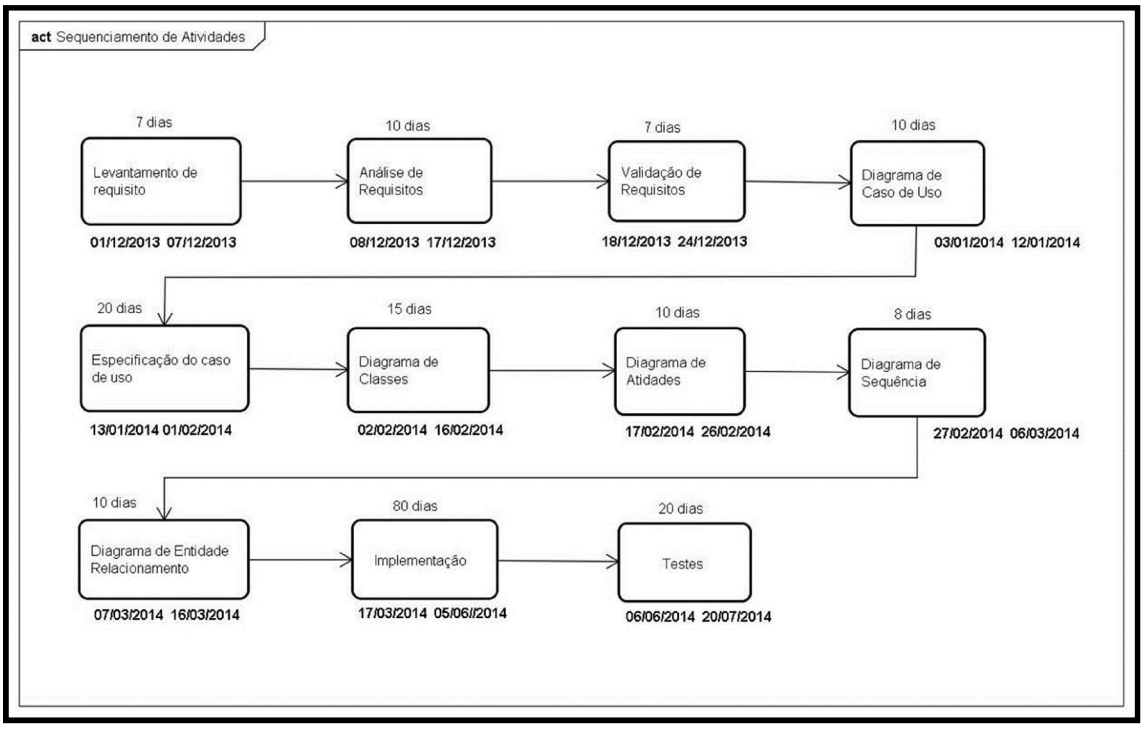

Figura 28 – Sequenciamento das Atividades.

O diagrama apresentado ilustra o tempo necessário para conclusão do projeto de todas as atividades do trabalho de conclusão de curso.

5 – IMPLEMENTAÇÃO DO APLICATIVO

Nesta seção serão apresentados as implementações dos aplicativos web e de mobilidade integrados para o sistema de Gestão de Tratos Culturais. Para implementação foi utilizado *Java Android, Java EE* com *Frameworks Prime Faces* e o banco *PostgreSQL*.

5.1 – APLICAÇÃO WEB

A aplicação Web foi desenvolvida para os cadastros, apontamentos, relatórios, notificações, localização de dispositivos, planejamentos das operações, vindo ajudar a tomadas de decisões e acompanhamento do processo um todo.

O sistema Web está dividido em camadas:

BO (*Business Object*): Camada responsável pelas regras de negócio do sistema ou seja, responsável pela validação dos objetos, sendo assim os objetos não tem acesso direto ao banco de dados.

DAO (*Data Acess Object*): Camada responsável pela conexão direta ao banco de dados realizando a persistência de dados com os objetos da camada BO com o Banco.

VO (*Value Object*): Sendo a camada responsável por conter as classes encapsuladas de dados guardando os objetos e seus tipos fazendo as transações entre as camadas BO e DAO.

Utilizando camadas o projeto fica de fácil manutenção e vindo evitar possíveis erros.

A Figura 29 apresenta o projeto desenvolvido em camadas:

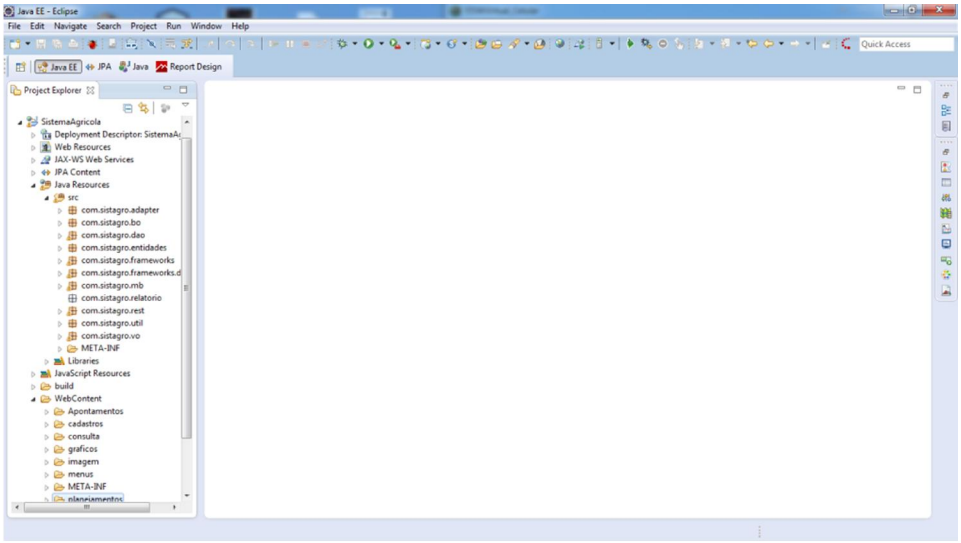

Figura 29 – Estrutura em Camadas.

5.1.1 – Tela de Login

Tela onde o usuário ou administrador acessa o sistema para utilizar as todas as funcionalidades, sendo controlado por perfil de acessos.

A Figura 30 apresenta Tela de Login:

Figura 30 – Tela de Login - Web.

5.1.2 – Tela de Cadastros

Tela onde o usuário ou administrador realizar os cadastros e parametrizar principais as principais funcionalidade do sistema que serão utilizadas tanto na *web* como na mobilidade.

A Figura 31 apresenta Tela de Cadastros:

Figura 31 – Tela de Cadastros - Web.

5.1.3 – Tela de Apontamentos Temporários

Tela onde o usuário ou administrador realizar as correções necessárias que a procedure encontrou ao processar o apontamento realizado na mobilidade.

A Figura 32 apresenta Tela de Apontamentos Temporários:

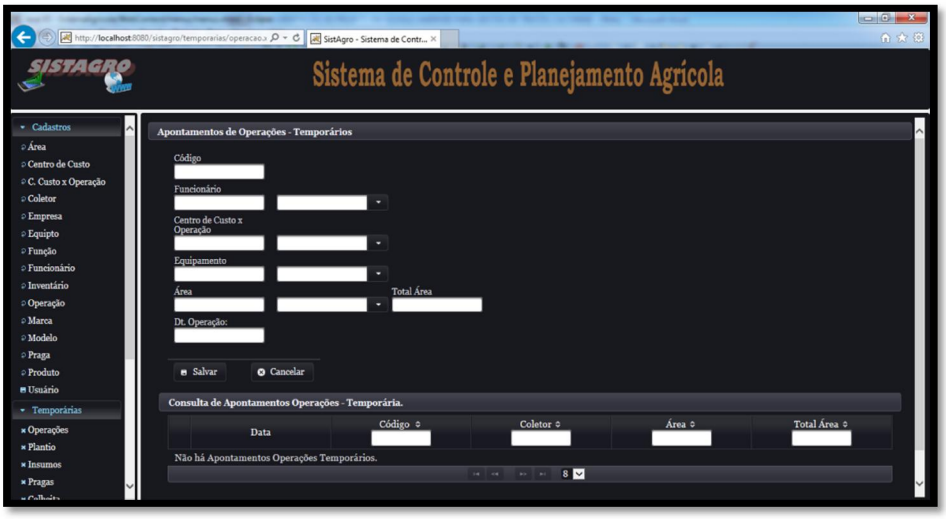

Figura 32 – Tela de Apontamentos Temporários – Web.

5.1.4 – Tela de Apontamentos

Tela onde o usuário ou administrador caso haja necessidade de realizar algum apontamento ou correções.

A Figura 33 apresenta Tela de Apontamentos:

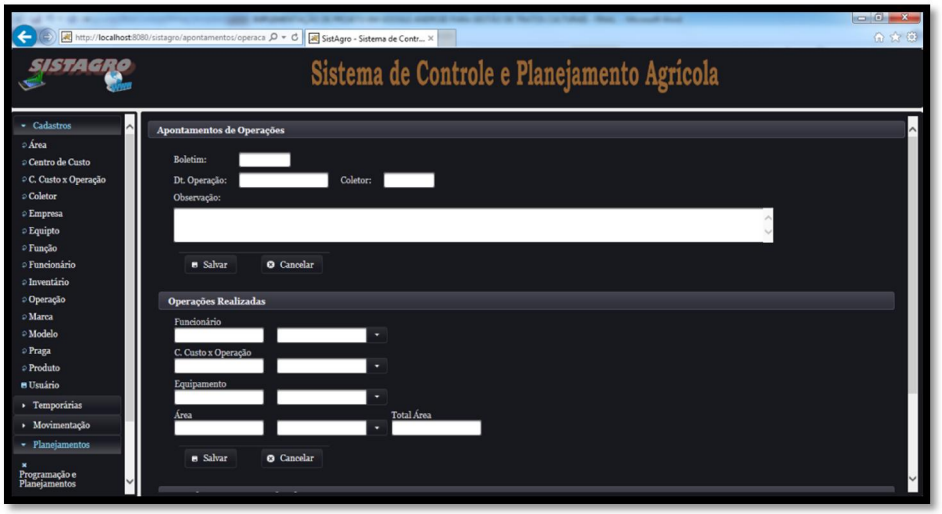

Figura 33 – Tela de Apontamentos – Web.

5.1.5 – Tela de Programação e Planejamento

Tela onde o usuário ou administrador para um melhor planejamento e programação das operações da safra toda, sendo enviado para o dispositivo para execução da operação na data prevista.

A Figura 34 apresenta Tela de Planejamentos:

Figura 34 – Tela de Planejamentos.

5.2.6 – Tela de Localização e Envio de Mensagem

Tela onde o usuário ou administrador receberá a localização através do GPS do dispositivo a cada certo tempo e podendo enviar mensagem para determinados aparelhos.

A Figura 35 apresenta Tela de Localização em Envio de Mensagens:

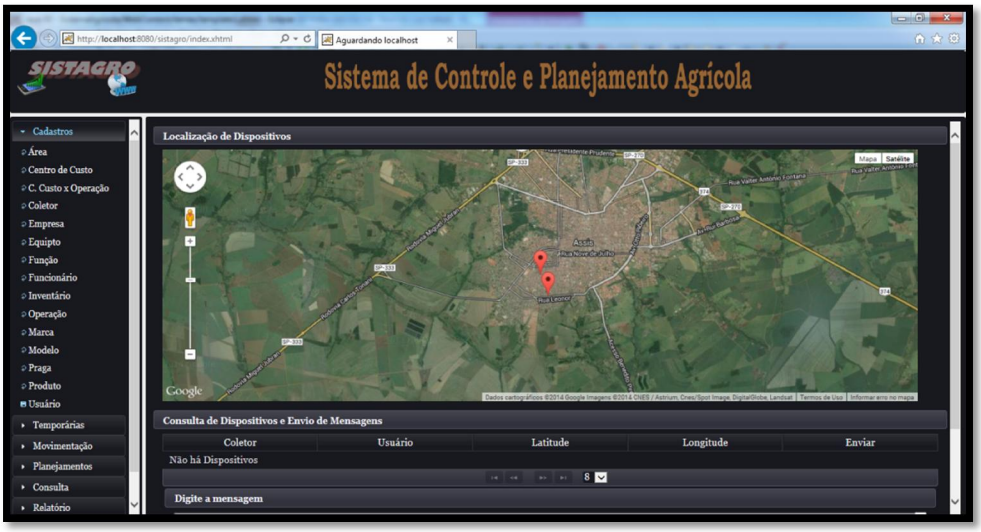

Figura 35 – Tela de Localização e Envio de Mensagens.

5.2 – APLICAÇÃO *GOOGLE ANDROID*

A aplicação *Google Android* foi desenvolvida para realização de apontamentos, análises dos dados apontados, notificações dos planejamentos, localização assim evitando apontamentos no papel e perda de informação, com envios dos apontamentos realizados conforme programando, mantendo o usuário atualizado do que está acontecendo no campo.

5.2.1 – Tela de Login

Tela onde o usuário ou administrador acessa para realizar apontamentos, consultas, gráficos e mensagens recebidas da aplicação *Web*.

A Figura 36 apresenta Tela de Login:

Figura 36 – Tela de Login – Google Android.

5.2.2 – Tela Menu Principal

Tela onde o usuário ou administrador terão acesso aos principais funcionalidades do sistema sendo controlado por nível de acessos, vindo evitar o usuário acessar menus não permitidos.

A Figura 37 apresenta Tela do Menu Principal:

Figura 37 – Tela do Menu Principal – Google Android.

5.2.3 – Menu Principal de Apontamentos

Tela onde o usuário ou administrador realizar os apontamentos dos processos realizados no campo, sendo uma operação, plantio, verificação de pragas, colheita e entrega do produto colhido.

A Figura 38 apresenta Tela do Menu Apontamentos:

Figura 38 – Tela de Menu Apontamentos – Google Android.

5.2.3 – Tela de Apontamento de Operações

Tela onde o usuário ou administrador realizar os apontamentos as operações realizadas nos processos, sendo uma interface amigável e simples para auxiliar o usuário. Tendo listas de consultas para facilitar a localização dos dados.

A Figura 39 apresenta Tela de Apontamento de Operação:

Figura 39 – Tela de Apontamentos de Operações – Google Android.

5.2.3 – Tela de Mensagens

Tela onde o usuário ou administrador verifica as mensagens recebidas pelo usuário da *web* ou programação do sistema, assim mantendo uma comunicação usuário mobilidade e usuário *web*.

A Figura 40 apresenta Tela de Mensagem:

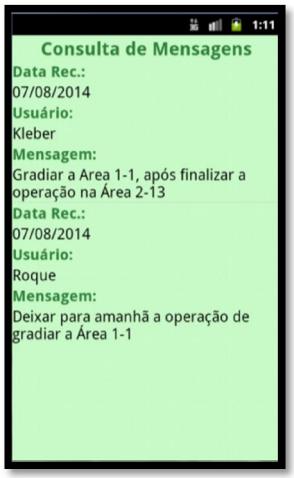

Figura 40 – Tela de Mensagens – Google Android.

6 – CONCLUSÃO

Com a área da agricultura é o setor que mais realiza movimentações na economia brasileira, mesmo sendo o menor produtor, sempre exposta à concorrência internacional, principalmente após a integração progressiva do Brasil nos mercados mundiais. Este fato abriu portas para ajustes de processo e novas tecnologias que ajudem a suprir as necessidades básicas como maior produção com baixo custo assim vindo elevar os lucros.

Em cima de estudos, o desenvolvimento de projeto de conclusão de curso, pode vim ajudar a esse mercado tão competitivo e ao mesmo tempo carente de tecnologias, vindo assim abrir grandes oportunidades de estudos já nas ferramentas *open-source*, ajudando a baratear as tecnologias e ao mesmo tempo agregando valores ao agronegócio.

Sendo utilizado uma ferramenta que certamente todos já tem em suas mãos, celulares, *tablets* e *internet*, sendo assim fica fácil de agregar valores e criar novos meios de tomar decisões que ajudaram no futuro ter uma produtividade maior, com o baixo custo.

Com o desenvolvimento de trabalho pretendo adquirir mais conhecimentos em relação ao planejamento, modelagem e desenvolvimento de software. Conhecendo novas ferramentas como o *Android*, JPA, JSF, *PrimeFaces* e o banco de dados *PostgreSQL*. Sendo assim criando aplicações com interfaces mais ricas e de baixo custo para o usuário final.

Para trabalhos futuros pretendo estar desenvolvendo na mobilidade o sistema de GCM vindo assim melhorar a comunicação entre o controle e os dispositivos através de mensagens dinâmicas, além de adaptar a aplicação para mapas, vindo assim facilitar para o usuário final.

REFERÊNCIAS

ABLESON, W. Frank; SEN, Robi; King, Chris, Ortiz, C. Enrique. **Android em Ação**. 3. ed. Tradução de Eduardo Kraszczuk. Rio de Janeiro: Elsevier, 2012.

Pereira, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. **Android Para Desenvolvedores.** 3. ed. Rio de Janeiro: Brasport, 2009.

LECHETA, Ricardo R. **Google Android**. – Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 1. ed. São Paulo: Novatec, 2009.

COSTA, Daniel G. ***JAVA* em Redes –** Recursos Avançados de Programação. 1. ed. Rio de Janeiro: Brasport, 2008.

SILVEIRA, Paulo; SILVEIRA, Guilherme; LOPES, Sérgio; MOREIRA, Guilherme; STEPPAT, Nico; KUNG, Fábio. **Introdução à arquitetura e *design* de software –** Uma visão sobre a Plataforma *Java***.** 1. ed. São Paulo: Elsevier, 2012.

JUNIOR, Alfredo Luiz dos Santos. **Integração de Sistema com *Java*.** 1. ed. Rio de Janeiro: Brasport, 2007.

HUBBARD, John R. **Programação com *Java*.** 2. ed. Tradução de Bookman, divisão da Artmed. São Paulo: Artmed, 2004.

CADENHEAD, Rogers; LEMAY, Laura. **Aprenda em 21 dias *Java* 2.** 4. ed. Rio de Janeiro: Elsevier, 2005.

DEITEL, Paul; DEITEL, Harvey. ***Java* como programar. 8**. ed. Tradução de FURMANKIEWICZ, Edson. São Paulo: Pearson, 2009.

HORSTMANN, Cay S.; CORNELL, Gary. **Core *Java* –** Volume I Fundamentos**. 8**. ed. Tradução de SCHAFRANSKI, Carlos; FURMANANKIEWICZ. São Paulo: Pearson, 2010.

SAMPAIO, Cleunton. ***JAVA Enterprise Edition*** 6 **–** Desenvolvendo Aplicações Corporativas**.** 1. ed. Rio de Janeiro: Brasport, 2011.

DEVEMEDIA. **Introdução à Java *Persistence* API - JPA**. Disponível em <http://www.devmedia.com.br/introducao-a-java-persistence-api-jpa/4590>. Acesso em 28/02/2014.

HLAVATS, Ian. ***Primefaces Starter –*** *Desing and develop awesome web user interfaces for desktop and Mobile devices with PrimeFaces and JSF2 using practical, hands-on examples***.** 1. ed. *Birmingham*: *Packt Publishing*, 2013.

SANTOS, Anderson Carlos Bueno dos; VISOLE, Marcos Cesar; VACARI, Isaque; MEIRA, Carlos Alberto Alves; PIMENTA, Tiago Augusto; MAZZOTTI, Bruno Franciscon. **Desenvolvimento web com PrimeFaces: uso no projeto Banco de Dados Pragas Quarentenárias**. <http://ainfo.cnptia.embrapa.br/digital/bitstream/item/23892/1/p052.pdf>. Acesso em 07/03/2014.

MOMJIAN, Bruce. ***PostgreSQL –*** *Introduction and Concepts***.** 2. ed. *Upper Saddle River*: *Addison-Wesley*, 2001.

DOUGLAS, Korry; DOUGLAS, Susan. ***PostgreSQL*.** 2. ed. *Birmingham*: *Sams Publishing*, 2005.

SIMON, Riggs; KROSING, Hannu. ***PostgreSQL 9 Administration Coockbook –*** *Solve real-word PostgreSQL problems with over 100 simple yet incredibly effective recipes***.** 1. ed. *Birmingham*: *Packt Publishing*, 2010.

SMITH, Gregory. ***PostgreSQL 9.0 High Performance –*** *Accelerate your PostgreSQL system and avoid the common pitfalls that can slow it down***.** 1. ed. *Birmingham*: *Packt Publishing*, 2010.

Ramos, Pedro. **Dimensões do Agronegócio Brasileiro Políticas, Instituições e Perspectivas**. Brasília: MDA, 2007.

Bilibio, Carolina. **Planejamento Estratégico na Empresa Agrícola Familiar.** São Luis: EDUFMA, 2009.

FOWLER, Martins; SCOTT, Kendall. **UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos.** 3. ed. Porto Alegre: Bookman, 2005.

REZENDE, Denis Alcides. **Engenharia de software e sistema de informação.** 3. ed. São Paulo: Brasport, 2005.

PRESSMAN, Roger S. **Engenharia de software** – Uma abordagem profissional. 7. ed. São Paulo: Artmed, 2011.

MARTINS, José Carlos Cordeiro. **Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML.** 5. ed. Rio de Janeiro: Brasport, 2010.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML -** Guia do Usuário**.** 2. ed. São Paulo: Elsevier, 2006.

VARGAS, Ricardo. **Manual prático do plano de projeto –** Aprenda a construir um plano de projeto passo a passo através de exemplos**.** 3. ed. Rio de Janeiro: Brasport, 2007.