WESLEY CAMARGOS PEGO

DESENVOLVIMENTO DE UM APLICATIVO MOBILE ANDROID DE CONTROLE DE ORDENS DE SERVIÇO PARA INFORMÁTICA ZN DE TEÓFILO OTONI - MG

FACULDADES UNIFICADAS DE TEÓFILO OTONI
TEÓFILO OTONI - MG
2017

WESLEY CAMARGOS PEGO

DESENVOLVIMENTO DE UM APLICATIVO MOBILE ANDROID DE CONTROLE DE ORDENS DE SERVIÇO PARA INFORMÁTICA ZN DE TEÓFILO OTONI – MG

Monografia apresentada ao Curso de Sistemas de Informação das Faculdades Unificadas de Teófilo Otoni, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.
Área de Concentração: Desenvolvimento Android.
Orientador: Prof. Amaury Gonçalves Costa.

FACULDADES UNIFICADAS DE TEÓFILO OTONI
TEÓFILO OTONI - MG
2017

rede de ensino DOCTUM

FACULDADES UNIFICADAS DE TEÓFILO OTONI
NÚCLEO DE TCC / SISTEMAS DE INFORMAÇÃO
Autorizado pela Portaria 4.012 de 06/123/2004 – MEC

FOLHA DE APROVAÇÃO

A monografia intitulada: *Desenvolvimento de um Aplicativo Mobile Android de controle de Ordens de Serviço para a Informática ZN de Teófilo Otoni - MG,*

elaborada pelo aluno Wesley Camargos Pego,

foi aprovada por todos os membros da Banca Examinadora e aceita pelo curso de Sistemas de Informação das Faculdades Unificadas de Teófilo Otoni, como requisito parcial da obtenção do título de

BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Teófilo Otoni, 20 de novembro de 2017

Professor Orientador: Amaury Gonçalves Costa

Professora Examinadora: Yvssa Carneiro Desmots Eliote

Professor Examinador: Marinho Soares

LISTA DE SIGLAS E ABREVIATURAS

API - *Application Programming Interface* (Interface de Programação de Aplicativos)
APK - *Android Package File* (Pacote Android)
AVD - *Android Virtual Device* (Dispositivo virtual do Android)
BD - Banco de Dados
DCL - Linguagem de Controle de Dados
DDL - Linguagem de Definição de Dados
DML - Linguagem de Manipulação de Dados
HAL - *Hardware Abstraction Layer*
HTTP - *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IBM - *Internacional Business Machines* (Máquinas de Negócios Internacionais)
IDE - *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
JDK - *Java Development Kit* (Kit de Desenvolvimento Java)
JSON - *JavaScript Object Notation*
OHA - *Open Handset Alliance*
PHP - *Hypertext Preprocessor*
POO - Programação Orientada à Objetos
RF - Requisitos Funcionais
RNF - Requisitos Não Funcionais
SDK - *Software Development Kit* (Kit de Desenvolvimento de *Software*)
SGBD - Sistema Gerenciador de Banco de Dados
SQL - *Structured Query Language* (Linguagem de Consulta Estruturada)
TI - Tecnologia da Informação
UML - *Unified Modeling Language* (Linguagem de Modelagem Unificada)
XML - *Extensible Markup Language* (Linguagem de Marcação Estendida)

LISTA DE ILUSTRAÇÕES

RESUMO

No mercado desde 2014 a Informática ZN está sem um controle efetivo das ordens de serviço, porém, faz-se necessário a melhoria desse processo, caso a empresa deseje continuar prestando serviço e atendimento de qualidade aos clientes, se destacando no mercado competitivo. Esta Monografia de Conclusão do Curso de Sistemas de Informação em questão, direcionada à área de desenvolvimento Android, tem como tema o “Desenvolvimento de um Aplicativo Mobile Android de Controle de Ordens de Serviço para Informática ZN de Teófilo Otoni - MG”. O projeto refere-se ao desenvolvimento de uma aplicação para a plataforma Android, destinada ao controle de ordens de serviço, a fim de aumentar a eficiência dos colaboradores e a resposta ao cliente em Teófilo Otoni e região. Portanto, almeja-se iniciar uma melhoria tecnológica na Informática ZN, a fim de também melhorar seu desempenho interno, e externo, de forma mais prática, rápida e segura.

Palavras-chave: Aplicativo; Informática ZN; desenvolvimento; Android.

SUMÁRIO

INTRODUÇÃO

A presente monografia tendo como título: “Desenvolvimento de um Aplicativo Mobile Android de Controle de Ordens de Serviço para Informática ZN de Teófilo Otoni - MG” concentra-se na área de desenvolvimento Android.

Tem como objetivo o desenvolvimento de um aplicativo Android para controle de ordens de serviço visando a melhoria na organização interna e melhorando automaticamente o tempo de resposta ao cliente.

A empresa Informática ZN, fundada em 2013 na cidade de Teófilo Otoni – MG, tem como objetivo proporcionar a seus clientes um serviço com alto padrão de qualidade, rapidez e confiabilidade. O foco da mesma é manutenção em computadores, *notebook*, *tablets* e *smartphones*. Os problemas começaram devido ao alto índice de procura e também pela nova sede da empresa, situada em uma região mais movimentada, o que fez com que a mesma ganhasse mais visibilidade. Tal expansão não foi acompanhada de uma reestruturação na forma do atendimento e recebimento dos equipamentos dos clientes no laboratório de reparos, gerando assim uma bola de neve em serviços atrasados. Alguns aparelhos, por exemplo, chegam a ficar por mais de 7 dias parados sem orçamento feito.

Como não há um sistema de controle das ordens de serviços e tampouco de clientes, tudo é escrito a punho no equipamento deixado pelo cliente e não há registro de números de protocolo, problemas relatados e valores de orçamento, ou seja, nenhuma ferramenta é utilizada para controle interno.

Para resolução do problema, foi proposto o desenvolvimento e implantação de uma aplicação para resolver tais problemas da Informática ZN. Para o desenvolvimento, foram levadas em consideração várias opções de plataformas móveis. Algumas delas foram inviabilizadas como, por exemplo, a plataforma iOS, devido o seu custo em equipamento e aprendizado da linguagem nativa. Por fim,

definiu-se o desenvolvimento para a plataforma Android, uma vez que o custo para produção de aplicativos é baixo e a maioria dos funcionários utilizam a mesma.

Diante deste contexto, a monografia aqui apresentada parte da seguinte questão: O desenvolvimento de um aplicativo de controle de ordens serviço seria capaz de agregar valor aos processos da empresa Informática ZN melhorando a qualidade e a eficiência do atendimento prestado?

Durante a pesquisa também foram definidas algumas hipóteses a fim de responder adequadamente à questão anterior, sendo elas:

Hipótese 0 – A empresa iria desistir do gerenciamento das ordens de serviço via aplicação mobile e iria optar pela aquisição de um livro de protocolos para registrar os atendimentos e registrar observações em uma planilha.

Hipótese 1 – A implantação do aplicativo na empresa Informática ZN seria ideal, porém não supriria todos os requisitos e acabaria não atendendo a todas as demandas da empresa, acabaria não resolvendo os problemas da mesma.

Hipótese 2 – A implantação do aplicativo na empresa Informática ZN seria ideal e teria uma boa aceitação por parte dos colaboradores e dos gestores da empresa, pois melhoraria todo processo de gestão da empresa e levaria a seus clientes um atendimento de melhor qualidade e eficiência.

Visando o objetivo geral, os objetivos específicos buscados com o trabalho foram:

- Coletar requisitos junto aos colaboradores.
- Planejar, mapear e implementar o banco de dados em um sistema de gerenciamento de banco de dados.
- Testar e validar os módulos junto aos colaboradores.
- Apresentar a aplicação completa aos colaboradores da empresa.
- Implantar aplicação.
- Realizar a capacitação dos colaboradores para com a aplicação.
- Acompanhar a utilização da aplicação.

A pesquisa foi realizada de forma bibliográfica e documental. Bibliográfica por haver uma pesquisa e a construção da base teórica, uma vez que houve a utilização de práticas de desenvolvimento já existentes e consolidadas por equipes e autores

renomados. Documental devido a utilização de conteúdo da internet sendo este, audiovisual ou escrito em forma de tutorial.

O trabalho classifica-se como laboratorial, pois o aplicativo foi desenvolvido em um laboratório especifico com testes de uso simulando o ambiente da empresa. Classificado também como intervencionista, uma vez que o cenário da empresa foi afetado sofrendo consideráveis mudanças em seu processo.

Devido aos vários tipos de linguagens, ficou definido como interdisciplinar, pelo uso de áreas diferentes da ciência referentes ao desenvolvimento *Java* e outras linguagens de programação, sistemas de gerenciamento de bancos de dados e metodologias próprias do Android.

Foi utilizado o método indutivo. A aplicação foi desenvolvida sob demanda e ajustada de acordo com as necessidades da empresa. Na conclusão deste trabalho, o aplicativo se mostrou de grande auxílio aos colaboradores da empresa.

A criação da aplicação mobile para gerenciamento de ordem de serviço da Informática ZN faz-se necessária porque atualmente, na empresa, não há nenhuma aplicação ou controle efetivo dos atendimentos feitos ao cliente e de cadastro de dados dos clientes, por exemplo o endereço. Como não há um controle, consequentemente a qualidade do serviço tende a cair e o tempo de reparo/serviço aumentar, afetando negativamente a empresa no mercado.

A solução proposta visa melhorar o tempo de orçamento e facilitar o controle interno, tendo protocolo conectado ao cliente e ao equipamento. Por exemplo, uma consulta de protocolo dentro do sistema exibiria as informações e os laudos técnicos informados durante e após o reparo do equipamento, podendo assim qualquer um que seja o utilizador do sistema no momento informar ao cliente todos os procedimentos que foram feitos e por quem foram feitos.

A aplicação foi desenvolvida para uma das plataformas que apresenta maior crescimento em número de utilizadores e qualidade de aplicativos, o Android. No desenvolvimento, foram utilizadas ferramentas criadas ao longo do tempo e experimentadas as funcionalidades novas que surgiam. Na esfera acadêmica, houve um enriquecimento do saber, uma vez que para utilizar as técnicas e metodologias de desenvolvimento para Android, uma vasta pesquisa foi feita, pois para a utilização das mesmas é necessário entender o seu funcionamento. Neste quesito também houve uma preparação pessoal, já que em projetos futuros o domínio da linguagem deverá ser maior.

A construção e finalização desta aplicação também terá efeito na sociedade pois será implantado em uma empresa real que lida diretamente com clientes, prazos e necessita de um controle em seus processos. Clientes bem atendidos e satisfeitos são sempre aqueles que fazem o negócio crescer e este crescimento consequentemente fará com que a Informática ZN aposte mais alto em tecnologia, elevando o nível de serviço na região onde atua.

Esta monografia está dividida em capítulos onde o primeiro, denominado, Principais Conceitos, explana sobre todo o conteúdo teórico que foi utilizado no desenvolvimento do aplicativo, dentre eles banco de dados, php, *webservice*, plataforma Android, UML.

No segundo capitulo há tudo que está relacionado ao desenvolvimento do aplicativo em si, explicando cada fase do processo como o levantamento dos requisitos e os testes.

E por fim no terceiro capitulo são mostrados os resultados do *feedback* dos colaboradores que serviram para validar ou não uma das hipóteses fornecidas.

1. PRINCIPAIS CONCEITOS

1.1 Linguagem de Programação e Marcação

O Android em sua codificação utiliza a linguagem de programação Java, e dá suporte a maioria dos seus recursos disponíveis. Já a linguagem Java tem suporte de integração com XML *(*acrônimo para *Extensible Markup Language* ou Linguagem de Marcação Estendida, em tradução livre) e a maioria dos APIs (*Application Programming Interface* ou em tradução livre, Interface de Programação de Aplicativo) do Java, que conseguem se relacionar com XML, têm suporte total no Android (GALPIN, 2009).

A tecnologia *Java*, segundo Perry (2016) "[...] é usada para desenvolver aplicativos para uma ampla variedade de ambientes, de dispositivos consumidores a sistemas corporativos heterogêneos".

E sobre XML, Pereira afirma o seguinte:

> O XML traz uma sintaxe básica que pode ser utilizada para compartilhar informações entre diferentes computadores e aplicações. Quando combinado com outros padrões, torna possível definir o conteúdo de um documento separadamente de seu formato, tornando simples para reutilizar o código em outras aplicações para diferentes propósitos (PEREIRA, 2009).

1.1.1 Programação Orientada a Objetos

A POO (Programação Orientada a Objetos) é um padrão de desenvolvimento seguido por linguagens como Java, Python, C# e C++, apesar destas linguagens serem diferentes todas seguem a mesma essência (FELIPE, 2015).

Para uma linguagem encaixar nos padrões da Orientação a Objetos, David (2009) aponta que se deve ter como base os 4 pilares básicos abaixo:

- Herança é um mecanismo que permite criar novas classes a partir de classes já existentes, consiste no compartilhamento de atributos e métodos assim aproveitando as características existentes da classe pai. Este mecanismo promove o reaproveitamento de código.

- Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação, assinatura, mas comportamentos distintos.

- Encapsulamento na programação significa separar o programa em partes, sendo como ideia principal tornar o software mais flexível e mais fácil de modificar e de adicionar novas implementações. Serve para o controle de acesso aos métodos e atributos de certa classe.

- Abstração segundo Vinicius (2012) é “utilizada para a definição de entidades do mundo real. Sendo onde são criadas as classes. Essas entidades são consideradas tudo que é real, tendo como consideração as suas características e ações”. Tal conceito tem como exemplo a figura 1 abaixo:

Figura 1: Abstrações do mundo real.

Entidade	Características	Ações
Carro, Moto	tamanho, cor, peso, altura	acelerar, parar, ligar, desligar
Elevador	tamanho, peso máximo	subir, descer, escolher andar
Conta Banco	saldo, limite, número	depositar, sacar, ver extrato

Fonte: http://www.devmedia.com.br/abstracao-encapsulamento-e-heranca-pilares-da-poo-em-java/26366

1.1.1.1 Linguagem Java

A linguagem Java, como definido anteriormente, serve de base para o desenvolvimento Android. A linguagem apresenta suas próprias regras e sintaxes, ou seja, tem uma forma de escrita própria e apresenta termos únicos que remetem a funções e procedimentos específicos. É derivada da linguagem C e estruturalmente falando é separada por pacotes (PERRY, 2016).

Sobre a estrutura básica do Java, constata-se que ela é uma linguagem orientada a objetos, ou seja, ela faz a representação dos objetos do mundo real na codificação. Um objeto em Java, assim como objetos reais, possui atributos e realiza ações, denominadas métodos. Objetos que contêm a mesma estrutura e são semelhantes são agrupados em uma classe. E por fim, o pacote, são conjuntos de classes semelhantes, porém que executam ações distintas.

Sobre a compilação do código Java, Perry (2016) diz o seguinte:

> Quando você programa na plataforma Java, escreve seu código-fonte em arquivos .java e depois os compila. O compilador verifica seu código nas regras de sintaxe da linguagem e depois grava *bytecode* em arquivos .class. *Bytecode* é um conjunto de instruções destinadas a executar em uma *Java virtual machine* (JVM). Ao incluir esse nível de abstração, o compilador Java difere-se de outros compiladores de linguagem, que escrevem instruções adequadas para o chipset de CPU no qual o programa é executado (PERRY, 2016).

1.2 Banco de Dados

Banco de dados para Elmasri e Navathe (2005, p. 4) “é uma coleção de dados relacionados. Dados são fatos que podem ser gravados e que possuem significados implícitos.”. Todo conteúdo gerado pela aplicação Android, será salvo de forma sistemática a fim de que sejam consultadas posteriormente para obtenção de informações.

A criação de um banco de dados, deve ser uma das primeiras coisas a serem feitas logo após a coleta dos requisitos. O banco, quando bem estruturado e montado é uma ferramenta facilitadora na parte da codificação, pois o mesmo possui ferramentas automáticas que executam certos processos em que a aplicação não precisa se preocupar.

Quanto a utilização e acesso de um banco de dados, o mesmo pode ser feito através de linhas de comando, porém o mais aconselhado é que se utilizem um SGBD (Sistema Gerenciador de Banco de Dados), que basicamente, é um sistema cujo o objetivo é gerenciar o acesso e a manutenção correta dos dados em um banco de dados (MIRAS, 2008).

O modelo de banco de dados utilizado será o relacional. A estrutura do banco relacional prevê que a sua criação seja baseada em entidade e seus relacionamentos. As entidades, são a personificação de algo no mundo real como por exemplo um usuário e uma ordem de serviço. A ordem de serviço possui seus atributos e se relaciona com os usuários resultando uma ação. Define-se então que quando um usuário abre uma ordem de serviço ele está executando um relacionamento, em linguagem de banco de dados.

1.2.1 SQL

O SQL é uma linguagem de manipulação de banco de dados relacional, a sua utilização se dá por meio dos SGBDs e é dividida em 3 grupos de acordo com suas funções. Sobre a linguagem SQL Ferrari (2007, p. 12) afirma o seguinte:

> A linguagem SQL é um conjunto de comandos (ou instruções) que permitem gerar enunciados, ou seja, linhas de comandos compostas por uma ou mais instruções. Alguns comandos permitem ou até mesmo exigem o uso de parâmetros adicionais, chamados de cláusulas e predicados. Basicamente, todos os comandos do SQL são verbos em inglês (*select, create, alter* etc.), e as cláusulas são preposições do mesmo idioma (*from, as, by, in* etc.).

1.2.1.1 Grupos de comandos SQL

Os comandos SQL como citado anteriormente, são classificados em 3 grupos, sendo eles:

- DDL (*Definition Data Language* ou Linguagem de Definição de Dados, em tradução livre): São todos os comandos utilizados para a criação e alteração das tabelas que compõem o banco de dados, basicamente são os comandos que definem a estrutura dos dados. Dentre estes comandos tem-se as funções CREATE, ALTER, DROP etc. [1]

- DML (*Data Manipulation Language* ou Linguagem de Manipulação de Dados, em tradução livre): São os comandos utilizados para manipular e extrair os dados existentes no banco. Dentre tais comandos tem-se INSERT, SELECT, UPDATE etc. [2]

- DCL (*Data Control Language* ou Linguagem de Controle de Dados, em tradução livre): Segundo Ferrari (2007, p. 12) esse grupo é “um conjunto de comandos usado em sistemas multiusuário para definir os privilégios de acesso aos dados a cada usuário”. Resumidamente tais comandos são para controlar o nível de acesso de usuários. Dentre os comandos tem-se o GRANT, DENY, etc.

[1] FERRARI, 2007, p. 12.
[2] Ibidem.

1.2.2 SGBD MySQL

O MySQL é um SGBD relacional que utiliza linguagem SQL (*Structured Query Language* ou Linguagem de Consulta Estruturada). Inicialmente foi desenvolvido para atender a pequenos e médios projetos, é multitarefas, multiusuários e atualmente pode ser utilizado em qualquer tipo de projeto desde pequeno ao grande, suas atualizações lhe deram grandes capacidades por isso atualmente é um dos SGBDs mais utilizados e continua em constante atualização (MIRAS, 2008). Uma das ferramentas que podem ser utilizadas por exemplo para gerenciar banco de dados é o *Postgres*, *MySQL Workbench* e *IBExpert*, por exemplo.

1.3 PHP

PHP (*Hypertext Preprocessor*) é definido, conforme sua documentação oficial[3], "uma linguagem de script *open source* de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML."

O código em si é executado no servidor e tem-se a resposta enviada para o navegador em forma de HTML.

Pela facilidade que é a criação de API's, já que o mesmo está no lado servidor, se torna uma ótima opção para criar serviços na web, conhecidos como *WebServices*.

1.4 WebService

O *WebService* é uma forma de integração e comunicação de sistemas, graças a este conceito a forma como os sistemas são construídos melhorou, permitindo separar e quebrar o sistema em partes. O *webservice* pode realizar uma chamada para um serviço de outro sistema para obter informações. Tal comunicação e troca de

[3] Disponível em: <https://secure.php.net/manual/pt_BR/>.

dados se dá por arquivos de diversos formatos, sendo que os mais usados atualmente são o JSON e o XML (LECHETA, 2015).

1.4.1 JSON

O JSON (*JavaScript Object Notation*), é um modelo usado para armazenamento e transmissão de informações no formato de texto. Por ser mais leve, em comparação com o XML para trafegar pela rede, tem sido bastante utilizado principalmente por aplicações Web devido a capacidade de estruturar informações de uma forma mais compacta do que a do XML, pelo o tamanho ser reduzido, como citado anteriormente, torna a troca de informações mais rápida. O JSON foi adotado por empresas de grande renome como Google e Yahoo justamente pela sua simplicidade, velocidade e leveza, já que as mesmas, em suas aplicações, precisam transmitir grandes volumes de dados (CORRÊA, 2012).

1.5 UML

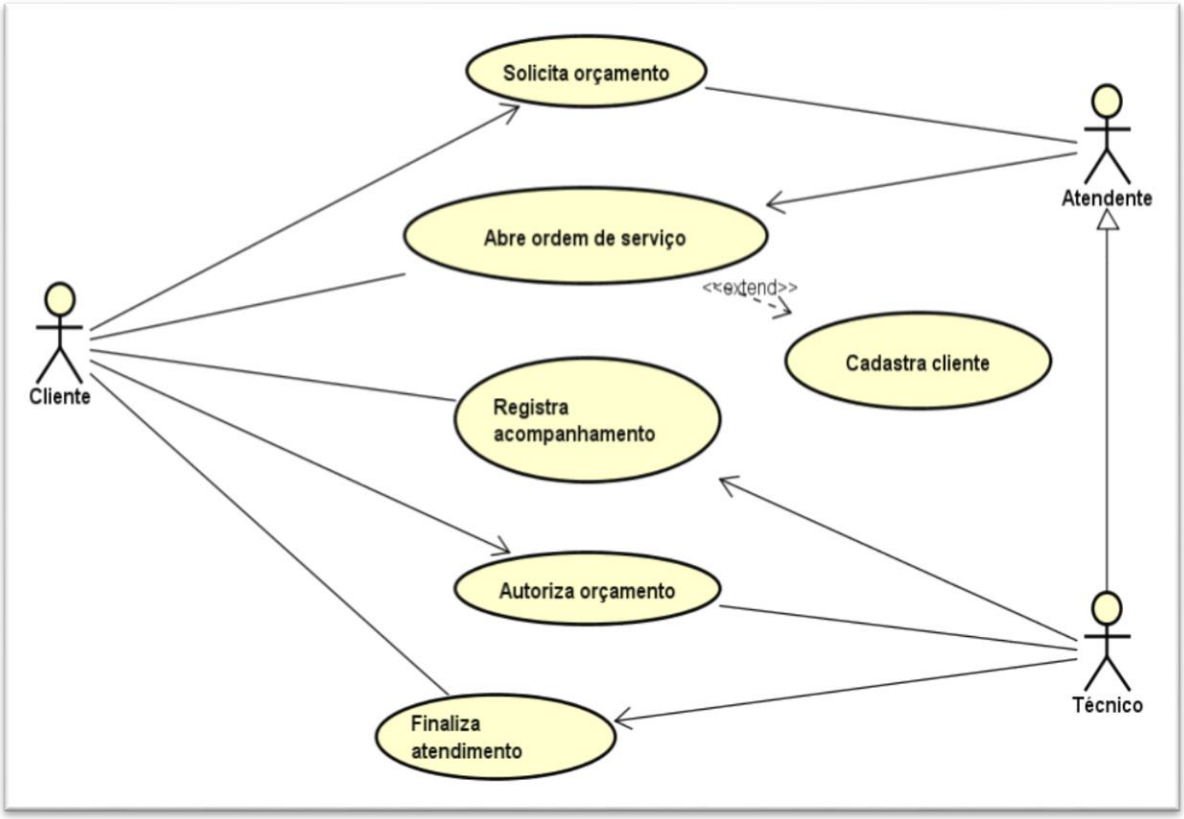
UML, é um acrônimo para o termo *Unified Modeling Language* que significa em Português, Linguagem de Modelagem Unificada. A UML se configura em uma linguagem utilizada para a especificação, construção e documentação visual dos componentes da aplicação.[4] A modelagem é a concepção da aplicação, antes da codificação.

Segundo Alhir (1999, p.11) o UML não é uma linguagem de programação e sim de modelagem. Um processo utilizado para documentar processos. A forma para utilizar essa linguagem é a construção de diagramas que são construídos por ferramentas próprias e com informações referentes ao projeto. Ainda para ALHIR os diagramas básicos são:

[4] Disponível em <http://www.uml.org/what-is-uml.htm>. Acessado em 19 out. 2017

- Modelo do usuário: que se referem a diagramas de caso de uso, que por sua vez têm como objetivo descrever as funcionalidades de um sistema. Esse diagrama, conforme exemplificado na figura abaixo (Figura 2), assim como os demais tem elementos figurativos próprios, com significado único, utilizados para identificar os elementos do projeto e do produto final. Os diagramas de caso de uso, permite uma visão geral de ações que serão e não serão possíveis de realização pelos usuários afetados pela aplicação, bem como a sua função na mesma. No caso do exemplo a aplicação referenciada há 3 atores, configurados pelo cliente, atendente e técnico que podem executar ações como a solicitação do orçamento e abertura da ordem de serviço.

Figura 2: Diagrama de caso de uso.



Fonte: Do Autor.

- Modelo Estrutural: modelo que permite a descrição da estrutura da aplicação em um momento específico. É representado pelo diagrama de objetos.

- Modelo Comportamental: representam a interação entre os elementos constituintes do sistema em um determinado tempo, apontando seus relacionamentos e atividades.

- Modelo de Ambiente: como a denominação informa, são diagramas que mapeiam a configuração dos elementos presentes no sistema e no ambiente.

Além desses diagramas definidos por Alhir, é possível a construção de outros diagramas conforme necessidades da organização, utilizando-se das premissas da linguagem.

1.6 Plataforma Android

A plataforma Android foi desenvolvida para dispositivos móveis, baseada no GNU/Linux e é produto de um grupo denominado *Open Handset Alliance* (OHA) liderado pelo Google.

O Android não é sucesso somente pela força e pela liderança do *Google*, embora seja de grande peso, mas sim graças a este grupo que é formado por várias empresas de renome como: Intel, Samsung, LG, Motorola, Sony, Acer, Dell, dentre outras (LECHETA, 2015, p. 26).

O objetivo do grupo OHA segundo Lecheta (2015. p. 26) seria “[...] definir uma plataforma única e aberta para celulares para deixar os consumidores mais satisfeitos com o produto final”. O Android está disponível também em outros dispositivos (plataformas) além dos *smartphones*, como TV (Google TV), relógios (Android *Wear*), óculos (Google *Glass*), carros (Android *Auto*) etc.

Segundo Lecheta (2015. p. 27) “O Android é um sistema operacional baseado no *kernel* Linux, que é responsável por gerenciar memória, processos, threads, segurança dos arquivos e pastas, além de redes e drivers. ”

Em um artigo feito por Ableson no site da IBM, também explicou o seguinte:

> [...] o Android é executado sobre um *kernel* Linux. Os aplicativos Android são gravados na linguagem de programação Java e são executados em uma

> máquina virtual (VM). É importante observar que a VM não é uma JVM[5], como você pode esperar, mas é uma *Dalvik Virtual Machine*, uma tecnologia de software livre. Cada aplicativo Android é executado em uma instância da *Dalvik VM*, que, por sua vez, reside em um processo gerenciado por kernel[6] Linux [...] (ABLESON, 2009)

Atualmente, já se tem a versão 8.0 da plataforma Android com codinome *Oreo*. Cada versão do Android faz referência a algum tipo de doce, por exemplo, a versão anterior 6.0 do sistema tem o codinome *Marshmallow*, e assim por diante.

1.6.1 Arquitetura Android

O sistema Android é no fim uma pilha de *softwares* que tem como base o código aberto Linux criado para vários tipos de dispositivos.

A arquitetura do sistema é dividida em várias camadas (Figura 3), sendo elas: *Kernel* do Linux, Camada de abstração de *hardware* (HAL), *Android Runtime*, Bibliotecas C/C++ nativas, Estrutura da Java API e Aplicativos do sistema. Cada camada é responsável pelos seus respectivos processos.

Na camada de Aplicativos do sistema (*System Apps*), é um conjunto de aplicativos principais que dentre eles tem o e-mail, envio de SMS, calendários, navegador de internet, contatos etc. São os aplicativos padrões de fábrica do sistema, mas, nada impede o usuário de instalar, por exemplo, outro navegador de internet.

Já a camada Estrutura da Java API (*Java API Framework*) é um conjunto completo de recursos do sistema Android disponível pelas APIs programadas na linguagem Java, tais APIs formam blocos de programação que são necessários para criação de aplicativos Android simplificando assim a reutilização de componentes e serviços de sistema.[7]

Na próxima camada tem-se uma divisão em dois grupos, a camada de Bibliotecas C/C++ nativas (*Native C/C++ Libraries*) e a *Android Runtime*. As Bibliotecas C/C++ nativas são vários componentes e serviços principais do sistema

---

[5] O JVM acrônimo para *Java Virtual Machine* (Máquina Virtual Java) é um programa que carrega e executa os aplicativos codificados em linguagem Java.
[6] O kernel é o núcleo do sistema operacional, o componente central do sistema.
[7] Disponível em: <https://developer.android.com/guide/platform/index.html?hl=pt-br>.

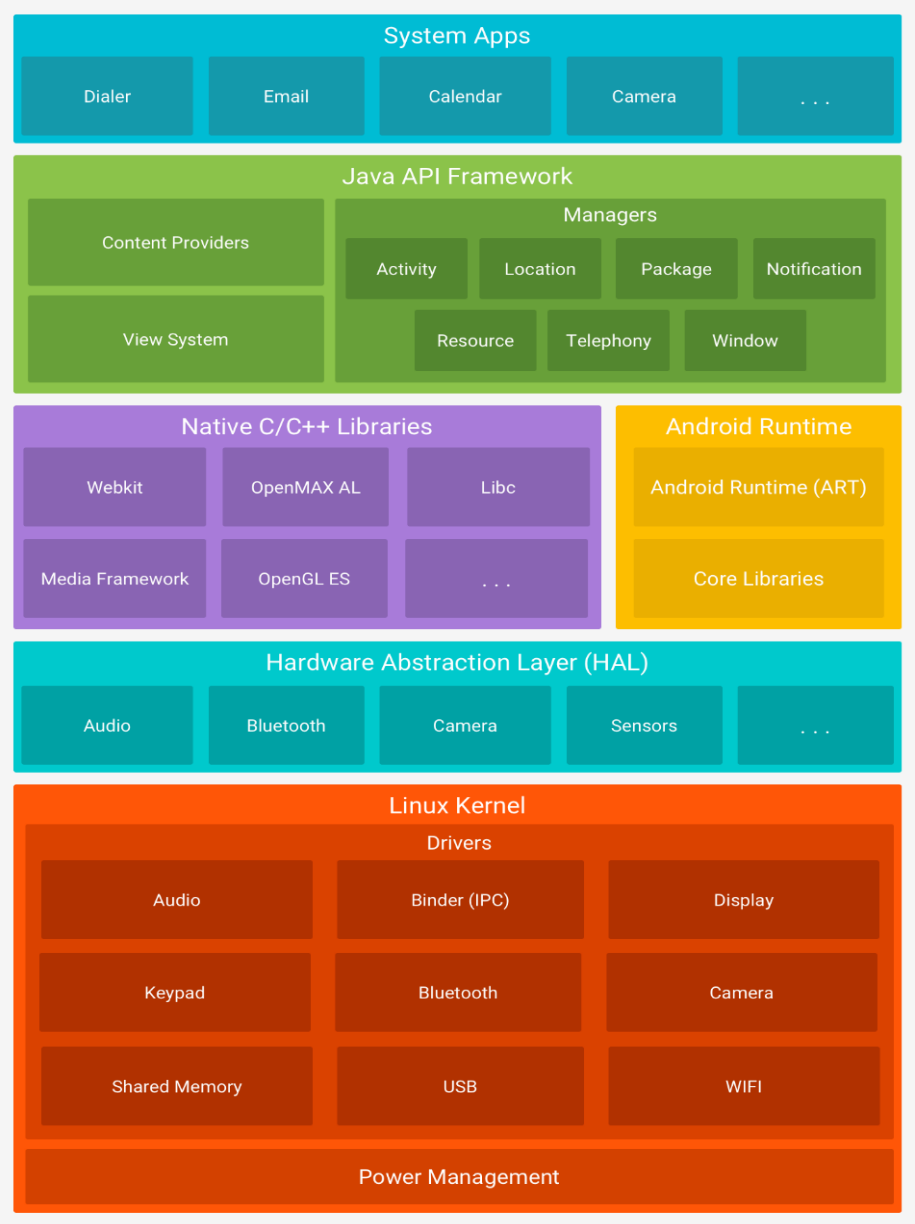
Android, como ART e HAL, explicados na próxima camada, e são implementados em código nativo que exige bibliotecas nativas programadas em C e C++.[8]

Na camada *Android Runtime* está a máquina virtual *Dalvik* para dispositivos com Android 4.4 e inferior e a máquina virtual ART para Android 5.0 e superior. Na máquina ART cada aplicativo executa o próprio processo como uma instância própria do *Android Runtime* (ART). Já na máquina *Dalvik* cada aplicação executa dentro do seu próprio processo. Na camada de abstração de *hardware* (HAL) (*Hardware Abstract Layer*), é uma camada que expõem as capacidades de hardware do dispositivo para a API de maior nível, faz-se uma interface de conexão entre modulo hardware e sistema.[9] Na base da plataforma *Android* está o *Kernel* do Linux, que é responsável por gerenciar memória, processos, threads, segurança e drivers.

A pilha de camadas do sistema pode ser melhor entendida na figura 3 a seguir:

[8] Disponível em: <https://developer.android.com/guide/platform/index.html?hl=pt-br>.
[9] Ibidem.

Figura 3: A pilha de software do Android.



Fonte: Guia API, Arquitetura[10]

1.6.2 Activity

Activity é basicamente o componente mais importante na criação de um aplicativo, ela representa uma tela da aplicação e é responsável por tratar os eventos

[10] Disponível em: <https://developer.android.com/guide/platform/index.html?hl=pt-br>.

gerados na mesma, normalmente esta classe herda a classe *android.app.Activity* ou de alguma subclasse desta, vale ressaltar também que para a adição de conteúdo dentro das *activitys* deve-se utilizar os arquivos de layout do *Android* do tipo XML, que definem os elementos visuais, a disposição de tais elementos, cores de botões etc (LECHETA, 2015, p. 79-81, 96-97).

Um aplicativo pode conter várias *activitys*, por exemplo, entrar nas configurações de um aplicativo, ao clicar nesta suposta opção de configuração provavelmente irá iniciar uma nova *Activity* acima daquela que estaria sendo utilizada no momento, podendo variar conforme o desenvolvedor e tipo de aplicação.

As *activitys* têm um ciclo de vida bem definido (Figura 2), tal ciclo consiste em um tipo de estado assumido pela *activity*, utilizando do exemplo imaginário do parágrafo anterior, ao clicar em configuração a *activity* atual seria pausada e iniciada uma nova *activity* com as configurações, ao voltar a *activity* das configurações seria pausada e a que estava inicialmente sendo utilizada retomada. Tais ciclos são métodos definidos como:

- onCreate (bundle), este método é chamado uma única vez obrigatoriamente, a chamada deste faz-se necessária para executar a aplicação.

- onStart, este método é chamado quando a *activity* está ficando visível para o usuário.

- onRestart, chamado quando uma *activity* foi parada temporariamente e está sendo reiniciada.

- onResume, é chamado quando a activity está no topo da pilha de *activitys*, chamada também de "*Activity Stack*", dessa forma sendo executada como *activity* principal e interagindo com o usuário. Este método representa então que a activity está em execução no momento como principal.

- onPause, sempre que a tela da *activity* fechar este método será chamado, podendo ser pelo usuário ter pressionado o botão home ou o botão voltar para sair da aplicação ou até mesmo em caso do recebimento de uma ligação, este

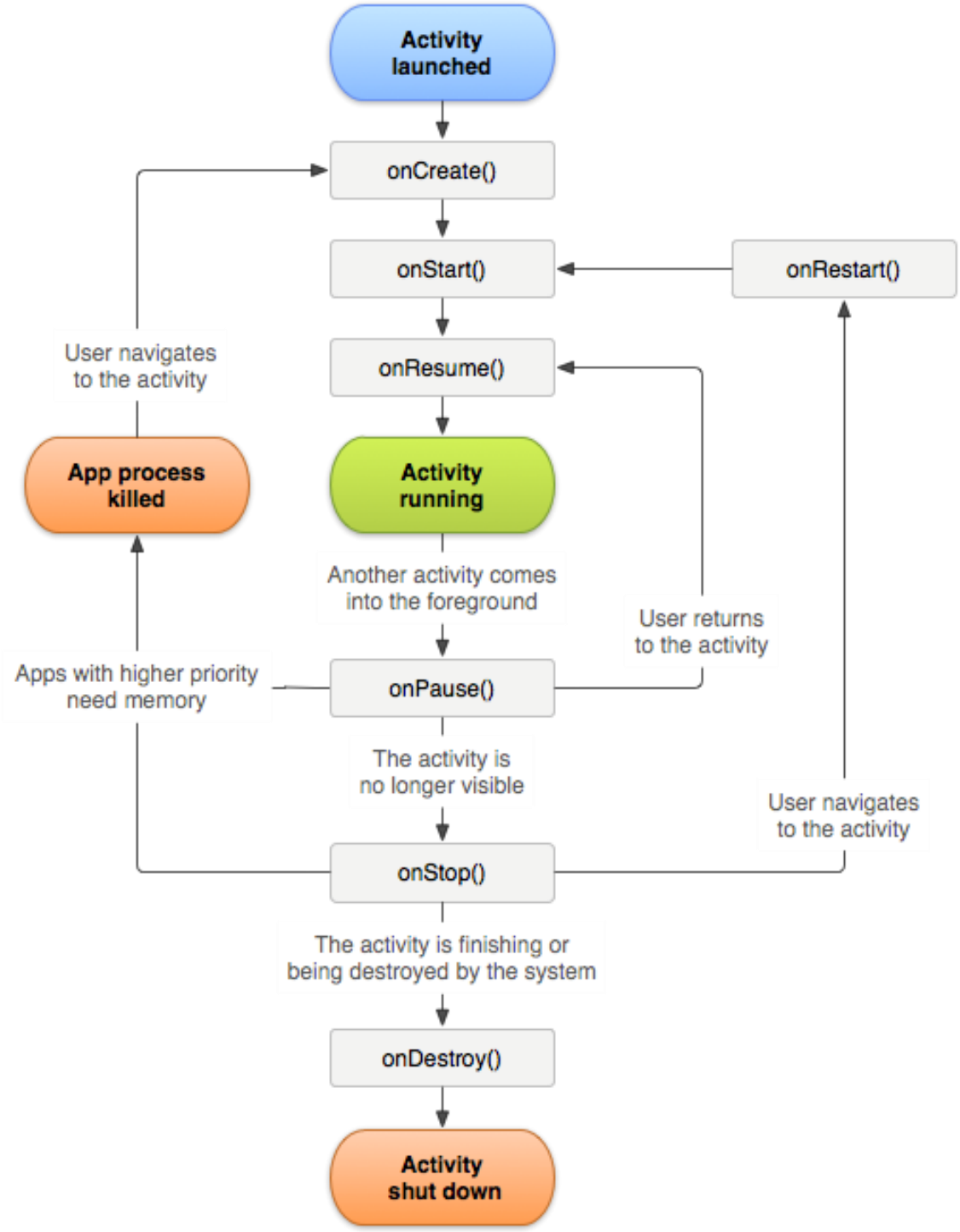
método salvara o estado da aplicação para que volte a executar posteriormente.

- onStop, chamado posteriormente ao *onPause* e indica que a activity está sendo encerrada e não estará mais visível ao usuário. Depois de parada a activity ainda poderá ser reiniciada pelo método *onRestart*. Caso fique muito tempo em segundo plano e o sistema precise liberar mais espaço na memória o método *onDestroy* será automaticamente executado para remover completamente da pilha de *activitys*.

- onDestroy, é o método que literalmente encerra a activity, podendo ser chamado também pelo sistema operacional a fim de liberar recursos ou pode ser chamado pela aplicação pelo método *finish*() da classe *activity*. Após o uso do *onDestroy* a *activity* é removida completamente da pilha de *activitys* e seu processo no sistema operacional é completamente encerrado. LECHETA (2015, p. 101-102).

Os métodos acima descritos, são exemplificados na figura (Figura 4) a seguir junto ao ciclo das mesmas:

Figura 4: Ciclo de vida da Activity.



Fonte: Guia API, Atividades[11]

[11] Disponível em: <https://developer.android.com/guide/components/activities.html?hl=pt-br>.

1.6.3 Ferramenta para Desenvolvimento Android

Para o desenvolvimento do projeto será utilizado a ferramenta oficial chamada de Android Studio, a mesma é a IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, em tradução livre) oficial de Desenvolvimento para Android (LECHETA, 2015, p. 44).

A ferramenta pode ser encontrada e baixada no site oficial do desenvolvedor Android, disponível no endereço https://developer.android.com/studio/, não há custos para download e já vem com o SDK integrado ao instalador. O SDK acrônimo para *Software Development Ki*t (Kit de desenvolvimento de software), por sua vez, auxilia o desenvolvedor em seu projeto com suas ferramentas e APIs. O Android SDK é definido por Lecheta (2015, p. 42) como “software utilizado para desenvolver aplicações no Android, que tem emulador para simular o dispositivo, ferramentas utilitárias e uma API completa para a linguagem Java, com todas as classes necessárias para desenvolver as aplicações”.

1.6.3.1 Android Studio

Como citado anteriormente O Android Studio é o IDE oficial do Android, criado especificamente para o Android. O mesmo foi apresentado ao público em 2013 e em 2014 já era a ferramenta padrão para desenvolvimento Android, o IDE é baseado no Intellij IDEA da JetBrains (LECHETA, 2015, p. 44).

Segundo o site oficial do desenvolvedor Android[12] o Android Studio “acelera o desenvolvimento e ajuda a criar aplicativos da mais alta qualidade para todos os dispositivos Android. Ele oferece ferramentas personalizadas para desenvolvedores do Android, incluindo ferramentas avançadas[...]”.

[12] Disponível em: <https://developer.android.com/studio/features.html?hl=pt-br>.

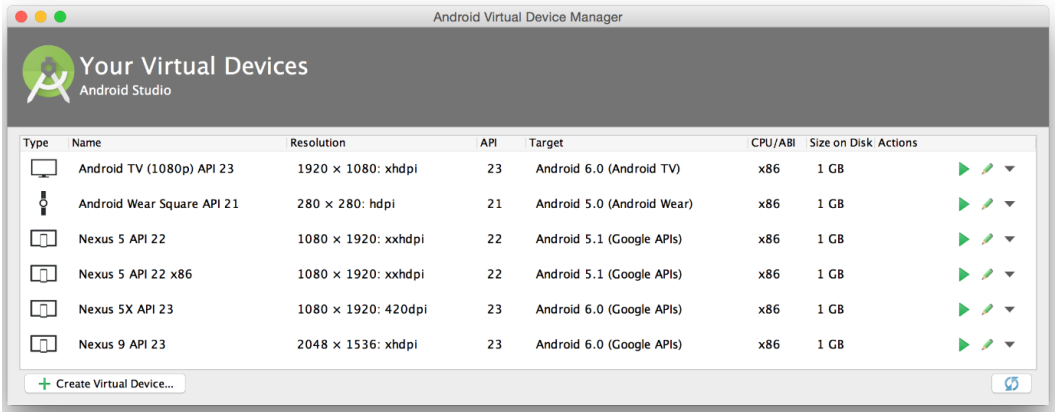
1.6.3.2 Emuladores

O Android Studio dá a possibilidade da utilização de emuladores na hora da execução do projeto, sendo os principais o AVD que vem incluso ao Android SDK após a instalação do Android Studio e o Emulador *Genymotion*.

O AVD (*Android Virtual Device* ou em tradução livre, Dispositivo virtual Android) é um meio mais prático de emulação de dispositivo, sendo que se tem várias opções de configuração de hardware quanto de *software*, tudo para testar o seu projeto em vários tipos diferentes de versões de sistema dentre outras opções de personalização.

Para gerenciar os dispositivos virtuais usa-se o AVD *Manager* (Figura 5) que é onde pode-se criar novas AVDs, definir o perfil de hardware, executar ou encerrar uma AVD etc. Esse gerenciador de dispositivos virtuais pode ser acessado a partir do menu *Tools* > Android > AVD *Manager* no Android Studio. Na figura 5 pode-se ver a tela principal do AVD *Manager*: [13]

Figura 5: AVD Manager.

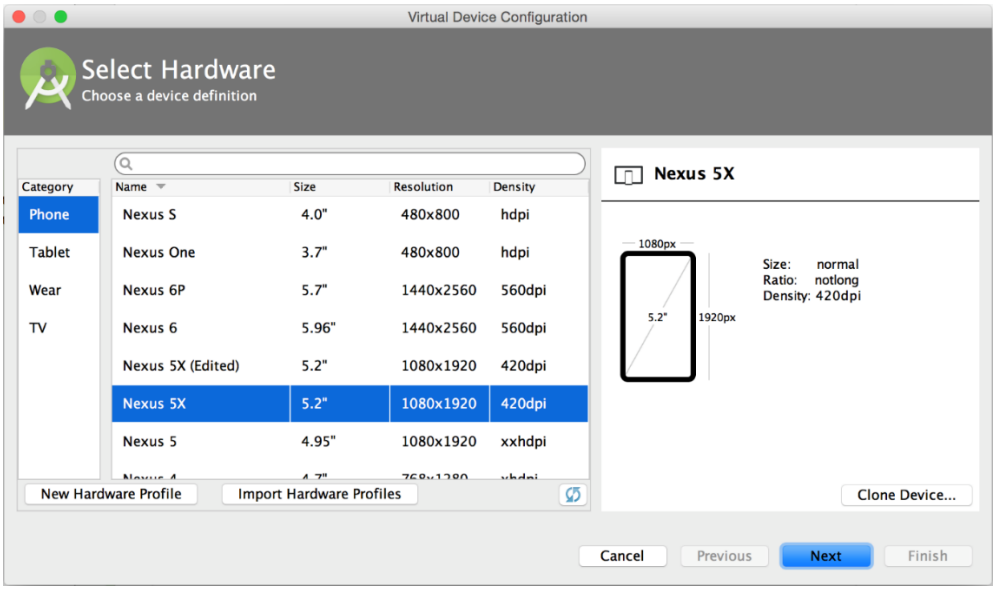


Fonte: https://developer.android.com/studio/run/managing-avds.html?hl=pt-br

Para criar uma nova AVD deve-se utilizar o botão Criar Virtual *Device* no AVD *manager*, abrindo assim a tela de configuração de perfil de *hardware* como na Figura 6 a seguir:

[13] Disponível em: <https://developer.android.com/studio/features.html?hl=pt-br>.

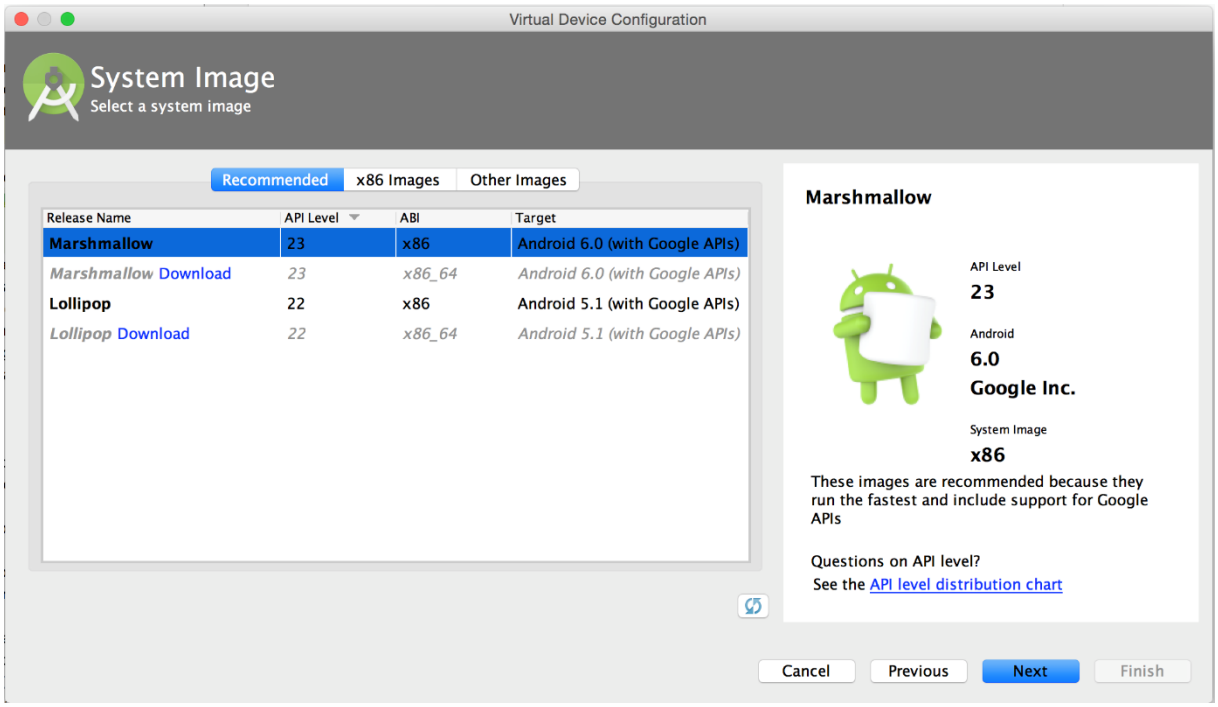
Figura 6: Escolha do tipo de dispositivo e Perfil de Hardware.



Fonte: https://developer.android.com/studio/run/managing-avds.html?hl=pt-br

Após escolhidas as definições de *hardware*, clicando em *next* (próximo) tem-se a opção de escolher qual versão do sistema o dispositivo virtual vai ter, terá versões que precisara de internet para ser feito o download, como na Figura 7 abaixo:

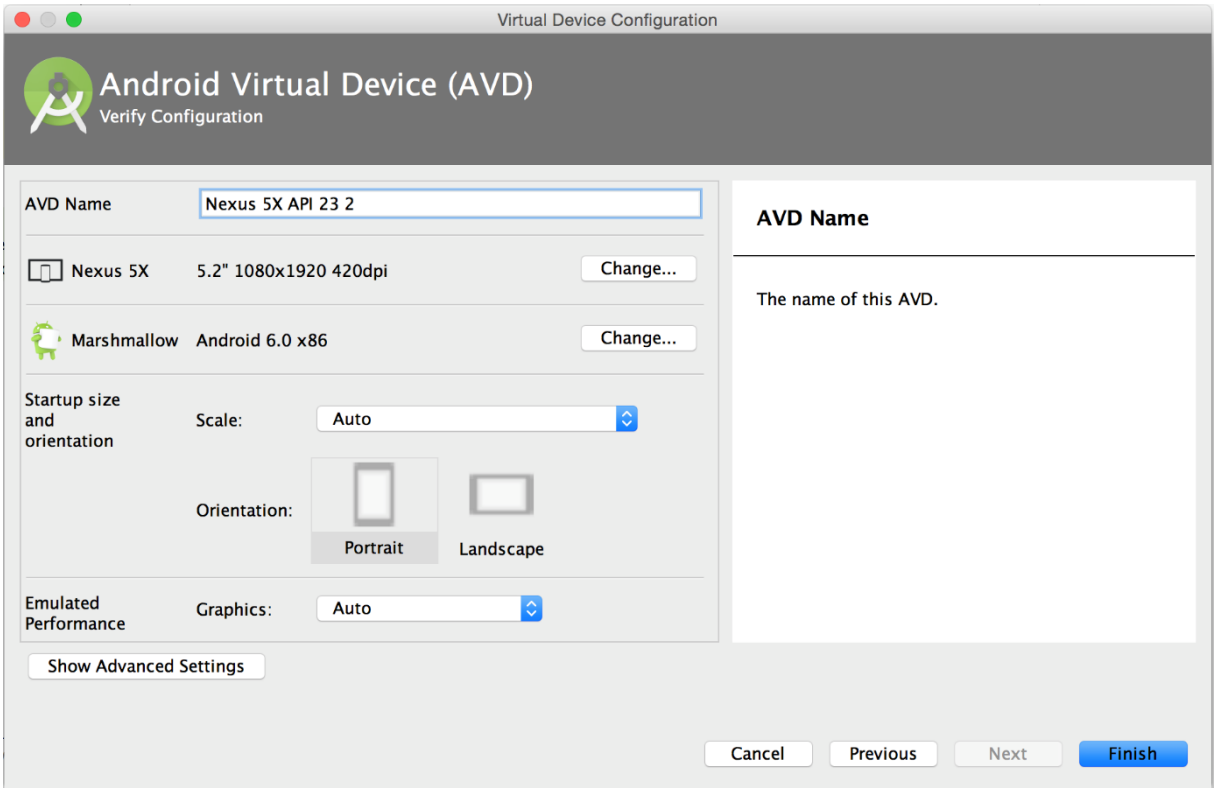
Figura 7: Seleção de Sistema a ser baixado e instalado.



Fonte: https://developer.android.com/studio/run/managing-avds.html?hl=pt-br

Selecionado a versão do sistema clica-se novamente em *next* (próximo) chegando à parte de verificar as opções que foram escolhidas, a fim de, se caso estiver algo errado, poderá ser alterado antes de concluir. Existe também a opção de configurações avançadas (*Show Advanced Settings*) que dará várias opções avançadas de personalização do dispositivo virtual. Abaixo (Figura 8) tela de verificar as configurações escolhidas:

Figura 8: Verificação das opções previamente escolhidas.



Fonte: https://developer.android.com/studio/run/managing-avds.html?hl=pt-br

Clicando em *Finish* (finalizar) tem-se o dispositivo virtual criado a partir das suas escolhas, já podendo ser iniciado e utilizado normalmente para os testes. Na figura 9 a seguir pode-se ver o dispositivo criado iniciado e pronto para uso.

Figura 9: Dispositivo Virtual Iniciado.

Fonte: https://developer.android.com/studio/run/emulator.html?hl=pt-br

Tudo isso incluso no Android Studio por padrão, só há necessidade de internet para algumas versões do sistema que ainda podem não terem sido transferidas para o computador como citado anteriormente.

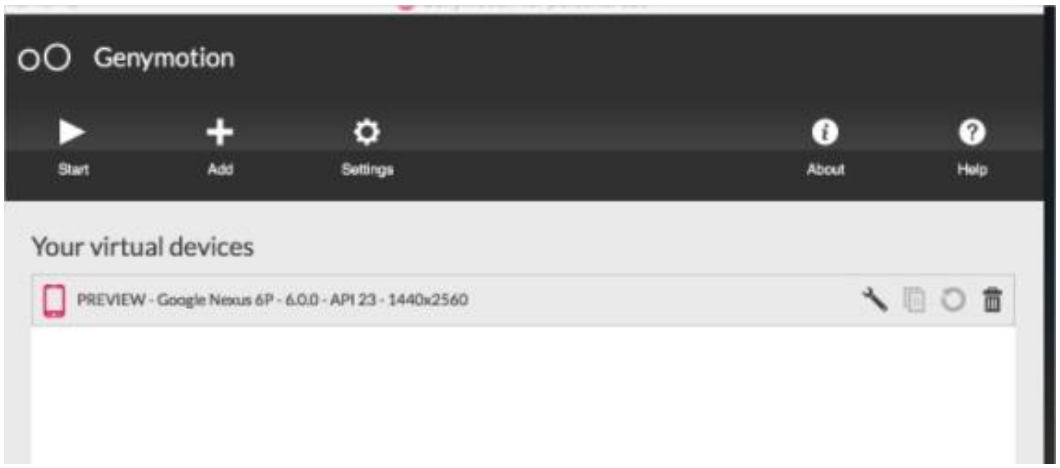
Já dentro do dispositivo virtual tem-se várias opções para simular o uso real de uma aplicação como se fosse realmente um smartphone, por exemplo, simular uma ligação, acessar os arquivos internos etc. A interação com o dispositivo é da mesma forma como se fosse um dispositivo real, mas usando o mouse e o teclado do próprio computador.[14]

A outra opção de emulador que o Android Studio dá é o *Genymotion*, porém a sua instalação deve ser feita de forma manual, mas a sua utilização segue o mesmo padrão de criação de dispositivos do AVD, pode-se criar dispositivos com personalização de tela e perfis de *hardware* específicos como no AVD *manager*, o

[14] Disponível em: <https://developer.android.com/studio/run/emulator.html?hl=pt-br>.

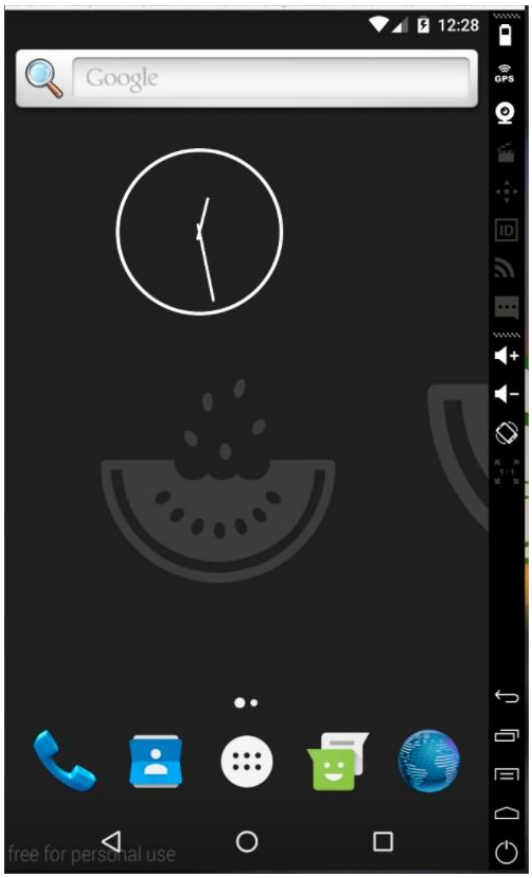
diferencial é que o *Genymotion* é de certa forma mais leve que o emulador padrão do Android Studio, principalmente em processadores da marca AMD.

Figura 10: Genymotion Device Manager.



Fonte: Do autor.

Figura 11: Dispositivo Virtual do Genymotion Iniciado.



Fonte: Do autor.

1.7 Engenharia de Software

Um *software* é um elemento lógico, desenvolvido e projetado, que em tese não se desgasta e na maioria das vezes é feito sob medida (PARREIRA JUNIOR, s/a.).
A maioria dos problemas referentes a um software é que sua estimativa de custos e prazos de desenvolvimento são imprecisos e que muitas vezes o software desenvolvido é insuficiente e não atende à demanda. Com a ciência destes problemas e visando a melhoria da qualidade, a produção de um software passou a ser vista como um produto comercializável, fruto de uma engenharia, a Engenharia de Software.

As ferramentas que a engenharia de software disponibiliza, permite a automatização nos processos e organização dos métodos de produção de um software. Há vários processos de software que se diferem na sua constituição, porém há atividades que conforme Sommerville (2007 apud PARREIRA JUNIOR, s/a, p. 11) são a especificação do software, o projeto e implementação, sua validação e evolução.

Conceituando as atividades, temos como a especificação do software o processo onde há a coleta de requisitos do software. Requisitos, segundo Paula Filho (2000, p.13), são as características que definem os critérios de aceitação de um produto de *software*. Geralmente essas coletas são feitas com o cliente e *stakeholders*[15] por meio de entrevistas e questionários, visando identificar quais serão os objetivos que devem ser alcançados pelo software ou mais especificamente, que ações ele precisa realizar e como são estruturados esses processos. Essas entrevistas assim como conceitua Parreira Junior (s/a, p. 13), deve-se ter um cuidado redobrado nesse estilo de coleta de dados pois as entrevistas podem ser feitas com pessoas erradas, em momentos errados e consequentemente obter respostas erradas. O entrevistador deve então, estar bem seguro do que irá fazer e ter um plano bem estruturado.

[15] Denominação utilizada para definir pessoas que de algum modo podem influenciar no sucesso ou insucesso do projeto, como donos do negócio, gerentes, desenvolvedores e demais colaboradores.

Após a coleta de requisitos, os mesmos devem ser agrupados para a criação do escopo do projeto, que conterá as especificações do mesmo de modo geral. Vale ressaltar que os requisitos podem ser alterados e com eles (como citado anteriormente) os custos e prazos podem se alterar, portanto deve-se ter uma atenção maior nessa fase para que futuramente não haja grandes empecilhos.

No desenvolvimento do software há ainda mais ramificações pois há modelos distintos para que o mesmo seja espelhado. Um desses modelos é o evolucionário, onde é construído um modelo básico funcional de software, apresentado ao cliente e a cada fase definida, este protótipo ganha mais funcionalidades. Este processo se repete até que o produto esteja concluído (DAVIS, 1991). Neste modelo são realizados vários testes ao longo do desenvolvimento e realizado alguns ajustes que podem ser solicitados pelo cliente de forma menos custosa.

Após a finalização do desenvolvimento e antes que a versão final seja entregue ao cliente documentos como termo de aceite, manuais e tutoriais de uso da aplicação. A última etapa, porém, é a que consome mais tempos e custo para o cliente pois acompanhar a evolução da aplicação consiste em deixa-la em constante atualização, manutenções, e realizar alterações que incrementarão a mesma até que sua vida útil seja atingida.

Todo o processo descrito, principalmente se for feito em equipe, deve ser devidamente documentado. Cada método escolhido para o desenvolvimento, como metodologias ágeis ou métodos mais tradicionais, têm os seus próprios modelos de documentos. Estes modelos não precisam, no entanto serem seguidos e aplicados à risca, pois podem ser customizados, sem que seu padrão de qualidade se perca, é claro, a fim de melhor atender a realidade da criação do *software*.

1.7.1 Modelos de Software

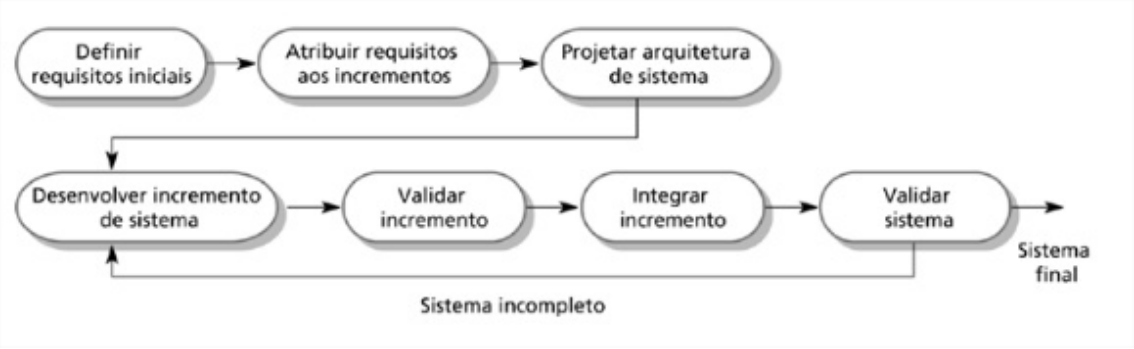
Ao iniciar a construção do software propriamente dito, a equipe desenvolvedora deverá definir também, em qual paradigma de desenvolvimento ele será feito. Conforme Sommerville (2007, p. 44-47) afirma, existem alguns modelos básicos que são:

- Modelo em Cascata – se trata de um modelo mais genérico, onde há um desenvolvimento linear e atividades que se simplificam em análise e definição dos requisitos, projeto do sistema, implementação, testes de unidade, teste de sistema e a manutenção. [16]

- Modelo Evolucionário – esse modelo se configura no desenvolvimento de parte do projeto e a apresentação ao cliente. A partir daí o programa é testado, e retorna a equipe de desenvolvimento caso haja alterações a serem feitas. Há atividades simultâneas sendo realizadas e diferentemente do modelo em Cascata, há uma flexibilidade maior ao realizar mudanças e o produto final entregue ao cliente de uma forma muito mais segura.[17]

Ainda há o modelo Iterativo e Incremental (SOMMERVILLE, 2007, p. 47), onde a entrega incremental é um meio de intermédio permitindo que o cliente/usuário identifique os serviços a serem fornecidos pelo sistema e definam com serão realizados e quais são os mais importantes e já obtenha parte da aplicação final para ser utilizada antes que o projeto seja concluído, conforme a figura 12 abaixo:

Figura 12: Atividades do modelo Iterativo e Incremental



Fonte: SOMMERVILLE, 2007, p. 46

Além de realizar atividades básicas como a definição dos requisitos, o projeto é dividido em partes que são codificadas e implementadas separadamente. A cada ciclo, após testes e validações, há a entrega de uma parte da aplicação em modelo.

[16] SOMMERVILLE, 2007, p. 44
[17] SOMMERVILLE, 2007, p. 45

de produção, para ser utilizado pelo cliente. Após essas atividades o ciclo recomeça e ao se aproximar da entrega final, a aplicação toma sua forma total sem que necessariamente o cliente aguarde um tempo muito grande para utilizar toda a aplicação. A aproximação entre a equipe de desenvolvimento e os *stakeholders,* as constantes validações, verificações e testes, faz com que o produto final seja mais fiel ao que foi idealizado no início do projeto.

1.7.2 Gestão da Qualidade

Outro tópico importante a ser frisado é a preocupação com a qualidade do software criado. Mesmo utilizando as melhores práticas para coleta de requisitos e modelos de construção, se a aplicação não se adequar as expectativas do cliente ele perde o seu valor. Segundo Garvin (1984 apud PRESSMAN, 2011, p. 359) “a qualidade é um conceito complexo e multifacetado” e isso se dá pelo fato da subjetividade presente neste item.

Em suma, Pressman (2011, p. 360) define a qualidade, mais especificamente a qualidade do Projeto de Software, sendo como “o grau de atendimento às funções e características especificadas no modelo de requisitos”. Ou seja, o primeiro passo para definir se o produto final possui qualidade aceitável, é verificando o quanto daquilo que foi solicitado, foi realmente atendido, enfatizando novamente a importância de uma coleta de requisitos bem estruturada.

Como é sabido que mudanças ocorrem no decorrer do projeto de da codificação do produto final, a gestão da qualidade não deve se ater apenas as fases iniciais do projeto, e deve então ser uma gestão ativa e deve possuir ramificações que verifiquem e atuem diretamente na qualidade do desempenho do produto final, recursos utilizados, confiabilidade, conformidade[18], durabilidade, facilidade de manutenção, estética do produto final e a percepção que o cliente têm da equipe/organização desenvolvedora e a reputação de produtos que já se encontram no mercado (GARVIN 1987 apud PRESSMAN, 2011, p. 361).

[18] Este termo diz respeito ao quanto o produto está de acordo com os padrões da organização e regras do projeto.

2. DESENVOLVIMENTO

Este capitulo tem como objetivo apresentar como foi realizado o desenvolvimento da aplicação, desde a coleta de requisitos até os testes realizados.

2.1 Desenvolvimento do Aplicativo

O presente projeto surgiu de um problema em especial: o retorno ao cliente de forma rápida e eficiente. Foram feitas algumas reuniões com os responsáveis da empresa e foram definidos os requisitos.

2.1.1 Levantamento de Requisitos

Os requisitos são de extrema importância já que é a partir deles que o projeto será construído, se bem definidos haverá um *software* totalmente adaptado a empresa gerando assim a satisfação das duas partes envolvidas e se não forem bem definidos provavelmente haverá um *software* que não se adapte à empresa.

Os requisitos são divididos entre requisitos funcionais e não funcionais. Os requisitos funcionais se caracterizam nos comportamentos que uma aplicação deverá conter a partir da iteração com seus usuários e já os requisitos não funcionais dizem respeito a certas características implícitas e comportamentos que a aplicação deverá conter. Utilizando da analogia, em um caixa eletrônico temos como exemplo os tipos de transações que são suportadas, o que depende de cada cartão, ou seja, cada

usuário, como um requisito funcional e a facilidade de uso, erros e tempo de resposta, como requisitos não funcionais (PAULA FILHO, 2000, p.13).

O levantamento dos requisitos, foi feito a partir da reunião deste pesquisador com os responsáveis, como citado anteriormente, e a partir da sua experiência própria na área, fazendo com que tais requisitos sejam definidos minuciosamente com poucas chances de erro.

Os requisitos foram definidos por Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF).

Os RF são requisitos que descrevem quais serão as funcionalidades que o software deverá possuir, conforme o quadro 1 a seguir:

Quadro 1: Requisitos Funcionais (RF).

Código	**Descrição Requisito**
RF1	Cada colaborador da empresa deverá ter o seu usuário de acesso único e exclusivo com senha para acesso.
RF2	A aplicação deverá salvar todos os dados do cliente em um banco de dados.
RF3	A aplicação deverá salvar todas as ordens de serviços em um banco de dados.
RF4	A tela inicial do aplicativo deverá mostrar todas as ordens de serviço com as informações básicas.
RF5	A aplicação deverá ter uma forma de atualização das ordens de serviço em forma de acompanhamentos.
RF6	A aplicação deverá salvar todos os acompanhamentos em um banco de dados.

Fonte: Do Autor.

Já os RNF são os requisitos que formam as características que o *software* apresentará, conforme o quadro 2 a seguir:

Quadro 2: Requisitos Não Funcionais (RNF).

Código	Descrição
RNF1	O usuário deverá usar um aparelho com o sistema Android 5.0 ou superior.
RNF2	A aplicação deverá ser simples e de fácil manuseio.
RNF3	As consultas deverão serem feitas de forma rápida.
RNF4	A aplicação deverá ter campo de auto completar buscando no banco os clientes cadastrados.
RNF5	A aplicação deverá exibir uma mensagem caso a internet não esteja ativa.

Fonte: Do Autor.

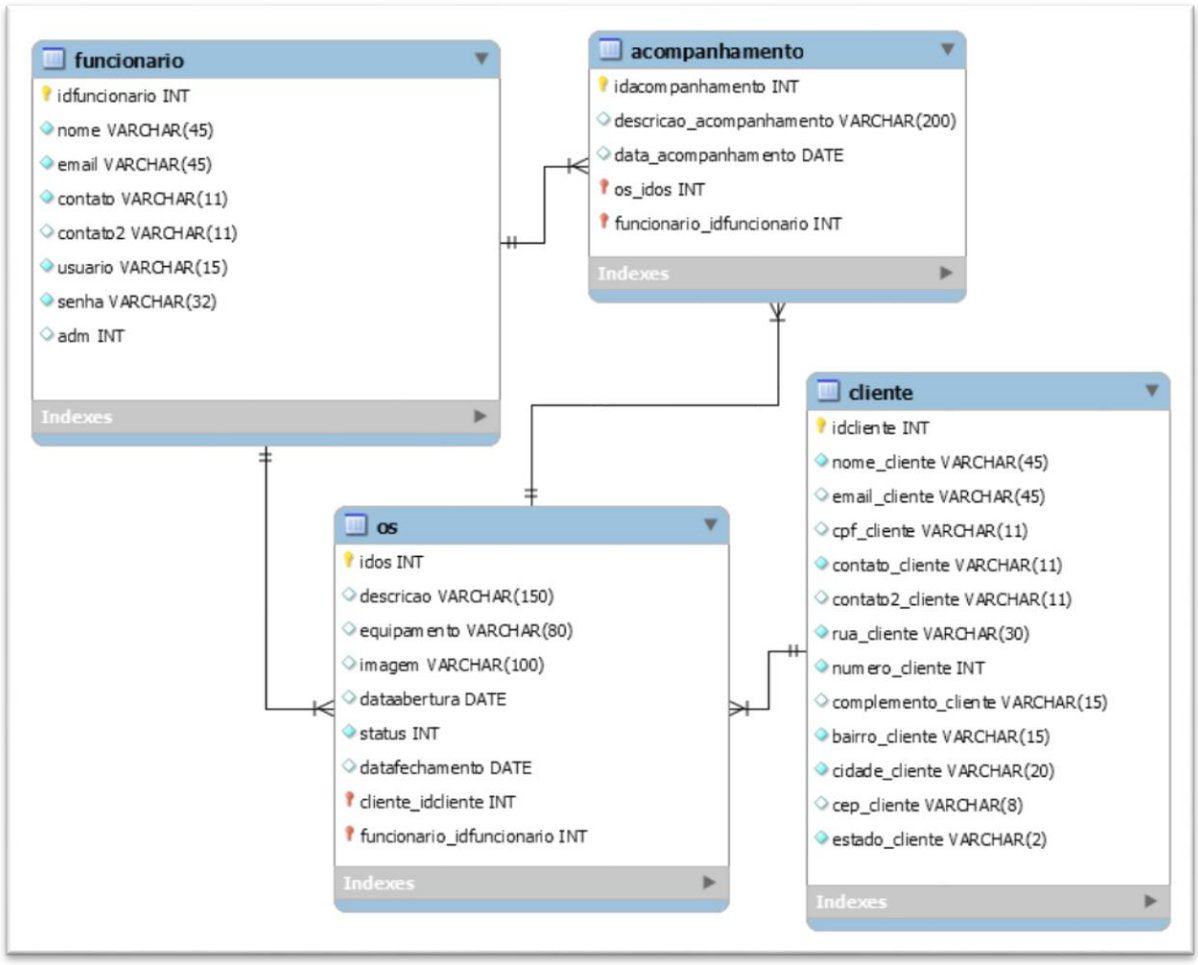
Ao finalizar a coleta de requisitos foi definido também que o nome do aplicativo seria: Controle O.S. ZN. E sua cor de base seria o vermelho conforme a logomarca da empresa.

2.1.2 Modelagem Banco de Dados

Com os requisitos definidos foi planejado e modelado um bando de dados no qual os dados de clientes, ordens de serviço, acompanhamentos e de usuários fossem salvos. A opção de SGBD escolhida foi o MySQL (MariaDB) que é rápido e fácil além de atualmente aceitar grande quantidade de dados.

Na modelagem ficaram definidas 4 tabelas conforme a figura a seguir:

Figura 13: Banco de dados.



Fonte: Do Autor.

A tabela “cliente” serve para salvar todos os dados do cliente, por exemplo, nome, email, telefone para contato etc.

A tabela “os” serve para salvar os dados das ordens de serviços, como o id do cliente do relacionamento com a outra tabela, a descrição do problema e o equipamento que foi deixado pelo cliente.

A tabela “acompanhamento” é para salvar os dados do acompanhamento das ordens de serviço.

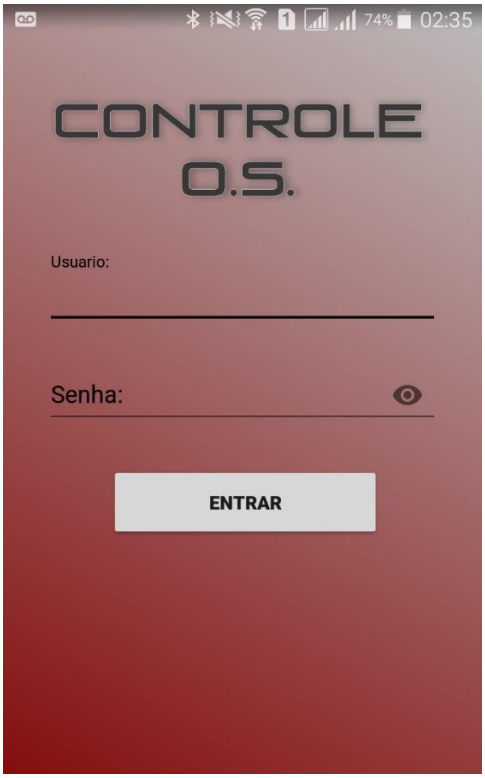
A tabela “funcionario” serve para a salvar os dados do colaborador da empresa, os dados de usuário e senha para o login no sistema serão salvos nesta tabela, atenção a coluna de “adm” que se for definida com o valor 1 o usuário é administrador do sistema e tem permissões especiais e se for 0 é um usuário padrão com acesso limitado.

Toda a manipulação do modelo foi feita pelo programa MySQL Workbench.

2.1.3 Desenvolvimento da Aplicação

Ao iniciar o desenvolvimento foram feitas as telas da aplicação (*Activitys*). A primeira tela do aplicativo desenvolvida foi a do login conforme abaixo na figura 14, nela o usuário deve fornecer o seu login, usuário e senha, e clicar em “entrar” para ter acesso ao sistema.

Figura 14: Tela de Login.



Fonte: Do Autor.

A próxima tela desenvolvida para ser aberta ao login ter sido feito com sucesso foi a da tela base de toda a aplicação, que é a com todas as ordens de serviço listadas, conforme a figura 15 abaixo, nela, cada ordem de serviço é exibida separada por cartões, e dentro dos cartões temos a informações básicas da ordem de serviço e 2 botões em forma de imagem. A lista de ordens de serviço é composta por uma *recyclerView,* que é uma lista feita por cartões pré-definidos sem limite de tamanho.

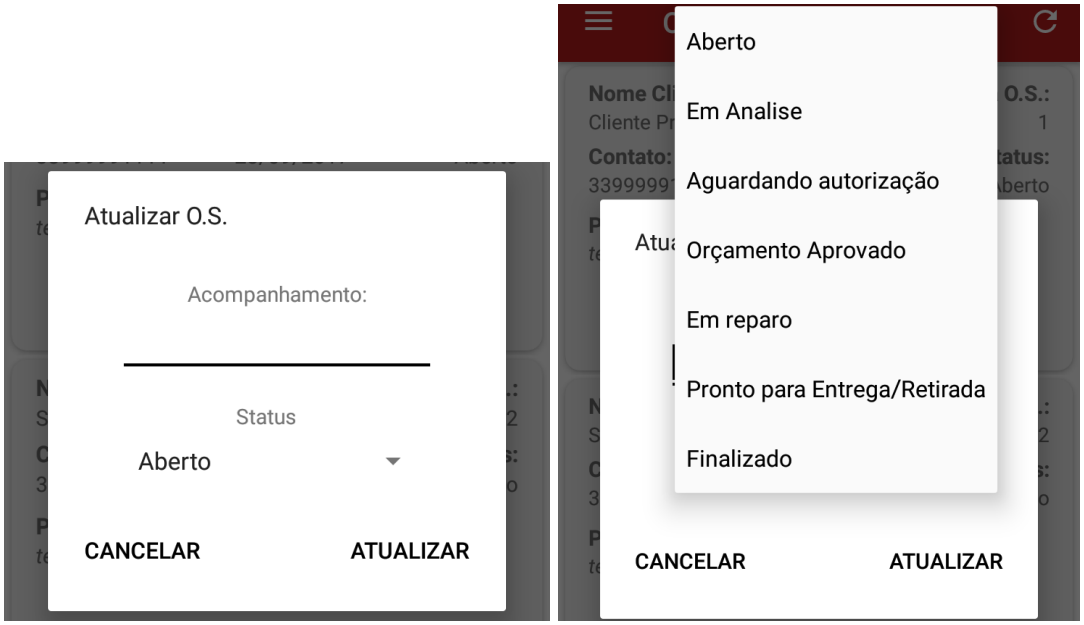
Figura 15: Tela Base da Aplicação.

Fonte: Do Autor.

Para os botões do cartão, foram implementadas as seguintes ações:

- Ao clicar no botão “+” do cartão da lista, irá apresentar uma tela de adição de acompanhamento e de alteração de status da ordem de serviço, conforme a figura 16. Tal tela é uma apresentação personalizada de um *alertDialog*, composta por um campo para adição de texto podendo ser várias linhas e no campo de Status, que é um *Spinner*, ao clicar terá as opções de status que a ordem de serviço pode estar.

Figura 16: Tela Adição de Acompanhamento e Status.



Fonte: Do Autor.

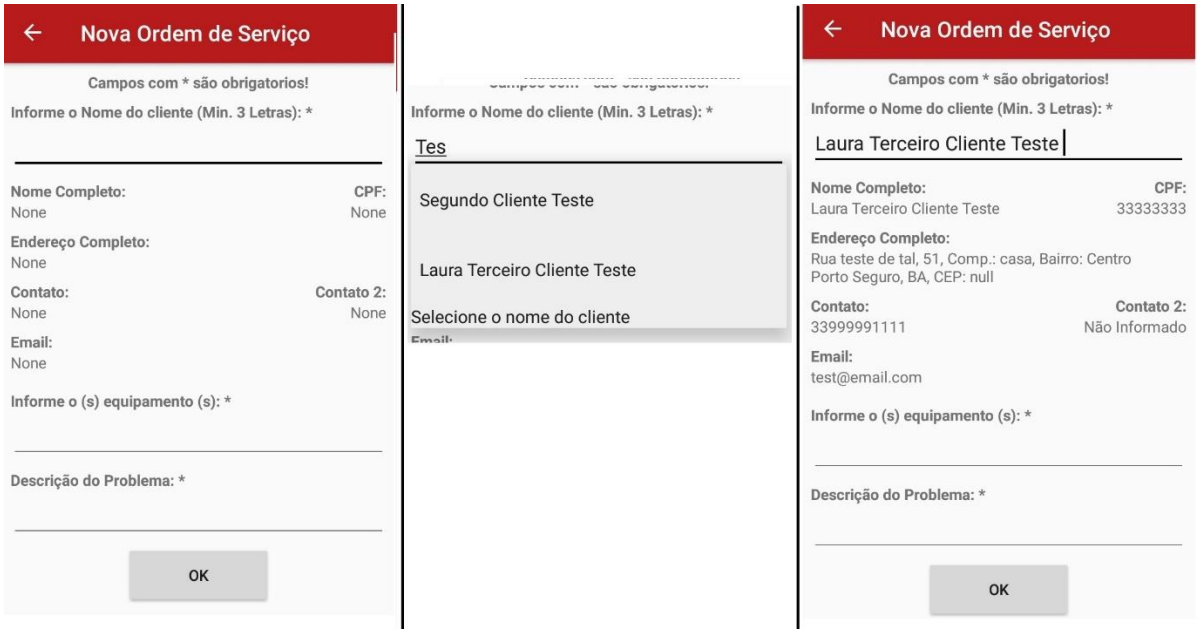
- Ao clicar no segundo botão do cartão da lista, que tem o ícone , será aberta uma nova tela com todas as informações daquela ordem de serviço conforme a figura 17 a seguir. Tal tela é composta por vários *TextView* e também há uma *recyclerView*, já que não há limite de acompanhamento por ordem de serviço, tal componente é utilizado para exibição de todos os acompanhamentos da ordem de serviço que foi aberta, ao clicar no acompanhamento é exibido também qual usuário do sistema (colaborador) que adicionou a mesma. Na figura só há um acompanhamento, mas pode ser adicionada mais a qualquer momento.

Figura 17: Tela Ordem de Serviço.

Fonte: Do Autor.

A próxima tela desenvolvida foi a de criar uma nova ordem de serviço, conforme na figura 18 abaixo, tal tela é composta por um campo *AutoCompleteTextView* que é usado para exibir as opções encontradas conforme o que foi digitado, há também alguns *TextView* para a exibição dos principais dados do cliente, a fim de confirmação com o mesmo na hora da criação da ordem e por fim há dois campos com a finalidade de serem completado com as informações do equipamento que foi deixado pelo cliente na empresa e a descrição do problema. Ressalta-se o campo de auto completar que ao ser selecionado um cliente, deverá informar os dados do mesmo nos *TextView* abaixo dele.

Figura 18: Tela Nova Ordem de Serviço.

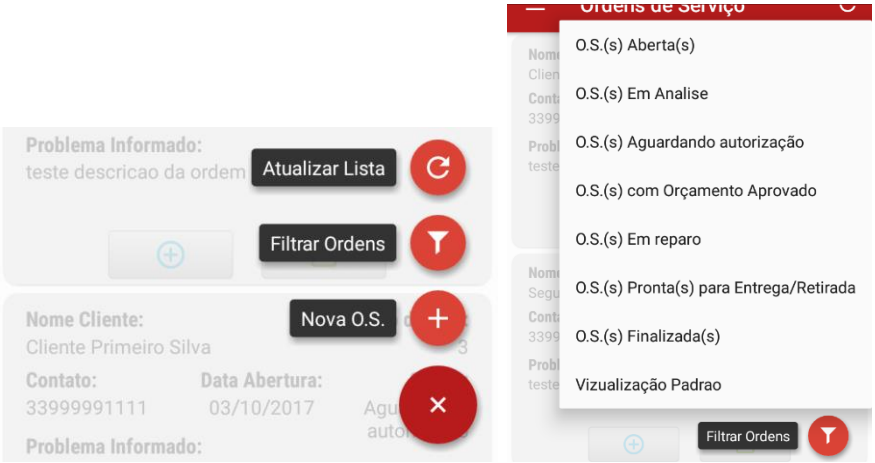


Fonte: Do Autor.

Ainda no teor de criação das telas foram feitos o que se pode chamar de detalhes, que podem fazer grande diferença na visão do usuário ao utilizar o mesmo, tais detalhes seriam como o botão flutuante que abre mais opções, conforme a figura 19 abaixo, dentre as opções temos:

- Atualizar Lista: que é para atualizar a lista das ordens de serviço;
- Filtrar Ordens: que é para filtrar a exibição das ordens de serviço pelo status da mesma.
- Nova O.S.: que é para a criação de uma nova ordem de serviço.

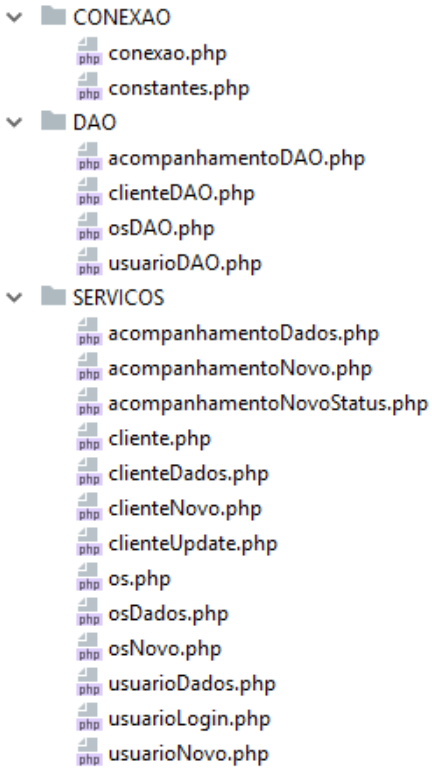
Figura 19: Botão Flutuante Expandido e Botão Filtrar.



Fonte: Do Autor.

Após o desenvolvimento de tais telas foram criados os arquivos PHP que são usados na manipulação dos dados, eles fazem a conexão da aplicação com o banco através do Web Service. Tais arquivos ficaram na seguinte estrutura conforme a figura 20 abaixo:

Figura 20: Estrutura Arquivos WebService.



Fonte: Do Autor.

Dentro da pasta “CONEXAO” tem-se dois arquivos sendo eles “conexão.php” e “constantes.php”. O arquivo constantes.php, como o Trecho de código 1 abaixo, é onde se define o tipo do banco, o host do banco de dados, o nome do banco, e o nome do usuário do banco e a senha.

Trecho de código 1: Exemplo Arquivo Constantes.

```
<?php
  define('DSN', 'mysql:host=127.0.0.1;dbname=osdb');
  define('DB_USER', 'root');
  define('DB_PASSWORD', 'senhadobanco');
```

Fonte: Do Autor.

Já o arquivo “conexão.php”, como o Trecho de código 2 abaixo, usa o arquivo de constantes, citado anteriormente, para se conectar ao banco de dados.

Trecho de código 2: Exemplo Arquivo Conexao.

```
<?php
require "constantes.php";
function conectar() {
    try {
        $pdo = new PDO(DSN, DB_USER, DB_PASSWORD);
    } catch (PDOException $e) {
        echo $e->getMessage();
    }
    return $pdo;
}
```

Fonte: Do Autor.

Os arquivos da pasta “DAO”, que é um acrônimo de *Data Access Object*, são os objetos de acesso a dados onde estão centralizadas todas as funções de cada tabela do banco. Um exemplo de função, como o Trecho de código 3 abaixo, é que ele recebe o “pdo” que é basicamente a conexão criada anteriormente e, neste caso, o id do cliente, tal função retorna para a aplicação todos os dados daquele id, logo, retorna todos os dados do cliente.

Trecho de código 3: Exemplo de Função de um Arquivo DAO.

```
function getClienteById($pdo, $id) {
    $stmt = $pdo->prepare("SELECT * FROM cliente WHERE
idcliente = :id");
    $stmt->bindValue(":id", $id);
    $stmt->execute();
    if ($stmt->rowCount()) {
        $linha = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return $linha;
    }
}
```

Fonte: Do Autor.

Esta função não é chamada a partir do nada, e sim chamada dos serviços que estão na pasta “SERVICOS”, nesta pasta estão todos os serviços disponíveis, cada serviço equivale a um retorno diferente, para a aplicação ou para o banco. Cada

serviço realiza solicitações, no meu caso, POST via rede através do protocolo HTTP. No exemplo a seguir (Trecho de código 4), recebe-se um id via POST da web, o serviço pega este id e passa para a função do arquivo DAO juntamente com o pdo que é a conexão. O retorno deste serviço e um objeto ou uma lista de objetos no formato JSON.

Trecho de código 4: Exemplo de Arquivo de Serviço.

```
<?php
require_once '../DAO/clienteDAO.php';
require_once '../CONEXAO/conexao.php';
$response = array();

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (isset($_POST['idcliente'])) {
        $pdo = conectar();
        if (getClienteById($pdo, $_POST['idcliente'])) {

            $cliente = getClienteById($pdo,
$_POST['idcliente']);
            $response['error'] = false;
            $response['cliente'] = $cliente;

        } else {
            $response['error'] = true;
            $response['message'] = "cliente invalido";
        }
    } else {
        $response['error'] = true;
        $response['message'] = "Preencha todos os campos";
    }
}
echo json_encode($response);
```

Fonte: Do Autor.

Passando um id de um cliente para o arquivo anterior haverá a resposta em forma de um objeto JSON conforme na figura 21 abaixo:

Figura 21: Objeto JSON retornado pelo Servidor.

```
{
    "error": false,
    "cliente": [
        {
            "idcliente": "1",
            "nome_cliente": "Cliente Primeiro Silva",
            "email_cliente": "huehue@hotmail.com",
            "cpf_cliente": "11111111",
            "contato_cliente": "33999991111",
            "contato2_cliente": "33999991111",
            "rua_cliente": "Rua tal tal tal",
            "numero_cliente": "547",
            "complemento_cliente": "Ap 304",
            "bairro_cliente": "Sao jacinto",
            "cidade_cliente": "Teofilo Otoni",
            "cep_cliente": "00000000",
            "estado_cliente": "MG"
        }
    ]
}
```


Fonte: Do Autor.

Após o desenvolvimento foi atribuído um ícone ao aplicativo e em seguida gerado o arquivo .apk, que é basicamente o arquivo de instalação do aplicativo, sendo necessário ter o sistema Android na versão 5.0 ou superior para funcionar.

2.1.4 Testes da Aplicação

Este subcapítulo tem a finalidade de realizar testes na aplicação e comprovar o objetivo do desenvolvimento do mesmo.

Na tela inicial, conforme citado anteriormente, o usuário da aplicação deverá informar o seu usuário de acesso juntamente com a senha e em seguida pressionar o botão entrar para ter acesso a todo o sistema.

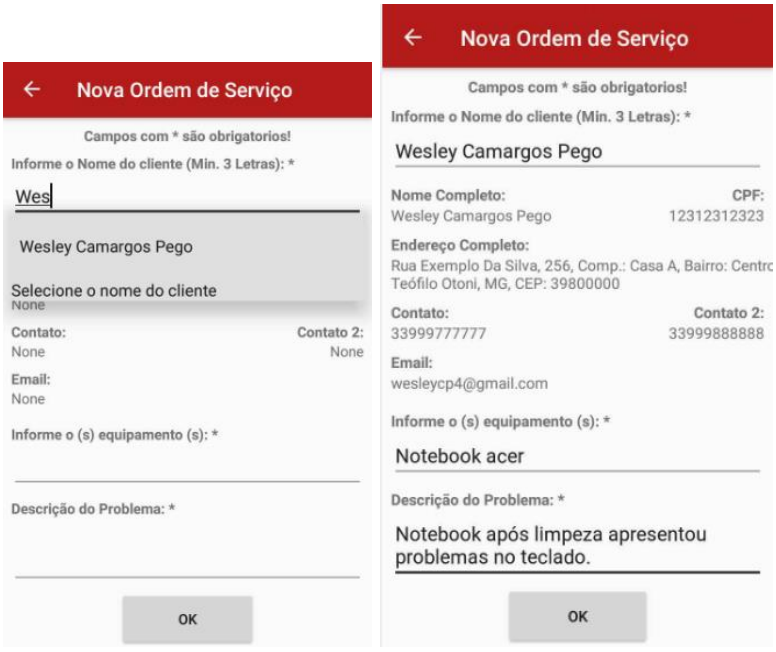
Dentro do sistema para criar uma nova ordem de serviço é necessário cadastrar o cliente previamente, conforme a figura 22 abaixo.

Figura 22: Cadastrar Novo cliente.

Fonte: Do Autor.

Na tela seguinte, após cadastro do cliente, será exibida a tela de cadastramento de nova ordem de serviço conforme a figura 23 a seguir:

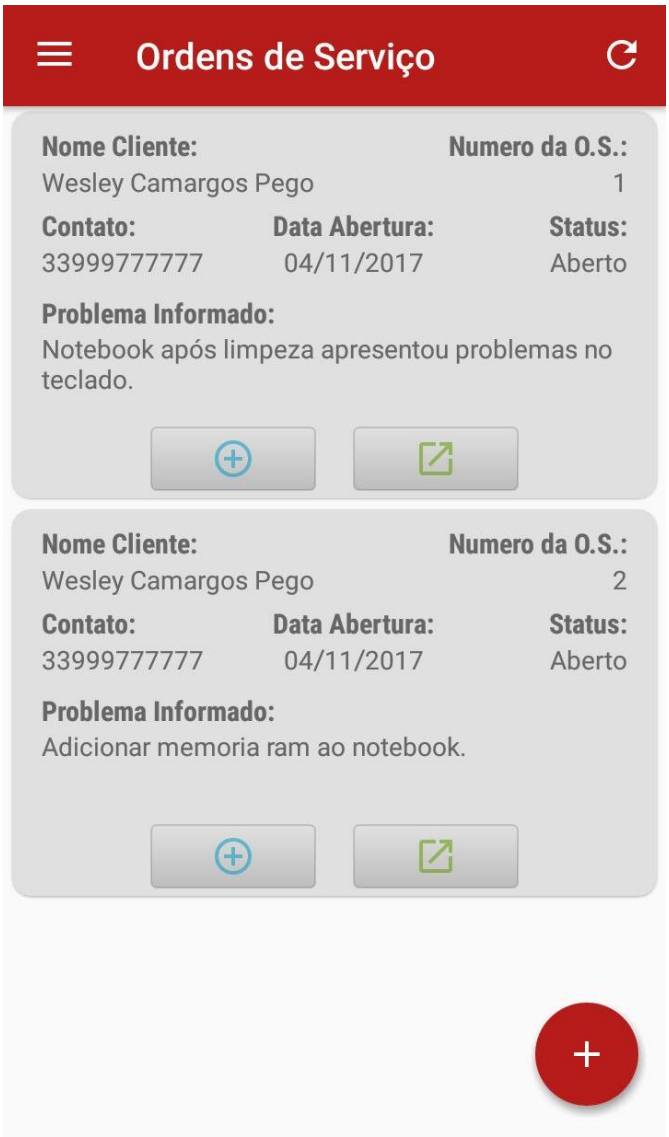
Figura 23: Cadastrar Nova Ordem de Serviço.



Fonte: Do Autor.

Conforme a figura 23 vista anteriormente ao começar a escrever o nome do cliente a aplicação já exibe os clientes que contém os caracteres informados. Ao selecionar o cliente na lista todos os dados do mesmo são carregados e exibidos abaixo a fim de confirmar com o cliente se está tudo correto, após confirmação, deve-se informar os equipamentos que estão sendo deixados para reparo e a descrição do problema. Ao clicar em ok a aplicação retorna para a tela com todas as ordens de serviço conforme a figura 24 abaixo.

Figura 24: Tela Inicial com todas as O.S.



Fonte: Do Autor.

3. RESULTADOS

Este capitulo tem como finalidade apresentar o retorno obtido ao consultar os colaboradores da Informática ZN após usarem a aplicação por alguns dias.

Uma empresa deve-se ter o mínimo em organização para se ter bons resultados na sua área, neste quesito a Informática ZN estava deixando a desejar, atrasos e falta de controle interno.

Assim, diante de tal problema, foi desenvolvido uma aplicação para a plataforma Android que foi testada pelos próprios colaboradores da empresa por alguns dias a fim de acertar os detalhes da mesma e de responderem um breve questionário.

O questionário organizado no quadro 4 abaixo, contém as perguntas realizadas aos colaboradores sobre a aplicação atribuindo para cada pergunta a quantidade de colaboradores que responderam sim ou não.

Quadro 3: Opinião dos colaboradores.

Pergunta	SIM	NÃO
O aplicativo é útil?	3	0
O aplicativo atendeu às expectativas?	3	0
O aplicativo é de fácil manuseio?	3	0
O aplicativo melhorou o tempo de retorno ao cliente?	3	0

Fonte: Do Autor.

Após o teste do aplicativo o mesmo apresentou certas vantagens que dentre elas estão:

- Pode ser acessado de qualquer lugar, desde que tenha internet disponível.

- Interface de fácil aprendizado e fixação.
- Aplicação leve.

O aplicativo marca o início da melhoria na infraestrutura da Informática ZN a fim de melhorar o tempo de resposta ao cliente e ao controle de equipamentos dos mesmos.

CONSIDERAÇÕES FINAIS

Esta monografia apresentou o desenvolvimento de uma aplicação Android, para controle de ordens de serviço na empresa Informática ZN.

Para se chegar ao objetivo geral da pesquisa foram, ao longo do trabalho, respondidos os objetivos específicos a seguir:

Coletar requisitos junto aos colaboradores:

O levantamento de requisitos se deu através de conversas com os gestores da empresa, a fim de colher as informações e requisitos necessários para o desenvolvimento.

Planejar, mapear e implementar o banco de dados em um sistema de gerenciamento de banco de dados:

Com os requisitos definidos, iniciou-se o desenvolvimento partindo inicialmente da criação da base de dados com seus respectivos campos utilizando o SGBD MySQL.

Testar e validar os módulos junto aos colaboradores:

A cada modulo funcional, os colaboradores poderiam testar e validar se estava tudo conforme o planejado.

Apresentar a aplicação completa aos colaboradores da empresa:

A apresentação da ferramenta foi necessária a fim de obter o melhor entendimento do funcionamento do mesmo e garantir a boa utilização do mesmo.

Implantar aplicação:

Após a apresentação a aplicação foi implantada e foi liberada a utilização do mesmo aos colaboradores.

Realizar a capacitação dos colaboradores para com a aplicação:

A capacitação dos colaboradores foi feita de forma rápida já que a aplicação tem uma interface limpa e de fácil entendimento.

Acompanhar a utilização da aplicação:

O acompanhamento é necessário a fim de que a aplicação possa ter melhorias e evoluir junto com a empresa.

Seguindo para a questão investigativa do trabalho que era baseado no desenvolvimento de um aplicativo de controle de ordens serviço para melhorar a qualidade do atendimento e a eficiência, foi realizado testes com os colaboradores com o objetivo de analisar a aplicação para enfim validar ou não as hipóteses.

Hipótese 0 – A empresa iria desistir do gerenciamento das ordens de serviço via aplicação mobile e iria optar pela aquisição de um livro de protocolos para registrar os atendimentos e registrar observações em uma planilha.

A hipótese acima foi invalidada, visto que uma empresa no mercado atual tem que se manter sempre atualizada a fim de oferecer o melhor serviço pelo menor tempo possível, sendo assim, o uso de uma planilha seria um retrocesso enorme, desde segurança a usabilidade.

Hipótese 1 – A implantação do aplicativo na empresa Informática ZN seria ideal, porém não supriria todos os requisitos e acabaria não atendendo a todas as demandas da empresa, acabaria não resolvendo os problemas da mesma.

A hipótese acima foi invalidada, visto que o desenvolvimento da aplicação foi feito moldado para a empresa de acordo com os requisitos coletados.

Hipótese 2 – A implantação do aplicativo na empresa Informática ZN seria ideal e teria uma boa aceitação por parte dos colaboradores e dos gestores da empresa, pois melhoraria todo processo de gestão da empresa e levaria a seus clientes um atendimento de melhor qualidade e eficiência.

A hipótese descrita acima foi validada, pois, durante a fase de implantação e testes foi possível analisar que o uso da aplicação melhorou todo o processo de controle da empresa, e ainda, foi possível perceber uma melhora no tempo de resposta ao cliente, já que a aplicação tem os dados do mesmo a fim de facilitar o contato com o cliente.

A conclusão desse trabalho resultou no desenvolvimento de um aplicativo para Android e o mesmo se mostrou ser de grande valia para Informática ZN, influenciando diretamente na forma de como a empresa trabalha, mudando a mesma para melhor.

Como implementações futuras o autor sugere o desenvolvimento de um módulo para o cliente acompanhar a ordem de serviço e conversar com o técnico ou atendente pelo próprio aplicativo e sugere também o desenvolvimento do sistema para

plataforma *web* com responsividade a fim de utilização em outros dispositivos já que o aplicativo está limitado atualmente a plataforma Android.

REFERÊNCIAS

ABLESON, Frank. *Introdução ao Desenvolvimento do Android*. 2009. Disponível em: <https://www.ibm.com/developerworks/br/library/os-android-devel/>. Acessado em 27 mai. 2017.

ALHIR, Sinan Si. *Methods & Tools: Practical knowledge source for software development professionals. Martinig & Associates*, Suíça, Ano 1, n, 7, p.11-18, set. 1989. Disponível em: <http://www.methodsandtools.com/mt/download.php?spring99>. Acessado em 23 out. 2017

CORRÊA, Eduardo. *Introdução ao formato JSON.* 2012. Disponível em: <http://www.devmedia.com.br/introducao-ao-formato-json/25275>. Acessado em 21 ago. 2017.

DAVID, Hailton. *Encapsulamento, Polimorfismo, Herança em Java*. 2009. Disponível em: <http://www.devmedia.com.br/encapsulamento-polimorfismo-heranca-em-java/12991>. Acessado em 28 ago, 2017.

DAVIS, Alan M. *Operational Prototyping: A new Development Approach*. IEEE Software, 1992. Disponível em: <http://ieeexplore.ieee.org/document/156899/>. Acessado em 28 ago, 2017.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de banco de dados*. 4. ed. São Paulo: Pearson-Addison-Wesley, 2005.

FELIPE, Eduardo. *Principais conceitos da Programação Orientada a Objetos*. 2015. Disponível em: <http://www.devmedia.com.br/principais-conceitos-da-programacao-orientada-a-objetos/32285>. Acessado em 28 ago, 2017.

FERRARI, Fabricio Augusto. *Crie Banco de Dados em MySQL*. São Paulo: Digerati Books, 2007.

GALPIN, Michael. *Trabalhando com XML no Android*. 2009. Disponível em: <https://www.ibm.com/developerworks/br/opensource/library/x-android/>. Acessado em 27 mai. 2017.

PARREIRA JUNIOR, Walteno Martins. *Engenharia de Software*. Ituiutaba: UEMG-FEIT, s/a. Notas de aula (ad usum privatum).

KAMADA, Terumi P. B. et al. *Análise das Plataformas de Desenvolvimento Mobile aplicados na Área Educacional, usando Android e Windows Phone.* CINTED - UFRGS. 2012. Disponível em <http://www.seer.ufrgs.br/index.php/renote/article/view/30916/19896>. Acessado em 28 mai. 2017.

LECHETA, Ricardo R. *Google Android – Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. 5. ed. São Paulo: Novatec, 2015.

MARINHO, Augusto. *Utilizando XML em aplicativos Android - Revista Mobile Magazine 44*. Disponível em: <http://www.devmedia.com.br/utilizando-xml-em-aplicativos-android-revista-mobile-magazine-44/25996>. Acessado em 28 mai. 2017.

MIRAS, Marcos. *Introdução ao MySQL*. 2008. Disponivel em: <https://www.vivaolinux.com.br/artigo/Introducao-ao-MySQL>. Acessado em 28 mai. 2017.

PAULA FILHO, Wilson de Pádua. *Engenharia de Software: Fundamentos, Métodos e Padrões*. 1. ed. Rio de Janeiro: LTC, 2000.

PERRY, J Steven. *Introdução à programação Java, Parte 1: Fundamentos da linguagem Java*. 2016. Disponível em: <https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/>. Acessado em 28 mai. 2017.

PEREIRA, Ana Paula. *O que é XML?*. 2009. Disponível em: <https://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm>. Acessado em 28 mai. 2017.

RICARDO, José. *Introdução ao MySQL*. Disponível em: <http://www.devmedia.com.br/introducao-ao-mysql/27799>. Acessado em 28 mai. 2017.

VINICIUS, Thiago. *Abstração, Encapsulamento e Herança: Pilares da POO em Java*. 2012. Disponível em: <http://www.devmedia.com.br/abstracao-encapsulamento-e-heranca-pilares-da-poo-em-java/26366>. Acessado em 28 ago, 2017.

Atividades. Disponível em: <https://developer.android.com/guide/components/activities.html?hl=pt-br>. Acessado em 28 ago, 2017.

Arquitetura da plataforma. Disponível em: <https://developer.android.com/guide/platform/index.html?hl=pt-br>. Acessado em 28 ago, 2017.

Conheça o Android Studio. Disponível em: <https://developer.android.com/studio/intro/index.html>. Acessado em 14 abr. 2017.

Manual do PHP. Disponível em: <https://secure.php.net/manual/pt_BR/>. Acessado em 20 set. 2017.

Executar aplicativos no Android Emulator. Disponível em: <https://developer.android.com/studio/run/emulator.html?hl=pt-br>. Acessado em 28 ago, 2017.

O que é a Tecnologia Java e porque preciso dela?. Disponível em: <http://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acessado em 28 mai. 2017.

O que é o PHP?. Disponível em: <https://secure.php.net/manual/pt_BR/intro-whatis.php>. Acessado em 20 set. 2017.

Tudo de que você precisa para criar aplicativos no Android. Disponível em: <https://developer.android.com/studio/features.html?hl=pt-br>. Acessado em 28 ago, 2017.

Types of SQL Statements. Disponível em: <https://docs.oracle.com/cd/B14117_01/server.101/b10759/statements_1001.htm>. Acessado em 28 ago, 2017.