**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA**

**ETEC DA ZONA LESTE**

**Técnico em Informática**

**Lucas Lima Martins Pereira**

**Mike Alves Spatafora**

**Wesley Vieira de Lima**

**BUY+:**

**São Paulo**

**2017Lucas Lima Martins Pereira**

**Mike Alves Spatafora**

**Wesley Vieira de Lima**

**BUY+:**

Trabalho de conclusão de curso apresentado ao Curso Técnico em Informática da ETEC Zona Leste, orientado pelo Prof. Rogério Bezerra Costa, como requisito final para obtenção do título de Técnico em Informática.

**São Paulo**

**2017**

**RESUMO**

O mundo está cada vez mais informatizado, todo dia novos aplicativos, gadgets e vestíveis, chegam às lojas, o ser humano está se limitando a realizar suas tarefas de modo mais prático e rápido, tudo deve ser feito do lugar onde está, tudo precisa ser resolvido em menos de cinco minutos, não é necessário pegar a fila do banco, ir ao supermercado, e essa informatização não tem previsão de encerramento, portanto cabe a todos desenvolver e consumir todos os tipos de aplicações, porque isso é o futuro.

Palavra chave: aplicativos, humano, futuro.

**ABSTRACT**

The world is increasingly computerized, every day new applications, gadgets and wearable, come to stores, the human being is confined to perform their tasks more practical and fast, everything must be done wherever is it, everything needs to be solved in less time as possible, it is not necessary to pick up the bank queue, go to the super market, and this computerization has no closure forecast, so it is up to all to develop and consume all kinds of applications, because that is the future.

Keyword: applications, human, future.

**LISTA DE FIGURAS**

[Figura 1 - Avanço do Android em 2017. 8](#_Toc500531858)

[Figura 2 - Arquitetura Android 9](#_Toc500531859)

[Figura 3 - Funcionamento do Cordova 11](#_Toc500531860)

[Figura 4 - Exemplo de WebView 12](#_Toc500531861)

[Figura 5 - tag <html> 13](#_Toc500531862)

[Figura 6 - tag <head> 13](#_Toc500531863)

[Figura 7 - tag <body> 14](#_Toc500531864)

[Figura 8 - Resultado das tags 15](#_Toc500531865)

[Figura 9 - tag título <h1> 16](#_Toc500531866)

[Figura 10 – Resultado tag título <h1> 16](#_Toc500531867)

[Figura 11 - tag de parágrafo <p> 17](#_Toc500531868)

[Figura 12 - Resultado tag <p> 17](#_Toc500531869)

[Figura 13 - Código para alteração de paleta de cores 19](#_Toc500531870)

[Figura 14 - Adicionando Recycler a um layout 20](#_Toc500531871)

[Figura 15 - Identificador conectado ao gerenciador de layout 21](#_Toc500531872)

[Figura 16 - Elevação de layout 22](#_Toc500531873)

[Figura 17 - Exemplo de DER 24](#_Toc500531874)

[Figura 18 - Diagrama de caso de uso 26](#_Toc500531875)

[Figura 19 - Diagrama de classe 26](#_Toc500531876)

[Figura 20 - Diagrama de pacotes 27](#_Toc500531877)

[Figura 21 - Diagrama de sequência 27](#_Toc500531878)

[Figura 22 - Diagrama de comunicação 28](#_Toc500531879)

[Figura 23 - Diagrama de máquinas de estado 28](#_Toc500531880)

[Figura 24 - Diagrama de atividade 29](#_Toc500531881)

[Figura 25 - Diagrama de visão geral 30](#_Toc500531882)

[Figura 26 - Diagrama de componentes 31](#_Toc500531883)

[Figura 27 - Diagrama de implantação 31](#_Toc500531884)

[Figura 28 - Diagrama de estrutura composta 32](#_Toc500531885)

[Figura 29 - Diagrama de tempo 32](#_Toc500531886)

**SUMÁRIO**

[1. INTRODUÇÃO 7](#_Toc500532432)

[2. REFERENCIAL TEÓRICO 8](#_Toc500532433)

[2.1. ANDROID 8](#_Toc500532434)

[2.2. APLICAÇÕES 10](#_Toc500532435)

[2.3. WEBVIEW 11](#_Toc500532436)

[2.4. HTML 12](#_Toc500532437)

[2.4.1. Estrutura HTML 13](#_Toc500532438)

[2.4.2. Demais Tags HTML 15](#_Toc500532439)

[2.5. Material Design 18](#_Toc500532440)

[2.5.1. Iniciando 18](#_Toc500532441)

[2.5.2. Criação 19](#_Toc500532442)

[2.5.3. Compatibilidade 22](#_Toc500532443)

[2.6. Banco de Dados 23](#_Toc500532444)

[2.6.1. MER 24](#_Toc500532445)

[2.6.2. DER 24](#_Toc500532446)

[2.6.3. UML 25](#_Toc500532447)

[REFERÊNCIAS 33](#_Toc500532448)

# INTRODUÇÃO

Os aplicativos surgiram a pouco tempo, desde então a vida dos usuários mudou, para tudo existe um aplicativo, não é necessário ir ao banco porque é possível pagar as contas pelo celular e ainda abrir uma conta, só tirar uma foto e enviar, se sentir vontade de comer algo fora do comum, é só encontrar o restaurante pelo aplicativo e pedir, hoje tudo está mais acessível, rápido e prático.

Alguns nos fornece o serviço de comparação de preços, pois outro item que os usuários pesquisam bastante é preço, alguns são voltados para hotéis, outros para passagens aéreas e outros para itens como eletrônicos e de consumo.

Muitas pessoas vão até o supermercado para fazer suas compras, seja ela grande ou pequena, realizadas nas lojas de varejo e não é sempre vantajoso para o cliente, pensando no consumidor e nos aplicativos de comparação de preços é possível realizar esse tipo de prestação de serviço.

Ao contrário do varejo o atacado é possível obter mais economia, muitas pessoas vão até o supermercado e não percebem que outras pessoas estão comprando o mesmo produto que elas, agregando o sistema de compra do atacado ao aplicativo, podemos criar um sistema onde as pessoas podem listar os produtos e disponibilizar a outros usuários do sistema, assim encontrando outras pessoas com a lista em comum e se encaminhar a um supermercado com vendas em atacado e comprar os produtos com maior economia.

# REFERENCIAL TEÓRICO

Nos capítulos a seguir se encontra o referencial teórico do projeto, todos os temas apresentados serão empregados e servem como base para a criação do sistema. Será apresentado os assuntos sobre Android e seus sistemas, Cordova e seu funcionamento, estruturação com HTML, o sistema que será utilizado como design e o banco de dados, com seus tipos e diagramas.

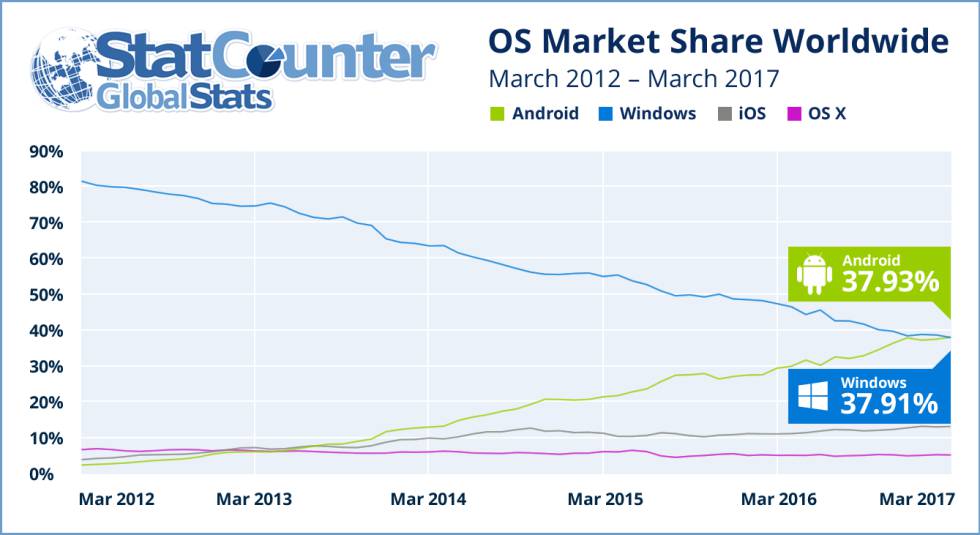
**COLOCAR ESTUDO SOBRE COMPRAS ‘**

## Android

O Android é um sistema operacional desenvolvido especialmente para dispositivos móveis, sua fundação ocorreu em 2003 pela Android Inc. e seu objetivo era fornecer um sistema aberto e flexível para todos os fabricantes, algo similar com o que Bill Gates pensou para o Windows, fornecendo primeiramente para IBM e posteriormente para outras fabricantes, em 2005 o Android passou a ser controlado pelo Google e essa mudança melhorou ainda mais o desenvolvimento (MONTEIRO, 1998).

No jornal eletrônico El País, Zuriarrain (2017) descreve o crescente aumento de celulares que possui o sistema Android, ele já ultrapassa o veterano Windows, 37,93% (Android) e 37,91% (Windows), isso mostra que as pessoas preferem utilizar o celular ao invés do computador e o avanço em hardware por parte das empresas mostra o interesse em dar todo suporte ao usuário neste quesito.

Figura 1 - Avanço do Android em 2017.



Fonte: (EL PAÍS 2017)

Weiszflog (2015) diz que arquitetura é uma organização interna dos componentes de um computador, com referência específica ao modo como se transmitem os dados, e explicando em outras palavras, a arquitetura empregada no Android é dividida em vários pavimentos que se comunicam entre si.

Figura 2 - Arquitetura Android



Fonte: (BURNETTE, 2008)

Na superfície dessa arquitetura encontramos as aplicações, tudo que o usuário é capaz de enxergar, como, o navegador, câmera, galeria, contatos e home (GOMES; FERNANDES; FERREIRA, 2012).

O framework de aplicações funciona como um organizador de tarefas, o usuário gera uma ação ao tocar em um aplicativo, essa ação se tornam atividades e o agrupamento dessas atividades resulta em uma tarefa, a cada nova tarefa, a anterior passa a ser executada em segundo plano, o usuário tem permissão para circular por essas atividades, continuando ou encerrando a tarefa, mas o framework pode encerrar uma tarefa automaticamente, se possuir muitas tarefas em segundo plano o sistema começa a encerrar as aplicações para recuperar memória (BORDIN, 2012).

O ciclo de vida das aplicações se dá ao framework, mas este possui vários departamentos e cada gerencia um tipo de tarefa, como por exemplo, a organização das pilhas de tarefas é de responsabilidade do *Activity Manager*, o *Package Manager* registra todas as aplicações, *Windows Manager* gerencia a tela do dispositivo e cria superfícies para as aplicações, *Telephony Manager* dá permissão a outras aplicações serviços relacionados da telefonia, *Resource Manager* permite o acesso a tudo que faz parte de um aplicativo mas não é código, como, bitmap e sons, *Notification Manager* avisa o usuário de algum evento e *Location Manager* que concede a localização do dispositivo para as aplicações (BORDIN, 2012).

Para que todas essas aplicações funcionem, a arquitetura disponibiliza uma biblioteca, ela é utilizada por diversos componentes dos sistemas, exemplo, cada parte do sistema utiliza uma linguagem de programação, então, em um departamento são armazenados as bibliotecas de C e C++, outro departamento da suporte ao áudio, vídeo e imagem, também existe um gerenciador que serve de intermediador ao display e que compõe as camadas de imagens 2D e 3D, e por último um banco de dados, o SQLite (BORDIN, 2012).

Como foi mencionado mais acima, o Linux já vem embarcado com muitas utilidades, principalmente para desenvolvedores, ao desenvolver algum algoritmo, sequência de códigos de um programa, o sistema operacional já disponibiliza um compilador, sistema que verifica o código e que também podemos chamar de máquina virtual, se todas as linhas estão escritas corretamente ele pode executar o programa na sequência, e isso não é diferente no Android, grande parte dos aplicativos disponibilizados para o sistema são escritos na linguagem de programação Java, ela possui um compilador embarcado, ou seja, este processo é realizado separadamente do sistema mas tudo isso acontece dentro do Android Runtime que também disponibiliza uma máquina virtual, chamada Dalvik VM, este é destinado a todos os aplicativos que não são construídos em Java (GOMES; FERNANDES; FERREIRA, 2012).

## APLICAÇÕES

Ao pensar em criar um aplicativo para uma plataforma devemos analisar qual delas será mais vantajoso ingressar primeiro, entre as plataformas mais populares está o Android e iOS, cada uma delas possui uma singularidade, o Android é escrito em Java e iOS utiliza o Objective-C e escrever aplicações para cada uma é complexo e exige um conhecimento específico (LOPES, 2016).

A linguagem Web caminha no sentido oposto dessa fragmentação das plataformas, ela oferece o HTML, CSS e JavaScript, tudo isso é utilizado para os navegadores porém não são um aplicativo instalável que está disponível nas lojas oficiais (LOPES, 2016).

O Cordova surgiu basicamente para suprir essa necessidade, ele funciona como uma casca, essa casca utiliza toda linguagem Web além de oferecer um serviço a mais, o WebView, esse serviço funciona como um navegador para o aplicativo e executa todo código, ele também tem acesso a todos os drivers do smartphone, resultando em um aplicativo multiplataforma ou aplicação híbrida (LOPES, 2016).

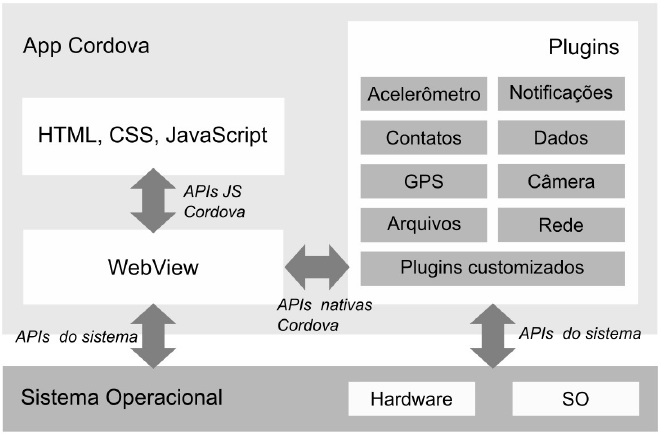
## WEBVIEW

Todos sistemas operacionais atuais do Android e iOS possuem o WebView, ele funciona como um navegador escondido, esperando ser chamado, um exemplo do WebView é o funcionamento de páginas quando o usuário está nas redes sociais, o Twitter não abre o navegador e sim o WebView, ele é como um motor de renderização de HTML (LOPES, 2016).

Muitos aplicativos utilizam o utilizam para exibir alguns conteúdos, como o Facebook, porém o Cordova melhorou todo o sistema. Toda interface de usuário, toda aplicação será feita em HTML, CSS e JavaScript e executada dentro do WebView (LOPES, 2016).

A capacidade do Cordova vai muito além, através de uma API JavaScript, biblioteca da linguagem JavaScript, é possível acessar recursos nativos da plataforma, cada um desses recursos está encapsulado em um plugin diferente, então se é necessário ter acesso aos contatos do aparelho é só instalar o plugin *cordova-plugin-contacts* e a aplicação terá acesso aos contatos tanta da plataforma Android quanto do iOS (LOPES, 2016).

Figura 3 - Funcionamento do Cordova



Fonte: (**Aplicações mobile híbridas com Cordova e PhoneGap, p. 120)**

Figura - Exemplo de WebView



**Fonte: (Autoria própria, 2017)**

## HTML

O HyperText Markup Language (HTML) é uma linguagem Web, mais precisamente uma linguagem de estruturação, a linguagem surgiu com o propósito de padronizar o modo como os navegadores funcionavam, o problema de fragmentação não é novo, esse padrão ocorreu em meados de 1990 e vem se desenvolvendo até os dias atuais, o HTML se encontra na 5° geração (LOPES; KOMURA, 2011).

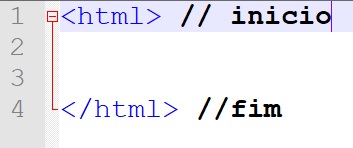
A única linguagem que o navegador consegue entender é o HTML, como mencionado acima, está linguagem é responsável pela estruturação da página, o nome dessas estruturas são chamadas de *tags,* essas tags tem a seguinte sintaxe <nometag> e cada uma possui uma função dentro do código (CAELUM, 2016).

### Estrutura HTML

Segundo CAELUM (2016, p.7), “Um documento HTML válido precisa seguir obrigatoriamente a estrutura composta pelas tags <html>, <head> e <body> e a instrução <!DOCTYPE>.”

A tag <html> é responsável por delimitar o código, ao comparar com algo mais próximo a língua portuguesa seria como, início e fim do código (CAELUM, 2016).

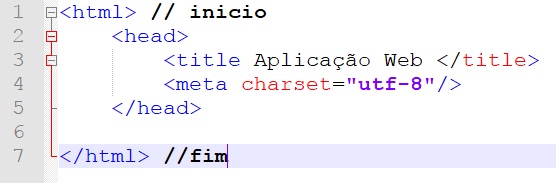
Figura 5 - tag <html>



Fonte: **(Autoria própria, 2017)**

A tag seguinte é a <head> nela é inserido dados que são importantes para o navegador, como título e a codificação da língua que será exibida. A língua portuguesa utiliza a codificação UTF-8 (CAELUM, 2016).

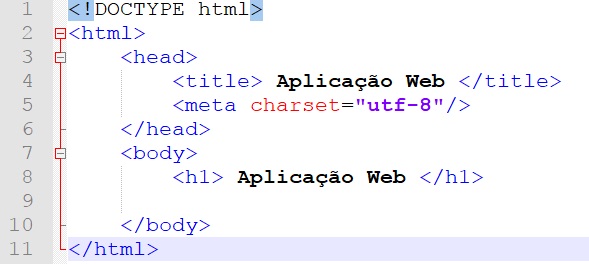
Figura 6 - tag <head>



Fonte: **(Autoria própria, 2017)**

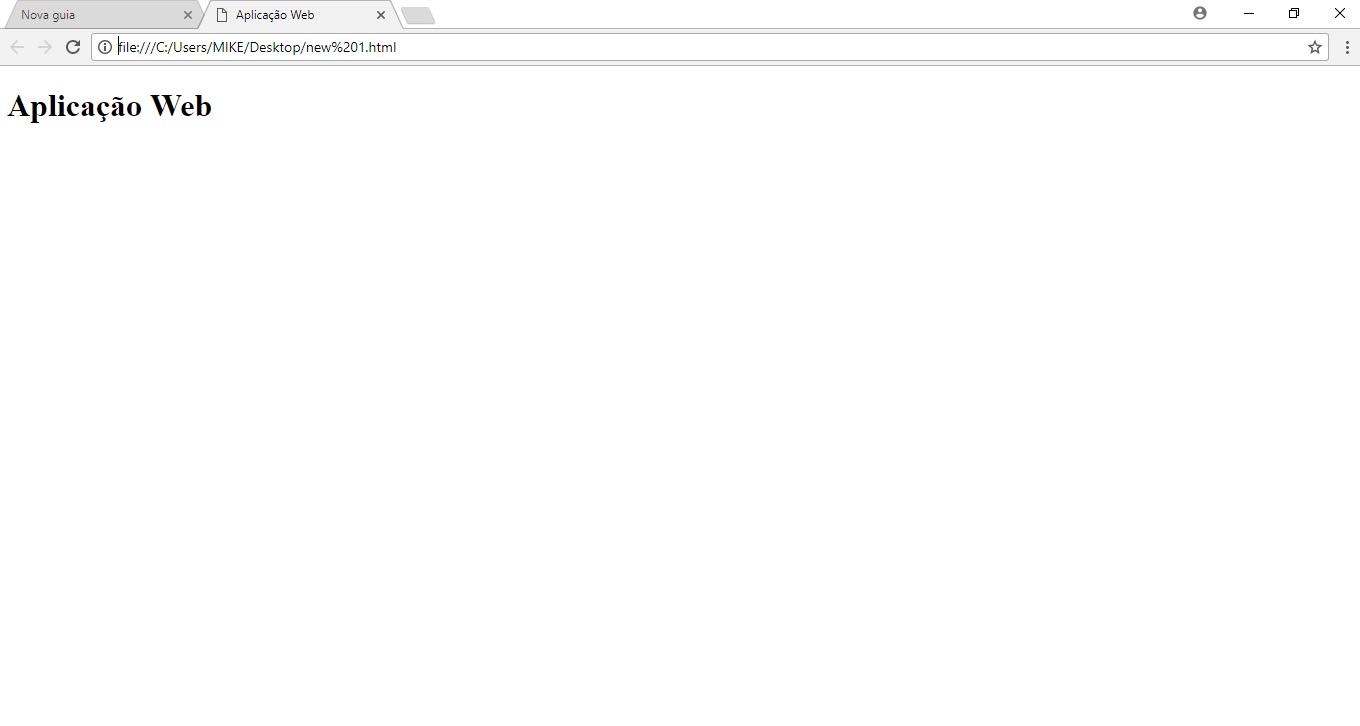
Após inserir as tags <html> e <head> é necessário inserir a última tag obrigatória em um documento HTML, a <body>, esta é responsável por abrigar todo o corpo do código, tudo que é acrescentado dentro do <body> será apresentado pelo navegador, além disso é importante comunicar qual versão do HTML o navegador deve compilar, ou seja, executar, como a internet não utiliza versões mais antigas então é acrescentado o <!DOCTYPE html>, este é o identificador que fará com que o código seja executado pelo HTML5 (CAELUM, 2016).

Figura 7 - tag <body>



Fonte: **(Autoria própria, 2017)**

**Figura 8 - Resultado das tags**



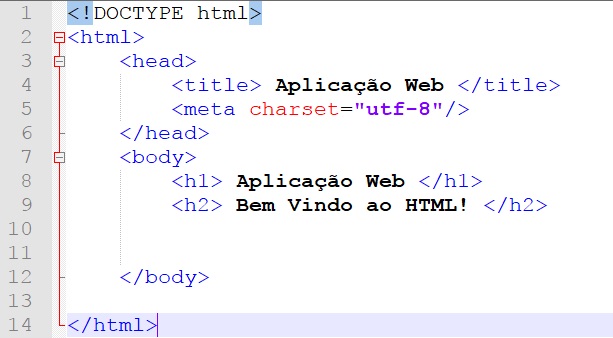
Fonte: **(Autoria própria, 2017)**

### Demais Tags HTML

O HTML é composto por mais tags, cada uma possui uma função e um significado, dentre elas estão as tags de títulos, parágrafo e ênfase (CAELUM, 2016).

A tag de títulos é representado por <h1>, também chamado de heading, ele possui graus de relevância começando pela <h1> até <h6>, sendo <h1> o título principal e <h6> o menor (CAELUM, 2016).

Figura - tag título <h1>



Fonte: **(Autoria própria, 2017)**

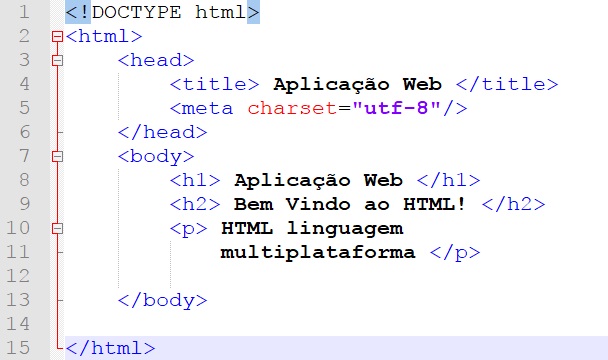
Figura 10 – Resultado tag título <h1>



Fonte: Autoria própria

Para construir um parágrafo é utilizado a tag <p>, se for necessário escrever vários parágrafos é utilizado várias tags para identificar o começo e o fim de cada um (CAELUM, 2016).

Figura 11 - tag de parágrafo <p>



Fonte: **(Autoria própria, 2017)**

Figura - Resultado tag <p>



Fonte: **(Autoria própria, 2017)**

## Material Design

O Material Design é um guia abrangente para design visual, de movimento e de interação para diversas plataformas e dispositivos. O Android agora é compatível com aplicativos do Material Design. Para usar o Material Design nos aplicativos Android, siga as orientações definidas nas e use os novos componentes e funcionalidades disponíveis no Android 5.0 (API de nível 21) e em posteriores (GOOGLE, 2017).

*MDL torna fácil adicionar o estilo “*material design*” aos seus websites. (…) É um framework agnóstico, isto é, significa que pode ser usado com quaisquer uma das ferramentas que surgem rapidamente no horizonte front-end. MDL é leve de carregar e estritamente focado em aplicar o “*material design*” aos websites.*

O Android fornece os seguintes elementos para criar aplicativos do Material Design:

* Um novo tema.
* Novos widgets para vistas complexas.
* Novas APIs para sombras e animações personalizadas.

### 2.5.1. Iniciando

Antes de iniciar a aplicação é necessário primeiro rever as especificações do Material Design, depois aplique o tema do Material e crie os layouts com as orientações do Material Design, altere as elevações para inserção de sombras, utilize os widgets do sistema para as listas e cartões e personalize as animações (GOOGLE, 2017).

**2.5.2. Criação**

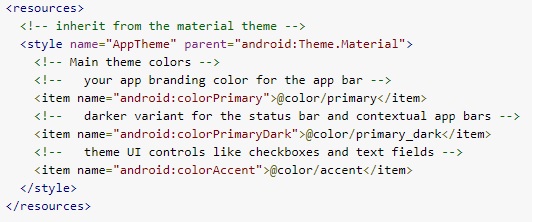
**2.5.2.1. Tema**

O tema do Material é definido como:

* @android:style/Theme.Material (versão escura)
* @android:style/Theme.Material.Light (versão clara)
* @android:style/Theme.Material.Light.DarkActionBar

É possível personalizar a aparência do tema do Material com uma paleta de cores, para isso se utiliza os comados da figura (CORDEIRO, 2016).

Figura 13 - Código para alteração de paleta de cores



Fonte: (GOOGLE, 2017).

#### 2.5.2.2. Barra de tarefas

Para definir uma cor personalizada para a barra de status, use o atributo *android:statusBarColor*  ao ampliar o tema do Material. Por padrão, *android:statusBarColor* é a mesma cor de *android:colorPrimaryDark* (GOOGLE, 2017).

**2.5.2.3. Listas e cartões**

Para criação de páginas com grandes conjuntos de dados que podem ser rolados se utiliza o *Widget RecyclerView*. Para a utilização do *RecyclerView* é necessário especificar um adaptador e um gerenciador de layout, para criação do adaptador, amplie a classe *RecyclerView.Adapter*. O gerenciador de layout insere as visualizações dos itens dento do *RecyclerView* (GOOGLE, 2017).

* *LinearLayoutManager* exibe itens em uma lista de rolagem vertical ou horizontal.
* *GridLayoutManager* exibe itens em uma grade.
* *StaggeredGridLayoutManager* exibe itens em uma grade escalonada.

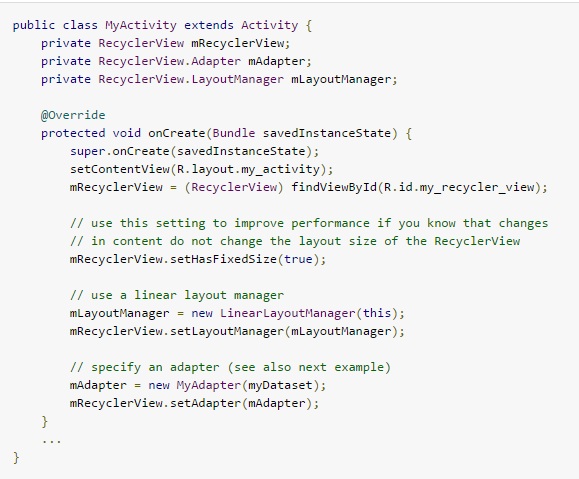
Figura 14 - Adicionando Recycler a um layout



Fonte: (GOOGLE, 2017)

Após a adicionar *RecyclerView* ao layout, utilize o identificador para o objeto, conecte-o a um gerenciador de layout e anexe um adaptador para que os dados sejam exibidos (GOOGLE, 2017).

**Figura 15 - Identificador conectado ao gerenciador de layout**



Fonte: (GOOGLE, 2017)

#### 2.5.2.4. **Elevação**

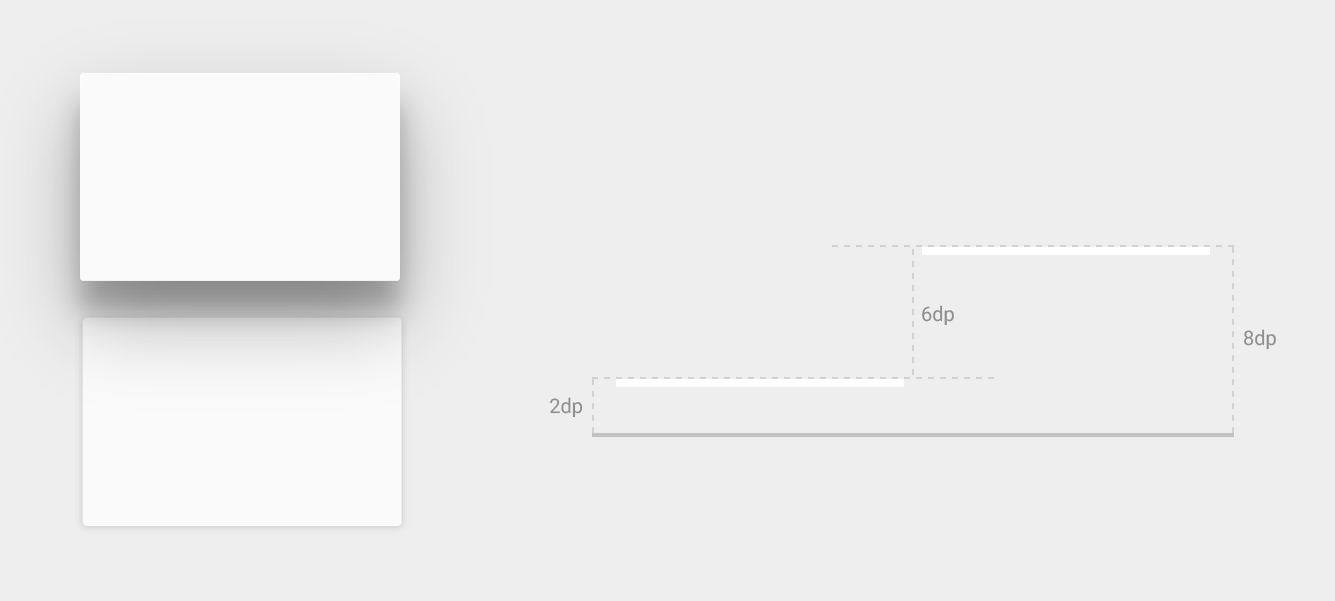
O valor de Z para uma visualização tem dois componentes:

* Elevação: O componente estático.
* Movimentação: O componente dinâmico usado para animações.

Z = elevation + translationZ

(GOOGLE, 2017).

Figura 16 - Elevação de layout



Fonte: (Developer Adroid, 2017)

Para alterar a elevação do layout, utilize o atributo *android:elevation*. Para definir a elevação de uma atividade, use o método *View.setElevation()*. Para definir a elevação de uma visualização, use o método *[View.setTranslationZ()](https://developer.android.com/reference/android/view/View.html" \l "setTranslationZ(float))* (GOOGLE, 2017).

#### 2.5.2.5. Contorno de sombras

Para definir um contorno personalizado, amplie a classe *ViewOutlineProvider*, neutralize o método *getOutline()* e atribua um nono contorno no método *View.setOutlineProvider()* (GOOGLE, 2017).

#### 2.5.2.6 Recortes

Para recortar a área de um contorno utilize o método *View.serClipToOutLine()* ou o atributo *andoird:clipToOutLine*. Para recortar como desenhável, defina o desenhável como o segundo plano da visualização e chame o método *View.SetClipToOutline()* (GOOGLE, 2017).

### 2.5.3. Compatibilidade

Para configurar o aplicativo para que ele use temas antigos para que o mesmo seja compatível com versões anteriores do Android, defina um tema antigo em *re/values/styles.xml*, defina um tema com o mesmo nome herdado do tema em res/values-v21/styles.xml e por fim defina o tema como o tema do aplicativo no arquivo manifesto (GOOGLE, 2017).

## Banco de Dados

Segundo Date (2004), um banco de dados é um sistema de armazenamento, controle, manipulação e pesquisa de dados que tenham um sentido ou significado para o usuário, individuo ou organização que o sistema deve servir.

Silberschatz, Korth e Sudarshan (2006) dizem que desde que a humanidade começou a usar dados e informações a seu favor para se adaptar ao ambiente ou para adaptá-lo para a sobrevivência da espécie, foi essencial que criássemos sistemas para organizar dados, como a matemática para organizarmos grandezas numéricas, ou a língua escrita, para que ideias e informações fossem transmitidas logicamente, mas mesmo assim esses dois exemplos são muito inseguros, pois quando se é necessário esconder informações que podem dar vantagens extras e injustas para terceiros, elas quase não tem nenhuma segurança, e por isso sempre tentamos criar novos meios de esconder e controlar o acesso a informações e dados, como a criptografia, que pode ser usada para repassar informações sem a chance dela ser usada por terceiros que não teriam a necessidade ou a autorização para acessa-los, mas mesmo usando criptografia em puros textos, ou seja, textos inteiros, é possível com a tecnologia atual dos computadores burlar tais sistemas de criptografia simplórios, pois esses meios podem ser facilmente quebrados por qualquer computador com poder de processamento razoável, e por isso atualmente se é usado um sistema muito mais seguro, com criptografia de acesso não linear muito mais eficiente quando se trata de guardar dados chamado Sistema de Gerenciamento de Banco de Dados (SGBD).

Um Sistema de Gerenciamento de Banco de Dados (SGBD), é um sistema de armazenamento de dados que os guarda e gerencia, para que seja possível de uma forma simples, fácil e eficiente, a recuperação desses dados, e para isso, os bancos de dados definem estruturas para armazenar esses grandes blocos de informação de um modo que se os dados precisarem ser acessados por dois usuários ao mesmo tempo, o sistema evite retornar resultados nulos garantindo a segurança dos dados e do sistema, principalmente contra falhas e acessos não autorizados (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

Elmasri e Navathe (2011) citam que os sistemas de banco de dados utilizam uma técnica de modelagem de bancos chamada de Modelo Entidade Relacionamento (MER) e Diagrama Entidade Relacionamento (DER), que faz com que suas tabelas tenham uma representação gráfica de suas relações, facilitando o entendimento dos requisitos que o banco de dados deverá ter e como deverá funcionar.

### MER

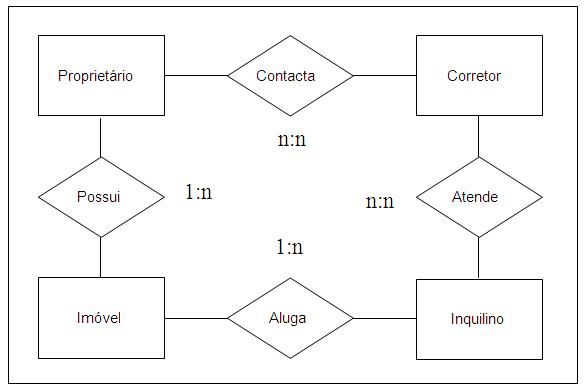
É um sistema de modelagem abstrato que procura descrever as classes, suas características, e suas relações de forma que demonstrem suas dependências, além de que um Modelo Entidade-Relacionamento (MER), precisa ser descrito de um modo conceitual para que seja interpretado sempre da mesma maneira para facilitar o entendimento de como o sistema deverá funcionar (THIZON, 2011).

### DER

Como Thizon (2011) explica, diferentemente do Modelo Entidade-Relacionamento, o Diagrama Entidade-Relacionamento procura representar um sistema seguindo certas regras, para que seja facilitado o entendimento dos requisitos e a comunicação dos integrantes que irão desenvolver o sistema, que no caso do MER, que pode dificultar o entendimento do funcionamento que o sistema deverá ter por ser um modelo abstrato e conceitual demais.

Todo Diagrama de Entidade-Relacionamento segue certas regras, como duas classes precisam ser relacionadas com alguma ação ou manipulação, e essa relação precisa ter um identificador de que tipo de relação as classes tem entre si, como um (1) proprietário pode contatar vários (n) corretores, e um (1) corretor pode ser contatado por vários proprietários, assim sempre permanecendo o maior caso possível, que neste exemplo seria n:n (LARMAN, 2007).

Figura 17 - Exemplo de DER



Fonte: (RODRIGUES, 2016)

Cada evento, classe e relação são representados por imagens, sendo elas:

* As Classes são as tabelas possíveis, representadas por retângulos com o nome representando as tabelas.
* O relacionamento é descrito por ações que ficam em losangos entre as classes relacionadas.
* A relação é explicada pelos ligamentos possuindo uma associação binaria de suas Classes representadas por uma linha entre as elas.
* As associações binarias representam os casos quantitativos de usos que uma classe pode ter com a outra, como por exemplo um proprietário pode ter vários imóveis, e um imóvel pode ter apenas um proprietário (1..\*), onde 1 representa um único caso e \* ou N representam vários, podendo ter outros valores como nulo e números limitados.

Um banco de dados, mesmo em um sistema simples, implementa muita segurança e utilidades de controle, fazendo com que o sistema possa ser desenvolvido de uma forma mais rápida, eficiente, integra e segura, para que o sistema seja mais bem otimizado e seguro (DATE, 2004).

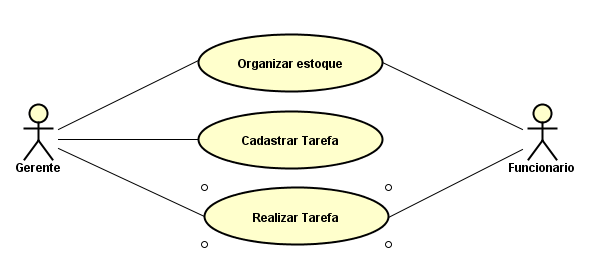
### UML

A UML ou Linguagem de Modelagem Unificada, é uma linguagem de modelamento gráfico que por meio de diagramas consegue ser modelar todos os domínios necessários de uma aplicação. Nos últimos anos a UML tem sido a linguagem de modelamento adotado pela indústria da engenharia de software, e um dos principais motivos de ter sido adotada pela indústria de softwares é que foi feita com orientação a objetos, fazendo com que possa ser utilizada com as novas tecnologias, já que quase todas as linguagens de alto nível são orientadas a objetos (GUEDES, 2011).

Guedes (2011) cita que existem aproximadamente treze diagramas diferentes, sendo eles os diagramas de:

* Diagrama de Casos de Uso: tem como utilidade descrever visualmente as funcionalidades propostas para o sistema, onde se tem os atores, que são os tipos de usuários dos sistemas, caso de uso, que são as ações possíveis no sistema, como exemplo, tanto um gerente quanto qualquer funcionário podem organizar o estoque e realizarem tarefas, mas as tarefas só podem ser cadastradas pelo gerente.

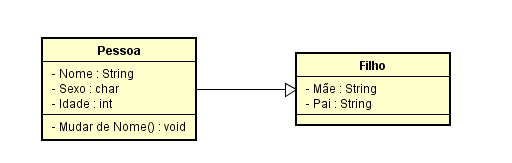
Figura 18 - Diagrama de caso de uso



Fonte: Autoria Própria

* Diagrama de Classe, que demonstra as classes, métodos, atributos, operações e relações entre os objetos, como exemplo, toda pessoa tem nome, sexo e idade, e no diagrama de classe ela é representada no segundo divisor do retângulo que representa a classe, todos os atributos e métodos podem ter diferentes tipos de acessibilidade, como privado, que é representado pelo símbolo de subtração (-), todos os métodos ficam no terceiro divisor, com sua acessibilidade e nome, que no exemplo apenas a classe pessoa tem de mudar de Nome, a classe filho herda todos os atributos e métodos de Pessoa, isso é simbolizado pela linha seguida pela seta, que representa que toda pessoa é um filho que tem mãe e pai.

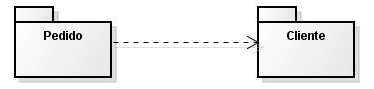
Figura - Diagrama de classe



Fonte: (Autoria Própria,2017)

* Diagrama de Pacotes, descreve os pacotes, para ilustrar a arquitetura do sistema mostrando o agrupamento entre suas classes, onde por exemplo, um cliente tem que ter pedidos, e todas as classes entre o pedidos, como produtos ficam referentes no mesmo.

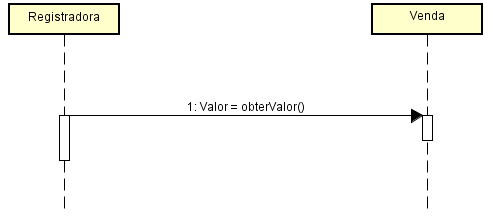
Figura - Diagrama de pacotes



Fonte: Silva (2017)

* Diagrama de Sequência, que demonstra os processos passo a passo que deverão ser seguidos pelo sistema, para uma venda ser feita é necessário saber o valor da total, onde os eventos ficam no topo do diagrama em retângulos, e a transição dos evento por uma linha seguido por uma seta para ditar o sentido.

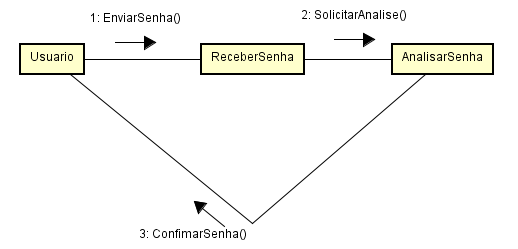
Figura - Diagrama de sequência



Fonte: Autoria Própria

* Diagrama de Comunicação, é um diagrama visual que ilustra como e quais mensagens podem ser trocados entre objetos e seus relacionamentos, como um usuário pode colocar uma senha essa senha será analisada e será autorizado o uso do sempre seguindo a ordem de seta, e seguindo a ordem de eventos.

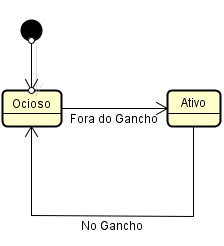
Figura - Diagrama de comunicação



Fonte: Autoria Própria

* Diagrama de Maquinas de Estados, é um diagrama que demonstra a transição entre o estado dos objetos, como um telefone que quando fora do gancho está ativo, e se posto de volta no gancho é colocado como ocioso, o estado inicial é representado por um círculo preto, os estados são retângulos com o nome do estado e os eventos são representados por setas com texto simbolizando o evento necessário.

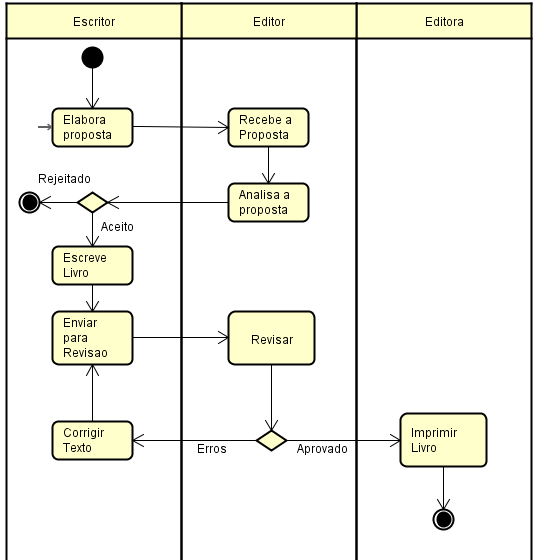
Figura - Diagrama de máquinas de estado



Fonte: Autoria Própria

* Diagrama de Atividade, é um diagrama gráfico de fluxo que representa as ações possíveis em um sistema, onde temos os atores, o evento inicial, os eventos, representados por retângulos, e as transições são repassadas por setas, o evento final é representado por um círculo com moldura, os testes são feitos por losangos, onde se pode ter múltiplos resultados.

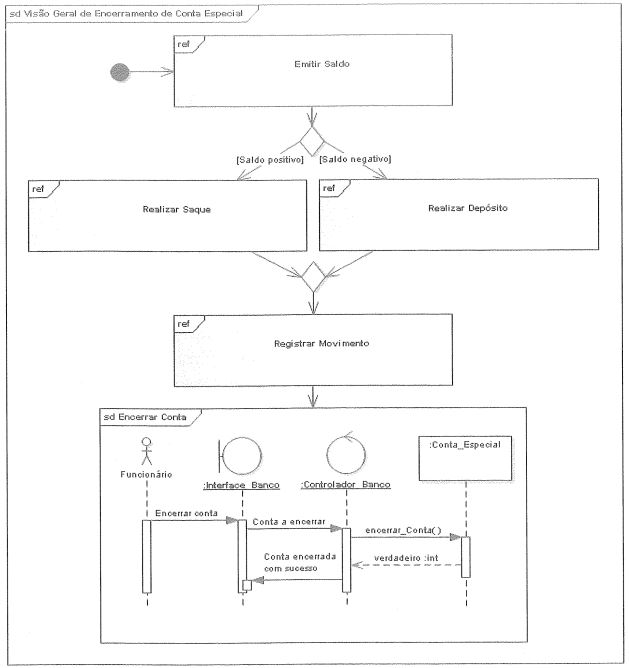
Figura - Diagrama de atividade



Fonte: Autoria Própria

* Diagrama de Visão Geral de Interação, são basicamente diagramas de atividade, mas que demonstram a interação entre outros diagramas, para demonstrar o controle de fluxo entre diagramas, onde o a transição ocorre no sentido da seta.

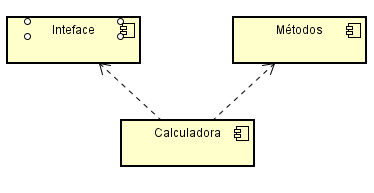
Figura - Diagrama de visão geral



Fonte: Guedes (2011)

* Diagrama de Componentes, que representa graficamente que classes terão relações entre si levando em conta a noção dos componentes de trabalho, como bibliotecas e classes externas por herança, como exemplo, uma calculadora precisa de uma interface para que aja entrada de dados por parte do usuários e precisa de métodos, para efetuar os cálculos.

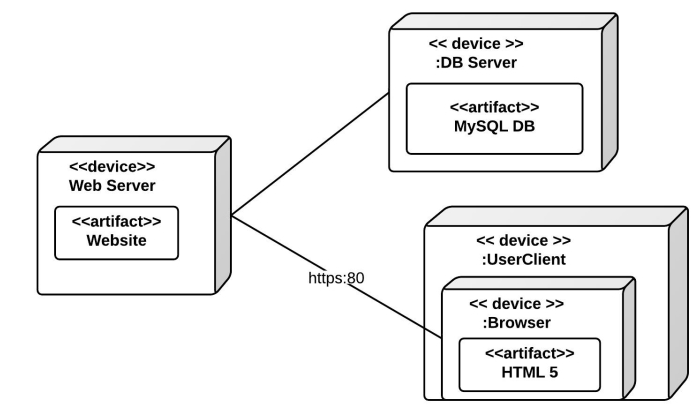
Figura - Diagrama de componentes



Fonte: Autoria Própria

* Diagrama de Implantação, que descreve informações gerados pelo software, como neste exemplo em que o usuário visualiza um website (<<artifact>>) por um aparelho (<<device>>), e tem como resultado de visualização dados do banco de dados e do html5.

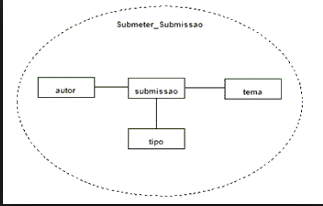
Figura - Diagrama de implantação



Fonte: Lucidchart (2015)

* Diagrama de Estrutura Composta, que demonstra a colaboração interna de classes, onde a classe autor tem um relacionamento de dados com submissão e submissão tem um relacionamento com todas as outras classes.

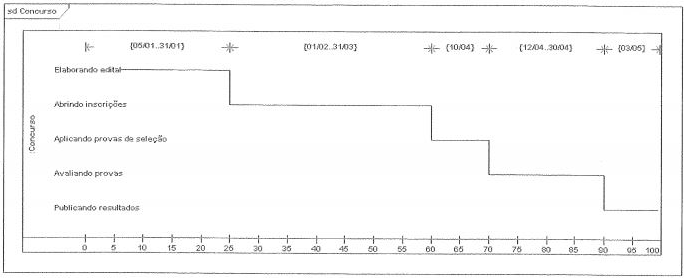
Figura - Diagrama de estrutura composta



Fonte: Delfim (2008)

* Diagrama de Tempo, representa as mudanças de estados de um objeto por eventos temporais, aonde os eventos percorrem uma linha temporal e assim que os eventos são completos simbolizado com um símbolo de asterisco no momento temporal.

Figura - Diagrama de tempo



Fonte: Guedes (2011)

A UML é essencial na área da tecnologia, pois se equipara a documentação de desenho mecânico criado por um engenheiro, que é usada para catalogar e facilitar no futuro manutenções e melhorias no sistema (ARMAND, 2009).

# REFERÊNCIAS

ARMAND. **Porque usar UML?**2009. Disponível em: <http://devbrasil.net/profiles/blogs/porque-usar-uml>. Acesso em: 07 dez. 2017.

BORDIN, Maycon Viana. **Introdução a arquitetura android.** 2012. 6 f. Monografia (Especialização) - Curso de Bacharelando em Sistemas de Informação, Sociedade Educacional Três de Maio, Três de Maio, 2012. Disponível em: <http://sites.setrem.com.br/stin/2012/anais/Maycon.pdf>. Acesso em: 15 nov. 2017.

BURNETTE, Ed. **How Android works:** The big picture. 2008. Disponível em: <http://www.zdnet.com/article/how-android-works-the-big-picture/>. Acesso em: 15 nov. 2017.

CAELUM. **Desenvolvimento web com HTML, CSS e JavaScript.**São Paulo: Caelum, 2016.

CORDEIRO, Felipe. **Introdução ao Material Design.**2015. Disponível em: <http://www.androidpro.com.br/android-material-design-introducao/#>. Acesso em: 07 dez. 2017.

DATE, C J. **Introdução a Sistemas de Bancos de dados.**São Paulo: Elsevier, 2004.

DELFIM, Samuel Martins. **Diagrama de estrutura composta.**2008. Disponível em: <http://www.portalarquiteto.com.br/diagrama-de-estrutura-composta/>. Acesso em: 07 dez. 2017.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados.**São Paulo: Pearson, 2011.

GOMES, Rafael Caveari; FERNANDES, Jean Alves R.; FERREIRA, Vinicius Corrêa. **Sistema operacional android.** 2012. 32 f. Dissertação (Mestrado) - Curso de Sistema de Computação Para Telecomunicações, Universidade Federal Fluminense, Rio de Janeiro, 2012.

GOOGLE. **Em operação com o material design.**2017. Disponível em: <https://developer.android.com/stories/index.html>. Acesso em: 07 dez. 2017.

GUEDES, Gilleanes T. A.. **UML 2:**uma abordagem prática. 2. ed. São Paulo: Novatec, 2011.

LOPES, Felipe Mariani; KOMURA, Gustavo Tochi. **Curso de Construção de Sites:**HTML/CSS. Paraná: Universidade Federal do Paraná, 2011. Disponível em: <http://www.inf.ufpr.br/instrutores/arquivos/sites/html.pdf>. Acesso em: 07 dez. 2017.

LOPES, Sergio. **Aplicações mobile híbridas com Cordova e PhoneGap.**São Paulo: Casa do Código, 2016.

LUCIDCHART. **Diagrama de implementação.**2015. Disponível em: <https://www.lucidchart.com/pages/pt/diagrama-de-implementação>. Acesso em: 07 dez. 2017.

MELO, Isaque. **Material Design Lite: O framework do Google.**2015. Disponível em: <https://tableless.com.br/material-design-lite-o-framework-google/>. Acesso em: 07 dez. 2017.

MONTEIRO, João Bosco. **Google android:** crie aplicações para celulares e tablets. São Paulo: Casa do Código, 1998.

PREZOTTO, Ezequiel Douglas; BONIATI, Bruno Batista. **Estudo de Frameworks Multiplataforma Para Desenvolvimento de Aplicações Mobile Híbridas**. 2014. 8 f. Tese (Doutorado) - Curso de Tecnologia em Sistemas Para Internet, Universidade Federal de Santa Maria, Rio Grande do Sul, 2014. Disponível em: <http://www.eati.info/eati/2014/assets/anais/artigo8.pdf>. Acesso em: 06 dez. 2017.

RODRIGUES, Joel. **Modelo Entidade Relacionamento (MER) e Diagrama Entidade-Relacionamento (DER).**2016. Disponível em: <https://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>. Acesso em: 07 dez. 2017.

SILBERSCHATZ, Abraham; KORTH, Henry F; SUDARSHAN, S. **Sistema de Banco de Dados.**5. ed. Rio de Janeiro: Campus, 2006.

SILVA, A. C. **Unified Modeling Language (UML)**. Disponível em: <http://www.deinf.ufma.br/~acmo/MOO\_Pac.pdf>. Acesso em: 07 dez de 2017.

THIZON, Edson. **MODELAGEM DE DADOS.**2011. Disponível em: <http://paginas.unisul.br/edson.thizon/PDF/aula\_modelo\_dados\_er1.pdf>. Acesso em: 06 dez. 2017.

WEISZFLOG, Walter. **Dicionário Brasileiro da Língua Portuguesa.** São Paulo: Editora Melhoramentos Ltda., 2015. Disponível em: <http://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/arquitetura/>. Acesso em: 15 nov. 2017.

ZURIARRAIN, José Mendiola. **Android já é o sistema operacional mais usado do mundo:** Plataforma do Google supera Windows pela primeira vez na história, indicando o poder do celular em relação ao PC. 2017. Disponível em: <https://brasil.elpais.com/brasil/2017/04/04/tecnologia/1491296467\_396232.html>. Acesso em: 15 nov. 2017.