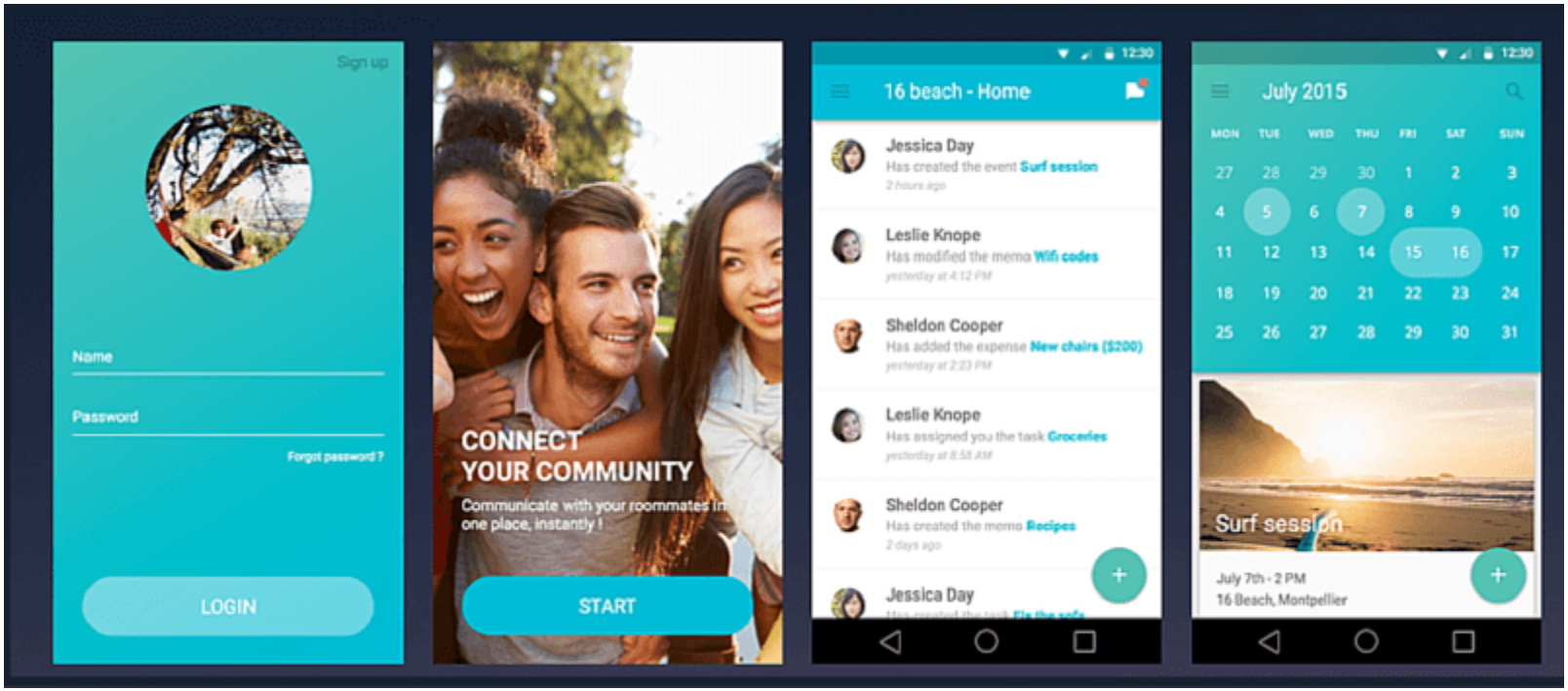
Conteúdo

O que são as Views



Simplificando, a View é um retângulo na tela que mostra algum conteúdo. Ela pode ser uma imagem, um pedaço de texto, um botão ou qualquer outra coisa que o aplicativo pode exibir. E todas as Views juntas formam o layout da interface.

Tudo o que você vê e/ou interage em seu aplicativo é chamado de **interface de usuário**, ou **UI**(do inglês, User Interface).

Existem diferentes tipos de **Views**:

Uma **View** que mostra algum texto é chamada de **TextView**

Uma **View** que mostra uma imagem é chamado de **ImageView**

Uma **View** que mostra um botão é chamado, claro, de **Button**

Além dessas, há muitos outros tipos de Views no Android, mas, por agora, vamos nos concentrar nesses três primeiros.

Mencionei que cada view é um retângulo na tela, mas onde estão todos os retângulos?

Bem, tecnicamente eles são invisíveis, mas vou destacá-los para você, assim poderá ver onde estão os limites de cada retângulo.


17:03
Explore
HOME
CATEGORIES
AUDIO
USER LISTS
RECOMMENDED TOPICS
Humans vs. Machines
The singularity is coming! Prepare with these reads
24 Books left
TRENDING BOOKS
Nonviolent Communication
A Language of Life
Marshall B. Rosenberg, PhD
Why Everyone (Else) Is a Hypocrite
Evolution and the Modular Mind
Robert Kurzban
Slide 2
Presenter view

Como você pode ver, cada aplicativo pode ser dividido em Views individuais que o compõem. Em outras palavras, as Views são os componentes básicos que você usa para construir o layout do seu aplicativo.

Desvendando o XML das Views

O que é XML?

XML, do inglês eXtensible Markup Language, é uma linguagem de marcação recomendada pela W3C para a criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais. A linguagem XML é classificada como extensível porque permite definir os elementos de marcação.

Linguagem de Marcação?

Linguagem de marcação é um agregado de códigos que podem ser aplicados a dados ou textos para serem lidos por computadores ou pessoas. Por exemplo, o HTML é uma linguagem de marcação para organizar e formatar um website, já o XML tem o mesmo conceito, mas para padronizar uma sequência de dados com o objetivo de organizar, separar o conteúdo e integrá-lo com outras linguagens.

Principais Características do XML

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.

Portanto, uma das suas principais características é sua portabilidade, pois, por exemplo, um banco de dados pode escrever um arquivo XML para que outro banco consiga lê-lo.

Aplicações

Alguns dos propósitos do XML são: auxiliar os sistemas de informação no compartilhamento de dados (especialmente via internet), codificar documentos e inserir seriais nos dados comparando o texto com o de outras linguagens baseadas em serialização.

Quando você recebe atualizações vindas de uma assinatura de RSS, isso só foi possível porque a fonte em questão disponibilizou um arquivo XML que pode fornecer o feed ao programa que fez a leitura instalado em seu computador.

W3C, ou World Wide Web Consortium, é um consórcio de empresas de tecnologia que visa padronizar a criação e interpretação de conteúdos para websites. Foi fundada em 1994 por Tim Berners-Lee, o criador da internet, para extrair o máximo que a rede pode oferecer.

Exemplos de Código XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<cd>
<titulo>Death Magnetic</titulo>
<artista>Metallica</artista>
<ano>2008</ano>
</cd>

<?xml version="1.0" encoding="iso-8859-1"?>
<receita nome="Pão Caseiro" tempo_de_preparo="10 minutos" tempo_assar="40 minutos">
  <titulo>Pão Caseiro Simples</título>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xicaras">Farinha</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1" unidade="xicaras" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="xicaras">Açúcar</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture o fermento com o açúcar e espere aproximadamente cinco minutos.</passo>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, modele o formato e coloque na forma. Asse em forno médio.</passo>
  </instrucoes>
</receita>
```

Agora vamos ver mais tecnicamente como funcionam as Views. Internamente uma parte das Views são compostas por **XML**, ou seja, utilizamos a **sintaxe** dessa tecnologia para construir nossos componentes.

Mas o que é sintaxe? Sintaxe significa regras que definem alguma coisa, nesse caso, se o XML da View está correto ou não.

Este é o código **XML** para o elemento **TextView**.

```
<TextView
    android:text="Olá Androideiro!"
    android:background="@android:color/darker_gray"
    android:layout_width="150dp"
    android:layout_height="75dp" />
```

Nós sempre temos que começar escrevendo nosso XML abrindo o elemento (tag) através da utilização de um (<), seguido do nome do componente. Este exemplo poderia ser uma ImageView, Button, ou qualquer outra View.

Então nós temos uma lista de atributos, um por linha e no final temos (/>) fechando nossa tag.

Outra forma de fazer o fechamento da tag do elemento, é utilizar uma tag separada para o fechamento. Aqui está o exemplo de um LinearLayout onde isso pode ser utilizado.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <TextView
        android:text="Ola Androideiro!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"  />

    <TextView
        android:text="Bora criar apps"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"  />

</LinearLayout>
```

Você pode perceber que no meio do LinerarLayout temos dois elementos TextView. Estes são chamados de elementos filhos dentro do LinearLayout pai.

Outra característica dos elementos XML das Views, são os textos adicionais dentro da tag que são chamados de atributos. Os atributos são características que determinam o comportamento ou a aparência da View no Android.

android:Text=”Ola Androideiro!”

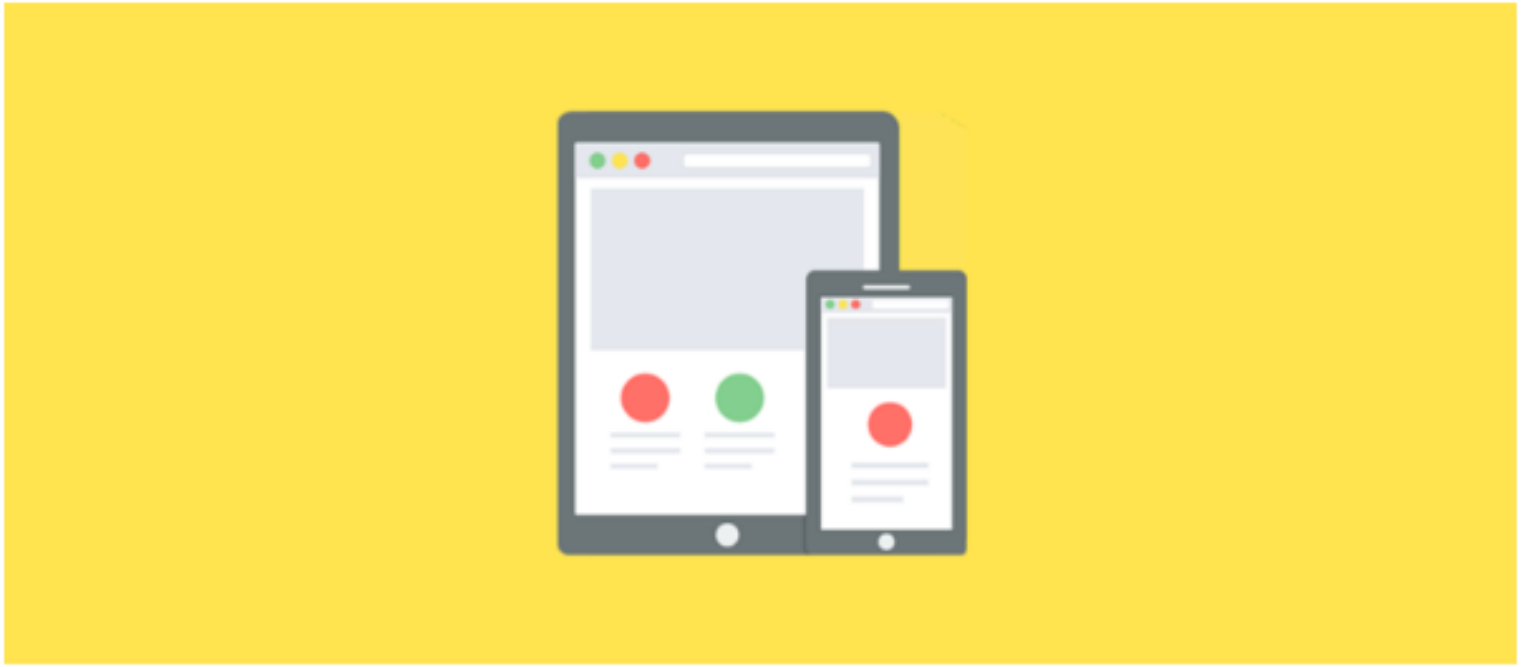
android:background=”@android:color/black”

android:layout_width=”150dp”

android:layout_height=”750dp”

Analisando os atributos mais a fundo, temos o nome do atributo do lado **esquerdo**. Do lado **direito**, temos o valor do atributo. É importante notar que os valores devem ser colocados entre **aspas** como parte da sintaxe do XML.

Trabalhando com Views Responsivas

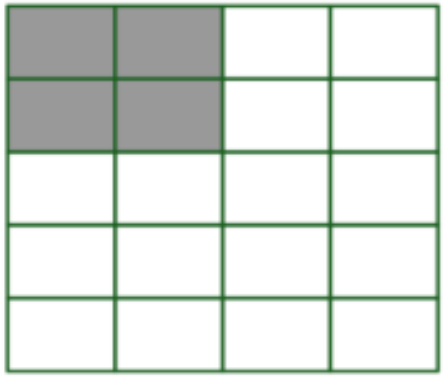


Um dos desafios para o desenvolvedor Android é conseguir criar layouts que se encaixam nos vários tamanhos e resoluções diferentes de telas de dispositivos presentes no mercado. É importante que todo aplicativo suporte o máximo de telas diferentes para que funcione, sem problemas, em vários dispositivos.

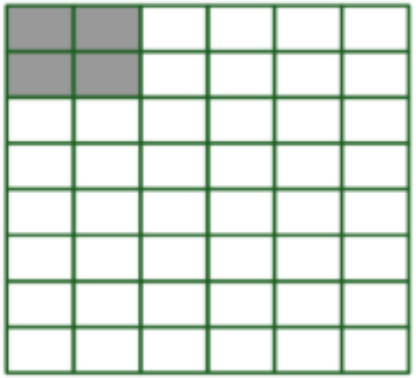
Os retângulos desenhados abaixo representam diferentes dispositivos com o mesmo tamanho físico, mas com diferentes resoluções.

A tela do telefone é composta de muitos pequenos quadrados, que são os Pixels. Digamos que temos uma View (um botão, por exemplo) com o tamanho de 2 pixels de altura por 2 pixels de largura. Veja abaixo como ficaria nosso botão em diferentes resoluções.

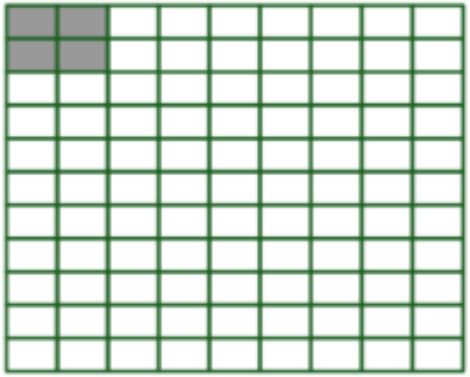
Não ficou legal, né? Na resolução xhdpi, fica difícil pressionar o botão de tão pequeno!



Resolução Média
(mdpi)

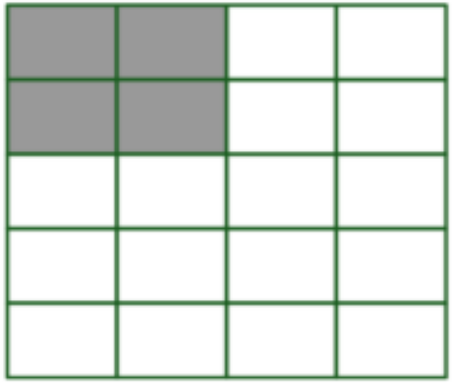


Resolução Alta
(hdpi)

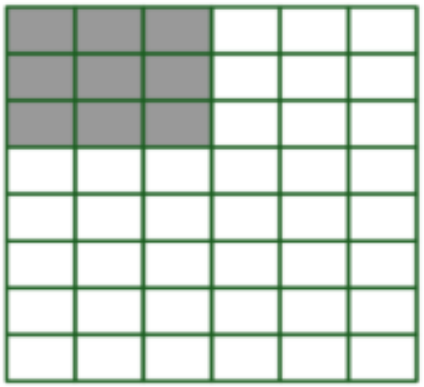


Resolução Extra-Alta
(xhdpi)

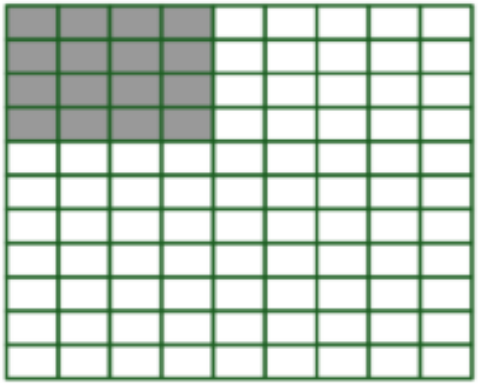
É por isso que temos a medida de **Pixels Independente de Densidade** (DP). Eu posso definir o meu botão com tamanho de **2dps de altura por 2dps de largura**.



Resolução Média
(mdpi)



Resolução Alta
(hdpi)



Resolução Extra-Alta
(xhdpi)

Você não tem que se preocupar em fazer funcionar nas diferentes resoluções, tudo o que você precisa se lembrar é de definir o seu tamanho utilizando a medida de DP.

As 06 Views mais usadas no Android

Definir quais Views utilizar em um layout é uma parte crítica do desenvolvimento Android. Vamos dar uma olhada nas Views mais comuns que são utilizadas na maioria dos layouts dos aplicativos.

Com apenas essas **06 opções** (juntamente com os ViewGroups) é possível criar layouts extremamente bonitos e usáveis.

- **TextView** – Exibe um de texto formatado
- **ImageView** – Exibe um uma imagem
- **Button** – É utilizado para e executar uma ação ao ser clicado
- **ImageButton** – Exibe uma imagem com comportamento de um botão
- **EditText** – É um campo de texto editável para a entrada de dados
- **ListView** – É uma lista de itens que contém outras Views

Veja abaixo um resumo das funcionalidade de cada uma e como você pode utiliza-las em seu aplicativo.

TextView

A principal utilidade do TextView é exibir o texto na tela de um aplicativo Android. Embora isso possa parecer uma tarefa simples, a classe TextView contém uma lógica complexa que lhe permite exibir um texto formatado, hyperlinks, números de telefone, e-mails e outras funcionalidades úteis.

-

<TextView

- android:text="Olá Androideiro!"
- android:background="@android:color/darker_gray"
- android:layout_width="150dp"
- android:layout_height="75dp" **/>**

ImageView

Como o nome já diz, o ImageView é projetado especificamente para exibir imagens na tela. Isso pode ser usado para a exibição de recursos armazenados no aplicativo ou para a exibição de imagens que são baixadas da internet.

```
<ImageView
android:src="@drawable/imagem"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:scaleType="center"/>
```

- O Button é um dos controles mais básicos disponíveis em um aplicativo. Ele responde a cliques do usuário e chama um método em seu código para que você possa responder de forma apropriada quando o usuário pressiona o botão.
- O ImageButton é um combinado de um Button com uma ImageView, juntando as caracteristicas de cada um em um só lugar.

```
<Button
android:layout_height="wrap_content"
android:layout_width="wrap_content"
android:text="Meu Botão"
android:onClick="clicar" />
<ImageButton
android:id="@+id/imageButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/imagem" />
```

O EditText é uma extensão da TextView e permite aos usuários editar/entrar o texto através de uma entrada do teclado.

```
<EditText
android:id="@+id/email"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:hint="Preencha seu melhor email..."
android:inputType="textEmailAddress" />
```

A ListView é usada para exibir uma coleção de itens de forma linear em uma única coluna. Cada item individual pode ser selecionado para a exibição de mais detalhes ou realizar uma ação relacionada a esse item

```
<ListView
    xmlns:android="http://schemas.android.com/apk/res/and
    roid"
android:id="@+id/listview"
android:layout_width="match_content"
android:layout_height="match_content" />
```

Personalização de Views e seus atributos

Essa é uma parte muito divertida e extremamente importante dentro do desenvolvimento Android, a personalização e customização do design das nossas Views. Existem vários atributos disponíveis que podemos utilizar para melhorar o visual e a usabilidade dos nossos layouts.

Identificando a View

A primeira coisa que devemos entender é o atributo **android:id**, que é responsável por identificar a View, ou seja, dar um nome único ao componente.

<Button android:id**="**@+id/meuBotao" **/>**

Este ID pode então ser acessado dentro do código Java.

```
Button meuBotao = (Button) findViewById(R.id.meuBotao);
```

Cada View tem muitos atributos diferentes, que podem ser aplicados em diferentes propriedades. Certas propriedades são compartilhadas entre muitas Views.

Tamanho da View: **layout_width** e **layout_height**

- **wrap_content**: Adaptar ao conteúdo da própria View
- **match_parent**: Adaptar a View pai

Tamanho do Texto: **textSize**

- Usar a medida de **SP** (Pixel Independente de Escala)
- Outra alternativa **textAppearance**

Cor do Texto: **background** e **textColor**

- Aceita hexadecimal **#F1F1F1**
- Nome da cor **android:color/black**

As propriedades mais importantes de um ImageView são a imagem a ser utilizada, e a escala e posicionamento desta imagem dentro da View.

Endereço da Imagem: **src**

Imagens **android:drawable/imagem**

XMLs de efeitos

Escala da Imagem: **scaleType**

centerCrop, center e centerInside

fitCenter, fitStart, fitEnd e fitXY

matrix

Conclusão: Saiba utilizar as Views Corretamente

Você aprendeu qual a importância das **Views** dentro do desenvolvimento Android e viu também que pequenos detalhes podem fazer a diferença quando construimos nosso layout. As Views são componentes separados que quando são utilizados em conjunto tem um poder enorme no design do nosso aplicativo.

Todos os componentes da interface de usuário no Android são feitos, em parte, utilizando o conceito de **XML**. Você viu como é fácil utilizar esse conceito, porém é preciso atenção na sintaxe para não ter problemas na hora de criar seus layouts.