Desenvolvimento para dispositivos móveis

Views e Layouts

Professor Msc. Fabio Pereira da Silva
E-mail: fabio.pereira@faculdadeimpacta.com.br

Github

- Link com os códigos gerados na aula passada:
- https://github.com/fabiodasilva500/Aulas-Mobile/tree/Aula-7-Ciclo-de-vida-Activity

Conceitos de layout

- A primeira interação do usuário final da sua aplicação será através do layout, a parte gráfica que mostra o que o aplicativo tem a oferecer.
- É de conhecimento geral que se uma interface é má construída ou difícil de entender, é provável que a aplicação seja fechada e esquecida. Interfaces gráficas bem implementadas, assim como a lógica da aplicação bem escrita, contribuem para uma melhor experiência de usuário e consequentemente para o uso contínuo do software.

Conceitos de layout

- Layouts determinam como será a distribuição dos componentes em tela na aplicação Android.
- Existem vários tipos de Layouts que podem ser trabalhados na aplicação, como o Linear Layout e o Relative Layout.
- Dependendo do cenário no curso do processo de desenvolvimento de software um deles pode ser considerado mais aplicável.

Conceitos de layout

- Na plataforma Android, o layout de uma aplicação diz quais os elementos que uma interface possui e como se comportam.
- Todos os elementos no layout são construídos usando a hierarquia de View e ViewGroup.

Conceitos de layout

- View geralmente representa algo que o usuário pode ver e interagir na aplicação. Por exemplo, botões, áreas de texto, spinner, checkbox, radio buttons, etc.
- ViewGroup é um tipo de elemento que contém views. ViewGroup diz como os elementos de interface se comportam na aplicação, porém não é visível. Apesar de presente na aplicação, o usuário final não consegue ver nem interagir com ela.

Conceitos de layout

View	ViewGroup
Geralmente representam elementos na interface que o usuário possa interagir Button, EditText, Switch, ImageView, CheckBox	Elementos de layout que definem o comportamento de outros elementos. Um agrupador de elementos de interface LinearLayout, RelativeLayout, ConstraintLayout, RecyclerView

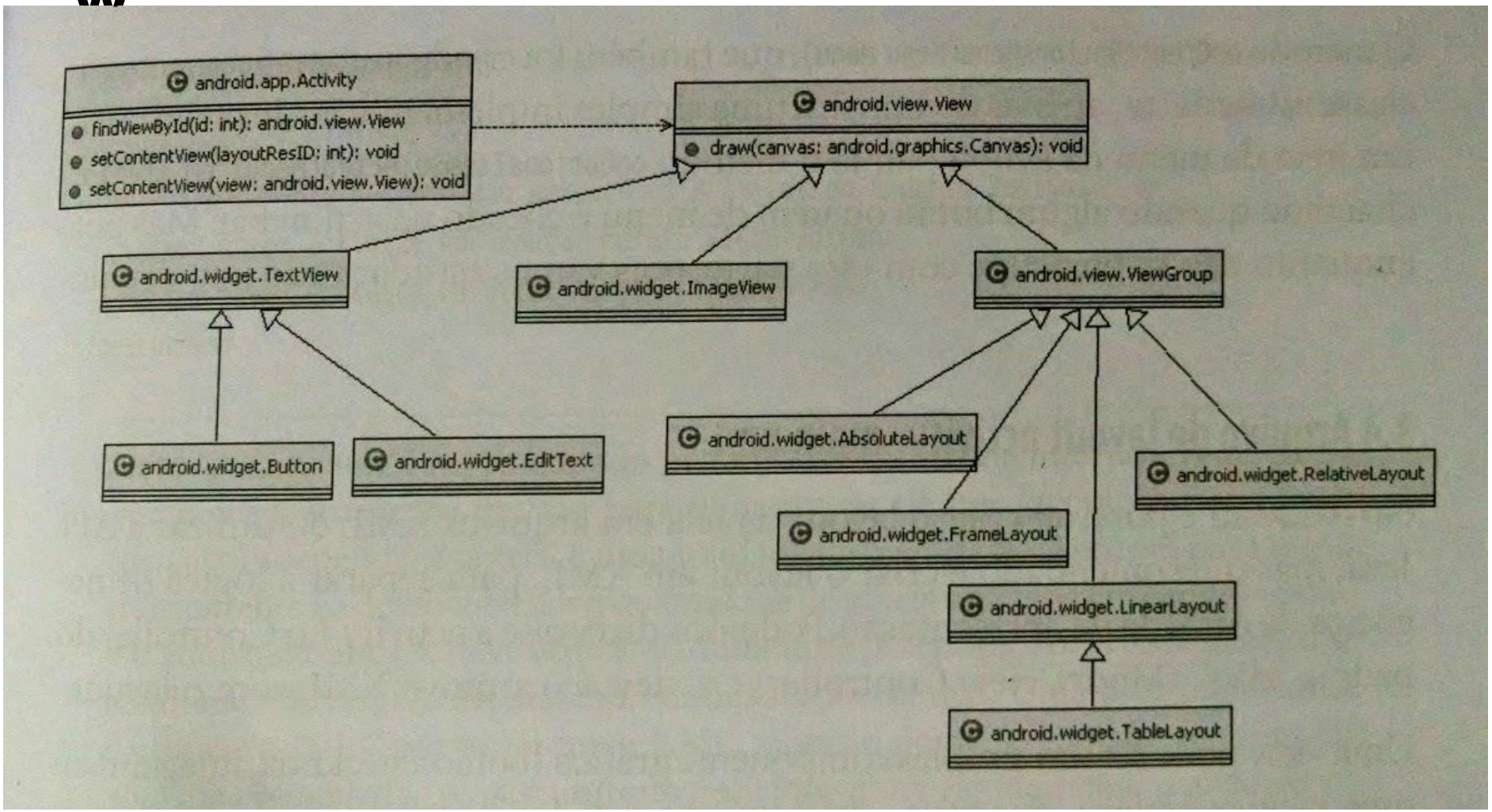
View

- A classe android.view.View é a superclasse de todo e qualquer componente visual de um aplicativo Android
- Suas subclasses são utilizadas para criar a interface gráfica do aplicativo
- Os componente visuais são divididos em duas categorias:
  - Widgets
  - Gerenciadores de layout

View

- Widgets são componentes visuais das telas do aplicativo, que herdam diretamente da classe View
  - Botão: Button
  - Imagem: ImageView
  - Texto: TextView
- Gerenciadores de layout, ou simplesmente layout são subclasses de android.view.ViewGroup
  - Esta é subclasse da View
  - São os componentes responsáveis por organizar os conteúdos de uma tela do aplicativo
    - Horizontalmente
    - Verticalmente

Vie
w

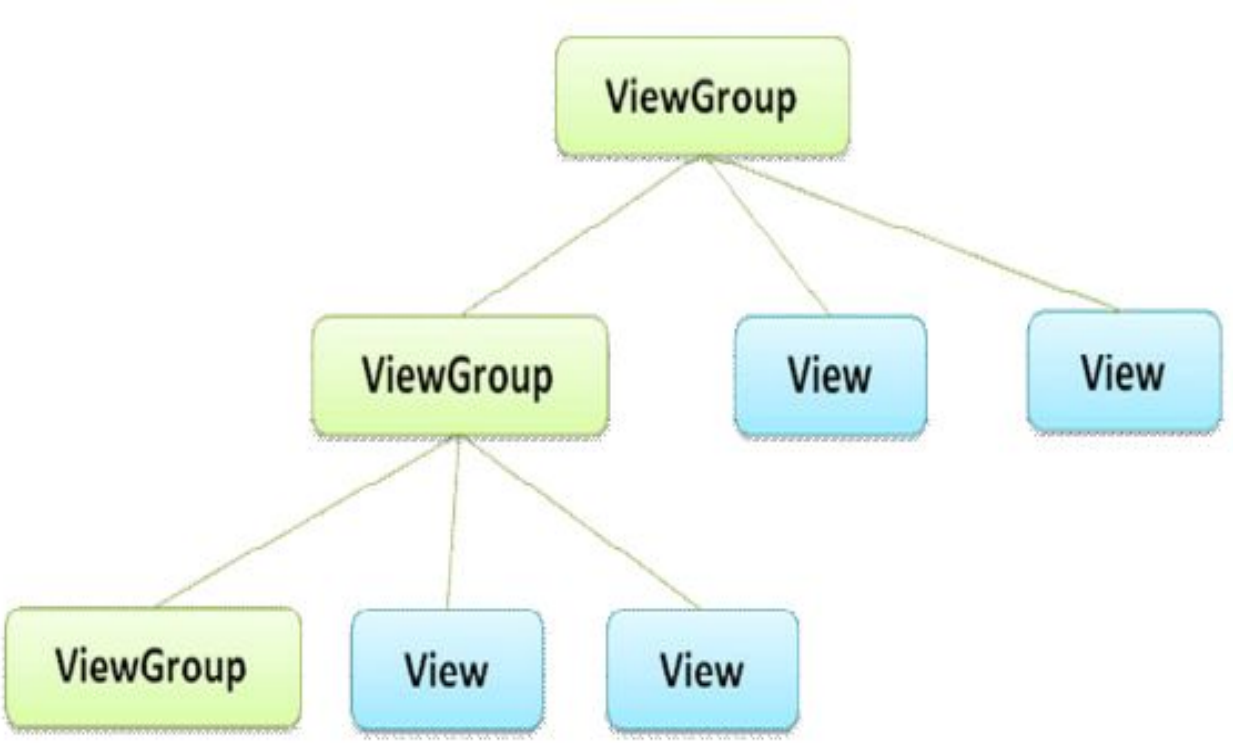


LECHETA, R. R. Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK. 5.ed. São Paulo:
Novatec, 2010

ViewGrou p

- As principais classes de layout são:
  - FrameLayout: mais comum e mais simples; o componente preenche a tela inteira. Quando há dois componentes, um sobrepões o outro
  - LinearLayout: organizar componentes na vertical ou horizontal
  - TableLayout: Filho de LinearLayout, organiza os componentes um uma tabela com linhas e colunas
  - RelativeLayout: posiciona um elemento relativo a outro existente
  - ConstraintLayout: similar ao relative layout, porém mais flexível, e mais fácil de utilizar com o editor de layout

Hierárquia de View e View Group



Arquivo de Layout

- Um layout pode ser construído de duas formas:
  - Construir os elementos de tela de um arquivo XML (estático)
  - Instanciar elementos em tempo de execução (dinâmico)
- Ambos tem vantagens e desvantagens, e podem ser utilizados conjuntamente para gerenciar os componentes de tela do aplicativo
  - Por exemplo, você pode colocar o layout padrão em arquivos XML e depois colocar ou modificar componentes de forma dinâmica em tempo de execução

Arquivo de Layout

- Declarando elementos de tela no XML provê uma melhor separação entre a apresentação (telas) e o controle do comportamento (eventos, feitos em Kotlin)
- Separando o código, fica fácil, por exemplo, definir telas diferentes para diferentes orientações de tela
- Entretanto, ao criar os elementos em tempo de execução prove maior flexibilidade
  - Componentes que serão exibidos dependendo de

Arquivo de Layout - XML

- Ao criar um layout estático, ou seja, em um arquivo de recurso de XML, todo o código é escrito com marcadores XML e configuração dos atributos
  - Um marcador XML é uma classe no SDK
- Um arquivo de layout:
  - sempre estará na para res/layout do seu projeto;
  - será um arquivo XML;
  - Deve conter um elemento raiz, normalmente um elemento filho de ViewGroup

Arquivo de Layout - XML

- Dentro do elemento raiz é possível adicionar componentes, tanto widgets (botões, imagens, campo de texto) quanto outros layouts
  - Constrói-se um hierarquia de Views para definir o layout
- No projeto já utilizamos Views (layout e widgets), para criar as telas do aplicativo
  - Por exemplo a tela de login (simplificada, no exemplo a seguir)
  - Ela contém um elemento raiz de Layout (LinearLayout) e campos para texto (TextView) e entrada de dados (EditText)

Arquivo de Layout - XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text_size="14sp"
        android:text="Olá" />
</LinearLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:text="Usuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textoUsuario" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:id="@+id/campoUsuario" />
    <TextView
        android:text="Senha"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textoSenha" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/campoSenha" />
</LinearLayout>
```

Definições de Layouts

- **match_parent:** Quando um elemento possui este valor no atributo largura ou altura (width ou height), seu tamanho será igual o tamanho do elemento de nível acima.
- **wrap_content:** Quando um elemento possui este valor no atributo largura ou altura (width ou height), seu tamanho será o suficiente para acomodar seu conteúdo. Significa que o elemento cresce ou diminui de acordo com o conteúdo.

Arquivo de Layout - XML

- Somente criar um arquivo XML de layout não indica que ele será utilizado
- Sua utilização é feita dentro do método onCreate na Activity correspondente
- A inclusão de um arquivo de Layout na activity é feita pelo método setContenView(), enviando como parâmetro o recurso de layout (arquivo XML).
- Por exemplo, colocamos recurso login.xml na Activity MainActivity da forma a seguir:

Arquivo de Layout - XML

```
class MainActivity : DebugActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.login)

        ...
    }
}
```

Principais atributos

- Cada objeto de View ou Layout aceita uma série de atributos
  - Configuração e personalização do objeto
  - Largura, altura, nome, texto, tipo...
- Para todo elemento de tela, seja um layout ou widget, existem alguns atributos básicos, herdados da Classe View ou do ViewGroup:
  - android:id: define um identificador único para o componente
  - android:layout_height: define a altura de um layout ou view
  - android:layout_width: define a largura de um layout ou view

Atributos de Layout

Gravity

- Alinhamento dentro do elemento

```
android:gravity="center"
```

- Opções disponíveis

```
Center, right, left,
bottom, top,
center_horizontal,
center_vertical
```

LayoutGravity

- Alinhamento do elemento com relação ao pai

```
android:layout_gravity="center
"
```

- Opções disponíveis

```
Center, right, left, bottom,
top, center_horizontal,
center_vertical
```

Atributos de Layout

Background

- Cor de fundo do elemento

```
android:background="#fff"
```

- Aplicável para View e ViewGroup

textColor

- Cor do texto do elemento

```
android:textColor="#0AA37D"
```

- Aplicável para elementos que possuem texto. Ou seja, LinearLayout, Relative, Constraint não possuem a opção.

Altura e Largura da View

- Abra o arquivo activity_tela_inicial do projeto
- Identifique o Layout da tela da tela que foi criada (marcador raiz, finalizado com Layout) e os parâmetros android:layout_height e android:layout_width
- Eles estão configurados com o valor match_parent
  - O layout vai ocupar a tela inteira, tanto na largura quanto na altura

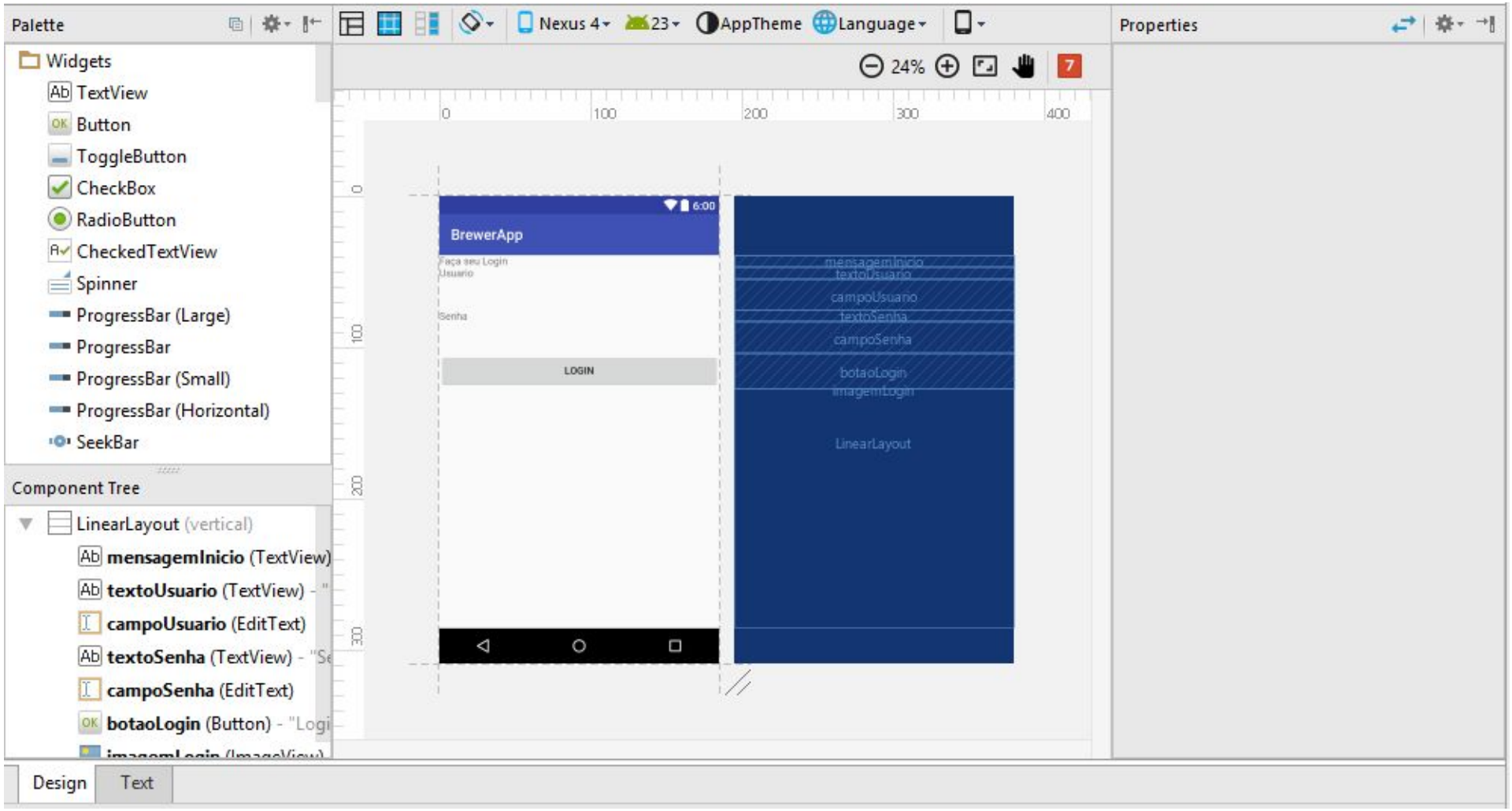
Altura e Largura da View

- A altura e largura podem receber os seguintes valores:
  - Número inteiro: tamanho que vai ocupar, em pixels. Por exemplo: 50dp
    - Utilizar dp (density independent pixel) converte os pixels corretamente de acordo com a densidade/resolução da tela
  - match_parent: o componente vai ocupar todo o tamanho definido pelo pai
  - wrap_content: o componente ocupa apenas o espaço necessário na tela

XML - Editor Visual

- O Android Studio disponibiliza um editor visual para facilitar a criação de Layout em XML
  - Os arquivos de Layout são abertos neste editor visual
- Os componentes podem ser colocador utilizando “arraste e solte” e os parâmetros configurados nas propriedade do lado direito
- Também é possível configurar o XML manualmente, trocando a visualização na parte inferior do editor
- Tanto no editor quanto no arquivo XML é possível ter uma pré-visualização da tela

XML - Editor Visual



FrameLayout, altura e largura

- Vamos entender melhor como funciona a configuração da altura e da largura dos elementos
- No projeto da aula altere o layout da tela inicial (activity_tela_inicial.xml) para FrameLayout
  - Mais simples dos gerenciadores
  - Utilizado quando a tela tem apenas um componente que pode preencher a tela inteira ou para empilhar um componente sobre o outro
  - Quando vários componentes são adicionados o último componente ficará na frente de todos

FrameLayout, altura e largura

- Troque o valor dos parâmetros de altura e largura para wrap_content
  - Ou seja, o tamanho do layout será somente o suficiente para conter os elementos que estão dentro dele
- Coloque o atributo android:background:"#aaaaaa" no layout para definir a cor de fundo
- Configure a altura e largura do botão de sair com wrap_content
- Ou seja, a altura e largura do botão terá o tamanho suficiente para caber o botão

FrameLayout, altura e largura

- O Arquivo de Layout ficará parecido com o seguinte:

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android
    "  android:background="#aaaaaa"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <Button
        android:id="@+id/botaoSair"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textoInicial
        "  android:text="Sair" />
</FrameLayout>
```

FrameLayout, altura e largura

- Veja no editor visual que tanto o layout quanto o botão ocupam apenas o espaço necessário, delimitado pelo fundo cinza da imagem
  - O resto da cor branca da tela não faz parte do layout

FrameLayout, altura e largura

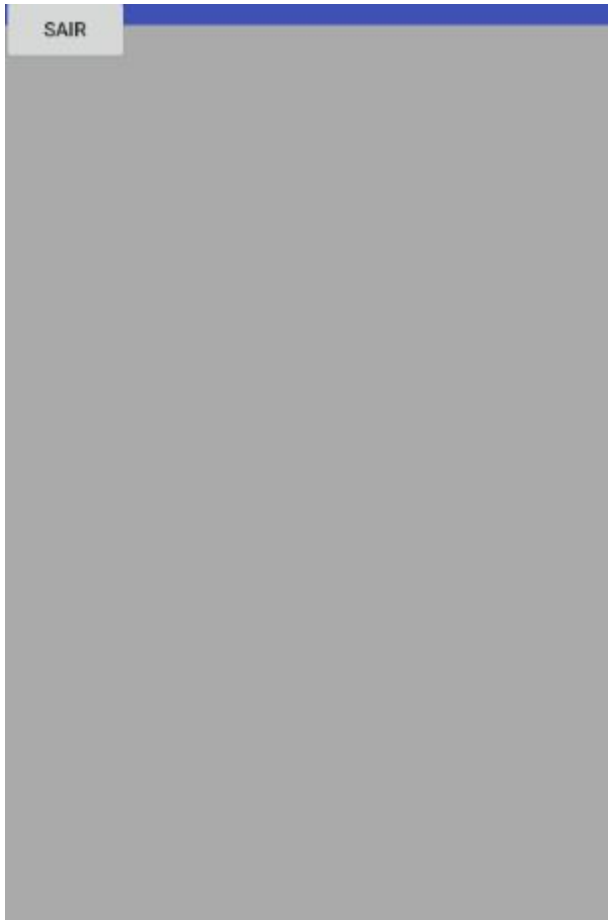
- Troque o valor da altura e largura do layout para o valor match_parent, da

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#aaaaaa"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/botaoSair"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textoInicial"
        android:text="Sair" />
</FrameLayout>
```

FrameLayout, altura e largura

- Veja que agora o fundo cinza aparece na tela inteira
  - O “parent” do Layout é o próprio espaço disponível na tela do aplicativo



FrameLayout, altura e largura

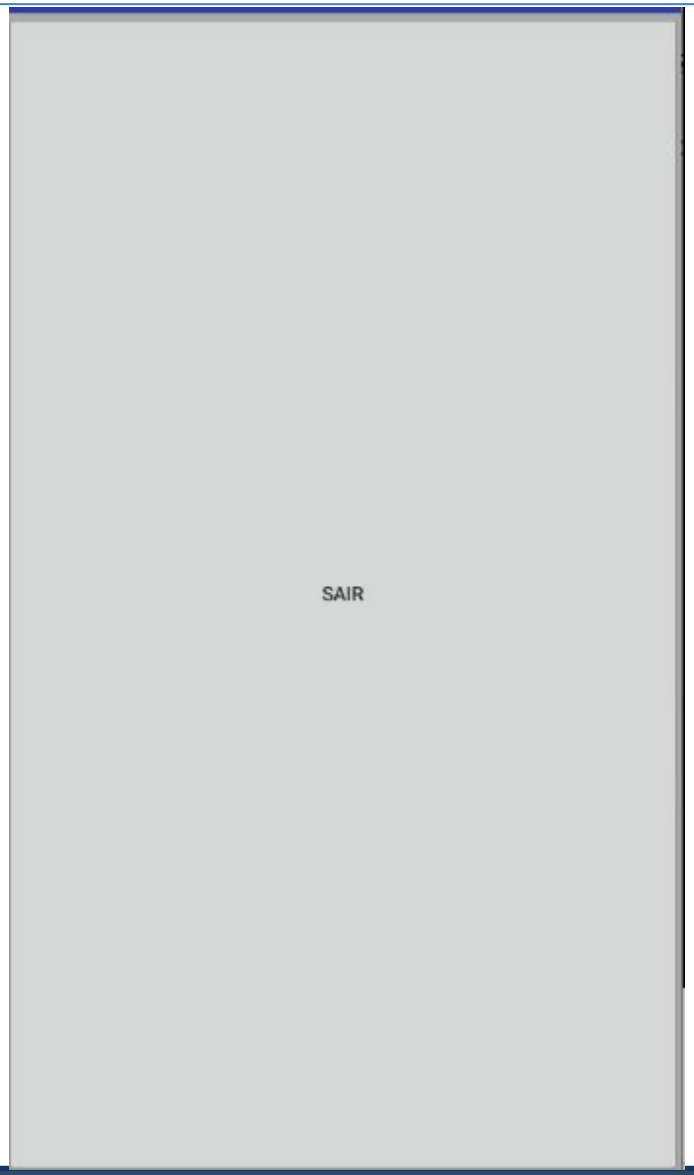
- Troque o valor da altura e largura do botão para match_parent, da seguinte forma:

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#aaaaaa"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/botaoSair"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/textoInicial"
        android:text="Sair" />
</FrameLayout>
```

FrameLayout, altura e largura

- Agora o botão é esticado para preencher o espaço do layout
- O “parent” do botão é o layout



FrameLayout, altura e largura

- É possível combinar valores diferentes para altura e largura, e assim configurar o componente como for necessário
- Por exemplo, configure a largura do botão com match_parent e altura para

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#aaaaaa"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/botaoSair"
        android:layout_width="match_parent"
        android:layout_height= "wrap_content"
        android:layout_below="@+id/textoInicial"
        android:text="Sair" />
</FrameLayout>
```

FrameLayout, altura e largura

- Dessa forma, o espaço do botão ocupará o espaço do pai (Layout) em toda a largura, mas a altura será apenas o necessário para caber o botão

FrameLayout - Exemplo

- Outro exemplo do uso de FrameLayout
- Na tela login.xml coloque um componente de FrameLayout em volta do botão de login
  - Hierarquia de layouts
  - Configure a altura com wrap_content e a largura com match_parent
- Depois do botão, ainda dentro deste FrameLayout coloque uma ProgressBar

FrameLayout - Exemplo

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <Button
        android:text="Login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/botaoLogin" />

    <ProgressBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/progressBar"/>
</FrameLayout>
```

FrameLayout - Exemplo

- Veja que o botão e a barra de progresso aparecem sobrepostas (bordas da barra de progresso em rosa)

FrameLayout - Exemplo

- Obs: A partir do Android 5.0, o SDK trabalha com a propriedade elevation para os botões, colocando eles sempre a frente no FrameLayout, mesmo que venha antes de outro componente no arquivo de layout
- Para remover a propriedade elevation do botão coloque o seguinte parâmetro no componente de botão:

```
android:stateListAnimator="@null"
```

LinearLayout

- O gerenciador Linear layout funciona, como o nome sugere, de forma linear, permitindo basicamente duas orientações: vertical e horizontal.
- Seu comportamento padrão é colocar todos os elementos, seguindo a ordem em que aparecem no código, lado a lado (no caso horizontal) ou um abaixo do outro (no caso vertical).

LinearLayout

- Devemos configurar um layout ou qualquer componente gráficos através das propriedades de cada elemento que pode ser configurada de forma visual ou via código.
- OBS: Não são todas as propriedades que podem ser configuradas visualmente. Algumas só podem ser definidas via código.
- Para configurar as propriedades de forma visual, clicamos no elemento visual a ser configurado (layout ou componente gráfico) e realizamos as alterações através da janela PROPERTIES.
- Para manipular as propriedades via código XML, clicamos na opção com o nome da activity.xml. Para visualizar de forma gráfica clicamos em Graphical Layout.

LinearLayout

- Na criação de layouts em Android, sempre chamamos um parâmetro usando o prefixo "android:“ e em seguida um pósfixo que define qual propriedades será manipulada.
- Exemplo: android:layout_width="match_parent"
- Este parâmetro define o tamanho da largura da tela que estamos criando. Este tamanho pode ser um tamanho fixo (em pixels, density pixels, ou outras unidades de formatação) ou em tamanhos expansíveis.

LinearLayout

- Existem 3 tamanhos expansíveis em Android:
- fill_parent: Com esta opção, o tamanho do parâmetro será máximo (ou seja, o tamanho da tela corrente)
- wrap_content: Com esta opção, o tamanho do parâmetro será mínimo, tendo como base os componentes-filhos do layout atual.
- match_parent: Mantém o tamanho herdado pelo componente-pai. Caso não haja um componente-pai, o tamanho será máximo (ou seja, o tamanho da tela).

LinearLayout

- Classe android.widget.LinearLayout
- Um dos layouts mais utilizados
- Organiza uma sequência de componentes na horizontal (padrão) ou vertical
  - Utilize o atributo android:orientation
  - Valores: horizontal ou vertical
  - Se o atributo não for especificado, será utilizado o layout horizontal
- O elementos serão colocados no próximo espaço disponível, na horizontal ou vertical

LinearLayout - Exemplos

- Abra novamente o arquivo da tela de login (login.xml)
- Altere a largura de todos os campos da tela para wrap_content
- Altere a propriedade android:orientation do layout para horizontal
- Os componentes são colocados um ao lado do outro

LinearLayout - Exemplos

- O LinearLayout Horizontal sempre coloca os elementos um ao lado do outro, mesmo que não haja mais espaço na tela
- Volte para o distribuição vertical (android:orientation="vertical") para manter o layout correto

LinearLayout: alinhamento

- É possível alinhar os componentes na tela com o atributo android:layout_gravity. Os seguintes valores são válidos:
  - top: cima
  - bottom: baixo
  - left: esquerda
  - right: direita
  - center_vertical: centralizado na vertical
  - center_horizontal: centralizado na horizontal
  - center: centralizado na horizontal e vertical

LinearLayout: alinhamento

- Brinque um pouco com a ProgressBar da tela de login
  - Configure a altura e largura para wrap_content
  - Coloque o atributo android:layout_gravity com os valores center, left e right, um de cada vez para ver o alinhamento

```
android:layout_gravity="center"
```

Peso e relevância

- É possível organizar os componentes de uma tela atribuindo um peso para cada um deles
  - Aqueles com maior peso ocuparão um maior espaço
- Utiliza-se o atributo android:layout_weight
- Se quisermos que o FrameLayout em volta do botão de login **ocupe o restante do espaço disponível na tela**, configuramos o atributo layout_weight da seguinte forma:
  - android:layout_weight="1"

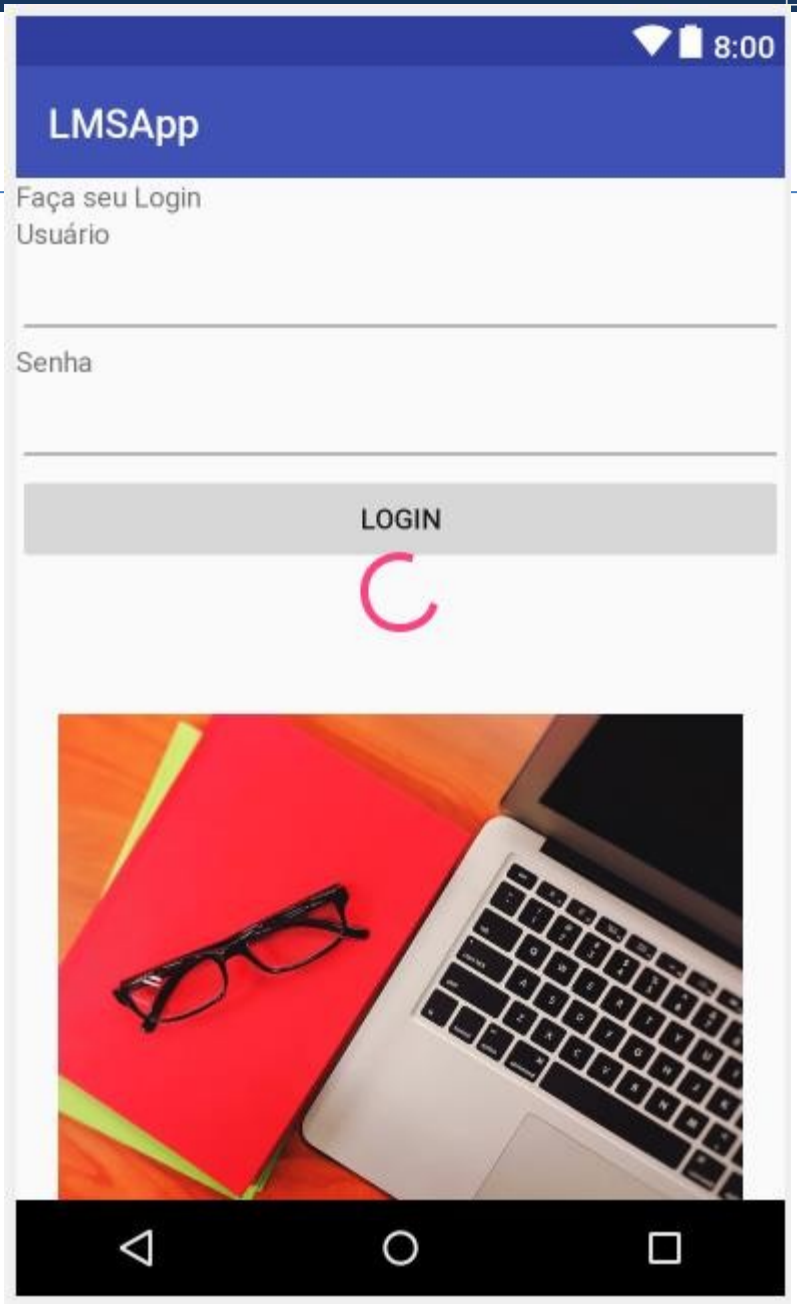
Peso e relevância

- Ao definir android:layout_weight="1" apenas para o FrameLayout informamos que este componente deve ocupar todo o espaço restante disponível na tela
- Para isso precisamos remover a configuração de altura com android:layout_height="0dp", para que ela seja feita automaticamente pelo peso

Peso e relevância

- Pode-se configurar vários elementos de tela com pesos diferentes
- Configure imagem da tela de login peso 2
- Como o FrameLayout do botão de login está com peso 1, o total do espaço restante foi dividido em 3 partes (2 da imagem + 1 do FrameLayout)
  - A imagem ocupará 2/3
  - O FrameLayout ocupará 1/3

Peso e relevância



GridLayout

- A palavra “grid”seria traduzida como “grade”ou “rede”e, como podemos intuir, grid layout se trata de um gerenciador com funcionamento semelhante a uma grade. Permitindo que se defina um número de linhas e colunas, os elementos são colocados respeitando a indicação horizontal (coluna) e vertical (linha).

RelativeLayout

- android.widget.RelativeLayout
- O RelativeLayout é utilizado para posicionar os componentes em relação a outros: acima, abaixo, ao lado
- É necessário que seja definido um ID para cada componente dentro do RelativeLayout
  - O posicionamento de um elemento depende do outro
- Cada componente dentro do RelativeLayout deve ser configurado com atributos que indicam posição, alinhamento e espaços

RelativeLayout

- Relative Layout é um layout do tipo relativo, ou seja, ao contrário do Linear Layout, que especifica sempre uma direção horizontal ou vertical, no relative layout posicionamos os elementos por referência à outros elementos.
- Por exemplo, dizemos se o botão estará abaixo de um campo de texto, do lado direito ou até mesmo em cima dele.

RelativeLayout

- android:layout_below: Posiciona abaixo de um componente
- android:layout_above: Posiciona acima de um componente
- android:layout_toRightOf: Posiciona a direita de um componente
- android:layout_toLeftOf: Posiciona a esquerda de um componente
- android:layout_alignParentTop: alinha acima do layout pai
- android:layout_alignParentBottom: alinha abaixo do layout pai

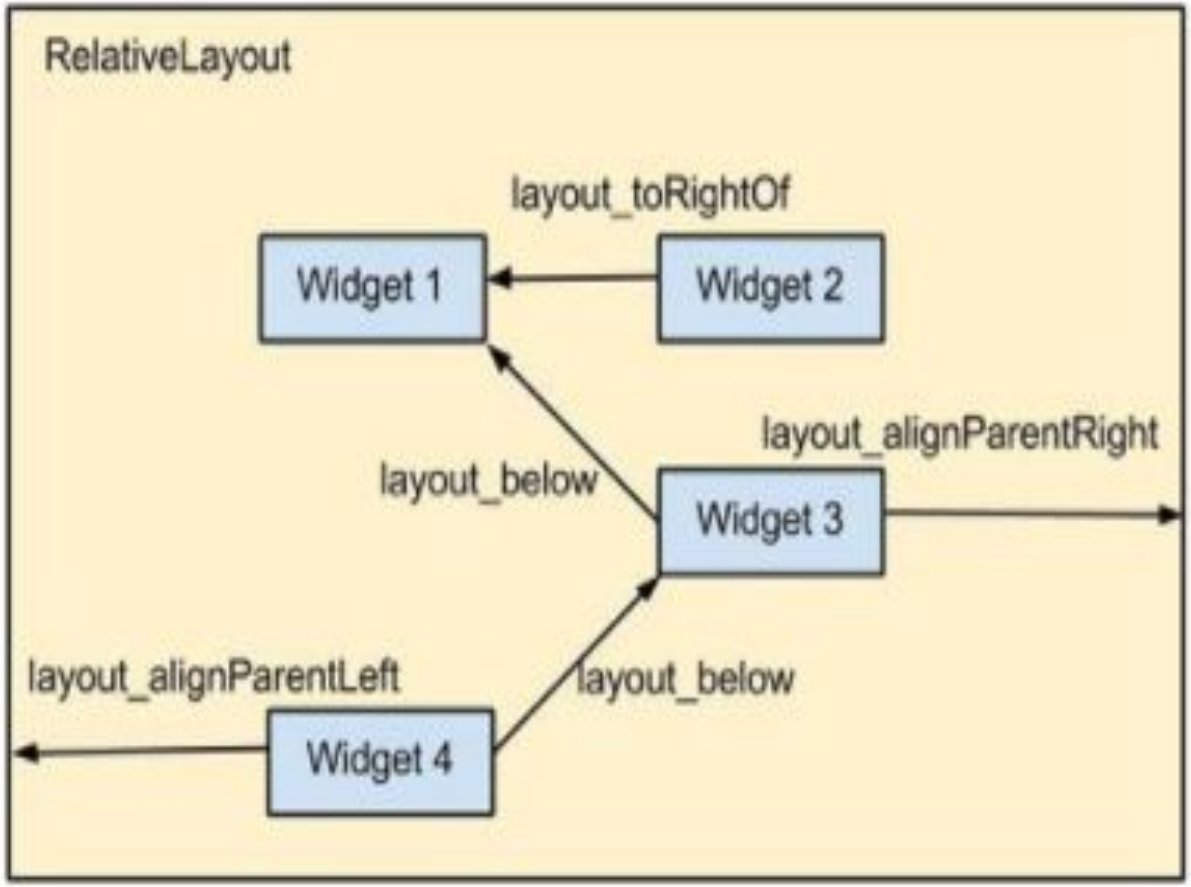
RelativeLayout

- android:layout_alignParentRight: alinha a direita do layout pai
- android:layout_alignParentLeft: alinha a esquerda do layout pai
- android:layout_alignTop: alinha acima de um componente
- android:layout_alignBottom: alinha abaixo de um componente
- android:layout_alignRight: alinha a direita de um componente
- android:layout_alignLeft: alinha a esquerda de um componente

RelativeLayout

- android:layout_marginTop: margem superior do componente
- android:layout_marginBottom: margem abaixo do componente
- android:layout_marginRight: margem a direita do componente
- android:layout_marginLeft: margem a esquerda do componente

RelativeLay



Um dos layouts mais complexos, porém se dominado pode ser o diferencial no seu layout.

RelativeLayout - Exemplo

- Vamos criar o formulário de login utilizando o RelativeLayout
- Crie um novo arquivo de layout chamado login_relative.xml e escolha o layout RelativeLayout no wizard
- Agora vamos começar a colocar os componentes:
- Texto de boas vindas: primeiro componente. Será posicionado automaticamente no início

```
<TextView
    android:id="@+id/mensagemInicio"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/mensagem_login"
 />
```

RelativeLayout - Exemplo

- Label do nome do usuário:
  - Largura: 55dp
  - Posicionado abaixo do elemento anterior (layout_below)
  - Margem superior de 15dp (layout_marginTop)

```
<TextView
    android:id="@+id/textoUsuario"
    android:layout_width="55dp"
    android:layout_height="wrap_content"
    android:text="Usuario"
    android:layout_below="@id/mensagemInicio"
    android:layout_marginTop="15dp"
```

RelativeLayout - Exemplo

- Campo para nome do usuário:
  - Largura: match_parent
  - Posicionado a direita do seu label (layout_toRightOf)
  - Alinhar com o topo do seu label (layout_alignTop)

```
<EditText
    android:id="@+id/campoUsuario"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/textoUsuario"
    android:layout_alignTop="@id/textoUsuario"
/>
```

RelativeLayout - Exemplo

- Label da senha:
  - Largura: 55dp
  - Posicionado abaixo do campo do usuário (layout_below)
  - Margem superior de 15dp (layout_marginTop)

```
<TextView
    android:id="@+id/textoSenha"
    android:layout_width="55dp"
    android:layout_height="wrap_content"
    android:text="Senha"
    android:layout_below="@id/campoUsuario"
    android:layout_marginTop="20dp"
```

RelativeLayout - Exemplo

- Campo da senha:
  - Largura: match_parent
  - Posicionado a direita do label da senha (layout_toRightOf)
  - Alinhar com o topo do seu label (layout_alignTop)

```
<EditText
    android:id="@+id/campoSenha"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:layout_toRightOf="@id/textoSenha"
    android:layout_alignTop="@id/textoSenha"
```

RelativeLayout - Exemplo

- Botão:
  - Largura: wrap_content
  - Posicionado abaixo do campo senha (layout_below)
  - Alinhar com a direita do layout (layout_alignParentRight)

```
<Button
    android:id="@+id/botaoLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Login"
    android:layout_below="@id/campoSenha"
    android:layout_alignParentRight="true"
```

RelativeLayout - Exemplo

- Imagem:
  - Largura: match_parent
  - Posicionado abaixo do botão (layout_below)
  - Posicionado abaixo no layout pai (layout_alignParentBottom)

```
<ImageView
    android:id="@+id/imagemLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/botaoLogin"
    android:layout_alignParentBottom="true"
```

TableLayout

- Este layout comumente é usado quando precisamos listar vários componentes em uma mesma linha, ao longo de uma mesma tela, no formato de uma tabela. Por exemplo, criar um layout com 18 TextView's divididas 3 a 3, ao longo de 6 linhas.
- Para criar as linhas em um TableLayout usamos o componente TableRow.
- O número de colunas no componente TableLayout é definido pelo objeto TableRow que contém a maioria dos componentes.
- A altura de cada linha é determinada pelo componente mais alto dessa linha. Da mesma forma, a largura de uma coluna é definida pelo elemento mais largo nessa coluna – a não ser que configuramos para que as colunas da tabela se alonguem para preencher a largura da tela.

TableLayout

- Por padrão, os componentes são adicionados da esquerda para direita em uma linha, mas podemos especificar o local exato de um componente.
- OBS: por padrão as linhas e colunas são numeradas a partir de 0.
- Uma propriedade importante é android:stretchColumns="0,1“ que indica que as colunas devem ser alongadas horizontalmente,

ScrollVie w

- Componente utilizado para criar uma barra de rolagem quando existem muitos elementos na tela
- Uma ScrollView deve conter apenas um componente filho
- O tamanho deste componente filho irá determinar se deve haver uma barra de rolagem ou não
  - Ou seja, dentro de uma ScrollView deve haver um Layout

ScrollVie w

- Exemplo: coloque uma ScrollView em volta do Layout principal na tela activity_tela_inicial.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent
        "  android:orientation="vertical"
        android:background="#aaaaaa">

        ...
    </LinearLayout>
</ScrollView>
```

ScrollVie w

- Dentro do LinearLayout, altere a altura do elemento de texto e do botão para 500dp
- Execute o app e veja a barra de rolagem fu

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:background="#aaaaaa">
        <TextView
            android:id="@+id/mensagemInicial"
            android:layout_width="match_parent"
            android:layout_height="500dp"
            android:text="TextView" />
        <Button
            android:id="@+id/botaoSair"
            android:layout_width="match_parent"
            android:layout_height="500dp"
            android:text="Sair" />
    </LinearLayout>
</ScrollView>
```

Github

- Link com os códigos gerados nesta aula:
- https://github.com/fabiodasilva500/Aulas-Mobile/tree/Aula-8-Views-Layouts

Referências

- Aula baseada nos textos do livro:

  LECHETA, R. R. Android Essencial com Kotlin. Edição: 1ª ed. Novatec, 2017.

- http://www.devmasterteam.com/
- http://www.telecom.uff.br/pet/petws/downloads/tutoriais/prog_android/Tutorial_programacao_android.pdf
- http://www.regilan.com.br/wp-content/uploads/2014/03/Programa%C3%A7%C3%A3o-para-Android-Aula-02-Manipulando-Layout-TextView-ImageView-EditText-Button-Parte-01.pdf
- http://fernandoanselmo.orgfree.com/curso/curso02/Layouts.pdf