Tipos de leiaute RelativeLayout e LinearLayout

Apresentação

Trabalhar com *layouts* é indispensável em qualquer ambiente de desenvolvimento, mas, quando se está desenvolvendo aplicativos, deve-se ter cuidado redobrado. Afinal, o usuário geralmente estará usando a interface com um espaço visual reduzido frente a uma tela de computador, por exemplo.

A plataforma Android disponibiliza alguns tipos de *layout* que permitem aos aplicativos terem uma robusta forma de interação com o usuário. Os tipos mais utilizados são o RelativeLayout e o LinearLayout. Conhecer seus recursos é fundamental.

Nesta Unidade de Aprendizagem, você verá o conceito desses dois tipos de *layout*, como reconhecê-los e como utilizá-los na construção de um aplicativo para Android.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Diferenciar os tipos RelativeLayout e LinearLayout.
- Exemplificar o uso de RelativeLayout e LinearLayout.
- Aplicar os tipos RelativeLayout e LinearLayout em projetos.

Desafio

Na plataforma Android, dentre os tipos de *layout* existentes, destaca-se o RelativeLayout. Em sua forma de utilização, a manipulação dos seus atributos no arquivo XML é a mais recomendada. Existem, todavia, erros clássicos e outros nem tanto que se pode enfrentar.

Você, como desenvolvedor de uma empresa, recebeu a seguinte tarefa:

Com base nessas informações, analise os trechos e aponte dois erros, um em cada.

Infográfico

O Android possibilita alguns tipos de *layout* para que os desenvolvedores possam aplicar nas interfaces de aplicativos. Entre eles, destaca-se o RelativeLayout e o LinearLayout.

Veja, no Infográfico, como funciona o RelativeLayout aplicado ao Android.

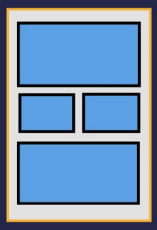
RELATIVELAYOUT APLICADO NO ANDROID

Layouts são muito importantes no Android, pois permitem que a interação dos usuários com os Apps seja a melhor possível.

RelativeLayout

Como o próprio nome sugere, o RelativeLayout mostra **a posição dos componentes em relação uns aos outros.** A posição pode ser especificada em relação a elementos consecutivos ou ao componente pai.

O RelativeLayout **é o layout mais flexível fornecido pelo Android**, pois permite posicionar elementos na tela. Por padrão, definem-se todos os componentes no canto superior esquerdo do *layout*.



Exemplo

Abaixo veja os atributos do RelativeLayout:

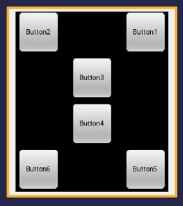
Id: Definição identificação Layout
Gravity: Define a posição x-y de um componente na tela.
IgnoreGravity: Este pode ser adicionado para ignorar a gravidade de um componente.

Em relação ao *layout* pai

Em relação a outras *views*

Disposição

Apresenta quatro diferentes métodos construtores:



RelativeLayout
(Contetxt context)

RelativeLayout
(Contetxt context, AttributeSet attrs)

RelativeLayout
(Contetxt context, AttributeSet attrs, int defStyleAttribute)

RelativeLayout
(Contetxt context, AttributeSet attrs, int defStyleAttribute, int defStyleRes)

Métodos

setGravity(): Define a gravidade dos filhos como centralizado, esquerda ou direita.

setHorizontalGravity(): Usado para posicionar componentes na horizontal.

setVerticalGravity(): Usado para posicionar componentes na vertical.

requestLayout(): Usado para requisitar *layout*.

setIgnoreGravity(): Usado para ignorar a gravidade de um determinado componente.

getGravity(): Usado para obter a posição de componentes.

getAccessibilityClassName(): Retorna a ClassName do objeto.

Vantagem do RelativeLayout

Tipo de *layout* poderoso para desenvolver **interfaces de usuário iterativas.**

☐ **Elimina o aninhamento** de ViewGroups.

☐ Mantém a **hierarquia de *layouts* plana**.

Nos casos em que houver a necessidade de utilizar vários grupos de LinearLayout aninhados, deve ser avaliada a troca para o RelativeLayout.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

Para que um aplicativo tenha sucesso, é inimaginável pensá-lo sem bons recursos de interface de usuário. Por isso, para um desenvolvedor, é importante conhecer quais são os tipos de *layout* disponíveis na plataforma onde se está trabalhando e como explorar esses recursos ao máximo.

No capitulo, Tipos de Layout RelativeLayout e LinearLayout, do livro *Desenvolvimento para Dispositivos Móveis*, você verá qual é o conceito dos tipos RelativeLayout e LinearLayout, como diferenciá-los e alguns exemplos práticos de como utilizá-los.

Boa leitura.

DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

Fabricio Machado da Silva

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Tipos de leiaute: `RelativeLayout` e `LinearLayout`

Objetivos de aprendizagem

Ao final deste texto, você apresentará os seguintes aprendizados:

- Diferenciar os tipos de leiaute `RelativeLayout` e `LinearLayout`.
- Exemplificar o uso do `RelativeLayout` e do `LinearLayout`.
- Aplicar os tipos `RelativeLayout` e `LinearLayout` em projetos.

Introdução

Neste capítulo, você conhecerá e entenderá os conceitos de leiaute para aplicativos Android, sendo os tipos mais trabalhados o `RelativeLayout` e o `LinearLayout`. Assim, abordaremos cada tipo para que você possa identificar e diferenciar e mostraremos sua aplicação individual em projetos de aplicativos para a plataforma Android.

Os tipos de leiaute

Como ficariam os aplicativos em Android se não pudéssemos trabalhar com leiautes? Certamente, esses aplicativos não possibilitariam a interface para a comunicação com o usuário, perdendo a sua maior característica e que faz com que os aplicativos sejam uma febre hoje: facilitar a vida das pessoas que os utilizam por meio de seus dispositivos móveis.

A seguir, descreveremos os leiautes `RelativeLayout` e `LinearLayout`, os quais aparecem na *palette* de leiautes da Figura 1.

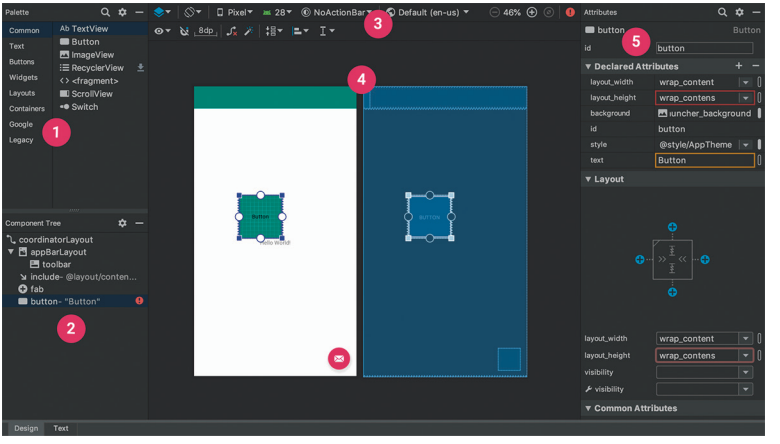


Figura 1. Editor de leiaute do Android Studio.

Fonte: Criar... (2019, documento *on-line*).

`RelativeLayout`

Na plataforma Android, o tipo de leiaute que possibilita posicionar os componentes entre si é chamado `RelativeLayout`. Por exemplo, temos em uma tela um componente `TextView` e desejamos posicionar um componente `Button` à sua esquerda; no arquivo XML, temos a possibilidade de fazer referência ao componente `TextView` para que o componente `Button` seja posicionado em determinando lugar tendo como atribuição a posição do componente `TextView`.

Na Figura 2, podemos perceber os componentes devidamente alinhados graças ao recurso do `RelativeLayout`.

Figura 2. RelativeLayout.
Fonte: Romanato (2016, documento *on-line*).

Podemos observar que o primeiro componente TextView está localizado no topo e o segundo componente TextView logo abaixo dele.

LinearLayout

Um dos tipos de leiaute mais utilizados quando desenvolvemos aplicativos para o Android, possibilita que os componentes sejam organizados tanto em posição vertical quanto horizontal, alinhando todos os componentes em uma única direção, conforme especificado.

Todos os componentes de um LinearLayout (Figura 3) são colocados um após o outro; logo, uma lista vertical terá somente um componente por linha, independentemente da dimensão de sua largura, e uma lista horizontal terá a altura de apenas uma linha (a altura do componente mais alto). O LinearLayout respeita as margens entre os componentes e o seu alinhamento (direita, centro ou esquerda).

Figura 3. `LinearLayout`.
Fonte: Romanato (2016, documento *on-line*).

Fique atento

Os leiautes têm um papel importante em aplicações Android, pois afetam diretamente a experiência do usuário com o aplicativo. Um leiaute pobre e mal implementado pode deixar uma memória ruim em relação ao aplicativo.

Utilização do `RelativeLayout` e do `LinearLayout`

O `RelativeLayout` e o `LinearLayout` apresentam algumas diferenças quanto à sua utilização. O primeiro certamente é considerado o mais complexo dos leiautes, porém, se dominado, possibilita algo realmente surpreendente e capaz de proporcionar uma experiência fantástica para os usuários.

Já o `LinearLayout` é, sem dúvida, o tipo de leiaute mais utilizado pelos desenvolvedores Android, e, se soubermos utilizar todos os seus recursos, também é possível obter leiautes bons.

Aplicação do RelativeLayout

O RelativeLayout possibilita que os componentes filhos determinem a sua posição em relação à posição do componente pai ou à alguma outra, nesse caso especificada por ID (o seu identificador), ou seja, podemos alinhar dois componentes pela borda direita ou posicionar um abaixo do outro, centralizado na tela, centralizado à esquerda, e assim por diante. Por padrão, todos os componentes filhos são desenhados no canto superior esquerdo do leiaute, portanto você deverá definir o posicionamento de cada componente usando as diversas propriedades de leiaute disponíveis (DEITEL; DEITEL; WALD, 2016).

Para referenciarmos a posição de um componente em relação ao outro, utilizamos a notação XML. No exemplo a seguir do XML do RelativeLayout referente à Figura 2, para posicionarmos um componente abaixo do outro, definimos XML android:layout_below e, assim, conseguimos posicioná-lo abaixo do outro que já esta na tela: o primeiro TextView. Já o terceiro TextView está posicionado ao lado direito do segundo TextView: a notação para completar isso é a android:layout_toRightOfe com o ID do elemento que se deseja alinhar. Para referenciarmos a posição de um componente em relação ao outro, utilizamos a notação XML android:layout_below.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/
res/android"
              android:layout _ width="match _ parent"
android:layout _ height="match _ parent">

    <TextView
        android:layout _ width="wrap _ content"
        android:layout _ height="wrap _ content"
      android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Ola Primeiro"
        android:id="@+id/textView4"
        android:layout _ marginTop="41dp"
        android:layout _ alignParentTop="true"
        android:layout _ alignParentLeft="true"
        android:layout _ alignParentStart="true" />
```

```
    <TextView
        android:layout _ width="wrap _ content"
        android:layout _ height="wrap _ content"
      android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Ola Segundo"
        android:id="@+id/textView5"
        android:layout _ below="@+id/textView4"
        android:layout _ alignParentLeft="true"
        android:layout _ alignParentStart="true"
        android:layout _ marginTop="74dp" />

    <TextView
        android:layout _ width="wrap _ content"
        android:layout _ height="wrap _ content"
      android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Ola Terceiro"
        android:id="@+id/textView6"
        android:layout _ alignTop="@+id/textView5"
        android:layout _ toRightOf="@+id/textView5"
        android:layout _ toEndOf="@+id/textView5"
        android:layout _ marginLeft="138dp"
        android:layout _ marginStart="138dp" />
</RelativeLayout>
```

Pelo código, podemos identificar os componentes e seu alinhamento. É possível perceber que o segundo `TextView` tem a notação `android:layout_below` dizendo que ele deverá ficar posicionado abaixo do componente referenciado, nesse caso o `TextView4` (DEITEL; DEITEL; WALD, 2016).

Podemos observar também em outros elementos de tela o mesmo comportamento, todavia com funcionalidades diferentes, como o `android:layout_alignTop`, que faz o componente ser posicionado seguindo o topo do referenciado.

Fique atento

O leiaute é a estrutura por trás da interface com o usuário. No Android, encontramos alguns leiautes padronizados além do `Relative` e `Linear`, como `WebView`, `ListView` e `GridView`.

Aplicação do `LinearLayout`

Conforme mencionado, o `LinearLayout` é o tipo de leiaute mais utilizado em aplicativos Android, pois possibilita o alinhamento dos componentes tanto na posição horizontal quanto vertical. Para decidir qual posição adotar no `LinearLayout`, utilizamos a notação `android:orientation`.

Para ilustrar isso, veja o código XML referente ao leiaute da Figura 3, que citamos anteriormente:

```
<?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:layout _ width="wrap _ content"
        android:layout _ height="wrap _ content"
      android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Ola DevMedia"
        android:id="@+id/textView" />

    <TextView
        android:layout _ width="wrap _ content"
        android:layout _ height="wrap _ content"
      android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Tudo bem?"
        android:id="@+id/textView3" />
</LinearLayout>
```

Agora, se quisermos alterar o leiaute e alinhar os componentes na horizontal, bastaria alterarmos o código conforme apresentado a seguir:

```
<?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">

    <TextView
        android:layout _ width="wrap _ content"
        android:layout _ height="wrap _ content"
      android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Ola DevMedia"
        android:id="@+id/textView" />

    <TextView
        android:layout _ width="wrap _ content"
        android:layout _ height="wrap _ content"
      android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Tudo bem?"
        android:id="@+id/textView3" />
</LinearLayout>
```

Perceba que, nesse segundo exemplo, a *tag* de orientação não está no código, situação em que o Android assume por padrão que a orientação será na horizontal.

Link

Para saber mais sobre os leiautes do Android, acesse o *link* a seguir.

https://qrgo.page.link/cJyAA

Referências

CRIAR uma IU com o Layout Editor. *Android Developers*, [*S. l.*], 2019. Disponível em: https://developer.android.com/studio/write/layout-editor?hl=pt-br. Acesso em: 4 jul. 2019.

DEITEL, P.; DEITEL, H.; WALD, A. *Android 6 para programadores*: uma abordagem baseada em aplicativos. 3. ed. Porto Alegre: Bookman, 2016. 618 p.

ROMANATO, A. Linear, Table e Relative Layouts com Android Studio. *DevMedia*, Rio de Janeiro, 2016. Disponível em: https://www.devmedia.com.br/linear-table-e-relative-layouts-com-android-studio/34127. Acesso em: 4 jul. 2019.

Leitura recomendada

DEITEL, P.; DEITEL, H.; DEITEL, A. *Android*: como programar. 2. ed. Porto Alegre: Bookman, 2015. 690 p.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

A contextualização nos principais tipos de *layout* da plataforma Android é necessária para trabalhar de forma segura com o desenvolvimento dos aplicativos. O Android oferece alguns tipos de *layout*, mas, sem dúvida, entre eles, destaca-se o uso do LinearLayout e do RelativeLayout.

Nesta Dica do Professor, você verá a aplicação do LinearLayout e do RelativeLayout.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **O Android disponibiliza muitos recursos para se trabalhar a interface com o usuário. Dentre eles, destacam-se os layouts, componentes de fundamental importância, sendo o LinearLayout o tipo mais utilizado. Assinale a alternativa correta acerca desse tipo de *layout*:**

A) O LinearLayout tem esse nome justamente porque possibilita o alinhamento de componenentes somente na horizontal.

B) O LinearLayout é o mais utilizado porque permite recursos mais impressionantes que o RelativeLayout.

C) O LinearLayout permite o alinhamento de componentes tanto na horizontal como na vertical.

D) O LinearLayout é o componente de *layout* mais utilizado na plataforma Android, por ser mais fácil de trabalhar.

E) O LinearLayout permite que os componentes sejam posicionados um em relação à posição do outro.

2) **O RelativeLayout, um tipo de *layout* muito poderoso, permite que a interface do aplicativo apresente uma experiência fantástica de interação com os usuários. Assinale a alternativa abaixo que descreve corretamente a principal característica do RelativeLayout:**

A) O RelativeLayout permite que um componente filho tenha a sua posição atribuída em relação à posição de outro componente.

B) O RelativeLayout permite que os componentes sejam alinhados na linha da vertica ou da horizontal sempre empilhados.

C) O RelativeLayout tem a vantagem, em relação ao LinearLayout, de permitir utilizar componentes em colunas.

D) O RelativeLayout é um complemento do LinearLayout, sua utilização está relacionada ao primeiro.

E) O RelativeLayout permite recursos fantásticos, mas sua utilização também requer componentes gráficos diferenciados.

3) **No LinearLayout, tem-se a possibilidade de manipular atributos que permitem aos componentes apresentarem formatos ou comportamentos diferenciados. Um atributo que pode ser manipulado, por exemplo, é o android:layout_weight. Assinale a alternativa abaixo que descreve o que ocorre quando se manipula o valor desse atributo:**

A) Permite que um item de maior peso seja posicionado mais ao fim da lista.

B) Permite que um item de maior peso seja posicionado mais ao topo da lista.

C) Permite que um item de maior peso seja colocado mais ao fundo dos demais.

D) Permite que um item de maior peso possa expandir e usar mais espaço na tela.

E) Permite que um item de maior peso seja exibido com prioridade ao carregar a tela.

4) **O RelativeLayout permite alcançar recursos fantásticos de interação com o usuário; porém, uma das grandes desvantagens desse tipo de *layout* é que, quando se altera um componente, pode ser necessário alterar todos os outros. Assinale a alternativa correta sobre por que isso ocorre:**

A) Ocorre pelo fato da hierarquia de componentes.

B) Ocorre pelo fato da relatividade da posição de um com o outro.

C) Ocorre pelo fato do RelativeLayout encapsular os componentes.

D) O RealtiveLayout não apresenta esse tipo de desvantagem.

E) Ocorre pela necessidade de ajuste manual no posicionamento dos componentes.

5) **"Indicado quando as posições dos componentes podem ser melhor descritas em relação a outro elemento (à esquerda) ou à área de fronteira do pai (lado direito, ou centrado)". De acordo com essa afirmativa, é correto dizer que se trata de:**

A) LinearLayout e RelativeLayout.

B) Sistemas de *layout*.

C) Somente LinearLayout.

D) Essa afirmativa não faz sentido para *layouts*.

E) Somente RelativeLayout.

Na prática

A vantagem da adoção de *layouts* para um aplicativo no Android é, sem dúvida, um recurso que merece atenção dos desenvolvedores. O LinearLayout é o padrão de *layout* a ser criado no Android, mas o RelativeLayout é um tipo muito poderoso para desenvolver interfaces iterativas.

Veja, Na prática, o estudo que Verônica, uma desenvolvedora de aplicações, fez para escolher entre utilizar o RelativeLayout ou o LinearLayout.

Verônica sabe que, sem dúvida, em qualquer aplicação desenvolvida, o *layout* é peça fundamental. Essa camada é responsável pela interação com os usuários da aplicação.

É pelo *layout* que o usuário irá gostar ou não do App; por isso, escolher o melhor tipo de *layout* do Android define o sucesso do App. Desse modo, ela estudou dois caminhos:

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

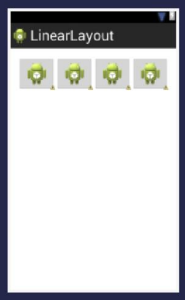
LINEARLAYOUT

Posiciona os componentes em uma única direção: vertical ou horizontal.

São respeitadas as margens e alinhamento (ao centro, à esquerda ou à direita, por meio da manipulação do atributo *gravity*).

Podem ser atribuídos pesos para cada componente, possibilitando que ocupem espaços diferenciados na tela, **evitando que pequenos componentes deixem espaços** desnecessários sobrando em tela.

É possível ajustar os atributos de componentes para que **preencham todos os espaços da tela.**



RELATIVELAYOUT

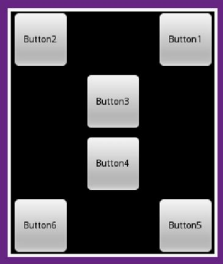
Posiciona os componentes de forma relativa uns aos outros.

A posição de cada um dos componentes pode ser especificada de acordo com a posição do componente irmão, independente das linhas horizontais ou verticais.

É um dos *layouts* mais utilizados no desenvolvimento de aplicações Android.

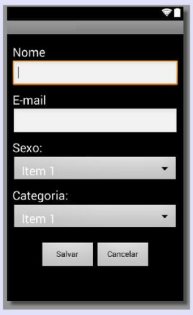
Muito indicado para criar **interfaces de usuário interativas.**

Elimina o aninhamento de ViewGroups e mantém a hierarquia de *layouts* plana, sendo esta a grande diferença entre ele e o LinearLayout.



Declaração do tipo de layout

Pode ser feita de duas formas para os dois tipos:



☐ **Declarando os componentes da Interface de Usuários em XML.**
O Android possibilita um vocabulário XML direto que corresponde às classes e subclasses de View, como *Widgets* e *Layout*.

A vantagem é separar melhor a apresentação do aplicativo do código que controla seu comportamento. **As descrições da IU são externas ao código do aplicativo,** ou seja, é possível modificá-las ou adaptá-las sem modificar o código-fonte e recompilar.

Ex.: É possível criar *layouts* XML para diferentes orientações de tela, diferentes tamanhos de tela de dispositivos e diferentes idiomas.

☐ **Instanciando os componentes do *layout* em tempo de execução.**
O aplicativo pode criar objetos View e ViewGroup e, assim, processar as suas propriedades.

Com base nas informações que estudou, **Verônica optou por utilizar o RelativeLayout,** pois gostaria de um *layout* mais arrojado e inovador que a permitisse **criar mais interações com o usuário.**

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Entenda como funciona o LinearLayout no Android

Neste vídeo, você verá como iniciar o uso do LinearLayout no Android Studio.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Como utilizar o LinearLayout

Neste vídeo, você assiste, de forma rápida e direta, como utilizar o LinearLayout.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

LinearLayout

Neste link, você acessa um site com o Guia do Desenvolvedor Android, onde há toda a explicação sobre o funcionamento do LinearLayout.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.