

DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

Ramon dos Santos Lummertz



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS

Tipos de leiaute: ConstraintLayout e TableLayout

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Diferenciar os tipos ConstraintLayout e TableLayout.
- Exemplificar o uso do ConstraintLayout e TableLayout.
- Aplicar os tipos ConstraintLayout e TableLayout em projetos.

Introdução

Uma boa usabilidade dos aplicativos *mobile* depende muito do modo como os componentes (botões, textos, imagens, também chamados de *views*) são organizados na tela do usuário. Por isso, em um projeto de desenvolvimento de aplicativos *mobiles*, podemos e devemos usar diferentes formas de distribuir as *views* na tela do *device*.

Pelo fato de boa parte do sucesso de uma aplicação depender de seu aspecto visual e pela dificuldade apresentada pela plataforma Android, com sua grande variedade de tamanhos de tela (GLAUBER, 2019), algumas formas de leiaute — que organizam os componentes de visualização em uma interface gráfica (DEITEL; DEITEL; DEITEL, 2015) — são disponibilizadas aos desenvolvedores. Duas delas são o ConstraintLayout e o TableLayout, focos deste capítulo. Assim, a partir de agora você compreenderá as diferenças entre esses dois tipos de leiautes e onde e quando devemos aplicá-los.

Diferenças entre ConstraintLayout e TableLayout

O ConstraintLayout possibilita criar leiautes complexos com uma hierarquia de visualização simples, sem o uso de grupos aninhados, e se assemelha ao RelativeLayout, pois os componentes são exibidos de acordo com os relacionamentos entre os componentes e o leiaute *parent*.

Ainda de acordo com Glauber (2019), o ConstraintLayout permite alinhar componentes baseados em regras.

Os *constraints*, que significam “restrições”, representam a essência por trás do funcionamento desse leiaute. Para estabelecer a posição de uma *view* no ConstraintLayout, você precisa adicionar ao menos uma *constraint* horizontal e uma vertical para a *view*. Cada *constraint* representa uma conexão ou um alinhamento em relação à outra *view*, ao leiaute *parent* ou a outra *view*, e define a posição da *view* a partir de seus eixos vertical e horizontal (*X* e *Y*), motivo pelo qual precisamos definir no mínimo essas duas relações, embora seja comum usar mais de duas relações. Na Figura 1, as *views* são relacionadas ou ancoradas com distâncias definidas em *dp*. O ícone  está restrito a 8dp da *view parent* superior e 16dp da *parent* esquerda. Já o  está restrito a 8dp da *view parent* superior e a 8dp da *view* à sua esquerda.

Por fim, o botão  fica a 16dp na *parent* esquerda e 16dp da *view* acima dele.

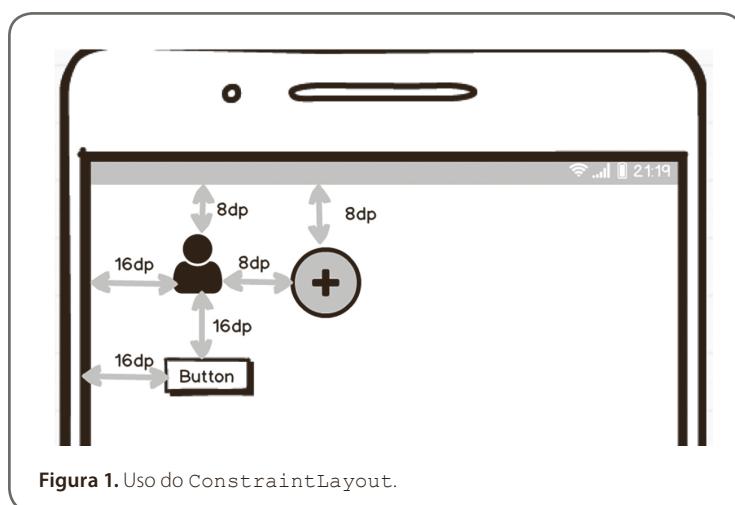


Figura 1. Uso do ConstraintLayout.

O TableLayout é um ViewGroup que agrupa exibições em linhas e colunas, que constitui uma especificação do LinearLayout que organiza seus filhos (*views*) de modo tabular. Você usará o elemento <TableRow> para criar uma linha na tabela. Cada linha tem zero ou mais células, e cada célula pode conter um objeto *view*. A Figura 2 mostra como o TableLayout exibe seus componentes. Esse layout é muito utilizado para criar formulários e telas de *login* e não exibe linhas de borda para suas linhas, colunas ou células.

Ainda conforme Glauber (2019), o TableLayout organiza o layout em formato de tabela, em que cada componente representa uma coluna e pode ocupar duas ou mais colunas.

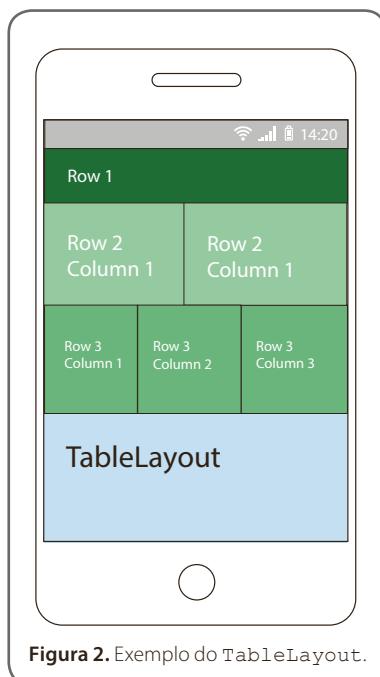


Figura 2. Exemplo do TableLayout.



Link

Você pode pesquisar mais sobre o uso do TableLayout acessando o *link* a seguir.

<https://qrgo.page.link/UcrW5>

Uso do ConstraintLayout e do TableLayout

Para exemplificar o uso dos dois leiautes, utilizaremos o Android Studio 3.4 e o Android SDK API 23.

ConstraintLayout

Conforme Build... (2019, documento *on-line*), o ConstraintLayout refere-se a um ViewGroup que teve sua versão estável em fevereiro de 2017. Esse leiaute está disponível em uma biblioteca de API (*Application Programming Interface*) compatível com o Android 2.3 (nível 9 da API) e superior.

Para definir a posição de uma view no ConstraintLayout, você deve adicionar pelo menos uma restrição horizontal e uma vertical para ela. Cada restrição (*constraint*) representa uma conexão ou um alinhamento para outra view, o leiaute *parent* ou uma diretriz invisível, além de definir a posição da view ao longo do eixo vertical ou horizontal (CONSTRAINTLAYOUT, 2019, documento *on-line*).

Atualmente, existem vários tipos de restrições, como Posicionamento relativo, Margens, Posicionamento centralizado, Posicionamento circular, Comportamento de visibilidade, Restrições de dimensão, *chains*, Objetos do Virtual Helpers e Otimizador (Figura 3).

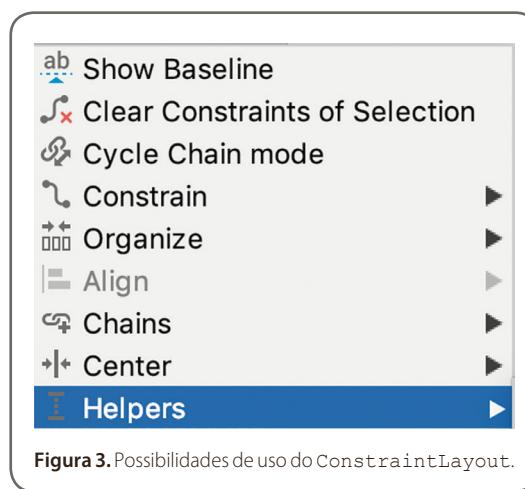


Figura 3. Possibilidades de uso do ConstraintLayout.



Exemplo

Para exemplificar, o trecho de código posiciona o buttonB à esquerda da ViewGroup parent.

```
<Button android:id="@+id/buttonB" ...  
    app:layout_constraintLeft_toLeftOf="parent" />
```

A seguir, mostraremos o uso das *chains* (correntes) para “acorrentar” os elementos em relação às suas posições (Figura 4).

```
<?xml version="1.0" encoding="utf-8"?>  
<android.support.constraint.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity">  
  
<ImageView  
    android:id="@+id/imgPerfil"  
    android:layout_width="128dp"  
    android:layout_height="128dp"  
    android:layout_marginStart="16dp"  
    android:layout_marginTop="16dp"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    tools:srcCompat="@tools:sample/avatars" />  
  
<TextView  
    android:id="@+id/txtName"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="24dp"  
    android:layout_marginEnd="8dp"  
    android:text="Fulano dos Santos"  
    android:textAppearance="@style/TextAppearance.AppCompat.Large"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toEndOf="@+id/imgPerfil"  
    app:layout_constraintTop_toTopOf="parent" />
```

```
<TextView  
    android:id="@+id/txtIdade"  
    android:layout_width="wrap_content"  
    android:layout_height="19dp"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:text="30 Anos"  
    app:layout_constraintEnd_toStartOf="@+id/txtEstadoCivil"  
    app:layout_constraintStart_toEndOf="@+id/imgPerfil"  
    app:layout_constraintTop_toBottomOf="@+id/txtName" />  
  
<TextView  
    android:id="@+id/txtEstadoCivil"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:text="Casado"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toEndOf="@+id/txtIdade"  
    app:layout_constraintTop_toBottomOf="@+id/txtName" />  
</android.support.constraint.ConstraintLayout>
```

Como você pode perceber, a TextView “txtIdade” está com as seguintes restrições:

```
app:layout_constraintEnd_toStartOf="@+id/  
txtEstadoCivil"
```

Ela deve aparecer antes do txtEstadoCivil app:layout_constraintStart_toEndOf="@+id/imgPerfil", ao fim do imgPerfil app:layout_constraintTop_toBottomOf="@+id/txtName" e abaixo do imgPerfil.



Figura 4. Exemplo do uso do ConstraintLayout.



Link

Para saber mais sobre o ConstraintLayout, acesse os *links* a seguir.

<https://qrgo.page.link/tuvTX>

<https://qrgo.page.link/fLWmQ>

TableLayout

No TableLayout, você construirá uma tabela com três linhas — a primeira com apenas uma coluna, na segunda dois TextView formarão as duas colunas e, por fim, na terceira, teremos três objetos, formando, assim, três colunas.

O Arquivo XML ficará conforme o código, cujo resultado você pode observar na Figura 5.

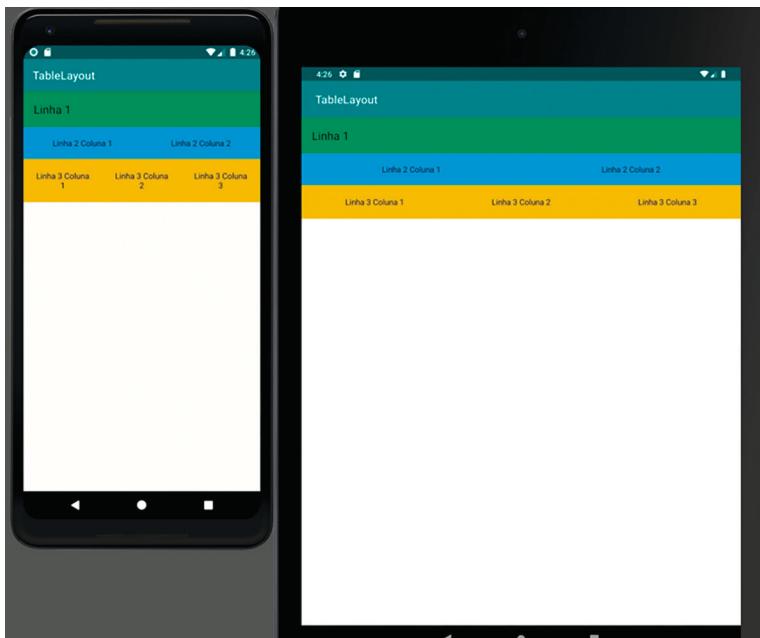


Figura 5. Exemplo do TableLayout.

Conforme TableLayout (2019, documento *on-line*), o TableRow determina que teremos uma linha e que cada componente dentro do <TableRow></TableRow> formará uma coluna. Por exemplo, se você adicionar mais um elemento ao primeiro TableRow, um botão, por exemplo, nossa TableRow, terá duas colunas.

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:shrinkColumns=""
    android:stretchColumns="*"
    android:background="#ffffffff">
    <!-- Linha 1 com uma coluna-->
    <TableRow
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:gravity="center_horizontal">
        <TextView
            android:id="@+id/TextView00"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20dp"
            android:text="Linha 1"
            android:layout_span="3"
            android:padding="18dip"
            android:background="#4CAF50"
            android:textColor="#0000"/>
    </TableRow>
    <!-- Linha 2 com 2 colunas -->
    <TableRow
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:gravity="center_horizontal">
        <TextView
            android:id="@+id/TextView01"
            android:text="Linha 2 Coluna 1"
            android:layout_weight="1"
            android:background="#03A9F4"
            android:textColor="#000000"
            android:padding="18dip"
            android:gravity="center"/>
        <TextView
            android:id="@+id/TextView02"
            android:text="Linha 2 Coluna 2"
            android:layout_weight="1"
            android:background="#03A9F4"
            android:textColor="#000000"
            android:padding="18dip"
            android:gravity="center"/>
    </TableRow>
```

```
<!-- Linha 2 com 3 colunas -->
<TableRow
    android:id="@+id/tableRow1"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">

    <TextView
        android:id="@+id/TextView03"
        android:text="Linha 3 Coluna 1"
        android:layout_weight="1"
        android:background="#FFC107"
        android:textColor="#000000"
        android:padding="20dip"
        android:gravity="center"/>

    <TextView
        android:id="@+id/TextView04"
        android:text="Linha 3 Coluna 2"
        android:layout_weight="1"
        android:background="#FFC107"
        android:textColor="#000000"
        android:padding="20dip"
        android:gravity="center"/>

    <TextView
        android:id="@+id/TextView05"
        android:text="Linha 3 Coluna 3"
        android:layout_weight="1"
        android:background="#FFC107"
        android:textColor="#000000"
        android:padding="20dip"
        android:gravity="center"/>
</TableRow>
</TableLayout>
```



Link

Você pode encontrar mais informações sobre o TableLayout no *link* a seguir.

<https://qrgo.page.link/UcrW5>

Segundo Table (2019, documento *on-line*) e Glauber (2019), o TableLayout apresenta três propriedades interessantes:

- `android:stretchColumns`: indica quais colunas se expandirão caso haja espaço na tela.
- `android:shrinkColumns`: aponta quais colunas ficarão menores caso necessário.
- `android:layout_span`: indica quantas colunas o componente ocupará.

Aplicação dos tipos ConstraintLayout e TableLayout em projetos

Agora que já conhece os dois tipos de leiautes trabalhados, você deve estar se perguntando: “qual devo usar”? Sua escolha precisa se basear na necessidade da tela a ser desenvolvida. O ConstraintLayout é o ideal, conforme Table (2019, documento *on-line*), para quase todas as possibilidades de telas, principalmente se precisar lidar com uma grande fragmentação de dispositivos. Em contrapartida, se a fragmentação não compreende um problema no projeto e a tela a ser desenvolvida não é complexa, o TableLayout acaba se tornando uma opção.

É preciso lembrar que a mesma tela pode ser feita com diferentes leiautes, por isso os dois leiautes descritos têm um cadastro simples, que contém nome, *e-mail*, senha e botão de enviar ou cancelar (Figura 6).

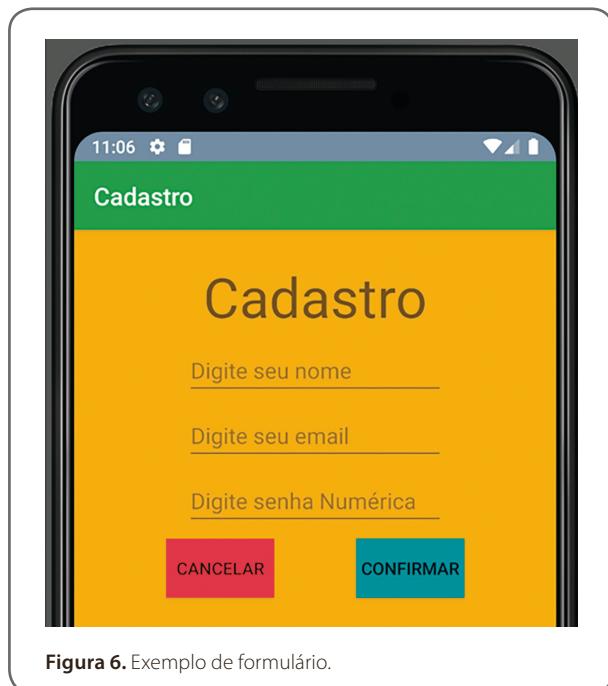


Figura 6. Exemplo de formulário.



Link

Para alguns exemplos de uso de leiautes, acesse o *link* a seguir.

<https://qrgo.page.link/RUDZi>

Uso do ConstraintLayout

Para reproduzir o exemplo da Figura 6, é importante que, ao adicionar os componentes à tela, você dê nome a eles, identificando-os facilmente, a fim de poder relacionar as *views*.

Incialmente, adicione o TextView a Cadastro.

```
<android.support.constraint.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@color/colorAccent"  
    tools:context=".MainActivity">  
  
    <TextView  
        android:id="@+id/txtCadastro"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_marginStart="8dp"  
        android:layout_marginTop="24dp"  
        android:layout_marginEnd="8dp"  
        android:text="Cadastro"  
        android:textAppearance="@style/TextAppearance.AppCompat.Display2"  
        app:layout_constraintEnd_toEndOf="parent"  
        app:layout_constraintStart_toStartOf="parent"  
        app:layout_constraintTop_toTopOf="parent" />
```

Observe que o TextView é nomeado como txtCadastro e, dessa forma, o próximo view a ser adicionado poderá se referenciar a ele (p. ex., o Campo nome).

```
<EditText  
    android:id="@+id/edtNome"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:ems="10"  
    android:hint="Digite seu nome"  
    android:inputType="textPersonName"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/txtCadastro" />
```

Novamente, é possível observar que o EditText tem um identificador edtNome, que se inicia abaixo da view txtCadastro.

Os botões têm uma peculiaridade: ambos estão um ao lado do outro, motivo pelo qual devemos acorrentá-los.

```
<Button  
    android:id="@+id btnCancel"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:background="@color/btnRed"  
    android:text="Cancelar"  
    app:layout_constraintEnd_toStartOf="@+id(btnConfirmar)"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/editPasw" />  
  
<Button  
    android:id="@+id btnConfirmar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:background="@color/btnGreen"  
    android:text="Confirmar"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toEndOf="@+id btnCancel"  
    app:layout_constraintTop_toBottomOf="@+id/editPasw" />
```

Observe que `btnCancel` está com o atributo:

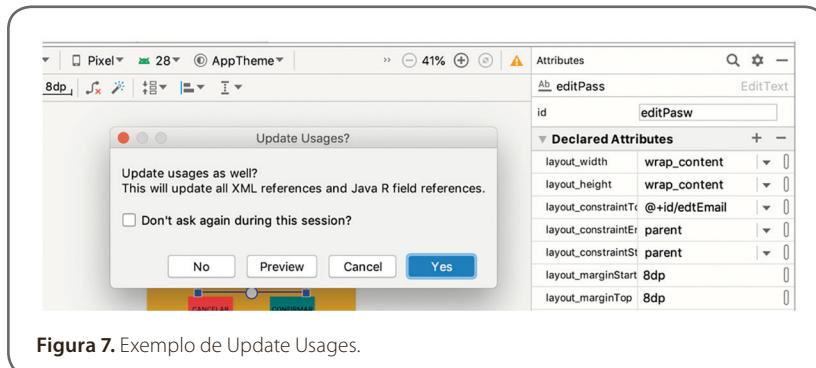
`layout_constraintEnd_toStartOf="@+id(btnConfirmar)"`

e que `btnConfirmar` está com o atributo:

`app:layout_constraintStart_toEndOf="@+id btnCancel".`

Dessa forma, os botões ficaram lado a lado, como na Figura 6.

Caso você se esqueça de nomear suas *views* e já tenha feito as *constraints*, recomenda-se renomeá-los na guia Attributes e fazer o Update Usages (Figura 7), para que suas *constraints* sejam resguardadas.



Ao fim, teremos o código descrito a seguir.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorAccent"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/txtCadastro"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="8dp"
        android:text="Cadastro"
        android:textAppearance="@style/TextAppearance.AppCompat.Display2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/edtNome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="Digite seu nome"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtCadastro" />
```

```
<EditText  
    android:id="@+id/edtEmail"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:ems="10"  
    android:hint="Digite seu email"  
    android:inputType="textEmailAddress"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/edtNome" />  
  
<EditText  
    android:id="@+id/editPasw"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:ems="10"  
  
    android:hint="Digite senha Numérica"  
    android:inputType="numberPassword"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/edtEmail" />  
  
<Button  
    android:id="@+id btnCancel"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:background="@color/btnRed"  
    android:text="Cancelar"  
    app:layout_constraintEnd_toStartOf="@+id(btnConfirmar)"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/editPasw" />  
  
<Button  
    android:id="@+id	btnConfirmar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:background="@color/btnGreen"  
    android:text="Confirmar"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toEndOf="@+id btnCancel"  
    app:layout_constraintTop_toBottomOf="@+id/editPasw" />  
/>
```

Uso do TableLayout

Para reproduzir o exemplo da Figura 6, o primeiro passo consiste em criar um TableLayout e a TableRow.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorAccent"
    tools:context=".MainActivity">
    <TableRow>

    </TableRow>
</TableLayout>
```

Os elementos que formam uma coluna são colocados dentro das TableRow.

```
<TableRow android:gravity="center_horizontal">
    <TextView
        android:id="@+id/txtCadastro"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="8dp"
        android:text="Cadastro"
        android:textAppearance="@style/TextAppearance.AppCompat.Display2" />
</TableRow>
<TableRow android:gravity="center_horizontal">
    <EditText
        android:id="@+id/editNome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:hint="Digite seu nome"
        android:inputType="textPersonName"></EditText>
</TableRow>
```

A última TableRow conterá dois botões e, portanto, duas colunas.

```
<TableRow android:gravity="center_horizontal">

    <Button
        android:id="@+id btnCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color	btnRed"
        android:text="Cancelar" />

    <Button
        android:id="@+id btnCancelar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:background="@color	btnGreen"
        android:text="Confirmar" />
</TableRow>
```

Por fim, obtemos o código descrito a seguir.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorAccent"
    tools:context=".MainActivity">
    <TableRow android:gravity="center_horizontal">
        <TextView
            android:id="@+id/txtCadastro"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="24dp"
            android:layout_marginEnd="8dp"
            android:text="Cadastro"
```

```
        android:textAppearance="@style/TextAppearance.AppCompat.Display2" />
    </TableRow>
    <TableRow android:gravity="center_horizontal">
        <EditText
            android:id="@+id/editNome"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:ems="10"
            android:hint="Digite seu nome"
            android:inputType="textPersonName"></EditText>
    </TableRow>
    <TableRow android:gravity="center_horizontal">
        <EditText
            android:id="@+id/editEmail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:ems="10"
            android:hint="Digite seu email"
            android:inputType="textEmailAddress" />
    </TableRow>
    <TableRow android:gravity="center_horizontal">
        <EditText
            android:id="@+id/editPasw"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:ems="10"
            android:hint="Digite sua senha"></EditText>
    </TableRow>
    <TableRow
        android:gravity="center_horizontal"      >
        <Button
            android:id="@+id btnCancel"
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:background="@color/btnRed"
            android:text="Cancelar" />
        <Button
            android:id="@+id btnCancel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:background="@color/btnGreen"
            android:text="Confirmar" />
    </TableRow>
</TableLayout>
```



Referências

BUILD a Responsive UI with ConstraintLayout. *Android Developers*, [S. l.], 2019. Disponível em: <https://developer.android.com/training/constraint-layout>. Acesso em: 26 jun. 2019.

CONSTRAINTLAYOUT. *Android Developers*, [S. l.], 2019. Disponível em: <https://developer.android.com/reference/android/support/constraint/ConstraintLayout>. Acesso em: 26 jun. 2019.

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

GLAUBER, N. *Dominando o Android com Kotlin*. São Paulo: Novatec, 2019. 1064 p.

TABLE. *Android Developers*, [S. l.], 2019. Disponível em: <https://developer.android.com/guide/topics/ui/layout/grid>. Acesso em: 26 jun. 2019.

TABLELAYOUT. *Android Developers*, [S. l.], 2019. Disponível em: <https://developer.android.com/reference/android/widget/TableLayout.html>. Acesso em: 26 jun. 2019.

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

Conteúdo:



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS