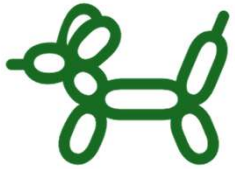
Desenvolvimento de Apps para Mobile

Fabrício Tonetto Londero

Aula 04 - Layouts

O que é um Layout?

É a estrutura visual de um aplicativo Android.



Define como os componentes (TextView, Button, ImageView, etc.) serão organizados na tela.

São definidos no arquivo XML (res/layout/).

Tipos de Layouts no Android

- **LinearLayout**
- **RelativeLayout**
- **ConstraintLayout** (recomendado pelo Google)
- **FrameLayout**
- **TableLayout**
- **GridLayout**
- **ScrollView** (para rolagem)

LinearLayout

Mais simples e fácil de entender

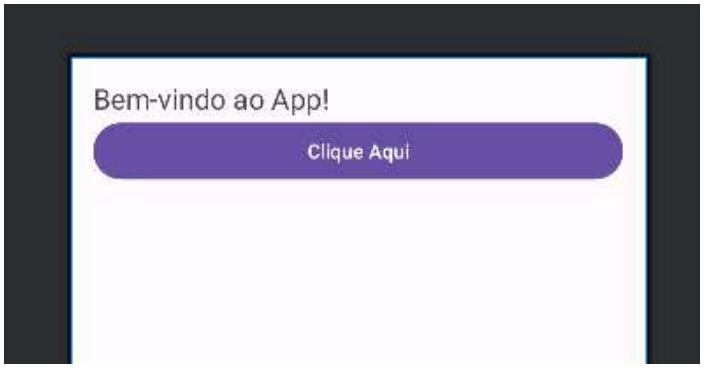
Organiza os elementos em uma única direção (vertical ou horizontal).

Vantagens: Fácil de usar, ideal para layouts simples.

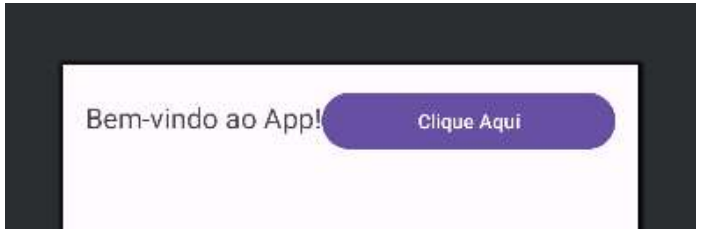
Desvantagens: Pode desperdiçar espaço, já que cada elemento ocupa toda a largura disponível.

```
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/andro
id"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="16dp">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Bem-vindo ao App!"
      android:textSize="20sp"/>

   <Button
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Clique Aqui"/>

</LinearLayout>
```

Vertical



Horizontal



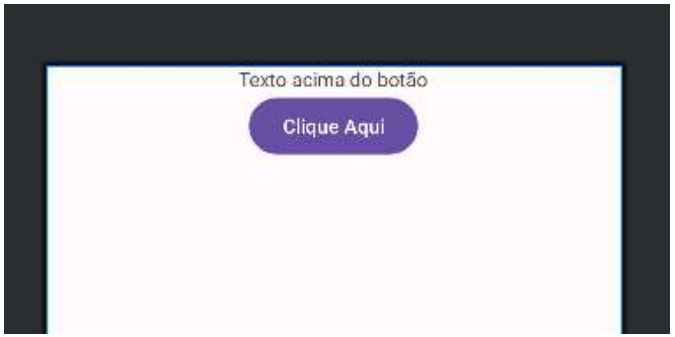
RelativeLayout

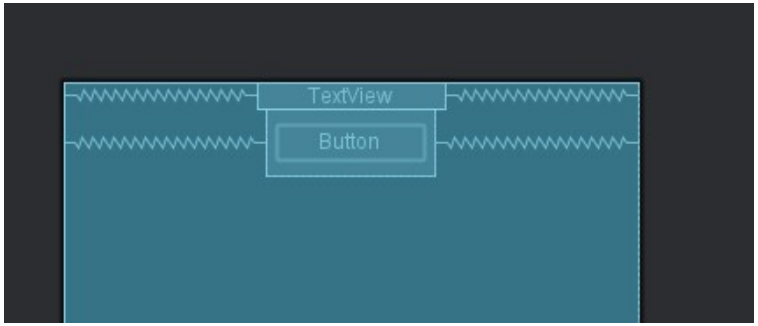
Pouco usado atualmente, substituído pelo ConstraintLayout

Vantagens: Flexível, permite posicionamento preciso.

Desvantagens: Código mais complicado, menos eficiente.

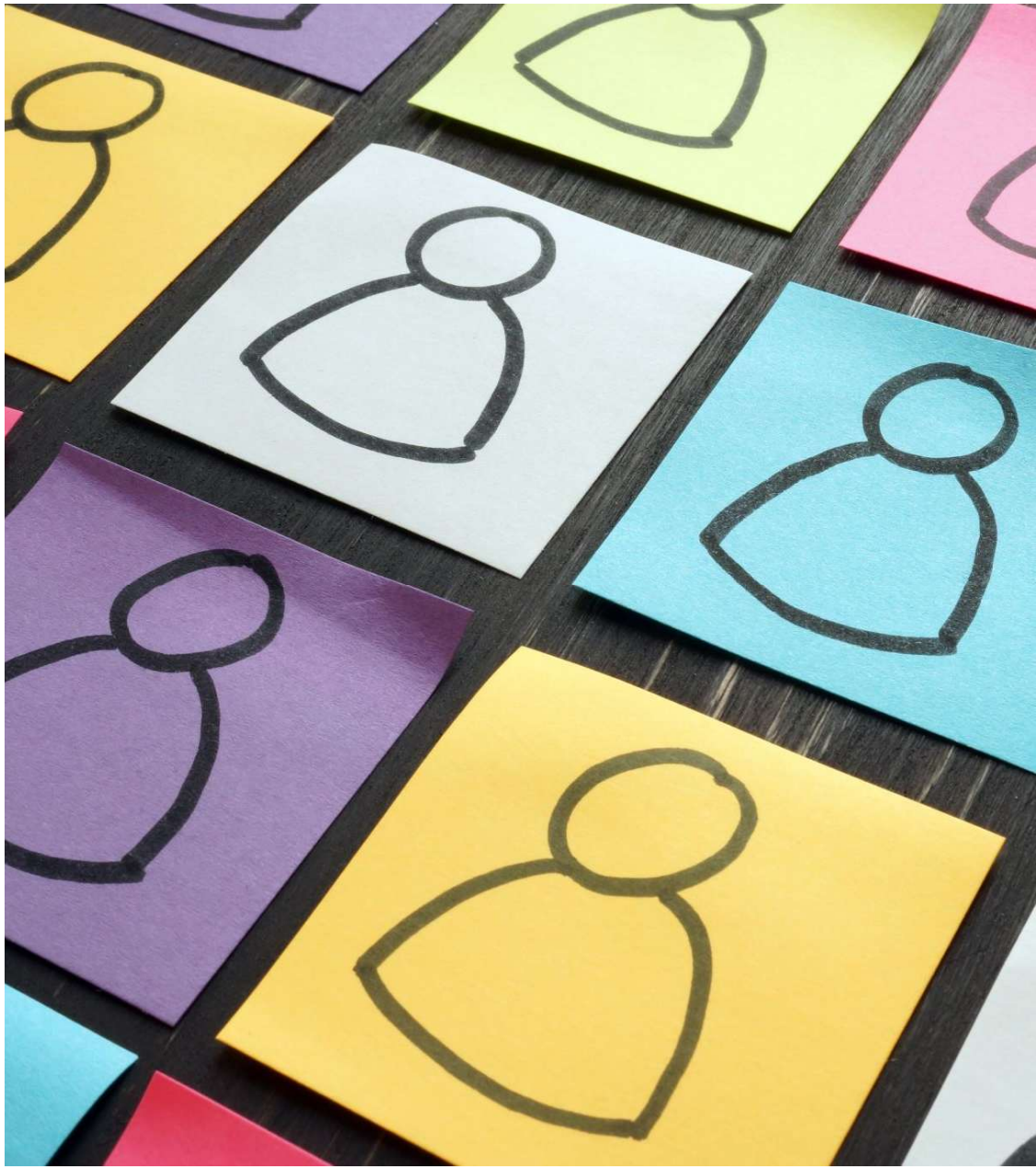
```
<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/andro
id"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Texto acima do botão"
      android:layout_centerHorizontal="true"/>

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Clique Aqui"
      android:layout_below="@id/textView"
      android:layout_centerHorizontal="true"/>
</RelativeLayout>
```



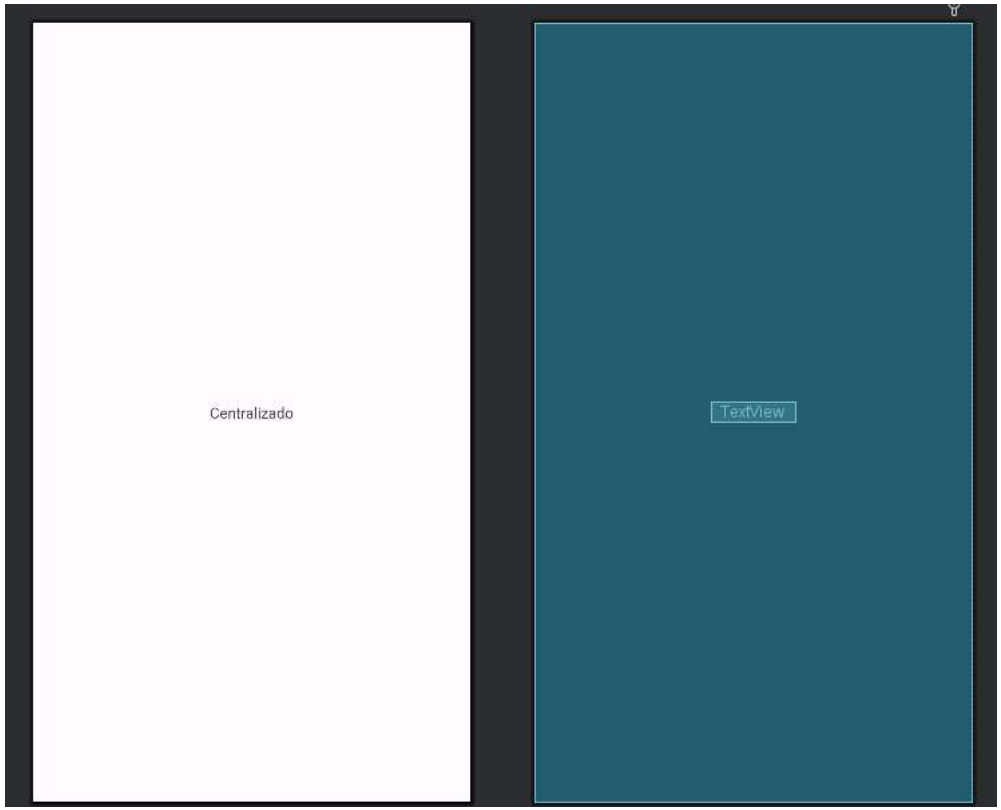


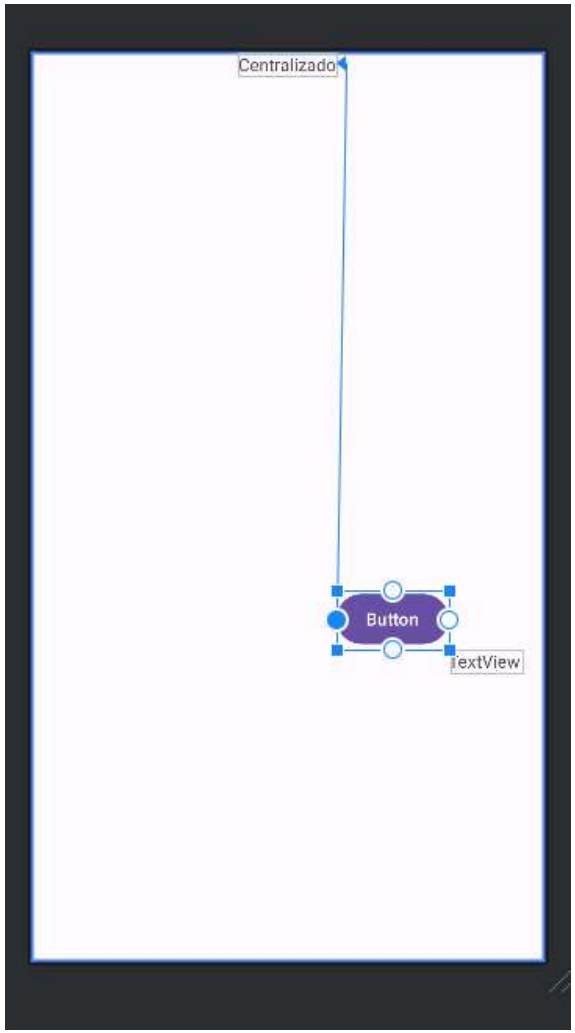
ConstraintLayout

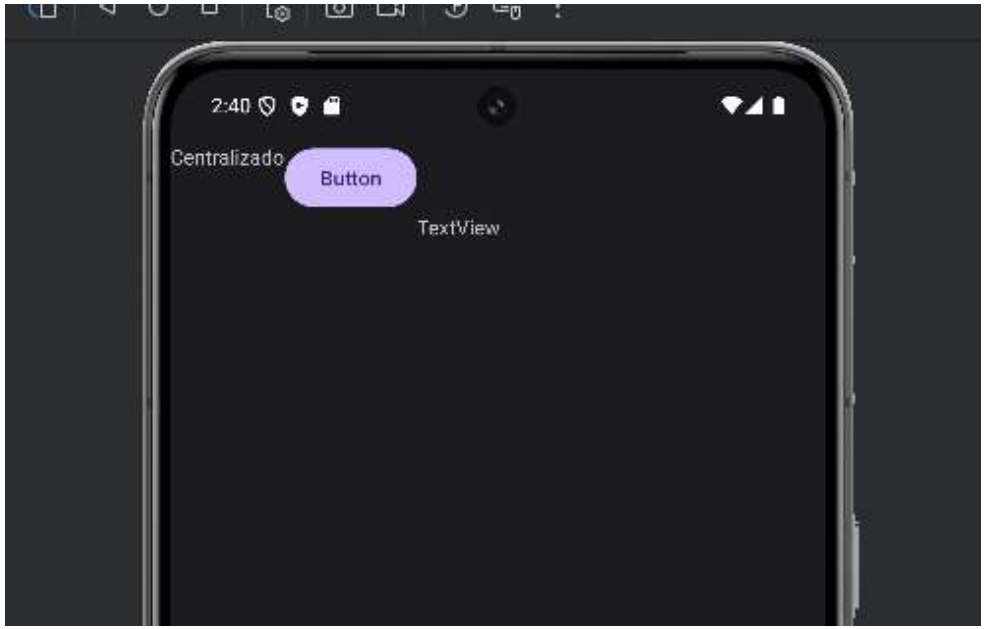
- Melhor opção atualmente
- Opção padrão
- **Substitui o RelativeLayout e LinearLayout**, pois é mais eficiente e flexível.
- **Vantagens**: Mais rápido, mais otimizado, recomendado pelo Google.
- **Desvantagens**: Mais complexo de aprender no início.



```
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Centralizado"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```




Centralizado
Button
TextView


2:40
Centralizado
Button
TextView

FrameLayout

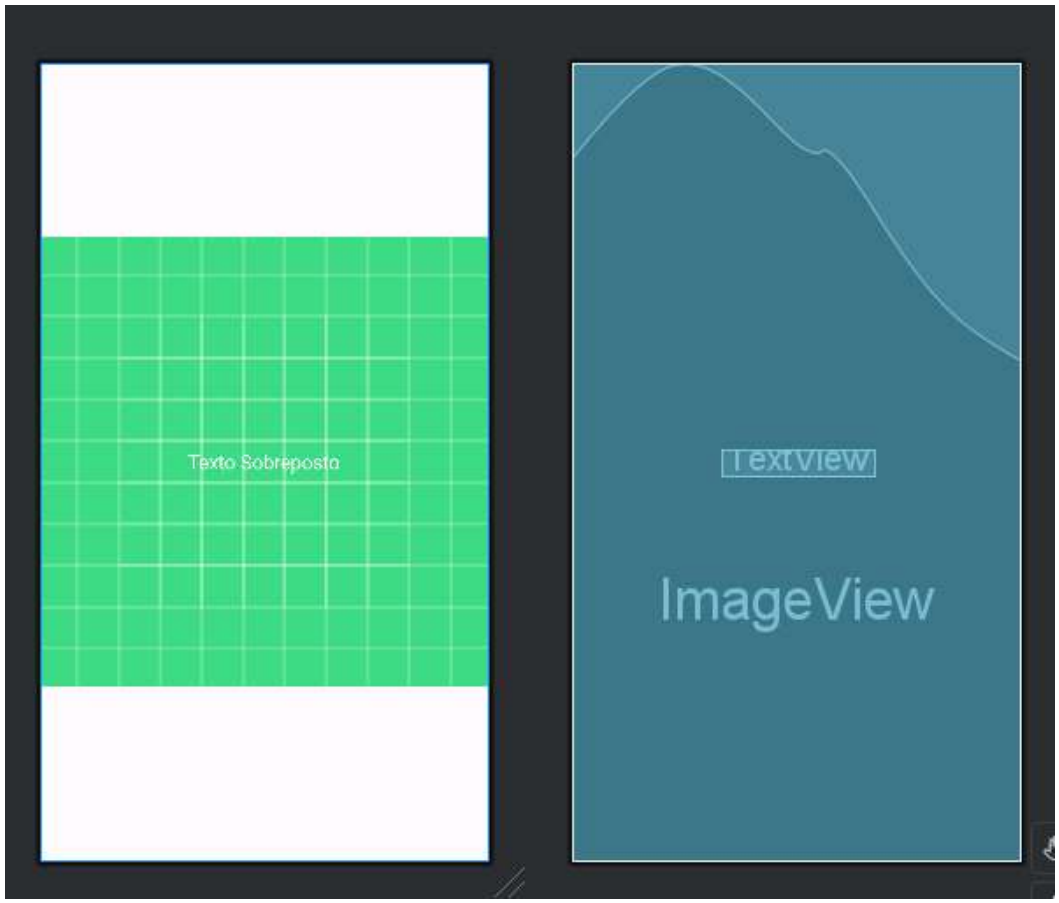
Para sobreposição de elementos

Usado para sobrepor elementos, como botões flutuantes e animações.

Vantagens: Forma de fazer sobreposição de elementos.

Desvantagens: Difícil de alinhar múltiplos elementos.

```
<FrameLayout

xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <ImageView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:src="@drawable/background"/>

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Texto Sobreposto"
      android:textSize="18sp"
      android:textColor="#FFFFFF"
      android:layout_gravity="center"/>
</FrameLayout>
```



TableLayout

Para tabelas, raramente usado

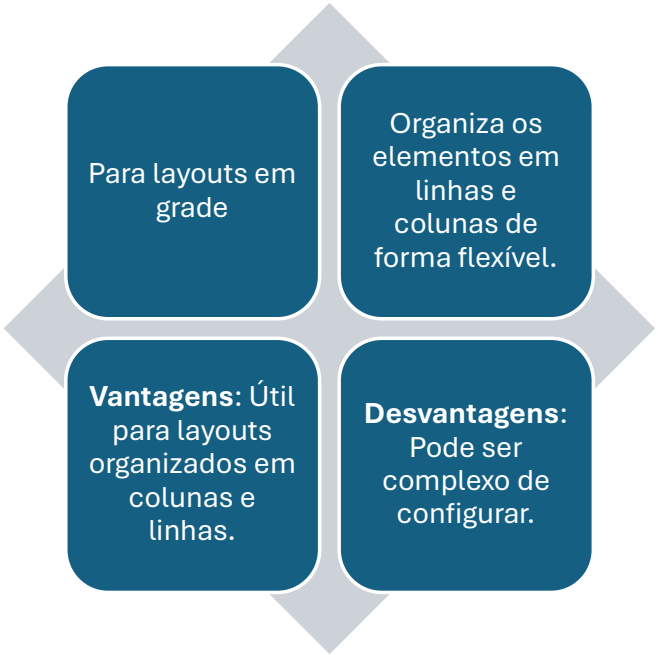
Organiza os elementos em linhas e colunas, como uma tabela.

Vantagens: Útil para tabelas.

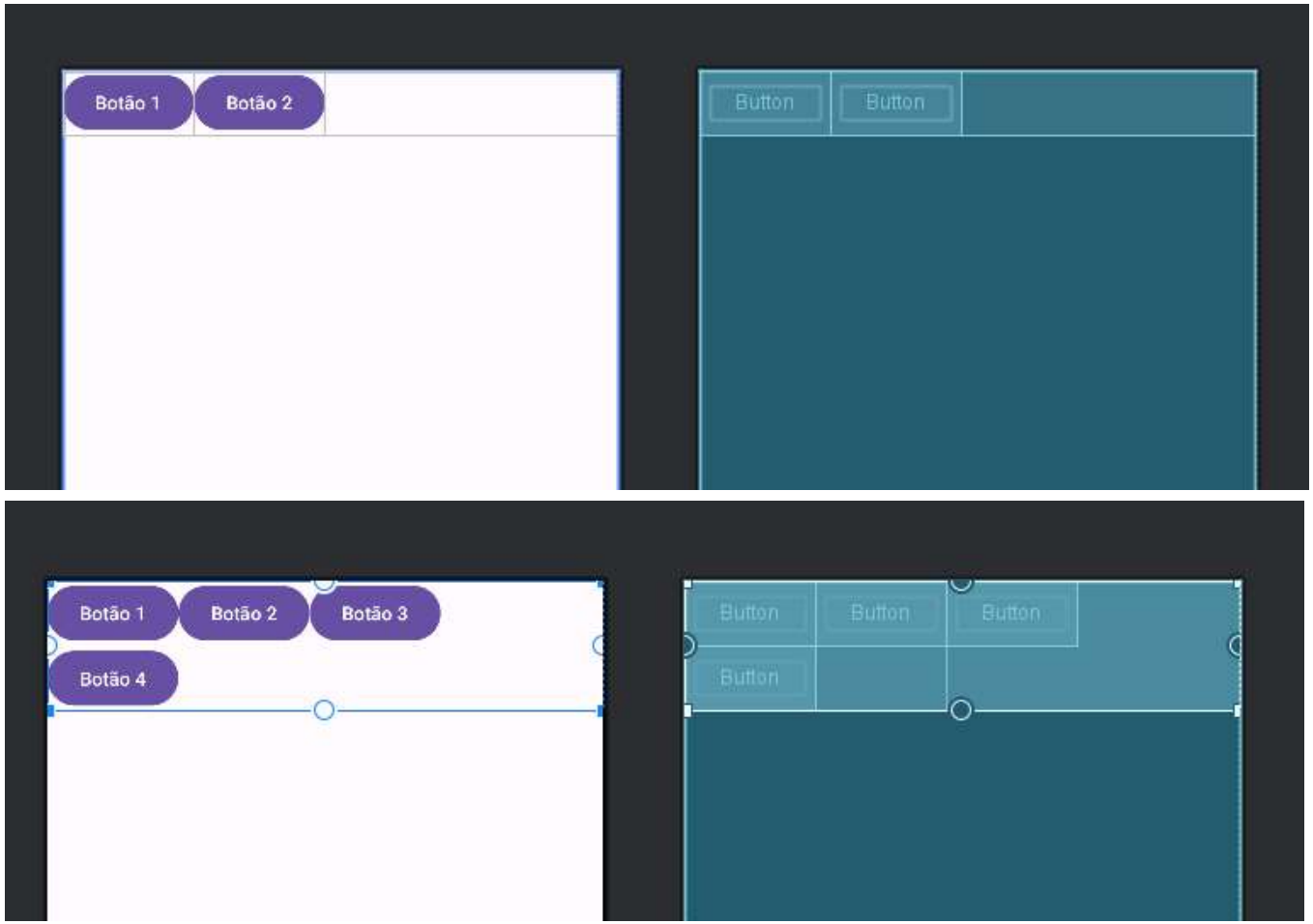
Desvantagens: Pouco flexível.

```
<TableLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content">

  <TableRow>
    <TextView
      android:text="Nome"
      android:layout_weight="1"/>
    <TextView
      android:text="Idade"
      android:layout_weight="1"/>
  </TableRow>

  <TableRow>
    <TextView
      android:text="Carlos"
      android:layout_weight="1"/>
    <TextView
      android:text="25"
      android:layout_weight="1"/>
  </TableRow>

</TableLayout>
```

GridLayout


Para layouts em grade
Organiza os elementos em linhas e colunas de forma flexível.
Vantagens: Útil para layouts organizados em colunas e linhas.
Desvantagens: Pode ser complexo de configurar.

```
<GridLayout

xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:columnCount="2">

   <Button
      android:text="Botão 1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

   <Button
      android:text="Botão 2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>
</GridLayout>
```



ScrollView

Usado quando o conteúdo da tela é maior que o espaço disponível.

Para permitir rolagem

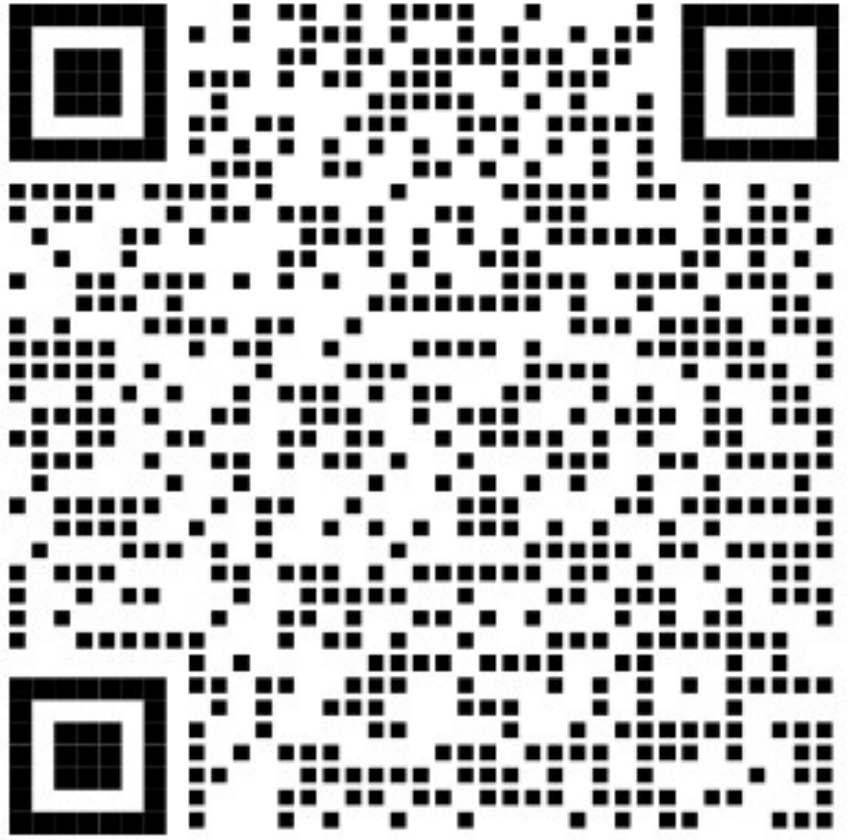
Vantagens: Permite rolagem.

Desvantagens: Não suporta múltiplos filhos diretos.

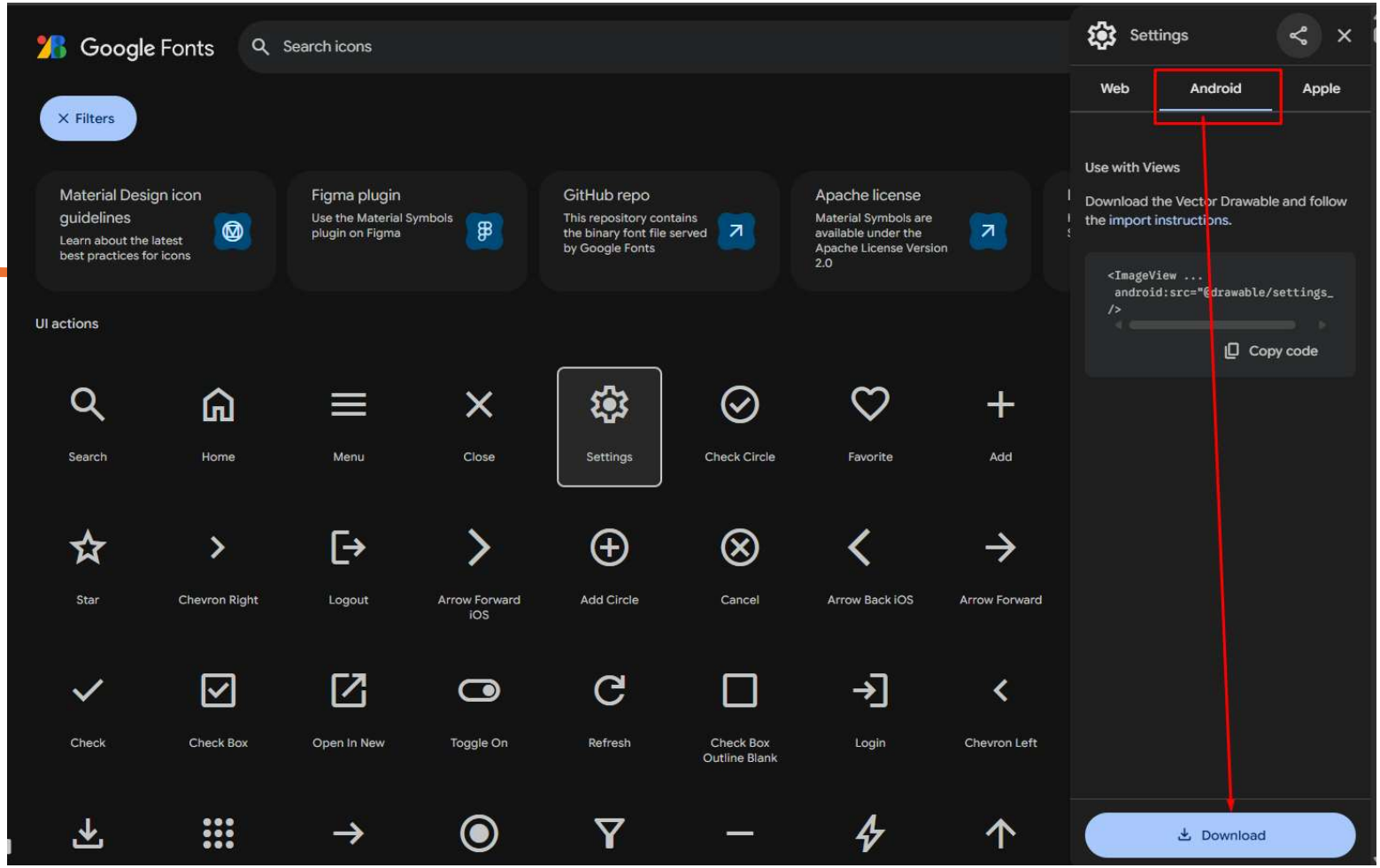
```
<ScrollView

xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
      android:text="Texto grande que precisa de rolagem..."
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

  </LinearLayout>
</ScrollView>
```

Adicionando ícones / imagens

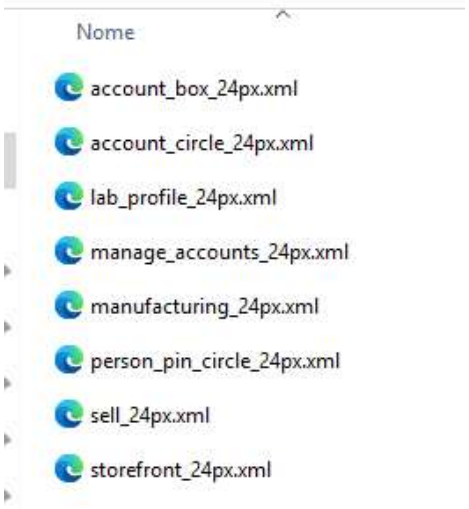
- https://fonts.google.com/icons?icon.size=24&icon.color=%23e3e3e3&icon.platform=android

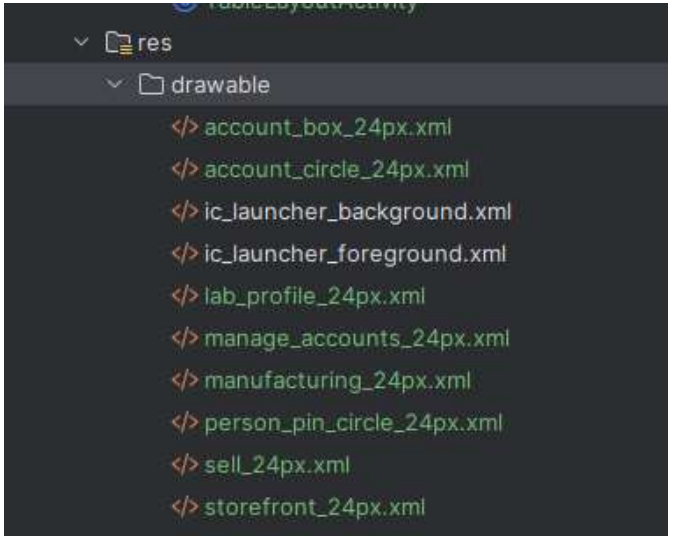


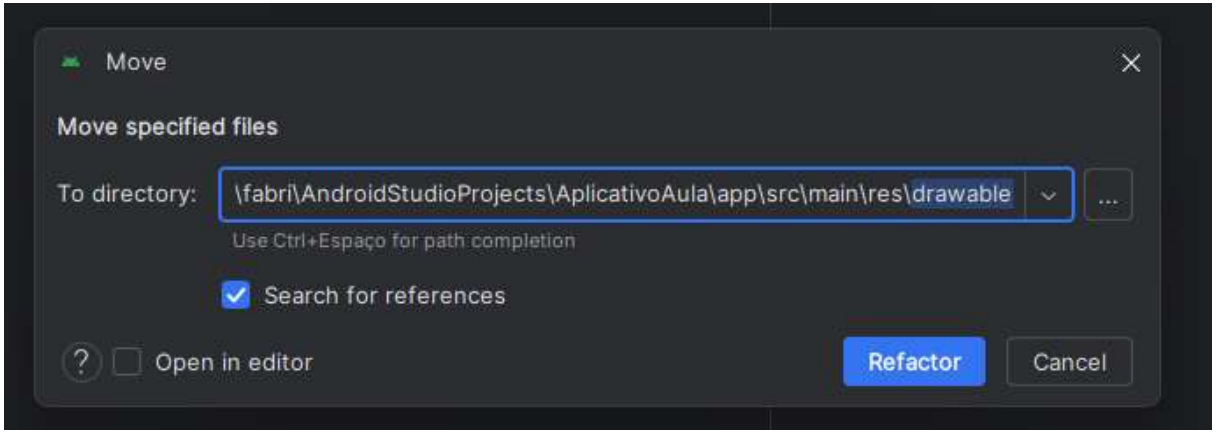
Adicionando ícones / imagens



Adicionando ícones / imagens







Exercício 1

- Use um **ConstraintLayout**.
- Adicione um **ImageView** centralizada na tela.
- Adicione uma **TextView** abaixo da imagem, alinhada ao centro.
- Adicione um **Button** fixado na parte inferior da tela.
- Faça a **TextView** e a **ImageView** ficarem centralizadas, independentemente do tamanho da tela.

Exercício 2

- Criar um layout que tenha uma lista de **TextViews** dentro de um **ScrollView** para permitir a rolagem.
- Dentro dele, adicione um LinearLayout com 10 TextViews com textos diferentes.
- Teste em um celular para verificar se a rolagem está funcionando corretamente.
- Adicione um **Button** fixado na parte inferior da tela que permaneça visível ao rolar a tela.

Exercício 3

- Criar um **FrameLayout** com elementos sobrepostos.
- Adicione uma ImageView de fundo.
- Adicione uma TextView com um título centralizado sobre a imagem.
- Adicione um Button sobreposto no canto inferior direito da tela.
- **Desafio Extra:** Faça com que o botão tenha um fundo semi-transparente para não esconder a imagem.

Exercício 4

- Criar um layout responsivo para um **painel de controle (dashboard)** usando **GridLayout**.
- Use um **GridLayout** com **2 colunas e 2 linhas**.
- Adicione **4 botões** representando seções diferentes (exemplo: "Usuários", "Vendas", "Relatórios", "Configurações").
- Cada botão deve ter um **ícone** e um texto abaixo do ícone.
- Faça os botões mudarem de cor ao serem pressionados