EDC
DR2 Des. Android

Desenvolvimento de Interfaces Android

Engenharia de Computação - 2020 / 4

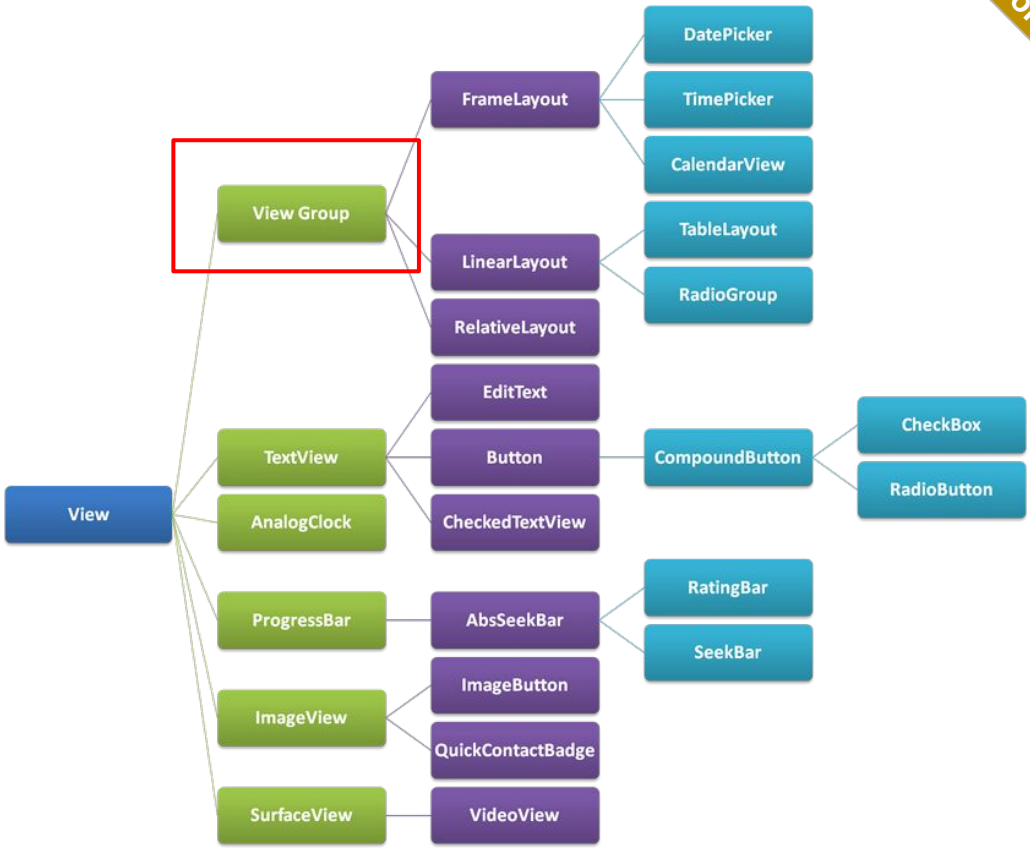
Android Layouts, Estilos e Temas

Views

A View em aplicativos Android é o componente básico para uma interface de usuário.

É a classe de nível mais alto para qualquer componente de interface do usuário ou widget que você pode usar em seus aplicativos.

Todos os widgets da interface do usuário que você viu antes, como Button e TextView, são subclasses da classe View.



Layouts

Um layout define a estrutura visual de uma interface de usuário, como uma Activity ou widget de aplicativo. Você pode declarar um layout de duas maneiras:

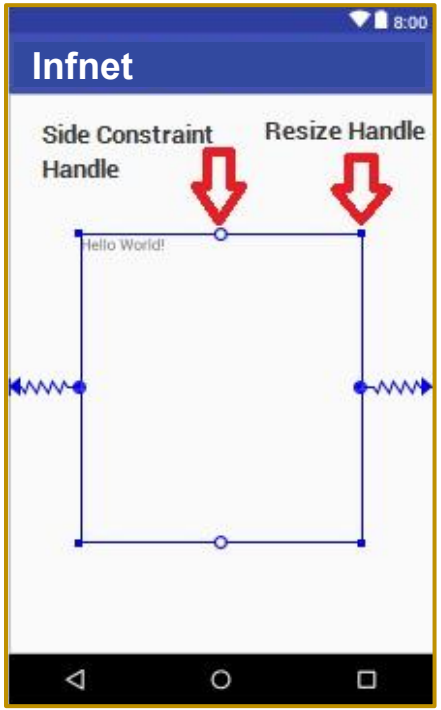
1. Declare elementos do usuário em XML: o Android fornece um vocabulário XML direto que corresponde às classes e subclasses de View, como widgets e layouts.
2. Instancie elementos de layout em tempo de execução: seu aplicativo pode criar objetos View e ViewGroup (e manipular suas propriedades) programaticamente.

Layouts - Constraint Layout

Constraint Layout é um poderoso novo mecanismo de layout lançado como parte da biblioteca Constraint Layout no Android Studio 2.2 e versões posteriores.

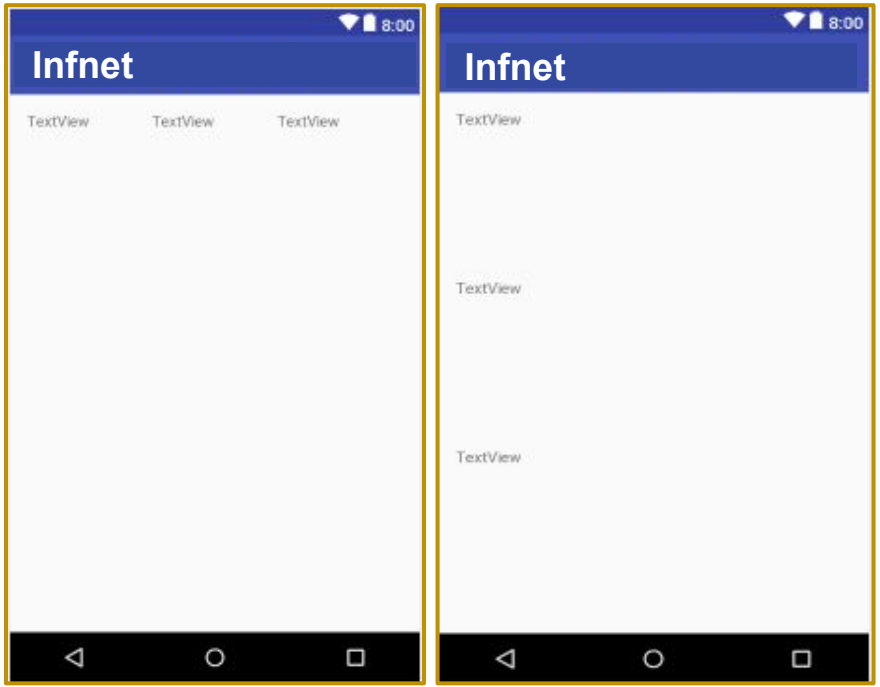
Ele é totalmente integrado ao editor de layout do Android Studio para que você possa construir o layout completo sem a necessidade de editar o XML manualmente.

Esse layout nos permite especificar as restrições que decidiriam a posição de cada subelemento dentro do layout.



Layouts - Linear Layout

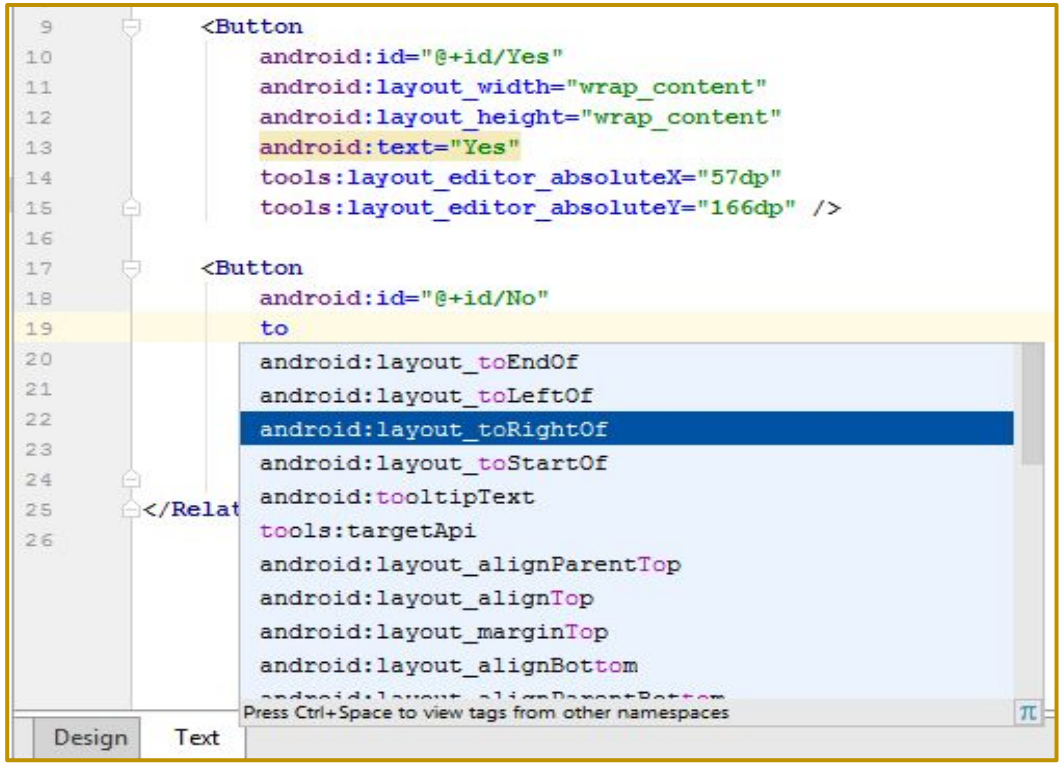
Um Linear Layout é um layout que organiza outras visualizações horizontalmente em uma única coluna ou verticalmente em uma única linha.



Layouts - Relative Layout

O RelativeLayout organiza os widgets em posições em relação uns aos outros.

Por exemplo, você pode posicionar um botão no layout à esquerda, direita ou em cima de outro botão.

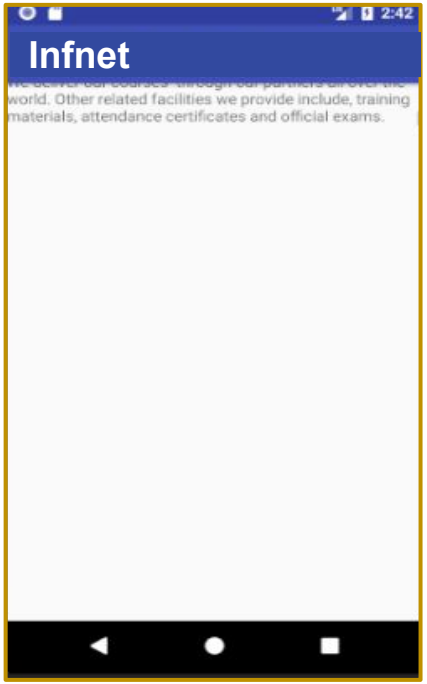


```
<TableLayout
        android:layout_width="368dp"
        android:layout_height="495dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
</TableLayout>
```

Layouts - ScrollView Layout

ScrollView é um grupo de visualização que permite que a hierarquia de visualização colocada dentro dele seja rolada.

```
<ScrollView

    android:layout_width="wrap_content"
    android:layout_height="40dp">
    <TextView

android:layout_width="wrap_content"
        android:layout_height="16dp"
        android:text“ x x x x x x x " />

</ScrollView>
```



Estilos e Temas

Construir seu aplicativo Android requer mais do que apenas escrever um código Kotlin.

Como já viu, você precisa construir o layout de suas telas para fornecer a interface do usuário. Esses arquivos de layout são arquivos XML salvos em /res/layout e são apenas um exemplo dos muitos arquivos de recursos que você pode adicionar ao seu aplicativo Android.

Se precisar de dados estáticos e de contato (imagens, strings ... etc.), Você deve adicioná-los como recursos em seu aplicativo, ou seja, externalizá-los.

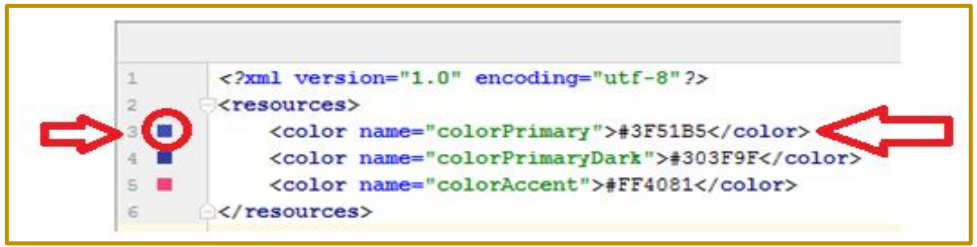
> Adicionar os recursos do seu aplicativo em uma pasta separada melhora sua experiência de desenvolvimento e produz um aplicativo mais robusto por vários motivos, como manutenção e depuração de seu aplicativo.

Estilos

O estilo é uma coleção de atributos que especifica a aparência e o formato de uma View ou janela.

Um estilo pode especificar atributos como altura, preenchimento, cor da fonte, tamanho da fonte, cor do plano de fundo e muito mais.

Um estilo é definido em um recurso XML separado do XML que especifica o layout.



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

https://material.io/resources/color

Temas

Quando um recurso é usado para definir o estilo de toda aplicação, ele é chamado de tema. Disponível para aplicativos voltados para o Android 5.0 (API de nível 21) ou superior.

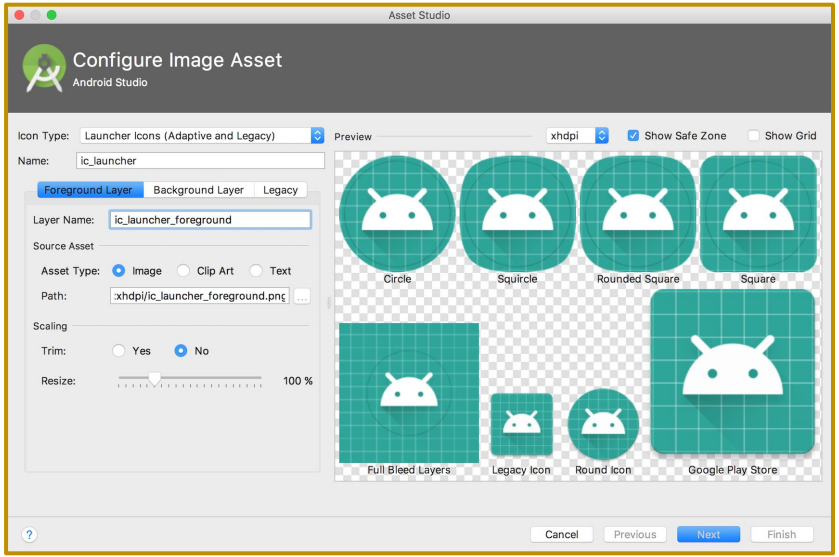
```
<application
     android:allowBackup="true"
     android:icon="@mipmap/ic_launcher"
     android:label="@string/app_name"
     android:roundIcon="@mipmap/ic_launcher_round"
     android:supportsRtl="true"
     android:theme="@style/AppTheme">
     <activity android:name=".MainActivity">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />

          <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
```

Ícones Adaptativos

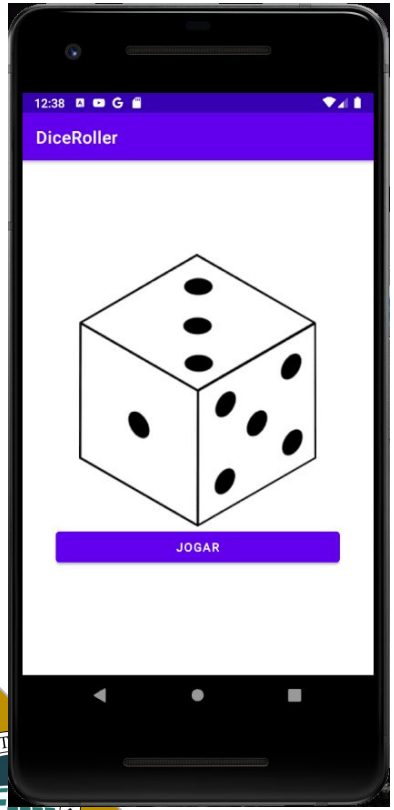
O Android 8.0 (API de nível 26) apresenta ícones de inicialização adaptáveis, que podem exibir uma variedade de formas em diferentes modelos de dispositivos.

Cada dispositivo fornece uma máscara, que o sistema usa para renderizar todos os ícones adaptáveis da mesma forma.

Um ícone de iniciador adaptável também é usado em atalhos, aplicativos de configurações, diálogos de compartilhamento e tela de visão geral.



Dice Roller



MainActivity.kt

```
8   class MainActivity : AppCompatActivity() {
9
10      val dado = arrayOf(R.drawable.dice_1, R.drawable.dice_2, R.drawable.dice_3,
11              R.drawable.dice_4, R.drawable.dice_5, R.drawable.dice_6)
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14
15          super.onCreate(savedInstanceState)
16          setContentView(R.layout.activity_main)
17
18          val btnJogar :Button! = this.findViewById<Button>(R.id.btnJogar)
19          btnJogar.setOnClickListener { it: View!
20
21              val imgDado :ImageView! = this.findViewById<ImageView>(R.id.imgDado)
22              val valor :Int = (0..5).random()
23              imgDado.setImageResource(dado[valor])
24          }
25      }
26  }
```

Exercício

Implementar a aplicação apresentada na aula.