MERGULHANDO NO TEMA...

Para alterar o layout de um componente utilizado na tela do aplicativo, é necessário o desenvolvimento de um arquivo XML auxiliar que é capaz de formatar um componente existente. É por meio dessa união que alcançamos as modificações necessárias para deixar a interface do usuário mais atraente e moderna.

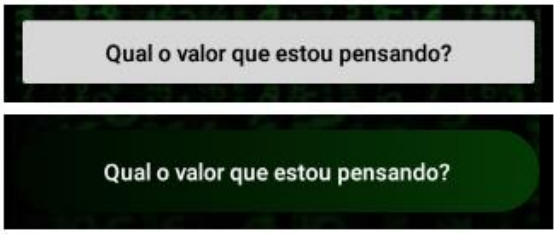


Figura 3- Button convencional e Button customizado na cor verde em degrade e com cantos arredondados.

A figura 3 demonstra um componente "Button" em seu layout convencional e um mesmo componente que sofreu uma customização via código XML.

Desenvolvimento do arquivo XML para customização do Button

Primeiramente, é necessário criar um arquivo na pasta "Drawable". Para isso vamos executar a seguinte sequência no projeto:

- Na estrutura do projeto, clique com o botão direito na pasta **"Drawable", "New"** e depois em **"Drawable Resource File"**, como mostra a figura 4.

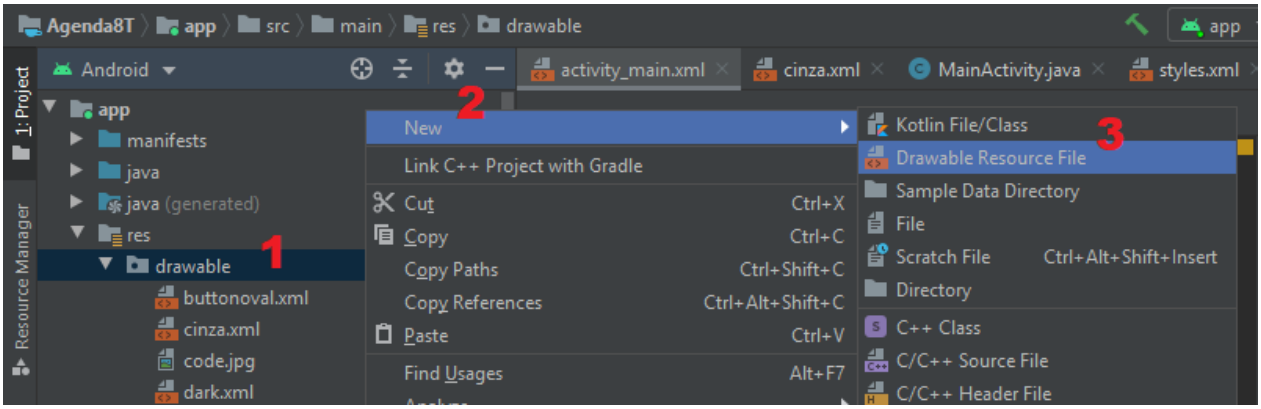


Figura 4- Novo arquivo XML no "Drawable".

Com o assistente aberto, escolha um nome para a sua customização, lembrando que podemos ter mais de uma customização no mesmo projeto e que, também, não é obrigatória a sua utilização nos botões. Podemos encontrar, em uma mesma tela, botões com estilos e layouts diferentes, como também nessa mesma tela podemos encontrar um botão com seu estilo padrão.

Sendo assim, em **"File Name"** associe um nome fácil e que permita lembrar qual é o estilo utilizado na customização, isso facilitará a sua localização e aplicação no componente.

Na opção **"Root Element"** escreva **"shape"** que é o recurso utilizado para a customização e tema do próximo tópico. Na sequência, aplique o botão **"Ok"** para efetivar o desenvolvimento do arquivo. Na figura 5, confira o processo desenvolvido anteriormente.

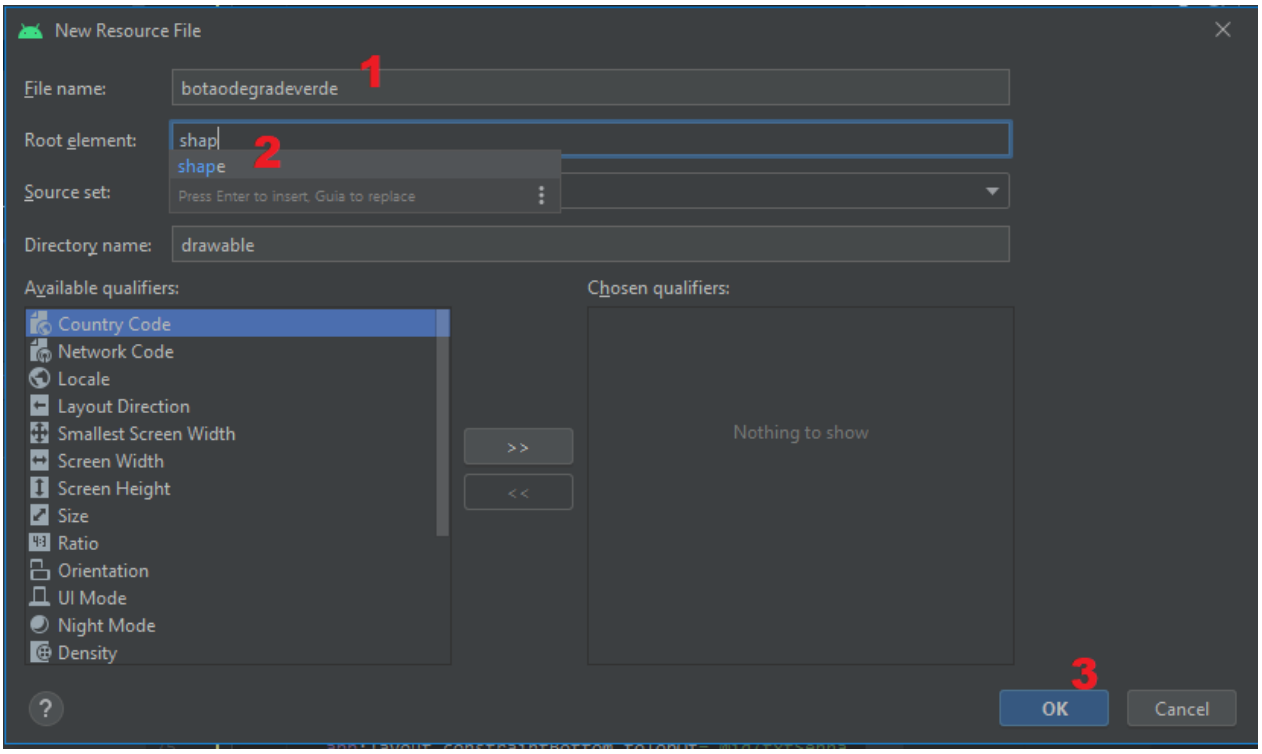


Figura 5- Assistente "New Resourse File".

Shape

No arquivo XML, desenvolvido anteriormente, vamos inserir alguns comandos para criar um formato, cor e tamanho para os componentes já existentes e conhecidos do Android Studio.

Vamos utilizar um recurso denominado "Shape", que é uma subclasse abstrata Drawable. Este recurso pode ser aplicado em locais que esperam a utilização de um Drawable como, por exemplo, em um botão, em uma caixa de texto ou até mesmo em um fundo da tela.

O "Shape" é muito interessante pelo fato de poder ser ampliado ou até mesmo diminuído sem perder a qualidade visual. Outra vantagem é que conseguimos manipular seus atributos mesmo

durante a execução do aplicativo. Podendo assim, mudar a cor de um botão durante uma rotina do usuário.

Para atribuir um formato para o “Shape” são utilizados os seguintes complementos da figura 6.

Valor	Descrição
`"rectangle"`	Um retângulo que preenche a visualização que o contém. É o formato padrão.
`"oval"`	Formato oval que se encaixa nas dimensões da visualização que o contém.
`"line"`	Linha horizontal da largura da visualização que a contém. Esse formato exige o elemento `<stroke>` para definir a largura da linha.
`"ring"`	Forma circular.

Figura 6- Formatos disponíveis para o “Shape. Fonte: https://developer.android.com/guide/topics/resources/drawable-resource?hl=pt-br

Gradient

É aplicado ao “Shape” para atribuir um gradiente de cor. Podemos definir o ângulo do gradiente gerado no “Shape”, como também podemos atribuir a cor inicial, a do meio e a cor final do gradiente.

Para atribuir o ângulo do gradiente, utilizamos o complemento **android:angle=" "** que trabalha com um número inteiro. O ângulo do gradiente é sempre informado em graus e em múltiplos de 45. Por exemplo, o valor 0 é da esquerda para a direita, o valor 90 é debaixo para cima.

O código a seguir gera um “Shape” retangular que conta com um gradiente de cor iniciado na cor preta e que termina na cor verde, esse gradiente inicia-se da esquerda para a direita.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <gradient
      android:angle="0"
      android:startColor="#000700"
      android:endColor="#043904"
      />
</shape>
```

Veja, na figura 7, o resultado desse “Shape” aplicado no “Button”.

Figura 7- Exemplo do resultado de um "Shape" aplicado ao "Button".

Solid

É aplicado ao "Shape" para atribuir uma cor sólida que preencherá toda a extensão do "Shape", sem nenhuma variação de tonalidade.

O código a seguir gera um "Shape" retangular com uma cor fixa na tonalidade verde.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid
      android:color="#043904"
      />
</shape>
```

Veja na figura 8 o resultado desse "Shape" aplicado no "Button".

Figura 8– Exemplo do resultado de um "Shape" aplicado ao "Button".

Corners

Sua aplicação é realiza apenas em um "**Shape**" do tipo "**rectangle**". Esse complemento é capaz de criar cantos arredondados para o "Shape". Tal complemento pode alterar todos os cantos de uma só vez, ou alterar os cantos individualmente. Veja os complementos para o "Corners" na figura 9.

`android:radius`

Dimensão. O raio para todos os cantos, como um valor de dimensão ou recurso de dimensão. É substituído para cada canto pelos atributos a seguir.

`android:topLeftRadius`

Dimensão. O raio para o canto superior esquerdo, como um valor de dimensão ou recurso de dimensão.

`android:topRightRadius`

Dimensão. O raio para o canto superior direito, como um valor de dimensão ou recurso de dimensão.

`android:bottomLeftRadius`

Dimensão. O raio para o canto inferior esquerdo, como um valor de dimensão ou recurso de dimensão.

`android:bottomRightRadius`

Dimensão. O raio para o canto inferior direito, como um valor de dimensão ou recurso de dimensão.

Figura 9- Atributos do "Coners", utilizado para arredondar os cantos de um "Shape". Fonte: https://developer.android.com/guide/topics/resources/drawable-resource?hl=pt-br

O código a seguir gera um "Shape" retangular que conta com uma cor fixa na tonalidade verde. É aplicado, também, nos cantos um arredondamento por meio do "Corners". É importante verificar que atribuímos, propositalmente, um valor para cada canto do "Shape", esse processo foi feito apenas para exemplificar a localização e função de cada complemento da figura 9.

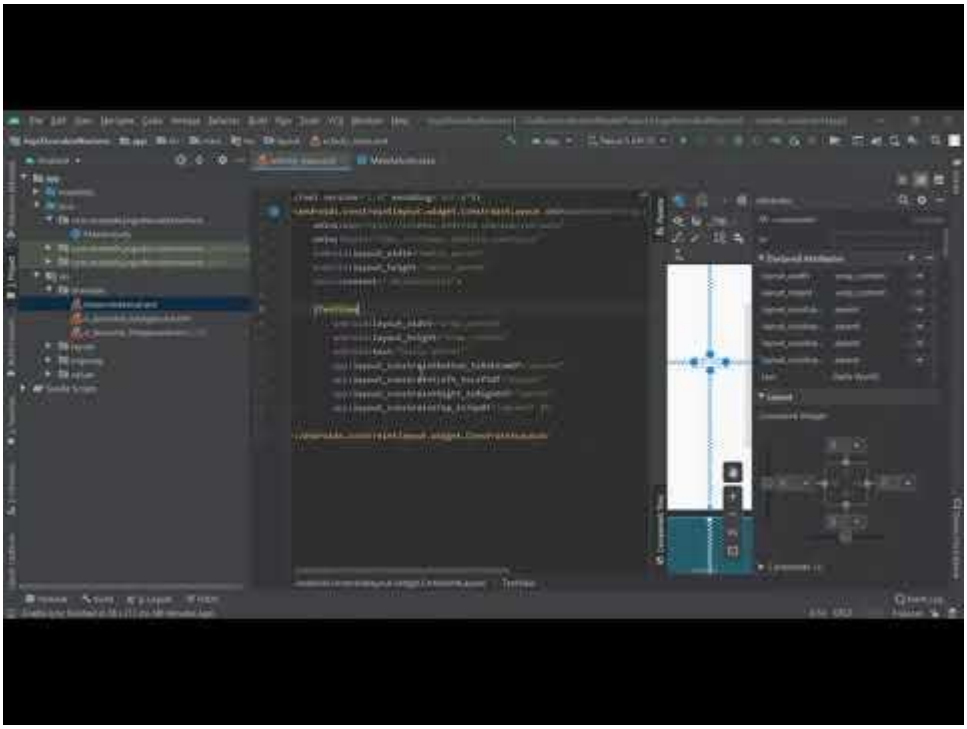
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid
      android:color="#043904"
      />
   <corners
      android:topLeftRadius="1000dp"
      android:topRightRadius="500dp"
      android:bottomLeftRadius="250dp"
      android:bottomRightRadius="100dp"
      />
</shape>
```

Veja na figura 10 o resultado desse “Shape” aplicado no “Button”.

Figura 10- Exemplo do resultado de um “Shape” aplicado ao “Button”.

Vamos desenvolver a tela para o aplicativo da Karla. Para isso, criaremos um projeto no Android Studio com o nome de **“JogoDescubraNumero”** na API de número **25** e com a linguagem de programação **“Java”.**

Assista ao vídeo 1 para gerar o primeiro botão customizado do aplicativo:



Vídeo 1 – Desenvolvimento do primeiro botão. Fonte: https://youtu.be/yrUZJrxvglY

A seguir, veja o código do arquivo XML da pasta **“Drawable”** , desenvolvido no vídeo 1 e com nome de **“botaoverdeoval.xml”.**

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="rectangle">
   <gradient
      android:angle="0"
      android:startColor="#043904"
      android:endColor="#000000"
      />
   <corners
      android:radius="50dp"
      />
</shape>
```

Veja, a seguir, o código do arquivo XML da pasta **“Layout”** alterado no vídeo 1 com nome de **“activity_main.xml”.**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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">

    <Button
        android:id="@+id/btnNovo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/txtDica"
        android:text="Novo"
        android:background="@drawable/botaoverdeoval"
        android:textColor="@android:color/white"
        />

    <TextView
        android:id="@+id/txtDica"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Valor de 1 até 10"
        android:textSize="18dp"
        android:textAlignment="center"
        android:layout_margin="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Texto do botão

Utilize o atributo **“android:textAllCaps="false"”** para o texto do botão não ficar em letras maiúsculas, veja o resultado na figura 11.

Figura 11- Texto do botão antes e depois do atributo android:textAllCaps="false" ser inserido.

Desenvolva mais dois arquivos XML na pasta **"Drawable"**:

- O primeiro com o nome de **"editgradienteverdepretoverde.xml"** que gerará o estilo para o componente **"EditText".** Veja o código a seguir:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="rectangle">
   <gradient
      android:angle="0"
      android:startColor="#043904"
      android:centerColor="#000000"
      android:endColor="#043904"
      />
   <corners
      android:radius="50dp"
      />
</shape>
```

- O seguindo arquivo, com o nome de **"editgradienteverdepretoverdeicone.xml"**, gerará o estilo para outro componente **"EditText"**, responsável por mostrar o valor oculto**.**

É importante ressaltar que foi utilizado o **"Corners"** para arredondar cada um dos cantos do retângulo de maneira individual nas extremidades: **android:bottomRightRadius** e **android:topLeftRadius,** e depois foi utilizado o **android:radius** para alterar os cantos não mencionados anteriormente.

Dessa forma, geramos uma forma diferente utilizando o retângulo, veja o código a seguir:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <corners
```

```
    android:bottomRightRadius="10dp"
    android:topLeftRadius="10dp"
    android:radius="40dp"
    />

  <gradient
    android:angle="0"
    android:startColor="#043904"
    android:centerColor="#000000"
    android:endColor="#043904"
    android:type="linear" />

  <size
    android:width="82dp"
    android:height="82dp"
    />

  <stroke
    android:width="2dp"
    android:color="#000000"
    />
</shape>
```

Você notou dois complementos novos? Vamos verificar a função de cada um deles.

Stroke

É responsável por criar uma linha ou traço no shape desenvolvido. Veja na figura 12 os complementos para o atributo **"Stroke".**

`android:width`

Dimensão. A espessura da linha, como um valor de dimensão ou recurso de dimensão.

`android:color`

Cor. A cor da linha, como um valor hexadecimal ou recurso de cor.

`android:dashGap`

Dimensão. A distância entre os traços da linha, como um valor de dimensão ou recurso de dimensão. Só é válida se `android:dashWidth` está definida.

`android:dashWidth`

Dimensão. O tamanho de cada linha de traço, como um valor de dimensão ou recurso de dimensão. Só é válida se `android:dashGap` está definida.

Figura 12- Atributo "Stroke" e seus complementos. Fonte: https://developer.android.com/guide/topics/resources/drawable-resource?hl=pt

Size

É responsável por criar um tamanho para o shape desenvolvido. Veja na figura 13 os complementos para o atributo **"Stroke".**

`android:height`

Dimensão. A altura do formato, como um valor de dimensão ou recurso de dimensão.

`android:width`

Dimensão. A largura do formato, como um valor de dimensão ou recurso de dimensão.

Figura 13– Atributo "Size" e seus complementos. Fonte: https://developer.android.com/guide/topics/resources/drawable-resource?hl=pt

Customizando o EditText

Vamos desenvolver mais dois “EditText” no projeto e utilizar a **opção android:background=””** para atribuir o arquivo XML que contém o **“Shape”**, utilizado para customizar cada um deles. Altere o código do arquivo **“activity_main.xml”** disponível em **“Res” -> “layout”**, conforme o código a seguir.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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">

   <EditText
      android:id="@+id/edtValorOculto"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="10dp"
      android:background="@drawable/editgradienteverdepretoverdeicone"
      android:enabled="false"
      android:text="?"
      android:textAlignment="center"
      android:textColor="@android:color/white"
      app:layout_constraintBottom_toTopOf="@id/edtValor"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.498"
      app:layout_constraintStart_toStartOf="parent" />

   <EditText
      android:id="@+id/edtValor"
      android:layout_width="match_parent"
      android:layout_height="50dp"
      android:layout_margin="10dp"
      android:background="@drawable/editgradienteverdepretoverde"
      android:hint="Digite o valor!"
      android:inputType="number"
      android:textAlignment="center"
      android:textColor="@android:color/white"
      android:textColorHint="@android:color/white"
      app:layout_constraintBottom_toTopOf="@id/btnEnviar"

      />
```

```
    <Button
        android:id="@+id/btnEnviar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/btnNovo"
        android:text="Enviar"
        android:background="@drawable/botaoverdeoval"
        android:textColor="@android:color/white"
        android:layout_margin="10dp"
        android:textAllCaps="false"
        />

    <Button
        android:id="@+id/btnNovo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/txtDica"
        android:text="Novo"
        android:background="@drawable/botaoverdeoval"
        android:textColor="@android:color/white"
        android:layout_margin="10dp"
        android:textAllCaps="false"
        />

    <TextView
        android:id="@+id/txtDica"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Valor de 1 até 10"
        android:textSize="18dp"
        android:textAlignment="center"
        android:layout_margin="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Removendo a ActionBar e alterando a StatusBar

Vamos remover a **“ActionBar”**, que é a barra que mostra o nome do aplicativo. Para isso, abra o arquivo **“styles.xml”**, disponível em **“Res” -> pasta “values”** e altere o código, conforme o exemplo:

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

Na sequência, vamos inserir o atributo "**<item name="android:statusBarColor">#000000</item>"** para mudar a cor da **"StatusBar"** que é a barra onde fica o relógio do dispositivo. No final o arquivo **"styles.xml"** deve ficar com esse conteúdo:

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:statusBarColor">#ff0000</item>
    </style>
</resources>
```

O resultado esperado para o projeto é como mostra a figura 14, caso o seu projeto esteja diferente, volte nas etapas e confira os códigos utilizados no desenvolvimento.

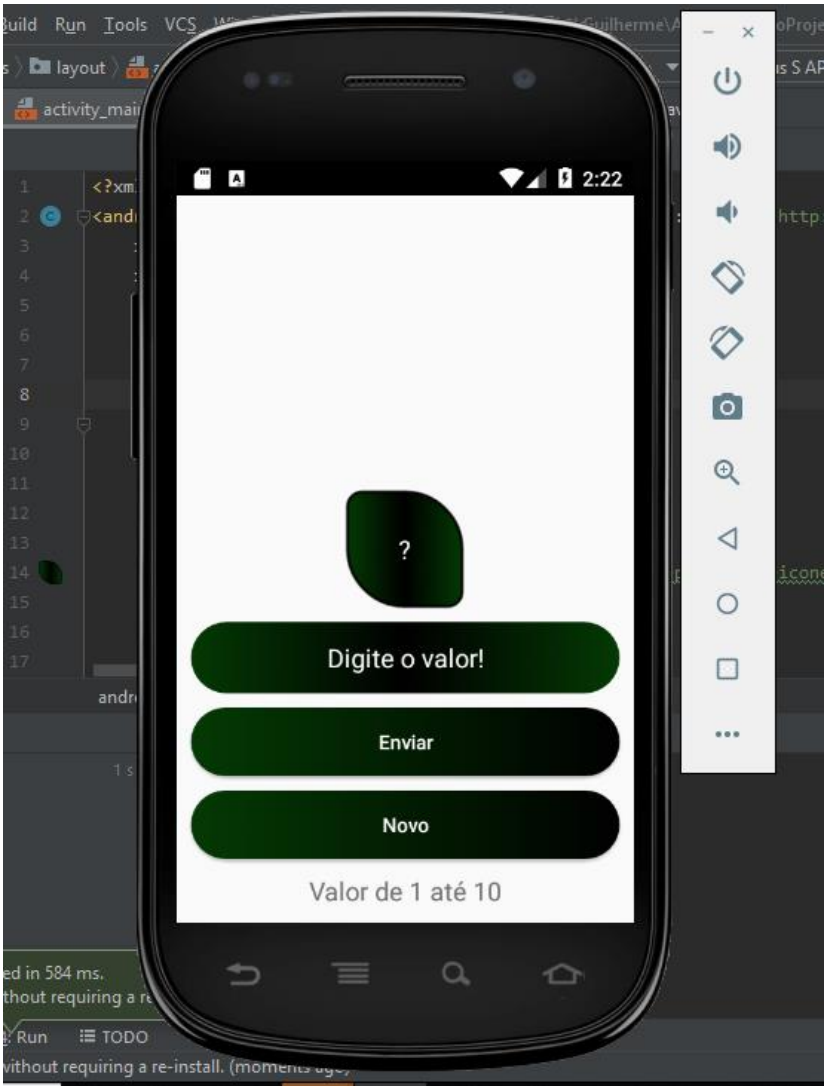
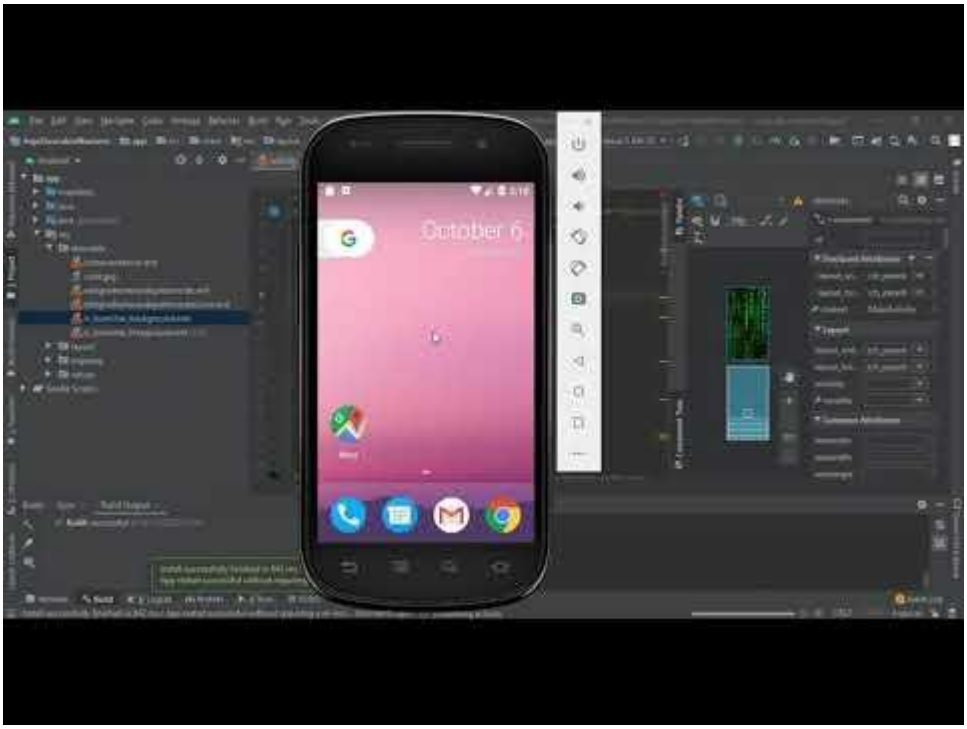


Figura 14-Layout parcial do projeto.

Inserindo a figura no fundo do aplicativo

No vídeo 2, a seguir, acompanhe o desenvolvimento da figura de fundo do aplicativo:



Vídeo 2 – Desenvolvimento da figura de fundo. Fonte: https://youtu.be/adR2lK9LN0w

O código do arquivo XML que contém o **“Shape”** é utilizado para customizar a figura de fundo. Está disponível em **“Res”** na pasta **“Drawable”** com o nome de **“transparentepretocinza.xml”.**

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <gradient
      android:endColor="@android:color/transparent"
      android:startColor="#FF000000"
      android:angle="90"
      android:centerColor="#AA000000"
      />
</shape>
```

Altere o código do arquivo **“activity_main.xml”** disponível em **“Res”** na pasta **“layout”**, conforme o código a seguir.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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">

<androidx.appcompat.widget.AppCompatImageView
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:scaleType="centerCrop"
   android:src="@drawable/code"
   />

<View
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/transparentepretocinza"
   />

<EditText
   android:id="@+id/edtValorOculto"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_margin="10dp"
   android:background="@drawable/editgradienteverdepretoverdeicone"
   android:enabled="false"
   android:text="?"
   android:textAlignment="center"
   android:textColor="@android:color/white"
   app:layout_constraintBottom_toTopOf="@id/edtValor"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintHorizontal_bias="0.498"
   app:layout_constraintStart_toStartOf="parent" />

<EditText
   android:id="@+id/edtValor"
   android:layout_width="match_parent"
   android:layout_height="50dp"
   android:layout_margin="10dp"
   android:background="@drawable/editgradienteverdepretoverde"
   android:hint="Digite o valor!"
   android:inputType="number"
   android:textAlignment="center"
   android:textColor="@android:color/white"
   android:textColorHint="@android:color/white"
   app:layout_constraintBottom_toTopOf="@id/btnEnviar"
```

```
    />

  <Button
    android:id="@+id/btnEnviar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toTopOf="@+id/btnNovo"
    android:text="Enviar"
    android:background="@drawable/botaoverdeoval"
    android:textColor="@android:color/white"
    android:layout_margin="10dp"
    android:textAllCaps="false"
    />

  <Button
    android:id="@+id/btnNovo"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toTopOf="@+id/txtDica"
    android:text="Novo"
    android:background="@drawable/botaoverdeoval"
    android:textColor="@android:color/white"
    android:layout_margin="10dp"
    android:textAllCaps="false"
    />

  <TextView
    android:id="@+id/txtDica"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Valor de 1 até 10"
    android:textSize="18dp"
    android:textColor="@android:color/white"
    android:textAlignment="center"
    android:layout_margin="10dp"
    app:layout_constraintBottom_toBottomOf="parent"
    />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Chegamos ao fim do desenvolvimento da tela do aplicativo **“JogoDescubraNumero”** que Karla terá que construir para seu irmão. Guarde esse projeto! Ele será utilizado nas próximas atividades.