



Layout, Views e Recursos



Layout, Views e Recursos

Vincular uma Activity a um recurso de layout é um exemplo de parte do padrão de arquitetura model-view-presenter (MVP). O padrão MVP é uma forma bem estabelecida de agrupar funções de aplicativos:



Views - são elementos da interface do usuário que exibem dados e respondem às ações do usuário. Cada elemento da tela é uma vista. O sistema Android oferece muitos tipos diferentes de views.



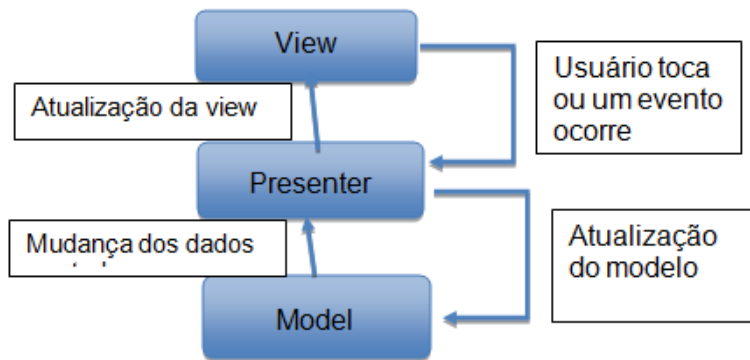
Presenters - conectar as views do aplicativo ao modelo. Eles fornecem para as visualizações com dados conforme especificado pelo modelo e também fornecem ao modelo a entrada do usuário a partir da visualização.



Model - especifica a estrutura dos dados do aplicativo e o código para acessar e manipular os dados.

O padrão MVP pode se representado por:





Views

A UI (User Interface) consiste em uma hierarquia de objetos chamados visualizações - cada elemento da tela é uma visualização. A classe View representa o bloco de construção básico para todos os componentes de IU e a classe base para classes que fornecem componentes de UI interativos, como como botões, caixas de seleção e campos de entrada de texto.

Uma vista tem uma localização, expressa como um par de coordenadas esquerda e superior, e duas dimensões, expressas como largura e altura. A unidade de localização e dimensões é o pixel independente de dispositivo (dp).

O sistema Android fornece centenas de visualizações predefinidas, incluindo aquelas que exibem:

-

Texto (TextView)

-

Campos para inserir e editar texto (EditText)



- Botões que os usuários podem tocar (botão) e outros componentes interativos
- Texto rolável (ScrollView) e itens roláveis (RecyclerView)
- Imagens (ImageView)

Você pode definir uma visualização para aparecer na tela e responder a um toque do usuário. Uma visualização também pode ser definida para aceitar a entrada de texto ou ficar invisível até que seja necessária. Você pode especificar as visualizações em arquivos de recursos de layout XML. Os recursos de layout são escritos em XML e listados dentro da pasta de layout na pasta res na visualização Projeto: Android.

Views groups

As views podem ser agrupadas dentro de um grupo de visualizações (ViewGroup), que atua como um contêiner de visualizações. O relacionamento é pai-filho, no qual o pai é um grupo de visualização e o filho é uma visualização ou grupo de visualização dentro do grupo. Os seguintes são grupos de visão comuns:

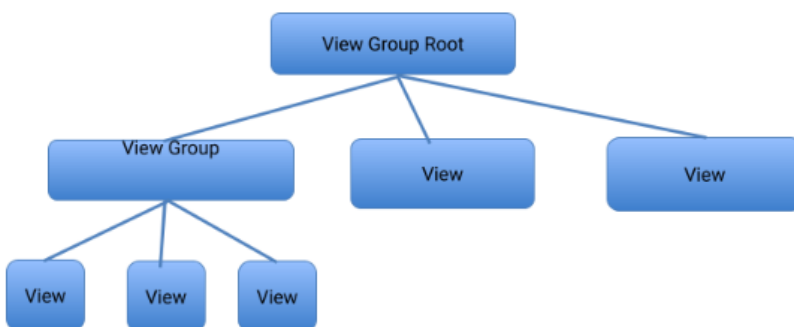
- ScrollView: um grupo que contém uma outra visualização filha e permite rolar a visualização filha.



RecyclerView: um grupo que contém uma lista de outras visualizações ou grupos de visualização e permite rolá-los adicionando e removendo visualizações dinamicamente da tela.

Layout view groups

As visualizações de uma tela são organizadas em uma hierarquia. Na raiz dessa hierarquia está um ViewGroup que contém o layout de toda a tela. As telas filho do grupo de visualizações podem ser outras visualizações ou outros grupos de visualizações, conforme mostrado na figura a seguir.




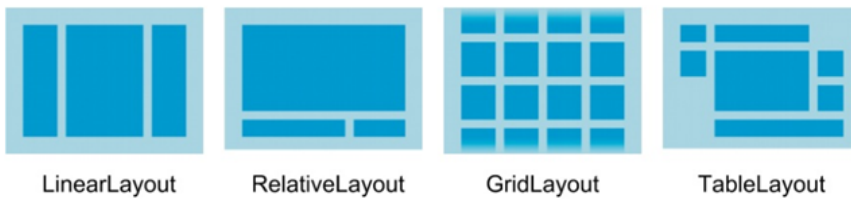
O grupo de visualização raiz.

O primeiro conjunto de visualizações filhas e grupos de visualizações cujo pai é a raiz.

Alguns grupos de visualização são designados como layouts porque organizam visualizações filhas de uma maneira específica e são normalmente usados como o grupo de visualização raiz. Alguns exemplos de layouts são:

- LinearLayout: Um grupo de visualizações secundárias posicionadas e alinhadas horizontalmente ou verticalmente.
- RelativeLayout: um grupo de visualizações filhas em que cada visualização é posicionada e alinhada em relação a outras visualizações dentro do mesmo grupo. Em outras palavras, as posições das visualizações filhas podem ser descritas em relação umas às outras ou aos pais do grupo.
- ConstraintLayout: um grupo de visualizações filhas usando pontos de ancoragem, arestas e diretrizes para controlar como as visualizações são posicionadas em relação a outros elementos no layout. ConstraintLayout foi projetado para tornar mais fácil arrastar e soltar visualizações no editor de layout.
- TableLayout: Um grupo de visualizações filhas organizadas em linhas e colunas. AbsoluteLayout: Um grupo que permite especificar localizações exatas (coordenadas x / y) de suas visualizações filhas. Layouts absolutos são menos flexíveis e mais difíceis de manter do que outros tipos de layouts sem posicionamento absoluto.
- FrameLayout: um grupo de visualizações filhas em uma pilha. FrameLayout é projetado para bloquear uma área da tela para exibição de uma única visão. As visualizações filho são desenhadas em uma pilha, com o filho adicionado mais recentemente no topo. O tamanho do FrameLayout é o tamanho de sua maior visualização secundária.

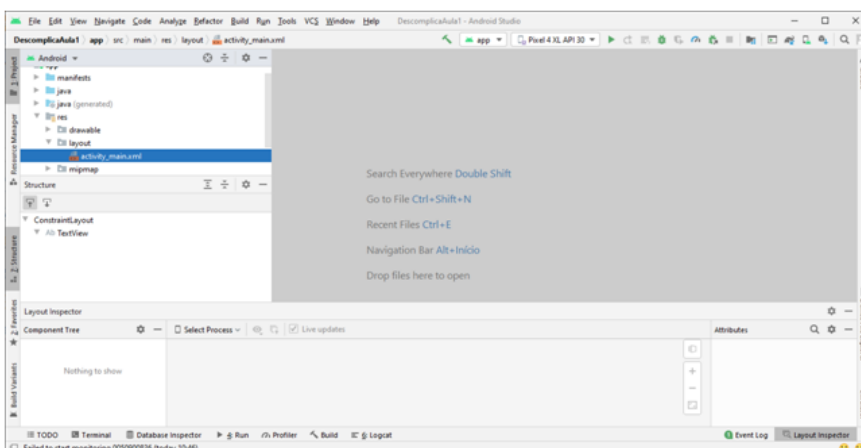
GridLayout: um grupo que coloca suas telas filho em um  de retangular que pode ser rolada.

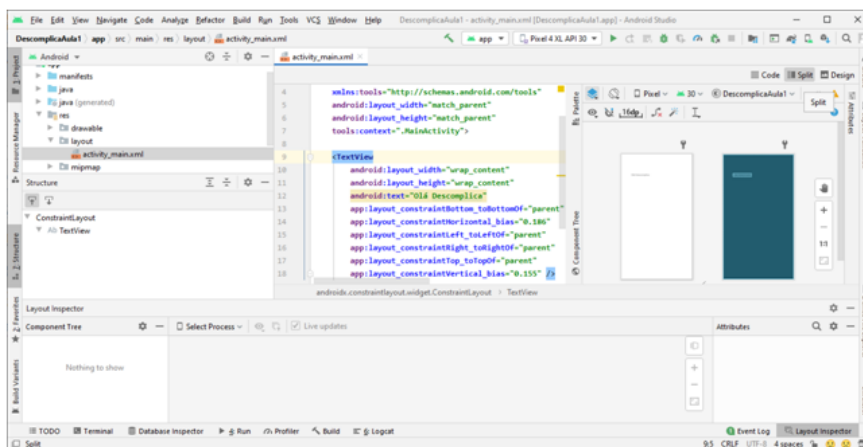
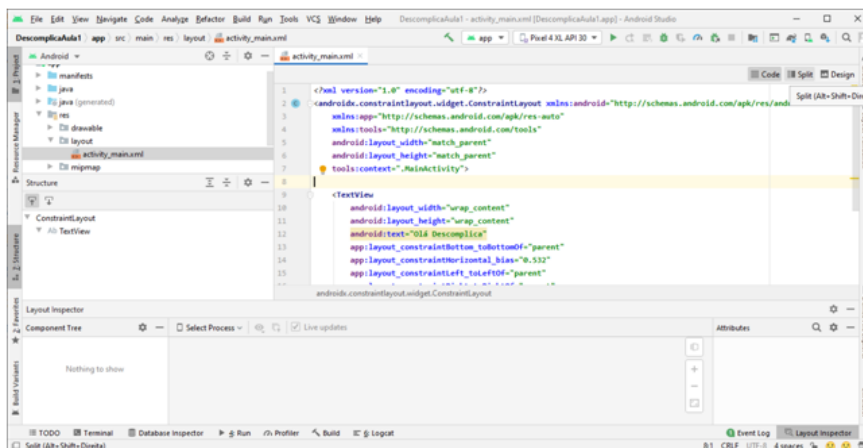
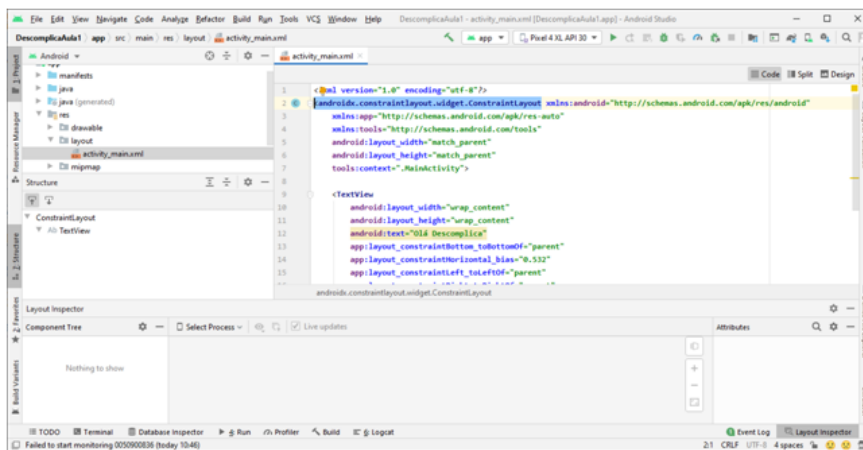


Usando o Editor de Layout

Use o editor de layout para editar o arquivo de layout. Você pode arrastar e soltar objetos de visualização em um painel gráfico e organizar, redimensionar e especificar propriedades para eles. Você vê imediatamente o efeito das alterações feitas.


Para usar o editor de layout, abra o arquivo de layout XML. O editor de layout aparece com a guia Design na parte inferior destacada. (Se a guia Texto estiver destacada e você vir o código XML, clique na guia Design.) Para o modelo de Atividade Vazia, o layout é como mostrado na figura abaixo.

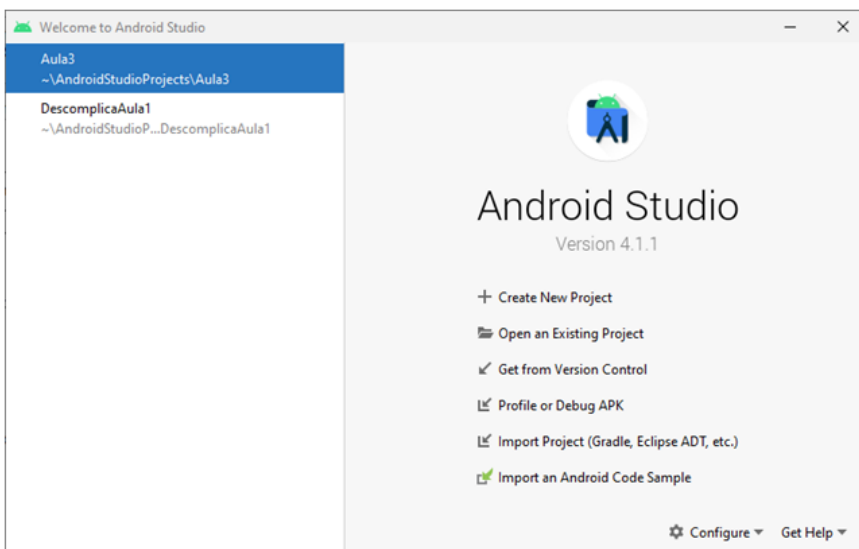
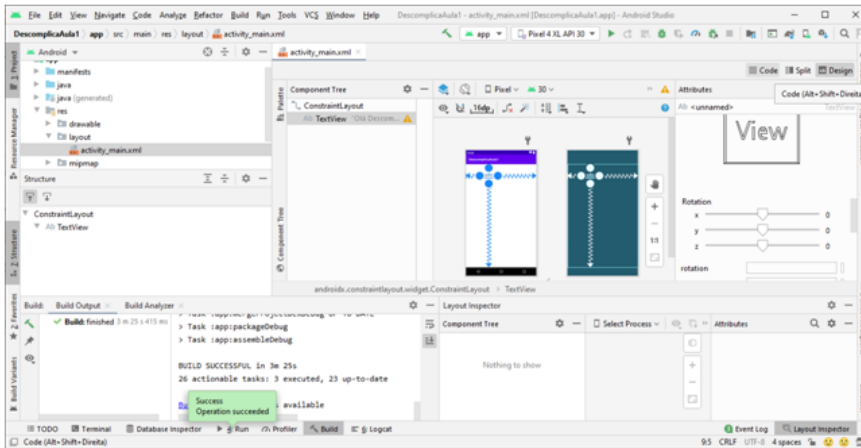




Usando XML

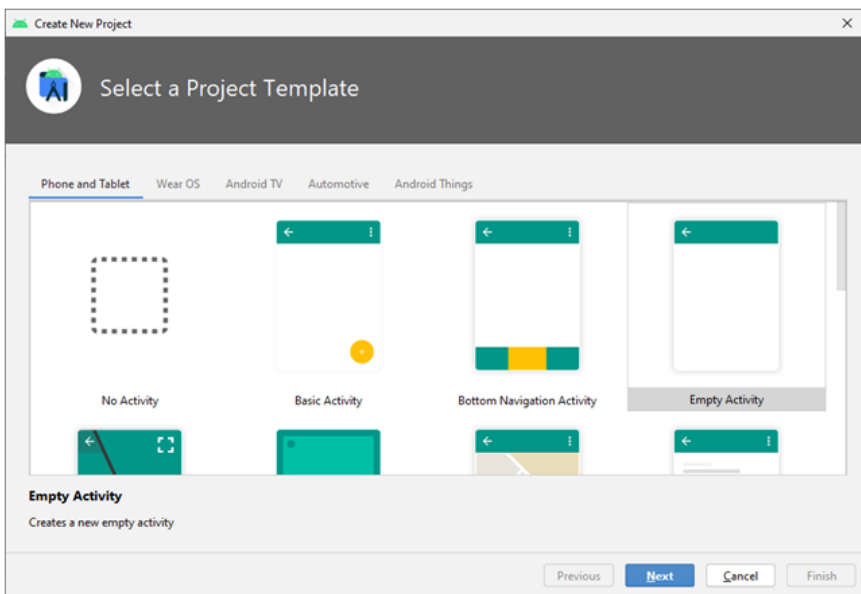
Às vezes, é mais rápido e fácil editar o código XML diretamente, especialmente ao copiar e colar o código para semelhantes Visualizações

Para visualizar e editar o código XML, abra o arquivo de layout XML. O editor de layout aparece com a guia Design na parte inferior em . Clique na guia Texto para ver o código XML. O seguinte mostra um snippet de código XML de um LinearLayout com um Button e um TextView:

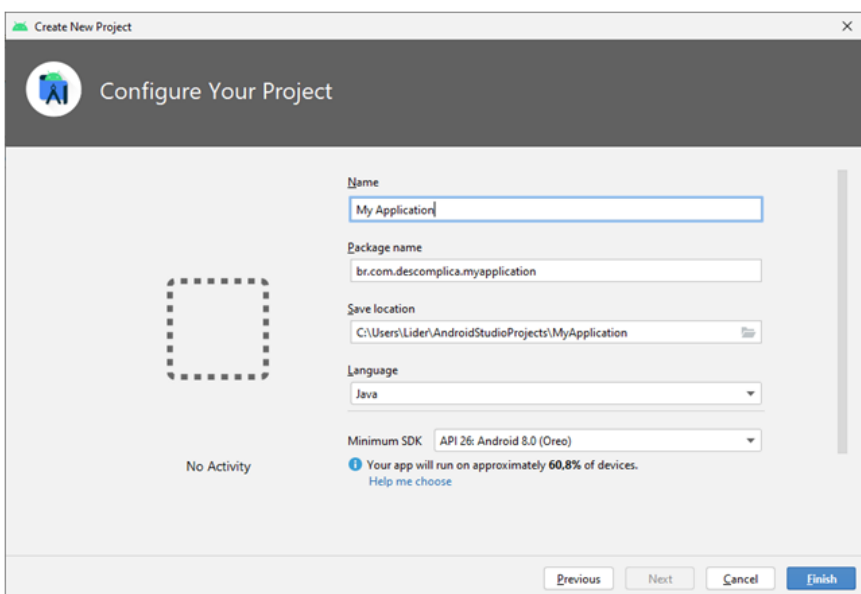


Clique em "Create New Project"



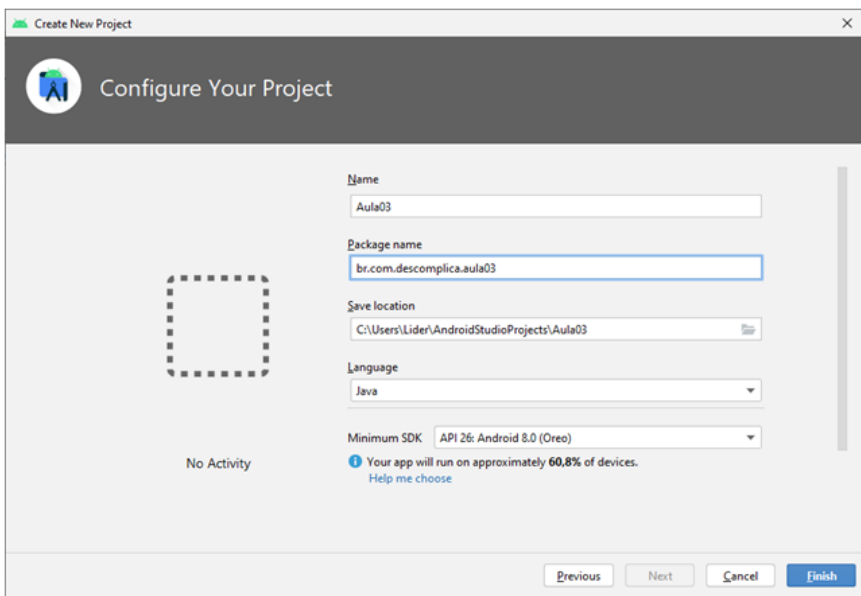


Clique no "Activity"

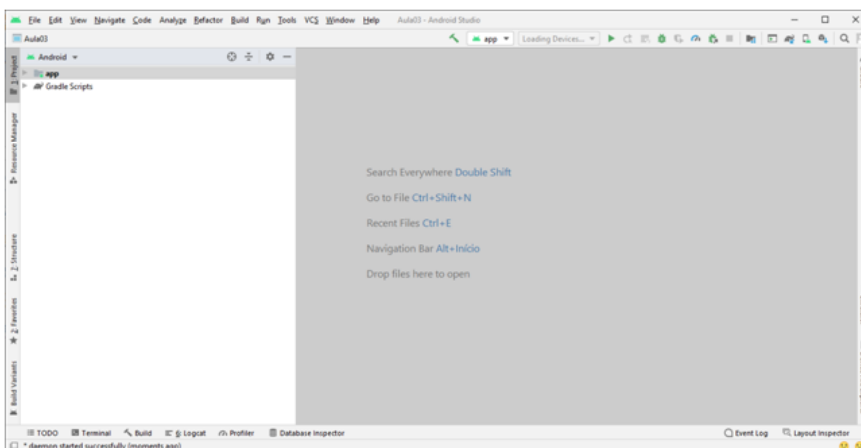


Nome "Aula03"

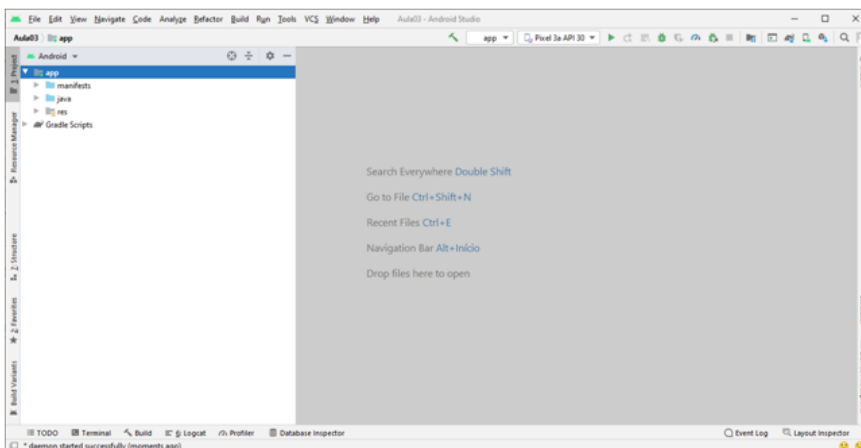




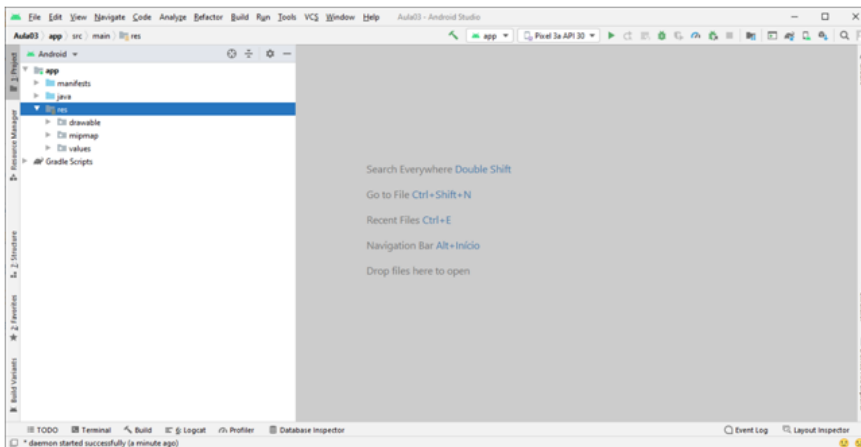
Clique em "Finish"



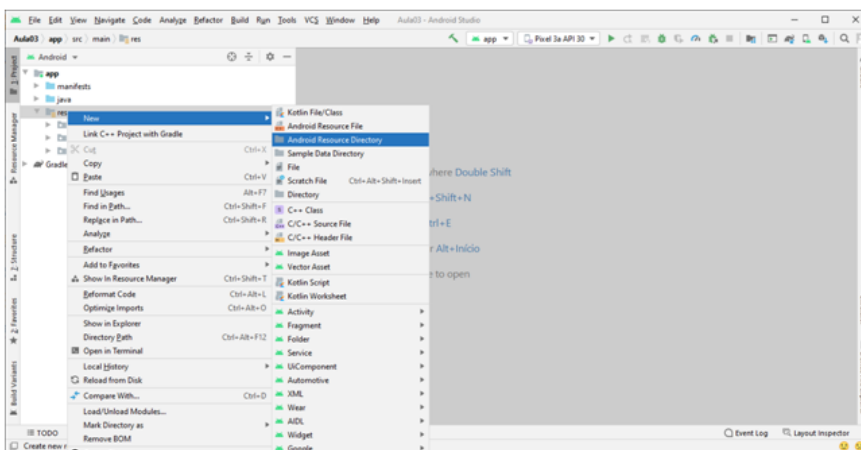
Selecione "app"



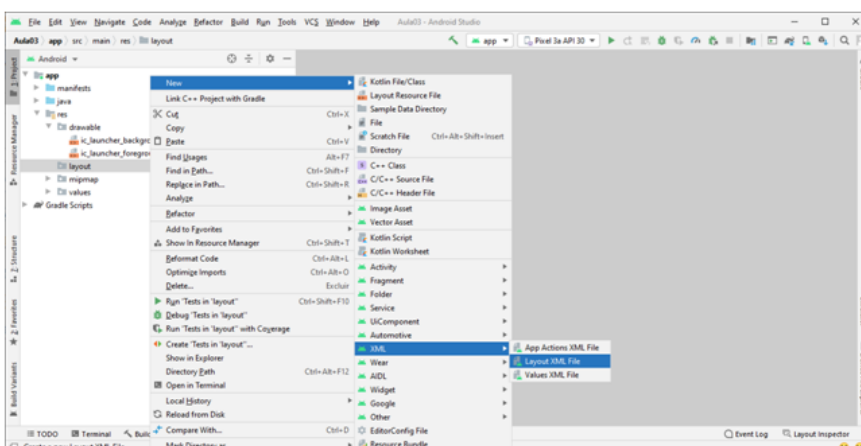
Selecione "res"

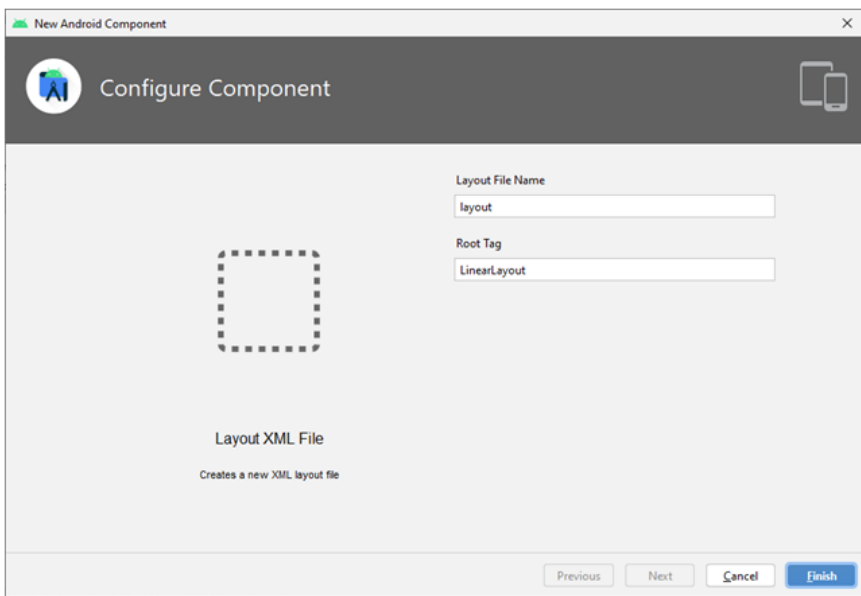


Clique com o botão direito

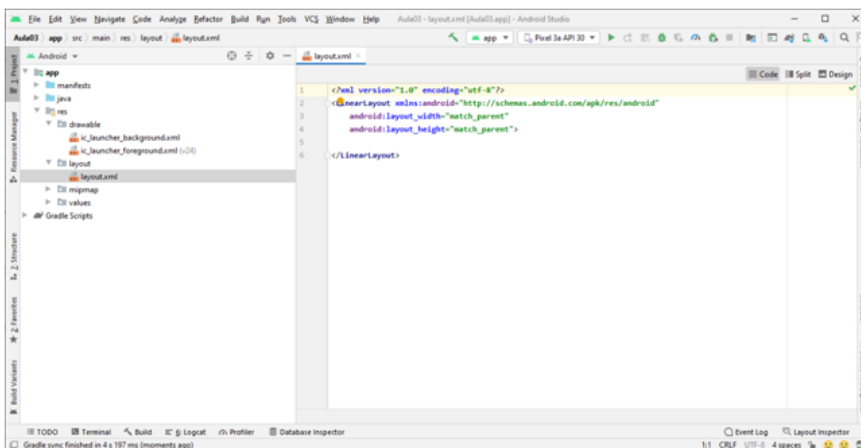
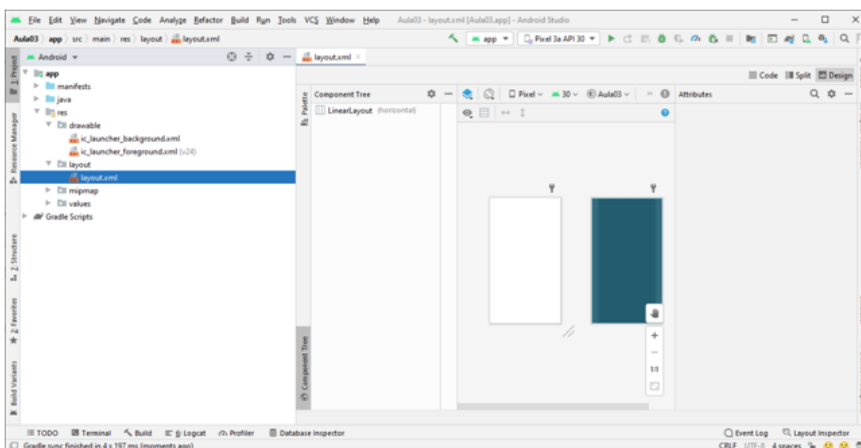


Selecione "layout"





LinearLayout



Atributos XML



As visualizações têm propriedades que definem onde uma visualização aparece na tela, seu tamanho, como a visualização se relaciona com outras visualizações e como ele responde à entrada do usuário. Ao definir visualizações em XML, as propriedades são chamadas de atributos.

Por exemplo, na seguinte descrição XML de um TextView, o android: id, android: layout_width, android: layout_height, android: background, são atributos XML que são traduzidos automaticamente para o TextView propriedades:


Cada visão e grupo de visão suporta sua própria variedade de atributos XML. Alguns atributos são específicos para uma vista (por exemplo, TextView suporta o atributo textSize), mas esses atributos também são herdados por quaisquer visualizações que possam estender o Classe TextView. Alguns são comuns a todas as visualizações, porque são herdados da classe raiz View (como o android: id atributo). Para obter descrições de atributos específicos, consulte a seção de visão geral da documentação da classe View.

Identificar uma View

Para identificar com exclusividade uma visualização e referenciá-la em seu código, você deve fornecer a ela um id. O atributo android: id permite que você especificar um id único - um identificador de recurso para uma visão.

```
android:id="@+id/botao_contador"
```



A parte "@ + id / button_count" do atributo acima cria um novo id chamado botao_contador para a visualização. Você usa o símbolo + para incluir  que você está criando um novo id.

Referenciando uma View

Para se referir a um identificador de recurso existente, omita o símbolo de mais (+). Por exemplo, para se referir a uma visão por seu id em outro atributo, como android:layout_toLeftOf para controlar a posição de uma visualização, você usaria:

```
android:layout_toLeftOf="@id/show_count"
```

Posicionando Views

Alguns atributos de posicionamento relacionados ao layout são necessários para uma visualização e aparecem automaticamente quando você adiciona a visualização ao Layout XML, pronto para você adicionar valores.

Posicionamento no LinearLayout

Por exemplo, LinearLayout deve ter estes atributos definidos:

- android:layout_width
- android:layout_height
- android:orientation



Os atributos android: layout_width e android: layout_height podem assumir um de três valores:



- match_parent - expande a visualização para preencher seu pai por largura ou altura. Quando o LinearLayout é a visualização raiz, ele se expande para o tamanho da tela do dispositivo. Para uma visão dentro de um grupo de visão raiz, ele se expande para o tamanho do grupo de visão pai.
- wrap_content - reduz as dimensões da vista apenas o suficiente para incluir seu conteúdo. (Se não houver conteúdo, a vista torna-se invisível.)
- Use um número fixo de dp (pixels independentes do dispositivo) para especificar um tamanho fixo, ajustado para o tamanho da tela do dispositivo. Por exemplo, 16 dp significa 16 pixels independentes de dispositivo

A orientação pode ser:

- horizontal: as visualizações são organizadas da esquerda para a direita.
- vertical: as visualizações são organizadas de cima para baixo.

Outros atributos que pode ser utilizados são:



Android: layout_gravity: este atributo é usado com uma visualização para controlar onde a visualização é organizada em seu grupo de visualizações pai. Por exemplo, o seguinte atributo centraliza a visualização horizontalmente na tela:



`android:layout_gravity="center_horizontal"`



- Padding é o espaço, medido em pixels independentes de dispositivo, entre as bordas da visualização e o conteúdo da visualização.
- `android:padding` – é aplicado o distanciamento de todas as bordas
- `android:paddingTop` – é aplicado o distanciamento no topo da borda
- `android:paddingBottom` – é aplicado o distanciamento na borda inferior
- `android:paddingLeft` – é aplicado o distanciamento na borda esquerda
- `android:paddingRight` – é aplicado o distanciamento na borda direita
- `android:paddingStart` – é aplicado o distanciamento no início da visualização
- `android:paddingEnd` – é aplicado o distanciamento no fim da visualização.

Posicionamento com `RelativeLayout`

Outro grupo de visualização útil para layout é `RelativeLayout`, que você pode usar para posicionar visualizações filhas em relação umas às outras ou o pai. Os atributos que você pode usar com `RelativeLayout` incluem o seguinte:

- `android:layout_toLeftOf`:



- android:layout_toRightOf: 
- android:layout_centerHorizontal:
- android:layout_centerVertical:
- android:layout_alignParentTop:
- android:layout_alignParentBottom:

Atributos relacionados ao estilo

Você especifica atributos de estilo para personalizar a aparência da visualização. Vistas que não têm atributos de estilo, como android:textColor, android:textSize e android:background assumem os estilos definidos no tema do aplicativo. A seguir estão os atributos relacionados ao estilo usados no exemplo de layout XML da seção anterior:

- Android:background
- android:text
- android:textColor



- android:textSize 
- android:textStyle

Arquivos de recursos

Os arquivos de recursos são uma forma de separar os valores estáticos do código para que você não precise alterar o próprio código para alterar os valores. Você pode armazenar todas as strings, layouts, dimensões, cores, estilos e texto de menu separadamente em arquivos de recursos. Os arquivos de recursos são armazenados em pastas localizadas na pasta res, incluindo

- drawable: Para imagens e ícones
- layout: Para arquivos de layout
- menu: Para itens de menu
- mipmap: Para coleções pré-calculadas e otimizadas de ícones de aplicativos usados pelo Launcher
- values: Para cores, dimensões, strings e estilos





A sintaxe para fazer referência a um recurso em um layout XML é a seguinte:

`@package_name:resource_type/resource_name`

-

O `package_name` é o nome do pacote no qual o recurso está localizado. Isso não é necessário ao fazer referência a recursos do mesmo pacote - ou seja, armazenados na pasta `res` do seu projeto

-

`resource_type` é a subclasse `R` para o tipo de recurso

-

`resource_name` é o nome do arquivo do recurso sem a extensão ou o valor do atributo `android:name` no XML elemento.

Por exemplo, a seguinte instrução de layout XML define o atributo `android:text` como um recurso de string:

`android:text="@string/Botão"`





Arquivos de recursos de valores

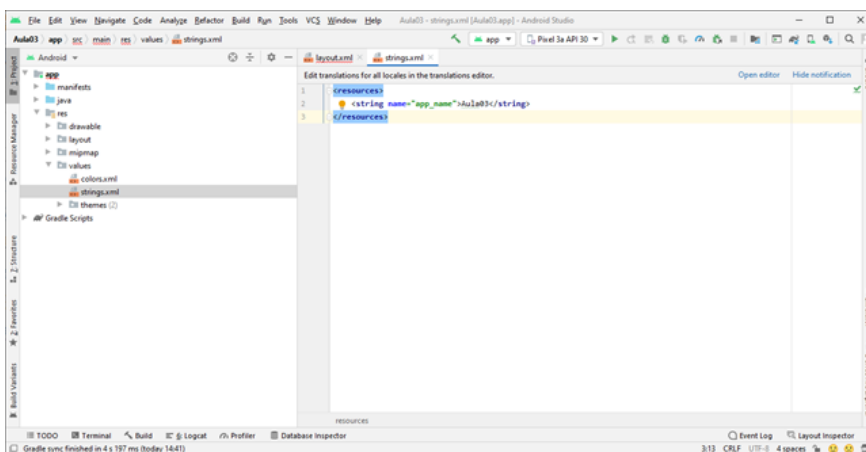
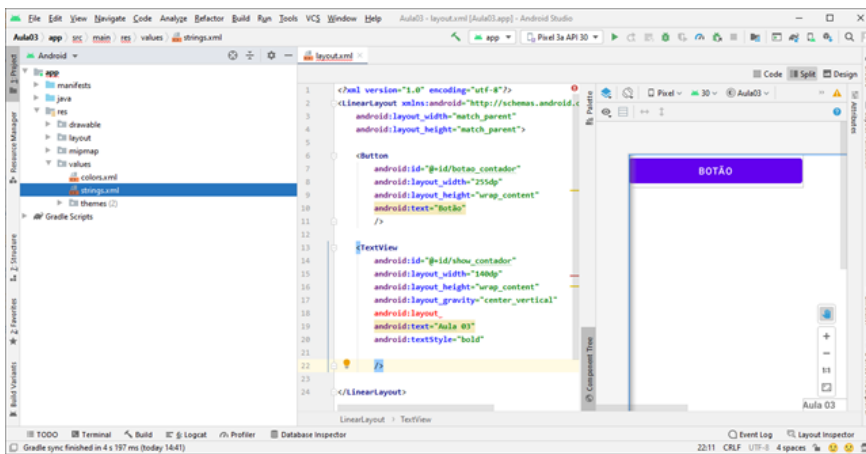
Manter valores como strings e cores em arquivos de recursos separados torna mais fácil gerenciá-los, especialmente se você usar mais de uma vez em seus layouts.

Por exemplo, é essencial manter strings em um arquivo de recurso separado para traduzir e localizar seu aplicativo, para que você possa criar um arquivo de recurso de string para cada idioma sem alterar seu código. Arquivos de recursos para imagens, cores, dimensões, e outros atributos são úteis para desenvolver um aplicativo para diferentes tamanhos e orientações de tela de dispositivo.

Strings

Os recursos de string estão localizados no arquivo `strings.xml` na pasta `values` dentro da pasta `res` ao usar o Projeto: Visualização do Android. Você pode editar este arquivo diretamente abrindo-o:

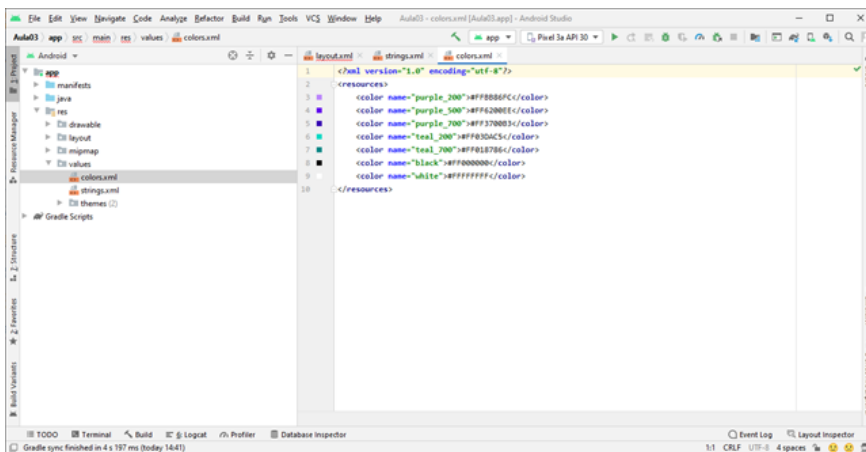




Cores

Os recursos de cores estão localizados no arquivo colors.xml na pasta de valores dentro da pasta res ao usar o Projeto: Android Visão. Você pode editar este arquivo diretamente.





Dimensões

As dimensões devem ser separadas do código para torná-las mais fáceis de gerenciar, especialmente se você precisar ajustar seu layout para diferentes resoluções de dispositivo. Também torna fácil ter dimensionamento consistente para visualizações e alterar o tamanho de múltiplas visualizações alterando um recurso de dimensão.


Os recursos de dimensão estão localizados em um arquivo `dimens.xml` na pasta de valores dentro da pasta `res` ao usar o Projeto: Visualização do Android. O `dimens.xml` mostrado na vista pode ser uma pasta contendo mais de um arquivo `dimens.xml` para diferentes resoluções do dispositivo. Por exemplo, o aplicativo criado a partir do modelo de atividade vazia fornece um segundo arquivo `dimens.xml` para 820 dp. Você pode editar este arquivo diretamente abrindo-o:

Estilos

Um estilo é um recurso que especifica atributos comuns, como altura, preenchimento, cor da fonte, tamanho da fonte e cor de fundo.

Os estilos destinam-se a atributos que modificam a aparência visualização.



Os estilos são definidos no arquivo `styles.xml` na pasta de valores dentro da pasta `res` ao usar a visualização Projeto: Android. Você pode  ditar este arquivo diretamente. Os estilos são abordados em um capítulo posterior, junto com a Especificação de design de material

Responder para visualizar cliques

Um evento de clique ocorre quando o usuário toca ou clica em uma visualização clicável, como `Button`, `ImageButton`, `ImageView` (tocando ou clicando na imagem) ou `FloatingActionButton`.

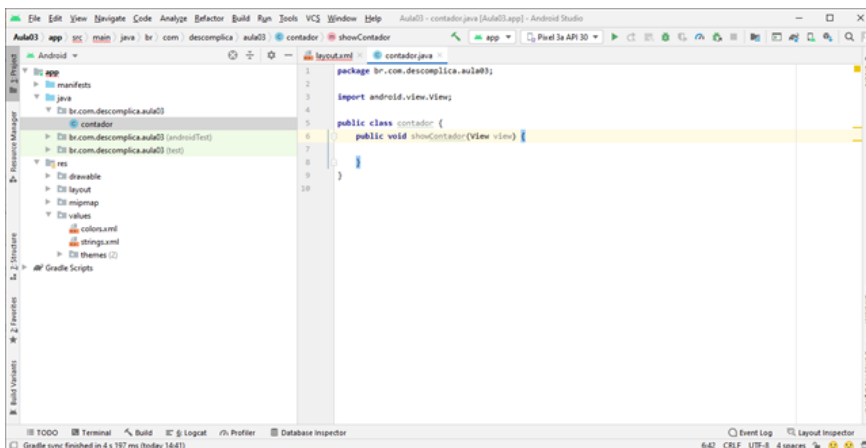
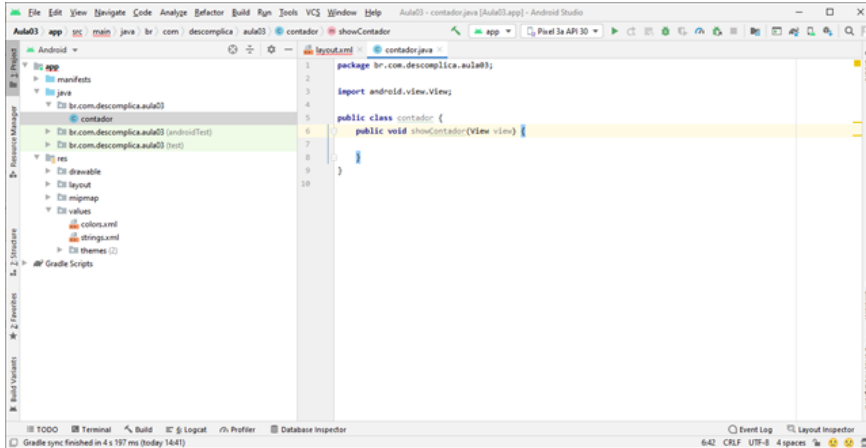
O padrão `model-view-presenter` é útil para entender como responder aos cliques de visualização. Quando um evento ocorre com uma visualização, o código do apresentador executa uma ação que afeta o código do modelo. Para fazer esse padrão funcionar, você deve:

- Escreva um método Java que execute a ação específica, que é determinada pela lógica do código do modelo - ou seja, a ação depende do que você deseja que o aplicativo faça quando esse evento ocorrer. Isso normalmente é conhecido como um manipulador de eventos.
- Associe este método manipulador de eventos à visualização, para que o método seja executado quando o evento ocorrer

Atributo `onClick`

O Android Studio fornece um atalho para configurar uma visualização clicável e para associar um manipulador de eventos à visualização: use o

atributo android: onClick com o elemento da visualização clicável no layout XML.



Para trabalhar com o atributo android: onClick, o método showToast () deve ser público, retornar void e exigir um parâmetro de visualização para saber qual visualização chamou o método.

Atualizando Views

Para atualizar o conteúdo de uma visualização, como substituir o texto em uma TextView, seu código deve primeiro instanciar um objeto

visualização. Seu código pode então atualizar o objeto, atualizando assim a visualização. Para referir-se à visualização em seu código, use `findViewById()` da classe `View`, que procura uma visualização com base na `id` do recurso. Por exemplo, a instrução a seguir define `mShowCount` como `TextView` com o `ID` do recurso `show_count`:

Atividade extra

Explique os diferentes tipos de `Layouts` disponíveis.

Referências bibliográficas

ABLESON, F.; SEN, R. *Android in action*. 2 ed. Manning Publications, 2011.

JOHNSON; T. M. *Java para dispositivos móveis*. São Paulo: Novatec, 2007.

LEE, V.; SCHNEIDER, H.; SCHEL, R. *Aplicações móveis*. São Paulo: Pearson, 2005.

Atividade Prática

Insira o código a seguir no `activity_main.xml`, execute o emulador e apresente o resultado



```
<?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:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

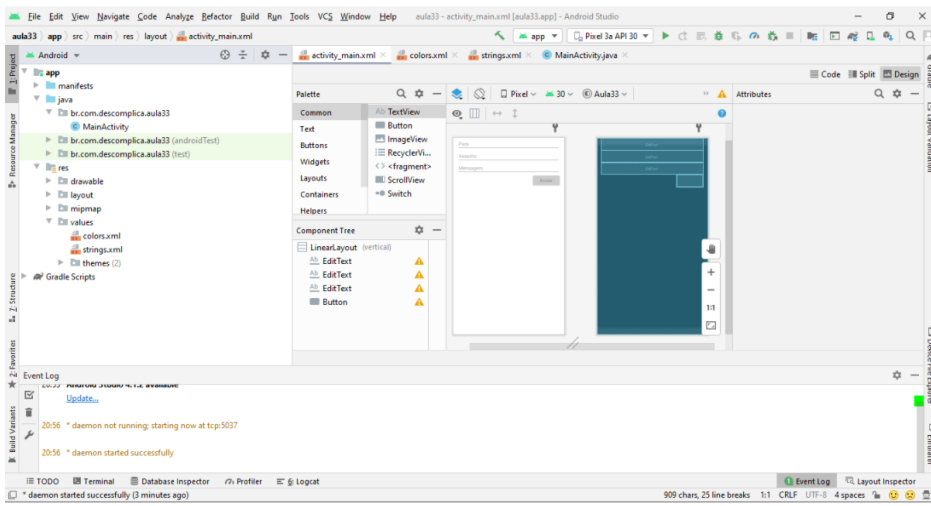
GABARITO

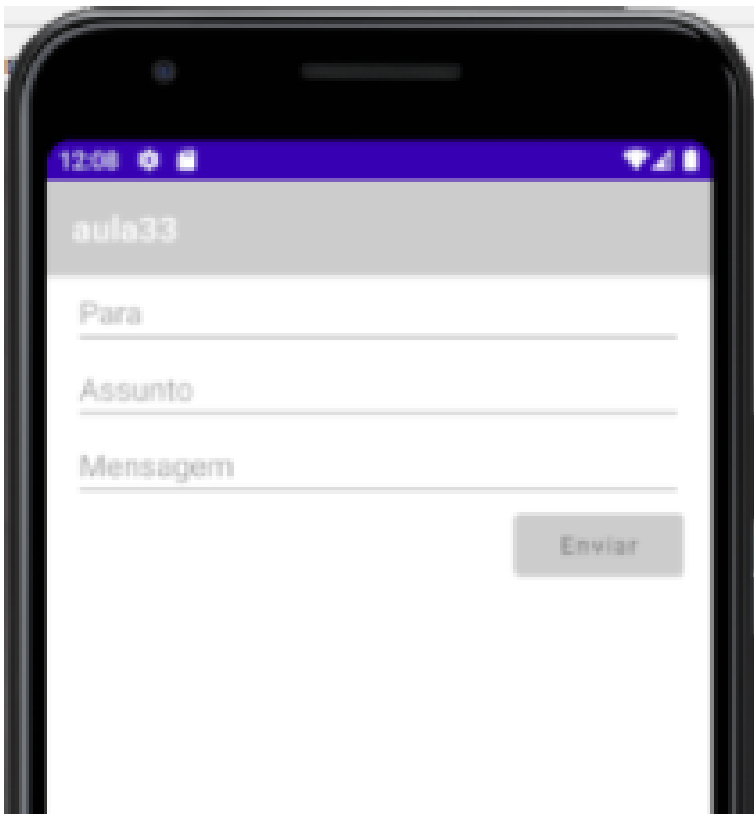
Activity_main.xml

```
<?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:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/assunto"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/mensagem"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:hint="@string/send"/>
</LinearLayout>
```

Modo Design







[Ir para questão](#)

