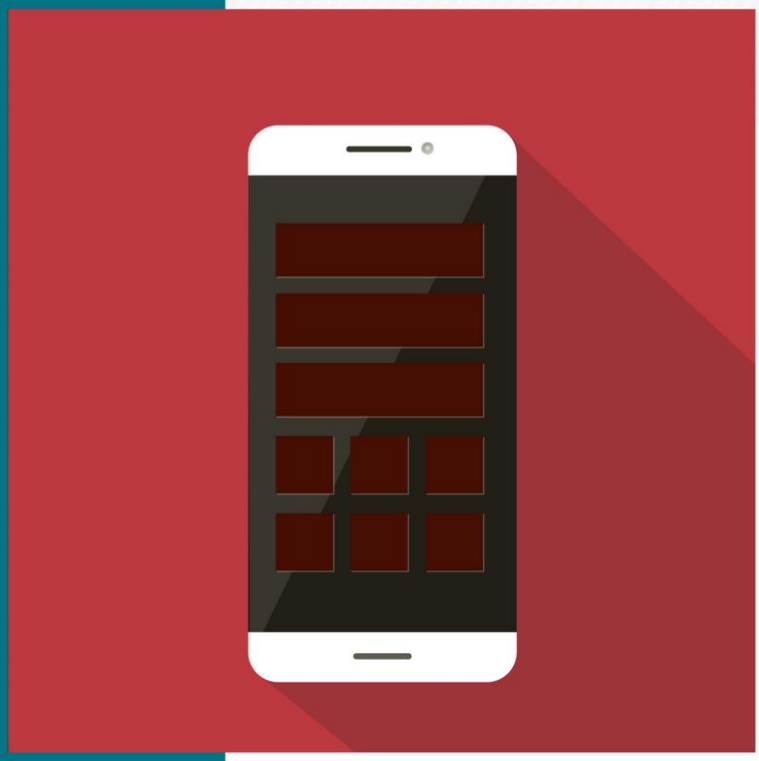
AGENDA 5

TRABALHANDO COM RELATIVELAYOUT NO DESENVOLVIMENTO DE ACTIVITYS



MERGULHANDO NO TEMA...

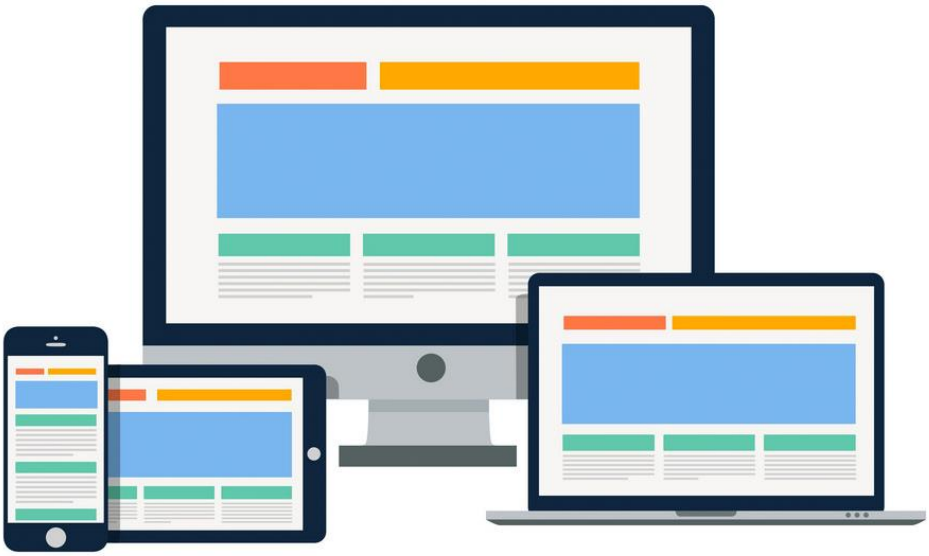


Imagem 3 – freepik.com

Segundo o site de documentação do Android Studio, o RelativeLayout é um agrupamento que exibe componentes filhos em posições relativas. A posição de cada componente "filho" pode ser especificada como relativa a um elemento "irmão" ou vizinho, por exemplo, o TextView está à esquerda ou abaixo de um EditText.

O componente também pode ter posições relativas à própria área do RelativeLayout denominado "pai", por exemplo, um Botão está alinhado à parte inferior, à esquerda ou no centro do RelativeLayout "pai" da Activity.

O RelativeLayout é superior ao seu antecessor LinearLayout. Um exemplo é que quando surge a necessidade do LinearLayout exibir vários componentes em uma linha, é necessário criar vários grupos aninhados de visualizações. O RelativeLayout é um utilitário muito eficiente, uma vez que ele pode eliminar esses grupos de visualização aninhados e manter a hierarquia de layout plana, o que melhora o desempenho. Se você estiver usando vários grupos aninhados de LinearLayout, poderá substituí-los por um único RelativeLayout.

RelativeLayout no arquivo XML (Extensible Markup Language) da Activity

Para alterar o modo do Layout é necessário abrir o arquivo XML da Activity que fica disponível nas seguintes guias "Text" ou "Code" dependendo da versão do Android Studio que você está utilizando.

Vamos alterar o complemento do padrão ConstraintLayout para RelativeLayout, veja como vai ficar o arquivo XML.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        />

</RelativeLayout>
```

Todo componente “filho” do RelativeLayout e que não tenha sua localização informada, adota por padrão a localização no canto superior esquerdo da tela. Veja a imagem 4.

Associar a localização de um componente ao RelativeLayout

É possível definir uma localização de um componente na tela utilizando como referência o próprio RelativeLayout. Veja a seguir alguns complementos utilizados para essa associação:

android: layout_centerHorizontal: se verdadeiro, centraliza esse filho horizontalmente dentro de seus pais.

android: layout_centerInParent: se verdadeiro, centraliza esse filho horizontal e verticalmente dentro de seus pais.

android: layout_centerVertical: se verdadeiro, centraliza esse filho verticalmente dentro de seu pai.

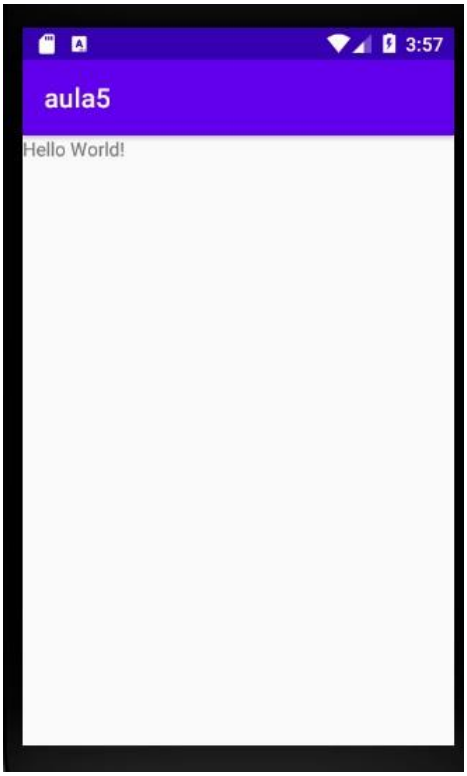


Imagem 4 – Localização padrão de um componente na tela, quando sua localização não é informada.

Para verificar outros complementos para associação da posição de um componente em uma tela, consulte o site oficial de desenvolvimento Android Studio, disponível em:https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams?hl=pt-br.

Neste momento, vamos alterar a posição do TextView para a parte central e superior da tela, este recurso está disponível por meio do complemento **android:layout_centerHorizontal.** Veja o código a seguir e o resultado na imagem 5.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```

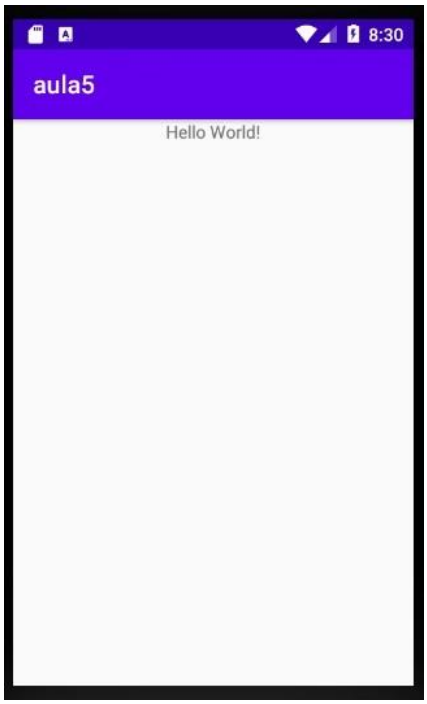


Imagem 5 – Componente TextView utilizando o complemento android:layout_centerHorizontal="true", para ficar centralizado na parte superior do RelativeLayout.

No código a seguir, vamos definir que o TextView possui um posicionamento central na Activity utilizando o **android:layout_centerInParent**. Verifique na imagem 6 o resultado esperado.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:layout_centerInParent="true"
        />

</RelativeLayout>
```

Imagem 6 – Componente TextView utilizando o complemento android:layout_centerInParent="true", para ficar centralizado no RelativeLayout.

Associar a localização de um componente a outro componente

É possível determinar a localização de um componente por meio de outro componente. No exemplo a seguir, vamos inserir mais um TextView com o texto de “Hello Word 2!” e determinar sua localização a partir do primeiro componente TextView.
Para que isso seja possível, é necessário criar um “Id” para o componente que irá servir de referência. Por padrão, vamos desenvolver o “Id” para todos os componentes utilizados na Activity.

Veja alguns complementos para a definição de localização dos componentes na tela, retirado do site oficial de desenvolvimento do Android Studio:

android:layout_above: Posiciona a borda inferior deste componente acima do componente âncora.

android:layout_below: Posiciona a borda superior deste componente abaixo do componente âncora.

android:layout_toLeftOf: Posiciona a borda direita deste componente à esquerda do componente âncora.

android:layout_toRightOf: Posiciona a borda esquerda deste componente à direita do componente âncora.

Veja o código a seguir no qual determinamos que o “Hello Word 2!” tem sua posição definida na próxima linha do “Hello Word!”, por meio do complemento android:layout_below. A imagem 7 apresenta o resultado.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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">

    <TextView
        android:id="@+id/txt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:layout_centerInParent="true"
        />

    <TextView
        android:id="@+id/txt2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/txt1"
        android:text="Hello World 2!"
        />
</RelativeLayout>
```

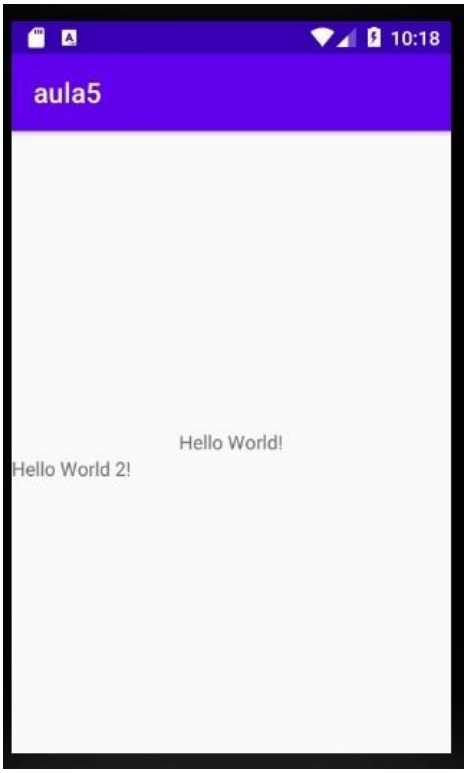


Imagem 7 – Definição da posição de um componente através de outro componente.

Alinhamentos de componentes no RelativeLayout

Vamos apresentar algumas formas de alinhamento de componentes disponíveis para o desenvolvimento de telas com o RelativeLayout. É importante que você efetue testes em componentes de um projeto com os complementos a seguir:

android:layout_alignParentBottom: Se verdadeiro, faz com que a borda inferior do componente corresponda à borda inferior do pai.

android:layout_alignParentEnd: Se verdadeiro, faz com que a borda final do componente corresponda à borda final do pai.

android:layout_alignParentLeft: Se verdadeiro, faz com que a borda esquerda do componente corresponda à borda esquerda do pai.

android:layout_alignParentRight: Se verdadeiro, faz com que a borda direita do componente corresponda à borda direita do pai.

android:layout_alignParentStart: Se verdadeiro, faz com que a borda inicial do componente corresponda à borda inicial do pai.

Chegou a sua vez de colocar a “mão na massa” e utilizar o RelativeLayout. Desenvolva um novo projeto no Android Studio com o nome de “Jogodavelha”, como mostra a imagem 8.

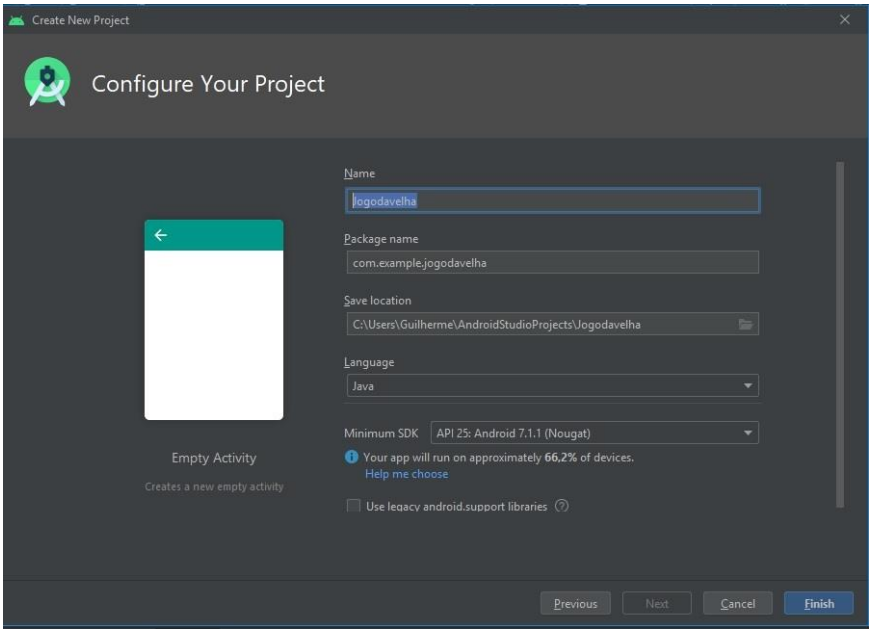


Imagem 8 – Dados para desenvolvimento de um novo projeto chamado “Jogodavelha”.

Vamos praticar o desenvolvimento de uma Activity utilizando o RelativeLayout. Por meio do arquivo XML da Activity “MainActivity” disponível em: **app > res > layout > activity_main.xml.** Utilize a guia “Text” ou “Code” na tela do Android Studio para acessar os códigos XML para desenvolver a tela conforme a imagem 9.

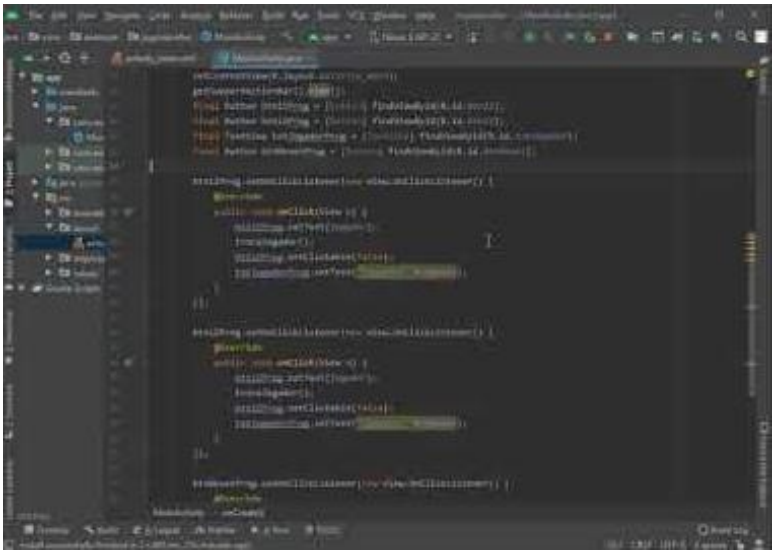
Imagem 9 – Tela e componentes da MainActivity do projeto “Jogodavelha”.

Utilize os códigos a seguir como apoio para realização da sua tela que utiliza o RelativeLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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/btn11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/btn22"
        android:layout_toLeftOf="@id/btn22"
        />
    <Button
        android:id="@+id/btn12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/btn22"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/btn13"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/btn22"
        android:layout_toRightOf="@id/btn22"
        />
    <Button
        android:id="@+id/btn22"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        />
    <Button
        android:id="@+id/btn23"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_toRightOf="@id/btn22"
        />
    <Button
        android:id="@+id/btn21"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_toLeftOf="@id/btn22"
        />
    <Button
        android:id="@+id/btn31"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_below="@id/btn22"
        android:layout_toLeftOf="@id/btn22"
        />
    <Button
        android:id="@+id/btn32"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/btn22"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/btn33"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/btn22"
        android:layout_toRightOf="@id/btn22"
        />

    <Button
        android:id="@+id/btnReset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="Reset" />

    <TextView
        android:id="@+id/txtJogador"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Jogador: X"
        android:textSize="30dp"
        android:textAlignment="center"
        android:textColor="@android:color/darker_gray"
        android:layout_centerHorizontal="true"
        />
</RelativeLayout>
```

Com base no vídeo 1, efetue a programação Java utilizada no projeto, com o arquivo disponível em: **app > java > com.example. jogodavelha > MainActivity.**



Vídeo 1 – Programação Java da MainActivity. Disponível em: https://youtu.be/SKwOl4QWkZI

Código Java utilizado no vídeo anterior para programação da MainActivity:

```
package com.example.jogodavelha;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    String jogador = "X";

    public void trocaJogador()
    {
        if(jogador.equals("X"))
        {
            jogador = "O";
        }
        else
        {
            jogador = "X";
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();
        final Button btn11Prog = (Button) findViewById(R.id.btn11);
        final Button btn12Prog = (Button) findViewById(R.id.btn12);
        final TextView txtJogadorProg = (TextView) findViewById(R.id.txtJogador);
        final Button btnResetProg = (Button) findViewById(R.id.btnReset);

        btn11Prog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btn11Prog.setText(jogador);
                trocaJogador();
                btn11Prog.setClickable(false);
                txtJogadorProg.setText("Jogador: "+jogador);
            }
        });

        btn12Prog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btn12Prog.setText(jogador);
                trocaJogador();
                btn12Prog.setClickable(false);
                txtJogadorProg.setText("Jogador: "+jogador);
            }
        });
```

```
        btnResetProg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btn11Prog.setClickable(true);
                btn11Prog.setText("");
                btn12Prog.setClickable(true);
                btn12Prog.setText("");
            }
        });

    }
}
```

Desenvolva a codificação Java para que o projeto funcione com todas as opções de jogadas. Na sequência, capture uma imagem do programa executado e envie para seu professor, juntamente com os códigos XML e Java utilizados para desenvolvimento do projeto “Jogodavelha”.

Para auxiliar no processo de aprendizagem dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes e de um livro que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeos: