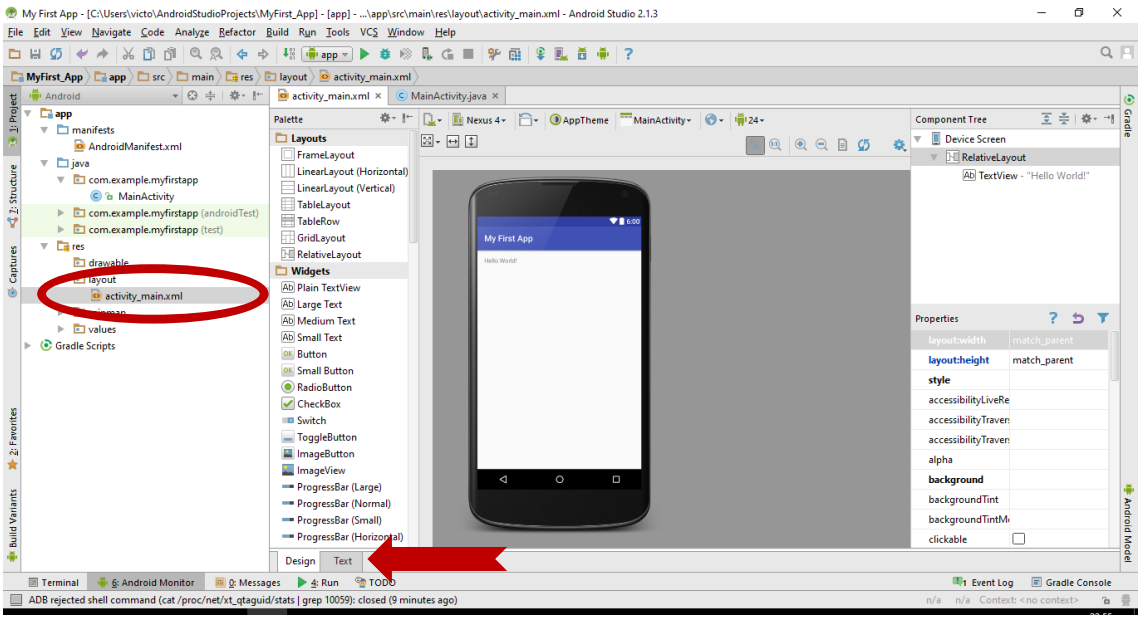
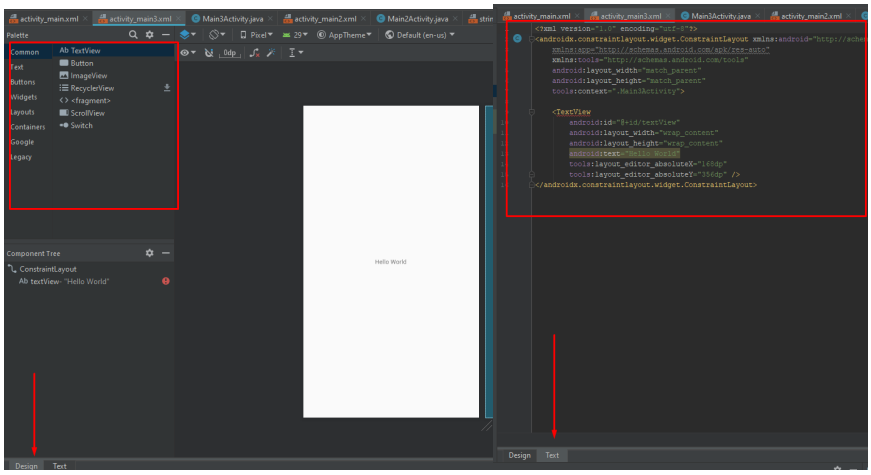
Desenvolvimento de Aplicativos Android

Criando Interface Simples

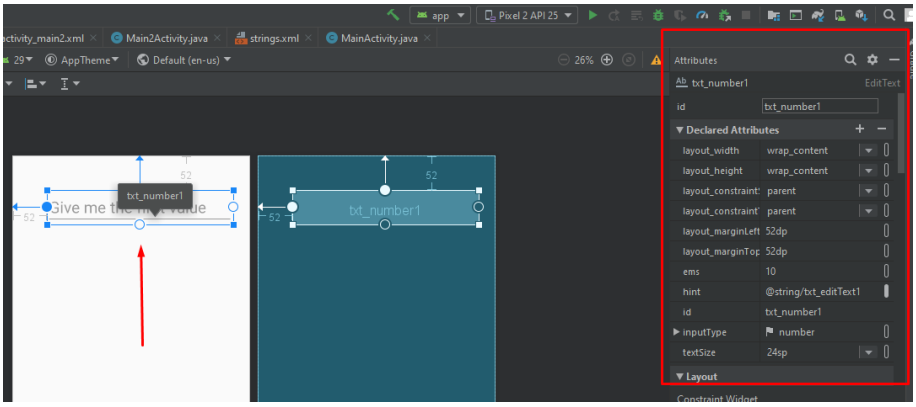
A partir do diretório res/layout/ abra o activity_main.xml. Este arquivo XML define o layout da sua activity. Ele contém o texto padrão "Hello World".



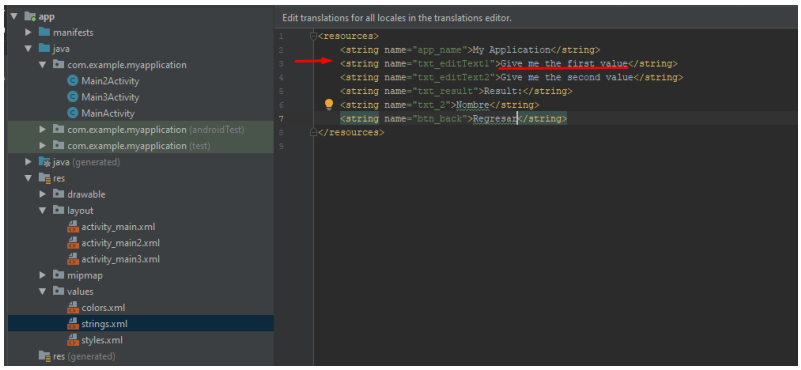
Android Studio tem componentes que pode ser adicionados pela interface ou por código xml, sim faz câmbios na parte de interface , o código xml será mudado automaticamente e vice versa.



Em desenvolvimento em Android é importante que cada componente tenha um ID irrepetível, para ser usado na parte lógica, assim mesmo , todas as propriedades dos componentes como Buttons, Texts, Widgets podem ser editados pela codigo e pela interface na janela Attributes



Os textos que vai na interface é recomendável que sejam mapeados num string.xml e chamados pela su código, assim quando desejamos usar um texto repetitivo, chamaremos só a uma instância



Você já está pronto para fazer as interfaces da prática, mas como conectar com a parte lógica?

Conexão com a parte Lógica

Cada interface tem uma parte visual .xml e outra lógica .java, para ler componentes é necessário buscar pela ID de cada componente com o método findViewById() em R "Variável de comunicação".

MainActivity.java

```
public class MainActivity extends AppCompatActivity
{

   private EditText et1;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      et1 = (EditText) findViewById(R.id.txt_number1); // ID from component
      Toast.makeText(this, "Value: " + et1.getText().toString(), Toast.LENGTH_SHORT).show();
```

```
    // The activity is created
    }
}
```

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">

   <EditText
      android:id="@+id/txt_number1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="52dp"
      android:layout_marginLeft="52dp"
      android:layout_marginTop="52dp"
      android:ems="10"
      android:hint="Give me the first Value"
      android:inputType="number"
      android:textSize="24sp"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />

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

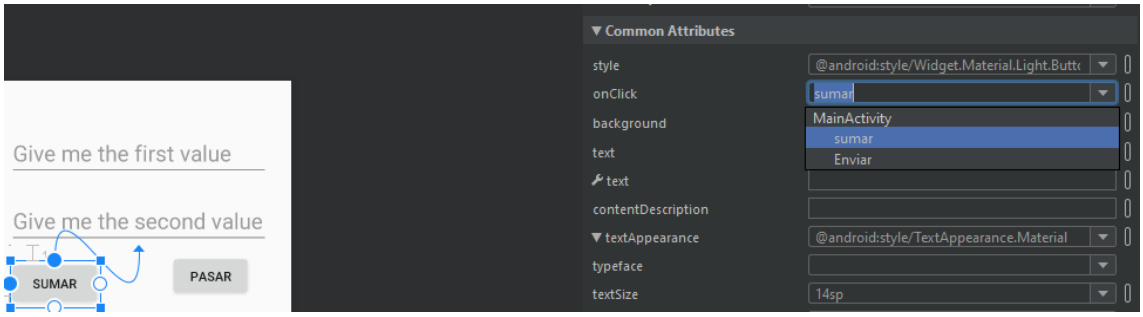
Os componentes Button tem uma propriedade OnClick para chamar a métodos da parte lógica. Os Métodos que vai ser chamados desde a interface devem ter um atributo View

MainActivity.java

```
public void sumar(View view)
{
    //code here
}
```

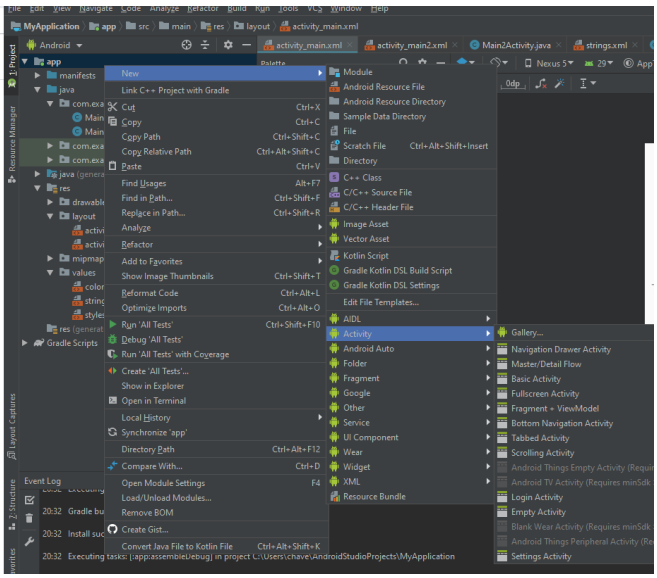
activity_main.xml

```
<Button
    android:id="@+id/btn_sumar"
    android:layout_width="99dp"
    android:layout_height="51dp"
    android:layout_marginStart="52dp"
    android:layout_marginLeft="52dp"
    android:layout_marginTop="16dp"
    android:onClick="sumar"
    android:text="sumar"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/txt_number1" />
```



Você já pode fazer aplicações com uma soa Activity. Mas como interactuar com mas de uma?

Para criar uma nova activity faz Click direito na app > New > Activity > Empty Activity , cada activity tem uma parte de interface (.xml) e outra lógica (.java),



Para chamar uma activity se deve definir um Intent que pode conter informação através de um mapeamento <key, value> e assim podemos enviar informações de cualquier Activity a outra . O chamamento de uma Activity se faz pela método StartActivity

MainActivity.java

```
public void Send(View view)
  {
     Intent i = new Intent(this, Main2Activity.class);
     i.putExtra("data", res.getText().toString());
     startActivity(i);
  }
```

activity_main.xml

```
<Button
     android:id="@+id/btn_pass"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_marginStart="228dp"
     android:layout_marginLeft="228dp"
```

```
    android:layout_marginTop="16dp"
    android:onClick="Enviar"
    android:text="Pasar"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/txt_number2" />
```

Para pegar informação desde outros Activity so precisamos da chave que foi enviado

Main2Activity.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);
    ed1 = (TextView) findViewById(R.id.txt_2);
    String dato = getIntent().getStringExtra("data");
    ed1.setText(dato);
  }
```

activity_main2.xml

```
<TextView
    android:id="@+id/txt_2"
    android:layout_width="146dp"
    android:layout_height="88dp"
    android:layout_marginStart="84dp"
    android:layout_marginLeft="84dp"
    android:layout_marginTop="68dp"
    android:text="@string/txt_2"
    android:textSize="24sp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Como obter os dados dos sensores de movimento

Agora que já fizemos nosso aplicativo com mais de uma tela, podemos continuar aprendendo para fazer aplicativos mais úteis. Uma grande vantagem está na utilização dos sensores dos dispositivos móveis. Para usar os sensores é necessário fazer a nossa *activity* implementar a classe **SensorEventListener**. Quando implementamos esta classe, obrigatoriamente devemos incluir dois métodos na *activity*: **onSensorChanged** e **onAccuracyChanged**.

O primeiro método será chamado toda vez que houver alguma alteração nos valores dos sensores que estamos monitorando, o segundo método é chamado quando ocorre alguma mudança na precisão dos valores obtidos. Para a nossa *activity* começar a escutar os sensores é necessário registrar um *listener* com os sensores que desejamos escutar e com qual frequência desejamos receber os eventos. O código a seguir é um exemplo de como receber os valores do acelerômetro:

Capturamos os dados do sensor.

```
public class MainActivity extends AppCompatActivity implements SensorEventListener {


   private  SensorManager mSensorManager;
   private  Sensor mAccelerometer;

   TextView coordinate;

   @Override
   protected void onCreate(Bundle savedInstanceState) {

      mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
      mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

      coordinate = (TextView) findViewById(R.id.textView2);
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

   }

   @Override
   protected void onResume() {
      super.onResume();
      mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
   }

   @Override
   protected void onPause() {
      super.onPause();
      mSensorManager.unregisterListener(this);
   }

   @Override
   public void onSensorChanged(SensorEvent event) {

      if(event.sensor.getType()== Sensor.TYPE_ACCELEROMETER) {
         float sensorX = event.values[0];
         float sensorY = event.values[1];
         float sensorZ = event.values[2];

         coordinate.setText(String.valueOf(sensorX) + " " + String.valueOf(sensorY) + " " + String.valueOf(sensorZ));
      }

   }

   @Override
   public void onAccuracyChanged(Sensor sensor, int accuracy)
   {

   }
}
```

Mostramos numa TexView

```
<TextView
      android:id="@+id/textView2"
      android:layout_width="212dp"
      android:layout_height="68dp"
      android:layout_marginStart="16dp"
      android:layout_marginLeft="16dp"
      android:layout_marginTop="12dp"
      android:text="TextView"
      android:textSize="24sp"
```

```
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/txt_result" />
```