Outros Sensores - Kotlin

Prof. Me. Hélio Esperidião

Layout.

```
<LinearLayout
   android:layout_height="match_parent"
   android:layout_width="match_parent"
   android:orientation="vertical">
   <TextView
      android:id="@+id/lblPressao"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text=""      />
   <TextView
      android:id="@+id/lblLuminosidade"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text=""    />
   <TextView
      android:id="@+id/lblTemperatura"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text=""     />
   <TextView
      android:id="@+id/lblProximidade"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text=""   />
</LinearLayout>
```

imports

```
import android.content.Context
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView
```

Implemente a interface SensorEvent Listener

```
class MainActivity : AppCompatActivity() , SensorEventListener {
```

Atributos

```
lateinit var acelerometro: Sensor
lateinit var sensorProximidade: Sensor
lateinit var sensorPressao: Sensor
lateinit var sensorLuminosidade: Sensor
lateinit var gerenciadorSensor: SensorManager
```

On Create

```
override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      inicializarSensores()
  }
```

inicializarAcelerometro(){

```
fun inicializarSensores(){

    gerenciadorSensor = getSystemService(Context.SENSOR_SERVICE) as SensorManager

    sensorTemperatura =  gerenciadorSensor.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE);
    sensorProximidade =  gerenciadorSensor.getDefaultSensor(Sensor.TYPE_PROXIMITY);
    sensorPressao =  gerenciadorSensor.getDefaultSensor(Sensor.TYPE_PRESSURE);
    sensorLuminosidade =  gerenciadorSensor.getDefaultSensor(Sensor.TYPE_LIGHT);

    gerenciadorSensor.registerListener(this, this.sensorTemperatura, SensorManager.SENSOR_DELAY_NORMAL);
    gerenciadorSensor.registerListener(this, this.sensorProximidade, SensorManager.SENSOR_DELAY_NORMAL);
    gerenciadorSensor.registerListener(this, this.sensorPressao, SensorManager.SENSOR_DELAY_NORMAL);
    gerenciadorSensor.registerListener(this, this.sensorLuminosidade, SensorManager.SENSOR_DELAY_NORMAL);

  }
```

Evento: onSensorChanged

```
override fun onSensorChanged(event: SensorEvent?) {
    if (event != null) {
      var tipoSensor = event.sensor.type
      if(tipoSensor==Sensor.TYPE_PRESSURE){
        var pressao:Float = Float.NaN
        pressao = event.values[0]
        val lblPressao: TextView = findViewById(R.id.lblPressao)
        lblPressao.text = "Pressão: "+pressao.toString();
      }else if(tipoSensor==Sensor.TYPE_AMBIENT_TEMPERATURE){
        var temperatura: Float = Float.NaN
        temperatura = event.values[0]
        val lblTemperatura: TextView = findViewById(R.id.lblTemperatura)
        lblTemperatura.text = "Temperatura: "+temperatura.toString();
      } else if(tipoSensor==Sensor.TYPE_LIGHT){
        var luminosidade:Float = Float.NaN
        luminosidade = event.values[0]
        val lblLuminosidade: TextView = findViewById(R.id.lblLuminosidade)
        lblLuminosidade.text = "Luminosidade: " +luminosidade.toString();
      }else if(tipoSensor==Sensor.TYPE_PROXIMITY){
        var proximidade:Float = Float.NaN
        proximidade = event.values[0]
        val lblProximidade: TextView = findViewById(R.id.lblProximidade)
        lblProximidade.text = "Proximidade:" + proximidade.toString();
      }
    }
  }
  override fun onAccuracyChanged(p0: Sensor?, p1: Int) {
  }
```

Existem vários outros sensores, a maioria funciona de forma semelhante aos exemplos

```
public static final String STRING_TYPE_TEMPERATURE = "android.sensor.temperature";
public static final int TYPE_ACCELEROMETER = 1;
public static final int TYPE_ACCELEROMETER_LIMITED_AXES = 38;
public static final int TYPE_ACCELEROMETER_LIMITED_AXES_UNCALIBRATED = 40;
public static final int TYPE_ACCELEROMETER_UNCALIBRATED = 35;
public static final int TYPE_ALL = -1;
public static final int TYPE_AMBIENT_TEMPERATURE = 13;
public static final int TYPE_DEVICE_PRIVATE_BASE = 65536;
public static final int TYPE_GAME_ROTATION_VECTOR = 15;
public static final int TYPE_GEOMAGNETIC_ROTATION_VECTOR = 20;
public static final int TYPE_GRAVITY = 9;
public static final int TYPE_GYROSCOPE = 4;
public static final int TYPE_GYROSCOPE_LIMITED_AXES = 39;
public static final int TYPE_GYROSCOPE_LIMITED_AXES_UNCALIBRATED = 41;
public static final int TYPE_GYROSCOPE_UNCALIBRATED = 16;
public static final int TYPE_HEADING = 42;
public static final int TYPE_HEAD_TRACKER = 37;
public static final int TYPE_HEART_BEAT = 31;
public static final int TYPE_HEART_RATE = 21;
public static final int TYPE_HINGE_ANGLE = 36;
public static final int TYPE_LIGHT = 5;
public static final int TYPE_LINEAR_ACCELERATION = 10;
public static final int TYPE_LOW_LATENCY_OFFBODY_DETECT = 34;
public static final int TYPE_MAGNETIC_FIELD = 2;
public static final int TYPE_MAGNETIC_FIELD_UNCALIBRATED = 14;
public static final int TYPE_MOTION_DETECT = 30;
```