OCR - Kotlin

Prof. Me. Hélio Esperidião

OCR significa

Optical Character Recognition, que em tradução livre significa Reconhecimento Óptico de Caracteres.

A tecnologia permite a conversão de documentos(imagens) em dados que o usuário pode pesquisar e editar.

Manifesto

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.aulaocr">
    <uses-permission android:name="android.permission.CAMERA" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AulaOCR">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data
            android:name="com.google.android.gms.vision.DEPENDENCIES"
            android:value="ocr" />
    </application>

</manifest>
```

Layout

```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">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <TextView
            android:id="@+id/tv_result"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <SurfaceView
            android:id="@+id/surface_camera_preview"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>

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

Dependências - Gradle

```
implementation 'androidx.camera:camera-camera2:1.0.0-beta03'
implementation 'com.google.android.gms:play-services-vision:15.0.1'
```

Atributos e onCreate

```
class MainActivity : AppCompatActivity() {
    private val PERMISSION_REQUEST_CAMERA = 100

    private lateinit var cameraSource: CameraSource
    private lateinit var recognizer :TextRecognizer
    private lateinit var surface_camera_preview: SurfaceView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        this.surface_camera_preview = findViewById(R.id.surface_camera_preview)
        iniciarCamera()
    }
```

isCameraPermissionGranted

```
fun isCameraPermissionGranted(): Boolean {
    return ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA) ==
            PackageManager.PERMISSION_GRANTED
}
```

requestForPermission()

```
private fun requestForPermission() {
    ActivityCompat.requestPermissions(this, arrayOf(Manifest.permission.CAMERA), PERMISSION_REQUEST_CAMERA)
}
```

onRequestPermissionsResult

```
@SuppressLint("MissingPermission")
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
    if (requestCode != PERMISSION_REQUEST_CAMERA) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        return
    }

    if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        if (isCameraPermissionGranted()) {
            cameraSource.start(surface_camera_preview.holder)
        } else {

            finish()
        }
    }
}
```

iniciarCamera()

```
this.recognizer = TextRecognizer.Builder(this).build()
this.cameraSource = CameraSource.Builder(applicationContext, recognizer)
        .setFacing(CameraSource.CAMERA_FACING_BACK)
        .setRequestedPreviewSize(1280, 1024)
        .setAutoFocusEnabled(true)
        .setRequestedFps(2.0f)
        .build()
```

```
this.surface_camera_preview.holder.addCallback(object : SurfaceHolder.Callback {
    override fun surfaceChanged(p0: SurfaceHolder?, p1: Int, p2: Int, p3: Int) {

    }

    override fun surfaceDestroyed(p0: SurfaceHolder?) {
        cameraSource.stop()
    }

    @SuppressLint("MissingPermission")
    override fun surfaceCreated(p0: SurfaceHolder?) {
        try {
            if (isCameraPermissionGranted()) {
                cameraSource.start(surface_camera_preview.holder)
            } else {
                requestForPermission()
            }
        } catch (e: Exception) {

        }
    }
})
```

Fim do iniciarCamera()

```
recognizer.setProcessor(object : Detector.Processor<TextBlock> {
        override fun release() {}

        override fun receiveDetections(detections: Detector.Detections<TextBlock>) {
            val items = detections.detectedItems

            if (items.size() <= 0) {
                return
            }

            val tv_result: TextView = findViewById(R.id.tv_result)
            tv_result.post {
                val stringBuilder = StringBuilder()
                for (i in 0 until items.size()) {
                    val item = items.valueAt(i)
                    stringBuilder.append(item.value)
                    stringBuilder.append("\n")
                }
                tv_result.text = stringBuilder.toString()
            }
        }
    })
}
```