UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

DEPARTAMENTO DE CIENCIAS DE LA COMUNICACIÓN

SOFTWARE

MATERIA:
Aplicaciones Móviles

Nombres:
Alisson Clavijo
Miguel Morales
Carlos Romero
Cristopher Zambrano

Deber 1

NRC:
15314

Docente:

ING.DORIS KARINA CHICAIZA ANGAMARCA

1. **Tema**
   Desarrollo de una aplicación móvil.
2. **Objetivos**
   - Desarrollar una aplicación móvil Android utilizando lenguaje de programación Kotlin con el IDE Android Studio para la conversión de divisas / monedas (al menos 10).
   - Utilizar los controles básicos como textview, radio button y radio group.
   - Agregar un icono a la aplicación.

3. **Desarrollo(Código)**
   MainActivity.kt

   En el código se ve la interfaz de usuario que incluye botones, spinners y un campo de texto para la entrada de la cantidad a convertir. La lógica de conversión se activa al hacer clic en un botón, obteniendo la cantidad y las monedas seleccionadas en los spinners. Además, se ha aplicado un filtro al campo de texto para permitir solo números decimales con un máximo de 5 dígitos, incluyendo 1 decimal. Utilizamos tasas de conversión definidas para calcular el resultado.

```
package com.example.prymonedas

import android.os.Bundle
import android.text.InputFilter
import android.text.Spanned
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.Spinner
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.util.regex.Pattern

class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
       super.onCreate(savedInstanceState)
       setContentView(R.layout.activity_main)

       val convertButton: Button = findViewById(R.id.btn_convert)
       convertButton.setOnClickListener { convertCurrency() }

       val opciones = arrayOf("USD", "EUR", "JPY", "GBP", "AUD", "CAD", "CHF", "CNH", "HKD",
"NZD")

       val spinner: Spinner = findViewById(R.id.sp_from)
       val spinner2: Spinner = findViewById(R.id.sp_to)

       val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, opciones)

       adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

       spinner.adapter = adapter
       spinner2.adapter = adapter

       val editText: EditText = findViewById(R.id.etn_currency)
       editText.setFilters(arrayOf<InputFilter>(DecimalDigitsInputFilter(5, 1)))
```

```
    }

    private fun convertCurrency() {
        val fromAmountEditText: EditText = findViewById(R.id.etn_currency)
        val spinner: Spinner = findViewById(R.id.sp_from)
        val spinner2: Spinner = findViewById(R.id.sp_to)

        val fromAmount = fromAmountEditText.text.toString()

        if (fromAmount.isEmpty()) {
            Toast.makeText(this, "Ingresa una cantidad válida", Toast.LENGTH_SHORT).show()
            return
        }

        val fromCurrency = spinner.selectedItem.toString()
        val toCurrency = spinner2.selectedItem.toString()

        if (fromCurrency == toCurrency) {
            Toast.makeText(this, "Selecciona monedas diferentes", Toast.LENGTH_SHORT).show()
            return
        }

        val conversionRate = getConversionRate(fromCurrency, toCurrency)
        val result = fromAmount.toDouble() * conversionRate

        val toastMessage = String.format("%.2f %s = %.2f %s", fromAmount.toDouble(),
fromCurrency, result, toCurrency)
        showResult(toastMessage)
    }

    private fun getConversionRate(currencyFrom: String, currencyTo: String): Double {
        val conversionRates = mapOf(
            "USD" to mapOf("EUR" to 0.85, "GBP" to 0.75, "JPY" to 104.54, "AUD" to 1.36, "CAD"
to 1.31, "CHF" to 0.91, "CNH" to 6.55, "HKD" to 7.75, "NZD" to 1.43),
            "EUR" to mapOf("USD" to 1.18, "GBP" to 0.89, "JPY" to 123.37, "AUD" to 1.62, "CAD"
to 1.56, "CHF" to 1.09, "CNH" to 7.81, "HKD" to 9.27, "NZD" to 1.71),
            "GBP" to mapOf("USD" to 1.33, "EUR" to 1.12, "JPY" to 138.59, "AUD" to 1.82, "CAD"
to 1.75, "CHF" to 1.22, "CNH" to 8.74, "HKD" to 10.38, "NZD" to 1.92),
        )

        return conversionRates[currencyFrom]?.get(currencyTo) ?: 1.0
    }

    private fun showResult(result: String) {
        val resultTextView: EditText = findViewById(R.id.et_result)
        resultTextView.setText(result)
    }

    private class DecimalDigitsInputFilter(digitsBeforeZero: Int, digitsAfterZero: Int) :
        InputFilter {
        private val mPattern: Pattern
        init {
            mPattern =
Pattern.compile("[0-9]{0,$digitsBeforeZero}+((\\.[0-9]{0,$digitsAfterZero})?)||(\\.)?")
        }
        override fun filter(
            source: CharSequence,
            start: Int,
            end: Int,
            dest: Spanned,
            dstart: Int,
            dend: Int
        ): CharSequence? {
```

```
            val matcher = mPattern.matcher(dest)
            return if (!matcher.matches()) "" else null
        }
    }
}
```

Observamos al final implementamos un filtro de entrada para que permita solo números decimales en el campo de texto para que muestre el resultado final.

ExampleUnitTest.kt

Implementamos un caso de prueba unitaria en kotlin, donde vemos que la suma de dos números es igual al resultado procedente de este.

```
package com.example.prymonedas

import org.junit.Test

import org.junit.Assert.*

/**
* Example local unit test, which will execute on the development machine (host).
*
* See [testing documentation](http://d.android.com/tools/testing).
*/
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

ExampleInstrumentedTest.kt

Aquí se usa el Android en sí para que se ejecute en un dispositivo real o emulado, donde se usarán JUnit en un entorno Android.

```
package com.example.prymonedas

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
* Instrumented test, which will execute on an Android device.
*
* See [testing documentation](http://d.android.com/tools/testing).
*/
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
```

```
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.prymonedas", appContext.packageName)
    }
}
```

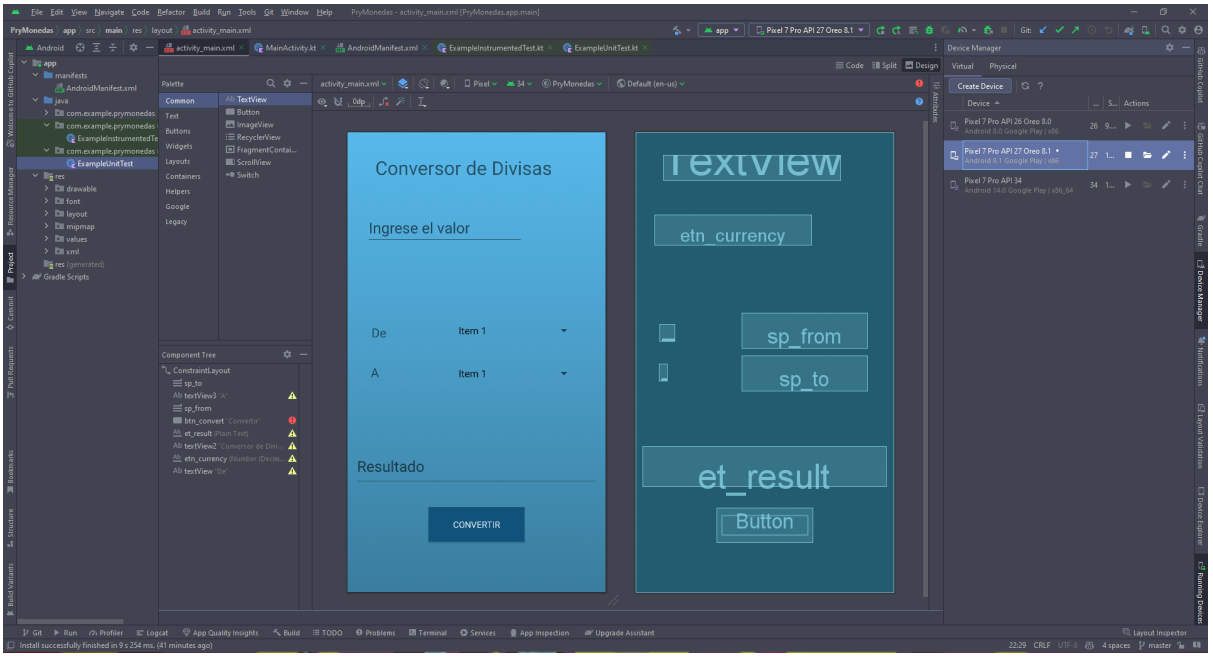
AndroidManifest.xml

Aquí solo vemos las configuraciones básicas de la aplicación Android. Vemos una breve descripción de los elementos.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PryMonedas"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>

</manifest>
```

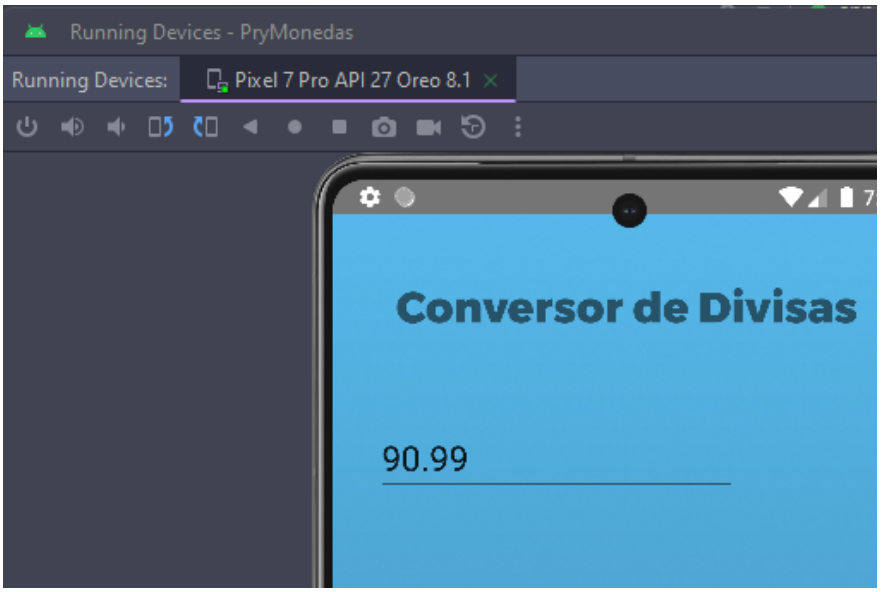
activity_main.xml

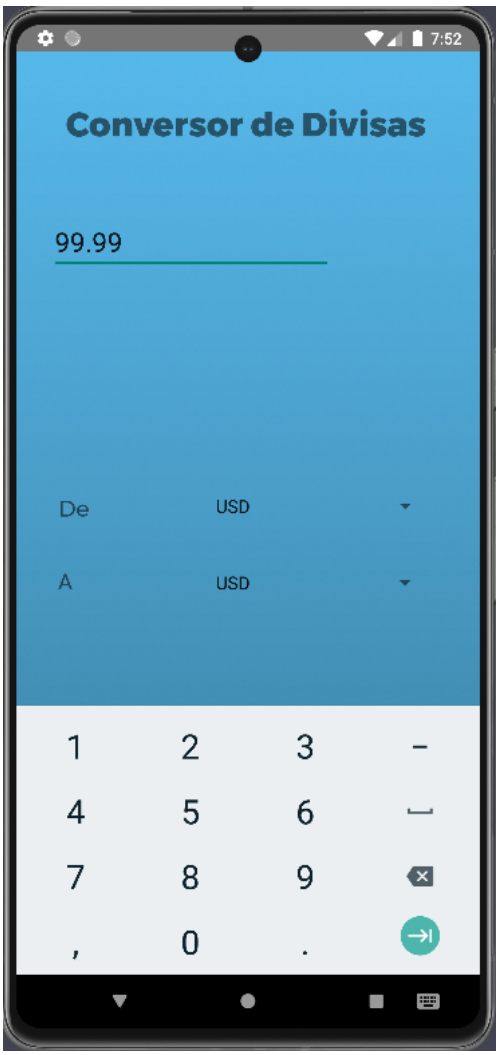
Aquí observamos la interfaz algo estéticos y minimalistas al ser el primer ejercicio de enseñanza cambiando un poco el fondo que es el color de sus botones antes dichos sus cuadros de textos, etc.

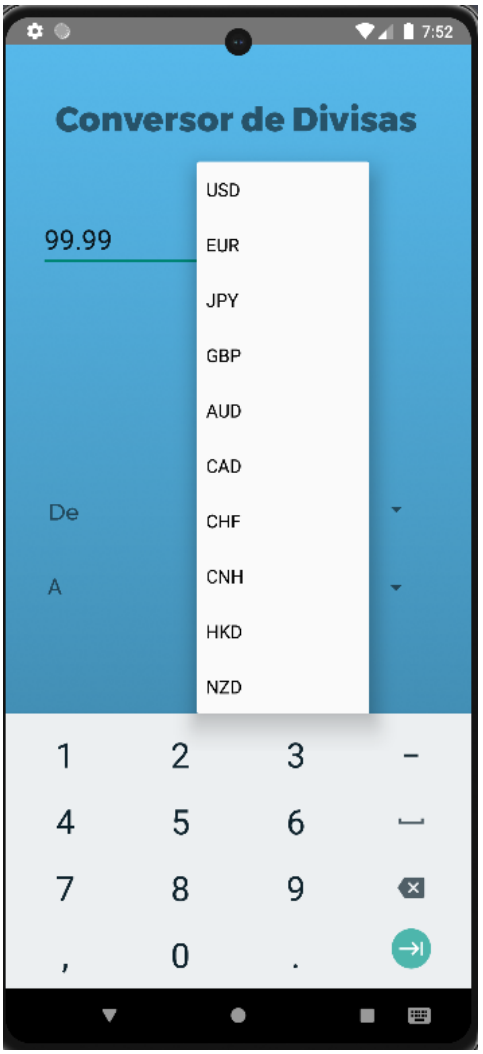


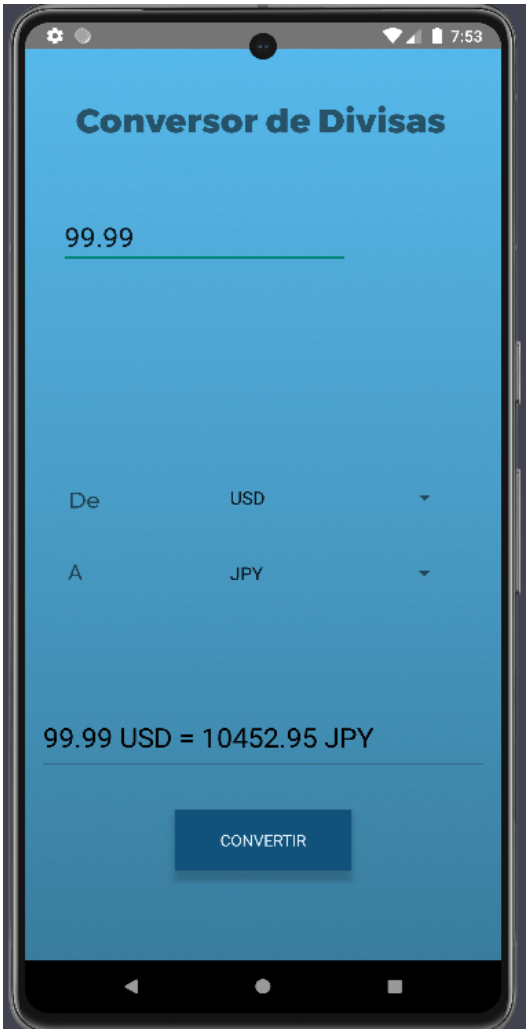
4. **Aplicación Móvil(Ejecución)**
    - **Api de desarrollo es API Oreo 8.1**

      Usamos el API Oreo 8.1 junto con el emulador para dar la experiencia y el uso que le estamos dando, tanto para dar la prueba práctica como para la ejecución del app.




7:52
Conversor de Divisas
99.99
De
USD
A
USD
1 2 3 -
4 5 6
7 8 9
, 0 .


7:52
Conversor de Divisas
99.99
USD
EUR
JPY
GBP
AUD
CAD
CHF
CNH
HKD
NZD
De
A
1 2 3 -
4 5 6
7 8 9
, 0 .



Hemos puesto también el icono de android el cual se puede visualizar para identificar nuestra app.

5. Conclusiones

- Kotlin es un lenguaje moderno, conciso y completamente interoperable con Java. Además, su sintaxis mejorada y características de seguridad lo convierten en una opción sólida para el desarrollo de aplicaciones Android.
- La elección de controles básicos como los TextView son esenciales para mostrar información, mientras que los RadioButton y RadioGroup facilitan la selección de la moneda de origen y destino de manera intuitiva.
- La inclusión de un icono personalizado ayudó a mejorar la identidad visual de la aplicación y hacer que sea más reconocible en la pantalla de inicio del dispositivo.

6. Referencias

- Eddy [@THEskril70]. (2022, March 31). Cómo subir tu proyecto desde ANDROID STUDIO a GITHUB. Youtube. https://www.youtube.com/watch?v=7nSt0MpwKAw
- Cómo agregar íconos giratorios a tu app. (n.d.). Android Developers. Retrieved November 21, 2023, from https://developer.android.com/develop/ui/views/components/spinner?hl=es-419