CURRENCY CONVERTER

Programación de Dispositivos Móviles

Sebastian Velasquez Perea

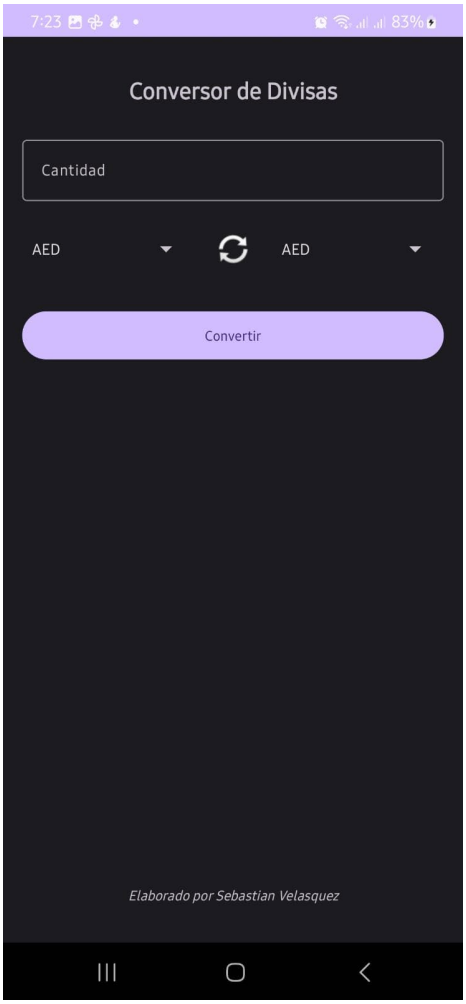
Universidad Autónoma Latinoamericana

18 de marzo del 2025

1.

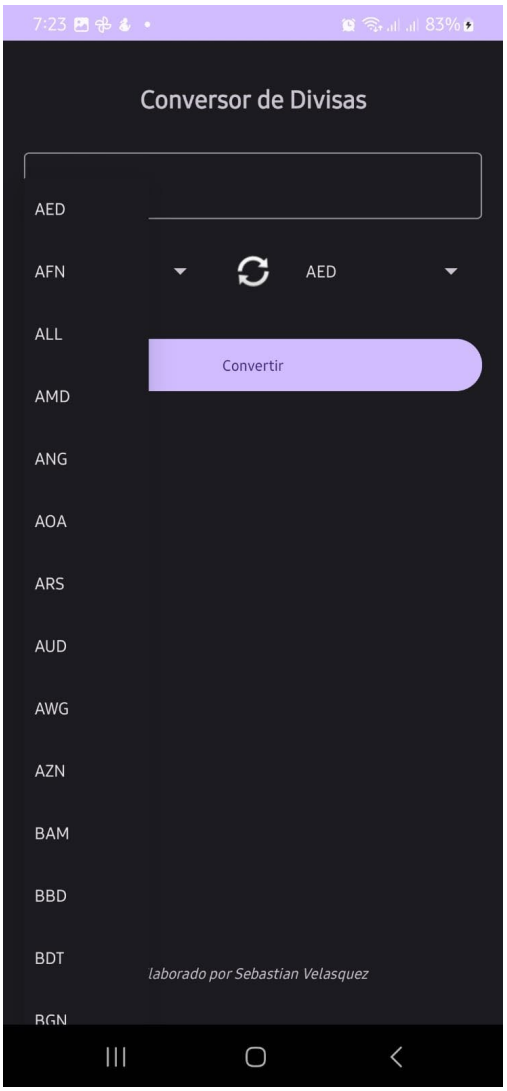
La aplicación Currency Converter es una herramienta moderna y eficiente para la conversión de divisas en tiempo real. Desarrollada como parte del curso de Programación de Dispositivos Móviles, esta aplicación demuestra la implementación de buenas prácticas de desarrollo en Android.

2. CAPTURAS DE PANTALLA DE LA APLICACIÓN



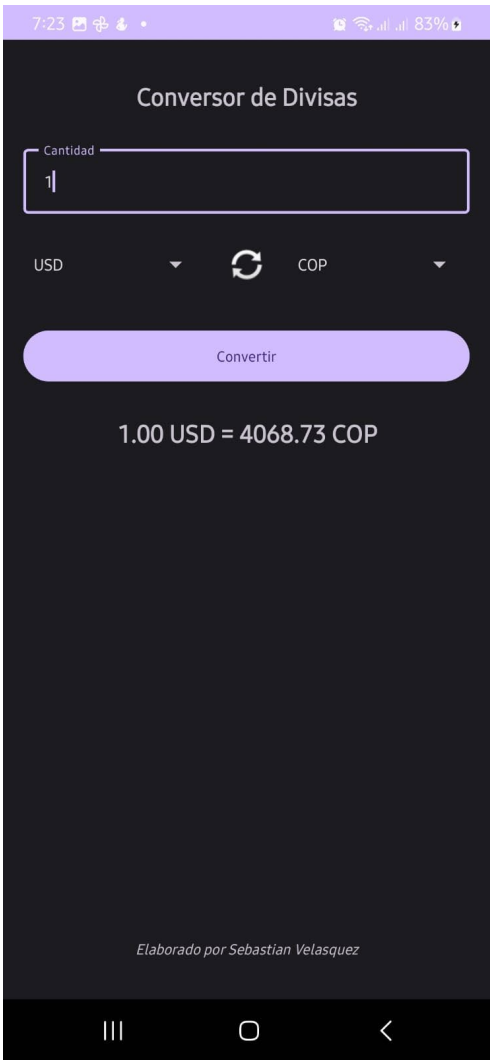
Pantalla Principal

Interfaz, Campo para ingresar cantidad, Selectores de moneda origen y destino, Botón de conversión.



Proceso de Selección de Moneda

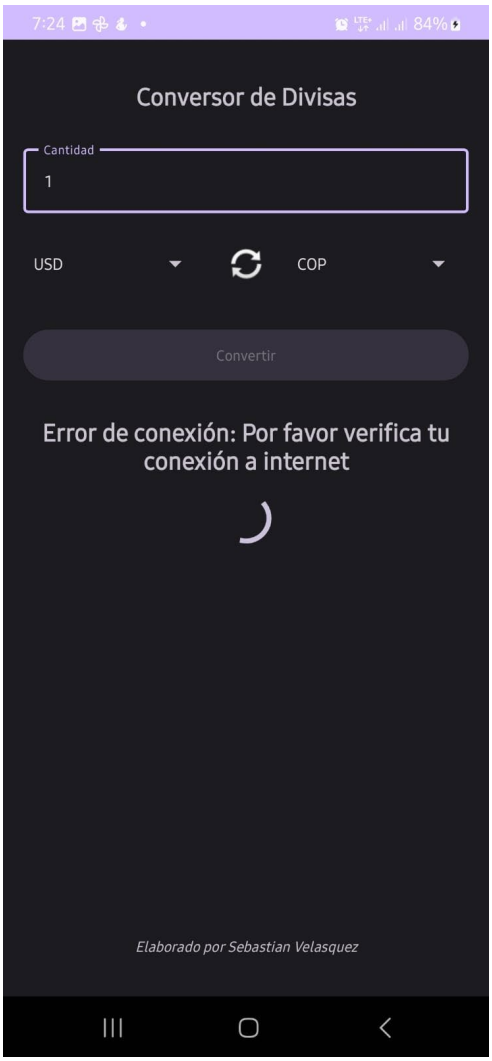
Lista desplegable de monedas disponibles, Organización alfabética de monedas



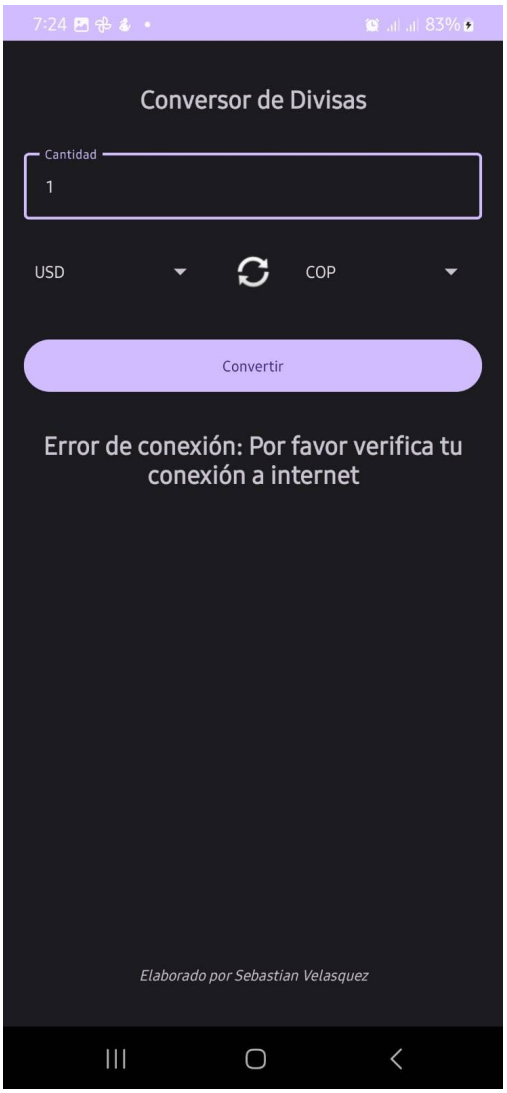
Resultado de la Conversión

- Formato [cantidad] [moneda origen] = [resultado] [moneda destino]

- Actualización en tiempo real (toca pagar una suscripción la gratuita es cada 12 horas)



Apartado de carga cuando hace la solicitud a la api



Manejo de errores ejemplo falta de internet

3. ESTRUCTURA DEL PROYECTO

El proyecto está organizado en los siguientes componentes:

Archivos Principales:

- MainActivity.kt: Actividad principal

- CurrencyViewModel.kt: Lógica

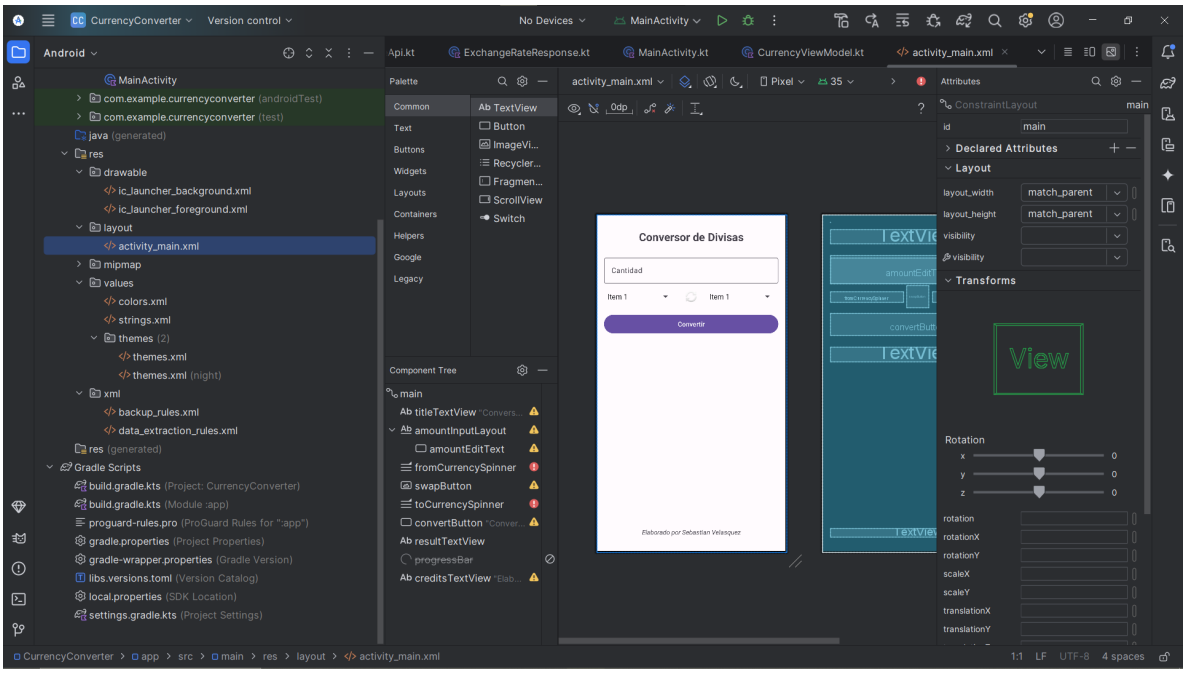
- ExchangeRateApi.kt: Interfaz de API

- ExchangeRateResponse.kt: Modelo de datos

Código principal

INTERFAZ DE USUARIO (MainActivity.kt):

// Título de la aplicación



LÓGICA DE la aplicación (CurrencyViewModel.kt):

```kotlin
/**
 * ViewModel que maneja la lógica de negocio para la conversión de divisas.
 * Gestiona las llamadas a la API y mantiene el estado de la UI.
 */
class CurrencyViewModel : ViewModel() {
    // LiveData para el resultado de la conversión
    private val _conversionResult = MutableLiveData<String>()
    val conversionResult: LiveData<String> = _conversionResult

    // LiveData para el estado de carga
    private val _isLoading = MutableLiveData<Boolean>()
    val isLoading: LiveData<Boolean> = _isLoading

    // LiveData para la lista de monedas disponibles
    private val _availableCurrencies = MutableLiveData<List<String>>()
    val availableCurrencies: LiveData<List<String>> = _availableCurrencies

    // Inicialización de la API de Retrofit
    private val api: ExchangeRateApi = Retrofit.Builder()
        .baseUrl("https://v6.exchangerate-api.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()
        .create(ExchangeRateApi::class.java)
```

Referencia a captura de pantalla 2 donde se despliega la lista de monedas.

```kotlin
/**
 * Carga la lista de monedas disponibles desde la API.
 * En caso de error, proporciona una lista predeterminada de monedas comunes.
 */
private fun loadAvailableCurrencies() {
    viewModelScope.launch {
        try {
            val response = api.getExchangeRates( baseCurrency: "USD")
            _availableCurrencies.value = response.conversion_rates.keys.toList().sorted()
        } catch (e: IOException) {
            _availableCurrencies.value = listOf("USD", "EUR", "GBP", "JPY", "MXN", "CAD", "AUD")
            _conversionResult.value = "Error de conexión: Por favor verifica tu conexión a internet"
        } catch (e: Exception) {
            _availableCurrencies.value = listOf("USD", "EUR", "GBP", "JPY", "MXN", "CAD", "AUD")
            _conversionResult.value = "Error: ${e.message ?: "Error desconocido"}"
        }
    }
}
```

Referencia a la tercera captura de pantalla conversión de divisas

```
    /**
     * Realiza la conversión de divisas utilizando la API.
     *
     * @param amount Cantidad a convertir
     * @param fromCurrency Código de la moneda de origen
     * @param toCurrency Código de la moneda de destino
     */
    fun convertCurrency(amount: Double, fromCurrency: String, toCurrency: String) {
        viewModelScope.launch {
            _isLoading.value = true
            try {
                val response = api.getExchangeRates(fromCurrency)
                val rate = response.conversion_rates[toCurrency] ?: throw Exception("Tasa de cambio no encontrada")
                val result = amount * rate
                _conversionResult.value = String.format("%.2f %s = %.2f %s", amount, fromCurrency, result, toCurrenc
            } catch (e: IOException) {
                _conversionResult.value = "Error de conexión: Por favor verifica tu conexión a internet"
            } catch (e: Exception) {
                _conversionResult.value = "Error: ${e.message ?: "Error desconocido"}"
            } finally {
                _isLoading.value = false
            }
        }
    }
}
```

Configuración de la API

```
package com.example.currencyconverter.api

import com.example.currencyconverter.data.ExchangeRateResponse
import retrofit2.http.GET
import retrofit2.http.Path

/**
 * Interface que define los endpoints para la API de ExchangeRate-API.
 * Utiliza Retrofit para realizar las llamadas HTTP.
 */
interface ExchangeRateApi {
    /**
     * Obtiene las tasas de cambio para una moneda base específica.
     *
     * @param baseCurrency Código de la moneda base (ejemplo: "USD", "EUR")
     * @return ExchangeRateResponse con las tasas de cambio para todas las monedas disponibles
     */
    @GET("v6/26410a8913d0dc1fc254ba46/latest/{base}")
    suspend fun getExchangeRates(@Path("base") baseCurrency: String): ExchangeRateResponse
}
```

4. Orden de la aplicación.

Inicio:

- Carga de la interfaz principal
- Inicialización de componentes
- Carga de monedas disponibles

4.2 Interacción del Usuario:

- Ingreso de cantidad
- Selección de monedas
- Activación de conversión

4.3 Proceso de Conversión:

- Validación de datos
- Llamada a la API
- Procesamiento de respuesta
- Visualización de resultados

5. NOTAS TÉCNICAS

5.1 Tecnologías Utilizadas:

- Kotlin como lenguaje principal
- MVVM (Model-View-ViewModel)
- Retrofit para llamadas API
- API Key: 26410a8913d0dc1fc254ba46

5.2 Requisitos:

- Android Studio

- Versión mínima de Android: 7.0 (API 24)

- Conexión a internet

- Permisos requeridos: INTERNET, ACCESS_NETWORK_STATE

CONCLUSIÓN

Al desarrollar esta aplicación de conversión de divisas aprendí diferentes cosas las cuales me ayudaran en mis proyectos futuros comenzando con el desarrollo Android lo que aprendí fue, dividir por packages, manejo de APIs usando retrofit, una interfaz de usuario que se actualice a tiempo, manejar errores, dar acceso a internet, ver lo funcional que son las APIs y lo importante de conocerlas

Este proyecto no solo me ha permitido aplicar los conocimientos teóricos que aprendimos en clase si no que ahora estamos en un caso práctico, también me ha dado una visión real de cómo se desarrolla una aplicación profesional para Android.