UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERA MECANICA ELECTRICA, ELECTRONICA Y DE SISTEMAS.

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

INFORME DE PRACTICA DE LABORATORIO IV

ESTUDIANTE:

APAZA RAMIREZ JIMMY EDSON

ASIGNATURA:

DESARROLLO BASADO DEN PLATAFORMAS II

CICLO: “IV” **GRUPO: “C”**

DOCENTE:

ING. RUELAS ACERO DONIA ALIZANDRA

PUNO - PERU

2024

Contenido del informe

I. Título

Implementación de una interfaz de usuario utilizando Jetpack Compose en Android

II. Objetivo

El objetivo de este código es crear una interfaz de usuario básica en Android usando el framework Jetpack Compose. El diseño incluye la disposición de varios elementos de texto organizados en un Column y un Row, aplicando estilos visuales como colores de fondo y alineación de los textos.

III. Librerías utilizadas

androidx.activity.ComponentActivity: Permite crear una actividad que soporta Jetpack Compose.

androidx.compose.foundation.layout.*: Contiene componentes como Column, Row, y Box para crear diseños visuales.

androidx.compose.foundation.background: Proporciona la capacidad de establecer colores de fondo en componentes.

androidx.compose.material3.*: Proporciona componentes de diseño y temas como Scaffold y Surface, adaptados a Material Design 3.

androidx.compose.runtime.Composable: Permite la creación de funciones composables, que son bloques de código reutilizables para construir la interfaz de usuario.

androidx.compose.ui.Alignment: Facilita la alineación de los elementos dentro de un contenedor.

androidx.compose.ui.Modifier: Proporciona modificadores para personalizar el comportamiento de los componentes visuales.

androidx.compose.ui.graphics.Color: Define los colores a utilizar en la interfaz.

androidx.compose.ui.tooling.preview.Preview: Permite previsualizar la interfaz de usuario en tiempo de desarrollo.

androidx.compose.ui.unit.dp y androidx.compose.ui.unit.sp: Facilitan la definición de dimensiones como el tamaño de los textos y los márgenes.

IV. Explicación básica del Código

El código define una actividad principal MainActivity que utiliza Jetpack Compose para construir una interfaz de usuario. La actividad invoca una función composable MyComplexLayout, que organiza varios componentes de interfaz (como texto y contenedores de color) en un diseño jerárquico.

Dentro de MyComplexLayout se utilizan componentes Column, Row, y Box para crear diferentes secciones de la pantalla:

Primera sección: Un Box que ocupa todo el ancho de la pantalla y se alinea en el centro, con un fondo de color cian y un texto "Ejemplo 1".

Segunda sección: Una fila (Row) con dos cajas (boxes), cada una ocupando la mitad del ancho de la pantalla, con fondos de color amarillo y verde, mostrando los textos "Ejemplo 2" y "Ejemplo 3".

Tercera sección: Otro Box que ocupa todo el ancho y muestra un texto "Ejemplo 4" en la parte inferior, con un fondo de color magenta.

La vista se distribuye equitativamente en tres secciones verticales, gracias al uso de weight(1f), que asegura que cada sección ocupe un tercio de la altura disponible.

V. Algoritmo utilizado

El código no implementa un algoritmo propiamente dicho, sino que organiza visualmente componentes de UI (interfaz de usuario) utilizando un diseño basado en columnas y filas. Cada componente se adapta automáticamente a los espacios asignados mediante el uso de modificadores de Jetpack Compose como fillMaxSize(), fillMaxWidth(), weight(), y background().

VI. Caso de prueba

Codigo a ejecutar:

```
package com.example.practica4

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                Surface(
                    modifier = Modifier
                        .fillMaxSize()
                        .padding(innerPadding),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MyComplexLayout()
                }
            }
        }
    }
}

@Composable
fun MyComplexLayout() {
    Column(
        modifier = Modifier.fillMaxSize()
    ) {
        Box(
            modifier = Modifier
                .fillMaxWidth()
                .weight(1f)
                .background(Color.Cyan),
            contentAlignment = Alignment.Center
        ) {
            Text(
                text = "Ejemplo 1",
                fontSize = 24.sp,
                fontWeight = FontWeight.Bold
            )
        }
```

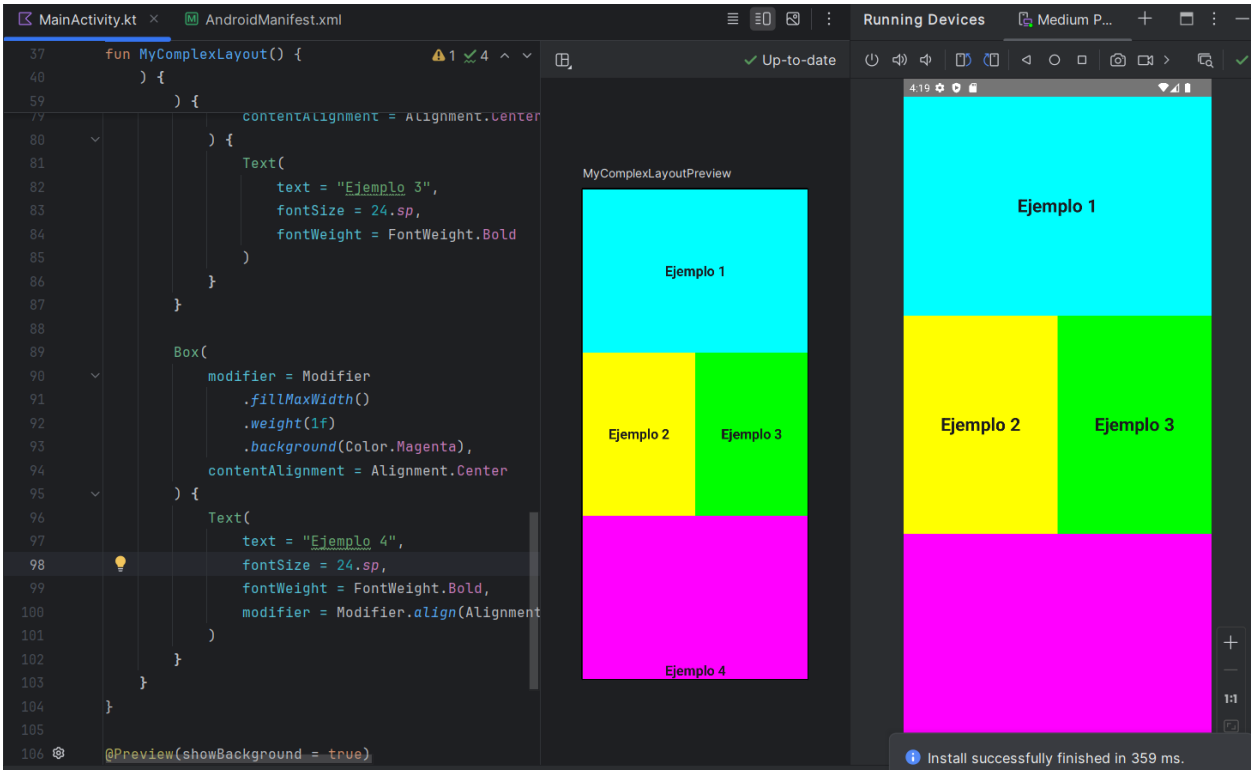
```
54
55          Row(
56              modifier = Modifier
57                  .fillMaxWidth()
58                  .weight(1f)
59          ) {
60              Box(
61                  modifier = Modifier
62                      .weight(1f)
63                      .fillMaxHeight()
64                      .background(Color.Yellow),
65                  contentAlignment = Alignment.Center
66              ) {
67                  Text(
68                      text = "Ejemplo 2",
69                      fontSize = 24.sp,
70                      fontWeight = FontWeight.Bold
71                  )
72              }
73
74              Box(
75                  modifier = Modifier
76                      .weight(1f)
77                      .fillMaxHeight()
78                      .background(Color.Green),
79                  contentAlignment = Alignment.Center
80              ) {
```

```
81                  Text(
82                      text = "Ejemplo 3",
83                      fontSize = 24.sp,
84                      fontWeight = FontWeight.Bold
85                  )
86              }
87          }
88
89          Box(
90              modifier = Modifier
91                  .fillMaxWidth()
92                  .weight(1f)
93                  .background(Color.Magenta),
94              contentAlignment = Alignment.Center
95          ) {
96              Text(
97                  text = "Ejemplo 4",
98                  fontSize = 24.sp,
99                  fontWeight = FontWeight.Bold,
100                 modifier = Modifier.align(Alignment.BottomCenter)
101             )
102         }
103     }
104 }
105
```

Ejecutado en preview:

Ejecutado en el emulador:

Codigo ejecutado en Preview y Emulador



VII. Conclusiones

El código proporciona un ejemplo sencillo pero poderoso de cómo se puede construir una interfaz de usuario en Android utilizando Jetpack Compose.

El uso de Column, Row, y Box permite organizar componentes de manera flexible y adaptable a diferentes tamaños de pantalla.

Gracias a Jetpack Compose, se puede crear una UI completamente declarativa, lo que facilita la reutilización de código y la previsualización en tiempo real.

Este enfoque también mejora la capacidad de mantenimiento del código, ya que las interfaces son más fáciles de entender y modificar en comparación con las basadas en XML tradicional en Android.