4 - Entrada de textos (TextField, OutlinedTextField y Button) - administración de estados

Las funciones TextField y OutlinedTextField permiten la entrada y modificación de texto.

Veamos con un ejemplo los parámetro mínimos que debemos configurar para tener una funcionalidad básica.

Problema 1

Permitir ingresar el nombre de usuario y su clave. Cuando se presione un botón mostrar en un Text un mensaje de error si no se ha cargado el nombre de un usuario o la clave tiene menos de 10 caracteres.

En pantalla debe aparecer una interfaz similar a:

El código fuente total es:

```
package com.tesji.compose7

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.*
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import com.tesji.compose7.ui.theme.compose7Theme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Login()
        }
    }
}

@Composable
fun Login() {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center
    ) {
        var usuario by remember { mutableStateOf("") }
        var clave by remember { mutableStateOf("") }
        var resultado by remember {mutableStateOf("Sin resultado")}
        OutlinedTextField (
            value = usuario,
            onValueChange = { usuario = it },
            label = {
                Text("Nombre de usuario")
            },
            modifier = Modifier
                .fillMaxWidth()
                .padding(10.dp),
            singleLine = true
        )
        OutlinedTextField (
            value = clave,
            onValueChange = { clave = it },
            label = {
                Text("Clave")
            },
            modifier = Modifier
                .fillMaxWidth()
                .padding(10.dp),
```

```
            singleLine = true,
            visualTransformation = PasswordVisualTransformation()
        )
        Button(
            onClick = {
                var cadena = ""
                if (clave.length < 10)
                    cadena += "La clave debe tener al menos 10 caracteres\n"
                if (usuario.length == 0)
                    cadena += "No puede dejar el usuario vacio"
                resultado = cadena
            },
            modifier = Modifier.padding(10.dp)
        ) {
            Text(text = "Confirmar")
        }
        Text(
            text = resultado,
            modifier = Modifier.padding(10.dp)
        )
    }
}
```

Para mostrar el cuadro de entrada de datos llamamos a la función composable 'OutlinedTextField' y le pasamos una serie de datos:

```
       OutlinedTextField (
           value = usuario,
           onValueChange = { usuario = it },
           label = {
               Text("Nombre de usuario")
           },
           modifier = Modifier
               .fillMaxWidth()
               .padding(10.dp),
           singleLine = true
       )
```

Los dos parámetros fundamentales de la función OutlinedTextField son el value que le pasamos una variable previamente creada llamado a la función remember:

```
       var usuario by remember { mutableStateOf("") }
```

La variable 'usuario' almacena los datos ingresados por teclado por el usuario, los mismos se actualizan según la función lambda enviada al parámetro onValueChange.

Los dos parámetros: 'value' y 'onValueChange' debemos especificarlos para que funcione nuestro editor de línea, los demás parámetros como 'label', 'modifier', 'singleLine' etc. son opcionales.

De forma similar trabajamos con la entrada de la clave, primero definimos la variable a partir de la llamada a la función remember:

```
       var clave by remember { mutableStateOf("") }
```

Y luego llamamos a la función 'OutlinedTextField':

```
       OutlinedTextField (
           value = clave,
           onValueChange = { clave = it },
           label = {
               Text("Clave")
           },
           modifier = Modifier
               .fillMaxWidth()
               .padding(10.dp),
           singleLine = true,
           visualTransformation = PasswordVisualTransformation()
       )
```

La llamada a la función composable 'Button' requiere como mínimo especificar el parámetro onClick con la función lambda que implementará la lógica de validación del ingreso de datos:

```
        Button(
            onClick = {
                var cadena = ""
                if (clave.length < 10)
                    cadena += "La clave debe tener al menos 10 caracteres\n"
                if (usuario.length == 0)
                    cadena += "No puede dejar el usuario vacio"
                resultado = cadena
            },
            modifier = Modifier.padding(10.dp)
        ) {
            Text(text = "Confirmar")
        }
```

Mediante la variable 'resultado', previamente definida:

```
        var resultado by remember {mutableStateOf("Sin resultado")}
```

Al asignar en el algoritmo codificado en el 'onClick':

```
          resultado = cadena
```

Se actualizar en forma automática el texto que muestra en la llamada a la función Text:

```
        Text(
            text = resultado,
            modifier = Modifier.padding(10.dp)
        )
```

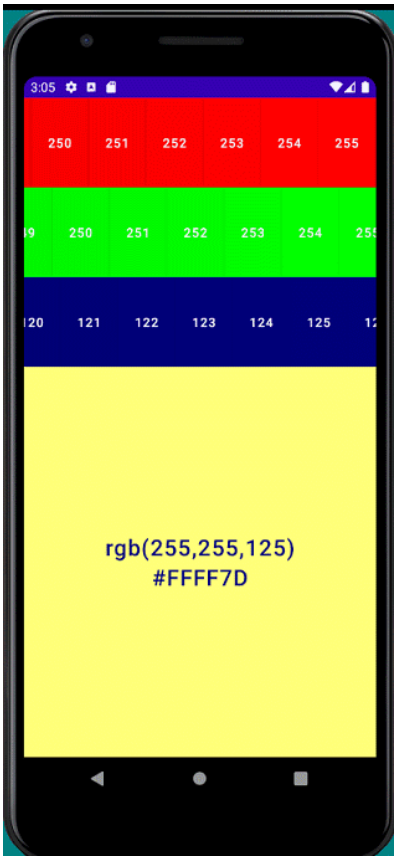
Problema 2

Disponer 3 LazyRow con 256 botones cada uno con los valores de 0 a 255, cada uno representa la cantidad de rojo, verde y azul.

En la parte inferior mostrar un botón con un color generado a partir de los valores rojo, verde y azul. Mostrar dicho valor en formato hexadecimal y decimal.

Procedemos a crear un proyecto llamado: Compose7b

En pantalla debe aparecer una interfaz similar a:



El código fuente total es:

```
package com.tesji.compose7b

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.selection.SelectionContainer
import androidx.compose.material.Button
import androidx.compose.material.ButtonDefaults
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            PantallaPrincipal()
        }
    }
}

@Composable
fun PantallaPrincipal() {
    var rojo by remember { mutableStateOf(0) }
    var verde by remember { mutableStateOf(0) }
    var azul by remember { mutableStateOf(0) }
    Column() {
        LazyRow() {
            items(256) { indice ->
                Button(
                    shape = RoundedCornerShape(0),
                    modifier = Modifier.height(100.dp),
                    onClick = {
                        rojo = indice
                    },
                    colors = ButtonDefaults.textButtonColors(
                        backgroundColor = Color(indice, 0, 0),
                    )
                )
                {
                    Text(text = "$indice",color= Color.White)
                }
            }
        }
        LazyRow() {
            items(256) { indice ->
```

```
            Button(
                shape = RoundedCornerShape(0),
                modifier = Modifier.height(100.dp),
                onClick = { verde = indice },
                colors = ButtonDefaults.textButtonColors(
                    backgroundColor = Color(0, indice, 0),
                )
            )
            {
                Text(text = "$indice",color= Color.White)
            }
        }
    }
    LazyRow() {
        items(256) { indice ->
            Button(
                shape = RoundedCornerShape(0),
                modifier = Modifier.height(100.dp),
                onClick = { azul = indice },
                colors = ButtonDefaults.textButtonColors(
                    backgroundColor = Color(0, 0, indice),
                )
            )
            {
                Text(text = "$indice",color= Color.White)
            }
        }
    }
    Button(
        shape = RoundedCornerShape(0),
        modifier = Modifier.fillMaxSize(),
        onClick = {},
        colors = ButtonDefaults.textButtonColors(
            backgroundColor = Color(rojo, verde, azul),
        )
    )
    {
        Column() {
            SelectionContainer {
                Text(
                    modifier = Modifier.fillMaxWidth(),
                    text = "rgb($rojo,$verde,$azul)", fontSize = 25.sp,
                    textAlign = TextAlign.Center,
                    color = Color(255-rojo,255-verde,255-azul)
                )
            }
            SelectionContainer() {
                Text(
                    modifier = Modifier.fillMaxWidth(),
                    text = "#${rojo.toString(16).padStart(2, '0').uppercase()}${
                        verde.toString(16).padStart(2, '0').uppercase()
                    }${azul.toString(16).padStart(2, '0').uppercase()}", fontSize = 25.sp,
                    textAlign = TextAlign.Center,
                    color = Color(255-rojo,255-verde,255-azul)
```

```
                            )
                        }
                    }
                }
            }
        }
```

Definimos tres variables de estado para almacenar el valor de rojo, verde y azul seleccionado:

```
    var rojo by remember { mutableStateOf(0) }
    var verde by remember { mutableStateOf(0) }
    var azul by remember { mutableStateOf(0) }
```

Creamos una columna:

```
    Column() {
```

Y dentro de la columna tres LazyRow, cada LazyRow crea en su interior 256 elementos de tipo Button y muestran los números de 0 a 255, además de fijar un color de fondo que varía entre el negro y el rojo más fuerte para la primera fila:

```
        LazyRow() {
            items(256) { indice ->
                Button(
                    shape = RoundedCornerShape(0),
                    modifier = Modifier.height(100.dp),
                    onClick = {
                        rojo = indice
                    },
                    colors = ButtonDefaults.textButtonColors(
                        backgroundColor = Color(indice, 0, 0),
                    )
                )
                {
                    Text(text = "$indice",color= Color.White)
                }
            }
        }
```

La segunda y tercer fila son idénticos a la primer fila:

```
        LazyRow() {
            items(256) { indice ->
                Button(
                    shape = RoundedCornerShape(0),
                    modifier = Modifier.height(100.dp),
                    onClick = { verde = indice },
                    colors = ButtonDefaults.textButtonColors(
                        backgroundColor = Color(0, indice, 0),
                    )
                )
                {
                    Text(text = "$indice",color= Color.White)
                }
            }
        }
        LazyRow() {
            items(256) { indice ->
                Button(
                    shape = RoundedCornerShape(0),
                    modifier = Modifier.height(100.dp),
                    onClick = { azul = indice },
                    colors = ButtonDefaults.textButtonColors(
                        backgroundColor = Color(0, 0, indice),
                    )
                )
                {
                    Text(text = "$indice",color= Color.White)
                }
            }
        }
```

Por último disponemos un botón para mostrar el fondo del mismo teniendo en cuenta los valores de las variables de estado:

```
        Button(
            shape = RoundedCornerShape(0),
            modifier = Modifier.fillMaxSize(),
            onClick = {},
            colors = ButtonDefaults.textButtonColors(
                backgroundColor = Color(rojo, verde, azul),
            )
        )
        {
            Column() {
                SelectionContainer {
                    Text(
                        modifier = Modifier.fillMaxWidth(),
                        text = "rgb($rojo,$verde,$azul)", fontSize = 25.sp,
                        textAlign = TextAlign.Center,
                        color = Color(255-rojo,255-verde,255-azul)
                    )
                }
                SelectionContainer() {
                    Text(
                        modifier = Modifier.fillMaxWidth(),
                        text = "#${rojo.toString(16).padStart(2, '0').uppercase()}${
                            verde.toString(16).padStart(2, '0').uppercase()
                        }${azul.toString(16).padStart(2, '0').uppercase()}", fontSize = 25.sp,
                        textAlign = TextAlign.Center,
                        color = Color(255-rojo,255-verde,255-azul)
                    )
                }
            }
        }
```