Tutorial lab 2

Camila Richter - 23183
Kevin Villagrán - 23584

Class MainActivity

```
class MainActivity : ComponentActivity() {  Camila Richter
    override fun onCreate(savedInstanceState: Bundle?) {  Camila Richt
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Lab2Theme {
                LemonApp() //llama a la función definida
                LemonadeText() //llama a la función para le título
            }
        }
    }
}
```

01

Lab2Theme

Clase de proyecto, llama a funciones LemonApp() y LemonadeText()

02

LemonApp()

Función que permite que se realicen las acciones solicitadas.

03

LemonadeText()

Función que muestra el título de la aplicación.

Funsión LemonApp()

Variables

01

currentStep: guarda el paso del programa en el que se encuentra.
clickCuantity: random usado en el step de exprimir el limón.
currentClicks: guarda la cantidad de clicks que da el usuario.

Surface

02

Modifier logra que la surface ocupe el máximo espacio.

When y Box

03

When: funciona como el patrón observer, cuando se encuentra en un step específico realiza operaciones específicas.
Box: permite que las imágenes se sobrepongan, la box está alineada en el centro (contentAlignment) y ocupa el máximo espacio posible (modifier)

```
@Composable  Camila Richter                                                    4  11  1
fun LemonApp() {
    var currentStep by remember { mutableStateOf( value: 1)} //declarar variable donde se guarde y recuerde la acción de
    var clickCuantity = (2 ≤ .. ≤ 4).random() //declarar variable donde se establece la cantidad de clicks para exprim
    var currentClicks by remember { mutableStateOf( value: 0)} // declarar variable conde se guarda la cantidad de click

    Surface(
        modifier = Modifier.fillMaxSize(), //ocupa toda la pantalla
        color = MaterialTheme.colorScheme.background
    ) {
        when(currentStep){ //observer, cuando se encuentra en un step específico muestra instrucciones específicas
            1 -> { // imagen del árbol de limones
                Box ( // se pueden sobreponer imágenes
                    contentAlignment = Alignment.Center, // Centra la imagen dentro de la Box
                    modifier = Modifier.fillMaxSize() // box ocupa espacio máximo
                ){
                    Image( // imagen de fondo
                        painter = painterResource(R.drawable.background1), // llama la imagen guardada para el fondo
                        contentDescription = null, //no se le da descripción al fondo
                        modifier = Modifier // modifica imagen de fondo
                            .offset(x=0.dp, y = -80.dp) // modifica posición de la imagen en la pantalla
                            .clip(RoundedCornerShape(20.dp)) // redondea las esquinas del cuadro
                            .width(260.dp) // ancho del cuadrado
                            .height(260.dp), // Altura del cuadrado
                        contentScale = ContentScale.Crop
                    )
```

```
        Text(text = stringResource(R.string.lemon_select), //llama al texto declarado en xml con la instrucción
            textAlign = TextAlign.Center, //alinea el texto en el centro
            lineHeight = 2.em, //espacio entre líneas
            fontSize = 27.sp, // tamaño de letra
            fontFamily = FontFamily.Monospace, // tipo de letra
            color = Color( color: 0xff20c4f0), // color de letra
            modifier = Modifier.offset(x=0.dp, y = 170.dp)) // ubicación del texto en la pantalla
        Spacer(modifier = Modifier.height(32.dp))
        Image(
            painter = painterResource(R.drawable.lemon_tree), //usa imagen importada
            contentDescription = stringResource(R.string.lemon_tree_content_description), //usa string de descri
            modifier = Modifier
                .offset(x=0.dp, y = -80.dp) //ubicación de la imagen
                .wrapContentSize()
                .clickable { //hace que pase al step 2
                    currentStep = 2
                }
        )
    }
}
```

Text, Spacer e Image

stringResource(R.string.lemonselect) llama al texto declarado en la xml. Los demás parámetros se usan para modificar el tamaño, color y alineación. El modifier.offset permite cambiar la ubicación de los elementos en la pantalla.
painterResource(R.drawable.lemon_tree) llama a la imagen importada y guardada en resources.
El .clickable permite que al hacer click sobre la imagen, la aplicación avance hacia el step 2

Step 2

Se utiliza nuevamente una box, la primera imagen importada es el fondo. Los parámetros usados en Text y la segunda Imagen son iguales a los de la primera para que se conserve la estética y diseño a lo largo de toda la aplicación. Solamente se llaman a imágenes y textos diferentes, los nuevos que muestra el app, específicos del step 2.

```
2 -> { // imagen del limón
    Box (// se pueden sobreponer imágenes
        contentAlignment = Alignment.Center, // Centra la imagen dentro de la Box
        modifier = Modifier.fillMaxSize() // box ocupa espacio máximo
    ){
        Image(// imagen de fondo
            painter = painterResource(R.drawable.background1),
            contentDescription = null,
            modifier = Modifier
                .offset(x=0.dp, y = -80.dp)
                .clip(RoundedCornerShape(20.dp))
                .width(260.dp) // Ancho deseado
                .height(260.dp), // Altura deseada
            contentScale = ContentScale.Crop
        )
        Text(text = stringResource(R.string.lemon_squeeze),
            textAlign = TextAlign.Center,
            lineHeight = 2.em,
            fontSize = 27.sp,
            fontFamily = FontFamily.Monospace,
            color = Color( color: 0xff20c4f0),
            modifier = Modifier.offset(x=0.dp, y = 170.dp)) //usa string declarada en xml
        Spacer(modifier = Modifier.height(32.dp))
```

```
            Image(
                painter = painterResource(R.drawable.lemon_squeeze), //usa imagen importada
                contentDescription = stringResource(R.string.lemon_content_description), //usa string de descri
                modifier = Modifier
                    .offset(x=0.dp, y = -80.dp)
                    .wrapContentSize()
                    .clickable { //hace que pase al step 3
                        println(clickCuantity) //imprime en logcat cuantos clicks fueron seleccionados
                        currentClicks++ // contador de la cantidad de clicks que ya dio el usuario
                        if (clickCuantity == currentClicks) { // si la cantidad asignada por el random es igual
                            currentStep = 3
                        }
                    }
            )
        }
    }
```

Image (step 2)

Esta imagen tiene código extra en su programación porque se debe hacer click varias veces sobre el limón para poder exprimirlo.

Modifier.clickable

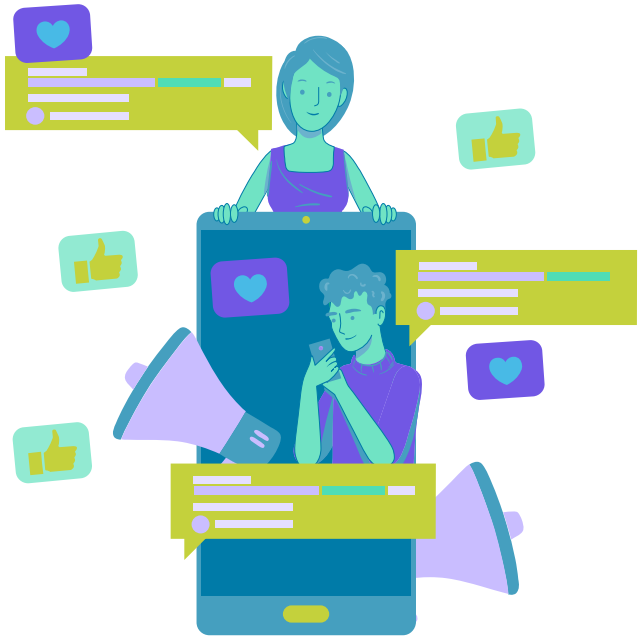
Permite que al dar click sobre la imagen pase al step 3, siempre y cuando cumpla con ls condiciones especificadas (en este caso la cantidad de clicks)

currentClicks++

Contador para agregar 1 a la cantidad de clicks cada vez que el usuario realice esta acción. De esta manera permite hacer la comparación y pasar al step 3.

Comparación

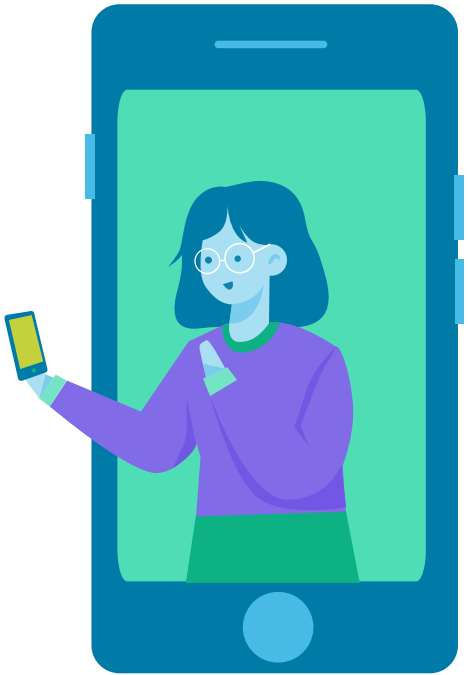
Si la cantidad de clicks dados por el usuario es igual a la cantidad de clicks determinados por el random (entre 2 y 4), permite continuar al step 3.



```
3 -> { // imagen del vaso de limonada
    Box (// se pueden sobreponer imágenes
        contentAlignment = Alignment.Center, // Centra la imagen dentro de la Box
        modifier = Modifier.fillMaxSize() // box ocupa espacio máximo
    ){
        Image( // imagen de fondo
            painter = painterResource(R.drawable.background1),
            contentDescription = null,
            modifier = Modifier
                .offset(x=0.dp, y = -80.dp)
                .clip(RoundedCornerShape(20.dp))
                .width(260.dp) // Ancho deseado
                .height(260.dp), // Altura deseada
            contentScale = ContentScale.Crop
        )
        Text(text = stringResource(R.string.lemonade_drink),
            textAlign = TextAlign.Center,
            lineHeight = 2.em,
            fontSize = 27.sp,
            fontFamily = FontFamily.Monospace,
            color = Color( color: 0xff20c4f0),
            modifier = Modifier.offset(x=0.dp, y = 170.dp)) //usa string declarada en xml
        Spacer(modifier = Modifier.height(32.dp))
        Image(
            painter = painterResource(R.drawable.lemon_drink), //usa imagen importada
            contentDescription = stringResource(R.string.glass_content_description), //usa string de desc
            modifier = Modifier
                .offset(x=0.dp, y = -80.dp)
                .wrapContentSize()
                .clickable { //hace que pase al step 4
                    currentStep = 4
                }
```

En este step de la aplicación se usan exactamente las mismas funciones y parámetros que en el step 1 y 2, nuevamente cambia en modifier.clickable ya que no se necesita ninguna condición adicional para que se pase al 4 step.

```
4 -> { // imagen del vaso vacío
    Box (// se pueden sobreponer imágenes
        contentAlignment = Alignment.Center, // Centra la imagen dentro de la Box
        modifier = Modifier.fillMaxSize() // box ocupa espacio máximo
    ){
        Image( // imagen de fondo
            painter = painterResource(R.drawable.background1),
            contentDescription = null,
            modifier = Modifier
                .offset(x=0.dp, y = -80.dp)
                .clip(RoundedCornerShape(20.dp))
                .width(260.dp) // Ancho deseado
                .height(260.dp), // Altura deseada
            contentScale = ContentScale.Crop
        )
        Text(text = stringResource(R.string.glass_restart),
            textAlign = TextAlign.Center,
            lineHeight = 2.em,
            fontSize = 27.sp,
            fontFamily = FontFamily.Monospace,
            color = Color( color: 0xff20c4f0),
            modifier = Modifier.offset(x=0.dp, y = 170.dp)) //usa string declarada en xml
        Spacer(modifier = Modifier.height(32.dp))
```



En este step de la aplicación se usan exactamente las mismas funciones y parámetros que en el step 1, 2 y 3, nuevamente cambia en modifier.clickable ya que se desea que el siguiente step sea el 1, ya que se reinicia la aplicación.

Función LemonadeText

```
@Composable  Camila Richter
fun LemonadeText() {
    Column (
        horizontalAlignment = Alignment.CenterHorizontally,
        modifier = Modifier.fillMaxSize()
    ){
        Text(text = stringResource(R.string.title),
            modifier = Modifier.offset(x=0.dp, y = 50.dp),
            textAlign = TextAlign.Center,
            color = Color( color: 0xffffc614),
            fontWeight = FontWeight.Black,
            lineHeight = 4.em,
            fontSize = 38.sp,
            fontFamily = FontFamily.Monospace,
            letterSpacing = 3.sp)
    }
}
```

Column

A diferencia del box, permite que un elemento esté arriba o debajo de otro. La columna ocupa toda la pantalla (Modifier.fillMaxSize()) y está centrada (Alignment.CenterHorizontally).

Text

Se llama al texto declarado previamente en el xml, de la misma manera que se realizó con las instrucciones de los steps, el offset sirve para cambiar la ubicación del texto en la pantalla y los demás parámetros modifican el tamaño, color y tipo de letra del texto.

Gracias!