

**EVALUACIÓN PARCIAL**  
**DESARROLLO DE APLICACIONES PARA ANDROID**  
**2024-10**

APELLIDOS Y NOMBRES DEL ESTUDIANTE:	CORREO ELECTRÓNICO:
Puchuri Galindo Aldo lois	Aldo_lois@hotmail.com

**Deberás leer detenidamente cada una de las indicaciones de la evaluación con la finalidad de cumplir con todos los puntos solicitados.**

**INSTRUCCIONES GENERALES:**

- Esta es una actividad individual.
- Si tuvieras consultas con respecto a lo solicitado en uno o varios puntos, deberás comunicarte oportunamente con tu docente para que la inquietud sea aclarada en un plazo prudente y puedas cumplir con los plazos de entrega de la actividad.
- Culminada la evaluación, deberás subir el archivo guardándolo con tu NRC, apellido y nombre.
- Es responsabilidad exclusiva del estudiante subir adecuadamente el documento solicitado corroborando que sea el correcto y que se haya cargado sin errores a la plataforma ISIL+.
- LAS EVALUACIONES ENTREGADAS FUERA DEL PLAZO ESTABLECIDO SERAN EVALUADAS CON 5 PUNTOS MENOS (plazo máximo de extemporaneidad 1 día), LUEGO DE ELLO YA NO SERAN REVISADAS.
- Colocar las imágenes y contenido de las líneas de programación de su proyecto en formato Word o PDF (las vistas que le han correspondido realizar).

**CONSIDERACIONES DEL ENTREGABLE**

- La actividad debe estar ordenada en cuanto a forma y fondo.
- Si se van a incluir imágenes de referencia en la actividad, debes revisar que estén colocadas de manera ordenada y alineada al texto. No colocar imágenes de mucho peso o gran tamaño.
- La actividad debe mostrar los puntos solicitados en el mismo orden en el que se han solicitado.
- Las fuentes de información utilizadas deben ser citadas utilizando las normas APA.

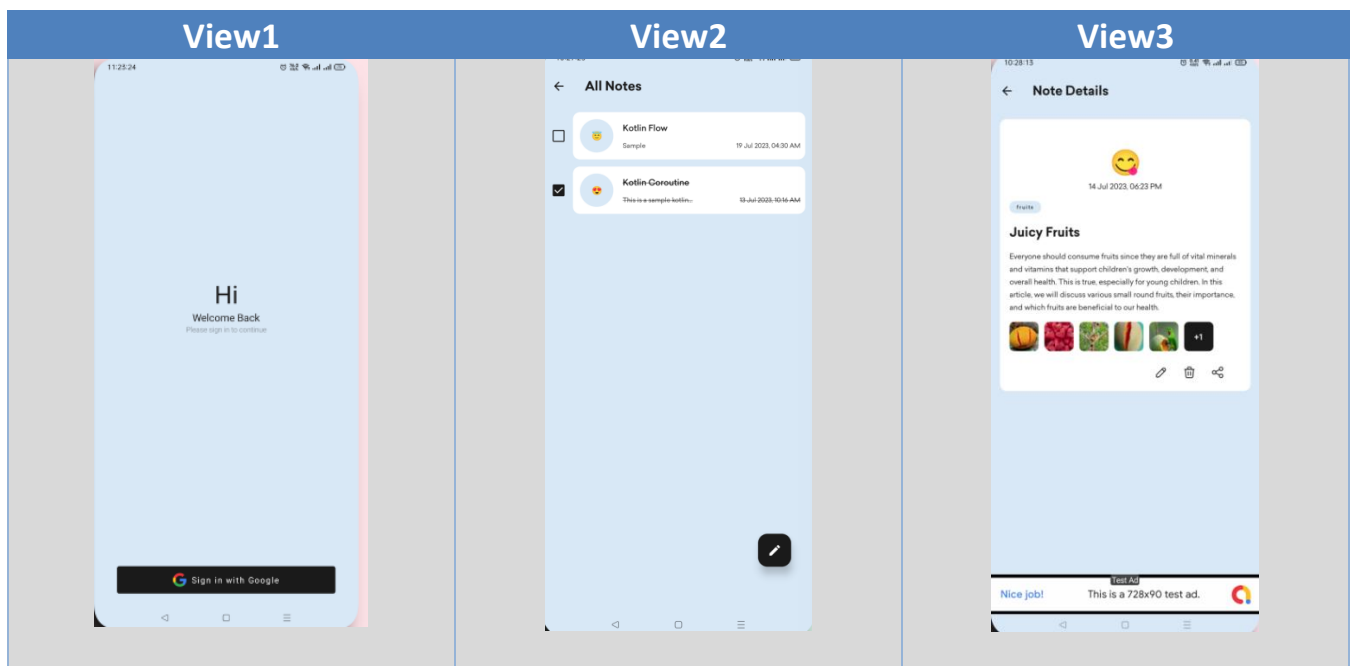
**CONTENIDO DE LA EVALUACIÓN:**

**1. CRITERIOS DE EVALUACIÓN**

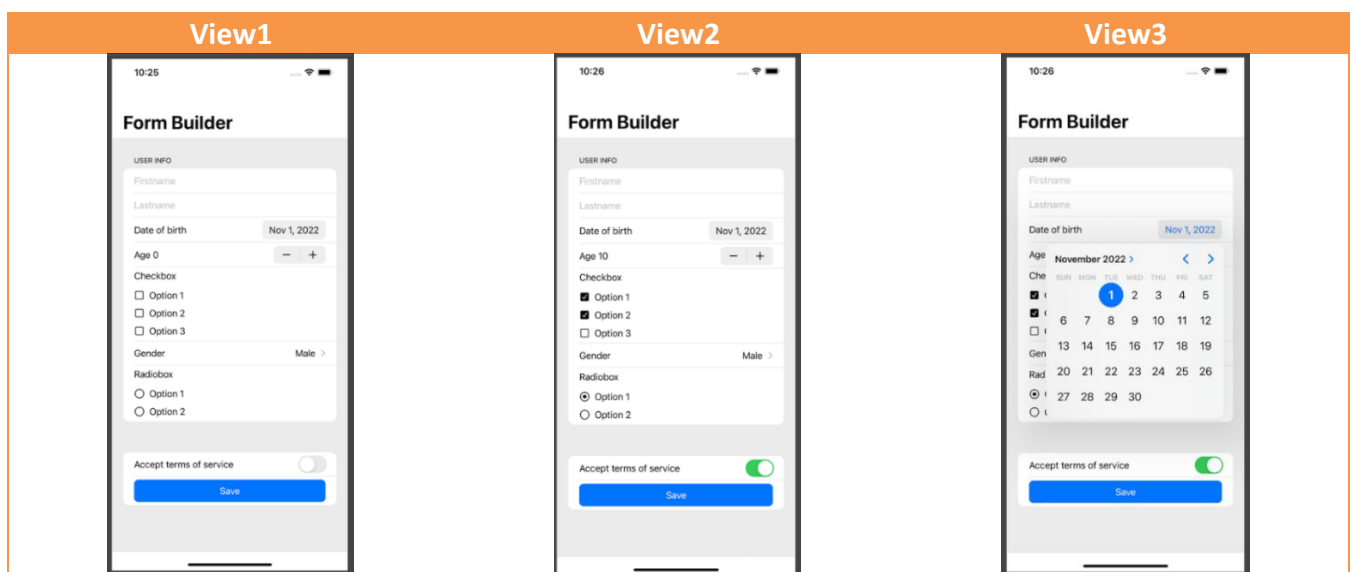
CRITERIOS	PUNTAJE
Plazo de entrega y hora de entrega (si lo realiza en la fecha y hora indicada)	5
Limpieza y calidad del contenido	12
Calidad en la presentación de las imágenes y contenidos del proyecto	3
<b>TOTAL</b>	<b>20</b>

## I. PREGUNTAS

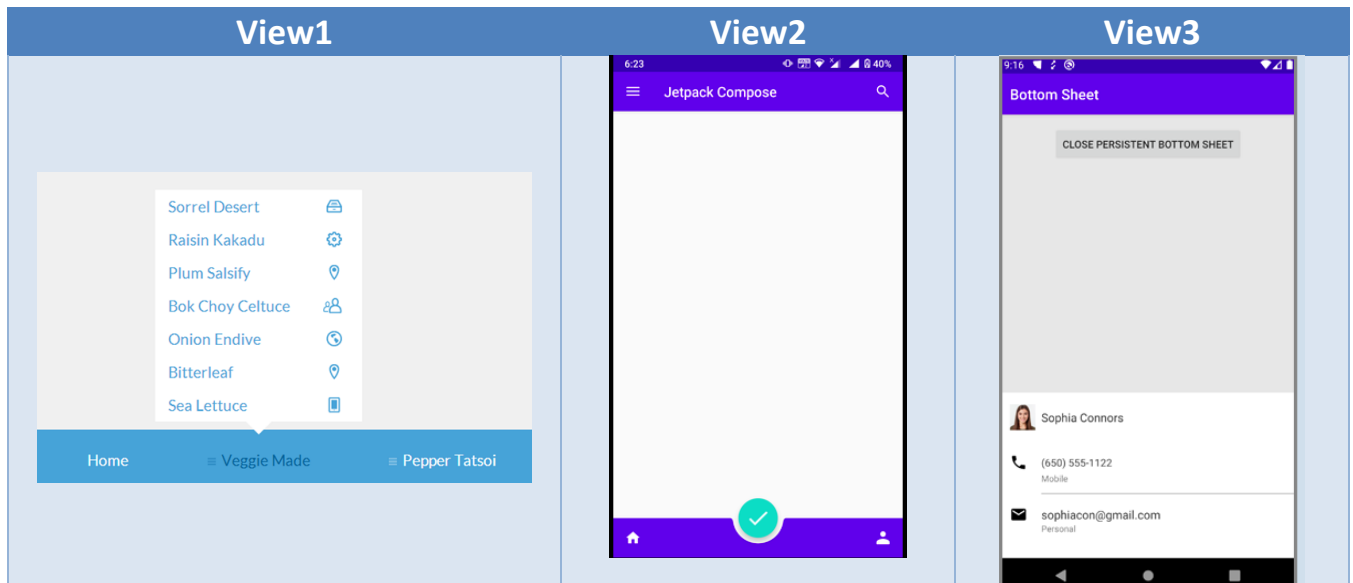
1. Desarrollar el siguiente proyecto (Utilizar todas las técnicas explicadas en aula) las imágenes son referenciales., los colores deben mantenerse o lo más cercano posible alas vistas de muestra
  - a. La vista del mainActivity debe invocar solo al navegación principal (debe estar limpia de funciones)
  - b. View1 vista de inicio (HI es un botón que te lleva al View2 y el boton en la parte inferior d el avista te lleva a la vista 3)
  - c. View2 el icono de navigation retorna al View1
  - d. View3 el icono de navigation retorna al View1



2. Desarrollar el siguiente proyecto (Utilizar todas las técnicas explicadas en aula)



## 3. Desarrollar las vistas de los diseños mostrados



```
@Composable
fun MainView(navController: NavHostController) {
    Box(contentAlignment = Alignment.BottomCenter, modifier=Modifier.fillMaxSize()) {
        Box(modifier = Modifier
            /* .background(Color.Red) */
            .fillMaxWidth()
            .height(500.dp)) {
            Column() {
                Boton1Main(navController)
                Spacer(modifier = Modifier.size(290.dp))
                Boton2Main(navController)
            }
        }
    }
}

@Composable
fun Boton1Main(navController: NavController) {
    val black= Color.Black
    Box(contentAlignment = Alignment.Center, modifier = Modifier
        .fillMaxWidth()
        .height(110.dp)) {
        Button(onClick = {navController.navigate("primer")}, colors =
            ButtonDefaults.buttonColors(Pink50) ) {
            Column(horizontalAlignment = Alignment.CenterHorizontally ) {
```

```

        Text("Hi", fontSize = 30.sp, fontWeight = FontWeight.Bold, color =
black)
        Text("Welcome back", fontSize = 20.sp,fontWeight = FontWeight.Bold
,color=black)
        Text("Please sing in to continue", fontSize = 16.sp,color=black)
    }

    }

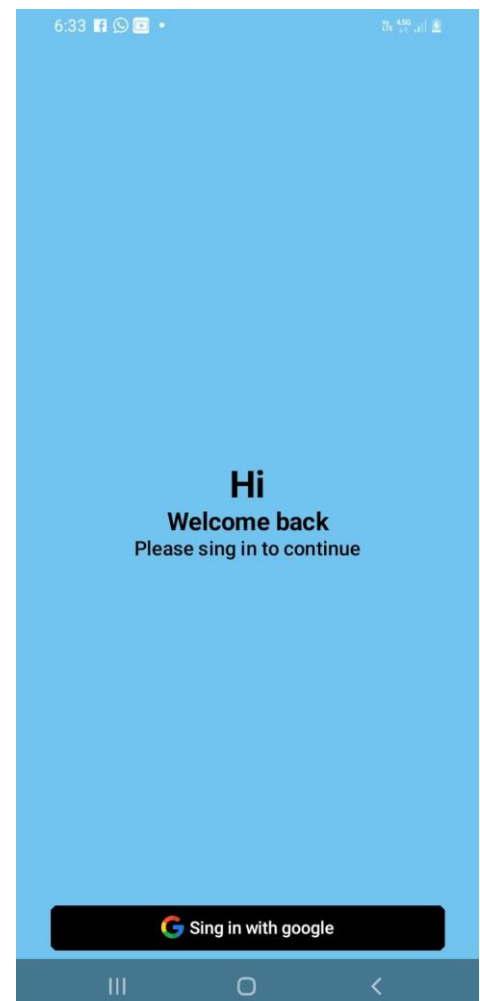
}

@Composable
fun Boton2Main(navController: NavController){
    Box(contentAlignment = Alignment.Center,modifier= Modifier
        /*.background(Color.Blue)*/
        .fillMaxWidth()
    ){
        Button(onClick = { navController.navigate("primer") }
,modifier=Modifier.width(350.dp)
        ,colors = ButtonDefaults.buttonColors(Color.Black)
        , shape = CutCornerShape(4.dp)) {
            Row{
                Image(painter = painterResource(id = R.drawable.google)
                    , contentDescription ="google"
                    , contentScale = ContentScale.Crop,
                    modifier=Modifier.size(20.dp))
                Spacer(modifier =Modifier.size(5.dp))

                Text("Sing in with google", color =
Color.White)
            }
        }
    }

}

```



```

@Composable
fun PrimerView(navController: NavController) {
    Scaffold (
        topBar = { botonTop(navController) },
        content = {paddingInterno->
            Surface(modifier=Modifier.padding(paddingInterno).fillMaxSize(),
                color=Pink50) {
                contenido()
            }
        },
        floatingActionButton = {botonbar(navController)
    }
)

}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun botonTop(navController: NavController) {
    TopAppBar(
        colors = TopAppBarDefaults.topAppBarColors(
            containerColor = Pink50

        ),
        title = {
            Row(
                modifier = Modifier.fillMaxWidth(),
                verticalAlignment = Alignment.CenterVertically,
            ) {
                IconButton(onClick = { navController.navigate("main") }) {
                    Icon(imageVector = Icons.Default.ArrowBack, contentDescription
= "Volver")
                }
                Text("All Notes"
                    , fontSize = 14.sp
                    , fontWeight = FontWeight.Bold
                    ,color= Color.White
                )
                Spacer(Modifier.size(48.dp))
            }
        },
    )
}

@Composable
fun contenido() {
    val checkedState=remember{ mutableStateOf(false) }

```

```

        val checkedState1=remember{ mutableStateOf(false)}
    LazyColumn(
        paddingTop= PaddingValues(10.dp)
    ) {

        item() {
            Box(contentAlignment = Alignment.Center,modifier= Modifier
                .fillMaxWidth()
                .height(110.dp)
            ){
                Row (verticalAlignment =
Alignment.CenterVertically,modifier=Modifier.fillMaxSize()){
                    Checkbox(checked = checkedState.value, onCheckedChange =
{checkedState.value=it}, colors = CheckboxDefaults.colors( // Cambiar el color del
cuadrito del checkbox
                        checkmarkColor = Color.Red, disabledCheckedColor =
Color.Blue)
                ) // Grosor y color del borde del checkbox)
                Box(contentAlignment = Alignment.Center,modifier= Modifier
                    .height(100.dp)
                    .fillMaxWidth()
                    .clip(RoundedCornerShape(20.dp))
                    .background(Color.White)

                    .padding(10.dp, 0.dp)){
                    Row(
                        verticalAlignment = Alignment.CenterVertically,
                        horizontalArrangement = Arrangement.SpaceBetween,
                        modifier = Modifier
                            .fillMaxWidth()
                    )
                    {
                        Box() {
                            Row {
                                AsyncImage(
                                    model =
"https://fastly.picsum.photos/id/615/200/300.jpg?hmac=ehJCfeX01-
ZbwBXgbYKroA97kTtoPKNoyEbCxnzsYfU",
                                    contentDescription = null,
                                    contentScale = ContentScale.Crop,
                                    modifier = Modifier
                                        .size(60.dp)
                                        .border(0.dp, Color.Black,
RoundedCornerShape(40.dp))
                                        .clip(RoundedCornerShape(40.dp))
                                )
                                Spacer(modifier = Modifier.size(10.dp))
                                Column() {
                                    Text(
                                        "Kotlin Flow",
                                        color = Color.Black,
                                        fontSize = 15.sp,
                                        fontWeight = FontWeight.Bold
                                    )
                                    Spacer(modifier = Modifier.height(10.dp))
                                    Text("sample", color = Color.Black, fontSize
= 14.sp,)
                                }
                            }
                        }
                    }
                }
            }
        }
    }

```

```

    }

    }

    Text("19 de Julio 2023 04:00 am", color = Color.Black,
fontSize = 8.sp,modifier=Modifier.align(Alignment.Bottom))

    }

    }

    }

    }

    item() {
        Box(contentAlignment = Alignment.Center,modifier= Modifier
            .fillMaxWidth()
            .height(110.dp)
        ){
            Row (verticalAlignment =
Alignment.CenterVertically,modifier=Modifier.fillMaxSize()){
                Checkbox(checked = checkedState1.value, onCheckedChange =
{checkedState1.value=it})
                Box(contentAlignment = Alignment.Center,modifier= Modifier
                    .height(100.dp)
                    .fillMaxWidth()
                    .clip(RoundedCornerShape(20.dp))
                    .background(Color.White)

                    .padding(10.dp, 0.dp)){
                        Row(
                            verticalAlignment = Alignment.CenterVertically,
                            horizontalArrangement = Arrangement.SpaceBetween,
                            modifier = Modifier
                                .fillMaxWidth()

                        )
                    }
                Box () {
                    Row {
                        AsyncImage(
                            model =
"https://fastly.picsum.photos/id/671/200/300.jpg?hmac=I6A0nYPaPDSVj8NEa141bRDDN6E61
7I8S8Pc3ulWHS8",

                            contentDescription = null,
                            contentScale = ContentScale.Crop,
                            modifier = Modifier
                                .size(60.dp)
                                .border(0.dp, Color.Black,
RoundedCornerShape(40.dp))

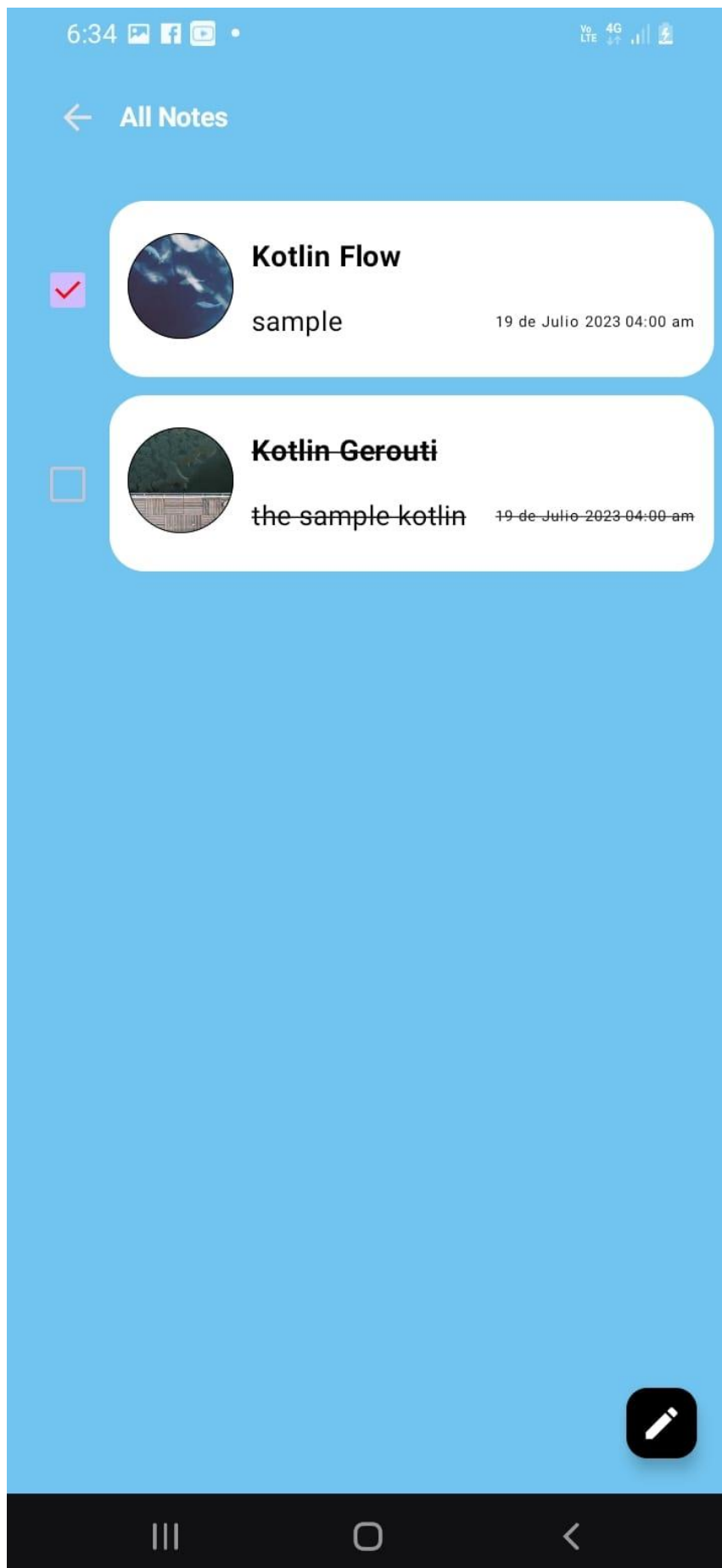
                                .clip(RoundedCornerShape(40.dp))

                        )
                        Spacer(modifier = Modifier.size(10.dp))
                    }
                }
            }
        }
    }
}

```







```

@Composable
fun PrimerView(navController: NavController) {
    Scaffold (
        topBar = { botonTop(navController) },
        content = {paddingInterno->
            Surface(modifier= Modifier
                .padding(paddingInterno)) {
                contenido2 ()
            }
        })
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun botonTop(navController: NavController) {
    TopAppBar(
        colors = TopAppBarDefaults.topAppBarColors(
            containerColor = Color.White,
            titleContentColor= Color.Red
        ),
        title = {
            Text("Form Builder"
                , fontSize = 24.sp
                , fontWeight = FontWeight.Bold
                ,color= Color.Black
            )
        },
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun contenido2() {
    var selectGender by remember { mutableStateOf("option1") }
    var switchState by remember { mutableStateOf(false) }
    val checkedState=remember{ mutableStateOf(false)}
    Box(contentAlignment = Alignment.Center,modifier= Modifier
        .fillMaxSize()
        .height(200.dp)
        .background(Color.Gray)
        .padding(15.dp)
    ){
        Column {
            Text("USER INFO",color=Color.Black, modifier = Modifier

```

```

        .fillMaxWidth()
        .padding(start = 30.dp))
Box(modifier= Modifier
    .fillMaxWidth()
    .height(600.dp)
    .clip(RoundedCornerShape(20.dp))
    .background(Color.White)
    .padding(start = 10.dp)){

    Column {
        var Hasta by remember { mutableStateOf("") }
        var Desde by remember { mutableStateOf("") }

        TextField(value = Desde, onValueChange = { newDesde ->
if(newDesde.length<=20) Desde = newDesde },
        placeholder = { Text("Firstname") }
        , keyboardOptions = KeyboardOptions(keyboardType=
KeyboardType.Text)

        , colors = TextFieldDefaults.outlinedTextFieldColors(
            focusedTextColor =Color.Black ,
            focusedBorderColor = Color.Gray,
            unfocusedBorderColor = Color.Gray,
            cursorColor = Color.Black
        ), modifier = Modifier.fillMaxWidth())

        TextField(value = Hasta,
            onValueChange = { newHasta -> if(newHasta.length<=20)
Hasta = newHasta },
        placeholder = { Text("Lastname ") }
        , keyboardOptions = KeyboardOptions(keyboardType=
KeyboardType.Text)

        , colors = TextFieldDefaults.outlinedTextFieldColors(
            focusedTextColor =Color.Black ,
            focusedBorderColor = Color.Gray,
            unfocusedBorderColor = Color.Gray,
            cursorColor = Color.Black
        ), modifier = Modifier.fillMaxWidth())

        Row(horizontalArrangement =
Arrangement.SpaceBetween,modifier = Modifier
            .height(50.dp)
            .fillMaxWidth()
            .background(Color.Red)
            .padding(horizontal = 10.dp) ){
            Text("DatePicker", color=Color.Black)

            val state= rememberDatePickerState()
            var showDialog by remember { mutableStateOf(false) }
            Button(onClick ={showDialog=true}, shape =
RoundedCornerShape(5.dp), colors = ButtonDefaults.buttonColors( Color.Gray)) {
                val date=state.selectedDateMillis
                date?.let { val localDate=
Instant.ofEpochMilli(it) .atZone (ZoneId.of ("UTC")) .toLocalDate ()

Text ("${localDate.month}${localDate.dayOfMonth},${localDate.year}")
                }
                if(showDialog){
                    DatePickerDialog(
                        onDismissRequest = { showDialog=false },

```

```

        confirmButton = {})
    {
        DatePicker(state =state)
    }

}

}

Text("Age 10")
Column {
    Text("Checkbox")
    val items=List(3){ " ${it + 1}" }
    LazyColumn {

        items(items) { iten ->
            Row {
                Checkbox(checked = checkedState.value,
onCheckedChange = {checkedState.value=it}, colors = CheckboxDefaults.colors( //
Cambiar el color del cuadrito del checkbox
                checkmarkColor = Color.Red,
disabledCheckedColor = Color.Blue))
                Text("Option $iten")
            }
        }
    }
}

Column {
    Text("Checkbox")
    val items=List(1){ " ${it + 1}" }
    LazyColumn {

        items(items) { iten ->
            Row {
                RadioButton(
                    selected =selectGender=="Option1",
                    onClick ={selectGender="Option1 "})
                Text(text = "Option $iten")
            }
        }
    }
}

}

}

}

```

## Form Builder

USER INFO

Firstname

Lastname

DataPicker

MAY17,2024

Age 10

Checkbox






☐ Option 1

☐ Option 2

☐ Option 3

Checkbox


☐ Option 1

11:54    • VoLTE 4.5G  

## Form Builder

USER INFO

Seleccionar fecha

18 de mayo de 

mayo de 2024 ▾ < >

d	l	m	m	j	v	s
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	