ASIGNATURA: Desarrollo de App Móviles.

TEMA: Diseño de interfaz: Componentes

ACTIVIDAD DE ENSEÑANZA – APRENDIZAJE – EVALUACIÓN: Implementación de Componentes en Interfaces Graficas de usuario en Android con Kotlin Multiplataforma.

TIEMPO DE LA ACTIVIDAD DE E-A: 2 horas

OBJETIVO: Diseñar interfaces de usuario mediante la utilización de los componentes fundamentales que proporciona Compose.

ORIENTACIONES GENERALES: Seguir la siguiente guía para el diseño de Interfaces graficas de usuario con componentes de la herramienta Jetpack Compose.
La actividad será evaluada y calificada en clase.
Valor de la actividad 5%.

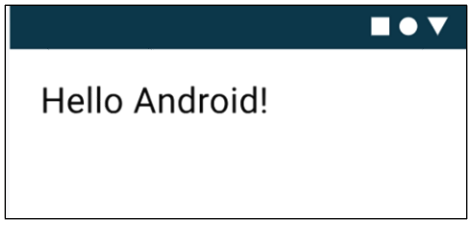
Conceptos.

Jetpack Compose. Es un moderno kit de herramientas declarativas de interfaz de usuario <<IU>> para Android. Tiene como propósito, facilitar la escritura y el mantenimiento de la *IU* de la app, proporcionando una *API declarativa* que permite renderizarla sin cambiar las vistas del frontend de manera imperativa.

Compose es un framework de IU declarativa, es decir se centra en especificar lo que se desea conseguir "resultado final".

Aspectos de funcionalidad.

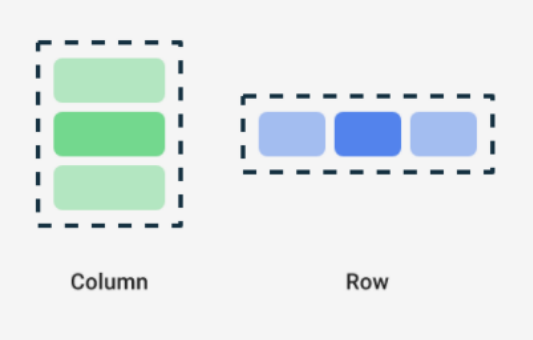
Anotación **@Composable.** Todas las funciones de componibilidad deben tener esa anotación que informa al compilador de **Compose** que está diseñada para convertir datos en IU. Por otra parte, permite que la función se utilice para construir la interfaz de usuario (IU) de una aplicación.



Profesor: Yamil Buenaños Palacios
ybuenano@usbbog.edu.co

Componentes de diseño.

Compose proporciona una colección de diseños listos para usar que ayudan a organizar los elementos de la IU y facilitan la definición de los diseños.



Ejemplo:

```
@Composable
fun Artistas () {
    Text("Marc Antony")
    Text("Grupo Niche")
}
```

- **Column** permite ubicar los elementos en sentido vertical en la pantalla.

```
@Composable
fun ArtistasColumn() {
    Column {
        Text("Marc Antony")
        Text("Grupo Niche")
    }
}
```

- **Row** permite ubicar los elementos en sentido horizontal en la pantalla.

```
@Composable
fun ArtistasRow() {
    Row {
        Text("Marc Antony")
        Text("Grupo Niche")
    }
}
```

- **Combinación Row y Column**

```
@Composable
fun ArtistFilaColumna(artist: Artist) {
    Row() {
        //Información de la Fila
        Column {
            //Información de la Columna
        }
    }
}
```

Modificadores de Compose

Los modificadores son objetos estándar de Kotlin. Para crear llama a una de las funciones de clase, por otra parte, permiten decorar o aumentar un elemento componible.

Características:

- Cambiar el tamaño, el diseño, el comportamiento y el aspecto del elemento componible
- Agregar información (p. ej., etiquetas de accesibilidad)
- Procesar entradas del usuario
- Agregar interacciones de nivel superior, (p. ej., hacer que un elemento sea apto para hacer clic, desplazable, arrastrable o ampliable)

```
@Composable
private fun Greeting(name: String) {
    Column(modifier = Modifier.padding(24.dp)) {
        Text(text = "Hello,")
        Text(text = name)
    }
}
```

```
@Composable
private fun Greeting(name: String) {
    Column(
        modifier = Modifier
            .padding(24.dp)
            .fillMaxWidth()
    ) {
        Text(text = "Hello,")
        Text(text = name)
    }
}
```

- **padding** coloca espacio alrededor de un elemento.
- **fillMaxWidth** hace que el elemento componible ocupe el ancho máximo que le otorga su elemento superior.

Ejemplo:

```
@Composable
fun ArtistasColumn() {
        Column(modifier = Modifier.padding(24.dp))  {
        Text("Marc Antony")
        Text("Grupo Niche")
    }
}
```

```
@Composable
fun ArtistasColumn2() {
        Column(
            modifier = Modifier.padding(24.dp)
            .fillMaxWidth()) {
        Text("Gilberto SantaRosa")
        Text("Choquibtown")
    }
}
```

```
@Composable
fun ArtistasRow() {
    Row(horizontalArrangement = Arrangement.SpaceBetween,
    modifier = Modifier.fillMaxWidth()) {
        Text("Celia Cruz")
        Text("Orquesta Guayacan")
    }
}
```

```
@Composable
fun ArtistasRow() {
    Row(horizontalArrangement = Arrangement.SpaceBetween,
    modifier = Modifier.padding(24.dp)) {
        Text("Celia Cruz")
        Text("Orquesta Guayacan")
    }
}
```

- **padding y size:** Petmiten establecer tamaño con el modificador.

```
@Composable
fun ArtistCard(/*...*/) {
    Row(
        modifier = Modifier.size(width = 400.dp, height = 100.dp)
    ) {
        Image(/*...*/)
        Column { /*...*/ }
    }
}
```

```
@Composable
fun ArtistCard(/*...*/) {
   Row(
      modifier = Modifier.size(width = 400.dp, height = 100.dp)
   ) {
      Image(
         /*...*/
         modifier = Modifier.requiredSize(150.dp)
      )
      Column { /*...*/ }
   }
}
```

Detalles de la estructura

- **@Composable**: Esta anotación marca la función como componible, lo que le permite crear una interfaz de usuario de manera declarativa.
- **remember:** función que garantiza que el estado se mantenga en todas las recomposiciones.
- **mutableState** se usa para representar el estado TextField.
- **value:** Vincula el valor de texto actual del estado.
- **onValueChange:** Actualiza el estado cada vez que el usuario escribe en el campo de texto.
- **it:** Es el nombre implícito del único parámetro de una lambda en Kotlin.

```
@Composable
fun  BasicTextFieldExample () {
   var texto = remember { mutableStateOf( "" ) }

   TextField(
      valor = texto.valor,
      onValueChange = { texto.valor = it },
      etiqueta = { Texto( "Ingrese su nombre" ) },
      marcador = { Texto( "John Doe" ) },
      singleLine = true ,
      modificador = Modifier
         .fillMaxWidth()

   )
}
```

- **Estilos de Texto**

  - **Color de Texto**

Color al atributo color para cambiar el color de todo el texto:

```
@Composable
fun TextColor() {
    Text("Color Cyan", color = Color.Cyan)
}
```

  - **Tamaño del Texto**

El tamaño de texto en Compose es representado por la clase TextUnit. Las unidades de medidas disponibles son: **sp**, **em** y **Unspecified** (toma valor por defecto o hereda del tema aplicado).

```
@Composable
fun TextSize() {
    Column {
        Text("Texto con 20sp", fontSize = 20.sp)
        Text("Texto con 10em", fontSize = 10.em)
    }
}
```

  - **Texto en Cursiva**

```
@Composable
fun ItalicText() {
    Text("Texto en cursiva", fontStyle = FontStyle.Italic)
}
```

- **Texto en negrilla**

```
Composable
fun FontWeightText() {
   Column {
      Text("Texto con grosor W500", fontWeight = FontWeight.W500)
      Text("Texto con grosor Extra Bold", fontWeight = FontWeight.ExtraBold)
   }
}
```

Campos de texto - Entradas de usuario

Son los campos editables que en una App permiten a los usuarios ingresar datos.

- **TextField** permite a los usuarios ingresar y modificar texto.

Ejemplo1

```
@Composable
fun SimpleTextFieldRelleno() {
   var text by remember { mutableStateOf("Hola") }

   TextField(
      value = text,
      onValueChange = { text = it },
      label = { Text("Label") }
   )
}
```

Ejemplo2

```
@Composable
fun SimpleTextField() {
   var text by remember { mutableStateOf("") }

   OutlinedTextField(
      value = text,
      onValueChange = { text = it },
      label = { Text("Nombres") }
   )
}
```

Ejemplo3

```
@Composable
fun IngrsoDatos() {
    var identificacion by remember { mutableStateOf(TextFieldValue()) }
    var nombre by remember { mutableStateOf(TextFieldValue()) }
    var apellido by remember { mutableStateOf(TextFieldValue()) }
    var dirección by remember { mutableStateOf(TextFieldValue()) }
    var email by remember { mutableStateOf(TextFieldValue()) }

    Column(modifier = Modifier.padding(16.dp)) {
        OutlinedTextField(
            value = identificacion,
            onValueChange = { identificacion = it },
            label = { Text("Identificación") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(8.dp))
        OutlinedTextField(
            value = nombre,
            onValueChange = { nombre = it },
            label = { Text("Nombres") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(8.dp))
        OutlinedTextField(
            value = apellido,
            onValueChange = { apellido = it },
            label = { Text("Apellidos") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(8.dp))
        OutlinedTextField(
            value = dirección,
            onValueChange = { dirección = it },
            label = { Text("Dirección") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(8.dp))
        OutlinedTextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth()
        )
    }
}
```

Estructura de un Botón

Ejemplo 4

```
@Composable
fun  BasicButtonExample () {
   Button(
      onClick = { /* Manejar el clic del botón */ },
      modifier = Modifier
         .padding( 16.dp )
         .fillMaxWidth(),
      shape = RoundedCornerShape( 8.dp ),
      enabled = true ,
      contentPadding = PaddingValues( 12.dp )
   ) {
      Text(text = "Haz clic en mí" )
   }
}
```

Actividad

Dar solución a los casos que se plantean en el presente documento mediante la utilización de la tecnología Kotlin Multiplatform y el Entorno de Desarrollo Android.

1. Trascribir cada uno de los ejemplos descritos en la presente guía y posteriormente ejecutarlos.

2. Construya una app con KMP, para dar solución al siguiente caso:
   Una empresa que se dedica a la fabricación de carrocerías para vehículos, con el propósito de gestionar el pago de sus empleados desea que cada empleado registre su información a través de una interfaz d usuario la cual debe contener las siguientes características: tipo de identificación, número de identificación, nombres, apellidos, teléfono, cargo, dependencia y número de horas trabajadas y un botón para generar la acción de guardar la información