

## Práctica Layout



# Jetpack Compose

# **ÍNDICE**

**1 – Organizando un perfil**

**2 – Pantalla de configuración**

**3 – Fila de productos destacados**

**4 – Espaciadores y proporciones**

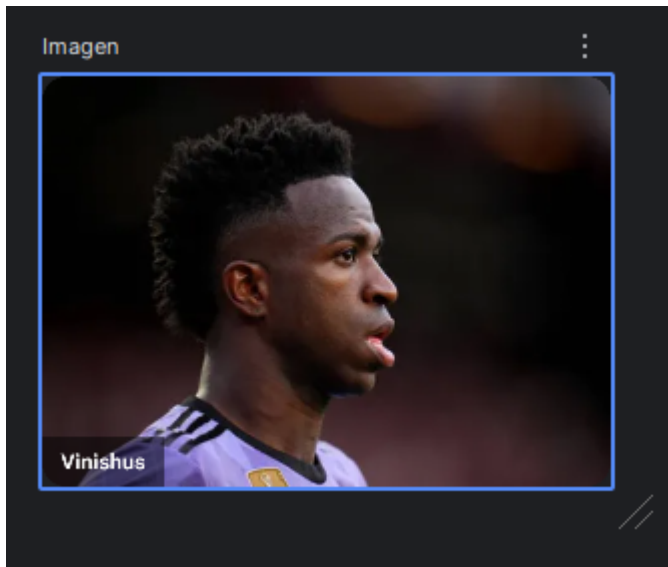
**5 – Vamos a mover los Box**

**6 – Diseño libre con Constraint Layout**

# 1 – Organizando un perfil

Crea una vista donde se muestre la foto de perfil de un usuario con su nombre y una breve descripción. Usa Box para superponer el texto sobre la imagen

```
@Preview
@Composable
fun Imagen() {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .height( height = 200.dp)
            .clip( shape = RoundedCornerShape( size = 16.dp))
    ) {
        val fotoPerfil = painterResource( id = R.drawable.fotovini)
        Image(
            painter = fotoPerfil,
            contentDescription = "Foto de perfil de Vinishus",
            modifier = Modifier.fillMaxSize(),
            contentScale = ContentScale.Crop
        )
        Text(
            text = "Vinishus",
            color = Color.White,
            fontWeight = FontWeight.SemiBold,
            modifier = Modifier
                .align(Alignment.BottomStart)
                .background(Color( color = 0x80000000))
                .padding(horizontal = 12.dp, vertical = 8.dp)
        )
    }
}
```

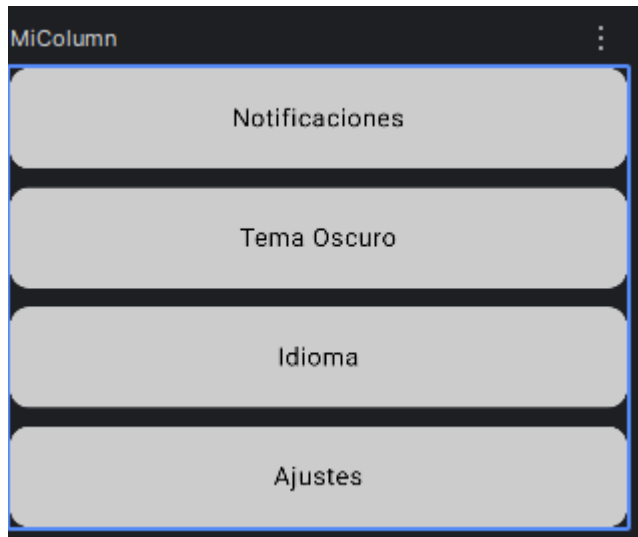


## 2 – Pantalla de configuración

Diseña una pantalla de ajustes de aplicación usando Column. Cada fila debe representar una opción de configuración (por ejemplo: notificaciones, tema oscuro, idioma...)

```
1 Usage
@Preview
@Composable
fun MiColumn() {
    Column(
        modifier = Modifier
            .fillMaxWidth(),
        verticalArrangement = Arrangement.spacedBy( space = 12.dp)
    ) {
        AjusteItem( titulo = "Notificaciones")
        AjusteItem( titulo = "Tema Oscuro")
        AjusteItem( titulo = "Idioma")
        AjusteItem( titulo = "Ajustes")
    }
}

4 Usages
@Composable
fun AjusteItem(titulo: String) {
    Box(
        modifier = Modifier
            .background(Color.LightGray, shape = RoundedCornerShape( size = 12.dp))
            .fillMaxWidth()
            .height( height = 64.dp)
    ) {
        Text(
            text = titulo,
            modifier = Modifier.align(Alignment.Center),
            style = MaterialTheme.typography.bodyLarge
        )
    }
}
```



En este ejercicio he usado 2 funciones, una es la principal que es 'MiColumn' y otra función de apoyo que es Ajusteltem, que básicamente sirve para no repetir código por cada box que quería introducir

### 3 – Fila de productos destacados

Diseña una pequeña sección horizontal dentro de una app de tienda online llamada “Productos destacados”. En esta fila deben aparecer tres productos uno al lado del otro, mostrando:

- Una imagen del producto.
- Su nombre.
- Y un precio simulado.

```

@Composable
fun ProductosDestacados() {
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 8.dp)
            .background(Color.Magenta)
    ) {
        Text(
            text = "Productos destacados",
            style = MaterialTheme.typography.titleMedium
        )
        Spacer(modifier = Modifier.height(12.dp))

        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.spacedBy(12.dp)
        ) {
            ProductoItemPequenio(
                nombre = "Pedri",
                precio = 140_000_000.0,
                imagenRes = R.drawable.pedri,
                modifier = Modifier.weight(1f)
            )
            ProductoItemPequenio(
                nombre = "Santi Cazorla",
                precio = 200_000.0,
                imagenRes = R.drawable.cazorla,
                modifier = Modifier.weight(1f)
            )
            ProductoItemPequenio(
                nombre = "Lusi Milla",
                precio = 4_000_000.0,
                imagenRes = R.drawable.milla,
                modifier = Modifier.weight(1f)
            )
        }
    }
}

```

```

@Composable
fun ProductoItemPequenio(nombre: String, precio: Double, imagenRes: Int,
    modifier: Modifier = Modifier) {

    val formatoEuro = NumberFormat.getCurrencyInstance(Locale(language = "es", country = "es"))

    Card(
        modifier = modifier,
        elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
        shape = RoundedCornerShape(size = 14.dp)
    ) {
        Column(
            modifier = Modifier.padding(all = 10.dp),
            verticalArrangement = Arrangement.spacedBy(space = 8.dp)
        ) {
            Image(
                painter = painterResource(id = imagenRes),
                contentDescription = "Imagen de $nombre",
                modifier = Modifier
                    .fillMaxWidth()
                    .aspectRatio(ratio = 1f)
                    .clip(shape = RoundedCornerShape(size = 10.dp)),
                contentScale = ContentScale.Crop
            )
            Text(
                text = nombre,
                style = MaterialTheme.typography.bodyMedium,
                maxLines = 1,
                overflow = TextOverflow.Ellipsis
            )
            Text(
                text = formatoEuro.format(number = precio),
                style = MaterialTheme.typography.bodyMedium.copy(fontWeight = FontWeight.SemiBold)
            )
        }
    }
}

```

El composable ProductosDestacados() crea una sección dentro de la app que muestra un título ("Productos destacados") y una fila de tarjetas con tres productos. Cada tarjeta incluye una imagen, el nombre y el precio formateado en euros.

ProductosDestacados() se encarga del diseño general y ProductoItemPequenio() se encarga específicamente de cada una de las tarjetas



## 4 – Espaciadores y proporciones

Crea una pequeña pantalla de inicio que contenga un título, un subtítulo y un botón de “Continuar”. Usa `Spacer` para distribuir los elementos verticalmente, de forma que el botón quede al final de la pantalla sin necesidad de forzar tamaños.

```
@Preview
@Composable
fun PantallaInicio() {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(all = 24.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {

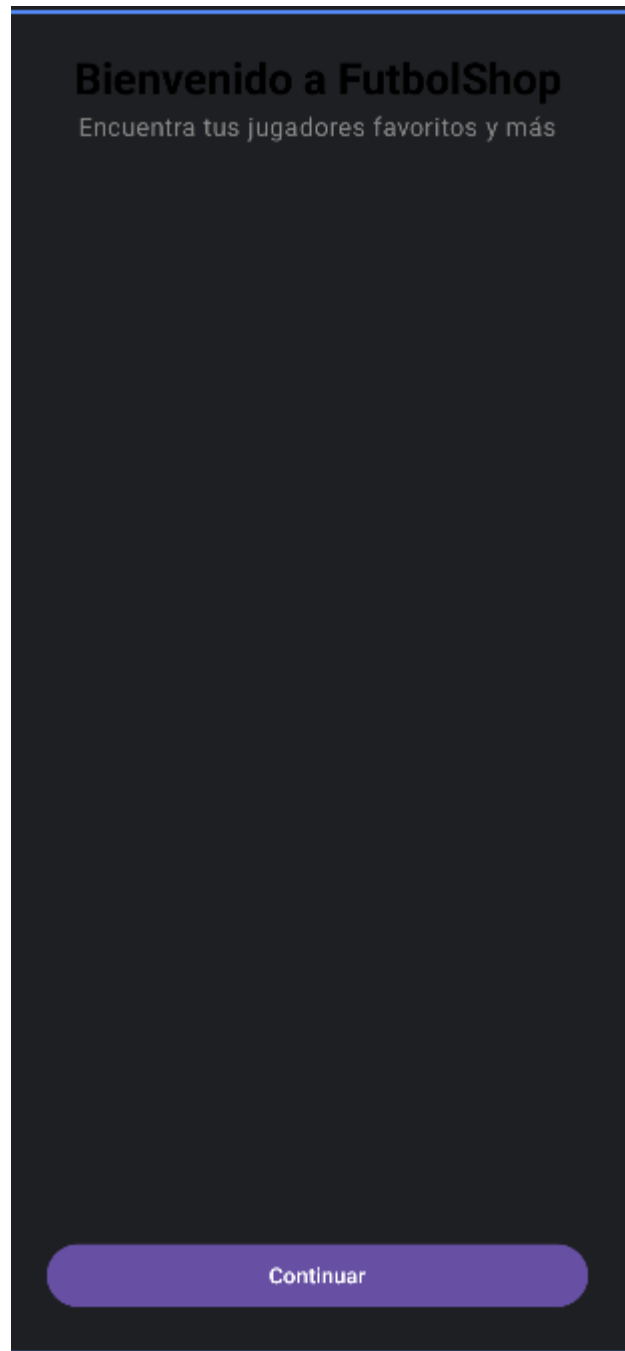
        Text(
            text = "Bienvenido a FutbolShop",
            style = MaterialTheme.typography.headlineMedium,
            fontWeight = FontWeight.Bold
        )

        Text(
            text = "Encuentra tus jugadores favoritos y más",
            style = MaterialTheme.typography.bodyLarge,
            color = Color.Gray
        )

        Spacer(modifier = Modifier.weight(weight = 1f))

        Button(
            onClick = { },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text(text = "Continuar")
        }
    }
}
```





## 5 – Vamos a mover los Box

Posiciones básicas: Mueve las cajas (Box) para conseguir las siguientes configuraciones:

- Todas alineadas arriba.

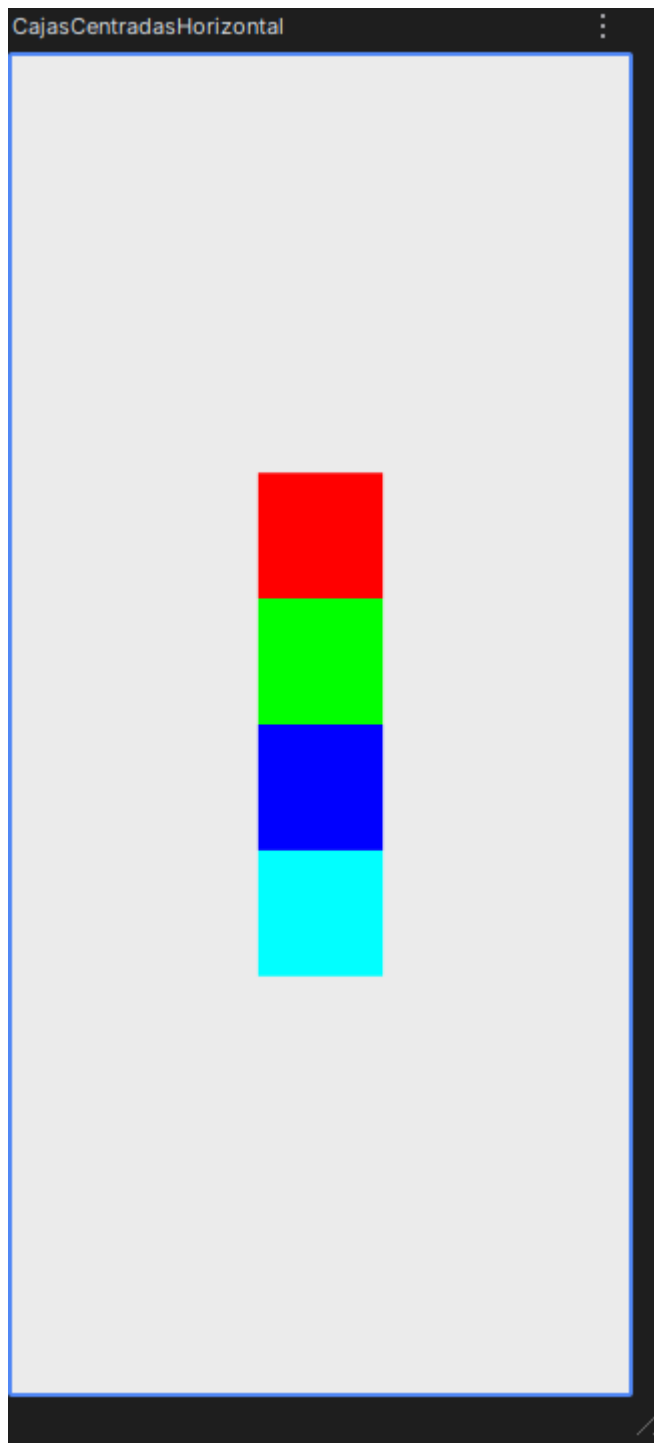
```
@Preview
@Composable
fun CajasAlineadasArriba() {
    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color(color = 0xFFE0E0))
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(all = 16.dp),
            horizontalAlignment = Alignment.Start
        ) {
            Row(modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.Start,
                verticalAlignment = Alignment.Top){

                Box(
                    modifier = Modifier
                        .size(size = 80.dp)
                        .background(Color.Red)
                )
                Box(
                    modifier = Modifier
                        .size(size = 80.dp)
                        .background(Color.Green)
                )
                Box(
                    modifier = Modifier
                        .size(size = 80.dp)
                        .background(Color.Blue)
                )
                Box(
                    modifier = Modifier
                        .size(size = 80.dp)
                        .background(Color.Cyan)
                )
            }
        }
    }
}
```



- Todas centradas en el eje horizontal.

```
@Preview
@Composable
fun CajasCentradasHorizontal() {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(Color(color = 0xFFEFEFEF)),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Box(
            modifier = Modifier
                .size(size = 80.dp)
                .background(Color.Red)
        )
        Box(
            modifier = Modifier
                .size(size = 80.dp)
                .background(Color.Green)
        )
        Box(
            modifier = Modifier
                .size(size = 80.dp)
                .background(Color.Blue)
        )
        Box(
            modifier = Modifier
                .size(size = 80.dp)
                .background(Color.Cyan)
        )
    }
}
```



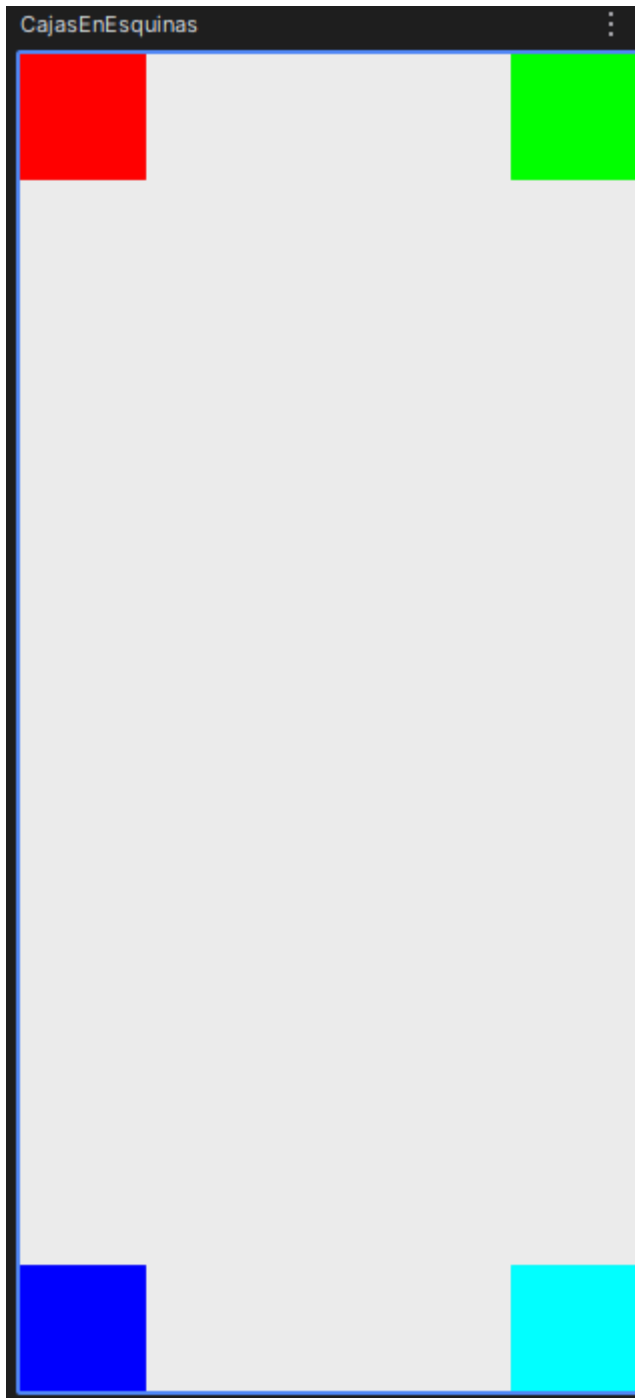
- Cada una en una esquina distinta de la pantalla.

```
@Preview
@Composable
fun CajasEnEsquinas() {
    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color( color = 0xFFE0E0E0))
    ) {
        Box(
            modifier = Modifier
                .size( size = 80.dp)
                .background(Color.Red)
                .align(Alignment.TopStart)
        )

        Box(
            modifier = Modifier
                .size( size = 80.dp)
                .background(Color.Green)
                .align(Alignment.TopEnd)
        )

        Box(
            modifier = Modifier
                .size( size = 80.dp)
                .background(Color.Blue)
                .align(Alignment.BottomStart)
        )

        Box(
            modifier = Modifier
                .size( size = 80.dp)
                .background(Color.Cyan)
                .align(Alignment.BottomEnd)
        )
    }
}
```



### Relaciones entre cajas:

- Haz que la caja azul (boxB) se coloque debajo de la caja roja.
- Haz que la caja verde (boxG) se coloque a la derecha de la caja roja.
- Haz que la caja cyan (boxC) quede encima de la roja

```
@Preview
@Composable
fun RelacionCajas() {
    ConstraintLayout(modifier = Modifier.fillMaxSize()) {
        val (boxC, boxR, boxB, boxG) = createRefs()

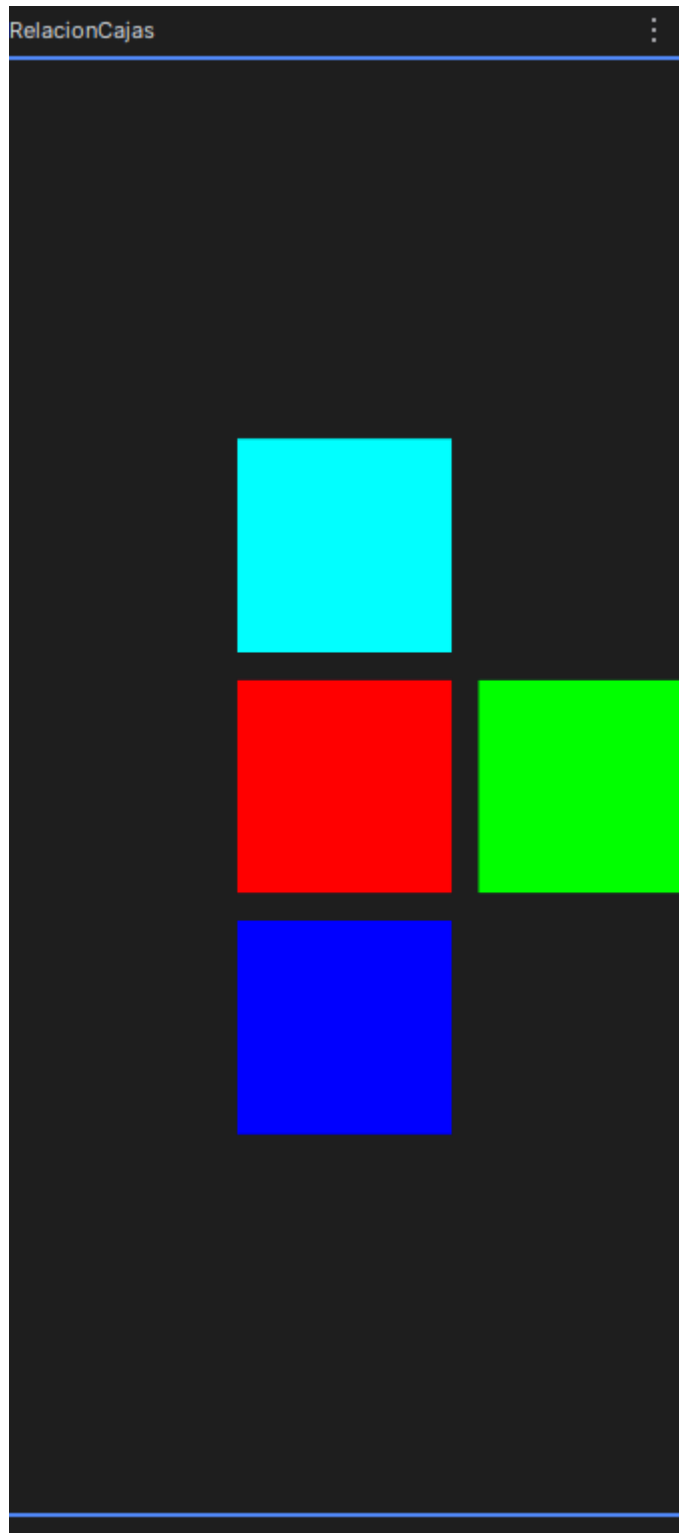
        Box(
            modifier = Modifier
                .size(size = 125.dp)
                .background(Color.Red)
                .constrainAs(ref = boxR) {
                    centerTo(other = parent)
                })

        Box(
            modifier = Modifier
                .size(size = 125.dp)
                .background(Color.Blue)
                .constrainAs(ref = boxB) {
                    top.linkTo(anchor = boxR.bottom, margin = 16.dp)
                    centerHorizontallyTo(other = boxR)
                })

        Box(
            modifier = Modifier
                .size(size = 125.dp)
                .background(Color.Green)
                .constrainAs(ref = boxG) {
                    start.linkTo(anchor = boxR.end, margin = 16.dp)
                    centerVerticallyTo(other = boxR)
                })

        Box(
            modifier = Modifier
                .size(size = 125.dp)
                .background(Color.Cyan)
                .constrainAs(ref = boxC) {
                    bottom.linkTo(anchor = boxR.top, margin = 16.dp)
                    centerHorizontallyTo(other = boxR)
                })
    }
}
```





**Alineación múltiple:**

- Alinea todas las cajas al centro verticalmente, pero distribúyelas horizontalmente: una a la izquierda, otra al centro, otra a la derecha y otra fuera del eje (por ejemplo, un poco desplazada usando margin).

```

@Composable
fun AlineacionMultiple() {
    ConstraintLayout(Modifier.fillMaxSize()) {
        val (left, center, right, off) = createRefs()

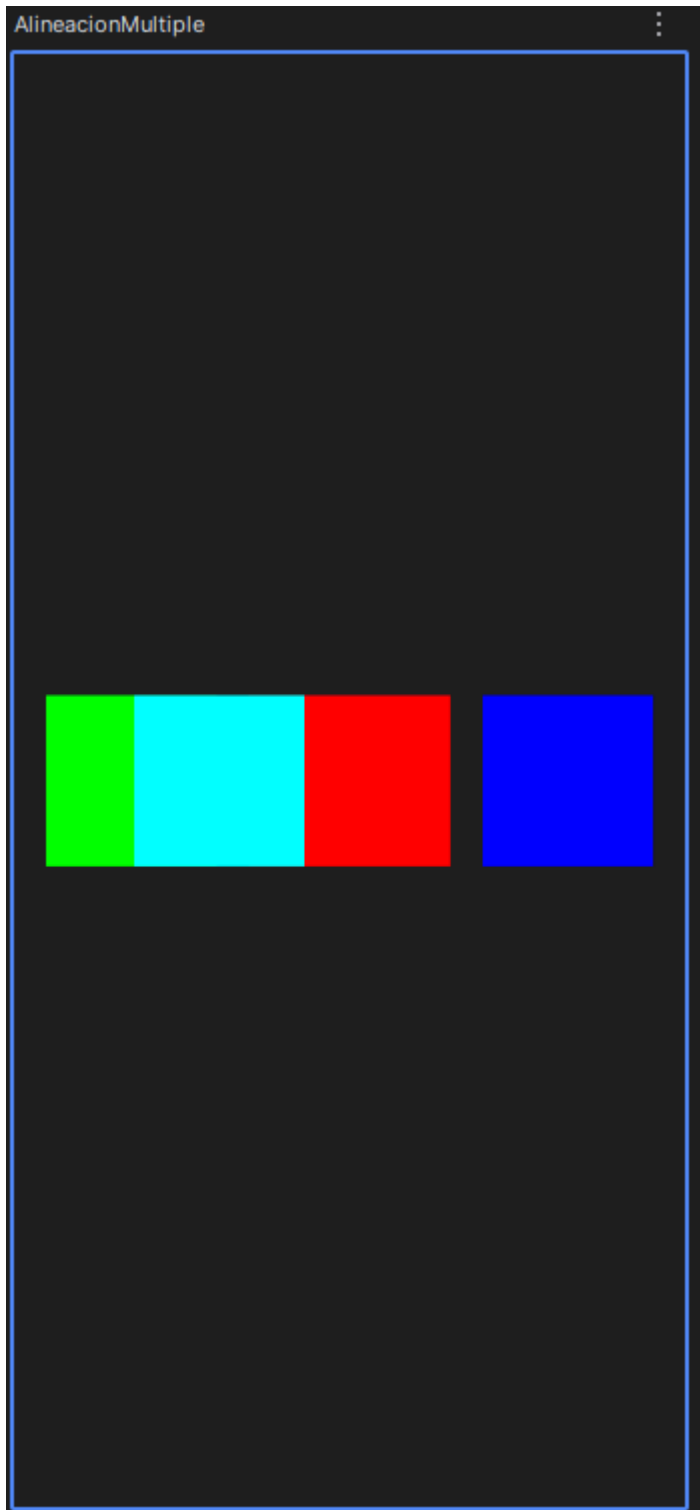
        Box(
            Modifier
                .constrainAs(ref = center) {
                    centerVerticallyTo(other = parent)
                    centerHorizontallyTo(other = parent)
                    width = Dimension.percent(percent = 0.30f)
                    height = Dimension.value(dp = 100.dp)
                }
                .background(Color.Red))

        Box(
            Modifier
                .size(size = 100.dp)
                .background(Color.Green)
                .constrainAs(ref = left) {
                    centerVerticallyTo(other = center)
                    end.linkTo(anchor = center.start, margin = 16.dp)
                    start.linkTo(anchor = parent.start, margin = 16.dp)
                })

        Box(
            Modifier
                .size(size = 100.dp)
                .background(Color.Blue)
                .constrainAs(ref = right) {
                    centerVerticallyTo(other = center)
                    start.linkTo(anchor = center.end, margin = 16.dp)
                    end.linkTo(anchor = parent.end, margin = 16.dp)
                })

        Box(
            Modifier
                .size(size = 100.dp)
                .background(Color.Cyan)
                .constrainAs(ref = off) {
                    centerVerticallyTo(other = center)
                    start.linkTo(anchor = left.end, margin = 16.dp)
                    end.linkTo(anchor = center.start, margin = 32.dp)
                })
    }
}

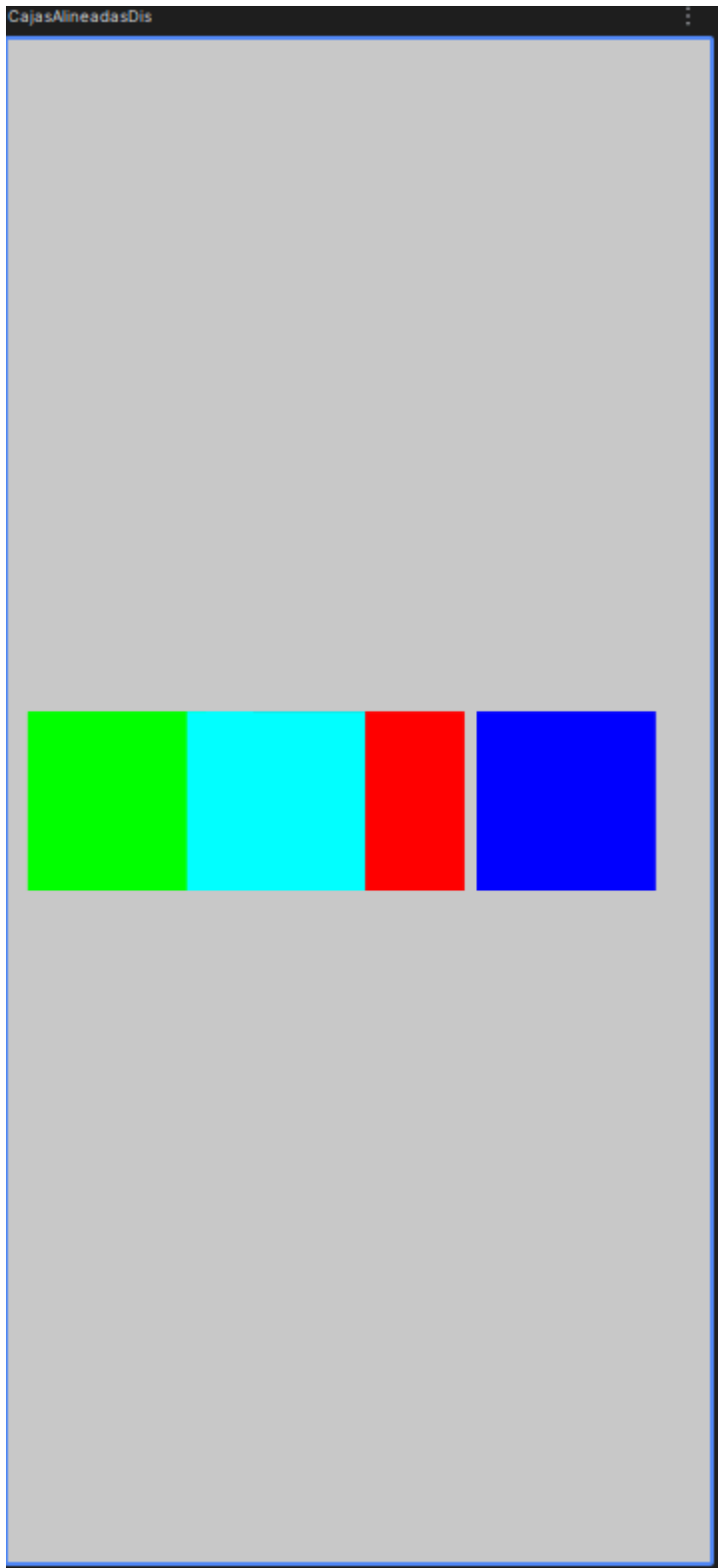
```



### Márgenes y separación:

- Añade márgenes de 16.dp entre las cajas.
- Experimenta con diferentes tamaños (`.size(100.dp)` o `.fillMaxWidth(0.3f)`) para ver cómo cambian las posiciones relativas.

```
@Preview
@Composable
fun CajasAlineadasDis() {
    ConstraintLayout(
        modifier = Modifier
            .fillMaxSize()
            .background(Color.LightGray)
    ) {
        val (cajaIzq, cajaCentro, cajaDer, cajaFuera) = createRefs()
        Box(
            modifier = Modifier
                .background(Color.Red)
                .constrainAs(ref = cajaCentro) {
                    width = Dimension.percent(percent = 0.3f)
                    height = Dimension.value(dp = 100.dp)
                    centerTo(other = parent)
                }
        )
        Box(
            modifier = Modifier
                .background(Color.Green)
                .constrainAs(ref = cajaIzq) {
                    width = Dimension.value(dp = 100.dp)
                    height = Dimension.value(dp = 100.dp)
                    centerVerticallyTo(other = cajaCentro)
                    end.linkTo(anchor = cajaCentro.start, margin = 32.dp)
                    start.linkTo(anchor = parent.start, margin = 16.dp)
                }
        )
        Box(
            modifier = Modifier
                .background(Color.Blue)
                .constrainAs(ref = cajaDer) {
                    width = Dimension.value(dp = 100.dp)
                    height = Dimension.value(dp = 100.dp)
                    centerVerticallyTo(other = cajaCentro)
                    start.linkTo(anchor = cajaCentro.end, margin = 16.dp)
                    end.linkTo(anchor = parent.end, margin = 40.dp)
                }
        )
        Box(
            modifier = Modifier
                .background(Color.Cyan)
                .constrainAs(ref = cajaFuera) {
                    width = Dimension.value(dp = 100.dp)
                    height = Dimension.value(dp = 100.dp)
                    centerVerticallyTo(other = cajaCentro)
                    start.linkTo(anchor = parent.start, margin = 100.dp)
                    horizontalBias = 0.2f
                }
        )
    }
}
```



**Creatividad libre:**

- Crea una forma o figura (por ejemplo una “cruz”, una “L”, o una “T”) usando solo las posiciones de las cajas.
- Usa colores y márgenes distintos para distinguirlas.

```

@Preview
@Composable
fun FiguraCruz() {
    ConstraintLayout(Modifier.fillMaxSize()) {
        val (c, up, down, left, right) = createRefs()

        Box(
            Modifier
                .size(size = 100.dp)
                .background(Color.Red)
                .constrainAs(ref = c) { centerTo(other = parent)
        )

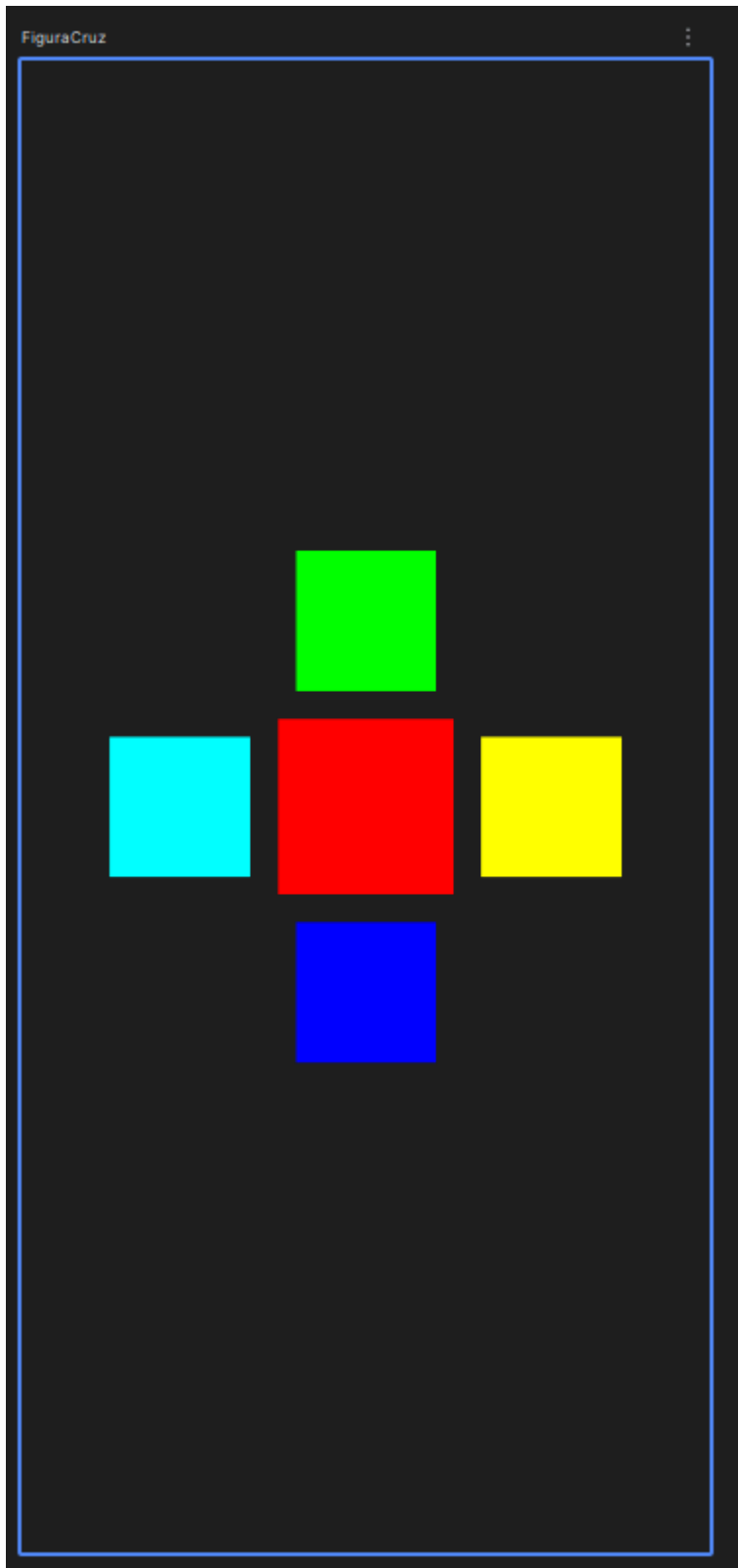
        Box(
            Modifier
                .size(size = 80.dp)
                .background(Color.Green)
                .constrainAs(ref = up) {
                    bottom.linkTo(anchor = c.top, margin = 16.dp)
                    centerHorizontallyTo(other = c)
                }
        )

        Box(
            Modifier
                .size(size = 80.dp)
                .background(Color.Blue)
                .constrainAs(ref = down) {
                    top.linkTo(anchor = c.bottom, margin = 16.dp)
                    centerHorizontallyTo(other = c)
                }
        )

        Box(
            Modifier
                .size(size = 80.dp)
                .background(Color.Cyan)
                .constrainAs(ref = left) {
                    end.linkTo(anchor = c.start, margin = 16.dp)
                    centerVerticallyTo(other = c)
                }
        )

        Box(
            Modifier
                .size(size = 80.dp)
                .background(Color.Yellow)
                .constrainAs(ref = right) {
                    start.linkTo(anchor = c.end, margin = 16.dp)
                    centerVerticallyTo(other = c)
                }
        )
    }
}

```



## **6 – Diseño libre con Constraint Layout**

**Diseña una pantalla de tarjeta de presentación en la que puedas colocar libremente varios elementos: una foto, el nombre, un botón de contacto y una breve biografía.**




```

@Composable
fun TarjetaPresentacion(
    onContactoClick: () -> Unit = {}
) {
    ConstraintLayout(
        modifier = Modifier
            .fillMaxSize()
            .padding( all = 20.dp)
            .background(Color( color = 0xFFFFF5F5))
    ) {
        val (foto, nombre, bio, boton) = createRefs()
        Image(
            painter = painterResource(id = R.drawable.milla),
            contentDescription = "Foto de perfil",
            contentScale = ContentScale.Crop,
            modifier = Modifier
                .size( size = 120.dp)
                .clip( shape = RoundedCornerShape( size = 60.dp))
                .constrainAs( ref = foto) {
                    top.linkTo( anchor = parent.top, margin = 16.dp)
                    start.linkTo( anchor = parent.start, margin = 16.dp)
                }
        )
        Text(
            text = "Luis Milla",
            style = MaterialTheme.typography.headlineMedium.copy(fontWeight = FontWeight.Bold),
            modifier = Modifier.constrainAs( ref = nombre) {
                top.linkTo( anchor = foto.top)
                start.linkTo( anchor = foto.end, margin = 16.dp)
                end.linkTo( anchor = parent.end, margin = 16.dp)
                width = Dimension.fillToConstraints
            }
        )
        Text(
            text = "Mediocentro creativo. Amante del control del juego y el fútbol champagne del Getafe."
            ,
            style = MaterialTheme.typography.bodyMedium,
            color = Color( color = 0xFF4A4A4A),
            modifier = Modifier
                .constrainAs( ref = bio) {
                    top.linkTo( anchor = nombre.bottom, margin = 100.dp)
                    start.linkTo( anchor = parent.start, margin = 16.dp)
                    end.linkTo( anchor = parent.end, margin = 16.dp)
                    width = Dimension.percent( percent = 0.80f)
                }
        )
        Button(
            onClick = onContactoClick,
            modifier = Modifier
                .constrainAs( ref = boton) {
                    bottom.linkTo( anchor = parent.bottom, margin = 24.dp)
                    end.linkTo( anchor = parent.end, margin = 24.dp)
                }
        ) {
            Text( text = "Contactar")
        }
    }
}

```

TarjetaPresentacionPreview



## Luis Milla

Mediocentro creativo. Amante del control del juego y el fútbol champagne del Getafe.

Contactar