

M7 - Layouts



Definició de **layout**

- Un layout és un contenidor que ens permet alinear els components gràfics de la nostra app.
- Sense un layout, tots els components que posem a l'app es col·locarien un a sobre de l'altre.
- Veurem els següents components de **Jetpack Compose** per organitzar els components de la nostra app: **box**, **column** i **row**.

Definició de layout



Column



Row



Box

Box

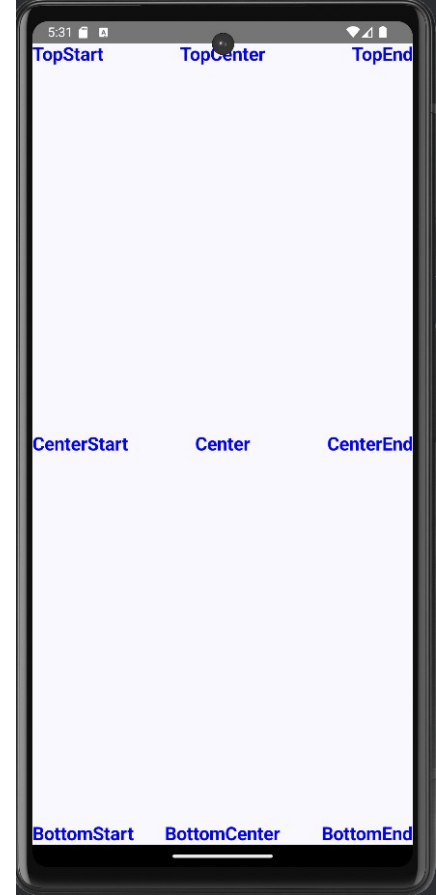
- Ens permet alinear els elements al centre, a dalt a la dreta, a dalt a l'esquerra...

```
@Composable
fun MyBox(modifier: Modifier = Modifier) {
    Box(modifier, contentAlignment = Alignment.Center) {
        Box(modifier = Modifier.width(50.dp).height(50.dp).background(Color.Red))
    }
}
```

Com podem veure, haurem d'usar l'annotation
@Composable

Box

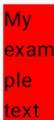
- Al codi anterior pots trobar la propietat *"contentAlignment"* que indica com s'alinearan els elements dins del box.
- Tenim les següents opcions d'alineació: *TopStart*, *TopCenter*, *TopEnd*, *CenterStart*, *Center*, *CenterEnd*, *BottomStart*, *BottomCenter*, *BottomEnd*.



Box

- Al box anterior li hem especificat les seves dimensions (width i height) però, si no especifiquem res, les mides del box s'adapten al seu contingut.
- Observa els següents exemples:

```
Box(modifier = Modifier.width(50.dp).background(Color.Red)){  
    Text(text = "My example text")  
}
```



My
exam
ple
text

```
Box(modifier = Modifier.width(50.dp).height(50.dp).background(Color.Red)){  
    Text(text = "My example text")  
}
```



My
exam

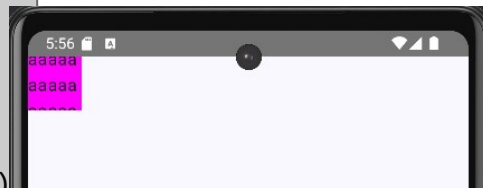
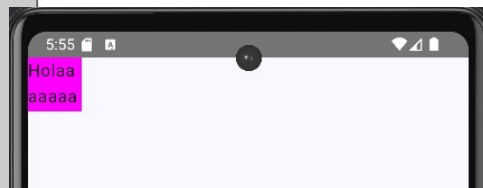
Box

- Quan tinguem un Box amb unes dimensions determinades que no permeten veure tot el contingut d'aquest, podem afegir la propietat d'scroll:

rememberScrollState()

- També el podem aplicar a **Column**

```
Box(modifier = Modifier
    .width(50.dp)
    .height(50.dp)
    .background(Color.Magenta)
    .verticalScroll(
        rememberScrollState()
    )){
    Text(text =
        "Holaaaaaaaaaaaaaaaaaaaaaaaaaaaaa")
}
```



Column

- Ens permet alinear els elements un sota l'altre.

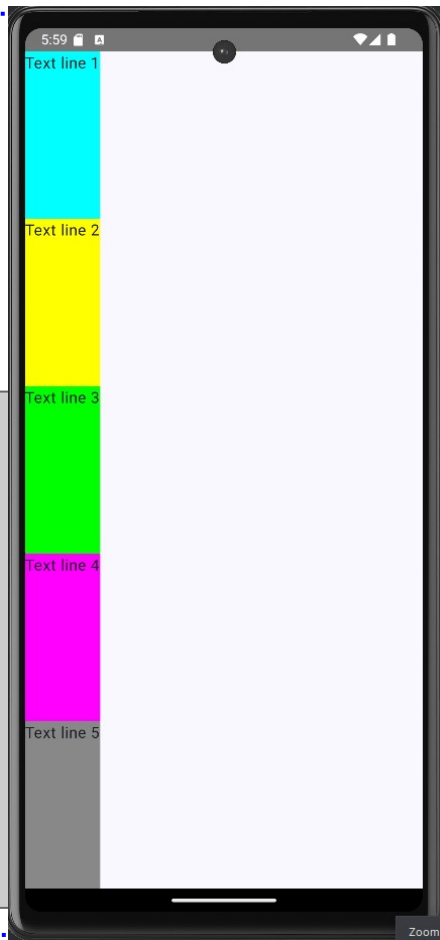
```
@Composable
fun MyColumn(modifier: Modifier = Modifier) {
    Column() {
        Text("Text line 1")
        Text("Text line 2")
        Text("Text line 3")
        Text("Text line 4")
        Text("Text line 5")
    }
}
```



Column

- Podem repartir l'alçada que ocuparà cada element amb la propietat "weight" (pes):

```
@Composable
fun MyColumn(modifier: Modifier = Modifier) {
    Column() {
        Text("Text line 1", Modifier.background(Color.Cyan).weight(1f))
        Text("Text line 2", Modifier.background(Color.Yellow).weight(1f))
        Text("Text line 3", Modifier.background(Color.Green).weight(1f))
        Text("Text line 4", Modifier.background(Color.Magenta).weight(1f))
        Text("Text line 5", Modifier.background(Color.Gray).weight(1f))
    }
}
```



Column

- Podem separar els elements a la nostra columna amb la propietat “verticalArrangement”:

```
@Composable
fun MyColumn(modifier: Modifier = Modifier) {
    Column(modifier.verticalArrangement = Arrangement.SpaceBetween) {
        Text("Text line 1", Modifier.background(Color.Cyan))
        Text("Text line 2", Modifier.background(Color.Yellow))
        ...
    }
}
```

- Comprova les opcions que hi ha: SpaceAround, SpaceEvenly...

Row

- Ens permet alinear els elements un al costat de l'altre.

```
@Composable
fun MyRow(modifier: Modifier = Modifier) {
    Row() {
        Text("Text line 1")
        Text("Text line 2")
        Text("Text line 3")
        Text("Text line 4")
        Text("Text line 5")
    }
}
```

Row

- Podem aplicar pesos igual que hem vist amb Column.
- També podem assignar un scroll, tot i que ara la propietat s'anomena "horizontalScroll" en lloc de "verticalScroll".

```
@Composable
fun MyRow(modifier: Modifier = Modifier) {
    Row(modifier.horizontalScroll(rememberScrollState())) {
        Text("Text line 1")
        Text("Text line 2")
        Text("Text line 3")
        Text("Text line 4")
        Text("Text line 5")
    }
}
```