

# LazyColumn, LazyRow LazyVerticalGrid, LazyHorizontalGrid

[illegible][illegible]



# LazyColumn

```
@Composable
fun LazyColumn(
    modifier: Modifier = Modifier,
    state: LazyListState = rememberLazyListState(),
    contentPadding: PaddingValues = PaddingValues(0.dp),
    reverseLayout: Boolean = false,
    verticalArrangement: Arrangement.Vertical = if (!reverseLayout) Arrangement.Top else Arrangement.Bottom,
    horizontalAlignment: Alignment.Horizontal = Alignment.Start,
    flingBehavior: FlingBehavior = ScrollableDefaults.flingBehavior(),
    userScrollEnabled: Boolean = true,
    content: LazyListScope.() -> Unit
)
```



# LazyColumn

## **contentPadding:**

**PaddingValues = PaddingValues(0.dp)**, → Agrega un padding antes del primer elemento o después del último.

```
val allContentPadding = PaddingValues(all = 16.dp)
val hvContentPadding = PaddingValues(horizontal = 16.dp, vertical = 16.dp)
val stebContentPadding = PaddingValues(start = 16.dp, top = 16.dp, end = 16.dp, bottom = 16.dp)

LazyColumn(contentPadding = hvContentPadding,
    ){
    :::
}
```

## **reverseLayout**

**Boolean = false** → Indica si visualmente se invertirá el orden en el que aparecerán los elementos.

reverseLayout = false

reverseLayout = true



# LazyColumn

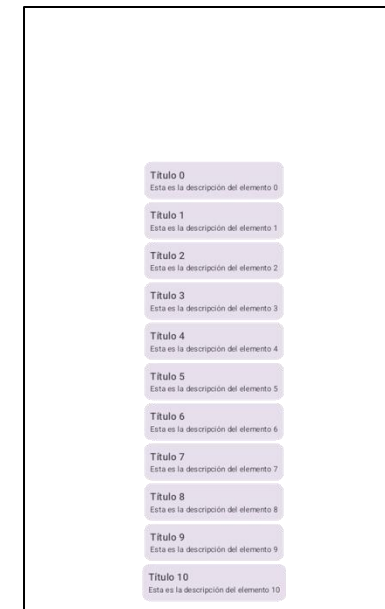
## ***verticalArrangement:***

**Arrangement.Vertical** → Indica cómo alinear verticalmente el conjunto de elementos dentro de las dimensiones del LazyColumn.

## ***horizontalAlignment:*** :

**Alignment.Horizontal** → Indica cómo se alinean horizontalmente los items mostrados en el LazyColumn.

```
LazyColumn(modifier = Modifier.fillMaxWidth()  
            .fillMaxHeight(),  
            verticalArrangement = Arrangement.Bottom,  
            horizontalAlignment = Alignment.CenterHorizontally,  
        )  
{  
    :::  
}
```





# LazyColumn

## **content:**

**LazyListScope.() -> Unit** → Indica el contenido que se mostrará en el LazyColumn.

```
LazyListScope
{
    fun item(key: Any? = null,
            contentType: Any? = null,
            content: @Composable LazyItemScope.() -> Unit
            )
}
```

### Añade un item al LazyColumn

content	es el composable que describe el item.
key	representa el identificador del item dentro del LazyColumn. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser estable o no estable. Se considera estable si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de content se asocian a la key del item. Si el key es estable y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es no estable se recrearán los estados internos (es decir, se reinician los valores recordados).
contentType	su valor representa el tipo del content. Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. null es un valor válido. indica que todos los items son del mismo tipo

```
LazyColumn(modifier = Modifier.fillMaxSize(),
            contentPadding = PaddingValues(16.dp),
            )
{
    item {
        ShowItem( Item(0) )
    }
    item {
        Box(modifier = Modifier.fillMaxWidth()
            .height(50.dp)
            .background(Color.Red, RectangleShape),
            contentAlignment = Alignment.Center
        ) {
            Text("Listado", color = Color.White, )
        }
    }
}

@Composable
fun ShowItem(item: Item)
{
    Card(modifier = Modifier.padding(bottom = 4.dp))
    {
        Column(modifier = Modifier.padding(10.dp))
        {
            Text(text= item.title, style= MaterialTheme.typography.titleMedium)
            Text(text= item.description, style= MaterialTheme.typography.bodySmall)
        }
    }
}
```

```
class Item(val value: Int)
{
    val title: String = "Título $value"
    val description: String = "Esta es la descripción del elemento $value"
}
```





# LazyColumn

## content:

**LazyListScope.() -> Unit** → Indica el contenido que se mostrará en el LazyColumn.

```
fun <T> LazyListScope.items(items: List<T>,
    key: ((item: T) -> Any)? = null,
    contentType: (item: T) -> Any? = { null },
    itemContent: @Composable LazyItemScope.(item: T) -> Unit
)
}
```

## Añade una lista de items al LazyColumn

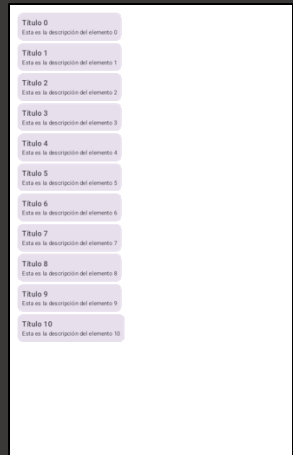
items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe como parámetro un elemento de la lista y describe cómo debe visualizarse
key	representa el identificador del item dentro del LazyColumn. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad <i>id</i> de un <i>entity</i> ). Se considera <b>no estable</b> si su valor procede de un valor mutable o del <i>index</i> del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la key del item. Si el key es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados). Se representa como una lambda que recibe como parámetro el elemento.
contentType	su valor representa el tipo del <i>content</i> . Las composiciones de <i>items</i> del mismo tipo podrían reutilizarse más eficientemente. <i>null</i> es un valor válido. indica que todos los items son del mismo tipo. Se representa como una lambda que recibe como parámetro el elemento.

```
val items = (0..10).map { index -> Item(index) }
```

```
LazyColumn(modifier = Modifier.fillMaxSize(),
    contentPadding = PaddingValues(16.dp),
)
{
    items(items = items)
    {
        item -> ShowItem(item)
    }
}
```

```
@Composable
fun ShowItem(item: Item)
{
    Card(modifier = Modifier.padding(bottom = 4.dp))
    {
        Column(modifier = Modifier.padding(10.dp))
        {
            Text(text= item.title, style= MaterialTheme.typography.titleMedium)
            Text(text= item.description, style= MaterialTheme.typography.bodySmall)
        }
    }
}
```

```
class Item(val value: Int)
{
    val title: String = "Título $value"
    val description: String = "Esta es la descripción del elemento $value"
}
```





# LazyColumn

## content:

**LazyListScope.()** -> Unit → Indica el contenido que se mostrará en el LazyColumn.

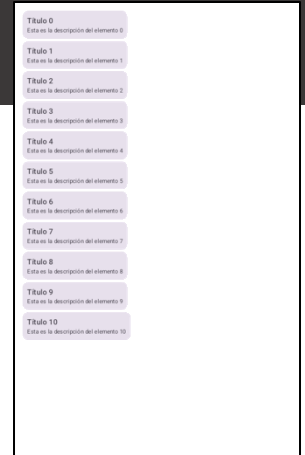
```
fun <T> LazyListScope.itemsIndexed(items: List<T>,
    key: ((index: Int, item: T) -> Any)? = null,
    contentType: (index: Int, item: T) -> Any? = {null},
    itemContent: @Composable LazyItemScope.(index: Int, item: T) -> Unit
)
```

## Añade una lista de items al LazyColumn

items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe como parámetro el index del elemento y el elemento de la lista
key	representa el identificador del item dentro del LazyColumn. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de content se asocian a la key del item. Si el key es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados)
contentType	Se representa como una lambda que recibe como parámetros el index del elemento y el elemento. su valor representa el tipo del content. Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <b>null</b> es un valor válido. indica que todos los items son del mismo tipo, Se representa como una lambda que recibe como parámetros el index del elemento y el elemento

```
val items = (0..10).map { index -> Item(index) }

LazyColumn(modifier = Modifier.fillMaxSize(),
    contentPadding = PaddingValues(16.dp),
)
{
    itemsIndexed(items = items)
    {
        index, item -> ShowItem(item)
    }
}
```





# LazyColumn

## content:

**LazyListScope.() -> Unit** → Indica el contenido que se mostrará en el LazyColumn.

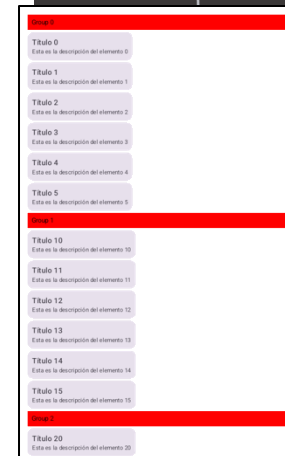
```
fun <T> LazyListScope.stickyHeader(key: Any? = null,
    contentType: Any? = null,
    content: @Composable LazyListItemScope.() -> Unit
)
```

Define un encabezado que se “pega” a la parte superior de la lista mientras se hace scroll. Esto significa que, al desplazarte, el elemento declarado con `stickyHeader` permanece visible en la parte superior hasta que otro `stickyHeader` lo reemplaza.

content	El composable que describe el encabezado
key	representa el identificador del item dentro del LazyColumn. Si no se especifica, su valor será el <code>index</code> del item (posición que ocupa el item dentro de la lista). La key puede ser estable o no estable. Se considera estable si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad <code>id</code> de un <code>entity</code> ). Se considera no estable si su valor procede de un valor mutable o del <code>index</code> del item. Los estados creados con <code>remember</code> o <code>rememberSaveable</code> dentro de <code>content</code> se asocian a la key del item. Si el key es estable y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es no estable se recrearán los estados internos (es decir, se reinician los valores recordados)
contentType	su valor representa el tipo del content. Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <code>null</code> es un valor válido. indica que todos los items son del mismo tipo,

```
val mapItems = (0..5).flatMap { group-> (0..5).map {
    index-> Item(group, index+group*10)
} }
    .groupBy { item -> item.group }

LazyColumn(modifier = Modifier.fillMaxSize(),
    contentPadding = PaddingValues(16.dp),
) {
    mapItems.forEach { entry-> stickyHeader()
    }
```



```
        modifier = Modifier.padding(bottom = 4.dp)
        .fillMaxSize()
        .clip(RectangleShape)
        .background(Color.Red)
        .padding(8.dp)
        content {
            item {
                Text(entry.key.toString())
            }
            item {
                Text(entry.value.toString())
            }
        }
    }
}
```





# LazyColumn

```
fun HorizontalDivider(modifier: Modifier = Modifier,  
    thickness: Dp = DividerDefaults.Thickness,  
    color: Color = DividerDefaults.color,  
)
```

*Línea horizontal que actúa como delimitador entre los items de un LazyColumn*

modifier      Por ejemplo `Modifier.padding(vertical = 16.dp)`  
thickness      representa el grosor de la línea  
color          Color de la línea.

```
val items = (0..10).map { Item(value = it) }
```

```
LazyColumn(modifier = Modifier.fillMaxSize(),  
    contentPadding = PaddingValues(horizontal = 16.dp),  
    verticalArrangement = Arrangement.Top,  
    horizontalAlignment = Alignment.Start  
)
```

```
{  
    items(items = items)  
    {  
        item -> ShowItem(item)  
    }  
}
```

```
@Composable  
fun ShowItem(item: Item)  
{  
    Column(modifier = Modifier.height(IntrinsicSize.Min))  
    {  
        Text(text = item.title, style = MaterialTheme.typography.titleMedium)  
        Text(text = item.description, style = MaterialTheme.typography.bodySmall)  
        HorizontalDivider(modifier = Modifier.padding(vertical = 16.dp),  
            thickness = 1.dp,  
        )  
    }  
}
```

<b>Título 0</b>
Descripción del 0
<b>Título 1</b>
Descripción del 1
<b>Título 2</b>
Descripción del 2
<b>Título 3</b>
Descripción del 3
<b>Título 4</b>
Descripción del 4
<b>Título 5</b>
Descripción del 5
<b>Título 6</b>
Descripción del 6
<b>Título 7</b>
Descripción del 7
<b>Título 8</b>
Descripción del 8
<b>Título 9</b>
Descripción del 9
<b>Título 10</b>
Descripción del 10



# LazyRow

```
@Composable
fun LazyRow(
    modifier: Modifier = Modifier,
    state: LazyListState = rememberLazyListState(),
    contentPadding: PaddingValues = PaddingValues(0.dp),
    reverseLayout: Boolean = false,
    horizontalArrangement: Arrangement.Horizontal = if (!reverseLayout) Arrangement.Start else Arrangement.End,
    verticalAlignment: Alignment.Vertical = Alignment.Top,
    flingBehavior: FlingBehavior = ScrollableDefaults.flingBehavior(),
    userScrollEnabled: Boolean = true,
    content: LazyListScope.() -> Unit
)
```



# LazyRow

## *contentPadding:*

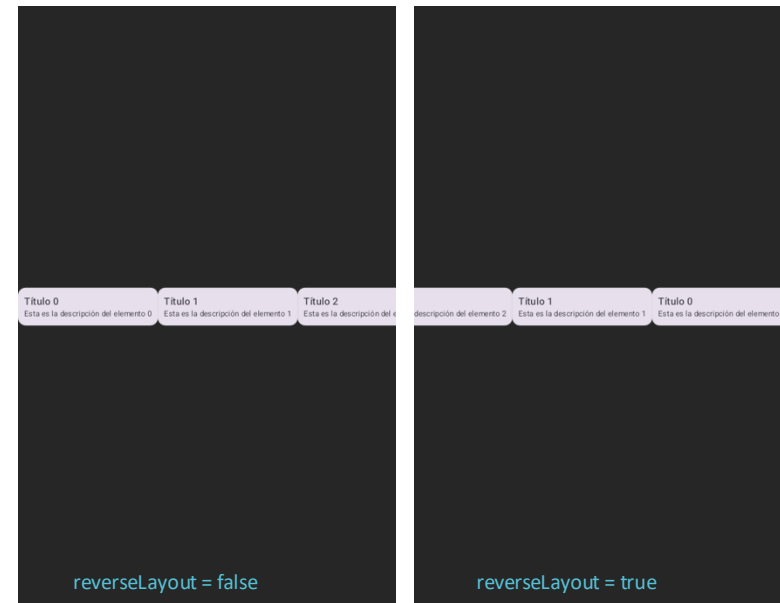
**PaddingValues = PaddingValues(0.dp)**, → Agrega un padding antes del primer elemento o después del último.

```
val allContentPadding = PaddingValues(all = 16.dp)
val hvContentPadding = PaddingValues(horizontal = 16.dp, vertical = 16.dp)
val stebContentPadding = PaddingValues(start = 16.dp, top = 16.dp, end = 16.dp, bottom = 16.dp)

LazyRow (contentPadding = hvContentPadding,
){
    :::
}
```

## *reverseLayout*

**Boolean = false** → Indica si visualmente se invertirá el orden en el que aparecerán los elementos.





# LazyRow

## ***horizontalArrangement:***

**Arrangement.Horizontal** → Indica cómo alinear horizontalmente el conjunto de elementos dentro de las dimensiones del LazyRow.

## ***verticalAlignment:*** :

**Alignment.Vertical** → Indica cómo se alinean verticalmente los items mostrados en el LazyRow.

```
LazyRow(modifier = Modifier.fillMaxSize(),
    horizontalArrangement = Arrangement.Start,
    verticalAlignment = Alignment.CenterVertically
)
{
    :::
}
```





# LazyRow

## content:

**LazyListScope.() -> Unit** → Indica el contenido que se mostrará en el LazyRow.

```
LazyListScope
{
    fun item(key: Any? = null,
            contentType: Any? = null,
            content: @Composable LazyItemScope.() -> Unit
            )
}
```

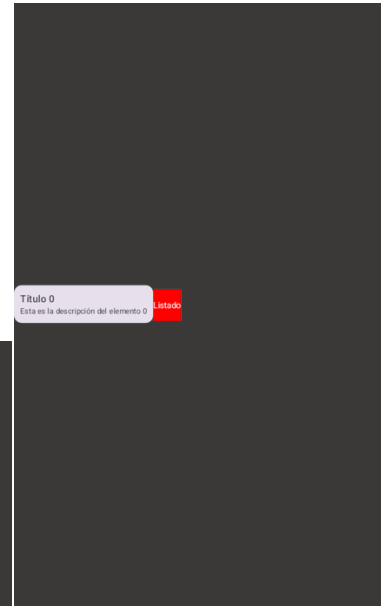
## Añade un item al LazyRow

content	es el composable que describe el item.
key	representa el identificador del item dentro del LazyRow. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad <i>id</i> de un <i>entity</i> ). Se considera no estable si su valor procede de un valor mutable o del <i>index</i> del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la <i>key</i> del item. Si el <i>key</i> es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el <i>key</i> es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados).
contentType	su valor representa el tipo del <i>content</i> . Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <i>null</i> es un valor válido, indica que todos los items son del mismo tipo

```
LazyRow(modifier = Modifier.fillMaxSize(),
        contentPadding = PaddingValues(16.dp),
        )
{
    item {
        ShowItem( Item(0) )
    }
    item {
        Box(modifier = Modifier.fillMaxWidth()
            .height(50.dp)
            .background(Color.Red, RectangleShape),
            contentAlignment = Alignment.Center
        ) {
            Text("Listado", color = Color.White, )
        }
    }
}

@Composable
fun ShowItem(item: Item)
{
    Card(modifier = Modifier.padding(bottom = 4.dp))
    {
        Column(modifier = Modifier.padding(10.dp))
        {
            Text(text= item.title, style= MaterialTheme.typography.titleMedium)
            Text(text= item.description, style= MaterialTheme.typography.bodySmall)
        }
    }
}
```

```
class Item(val value: Int)
{
    val title: String = "Título $value"
    val description: String = "Esta es la descripción del elemento $value"
}
```





# LazyRow

## content:

**LazyListScope.() -> Unit** → Indica el contenido que se mostrará en el LazyRow.

```
fun <T> LazyListScope.items(items: List<T>,
    key: ((item: T) -> Any)? = null,
    contentType: (item: T) -> Any? = { null },
    itemContent: @Composable LazyItemScope.(item: T) -> Unit
)
```

### Añade una lista de items al LazyRow

items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe como parámetro un elemento de la lista y describe cómo debe visualizarse
key	representa el identificador del item dentro del LazyRow. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de content se asocian a la key del item. Si el key es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados). Se representa como una lambda que recibe como parámetro el elemento.
contentType	su valor representa el tipo del content. Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. null es un valor válido. indica que todos los items son del mismo tipo. Se representa como una lambda que recibe como parámetro el elemento.

```
val items = (0..10).map { index -> Item(index) }
```

```
LazyRow(modifier = Modifier.fillMaxSize(),
    contentPadding = PaddingValues(16.dp),
)
```

```
{
    items(items = items)
    {
        item -> ShowItem(item)
    }
}
```

```
@Composable
fun ShowItem(item: Item)
{
    Card(modifier = Modifier.padding(bottom = 4.dp))
    {
        Column(modifier = Modifier.padding(10.dp))
        {
            Text(text= item.title, style= MaterialTheme.typography.titleMedium)
            Text(text= item.description, style= MaterialTheme.typography.bodySmall)
        }
    }
}
```

```
class Item(val value: Int)
{
    val title: String = "Título $value"
    val description: String = "Esta es la descripción del elemento $value"
}
```





# LazyRow

## content:

**LazyListScope.()** -> Unit → Indica el contenido que se mostrará en el LazyRow.

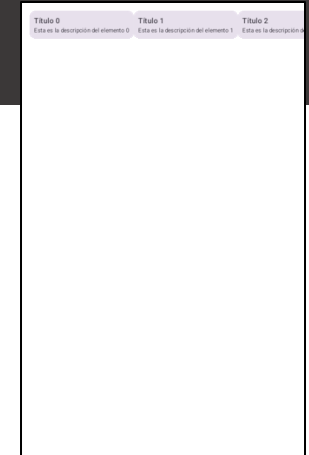
```
fun <T> LazyListScope.itemsIndexed(items: List<T>,
    key: ((index: Int, item: T) -> Any)? = null,
    contentType: (index: Int, item: T) -> Any? = {null},
    itemContent: @Composable LazyItemScope.(index: Int, item: T) -> Unit
)
```

## Añade una lista de items al LazyColumn

items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe como parámetro el index del elemento y el elemento de la lista
key	representa el identificador del item dentro del LazyRow. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la key del item. Si el key es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados) Se representa como una lambda que recibe como parámetros el index del elemento y el elemento.
contentType	su valor representa el tipo del <i>content</i> . Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <i>null</i> es un valor válido. indica que todos los items son del mismo tipo, Se representa como una lambda que recibe como parámetros el index del elemento y el elemento

```
val items = (0..10).map { index -> Item(index) }

LazyRow(modifier = Modifier.fillMaxSize(),
    contentPadding = PaddingValues(16.dp),
)
{
    itemsIndexed(items = items)
    {
        index, item -> ShowItem(item)
    }
}
```





# LazyRow

## content:

**LazyListScope.()** -> Unit → Indica el contenido que se mostrará en el LazyRow.

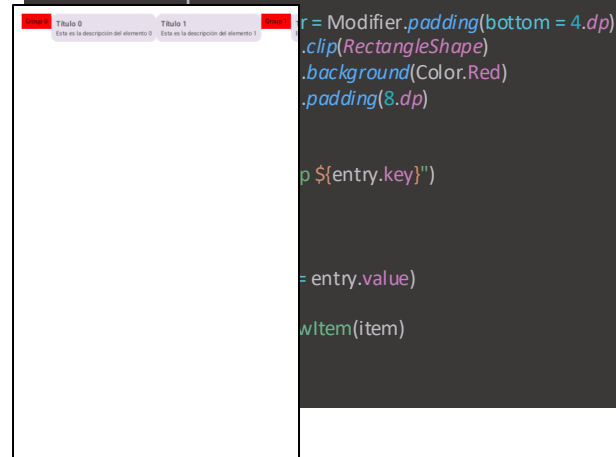
```
fun <T> LazyListScope.stickyHeader(key: Any? = null,
    contentType: Any? = null,
    content: @Composable LazyItemScope.() -> Unit
)
```

Define un encabezado que se “pega” a la parte superior de la lista mientras se hace scroll. Esto significa que, al desplazarte, el elemento declarado con stickyHeader permanece visible en la parte superior hasta que otro stickyHeader lo reemplaza.

content	El composable que describe el encabezado
key	representa el identificador del item dentro del LazyRow. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser estable o no estable. Se considera estable si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de content se asocian a la key del item. Si el key es estable y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es no estable se recrearán los estados internos (es decir, se reinician los valores recordados)
contentType	su valor representa el tipo del content. Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. null es un valor válido. indica que todos los items son del mismo tipo,

```
val mapItems = (0..5).flatMap { group-> (0..1).map {
    index-> Item(group, index+group*10)
} }
    .groupBy { item -> item.group }

LazyRow(modifier = Modifier.fillMaxSize(),
    contentPadding = PaddingValues(16.dp),
    )
{
    mapItems.forEach { entry-> stickyHeader()
    {
```







# LazyRow

```
fun VerticalDivider(modifier: Modifier = Modifier,  
    thickness: Dp = DividerDefaults.Thickness,  
    color: Color = DividerDefaults.color,  
)
```

*Línea vertical que actúa como delimitador entre los items de un LazyRow.*

modifier      Por ejemplo `Modifier.padding(horizontal = 16.dp)`  
thickness      representa el grosor de la línea  
color          Color de la línea.

```
val items = (0..10).map { Item(value = it) }
```

```
LazyRow(modifier = Modifier.fillMaxSize(),  
    contentPadding = PaddingValues(horizontal = 16.dp),  
    horizontalArrangement = Arrangement.Start,  
    verticalAlignment = Alignment.CenterVertically  
)
```

```
{  
    items(items = items)  
    {  
        item -> ShowItem(item)  
    }  
}
```

```
@Composable  
fun ShowItem(item: Item)  
{  
    Row(modifier = Modifier.height(IntrinsicSize.Min))  
    {  
        Column()  
        {  
            Text(text = item.title, style = MaterialTheme.typography.titleMedium)  
            Text(text = item.description, style = MaterialTheme.typography.bodySmall)  
        }  
        VerticalDivider(modifier = Modifier.padding(start = 10.dp, end = 10.dp),  
            thickness = 1.dp,  
        )  
    }  
}
```

Título 0	Título 1	Título 2	Título 3	Título 4	Título 5
Descripción del 0	Descripción del 1	Descripción del 2	Descripción del 3	Descripción del 4	Descripción del 5



# LazyVerticalGrid

```
@Composable
fun LazyVerticalGrid (
    columns: GridCells,
    modifier: Modifier = Modifier,
    contentPadding: PaddingValues = PaddingValues(0.dp),
    reverseLayout: Boolean = false,
    verticalArrangement: Arrangement.Vertical = if (!reverseLayout) Arrangement.Top else Arrangement.Bottom,
    horizontalArrangement: Arrangement.Horizontal = Arrangement.Start,
    userScrollEnabled: Boolean = true,
    content: LazyGridScope.() -> Unit
)
```



# LazyVerticalGrid

## contentPadding:

**PaddingValues = PaddingValues(0.dp)**, → Agrega un padding antes del primer elemento o después del último.

```
val allContentPadding = PaddingValues(all = 16.dp)
val hvContentPadding = PaddingValues(horizontal = 16.dp, vertical = 16.dp)
val stebContentPadding = PaddingValues(start = 16.dp, top = 16.dp, end = 16.dp, bottom = 16.dp)
```

```
LazyVerticalGrid (contentPadding = hvContentPadding)
{
    :::
}
```

## reverseLayout

**Boolean = false** → Indica si visualmente se invertirá el orden en el que aparecerán los elementos.

Título 0 Descripción del 0	Título 1 Descripción del 1	Título 2 Descripción del 2	Título 3 Descripción del 3	Descripción del 92	Descripción del 93	Descripción del 94	Descripción del 95
Título 4 Descripción del 4	Título 5 Descripción del 5	Título 6 Descripción del 6	Título 7 Descripción del 7	Título 88 Descripción del 88	Título 89 Descripción del 89	Título 90 Descripción del 90	Título 91 Descripción del 91
Título 8 Descripción del 8	Título 9 Descripción del 9	Título 10 Descripción del 10	Título 11 Descripción del 11	Título 84 Descripción del 84	Título 85 Descripción del 85	Título 86 Descripción del 86	Título 87 Descripción del 87
Título 12 Descripción del 12	Título 13 Descripción del 13	Título 14 Descripción del 14	Título 15 Descripción del 15	Título 80 Descripción del 80	Título 81 Descripción del 81	Título 82 Descripción del 82	Título 83 Descripción del 83
Título 16 Descripción del 16	Título 17 Descripción del 17	Título 18 Descripción del 18	Título 19 Descripción del 19	Título 76 Descripción del 76	Título 77 Descripción del 77	Título 78 Descripción del 78	Título 79 Descripción del 79
Título 20 Descripción del 20	Título 21 Descripción del 21	Título 22 Descripción del 22	Título 23 Descripción del 23	Título 72 Descripción del 72	Título 73 Descripción del 73	Título 74 Descripción del 74	Título 75 Descripción del 75
Título 24 Descripción del 24	Título 25 Descripción del 25	Título 26 Descripción del 26	Título 27 Descripción del 27	Título 68 Descripción del 68	Título 69 Descripción del 69	Título 70 Descripción del 70	Título 71 Descripción del 71
Título 28 Descripción del 28	Título 29 Descripción del 29	Título 30 Descripción del 30	Título 31 Descripción del 31	Título 64 Descripción del 64	Título 65 Descripción del 65	Título 66 Descripción del 66	Título 67 Descripción del 67
Título 32 Descripción del 32	Título 33 Descripción del 33	Título 34 Descripción del 34	Título 35 Descripción del 35	Título 60 Descripción del 60	Título 61 Descripción del 61	Título 62 Descripción del 62	Título 63 Descripción del 63
Título 36 Descripción del 36	Título 37 Descripción del 37	Título 38 Descripción del 38	Título 39 Descripción del 39	Título 56 Descripción del 56	Título 57 Descripción del 57	Título 58 Descripción del 58	Título 59 Descripción del 59
Título 40 Descripción del 40	Título 41 Descripción del 41	Título 42 Descripción del 42	Título 43 Descripción del 43	Título 52 Descripción del 52	Título 53 Descripción del 53	Título 54 Descripción del 54	Título 55 Descripción del 55
Título 44 Descripción del 44	Título 45 Descripción del 45	Título 46 Descripción del 46	Título 47 Descripción del 47	Título 48 Descripción del 48	Título 49 Descripción del 49	Título 50 Descripción del 50	Título 51 Descripción del 51
Título 48 Descripción del 48	Título 49 Descripción del 49	Título 50 Descripción del 50	Título 51 Descripción del 51	Título 44 Descripción del 44	Título 45 Descripción del 45	Título 46 Descripción del 46	Título 47 Descripción del 47
Título 52 Descripción del 52	Título 53 Descripción del 53	Título 54 Descripción del 54	Título 55 Descripción del 55	Título 40 Descripción del 40	Título 41 Descripción del 41	Título 42 Descripción del 42	Título 43 Descripción del 43
Título 56 Descripción del 56	Título 57 Descripción del 57	Título 58 Descripción del 58	Título 59 Descripción del 59	Título 36 Descripción del 36	Título 37 Descripción del 37	Título 38 Descripción del 38	Título 39 Descripción del 39
Título 60 Descripción del 60	Título 61 Descripción del 61	Título 62 Descripción del 62	Título 63 Descripción del 63	Título 32 Descripción del 32	Título 33 Descripción del 33	Título 34 Descripción del 34	Título 35 Descripción del 35
Título 64 Descripción del 64	Título 65 Descripción del 65	Título 66 Descripción del 66	Título 67 Descripción del 67	Título 28 Descripción del 28	Título 29 Descripción del 29	Título 30 Descripción del 30	Título 31 Descripción del 31
Título 68 Descripción del 68	Título 69 Descripción del 69	Título 70 Descripción del 70	Título 71 Descripción del 71	Título 24 Descripción del 24	Título 25 Descripción del 25	Título 26 Descripción del 26	Título 27 Descripción del 27
Título 72 Descripción del 72	Título 73 Descripción del 73	Título 74 Descripción del 74	Título 75 Descripción del 75	Título 20 Descripción del 20	Título 21 Descripción del 21	Título 22 Descripción del 22	Título 23 Descripción del 23
Título 76 Descripción del 76	Título 77 Descripción del 77	Título 78 Descripción del 78	Título 79 Descripción del 79	Título 16 Descripción del 16	Título 17 Descripción del 17	Título 18 Descripción del 18	Título 19 Descripción del 19
Título 80 Descripción del 80	Título 81 Descripción del 81	Título 82 Descripción del 82	Título 83 Descripción del 83	Título 12 Descripción del 12	Título 13 Descripción del 13	Título 14 Descripción del 14	Título 15 Descripción del 15
Título 84 Descripción del 84	Título 85 Descripción del 85	Título 86 Descripción del 86	Título 87 Descripción del 87	Título 8 Descripción del 8	Título 9 Descripción del 9	Título 10 Descripción del 10	Título 11 Descripción del 11
Título 88 Descripción del 88	Título 89 Descripción del 89	Título 90 Descripción del 90	Título 91 Descripción del 91	Título 4 Descripción del 4	Título 5 Descripción del 5	Título 6 Descripción del 6	Título 7 Descripción del 7
Título 82 Descripción del 82	Título 83 Descripción del 83	Título 84 Descripción del 84	Título 85 Descripción del 85	Título 0 Descripción del 0	Título 1 Descripción del 1	Título 2 Descripción del 2	Título 3 Descripción del 3

reverseLayout = false

reverseLayout = true



# LazyVerticalGrid

## ***horizontalArrangement:***

**Arrangement.Horizontal** → Indica cómo alinear horizontalmente el conjunto de elementos dentro de las dimensiones del LazyVerticalGrid.

## ***verticalAlignment:***

**Alignment.Vertical** → Indica cómo se alinean verticalmente los items mostrados en el LazyVerticalGrid.

```
LazyVerticalGrid(modifier = Modifier.fillMaxSize(),
    horizontalArrangement = Arrangement.Start,
    verticalAlignment = Alignment.CenterVertically
)
{
    ::::
}
```

<b>Título 0</b> Descripción del 0	<b>Título 1</b> Descripción del 1	<b>Título 2</b> Descripción del 2	<b>Título 3</b> Descripción del 3
<b>Título 4</b> Descripción del 4	<b>Título 5</b> Descripción del 5	<b>Título 6</b> Descripción del 6	<b>Título 7</b> Descripción del 7
<b>Título 8</b> Descripción del 8	<b>Título 9</b> Descripción del 9	<b>Título 10</b> Descripción del 10	<b>Título 11</b> Descripción del 11
<b>Título 12</b> Descripción del 12	<b>Título 13</b> Descripción del 13	<b>Título 14</b> Descripción del 14	<b>Título 15</b> Descripción del 15
<b>Título 16</b> Descripción del 16	<b>Título 17</b> Descripción del 17	<b>Título 18</b> Descripción del 18	<b>Título 19</b> Descripción del 19
<b>Título 20</b> Descripción del 20	<b>Título 21</b> Descripción del 21	<b>Título 22</b> Descripción del 22	<b>Título 23</b> Descripción del 23
<b>Título 24</b> Descripción del 24	<b>Título 25</b> Descripción del 25	<b>Título 26</b> Descripción del 26	<b>Título 27</b> Descripción del 27
<b>Título 28</b> Descripción del 28	<b>Título 29</b> Descripción del 29	<b>Título 30</b> Descripción del 30	<b>Título 31</b> Descripción del 31
<b>Título 32</b> Descripción del 32	<b>Título 33</b> Descripción del 33	<b>Título 34</b> Descripción del 34	<b>Título 35</b> Descripción del 35
<b>Título 36</b> Descripción del 36	<b>Título 37</b> Descripción del 37	<b>Título 38</b> Descripción del 38	<b>Título 39</b> Descripción del 39
<b>Título 40</b> Descripción del 40	<b>Título 41</b> Descripción del 41	<b>Título 42</b> Descripción del 42	<b>Título 43</b> Descripción del 43
<b>Título 44</b> Descripción del 44	<b>Título 45</b> Descripción del 45	<b>Título 46</b> Descripción del 46	<b>Título 47</b> Descripción del 47
<b>Título 48</b> Descripción del 48	<b>Título 49</b> Descripción del 49	<b>Título 50</b> Descripción del 50	



# LazyVerticalGrid

**columns:**

**GridCells** → Indica cuántas columnas se mostrarán.

```
columns = GridCells.Fixed(4)

Establece que las columnas deben ser 4 .

LazyVerticalGrid(modifier = Modifier.fillMaxSize(),
    columns = GridCells.Fixed(4)
)
{ :::: }
```

```
columns = GridCells.FixedSize(150.dp)

Establece el número de columnas en función de su tamaño.

LazyVerticalGrid(modifier= Modifier.fillMaxSize(),
    columns= GridCells.FixedSize(150.dp),
    horizontalArrangement= Arrangement.Center,
    verticalArrangement= Arrangement.Top,
)
{ :::: }
```

Título 0	Título 1	Título 2	Título 3
Descripción del 0	Descripción del 1	Descripción del 2	Descripción del 3
Título 4	Título 5	Título 6	Título 7
Descripción del 4	Descripción del 5	Descripción del 6	Descripción del 7
Título 8	Título 9	Título 10	Título 11
Descripción del 8	Descripción del 9	Descripción del 10	Descripción del 11
Título 12	Título 13	Título 14	Título 15
Descripción del 12	Descripción del 13	Descripción del 14	Descripción del 15
Título 16	Título 17	Título 18	Título 19
Descripción del 16	Descripción del 17	Descripción del 18	Descripción del 19
Título 20	Título 21	Título 22	Título 23
Descripción del 20	Descripción del 21	Descripción del 22	Descripción del 23
Título 24	Título 25	Título 26	Título 27
Descripción del 24	Descripción del 25	Descripción del 26	Descripción del 27
Título 28	Título 29	Título 30	Título 31
Descripción del 28	Descripción del 29	Descripción del 30	Descripción del 31
Título 32	Título 33	Título 34	Título 35
Descripción del 32	Descripción del 33	Descripción del 34	Descripción del 35
Título 36	Título 37	Título 38	Título 39
Descripción del 36	Descripción del 37	Descripción del 38	Descripción del 39
Título 40	Título 41	Título 42	Título 43
Descripción del 40	Descripción del 41	Descripción del 42	Descripción del 43
Título 44	Título 45	Título 46	Título 47
Descripción del 44	Descripción del 45	Descripción del 46	Descripción del 47
Título 48	Título 49	Título 50	Título 51
Descripción del 48	Descripción del 49	Descripción del 50	Descripción del 51
Título 52	Título 53	Título 54	Título 55
Descripción del 52	Descripción del 53	Descripción del 54	Descripción del 55
Título 56	Título 57	Título 58	Título 59
Descripción del 56	Descripción del 57	Descripción del 58	Descripción del 59
Título 60	Título 61	Título 62	Título 63
Descripción del 60	Descripción del 61	Descripción del 62	Descripción del 63
Título 64	Título 65	Título 66	Título 67
Descripción del 64	Descripción del 65	Descripción del 66	Descripción del 67
Título 68	Título 69	Título 70	Título 71
Descripción del 68	Descripción del 69	Descripción del 70	Descripción del 71
Título 72	Título 73	Título 74	Título 75
Descripción del 72	Descripción del 73	Descripción del 74	Descripción del 75
Título 76	Título 77	Título 78	Título 79
Descripción del 76	Descripción del 77	Descripción del 78	Descripción del 79
Título 80	Título 81	Título 82	Título 83
Descripción del 80	Descripción del 81	Descripción del 82	Descripción del 83
Título 84	Título 85	Título 86	Título 87
Descripción del 84	Descripción del 85	Descripción del 86	Descripción del 87
Título 88	Título 89	Título 90	Título 91
Descripción del 88	Descripción del 89	Descripción del 90	Descripción del 91
Título 92	Título 93	Título 94	Título 95

Título 0	Título 1	Título 2
Descripción del 0	Descripción del 1	Descripción del 2
Título 3	Título 4	Título 5
Descripción del 3	Descripción del 4	Descripción del 5
Título 6	Título 7	Título 8
Descripción del 6	Descripción del 7	Descripción del 8
Título 9	Título 10	Título 11
Descripción del 9	Descripción del 10	Descripción del 11
Título 12	Título 13	Título 14
Descripción del 12	Descripción del 13	Descripción del 14
Título 15	Título 16	Título 17
Descripción del 15	Descripción del 16	Descripción del 17
Título 18	Título 19	Título 20
Descripción del 18	Descripción del 19	Descripción del 20
Título 21	Título 22	Título 23
Descripción del 21	Descripción del 22	Descripción del 23
Título 24	Título 25	Título 26
Descripción del 24	Descripción del 25	Descripción del 26
Título 27	Título 28	Título 29
Descripción del 27	Descripción del 28	Descripción del 29
Título 30	Título 31	Título 32
Descripción del 30	Descripción del 31	Descripción del 32
Título 33	Título 34	Título 35
Descripción del 33	Descripción del 34	Descripción del 35
Título 36	Título 37	Título 38
Descripción del 36	Descripción del 37	Descripción del 38
Título 39	Título 40	Título 41
Descripción del 39	Descripción del 40	Descripción del 41
Título 42	Título 43	Título 44
Descripción del 42	Descripción del 43	Descripción del 44
Título 45	Título 46	Título 47
Descripción del 45	Descripción del 46	Descripción del 47
Título 48	Título 49	Título 50
Descripción del 48	Descripción del 49	Descripción del 50

```
columns = GridCells.Adaptative(150.dp)

Establece el número de columnas en función de su tamaño mínimo.

LazyVerticalGrid(modifier= Modifier.fillMaxSize(),
    columns= GridCells.Adaptative(150.dp),
    horizontalArrangement= Arrangement.Center,
    verticalArrangement= Arrangement.Top,
)
{ :::: }
```



# LazyVerticalGrid

## **content:**

**LazyListScope.() -> Unit** → Indica el contenido que se mostrará en el LazyGrid.

```
LazyGridScope
{
    fun item(key: Any? = null,
            span: (LazyGridItemSpanScope.(item: T) -> GridItemSpan)? = null,
            contentType: Any? = null,
            content: @Composable LazyItemScope.() -> Unit
            )
}
```

### *Añade un item al LazyRow*

content	es el composable que describe el item.
key	representa el identificador del item dentro del LazyGrid. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad <i>id</i> de un <i>entity</i> ). Se considera <b>no estable</b> si su valor procede de un valor mutable o del <i>index</i> del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la <i>key</i> del item. Si el <i>key</i> es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el <i>key</i> es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados).
contentType	su valor representa el tipo del <i>content</i> . Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <i>null</i> es un valor válido. indica que todos los items son del mismo tipo,
span	Define cuántas columnas ocupará cada elemento a mostrar. Por defecto vale 1 (un elemento ocupa una columna).. Se representa como una lambda que recibe como parámetro el elemento.



# LazyVerticalGrid

## content:

**LazyListScope.()** -> Unit → Indica el contenido que se mostrará en el LazyGrid.

```
fun <T> LazyGridScope.items(items: List<T>,
    key: ((item: T) -> Any)? = null,
    span: (LazyGridItemSpanScope.(item: T) -> GridItemSpan)? = null,
    contentType: (item: T) -> Any? = { null },
    itemContent: @Composable LazyGridItemScope.(item: T) -> Unit
)
```

## Añade una lista de items al LazyGrid

items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe un elemento de la lista y describe cómo se visualiza.
key	representa el identificador del item dentro del LazyGrid. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <i>estable</i> o <i>no estable</i> . Se considera <i>estable</i> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la key del item. Si el key es <i>estable</i> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <i>no estable</i> se recrearán los estados internos (es decir, se reinician los valores recordados).
contentType	Se representa como una lambda que recibe como parámetro el elemento. su valor representa el tipo del <i>content</i> . Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <i>null</i> es un valor válido. indica que todos los items son del mismo tipo. Se representa como una lambda que recibe como parámetro el elemento.
span	Define cuántas columnas ocupará cada elemento a mostrar. Por defecto vale 1 (un elemento ocupa una columna). Se representa como una lambda que recibe como parámetro el elemento.

```
val items = (0..50).map { Item(value = it) }
Box( contentAlignment = Alignment.Center )
{
    LazyVerticalGrid(modifier = Modifier.fillMaxSize(),
        columns = GridCells.Fixe dSize(120.dp),
        contentPadding = PaddingValues(all = 16.dp),
        horizontalArrangement = Arrangement.Center,
        verticalArrangement = Arrangement.Top,
    )
}

{
    items(items= items,
        span= {item-> GridItemSpan(if (item.value % 4 == 0) 2
            else 1
        )
    }
    item -> ShowItem(item)
}

@Composable
fun ShowItem(item: Item)
{
    Card(modifier = Modifier.padding(bottom=4.dp,
        end = 4.dp))
    {
        Column(modifier = Modifier.padding(4.dp))
        {
            Text(text = item.title,
                style = MaterialTheme.typography.titleMedium)
            Text(text = item.description,
                style = MaterialTheme.typography.bodySmall)
        }
    }
}
```

Título 0 Descripción del 0	Título 1 Descripción del 1	Título 2 Descripción del 2
Título 3 Descripción del 3	Título 4 Descripción del 4	Título 5 Descripción del 5
Título 6 Descripción del 6	Título 7 Descripción del 7	Título 8 Descripción del 8
Título 9 Descripción del 9	Título 10 Descripción del 10	Título 11 Descripción del 11
Título 12 Descripción del 12	Título 13 Descripción del 13	Título 14 Descripción del 14
Título 15 Descripción del 15	Título 16 Descripción del 16	Título 17 Descripción del 17
Título 18 Descripción del 18	Título 19 Descripción del 19	Título 20 Descripción del 20
Título 21 Descripción del 21	Título 22 Descripción del 22	Título 23 Descripción del 23
Título 24 Descripción del 24	Título 25 Descripción del 25	Título 26 Descripción del 26
Título 27 Descripción del 27	Título 28 Descripción del 28	Título 29 Descripción del 29
Título 30 Descripción del 30	Título 31 Descripción del 31	Título 32 Descripción del 32
Título 33 Descripción del 33	Título 34 Descripción del 34	Título 35 Descripción del 35
Título 36 Descripción del 36	Título 37 Descripción del 37	Título 38 Descripción del 38
Título 39 Descripción del 39	Título 40 Descripción del 40	Título 41 Descripción del 41
Título 42 Descripción del 42	Título 43 Descripción del 43	Título 44 Descripción del 44
Título 45 Descripción del 45	Título 46 Descripción del 46	Título 47 Descripción del 47
Título 48 Descripción del 48	Título 49 Descripción del 49	Título 50 Descripción del 50



# LazyVerticalGrid

## **content:**

**LazyListScope.()** -> Unit → Indica el contenido que se mostrará en el LazyGrid.

```
fun <T> LazyGridScope.itemsIndexed(items: List<T>,
    key: ((index: Int, item: T) -> Any)? = null,
    contentType: (index: Int, item: T) -> Any? = {null},
    itemContent: @Composable LazyItemScope.(index: Int, item: T) -> Unit
)
```

## *Añade una lista de items al LazyColumn*

items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe como parámetro el índice del elemento y el elemento de la lista
key	representa el identificador del item dentro del LazyGrid. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la key del item. Si el key es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados) Se representa como una lambda que recibe como parámetros el index del elemento y el elemento.
contentType	su valor representa el tipo del content. Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <b>null</b> es un valor válido. indica que todos los items son del mismo tipo, Se representa como una lambda que recibe como parámetros el index del elemento y el elemento.
span	Define cuántas columnas ocupará cada elemento a mostrar. Por defecto vale 1 (un elemento ocupa una columna).. Se representa como una lambda que recibe como parámetro el elemento.





# LazyHorizontalGrid

```
@Composable
fun LazyHorizontalGrid (
    rows: GridCells,
    modifier: Modifier = Modifier,
    contentPadding: PaddingValues = PaddingValues(0.dp),
    reverseLayout: Boolean = false,
    horizontalArrangement: Arrangement.Horizontal = if(!reverseLayout) Arrangement.Start else Arrangement.End,
    verticalArrangement: Arrangement.Vertical = Arrangement.Top,
    userScrollEnabled: Boolean = true,
    content: LazyGridScope.() -> Unit
)
```



# LazyHorizontalGrid

## contentPadding:

**PaddingValues = PaddingValues(0.dp)**, → Agrega un padding antes del primer elemento o después del último.

```
val allContentPadding = PaddingValues(all = 16.dp)
val hvContentPadding = PaddingValues(horizontal = 16.dp, vertical = 16.dp)
val stebContentPadding = PaddingValues(start = 16.dp, top = 16.dp, end = 16.dp, bottom = 16.dp)
```

```
LazyHorizontalGrid (contentPadding = hvContentPadding)
{
    :::
}
```

## reverseLayout

**Boolean = false** → Indica si visualmente se invertirá el orden en el que aparecerán los elementos.

Título 0 Descripción del 0	Título 1 Descripción del 1	Título 2 Descripción del 2	Título 3 Descripción del 3	Título 4 Descripción del 4	Título 5 Descripción del 5	Título 6 Descripción del 6	Título 7 Descripción del 7	Título 8 Descripción del 8	Título 9 Descripción del 9	Título 10 Descripción del 10	Título 11 Descripción del 11	Título 12 Descripción del 12	Título 13 Descripción del 13	Título 14 Descripción del 14	Título 15 Descripción del 15	Título 16 Descripción del 16	Título 17 Descripción del 17	Título 18 Descripción del 18	Título 19 Descripción del 19	Título 20 Descripción del 20	Título 21 Descripción del 21	Título 22 Descripción del 22	Título 23 Descripción del 23	Título 24 Descripción del 24	Título 25 Descripción del 25	Título 26 Descripción del 26	Título 27 Descripción del 27	Título 28 Descripción del 28	Título 29 Descripción del 29	Título 30 Descripción del 30	Título 31 Descripción del 31	Título 32 Descripción del 32	Título 33 Descripción del 33	Título 34 Descripción del 34	Título 35 Descripción del 35	Título 36 Descripción del 36	Título 37 Descripción del 37	Título 38 Descripción del 38	Título 39 Descripción del 39	Título 40 Descripción del 40	Título 41 Descripción del 41	Título 42 Descripción del 42	Título 43 Descripción del 43	Título 44 Descripción del 44	Título 45 Descripción del 45	Título 46 Descripción del 46	Título 47 Descripción del 47	Título 48 Descripción del 48	Título 49 Descripción del 49	Título 50 Descripción del 50	Título 51 Descripción del 51	Título 52 Descripción del 52	Título 53 Descripción del 53	Título 54 Descripción del 54	Título 55 Descripción del 55	Título 56 Descripción del 56	Título 57 Descripción del 57	Título 58 Descripción del 58	Título 59 Descripción del 59	Título 60 Descripción del 60	Título 61 Descripción del 61	Título 62 Descripción del 62	Título 63 Descripción del 63	Título 64 Descripción del 64	Título 65 Descripción del 65	Título 66 Descripción del 66	Título 67 Descripción del 67	Título 68 Descripción del 68	Título 69 Descripción del 69	Título 70 Descripción del 70	Título 71 Descripción del 71	Título 72 Descripción del 72	Título 73 Descripción del 73	Título 74 Descripción del 74	Título 75 Descripción del 75	Título 76 Descripción del 76	Título 77 Descripción del 77	Título 78 Descripción del 78	Título 79 Descripción del 79	Título 80 Descripción del 80	Título 81 Descripción del 81	Título 82 Descripción del 82	Título 83 Descripción del 83	Título 84 Descripción del 84	Título 85 Descripción del 85	Título 86 Descripción del 86	Título 87 Descripción del 87	Título 88 Descripción del 88	Título 89 Descripción del 89	Título 90 Descripción del 90	Título 91 Descripción del 91	Título 92 Descripción del 92	Título 93 Descripción del 93	Título 94 Descripción del 94	Título 95 Descripción del 95
-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	-------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------	---------------------------------

reverseLayout = false

reverseLayout = true



# LazyHorizontalGrid

## ***horizontalArrangement:***

**Arrangement.Horizontal** → Indica cómo alinear horizontalmente el conjunto de elementos dentro de las dimensiones del LazyHorizontalGrid.

## ***verticalAlignment:***

**Alignment.Vertical** → Indica cómo se alinean verticalmente los items mostrados en el LazyHorizontalGrid.

```
LazyHorizontalGrid(modifier = Modifier.fillMaxSize(),
    horizontalArrangement = Arrangement.Start,
    verticalAlignment = Alignment.CenterVertically
)
{
    ::::
}
```

<b>Título 0</b> Descripción del 0	<b>Título 1</b> Descripción del 1	<b>Título 2</b> Descripción del 2	<b>Título 3</b> Descripción del 3
<b>Título 4</b> Descripción del 4	<b>Título 5</b> Descripción del 5	<b>Título 6</b> Descripción del 6	<b>Título 7</b> Descripción del 7
<b>Título 8</b> Descripción del 8	<b>Título 9</b> Descripción del 9	<b>Título 10</b> Descripción del 10	<b>Título 11</b> Descripción del 11
<b>Título 12</b> Descripción del 12	<b>Título 13</b> Descripción del 13	<b>Título 14</b> Descripción del 14	<b>Título 15</b> Descripción del 15
<b>Título 16</b> Descripción del 16	<b>Título 17</b> Descripción del 17	<b>Título 18</b> Descripción del 18	<b>Título 19</b> Descripción del 19
<b>Título 20</b> Descripción del 20	<b>Título 21</b> Descripción del 21	<b>Título 22</b> Descripción del 22	<b>Título 23</b> Descripción del 23
<b>Título 24</b> Descripción del 24	<b>Título 25</b> Descripción del 25	<b>Título 26</b> Descripción del 26	<b>Título 27</b> Descripción del 27
<b>Título 28</b> Descripción del 28	<b>Título 29</b> Descripción del 29	<b>Título 30</b> Descripción del 30	<b>Título 31</b> Descripción del 31
<b>Título 32</b> Descripción del 32	<b>Título 33</b> Descripción del 33	<b>Título 34</b> Descripción del 34	<b>Título 35</b> Descripción del 35
<b>Título 36</b> Descripción del 36	<b>Título 37</b> Descripción del 37	<b>Título 38</b> Descripción del 38	<b>Título 39</b> Descripción del 39
<b>Título 40</b> Descripción del 40	<b>Título 41</b> Descripción del 41	<b>Título 42</b> Descripción del 42	<b>Título 43</b> Descripción del 43
<b>Título 44</b> Descripción del 44	<b>Título 45</b> Descripción del 45	<b>Título 46</b> Descripción del 46	<b>Título 47</b> Descripción del 47
<b>Título 48</b> Descripción del 48	<b>Título 49</b> Descripción del 49	<b>Título 50</b> Descripción del 50	



# LazyHorizontalGrid

## ROWS:

**GridCells** → Indica cuántas filas se mostrarán.

```
columns = GridCells.Fixed(4)
```

*Establece que las columnas deben ser 4 .*

```
LazyHorizontalGrid(modifier = Modifier.fillMaxSize(),  
    columns = GridCells.Fixed(4)  
)  
{ :: }
```

```
columns = GridCells.FixedSize(100.dp)
```

*Establece el número de columnas en función de su tamaño.*

```
LazyHorizontalGrid(modifier= Modifier.fillMaxSize(),  
    columns= GridCells.FixedSize(150.dp),  
    horizontalArrangement= Arrangement.Center,  
    verticalArrangement= Arrangement.Top,  
)  
{ :: }
```

Título 0 Descripción del 0	Título 4 Descripción del 4	Título 8 Descripción del 8	Título 12 Descripción del 12	Título 16 Descripción del 16	Título 20 Descripción del 20
Título 1 Descripción del 1	Título 5 Descripción del 5	Título 9 Descripción del 9	Título 13 Descripción del 13	Título 17 Descripción del 17	Título 21 Descripción del 21
Título 2 Descripción del 2	Título 6 Descripción del 6	Título 10 Descripción del 10	Título 14 Descripción del 14	Título 18 Descripción del 18	Título 22 Descripción del 22
Título 3 Descripción del 3	Título 7 Descripción del 7	Título 11 Descripción del 11	Título 15 Descripción del 15	Título 19 Descripción del 19	Título 23 Descripción del 23

Título 0 Descripción del 0	Título 9 Descripción del 9	Título 18 Descripción del 18	Título 27 Descripción del 27	Título 36 Descripción del 36	Título 45 Descripción del 45
Título 1 Descripción del 1	Título 10 Descripción del 10	Título 19 Descripción del 19	Título 28 Descripción del 28	Título 37 Descripción del 37	Título 46 Descripción del 46
Título 2 Descripción del 2	Título 11 Descripción del 11	Título 20 Descripción del 20	Título 29 Descripción del 29	Título 38 Descripción del 38	Título 47 Descripción del 47
Título 3 Descripción del 3	Título 12 Descripción del 12	Título 21 Descripción del 21	Título 30 Descripción del 30	Título 39 Descripción del 39	Título 48 Descripción del 48
Título 4 Descripción del 4	Título 13 Descripción del 13	Título 22 Descripción del 22	Título 31 Descripción del 31	Título 40 Descripción del 40	Título 49 Descripción del 49
Título 5 Descripción del 5	Título 14 Descripción del 14	Título 23 Descripción del 23	Título 32 Descripción del 32	Título 41 Descripción del 41	Título 50 Descripción del 50
Título 6 Descripción del 6	Título 15 Descripción del 15	Título 24 Descripción del 24	Título 33 Descripción del 33	Título 42 Descripción del 42	Título 51 Descripción del 51
Título 7 Descripción del 7	Título 16 Descripción del 16	Título 25 Descripción del 25	Título 34 Descripción del 34	Título 43 Descripción del 43	Título 52 Descripción del 52
Título 8 Descripción del 8	Título 17 Descripción del 17	Título 26 Descripción del 26	Título 35 Descripción del 35	Título 44 Descripción del 44	Título 53 Descripción del 53

```
columns = GridCells.Adaptative(150.dp)
```

*Establece el número de columnas en función de su tamaño mínimo.*

```
LazyVerticalGrid(modifier= Modifier.fillMaxSize(),  
    columns= GridCells.Adaptative(150.dp),  
    horizontalArrangement= Arrangement.Center,  
    verticalArrangement= Arrangement.Top,  
)  
{ :: }
```



# LazyHorizontalGrid

## **content:**

**LazyListScope.() -> Unit** → Indica el contenido que se mostrará en el LazyGrid.

```
LazyGridScope
{
    fun item(key: Any? = null,
            span: (LazyGridItemSpanScope(item: T) -> GridItemSpan)? = null,
            contentType: Any? = null,
            content: @Composable LazyItemScope.() -> Unit
            )
}
```

### *Añade un item al LazyRow*

content	es el composable que describe el item.
key	representa el identificador del item dentro del LazyGrid. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad <i>id</i> de un <i>entity</i> ). Se considera <b>no estable</b> si su valor procede de un valor mutable o del <i>index</i> del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la <i>key</i> del item. Si el <i>key</i> es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el <i>key</i> es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados).
contentType	su valor representa el tipo del <i>content</i> . Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <i>null</i> es un valor válido. indica que todos los items son del mismo tipo,
span	Define cuántas columnas ocupará cada elemento a mostrar. Por defecto vale 1 (un elemento ocupa una columna).. Se representa como una lambda que recibe como parámetro el elemento.



# LazyHorizontalGrid

## content:

**LazyListScope.()** -> Unit → Indica el contenido que se mostrará en el LazyGrid.

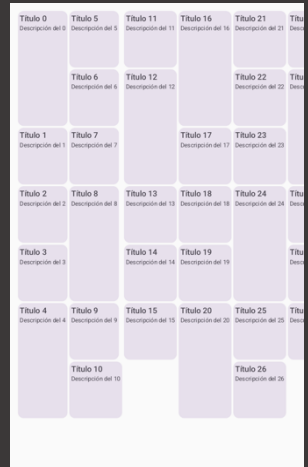
```
fun <T> LazyGridScope.items(items: List<T>,
    key: ((item: T) -> Any)? = null,
    span: (LazyGridItemSpanScope.(item: T) -> GridItemSpan)? = null,
    contentType: (item: T) -> Any? = { null },
    itemContent: @Composable LazyGridItemScope.(item: T) -> Unit
)
```

### Añade una lista de items al LazyGrid

items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe un elemento de la lista y describe cómo se visualiza.
key	representa el identificador del item dentro del LazyGrid. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la key del item. Si el key es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados).
contentType	Se representa como una lambda que recibe como parámetro el elemento. su valor representa el tipo del <i>content</i> . Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <i>null</i> es un valor válido. indica que todos los items son del mismo tipo.
span	Se representa como una lambda que recibe como parámetro el elemento. Define cuántas filas ocupará cada elemento a mostrar. Por defecto vale 1 (un elemento ocupa una fila).

```
val items = (0..50).map { Item(value = it) }
Box( contentAlignment = Alignment.Center )
{
    LazyHorizontalGrid(modifier = Modifier.fillMaxSize(),
        rows = GridCells.FixedSize(120.dp),
        contentPadding = PaddingValues(all = 16.dp),
        horizontalArrangement = Arrangement.Center,
        verticalArrangement = Arrangement.Top,
    )
}
{
    items(items= items,
        span= {item-> GridItemSpan(if (item.value % 4 == 0) 2
            else 1
        )
    }
    item -> ShowItem(item)
}
}

@Composable
fun ShowItem(item: Item)
{
    Card(modifier = Modifier.padding(bottom=4.dp,
        end = 4.dp))
    {
        Column(modifier = Modifier.padding(4.dp))
        {
            Text(text = item.title,
                style = MaterialTheme.typography.titleMedium)
            Text(text = item.description,
                style = MaterialTheme.typography.bodySmall)
        }
    }
}
```





# LazyHorizontalGrid

## **content:**

**LazyListScope.(.) -> Unit** → Indica el contenido que se mostrará en el LazyGrid.

```
fun <T> LazyGridScope.itemsIndexed(items: List<T>,
    key: ((index: Int, item: T) -> Any)? = null,
    contentType: (index: Int, item: T) -> Any? = {null},
    itemContent: @Composable LazyItemScope.(index: Int, item: T) -> Unit
)
```

## *Añade una lista de items al LazyColumn*

items	Representa una lista conteniendo los datos a mostrar.
itemContent	El composable en forma de lambda que recibe como parámetro el índice del elemento y el elemento de la lista
key	representa el identificador del item dentro del LazyGrid. Si no se especifica, su valor será el <i>index</i> del item (posición que ocupa el item dentro de la lista). La key puede ser <b>estable</b> o <b>no estable</b> . Se considera <b>estable</b> si su valor se mantiene inmutable entre recomposiciones (por ejemplo la propiedad id de un entity). Se considera no estable si su valor procede de un valor mutable o del index del item. Los estados creados con <i>remember</i> o <i>rememberSaveable</i> dentro de <i>content</i> se asocian a la key del item. Si el key es <b>estable</b> y único, Compose podrá reutilizar el estado guardado entre recomposiciones o cambios en la lista. Si el key es <b>no estable</b> se recrearán los estados internos (es decir, se reinician los valores recordados) Se representa como una lambda que recibe como parámetros el index del elemento y el elemento.
contentType	su valor representa el tipo del content. Las composiciones de items del mismo tipo podrían reutilizarse más eficientemente. <b>null</b> es un valor válido. indica que todos los items son del mismo tipo, Se representa como una lambda que recibe como parámetros el index del elemento y el elemento.
span	Define cuántas columnas ocupará cada elemento a mostrar. Por defecto vale 1 (un elemento ocupa una columna).. Se representa como una lambda que recibe como parámetro el elemento.