91. Listas, Grid y Menus

(En Android Developer AD)

En este apartado se incluye: * LazyColumn y LazyRow * Grid (cuadrícula)

91.1 Listas Diferidas (Lazy List)

Para mostrar listas, utilizamos LazyColumn o LazyRow. Estos componentes son similares a RecyclerView en el mundo de Android clásico, pero más fáciles de utilizar. Debemos proporcionar la lista de elementos y definir cómo se debe mostrar cada elemento. Por ejemplo:

```
LazyColumn {
    items(items) { item ->
        Text("Item: $item")
    }
    item {
        Text("otro")
    }
}
```

`items` Agreda una colección de elementos. `item` un elemento individual

91.2 Cuadrículas diferidas (LazyGrid)

`LazyVerticalGrid` muestra los elementos en una cuadrícula con deslazamiento vertical que ocupa varias columnas

El parámetro `columns` en `LazyVerticalGrid` controla el modo en que se forman las celdas en columnas. En el siguiente ejemplo, se muestran elementos de una cuadrícula con GridCells.Adaptive para que cada columna tenga al menos 128.dp ancho:

```
@Composable
fun PhotoGrid(photos: List<Photo>) {
    LazyVerticalGrid(
        columns = GridCells.Adaptive(minSize = 128.dp)
    ) {
        items(photos) { photo ->
            PhotoItem(photo)
        }
    }
}
```

Se usaran tantas columnas como se puedan incluir con los 128.dp de cada columna.
También es posible forzar un número de columnas fijo, adaptado el ancho , con `GridCells.Fixed` .

Se puede personalizar el diseño de columnas con `span` y `maxLineSpan` utilizandos con `item` e `items`

```
LazyVerticalGrid(
    // ...
) {
    item(span = {
        // LazyGridItemSpanScope:
        // maxLineSpan
        GridItemSpan(maxLineSpan)
    }) {
        CategoryCard("Fruits")
    }
    // ...
}
```

91.3 Padding del contenido

Para añadir espacio entre los bordes y el contenidos usamos PaddingValues con el parámetro `contentPadding`

```
LazyColumn(
    contentPadding = PaddingValues(horizontal = 16.dp, vertical = 8.dp),
) {
    // ...
}
```

91.4 Espaciado de contenido.

Para agregar espaciado entre elementos.

```
// solamente vertical
LazyColumn(
    verticalArrangement = Arrangement.spacedBy(4.dp),
) {
    // ...
}
// solamente horizontal
LazyRow(
    horizontalArrangement = Arrangement.spacedBy(4.dp),
) {
    // ...
}
```

y en los Grid tanto horizontal como vertical:

```
LazyVerticalGrid(
    columns = GridCells.Fixed(2),
    verticalArrangement = Arrangement.spacedBy(16.dp),
    horizontalArrangement = Arrangement.spacedBy(16.dp)
) {
    items(data) { item ->
        Item(item)
    }
}
```

91.5 Claves de elementos

La posición determina el elemento pulsado o seleccionado. Esto puede dar problemas si se reordenan las listas.
Para evitar este problema se pueden utilizar una clave (key) para cada elemento (como parámetro de items)

```
@Composable
fun MessageList(messages: List<Message>) {
    LazyColumn {
        items(
            items = messages,
            key = { message ->
                // Return a stable + unique key for the item
                message.id
            }
        ) { message ->
            MessageRow(message)
        }
    }
}
```

(mas detalles aqui)

91.6 Animar elementos.

Se utiliza el modificador animateItemPlacement en el elemento del contenido:

```
LazyColumn {
    items(books, key = { it.id }) {
        Row(Modifier.animateItemPlacement()) {
            // ...
        }
    }
}
```

91.7 Encabezados fijos (Experimental)

(Continuar aqui)

Apendice

Versión 0.5 (20-12-23)

¿Fue útil esta página?

☺ ☹