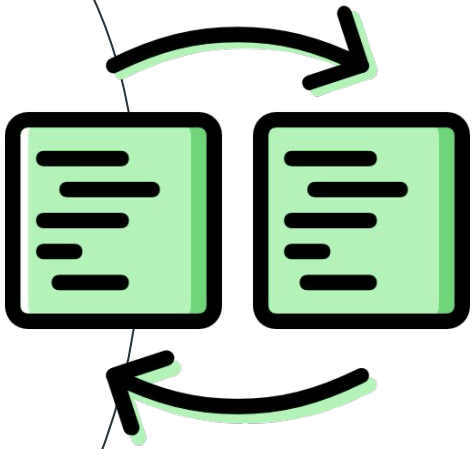
Anahí Salgado

@ANNCODE

State Hoisting with Jetpack Compose

@anncode

State



Cualquier valor que pueda cambiar en el tiempo.

archivo

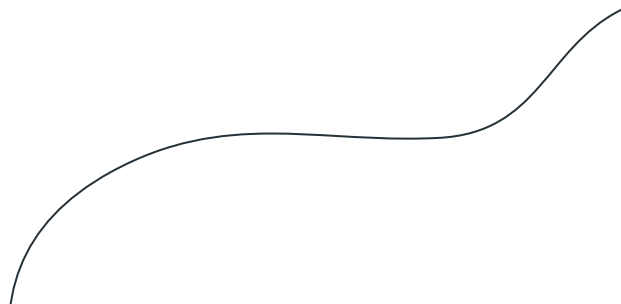
variable

Composables

```
@Composable
fun CounterContent() {
    Row {
        Button(onClick = { }) {
            Text(text = "+")
        }

        Text(text = "1")

        Button(onClick = { }) {
            Text(text = "-")
        }
    }
}
```

State

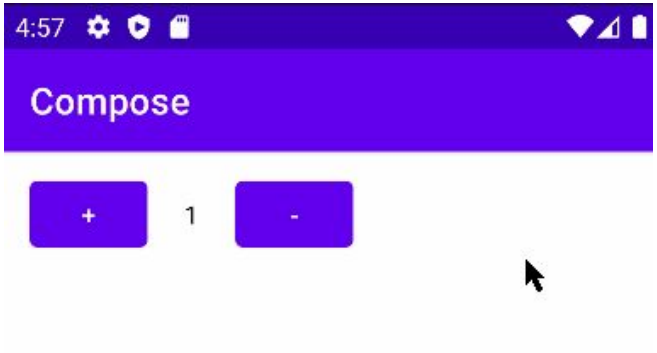
```
var quantity = 0
```



State

```
var quantity = 0
```

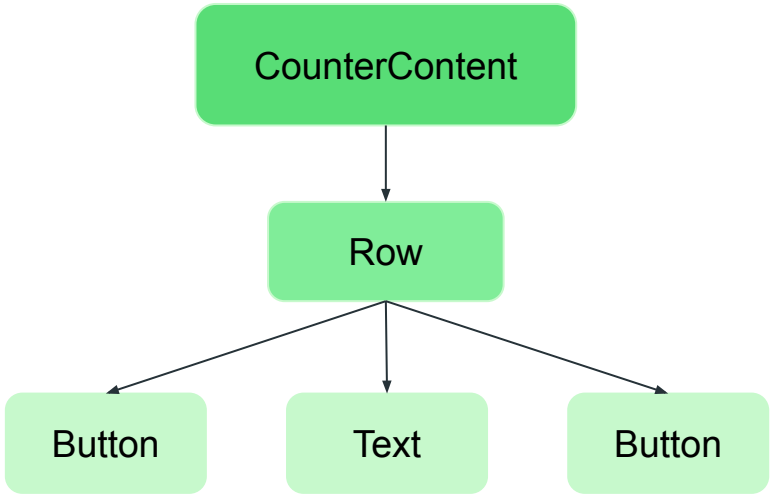
Composables




```
@Composable
fun CounterContent() {
    var quantity = 0
    Row {
        Button(onClick = { quantity++ }) {
            Text(text = "+")
        }
        Text(text = quantity)
        Button(onClick = { quantity++ }) {
            Text(text = "-")
        }
    }
}
```

Recomposition

Recomposition Tree

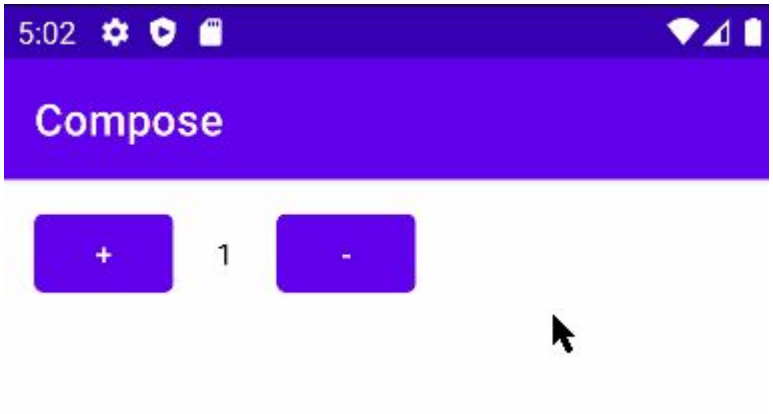



```
@Composable
fun CounterContent() {
    var quantity = 0
    Row {
        Button(onClick = { quantity++ }) {
            Text(text = "+")
        }
        Text(text = "")
        Button(onClick = { quantity++ }) {
            Text(text = "-")
        }
    }
}
```

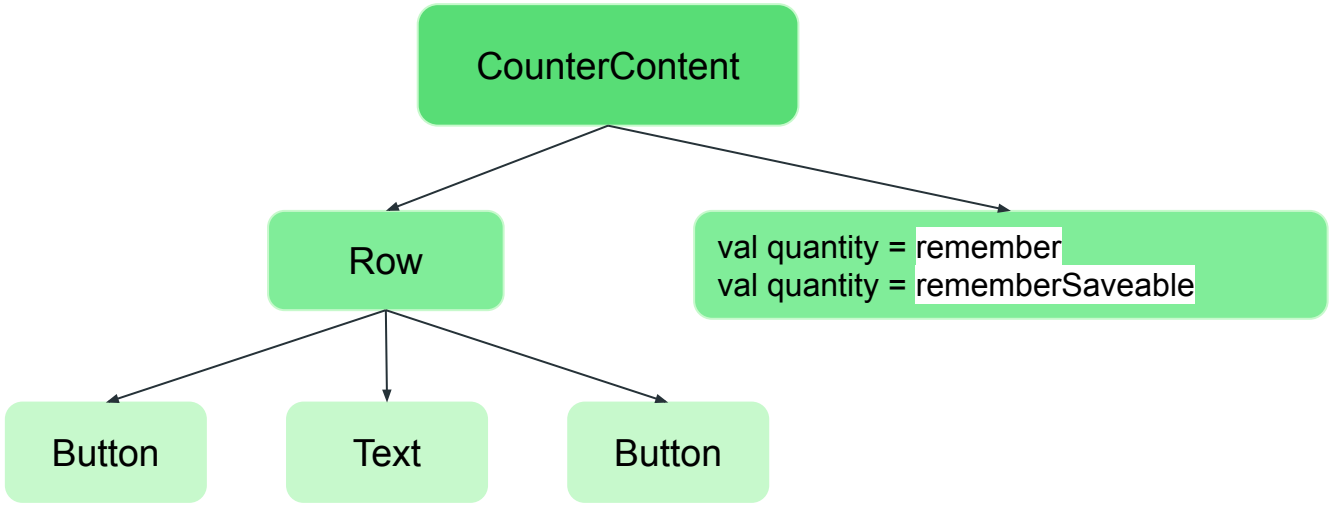
State en Compose

```
val quantity = remember { mutableStateOf(1)}
```

```
val quantity = rememberSaveable { mutableStateOf(1)}
```

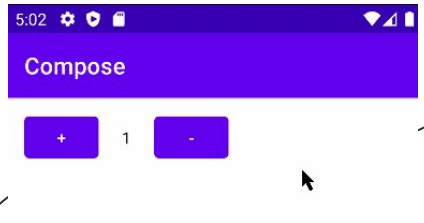


Recomposition Tree



Composables

```
@Composable
fun CounterContent() {
    val quantity = rememberSaveable { mutableStateOf(1)}
    Row {
        Button(onClick = { quantity.value++ }) {
            Text(text = "+")
        }
        Text(text = quantity)
        Button(onClick = { quantity.value-- }) {
            Text(text = "-")
        }
    }
}
```

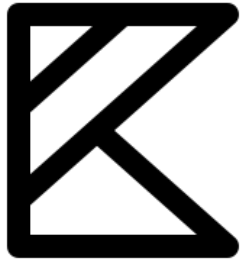


Programación Funcional

Inmutabilidad

- Ausencia del estado mutable

Composables Inmutables



- Ausencia del estado mutable

~~archivo~~
~~variable~~

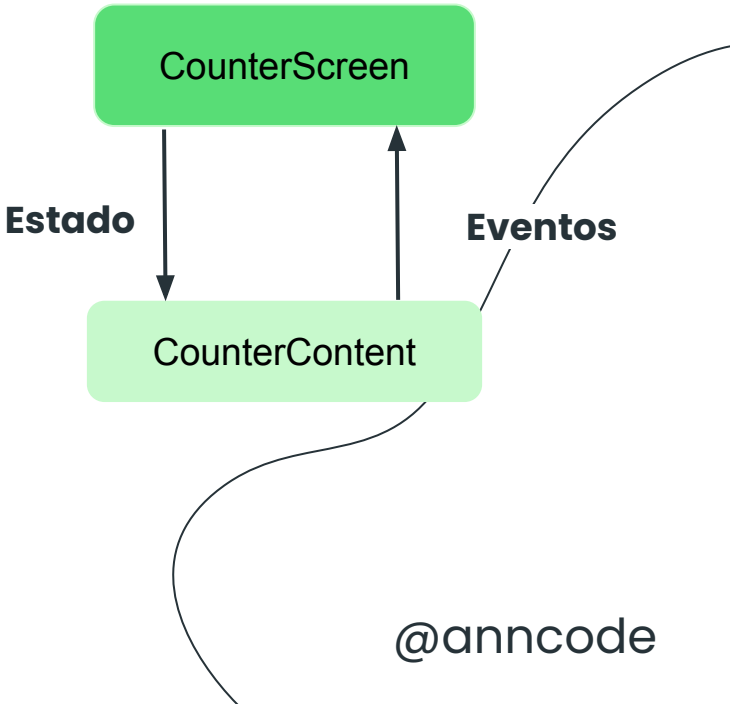
State Hoisting

State Hoisting

- **Patrón de Diseño**

  Estado va hacia abajo

  Eventos hacia arriba



Elevar el Estado

Composables

```
@Composable
fun CounterContent() {
    val quantity = rememberSaveable { mutableStateOf(1)}
    Row {
        Button(onClick = { quantity.value++ }) {
            Text(text = "+")
        }
        Text(text = "")
        Button(onClick = { quantity.value-- }) {
            Text(text = "-")
        }
    }
}
```

Composables

```
@Composable
fun CounterContent() {
    val quantity = rememberSaveable { mutableStateOf(1)}
    Row {
        Button(onClick = { quantity.value++ }) {
            Text(text = "+")
        }
        Text(text = "")
        Button(onClick = { quantity.value-- }) {
            Text(text = "-")
        }
    }
}
```

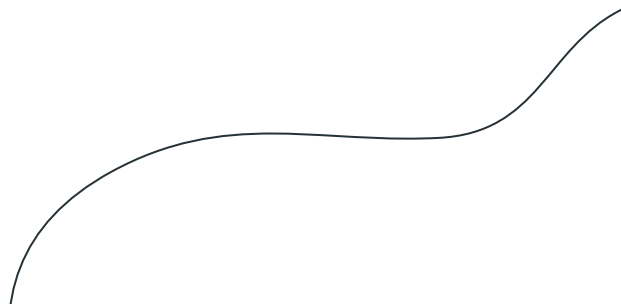
Composables

```
@Composable
fun CounterContent() {
    ____________________
    Row {
        Button(onClick = { ___________ }) {
            Text(text = "+")
        }
        Text(text = "")
        Button(onClick = { ___________ }) {
            Text(text = "-")
        }
    }
}
```

State Holder

ViewModel ✨

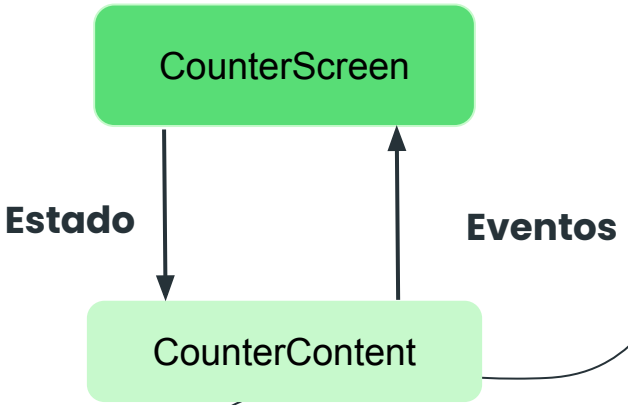
```
@Composable
fun CounterContent(
    quantity: Int,
    incrementQuantity: () -> Unit,
    decrementQuantity: () -> Unit,
) {









}
```



```
@Composable
fun CounterContent(
    quantity: Int,
    incrementQuantity: () -> Unit,
    decrementQuantity: () -> Unit,
) {
    Row {
        Button(onClick = { incrementQuantity() }) {
            Text(text = "+")
        }
        Text(text = quantity.toString())
        Button(onClick = { decrementQuantity() }) {
            Text(text = "-")
        }
    }
}
```

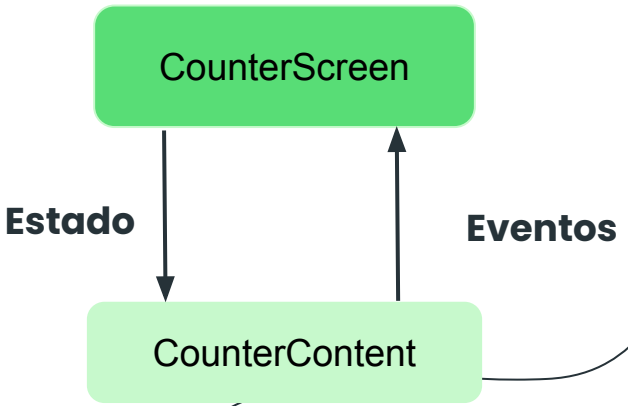
State Holder

```
@Composable
fun CounterScreen() {
    val quantity = rememberSaveable { mutableStateOf(1) }
    CounterContent(
        quantity = quantity.value,
        { quantity.value++ },
        { quantity.value-- }
    )
}
```



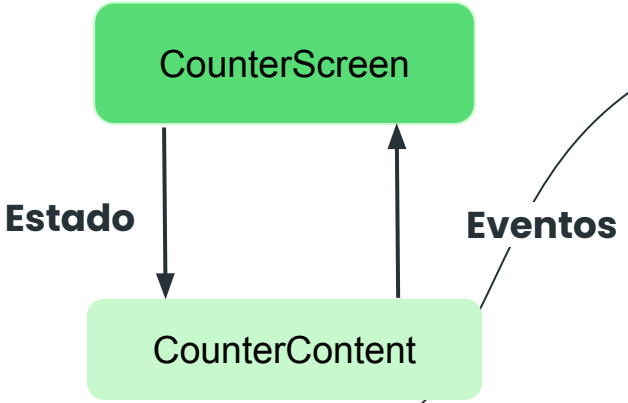
ViewModel

```
@Composable
fun CounterScreen(counterViewModel: CounterViewModel) {
    val quantity by counterViewModel.quantity.observeAsState(1)
    CounterContent(
        quantity = quantity,
        { quantity.increment() },
        { quantity.decrement() }
    )
}
```



State Hoisting

- **Patrón de Diseño**
  - Nos ayuda a separar responsabilidades.
  - Crear Composables más inmutables y fáciles de testear



@anncode

anahisalgado.com/cursos

@anncode

Anahí Salgado

@ANNCODE