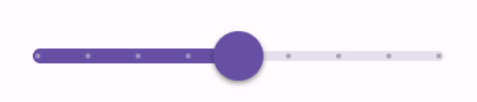
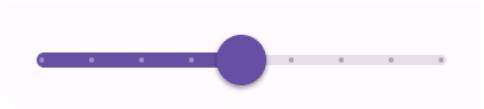
Slider

Compose

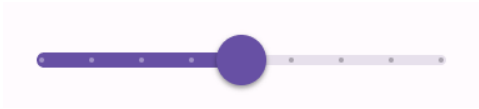


Slider



```
@Composable
fun Slider(
    value: Float,
    onValueChange: (Float) -> Unit,
    modifier: Modifier = Modifier,
    enabled: Boolean = true,
    valueRange: ClosedFloatingPointRange<Float> = 0f..1f,
    @IntRange(from = 0) steps: Int = 0,
    onValueChangeFinished: (() -> Unit)? = null,
)
```

Slider



value:

Float → Indica el valor actual del Slider.

valueRange:

ClosedFloatingPointRange<Float> = 0f..1f, → Indica el rango de valores que puede tener el Slider.

```
Slider(value = 33,
      valueRange = 0f..255f,
      ...
     )
```

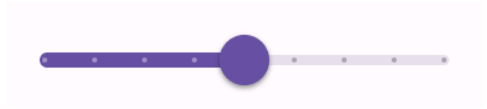
onValueChange:

() -> Unit → lambda que se ejecuta cuando el usuario modifica el valor del Slider al hacer *drag*.

```
var sliderValue by remember( mutableState(33) )

Slider(value = sliderValue,
      valueRange = 0f..255f,
      onValueChange = { newValue -> sliderValue = newvalue
                              : : :
                },
     )
```

Slider



steps:

Int = 0, → se utiliza para definir el número de intervalos entre el valor mínimo y el máximo. Esto significa que, en lugar de permitir cualquier valor dentro del rango (Slider continuo), el Slider se "redondeará" a los valores intermedios definidos por los pasos. Por ejemplo:

- Si se define steps = 0, el Slider será continuo y podrá tomar cualquier valor en el rango.
- Si se define steps = 4 en un Slider cuyo rango es de 0 a 100, el Slider tomará 5 valores posibles: 0, 25, 50, 75 y 100.

```
Slider(value = 0,
    valueRange = 0f..99f,
    steps = 10,
    ...
   )
```

Valores posibles: 0, 9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99

enabled:

Boolean = true → Indica si el *Slider* está habilitado.

onValueChangeFinished:

() -> Unit → Se ejecuta cuando el usuario finaliza la interacción con el Slider. Si `onValueChange` se llama cada vez que el valor cambia (por ejemplo, al arrastrar el Slider), `onValueChangeFinished` se invoca una vez que el usuario suelta el control.