EL COMPONENTE TEXTFIELD CON STATE

EL COMPONENTE TEXTFIELD CON STATE

- El componente **TextField** es el equivalente al componente **EditText** del framework de Android tradicional.

- En esta lección veremos cómo manejar correctamente el estado de este componente a través de **State**.

- Vamos a iterar nuestra aplicación de añadir alumnos incorporando un campo de introducción de texto **TextField** que permita al usuario escribir el nombre del alumno.

EL COMPONENTE TEXTFIELD CON STATE

- **TextField** acepta dos valores principales:
    - **value: String**

      El valor que mostrará el componente.
    - **onValueChange: (String) -> Unit**

      Lambda que devuelve el valor que tiene el componente cada vez que cambia.

EL COMPONENTE TEXTFIELD CON STATE

Cuando usamos **TextField**, es prácticamente obligatorio hacerlo de la mano de **State** de forma que podamos ver cómo el valor del componente cambia cada vez que se introduce texto.

```
val newStudentState = remember { mutableStateOf( value: "")}
TextField(
    value = newStudentState.value,
    onValueChange = {
        newInput -> newStudentState.value = newInput
    }
)
```

EL COMPONENTE TEXTFIELD CON STATE

- Se usa **mutableStateOf** para guardar el estado del componente **TextField.**

- Podríamos incorporar este snippet de código en nuestro componente **StudentList** pero implementaremos **State Hoisting** para no manejar estados en componentes internos.

```
val newStudentState = remember { mutableStateOf( value: "")}
TextField(
    value = newStudentState.value,
    onValueChange = {
        newInput -> newStudentState.value = newInput
    }
)
```

EL COMPONENTE TEXTFIELD CON STATE

Parametrizamos **StudentList**:

- **studentName: String**
  - Contiene el valor del **TextField**.
  - Como veremos en **MainScreen**, referencia a un **State**.
- **onStudentNameChange: (String)->Unit**
  - lambda que eleva el valor del componente **TextField** cuando cambia.

```
@Composable
fun StudentList(
    students: List<String>,
    onButtonClick: () -> Unit,
    studentName: String,
    onStudentNameChange: (String) -> Unit
) {
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        for (student in students) {
            StudentText(name = student)
        }
        TextField(
            value = studentName,
            onValueChange = onStudentNameChange
        )
        Button(
            onClick = onButtonClick
        ) { this: RowScope
            Text(text = "Add new student")
        }
    }
}
```

EL COMPONENTE TEXTFIELD CON STATE

MainScreen:

- **newStudentState: MutableState** El valor de **TextField** es un **State** y todas las variaciones que se produzcan sobre él dispararán la recomposición.
- Vemos como en las lambdas **onButtonClick** y **onStudentNameChange** se inserta un valor en la lista de estudiantes y se modifica el valor del componente **TextField** respectivamente.

```
@Composable
fun MainScreen() {
    val studentsState = remember { mutableStateListOf("Esther", "Jaime") }
    val newStudentState = remember { mutableStateOf( value: "") }
    Surface(
        color = Color.LightGray,
        modifier = Modifier.fillMaxSize()
    ) {
        StudentList(
            studentsState,
            { studentsState.add(newStudentState.value) },
            newStudentState.value,
            { newStudent -> newStudentState.value = newStudent }
        )
    }
}
```