EL PATRÓN STATE HOISTING

OpenWebinars

EL PATRÓN STATE HOISTING

- El patrón **State Hoisting** consiste en mover los estados al componente padre de tal forma que los hijos nunca tengan que manejarlos.

- El principal objetivo es reemplazar la variable de estado por dos argumentos en cada función composable hija:

    - **value: T** El valor para mostrar.

    - **onValueChange: (T) -> Unit** Evento (lambda) que dispara la modificación del **State.**

EL PATRÓN STATE HOISTING

El patrón **State Hosting** ofrece las siguientes ventajas:

- Manejar los estados de forma única y centralizada.
- Solo las funciones que manejan estados pueden modificarlos.
- Funciones composable hijas no tienen que preocuparse por manejar estados, solo pintar información y elevar eventos.

EL PATRÓN STATE HOISTING

Vamos a aplicar el patrón **State Hoisting** a la aplicación de alumnos de la lección anterior:

- Mover la lista de estudiantes al punto de entrada **MainScreen**.
- Parametrizar la función **StudentList** con el valor a mostrar y la función lambda de eventos de click.

EL PATRÓN STATE HOISTING

StudentList

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

EL PATRÓN STATE HOISTING

MainScreen

```
@Composable
fun MainScreen() {
    val studentsState = remember { mutableStateListOf("Esther", "Jaime") }
    Surface(
        color = Color.LightGray,
        modifier = Modifier.fillMaxSize()
    ) {
        StudentList(studentsState) {
            studentsState.add("Miguel")
        }
    }
}
```

EL PATRÓN STATE HOISTING

- El componente **StudentList** ya no sabe nada sobre estados.
- Le hemos aplicado las dos premisas del patrón **State Hoisting**, la parametrización de la lista de estudiantes y la función para elevar los eventos de click del botón añadir.
- Ahora es el componente **MainScreen** el encargado de manejar estados y de modificarlos.