**Sesión 1-05 Clase del 10 de octubre**

En esta sesión:

* Conoceremos el ciclo de vida de una activity y las funciones que son invocadas en ese ciclo de vida .
* Usaremos el componente TextField.
* Usaremos el estado de variables para modificar propiedades de los componentes.

1.- Creamos un nuevo proyecto. En la clase MainActivity ya tenemos un método del ciclo de vida:

**onCreate()**: Es llamada cuando la activity se crea. Aún no se ha pintado en pantalla.

Pero hay otros métodos en el ciclo de vida que actúan cuando una activity cambia de estado (por que la llevamos a segundo plano, la cerramos, cambiamos la orientación del dispositivo, etc.)

2.- Implementaremos los métodos del ciclo de vida y los programaremos para que simplemente escriban un Log indicando el nombre de la función en que se encuentran.

**onStart():**La activity está ya visible pero aún el usuario no puede interactuar con ella. Aquí se pueden inicializar componentes que dependen de la visibilidad de la activity.

**onResume():** La activity ya está en primer plano y se puede interactuar con ella.

**onPause():** La activity pierde el foco, pero aún es visible en el fondo, por ejemplo, cuando aparece un cuadro de diálogo o una **activity** nueva está apareciendo. Aquí se deben pausar procesos en segundo plano, como la reproducción de video.

**onStop():** La activity deja de ser visible para el usuario, por ejemplo, cuando ya está abierta otra activity en pantalla completa. Aquí por ejemplo se liberan conexiones a bases de datos, se guardan datos críticos

**onDestroy():** Justo antes de que la activity sea destruida y eliminada de la memoria.

**onRestart()**: Se llama cuando la activity se está **reiniciando** después de haber sido detenida.

3.- Vamos a crear un nuevo proyecto. En la pantalla debe haber una organización en columna con disposición horizontal y vertical centrada.

Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(8.*dp*), // Padding alrededor de la columna  
 verticalArrangement = Arrangement.Center,  
 horizontalAlignment = Alignment.CenterHorizontally  
)**{**

En la columna se dispondrán los componentes:

* Dos TextField para introducir el nombre y los apellidos. Ocupan todo el ancho de pantalla y, en vertical, el espacio que necesiten. Usan un tamaño de letra de 20.sp.
* Un Button para **aceptar** los datos introducidos. Ocupa el espacio que necesita.

Button(  
 onClick = {},  
  
) **{** Text(texto, fontSize = 20.*sp*)  
**}**

* Un Text para escribir el nombre completo (lo que hay en mensaje) cuando se ha aceptado al pulsar el Button. Ocupa todo el ancho de pantalla y, en vertical, el espacio que necesite. Usa un tamaño de letra de 20.sp.

Text(  
 text = "Hola, $nombre $apellidos",  
 fontSize = 20.*sp*, // Tamaño de letra 20.sp  
 fontWeight = FontWeight.Bold, // Negrita para destacar el mensaje  
 color = Color.Blue, // Color del texto  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(top = 16.*dp*) // Margen superior  
)

El código para incluir el **TextField** puede ser éste:

TextField( value=nombre,onValueChange =**{**nombre=it**}** ,  
 label=**{**Text("Nombre")**}**,  
 textStyle = LocalTextStyle.current.copy(fontSize = 20.sp),  
 modifier = Modifier.fillMaxWidth())

En la propiedad **value** de un **TextField** normalmente se asigna una variable String. Indica de donde se recibe el valor que se recibe en el componente.

La propiedad **onValueChange** contiene una lambda que se ejecutará cuando se produzca el evento **onValueChange** en el componente. En el caso, se programa que el contenido del **TextField (it)** se asigne a la variable **nombre**.

4.- De momento, si ejecutamos con este código, nos sorprenderá que no podemos introducir texto en el **TextField**.

Pero sigamos dando funcionalidad a los componentes.

Programamos en el **Button** que cuando se clique, se cargue en una variable String de nombre mensaje el texto “**Hola, $nombre $apellidos**”.

5.- Y para completar, establecemos en el componente **Text** que se rellene con el texto que hay en la variable **mensaje**.

6.- Al ejecutar veremos que nada de lo previsto funciona.

Jetpack Compose funciona con un **sistema de estado reactivo**. La UI se actualiza automáticamente solo cuando las “variables de control” son **variables de estado** (State). Cuando una variable es “normal” Jetpack Compose no sabe cuándo cambia su valor y NO “puede” actualizar componentes de la UI.

Además, el componente **TextField** requiere que en **value** se le asigne una variable de estado.

Una **variable de estado** como, por ejemplo la variable nombre, se declara con:

var nombre by remember **{** *mutableStateOf*("")**}**

Donde **remember** indica que se debe recordar el valor durante el ciclo de vida de la pantalla. Con **mutableStateOf** indicamos, aparte del valor inicial de la variable, que la variable puede mutar (modificarse) y puede notificar a los componentes de la pantalla cuando su valor cambia.

Declara las variables apellidos y mensaje como variables de estado y comprueba ahora la correcta funcionalidad.

6.- Organiza el código de los **TextField** para que cada componente se pinte dentro de una función **Composable**.

7.- Añade ahora a la pantalla tres **Buttons** con los textos Rojo, Verde, Azul. Cuando se pulsa cualquiera de esos **Buttons** se pintará el fondo de un **Box** que estará debajo con el color correspondiente al **Button**.