**Sesión 2-10 Clase del 27 de noviembre**

Debes descargar y abrir el proyecto **Sesion2\_10\_Base**.

Este es el resultado de la ejecución de este proyecto que tiene una única pantalla con un **TopAppBar** que incluye un **NavigationIconButton** y varios **IconButtons** *(dentro de actions de la TopAppBar):*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Vamos a crear un menú que se desplegará al pulsar en el **IconButton** de la derecha (el de menú).

1.- Para crear un menú, hay que incluir un componente **DropdownMenu** dentro de **actions** de **TopAppBar.**

// Menú desplegable  
DropdownMenu(  
 expanded = true,  
 onDismissRequest = **{ }**) **{**

**}**

El valor en **expanded** indicará si el menú está desplegado o no. En **onDismissRequest** se puede programar lo que se realiza cuando se clica fuera del menú estando desplegado.

2.- Las posiciones que contiene un menú se cargan dentro del scope de **DropdownMenu** mediante componentes **DropdownMenuItem**. Por ejemplo, para cargar una opción de menú **Previsualizar:**

DropdownMenu(  
 expanded = true,  
 onDismissRequest = **{ }**) **{** DropdownMenuItem(  
 text = **{** Text("Previsualizar") **}**,  
 onClick = **{**// Acción para Previsualizar**}** )

}

Si probamos la ejecución, veremos que se despliega el menú pero:

* No se despliega al pulsar el botón **Menú.**
* Permanece fijo en pantalla.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

3.- Para controlar el estado de desplegado del menú y el botón que lo despliega, hay que usar una variable de estado booleana. La definimos en nuestra función composable **TopMenuBar:**

// Estado para controlar la visibilidad del menú desplegable  
var showMenu by remember **{** *mutableStateOf*(false) **}**

En la propiedad **expanded** del componente **DropdownMenu** asignamos el valor de la variable de estado.

DropdownMenu(  
 expanded = showMenu,  
 onDismissRequest = **{ }**)

Para que el botón del menú despliegue el menú, en **onClick** del **IconButton** del menúasignamos el valor true a la variable de estado.

IconButton(onClick = **{** showMenu=true **}**) **{**

Podemos probar el funcionamiento y veremos que, al pulsar en el botón del menú, pero éste permanece fijo en pantalla.

4.- Programamos lo que se realiza cuando se pulsa en la opción de menú Previsualizar y que, al haber hecho la selección, desaparezca el menú de pantalla.

Dentro de la propiedad **onClick** del componente **DropdownMenuItem** programamos lo que se realizará y que la variable de estado de desplegado del menú tenga el valor false**:**

DropdownMenuItem(  
 text = **{** Text("Previsualizar") **}**,  
 onClick = **{** Log.d("PRUEBA","seleccionado previsualizar")  
 showMenu = false  
 **}**)

Podemos probar la funcionalidad, pero veremos que, desplegado el menú, si clicamos fuera del menú, no se cierra el menú. Para tener esta funcionalidad, hay que asignar false a la variable de estado en la propiedad **onDismissRequest** del componente **DropdownMenu.**

5.- Añade ítems de menú para que se muestre el siguiente menú y que tenga la funcionalidad correcta. Por ahora al pulsar cada opción se enviará un mensaje Log adecuado.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

6.- Vamos queremos que se vea una línea divisora antes del item Denunciar.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Esto es fácil, entre los ítems Descargar y Denunciar incluiremos un componente:

HorizontalDivider()

El aspecto de la línea divisora lo podemos cambiar con sus propiedades, por ejemplo:

HorizontalDivider(thickness = 3.*dp*,  
 color = Color.Blue)

7.- Si queremos que una opción de menú este deshabilitada, asignaremos false a su propiedad **enabled.**

enabled = false

Deshabilitamos la opción **Descargar** para probar. Si se quiere tener las opciones habilitadas o deshabilitadas en función de alguna condición, habrá que usar variables de estado.

8.- Si queremos que, a la izquierda de una opción de menú se muestre un icono, habrá que usar su propiedad **leadingIcon.** Por ejemplo, para la opción **Compartir:**

leadingIcon = **{** Icon(  
 imageVector = Icons.Filled.*Share*,  
 contentDescription = "Compartir icono"  
 )  
**}**

9.- Un menú se puede asociar a cualquier componente y en cualquier área de la pantalla. Vamos a ver, por ejemplo, como se crea un menú contextual sobre un componente Text. Primero, en **ContentHomeView**:

* Cambia el componente Box por un componente Column.
* Añade un Text a la columna para que represente el texto de una variable de estado **myText.**
* Establece que el Text sea clikable.

10.- Haz lo necesario para que, al hacer clic en el Text, se despliegue el menú correspondiente al siguiente código:

DropdownMenu(  
 expanded = showMenu,  
 onDismissRequest = **{** showMenu = false **}**) **{** DropdownMenuItem(  
 text = **{** Text("Borrar") **}**,  
 onClick = **{** showMenu = false  
 *Log.d*("MENU", "Opción Borrar seleccionada")  
 **}** )  
 DropdownMenuItem(  
 text = **{** Text("Duplicar") **}**,  
 onClick = **{** showMenu = false  
 *Log.d*("MENU", "Opción Duplicar seleccionada")  
 **}** )  
 DropdownMenuItem(  
 text = **{** Text("Editar") **}**,  
 onClick = **{** showMenu = false  
 *Log.d*("MENU", "Opción Editar seleccionada")  
 **}** )  
 DropdownMenuItem(  
 text = **{** Text("Asignar") **}**,  
 onClick = **{** showMenu = false  
 *Log.d*("MENU", "Opción Asignar seleccionada")  
 **}** )  
**}**

Para que el menú se despliegue justo debajo del texto, puedes incluir el texto y el menú dentro de un componente Box.

11.- Vamos a ver como se implementa la respuesta a otras acciones de clic sobre la pantalla o sobre componentes de la pantalla. En este caso vamos a desplegar el menú cuando se haga un “clic largo” sobre el texto.

Hay que reemplazar el modifier **clickable** por **combinedClickable**. Dentro de **combinedClickable** programamos las respuestas a las acciones de clic que queramos incluir, en nuestro caso solo para **onLongClick**.

12.- Vamos a crear un cuadro de diálogo que se abre cuando clicamos la opción de menú Editar. Antes de crear el cuadro de diálogo:

* Define en **ContentHomeView** una variable de estado **showDialog** para controlar si el cuadro está desplegado o no.
* Define otra variable de estado **inputText** que contenga un texto vacío. El objetivo de esta variable es usarla en un TextField para recoger un texto que se cargará en el componente Text de prueba.
* Establece que al seleccionar la opción de menú **Editar** se vaya a desplegar el cuadro de diálogo que haremos. Poner **showDialog** a true.

13.- A continuación, tienes el código para implementar el siguiente cuadro de diálogo y sus funcionalidades para que al confirmar se modifique el contenido del texto de prueba:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

if (showDialog) {  
 AlertDialog(  
 //Cuando se clica fuera del cuadro de diálogo  
 onDismissRequest = **{** showDialog = false **}**,  
 title = **{** Text(text = "Editar texto") **}**,  
 text = **{** Column **{** TextField(  
 //Una variable de estado iniciada a ""  
 value = inputText,  
 onValueChange = **{** inputText = **it }**,  
 label = **{** Text("Nuevo texto?") **}** )  
 **}  
 }**,  
 //Definición del botón confirmar y acciones que realiza  
 confirmButton = **{** TextButton(onClick = **{** myText = inputText // Actualizar el texto principal  
 showDialog = false  
 **}**) **{** Text("Aceptar")  
 **}  
 }**,  
 //definición del botón de cancelación y acciones  
 dismissButton = **{** TextButton(onClick = **{** showDialog = false **}**) **{** Text("Cancelar")  
 **}  
 }** )  
}

Fíjate en que, el componente **AlertDialog** se muestra cuando la variable **showDialog** está a true y que, dentro del **AlertDialog** se controla el cierre del cuadro de diálogo.