**Sesión 2-04 Clase del 5 de noviembre**

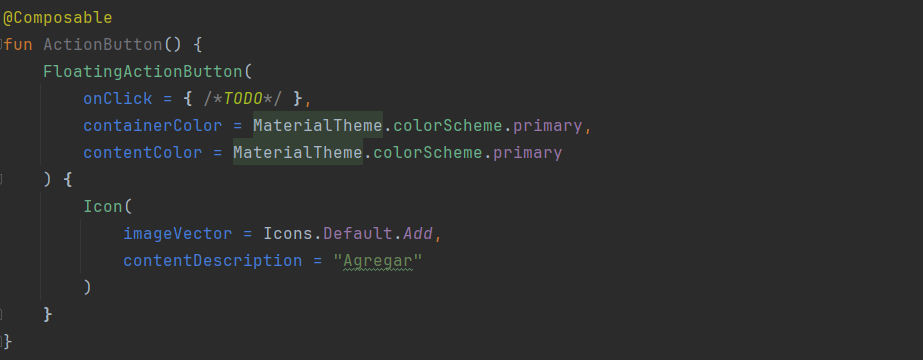
Continuamos con el proyecto de la sesión anterior para un **Scaffold**.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Principalmente, en esta sesión se trata de aprender a implementar navegación entre pantallas.

1.- Vamos a añadir al **Scaffold** un **Floating Action Button**. Antes de ellos vamos a componer este componente que ya hemos visto anteriormente. Lo vamos a desarrollar en una función componible **ActionButton** dentro del archivo de componentes **BarComponents**.



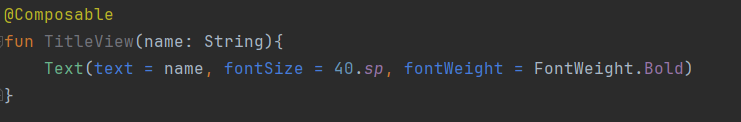
Y en **HomeView**, asignamos esa función a la propiedad **floatingActionButton** de **Scaffold**.

Imagen que contiene Rectángulo

Descripción generada automáticamente

2.- Ahora agregaremos funciones composable en BodyComponents para pintar:

* El texto del contenedor del Scaffold
* Un espaciador vertical
* Un Button



Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

3.- Hacemos que en la columna del container de **Scaffold** se pinten los componentes anteriores:

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Para pasar la lambda a **MainButton** también puedes usar esto (si te resulta más claro:



4.- Diseñamos la vista **DetailView** con el objetivo de dotar después de navegación a la app.

Texto

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

5.- Incluiremos navegación entre pantallas.

a.- Antes de implementar la navegación necesitamos incluir una dependencia (en archivo **Gradle de módulo**):

*implementation*("androidx.navigation:navigation-compose:2.6.0-alpha01")

b.- Editamos el archivo **NavManager** (de navigation).

Creamos una función **composable NavManager**.

En la función definimos una variable de estado para controlador de navegación.

Añadimos un componente **NavHost**. En **NavHost** establecemos la variable de estado para controlar la navegación. También asociamos rutas a funciones de composición de pantallas.

Texto

Descripción generada automáticamente

c.- Tenemos que enlazar **NavManager** con **HomeView** y con **DetailView**. Para ello, tanto en la función **HomeView**() como en la función **DetailView**() se tiene que recibir un objeto **NavController** (la variable de estado declarada en **NavManager**).

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Pero ahora tenemos un error de referencia a esas funciones desde **NavManager**. Lo arreglamos en **NavManager** inyectando en la llamada a esas funciones el objeto variable de estado **navController**.

Texto

Descripción generada automáticamente

d.- Por último, debemos dar funcionalidad a los botones de las pantallas dedicados a navegar de una pantalla a la otra. Lo que hay que realizar es relativamente simple:

En la función **ContentHomeView** tenemos el **onClick** para el botón que hará navegar a la pantalla **DetailView**:

* Esa función debe recibir por parámetro la variable de estado **navController**.
* En el **onClick** para el botón, usaremos **navController** para navegar a la ruta de **DetailView**.

Texto

Descripción generada automáticamente



Podemos probar que esto ahora funciona. Hay que cambiar en **MainActivity** la llamada a Home() por una llamada a la función **NavManager**().

e.- Hay que implementar que desde **DetailView**, pulsando el botón de retorno, se vuelva a **HomeView**. Se podría hacer igual que lo anterior pero no es adecuado, para volver atrás no se debe usar **navigate** a la ruta. Esto es porque todas estas llamadas las va almacenando Android en una pila en memoria y si hacemos mucha navegación, se producen muchos problemas, el que se ocupe más memoria es un problema menor.

Lo que se debe hacer es una llamada a un método volver atrás que, además, elimina automáticamente en la pila el **navigate** anterior. En **onClick** del Botón de **DetailView** tenemos que programar:

Texto

Descripción generada automáticamente

6.- No es nada normal tener botones de volver atrás como el anterior. Lo más normal es que estos botones se encuentren dentro de la Top Bar.

Vamos a crear un **IconButton** para añadirle después a la Top Bar de **DetailView**.

a.- Lo definimos en nuestro archivo de componentes **BarComponents**:

Texto

Descripción generada automáticamente

b.- Incluimos el componente en nuestra Top App Bar:

Texto

Descripción generada automáticamente