AWAKELAB

#programmingbootcamp

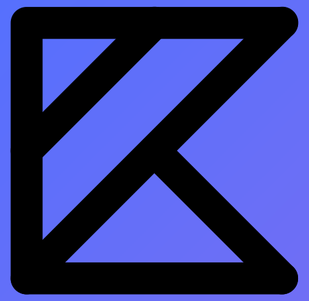
NAVEGACIÓN

¿QUÉ VAMOS A VER?

- Estructura de carpetas.
- Scaffold.
- TopBar.
- Navegación.
- Envío de parámetros.

¡VAMOS A COMENZAR!

AWAKELAB



ESTRUCTURAS DE CARPETAS

AWAKELAB

CARPETAS

- Es importante entender que no por siempre podemos estar agregando todas las funcionalidades dentro del archivo *MainActivity.kt*

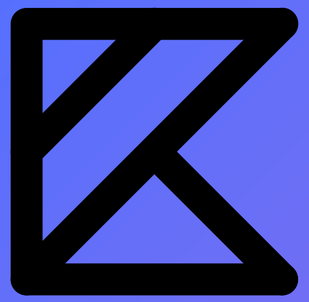
- Lo ideal es que dicho archivo reciba la navegación y algo de lógica generalizada, pero los detalles deben manejarse en diferentes secciones/carpetas que ayudarán a mantener un orden dentro de nuestros proyectos.

- **componentes:** Carpeta que va a contener los diferentes componentes que vamos a utilizar dentro de un proyecto. Se puede dividir en los espacios que utiliza un Scaffold en la pantalla o en componentes individuales como Buttons.

CARPETAS

- **navigation:** Es la carpeta que va a contener el *NavManager* en dónde alojaremos las rutas de nuestras diferentes vistas en el proyecto.
  También es la carpeta de dónde manejaremos el intercambio de datos entre una vista u otra.

- **views:** En esta carpeta almacenaremos las diferentes vistas que vamos a tener en nuestra aplicación.



SCAFFOLD

AWAKELAB

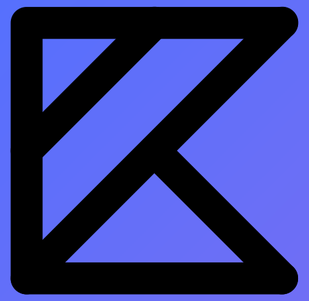
SCAFFOLD

- Es la estructura que actualmente **Jetpack Compose** propone como contenedor principal de cualquier vista/aplicación.

- Tiene la cualidad de seccionarse en tres partes dentro de una vista: **TopBar - Content - BottomBar.**

- Facilita el uso de diferentes estructuras dentro de sus propias vistas, y se integra muy bien con la navegación de **Jetpack Compose.**

- Solo debemos tener en cuenta que algunas de sus funcionalidades aún se encuentran en *“modo de prueba”*, es por eso que deberemos agregar una **etiqueta** que nos permita utilizar esta fase *“experimental”* sin ningún problema.



TOPBAR

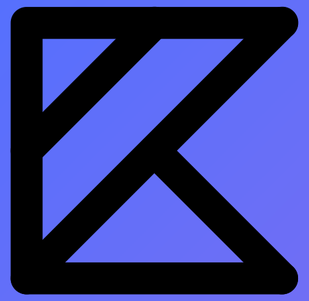
AWAKELAB

TOPBAR

- A pesar de ser una estructura que podría integrarse de manera independiente, converge muy bien con la estructura del **Scaffold** en **Jetpack Compose.**

- **TopBar** nos va a permitir generar esta barra de navegación superior que nos abre la puerta a múltiples funcionalidades como: *Menús desplegables*, *botones de navegación* e incluso *botones de interacción* que generan alguna acción.


```
topBar = {
    TopAppBar(
        title = { TitleBar(name = "Detail view") },
        colors = TopAppBarDefaults.mediumTopAppBarColors(
            containerColor = Color.Blue
        ),
        navigationIcon = {
            MainIconButton(icon = Icons.AutoMirrored.Default.ArrowBack) {
                navController.popBackStack()
            }
        }
    )
}
```



NAVEGACIÓN

AWAKELAB

NAVEGACIÓN

- **NavManager:** Dentro de este archivo podremos manejar las diferentes rutas que va a tener nuestra aplicación.

- Dentro tendremos **composable( )** para declarar y manejar cada una de las rutas, siendo incluso posible diseñarlas como si se tratara de una url, de tal manera que cada elemento añadido se trate de un dato que va a viajar de una vista a otra.

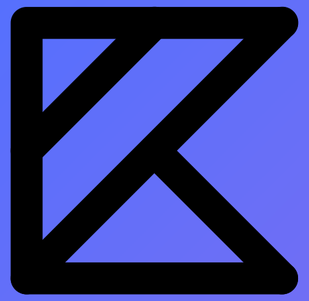
- **startDestination:** Corresponde a la vista que se mostrará por defecto al iniciar la navegación.

```
@Composable
fun NavManager() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = "Home" ) {
        composable( route: "Home") {
            HomeView(navController)
        }
        composable( route: "Detail/{id}/?{opcional}", arguments = listOf(
            navArgument( name: "id") { type = NavType.IntType },
            navArgument( name: "opcional") { type = NavType.StringType }
        )) {
            val id = it.arguments?.getInt( key: "id") ?: 0
            val opcional = it. arguments?.getString( key: "opcional") ?: ""
            DetailView(navController, id, opcional)
        }
    }
}
```

NAVEGACIÓN

- **NavController:** Debemos tener en cuenta que este elemento viajará desde la vista hasta el componente que accionará el cambio de vista, ya que será a través de *navController.navigate(“route”)* que podremos ir de una vista a otra.

- En caso de que no queramos viajar a una vista en especifico e ir directamente a la anterior, en lugar de *.navigate*, agregaremos **.popBackStack( )**



ENVÍO DE PARÁMETROS

AWAKELAB

ENVÍO DE PARÁMETROS

- Para realizar un envío de uno o más parámetros, lo haremos a través de la ruta del **composable**, agregando cada uno de los datos como si se tratara de una url web.

- Dichos argumentos serán almacenados en una lista, cada uno como un **navArgument**, que luego viajará a través del *navController*.

- En las otras vistas lo manejaremos por medio del **name** que le otorgamos en el *navArgument*.

```
composable( route: "Detail/{id}/?{opcional}", arguments = listOf(
    navArgument( name: "id") { type = NavType.IntType },
    navArgument( name: "opcional") { type = NavType.StringType }
)) {
    val id = it.arguments?.getInt( key: "id") ?: 0
    val opcional = it. arguments?.getString( key: "opcional") ?: ""
    DetailView(navController, id, opcional)
}
```

ENVÍO DE PARÁMETROS

- No debemos olvidarnos enviar el parámetro a través del evento *onClick* con el que se conectan ambas vistas, ya que este será el **puente** en el traspaso de datos.

```
MainButton(name = "Detail view", backColor = Color.Red, color = Color.White) {
    navController.navigate( route: "Detail/${id}/?${opcional}" )
}
```

- En este punto solo nos queda recibir los argumentos para manipular los datos dentro del *layout*, es más, podemos agregar lógica extra para considerar los datos que podrían ser nulos.

```
if (opcional == "") {
    Spacer(modifier = Modifier.height(0.dp))
} else {
    TitleView(name = opcional.orEmpty())
}
```

GRACIAS POR LA ATENCIÓN

AWAKELAB