Menús y Navegación

`Navigation Compose`

`ModalNavigationDrawer`

`TopAppBar`

`NavigationBar`

Navigation in Compose

libs.versions.toml

```
[versions]
:::
navigationCompose = "2.8.8"
kotlinxSerialization = "1.2.1"


[libraries]
:::
## Navigation compose
androidx-navigation-compose = { group = "androidx.navigation",
                                name = "navigation-compose",
                                version.ref = "navigationCompose"
                              }

kotlinx-serialization-json = { group = "org.jetbrains.kotlinx",
                               name = "kotlinx-serialization-json",
                               version.ref = "kotlinxSerialization"
                             }

[plugins]
:::
## Navigation compose
kotlin-serialization = { id = "org.jetbrains.kotlin.plugin.serialization",
                         version.ref = "kotlin"
                       }
```

build.gradle.kts

```
plugins {
    : : :
    // Navigation compose
    alias(libs.plugins.kotlin.serialization)
}

dependencies  {
  : : :
// Navigation compose
implementation(libs.androidx.navigation.compose)
implementation(libs.kotlinx.serialization.json)
}
```

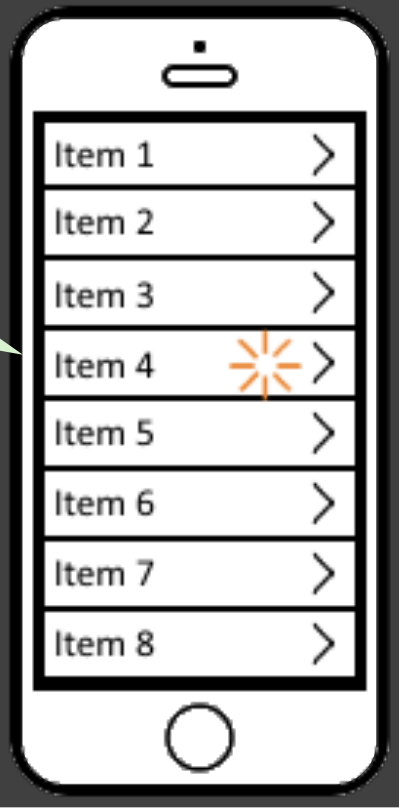
Navigation in Compose

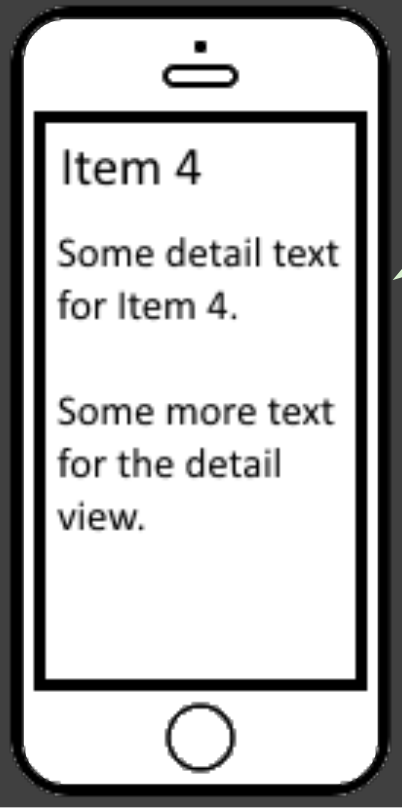
MasterScreen

DetailScreen

MasterScreen muestra a través de un LazyColumn() una lista de items.

Cuando se haga "click" en un determinado item, se navegará a DetailScreen para visualizar el detalle de dicho item.





DetailScreen visualiza el detalle de un determinado item. Cuando se invoque a este @Composable se le debe pasar como parámetro el identificador del item para poder localizarlo.

Al pulsar el botón "back" se regresará a MasterScreen.

La "ruta" de navegación a MasterScreen se representa a través del objeto *@Serializable* **MasterRoute**

```
//Ruta MasterScreen

@Serializable
object MasterRoute
```

La "ruta" de navegación a DetailScreen se representa a través de un objeto de la clase **DetailRoute**. Sus propiedades especifican los datos necesarios para poder navegar (invocar) al screen.

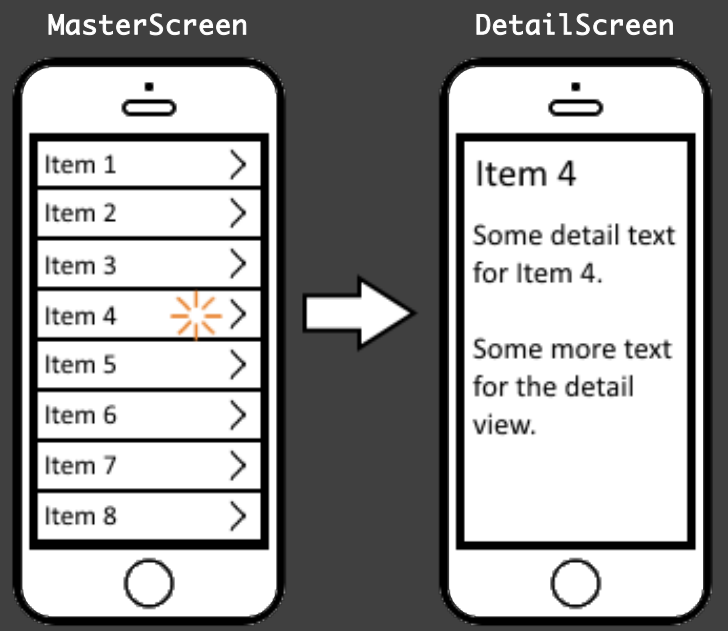
```
//Ruta DetailScreen

@Serializable
data class DetailRoute(val id: Long)
```

Navigation in Compose

```
class MainActivity : ComponentActivity()
{
  override fun onCreate(savedInstanceState: Bundle?)
  {
    super.onCreate(savedInstanceState)
    setContent {
                MainScreen()
               }
  }
}


@Composable
fun MainScreen()
{
  val navController = rememberNavController()

  NavHost(navController = navHostController, startDestination = MasterRoute)
  {
    composable<MasterRoute> {navBackStackEntry-> MasterScreen(navController = navController,
                                                              onItemClick= {itemId-> navController.navigate(route= DetailRoute(itemId)) }
                                                             )
                                }

    composable<DetailRoute> {navBackStackEntry -> val detailRoute : DetailRoute = navBackStackEntry.toRoute()
                                                  DetailScreen(navController = navController,
                                                               detailId = detailRoute.id
                                                              )
                                }
  }
}
```

navController es un objeto NavController que representa el controlador de la navegación. A través de ir se puede navegar a cualquier destino:

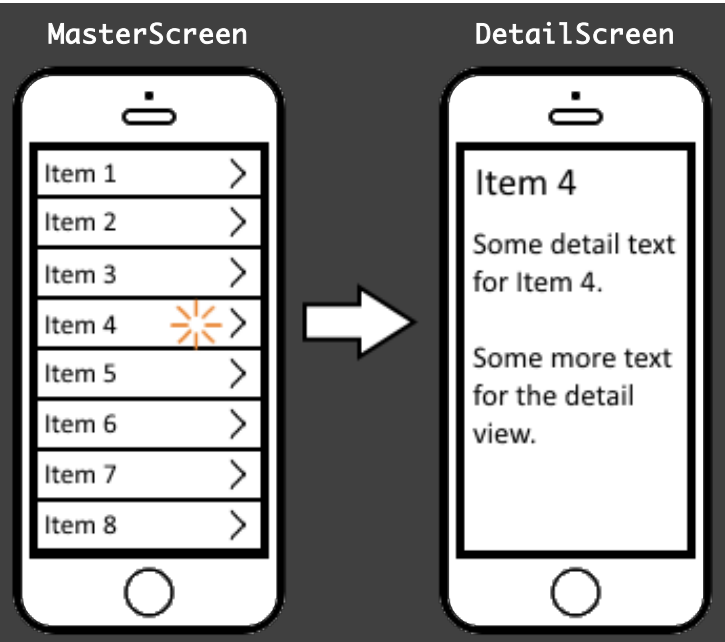
`navCotroller.navigate(route= objetoQueRepresentaElDestino)`



Navigation in Compose

```
class MainActivity : ComponentActivity()
{
  override fun onCreate(savedInstanceState: Bundle?)
  {
    super.onCreate(savedInstanceState)
    setContent {
                 MainScreen()
               }
  }
}


@Composable
fun MainScreen()
{
  val navController = rememberNavController()

  NavHost(navController = navHostController, startDestination = MasterRoute)
  {
    composable<MasterRoute> {navBackStackEntry-> MasterScreen(navController = navController,
                                                   onItemClick= {itemId-> navController.navigate(route= DetailRoute(itemId)) }
                                                  )
                            }

    composable<DetailRoute> {navBackStackEntry -> val detailRoute : DetailRoute = navBackStackEntry.toRoute()
                                                  DetailScreen(navController = navController,
                                                               detailId = detailRoute.id
                                                              )
                            }
  }
```

NavHost es un @Composable que actúa como el contenedor del grafo de navegación. Define las rutas de navegación de la aplicación vinculando a cada ruta su correspondiente *composable*.

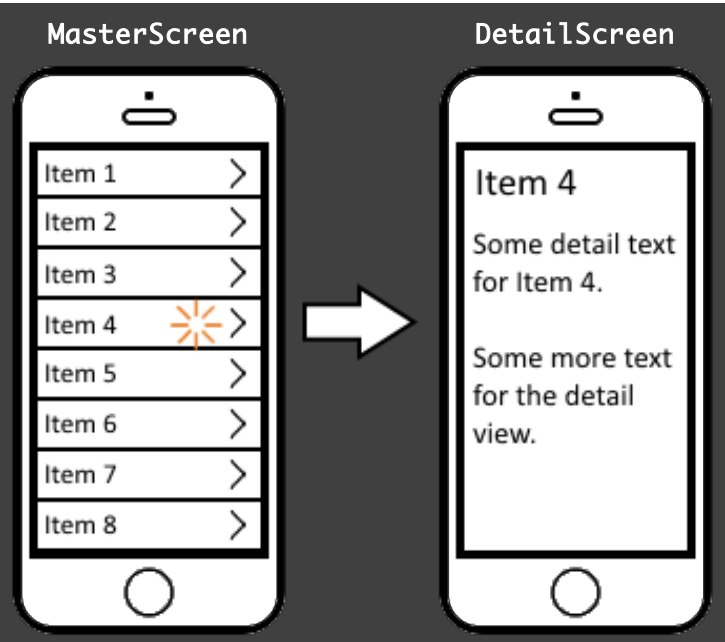


Navigation in Compose

```
class MainActivity : ComponentActivity()
{
  override fun onCreate(savedInstanceState: Bundle?)
  {
    super.onCreate(savedInstanceState)
    setContent {
                 MainScreen()
               }
  }
}


@Composable
fun MainScreen()
{
  val navController = rememb      ontroller()

  NavHost(navControlle      navHostController, st      estination = MasterRoute)
  {
    composable<MasterRoute> {navBackStackEntry-> MasterScreen(navController = navController,
                                                    onItemClick= {itemId-> navController.navigate(route= DetailRoute(itemId)) }
                                                   )
                            }

    composable<DetailRoute> {navBackStackEntry -> val detailRoute : DetailRoute = navBackStackEntry.toRoute()
                                                  DetailScreen(navController = navController,
                                                               detailId = detailRoute.id
                                                              )
                                }
  }
```

Representa un nodo (*destino*) del grafo de navegación.

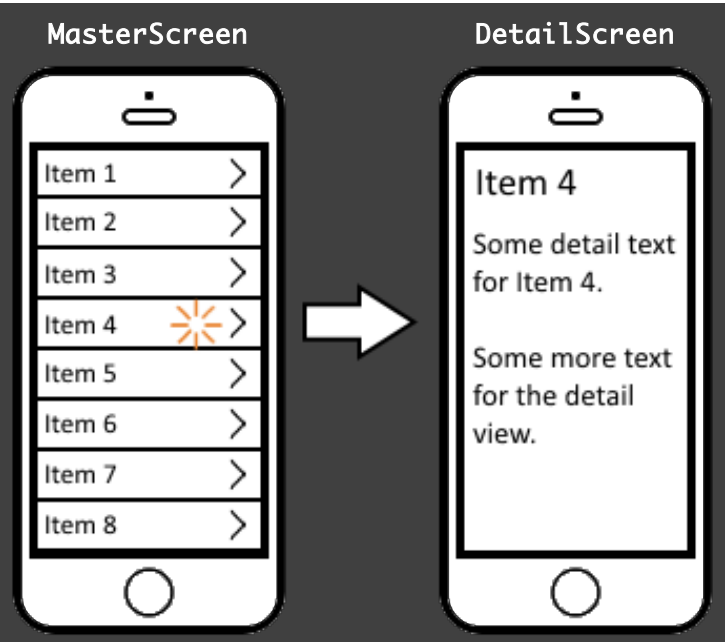
La lambda define cómo llega a dicho *destino* (cómo se invoca al @Composable vinculado a dicha ruta)



Navigation in Compose

```
class MainActivity : ComponentActivity()
{
  override fun onCreate(savedInstanceState: Bundle?)
  {
    super.onCreate(savedInstanceState)
    setContent {
                 MainScreen()
               }
  }
}


@Composable
fun MainScreen()
{
  val navController = rememberNavController()

  NavHost(navController = navHostController, startDestinat     = MasterRoute)
  {
    composable<MasterRoute> {navBackStackEntry-> MasterScreen(navController = navController,
                                                    onItemClick= {itemId-> navController.navigate(route= DetailRoute(itemId)) }
                                                  )
                            }

    composable<DetailRoute> {navBackStackEntry -> val detailRoute : DetailRoute = navBackStackEntry.toRoute()
                                                  DetailScreen(navController = navController,
                                                               detailId = detailRoute.id
                                                              )
                            }
  }
```

Vincula el nodo (destino) al correspondiente @Composable (invoca al @Composable).



Navigation in Compose

```
class MainActivity : ComponentActivity()
{
  override fun onCreate(savedInstanceState: Bundle?)
  {
    super.onCreate(savedInstanceState)
    setContent {
                MainScreen()
               }
  }
}


@Composable
fun MainScreen()
{
  val navController = rememberNavController()

  NavHost(navController = navHostController, startDestination = MasterRoute)
  {
    composable<MasterRoute> {navBackStackEntry-> MasterScreen(navController = navController,
                                                   onItemClick= {itemId-> navController.navigate(route= DetailRoute(itemId)) }
                                                  )

                            }

    composable<DetailRoute> {navBackStackEntry -> val detailRoute : DetailRoute = navBackStackEntry.toRoute()
                                                  DetailScreen(navController = navController,
                                                               detailId = detailRoute.id
                                                              )

                            }
  }
}
```

La transición entre pantallas (por ejemplo, del *origen* MasterScreen al *destino* DetailScreen) se representa por medio de una *lambda* que se le pasa como parámetro a la pantalla *origen. Esta lambda* encapsula la navegación a la ruta *destino* a través del `navController` .

Generalmente a cada pantalla se le pasa como parámetro el navController para que a través de él pueda gestionar su propio menú

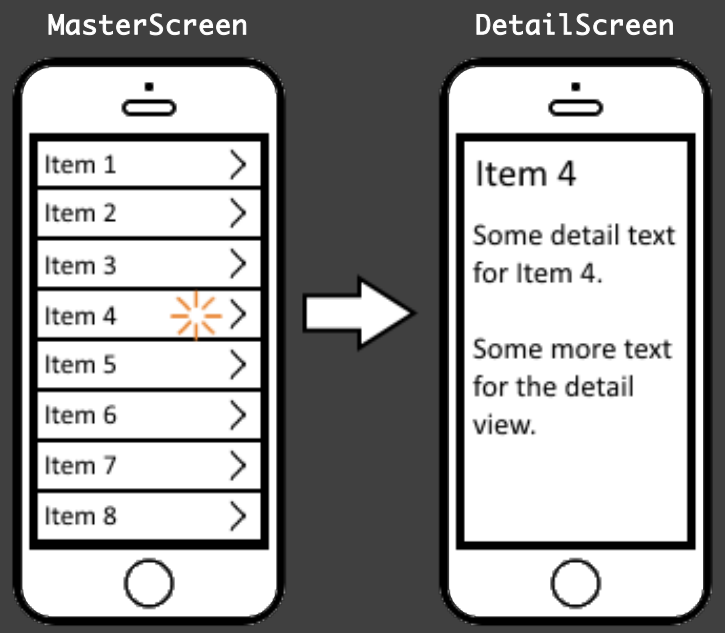
A la "ruta" destino se le pasan los datos que se precise utilizar para, posteriormente, poder utilizarlos al invocar al @Composable (en este caso el identificador del item). A su vez estos datos serán recibidos como parámetros por la *lambda*

Se obtiene el objeto que representa la "ruta" destino para extraer de él los datos necesarios para invocar al @Composable (en el ejemplo, el identificador del item).

Navigation in Compose

```
class MainActivity : ComponentActivity()
{
  override fun onCreate(savedInstanceState: Bundle?)
  {
    super.onCreate(savedInstanceState)
    setContent {
                MainScreen()
               }
  }
}


@Composable
fun MainScreen()
{
  val navController = rememberNavController()

  NavHost(navController = navHostController, startDestination = MasterRoute)
  {
    composable<MasterRoute> {navBackStackEntry-> MasterScreen(navController = navController,
                                                   onItemClick= {itemId-> navController.navigate(route= DetailRoute(itemId)) }
                                                  )
                          }

    composable<DetailRoute> {navBackStackEntry -> val detailRoute : DetailRoute = navBackStackEntry.toRoute()
                                                  DetailScreen(navController = navController,
                                                               detailId = detailRoute.id
                                                              )
                          }
  }
}
```

Se define cuál es el 1° destino de arranque



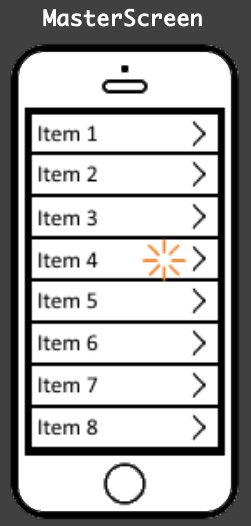
Navigation in Compose

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MasterScreen(navController: NavController,
                 onItemClick: (Long)->Unit
                )
{
  : : :
  Scaffold(modifier = Modifier.fillMaxSize(),
              : : :
           )
  {
    innerPadding -> : : :
                    LazyColumn(…)
                    {
                      items(items = items)
                      {
                         item-> Card(modifier= Modifier.fillMaxWidth()
                                                       .wrapContentHeight()
                                                       .clickable(onClick= { onItemClick(item.id) },
                                                                  role = Role.Button
                                                                 )
                                     )
                                {
                                  //
                                  // Representación de cada item
                                  //
                                }

                      }
                    }
  }
}
```

Recibe el navController para que el Scaffold pueda gestionar sus menús

Representa qué se ejecutará cuando el usuario haga "click" en los items de la lista. En sí, encapsula la transición a la pantalla DetailScreen.

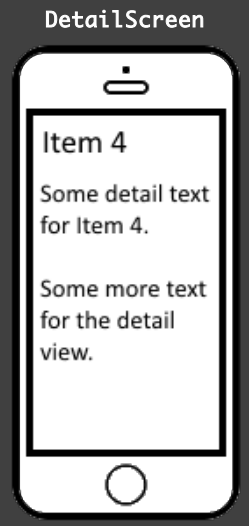
Ejecuta la transición a DetailScreen indicándole al mismo tiempo el identificador del item.



Navigation in Compose

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DetailScreen(itemId: Long, navController: NavController,)
{
  : : :
  Scaffold(modifier = Modifier.fillMaxSize(),
           topBar = { TopAppBar(title = { Text(text = "Pantalla de detalle",
                                               style = MaterialTheme.typography.titleLarge
                                              )
                                        },
                                navigationIcon = {
                                                   IconButton(onClick= { navController.popBackStack() } )
                                                   {
                                                     Icon(imageVector = Icons.AutoMirrored.Filled.ArrowBack,
                                                          contentDescription = "Volver a apntalla Master")
                                                   }
                                                 }
                                )
                    },
            :::
          )
{
   innerPaddings -> //
                    // Representación del item con identificador itemId
                     //
}
```

Representa la transición a la pantalla anterior



Navigation in Compose

```
@Composable
fun MasterScreen(navController: NavController,
                 onItemClick: (Long)->Unit
                )
{
  var resultFromDetail = navController.currentBackStackEntry?
                                      .savedStateHandle?
                                      .getStateFlow<String>("resultFromDetail", "OK")?
                                      .collectAsState() ?: remember { mutableStateOf("OK") }

  : : :
}

@Composable
fun DetailScreen(itemId: Long, navController: NavController,)
{
  : : :
  {
    navHostController.previousBackStackEntry?.savedStateHandle?.set("resultFromDetail", "CANCEL")
    navHostController.popBackStack()
}
  : : :
}
```

Es posible devolver valores desde el *destino* hacia el *origen* (desde DetailScreen a MasterScreen)

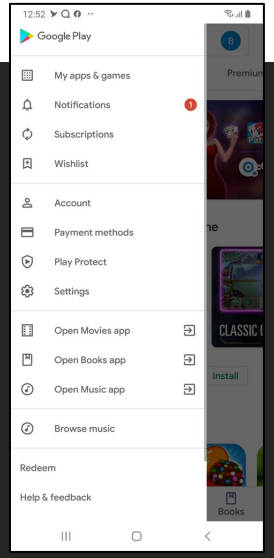
De esta forma se recuperaría en *origen* el valor (o valores) devueltos por *destino*

Scaffold

ModalNavigationDrawer TopAppBar NavigationBar

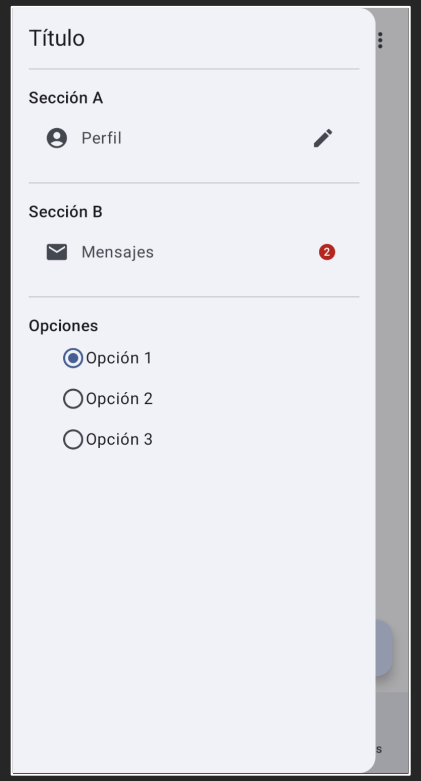
```
@Composable
fun Scaffold(
    modifier: Modifier = Modifier,
    topBar: @Composable () -> Unit = {},
    bottomBar: @Composable () -> Unit = {},
    snackbarHost: @Composable () -> Unit = {},
    floatingActionButton: @Composable () -> Unit = {},
    floatingActionButtonPosition: FabPosition = FabPosition.End,
    containerColor: Color = MaterialTheme.colorScheme.background,
    contentColor: Color = contentColorFor(containerColor),
    contentWindowInsets: WindowInsets = ScaffoldDefaults.contentWindowInsets,
    content: @Composable (PaddingValues) -> Unit
)
```

ModalNavigationDrawer + Scaffold




```
var drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
val coroutineScope = rememberCoroutineScope()

ModalNavigationDrawer(drawerState = drawerState,
                      drawerContent = { ModalDrawerSheet {
                                                  // Contenido del drawer
                                                }
                                      }
                     )
{
  Scaffold(modifier = Modifier.fillMaxSize(),
           topBar= {TopAppBar(title= { Text("App title") },
                              navigationIcon= { IconButton(onClick = { coroutineScope.launch {
                                                                         drawerState.open()
                                                                       }
                                                                     }
                                                           )
                                                {
                                                  Icon(imageVector= Icons.Filled.Menu, contentDescription= "Menú de navegación")
                                                }
                                              },
           :::
          )
  {
    innerPadding -> :::
  }
}
```

ModalNavigationDrawer + Scaffold




```
var drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
var drawerScrollState = rememberScrollState()
val coroutineScope = rememberCoroutineScope()
var currentOpcion by rememberSaveable { mutableStateOf(Opcion.Opcion1) }

ModalNavigationDrawer(drawerState = drawerState,
                      drawerContent = { ModalDrawerSheet {
                                                  // Contenido del drawer
                                                  Column(modifier = Modifier.verticalScroll(state = drawerScrollState).padding(horizontal = 16.dp) )
                                                  {
                                                    Spacer(modifier = Modifier.height(16.dp))
                                                    Text(text = "Título", style = MaterialTheme.typography.titleLarge)
                                                    HorizontalDivider( modifier = Modifier.padding(vertical = 16.dp) )
                                                    Text(text = "Sección A", style = MaterialTheme.typography.titleMedium)
                                                    NavigationDrawerItem(label = { Text("Perfil") },
                                                                         selected = false,
                                                                         onClick = { … },
                                                                         icon = { Icon(imageVector = Icons.Filled.AccountCircle, contentDescription = "Perfil") },
                                                                         badge = { Icon(imageVector = Icons.Filled.Edit, contentDescription = "Edit") },
                                                                        )
                                                    HorizontalDivider( modifier = Modifier.padding(vertical = 16.dp) )
                                                    Text(text = "Sección B", style = MaterialTheme.typography.titleMedium)
                                                    NavigationDrawerItem(label = { Text("Mensajes") },
                                                                         selected = false,
                                                                         onClick = {  },
                                                                         icon = { Icon(imageVector = Icons.Filled.Email, contentDescription = "Mensajes") },
                                                                         badge = { Badge { Text(text = "2") } }
                                                                        )
                                                    HorizontalDivider( modifier = Modifier.padding(vertical = 16.dp) )
                                                    Text(text = "Opciones", style = MaterialTheme.typography.titleMedium)
                                                    Opcion.values().forEach { opcion -> Row( modifier = Modifier.padding(start = 32.dp, top = 8.dp, bottom = 8.dp)
                                                                                                        .selectable(selected = currentOpcion == opcion,
                                                                                                                    onClick = { currentOpcion = opcion; /* Más acciones */ }
                                                                                                                   )
                                                                                           )
                                                                                    {
                                                                                        RadioButton(selected= currentOpcion == opcion, onClick = null)
                                                                                        Text(text = opcion.title)
                                                                                    }}}}}})
{
  Scaffold(…) { innerPadding -> ::: }
}
```

A través de navController se realizaría la correspondiente transición

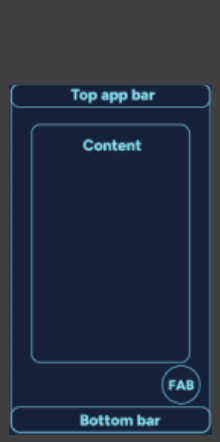
TopAppBar + Scaffold

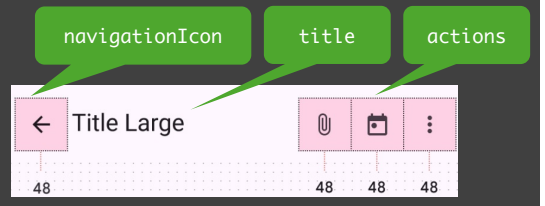
topBar:

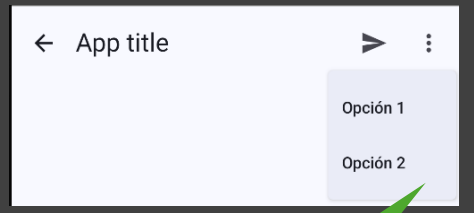
`@Composable () -> Unit = {}` → `composable` que representa la *top bar app* de la pantalla (normalmente un TopAppBar) .

```
@Composable
fun TopAppBar(title: @Composable () -> Unit,
              modifier: Modifier = Modifier,
              navigationIcon: @Composable () -> Unit = {},
              actions: @Composable RowScope.() -> Unit = {},
              windowInsets: WindowInsets = TopAppBarDefaults.windowInsets,
              colors: TopAppBarColors = TopAppBarDefaults.topAppBarColors(),
              scrollBehavior: TopAppBarScrollBehavior? = null
             )
```

Define la Top App Bar de la pantalla.







`title` El composable que describe el título

```
title = { Text("App title") }
```

`navigationIcon` representa el icono de navegación mostrado al comienzo de la *top app bar*.

```
navigationIcon = { IconButton(onClick = { navController.popBackStack() /* Acción de navegación: por ejemplo, abrir un Drawer
                                                                          o regresar a la pantalla anterior */ })
                                {
                                  Icon(imageVector = Icons.AutoMirrored.Filled.ArrowBack, contentDescription = "Menú de navegación")
                                }
                              }
```

`actions` las acciones que se muestran al final de la top app bar. El diseño por defecto es un Row que muestra IconButtons.

```
var expanded by remember { mutableStateOf(false) }

actions = { IconButton(onClick = { /* Acción de enviar */ }) { Icon(imageVector = Icons.AutoMirrored.Filled.Send, contentDescription = "Enviar") }
            IconButton(onClick = { expanded = true })        { Icon(imageVector = Icons.Default.MoreVert, contentDescription = "Mostrar menú")   }
            // Menú desplegable
            DropdownMenu(expanded = expanded, onDismissRequest = { expanded = false } )
            {
               DropdownMenuItem(onClick = { /* Acción para la opción 1 */; expanded = false}, text = { Text("Opción 1") })
               DropdownMenuItem(onClick = { /*Acción para la opción 2*/; expanded = false}, text = { Text("Opción 2") })
            }
          }
```

NavigationBar + Scaffold

bottomBar:

`@Composable () -> Unit = {}` → `composable` que representa la bottom bar de la pantalla (normalmente un NavigationBar) .

```
@Composable
fun NavigationBar(
    modifier: Modifier = Modifier,
    containerColor: Color = NavigationBarDefaults.containerColor,
    contentColor: Color = MaterialTheme.colorScheme.contentColorFor(containerColor),
    tonalElevation: Dp = NavigationBarDefaults.Elevation,
    windowInsets: WindowInsets = NavigationBarDefaults.windowInsets,
    content: @Composable RowScope.() -> Unit
)
```

Define la Bottom Bar de la pantalla.

`bottomBar` *El composable que describe la bottom bar. El diseño por defecto es un Row que muestra NavigationBarItems*

```
bottomBar = {  NavigationBar()
                {
                    NavigationBarItem(selected = true,
                                      onClick = { … },
                                      icon = { Icon(imageVector = Icons.Filled.Home, contentDescription = "Inicio") },
                                      label = { Text("Inicio") }
                                     )
                    NavigationBarItem(selected = false,
                                      onClick = { … },
                                      icon = { Icon(imageVector = Icons.Filled.Search, contentDescription = "Buscar") },
                                      label = { Text("Buscar") }
                                     )
                    NavigationBarItem(selected = false,
                                      onClick = { … },
                                      icon = { Icon(imageVector = Icons.Filled.Settings, contentDescription = "Preferencias") },
                                      label = { Text("Preferencias") }
                                     )
                }
            }
```

A través de navController se realizaría la correspondiente transición