Scaffold

Es un composable que representa el "esqueleto" de una pantalla.
Sus parámetros representan cada uno de los elementos de la pantalla:

- Top app bar
- Button bar
- Content
- SnakebarHost
- Floating Action Button

Scaffold

```
@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
)
```

Scaffold

content:

@Composable (PaddingValues) -> Unit → composable que representa el *content* de la pantalla.

```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
       topBar = { ... },
       bottomBar = { ... },
       snackbarHost = { ... },
      )
{
   innerPadding ->  Content(modifier = Modifier.padding(innerPadding),
                    snackbarHostState = snackbarHostState)
}


@Composable
fun Content(modifier: Modifier = Modifier, snackbarHostState: SnackbarHostState)
{
   : : :
}
```


```
class Item(val value: Int)
{
  val title: String = "Título $value"
  val description: String = "Esta es la descripción del elemento $value"
}
```

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") }
```

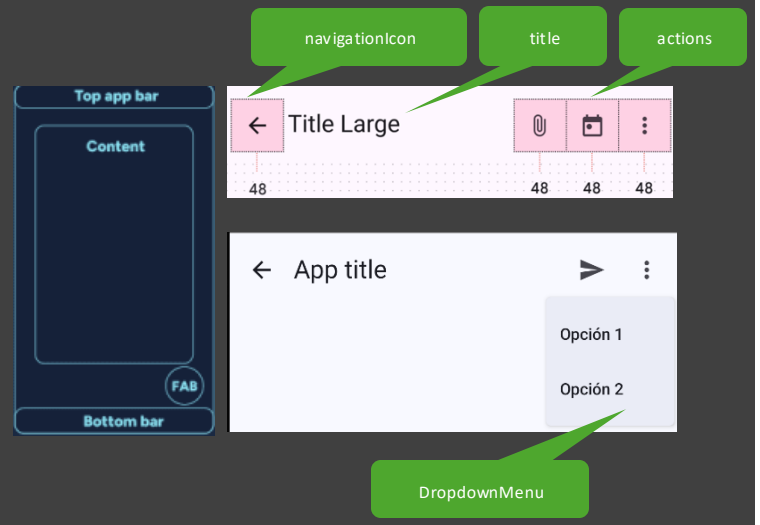
navigationIcon — representa el icono de navegación mostrado al comienzo de la *top app bar*.

```
navigationIcon = { IconButton(onClick = { /* Acción de navegación, por ejemplo abrir un Drawer o navegar a otra pantalla */ })
                   {
                    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") })
        }
       }
```



Scaffold

bottomBar:

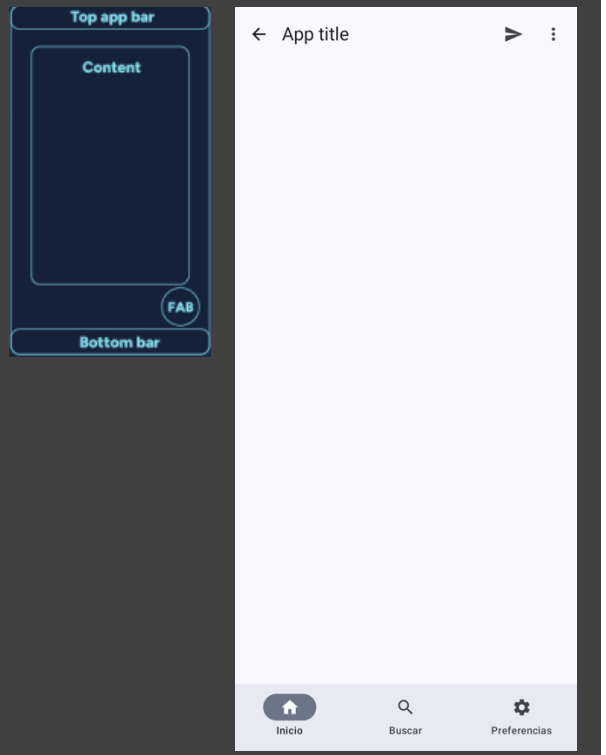
@Composable () -> Unit = {} → composable que representa la bottom bar de la pantalla (normalmente un NavigationBar) .

```
@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") }
                    )
        }
      }
```



Scaffold

snackbarHost:

@Composable () -> Unit = {} → composable que representa el SkaneBarHost con el que mostrar mensajes emergentes.

```
@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
)
```

el SkaneBarHost con el que mostrar mensajes emergentes.

snakeBarHost — *El composable que describe el SnakeBarHost*

```
var expanded by remember { mutableStateOf(false) }
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        topBar = { ... },
        bottomBar = { ... },
        snackbarHost = { SnackbarHost(hostState = snackbarHostState) },
       )
{
    innerPadding ->  Content(modifier = Modifier.padding(innerPadding), snackbarHostState = snackbarHostState)
}
```

```
@Composable
fun Content(modifier: Modifier = Modifier, snackbarHostState: SnackbarHostState)
{
    : : :
    LaunchedEffect(key)
    {
        val snakeBarResult = snackbarHostState.showSnackbar(message= "Mensaje",
                                                            duration= SnackbarDuration.Long,
                                                            actionLabel="Deshacer")
        when (snakeBarResult)
        {
            SnackbarResult.ActionPerformed -> { /* Acción Deshacer */}
            SnackbarResult.Dismissed       -> {}
        }
    }
    :::
}
```

Scaffold

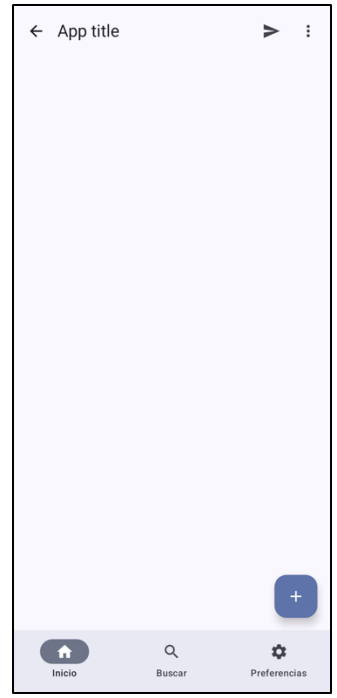
floatingActionButton:

@Composable () -> Unit = {} → composable que representa el botón principal de la pantalla (normalmente un FloatingActionButton) .

```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
       topBar = { ... },
       bottomBar = { ... },
       snackbarHost = { ... },
       floatingActionButton = { FloatingActionButton(onClick = { ... })
                          {
                            Icon(imageVector = Icons.Filled.Add, contentDescription = "Agregar")
                          }
                         },
     )
{
   innerPadding ->  Content(modifier = Modifier.padding(innerPadding),
                     snackbarHostState = snackbarHostState)
}


@Composable
fun Content(modifier: Modifier = Modifier, snackbarHostState: SnackbarHostState)
{
   : : :
}
```



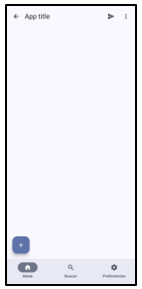
Scaffold

floatingActionButtonPosition:

FabPosition = FabPosition.End → define la ubicación en la pantalla del floatingActionButton .

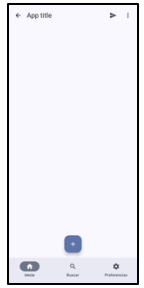
```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.Start
       )
{
   innerPadding -> Content( :::)
}
```



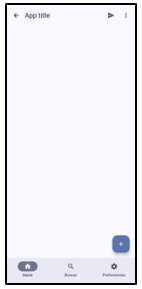
```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.Center
       )
{
   innerPadding -> Content( :::)
}
```



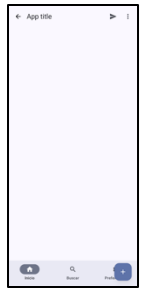
```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.End
       )
{
   innerPadding -> Content( :::)
}
```



```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.EndOverlay
       )
{
   innerPadding -> Content( :::)
}
```



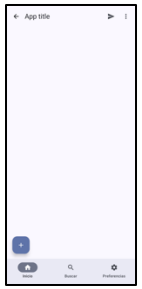
Scaffold

floatingActionButtonPosition:

FabPosition = FabPosition.End → define la ubicación en la pantalla del floatingActionButton .

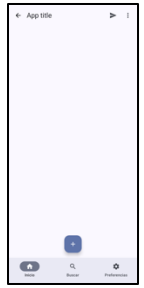
```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.Start
       )
{
   innerPadding -> Content( :::)
}
```



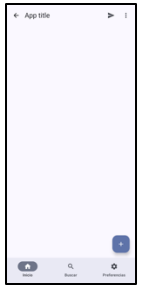
```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.Center
       )
{
   innerPadding -> Content( :::)
}
```



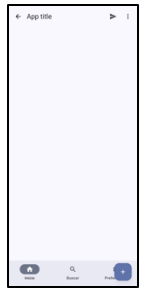
```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.End
       )
{
   innerPadding -> Content( :::)
}
```

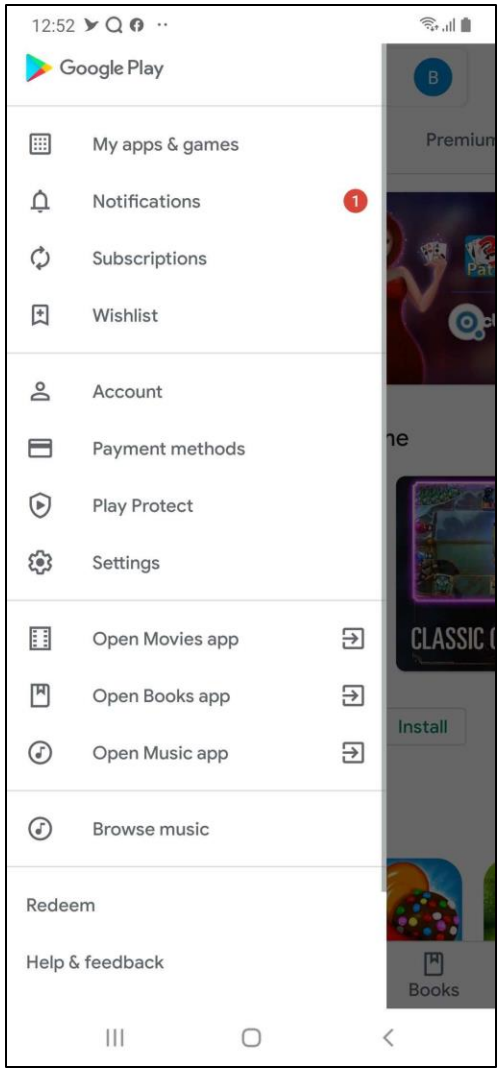


```
val snackbarHostState = remember { SnackbarHostState() }

Scaffold(modifier = Modifier.fillMaxSize(),
        :::
        loatingActionButton = { ::: },
        snackbarHost = FabPosition.EndOverlay
       )
{
   innerPadding -> Content( :::)
}
```



ModalNavigationDrawer + Scaffold



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) )

                                                                                                    ht(16.dp))
                                                                                                    erialTheme.typography.titleLarge)
                                                                                                    odifier.padding(vertical = 16.dp) )
                                                                                                    MaterialTheme.typography.titleMedium)
                                                                                                    Text("Perfil") },


                                                                                                    ector = Icons.Filled.AccountCircle, contentDescription = "Perfil") },
                                                                                                    Vector = Icons.Filled.Edit, contentDescription = "Edit") },

                                                                                                    odifier.padding(vertical = 16.dp) )
                                                                                                    MaterialTheme.typography.titleMedium)
                                                                                                    Text("Mensajes") },


                                                                                                    ector = Icons.Filled.Email, contentDescription = "Mensajes") },
                                                                                                    xt(text = "2") } }

                                                                                                    odifier.padding(vertical = 16.dp) )
                                                                                                    MaterialTheme.typography.titleMedium)
                                                                                                    -> Row( modifier = Modifier.padding(start = 32.dp, top = 8.dp, bottom = 8.dp)
                                                                                                     .selectable(selected = currentOpcion == opcion,
                                                                                                                 onClick = { currentOpcion = opcion; /* Más acciones */ }
                                                                                                                )


                                                                                                    on(selected= currentOpcion == opcion, onClick = null)
                                                                                                    = opcion.title)
                        ////
{
  Scaffold(...) { innerPadding -> ::: }
}
```