94. Selección tipo menú

Se incluye:

- Scaffold
- Menu
- TopAppBar
- BottomBar
- FloatingActionButton
- Drawer

El componente de layout usado con estos componentes es Scaffold

94.1 Scaffold

Seguir en AD

Piensa en Scaffold como el esqueleto básico de tu interfaz de usuario en una aplicación Compose. Proporciona una estructura estándar para implementar la mayoría de las interfaces de usuario comunes.

- Estructura de Scaffold:
- Scaffold tiene `slots` para diferentes partes de la interfaz, como la barra de aplicaciones (app bar), el contenido principal, un drawer (menú lateral), FABs (Floating Action Buttons), y barras de navegación o snackbar.

- Barra de Aplicaciones: Scaffold te permite agregar fácilmente una barra de aplicaciones en la parte superior de la pantalla, donde generalmente colocas el título de la pantalla, menús, y otras acciones.
- Contenido Principal: Aquí es donde colocas el contenido principal de tu pantalla. Scaffold se encarga de posicionar este contenido de manera que no se superponga con otros elementos como la barra de aplicaciones o el FAB.
- Drawer (Cajon): menú lateral (como el menú de hamburguesa), Scaffold tiene un slot para esto. Es útil para la navegación en la app.
- Floating Action Button (FAB): Para acciones primarias en tu pantalla, puedes usar un FAB. Scaffold te permite agregarlo en un lugar adecuado, generalmente en la esquina inferior derecha.
- Bottom Bar y Snackbar: Si necesitas una barra de navegación en la parte inferior o mostrar mensajes temporales (snackbars), Scaffold también te ofrece espacios para ellos.

Ejemplo:

```
@Composable
fun ScaffoldExample() {
    var presses by remember { mutableIntStateOf(0) }

    Scaffold(
        topBar = {
            TopAppBar(
                colors = topAppBarColors(
                    containerColor = MaterialTheme.colorScheme.primaryContainer,
                    titleContentColor = MaterialTheme.colorScheme.primary,
                ),
                title = {
                    Text("Top app bar")
                }
            )
        },
        bottomBar = {
            BottomAppBar(
                containerColor = MaterialTheme.colorScheme.primaryContainer,
                contentColor = MaterialTheme.colorScheme.primary,
```

```
            ) {
                Text(
                    modifier = Modifier
                        .fillMaxWidth(),
                    textAlign = TextAlign.Center,
                    text = "Bottom app bar",
                )
            }
        },
        floatingActionButton = {
            FloatingActionButton(onClick = { presses++ }) {
                Icon(Icons.Default.Add, contentDescription = "Add")
            }
        }
    ) { innerPadding ->
        Column(
            modifier = Modifier
                .padding(innerPadding),
            verticalArrangement = Arrangement.spacedBy(16.dp),
        ) {
            Text(
                modifier = Modifier.padding(8.dp),
                text =
                """
                    This is an example of a scaffold. It uses the Scaffold composable's parameters to create a screen with a
simple top app bar, bottom app bar, and floating action button.

                    It also contains some basic inner content, such as this text.

                    You have pressed the floating action button $presses times.
                """.trimIndent(),
            )
        }
    }
}
```

En este capítulo se incluyen otros componentes como:

- **TopAppBar y CenterAlignedTopAppBar** Es el equivalente básico de una toolbar. Puedes personalizar el título, los botones de acción, y el estilo en general. Por ejemplo
- **BottomAppBar**: Este no es exactamente una toolbar, pero se utiliza para una barra de navegación en la parte inferior de la pantalla. Es común en aplicaciones modernas, ofreciendo accesos directos a las funciones principales.
- **Custom AppBar**: Puedes crear tu propia AppBar personalizada usando Row o cualquier otro layout combinado con los componentes de Material Design como IconButton, Text, etc. Esto te da libertad total para diseñar algo único.
- **FloatingActionButton** : Este botón flotante es ideal para acciones principales en tus pantallas, como agregar un nuevo elemento, iniciar una nueva conversación, etc.
- **Drawer**: cajón

94.2 Menus

(ver) y el codigo

Veamos dos tipos de menús: **Dropdown Menu** y **Exposed Dropdown Menu** (Material3)

94.2.1 Dropdown Menu

Para los menús, puedes usar el componente **DropdownMenu**. Es ideal para mostrar una lista de opciones cuando el usuario interactúa con un botón o algún otro elemento de la interfaz. Por ejemplo:

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

Box(modifier = Modifier.fillMaxSize()) {
    IconButton(onClick = { expanded = true }) {
        Icon(Icons.Filled.MoreVert, contentDescription = null)
```

```
    }

    DropdownMenu(
        expanded = expanded,
        onDismissRequest = { expanded = false }
    ) {
        DropdownMenuItem(onClick = { /* Handle click */ }) {
            Text("Opción 1")
        }
        DropdownMenuItem(onClick = { /* Handle click */ }) {
            Text("Opción 2")
        }
        // Agrega más opciones aquí
    }
}
```

Declaración completa:

```
@Composable
fun DropdownMenu(
    expanded: Boolean!,
    onDismissRequest: (() -> Unit)?,
    modifier: Modifier! = Modifier,
    offset: DpOffset! = DpOffset(0.dp, 0.dp),
    properties: PopupProperties! = PopupProperties(focusable = true),
    content: (@Composable @ExtensionFunctionType ColumnScope.() -> Unit)?
): Unit
```

* expanded: Determina si el menú está abierto o cerrado * onDismissRequest: Lambda llamada cuando el usuario solicita descartar el menú (tap por fuera del menú o tap en el Back button) * offset: La cantidad de desplazamiento aplicado al menú según la dirección en que se expande * properties: Propiedades usadas para configurar el comportamiento del componente PopUp que DropdownMenu() usa en su interior * content: Espacio donde incluyes elementos DropdownMenuItem para representar los ítems del menú

94.2.1.1 Agregar items al menú.

Los items irán en el parámetro `content`

```
@Composable
fun DropdownMenuItem(
    onClick: (() -> Unit)?,
    modifier: Modifier! = Modifier,
    enabled: Boolean! = true,
    contentPadding: PaddingValues! = MenuDefaults.DropdownMenuItemContentPadding,
    interactionSource: MutableInteractionSource! = remember { MutableInteractionSource() },
    content: (@Composable @ExtensionFunctionType RowScope.() -> Unit)?
): Unit
```

- onClick: Es la lambda ejecutada cuando el usuario hace clic en el ítem
- enabled: Determina si ítem está habilitado o deshabilitado (tono grisáceo y no recibe eventos)
- contentPadding: Relleno aplicado al contenido del ítem
- content: El espacio donde agregas el componente Text para nombrar al ítem

94.2.2 Exposed Dropdown Menu

AD: https://developer.android.com/reference/kotlin/androidx/compose/material/ExposedDropdownMenuBoxScope Ver: https://blog.stackademic.com/how-to-create-an-exposed-dropdown-menu-in-jetpack-compose-74072495b769

https://www.composables.com/material3/exposeddropdownmenubox

Definición:

```
@Composable
open fun ExposedDropdownMenu(
    expanded: Boolean,
    onDismissRequest: () -> Unit,
    modifier: Modifier = Modifier,
```

```
    scrollState: ScrollState = rememberScrollState(),
    content: @Composable ColumnScope.() -> Unit
): Unit
```

94.3 TopAppBar

(ver)

Componentes de TopAppBar:

- **Título**: Normalmente muestra el nombre de la pantalla actual o de la aplicación.
- **Botones de Acción**: Puedes incluir varios íconos de acciones, como buscar, notificaciones, ajustes, etc.
- **Botón de Navegación**: A menudo es un ícono de menú (hamburguesa) o una flecha de retroceso, dependiendo del contexto de la pantalla.

Ejemplo:

```
TopAppBar(
    title = { Text("Mi AppBar") },
    navigationIcon = {
        IconButton(onClick = { /* manejar clic */ }) {
            Icon(Icons.Filled.Menu, contentDescription = "Menú")
        }
    },
    actions = {
        IconButton(onClick = { /* manejar clic */ }) {
            Icon(Icons.Filled.Search, contentDescription = "Buscar")
        }
        // Puedes agregar más iconos aquí
    }
)
```

Variantes:

Jetpack Compose ofrece varias variantes de TopAppBar, como **DefaultTopAppBar**, **CenterAlignedTopAppBar**, y **SmallTopAppBar**, cada una con diferentes estilos y propósitos.

94.4 BottomAppBar

Barra en la parte inferior.
Se usa para proporcionar acceso rápido a acciones y navegación en la parte inferior de la pantalla. Es especialmente útil para teléfonos grandes donde alcanzar la parte superior de la pantalla con una sola mano puede ser incómodo.

Características Clave:

- Acciones Primarias y Secundarias: Al igual que TopAppBar, puedes colocar acciones dentro de BottomAppBar. Estas pueden ser acciones primarias de tu aplicación o acciones secundarias relacionadas con la pantalla actual.
- Integración con FAB: Una característica distintiva de BottomAppBar es su capacidad para integrarse con un Floating Action Button (FAB). El FAB puede colocarse dentro del BottomAppBar, ya sea centrado, al final o encajado en un corte en el BottomAppBar.
- Navegación: Es común utilizar BottomAppBar para albergar componentes de navegación, como BottomNavigationItems, que permiten a los usuarios cambiar entre diferentes vistas o secciones de la aplicación.

```
BottomAppBar {
    IconButton(onClick = { /* manejar clic */ }) {
        Icon(Icons.Filled.Menu, contentDescription = "Menú")
    }
    // El resto de los íconos y funcionalidades van aquí
}
```

Uso Común:

BottomAppBar es ideal para aplicaciones con navegación basada en pestañas o cuando deseas mantener las acciones más importantes al alcance del pulgar del usuario. Consideraciones de Diseño:

Aunque BottomAppBar es muy conveniente, es importante considerar el equilibrio entre la cantidad de contenido y la facilidad de uso. Demasiados elementos pueden abrumar al usuario o hacer que la navegación sea complicada.

94.5 FloatingActionButton

```
Scaffold(
    floatingActionButton = {
        FloatingActionButton(onClick = { /* Acción al hacer clic */ }) {
            Icon(Icons.Filled.Add, contentDescription = "Agregar")
        }
    }
) {
    // El contenido de tu pantalla va aquí
}
```

94.6 Drawer

El Drawer, o cajón de navegación, es un panel que se desliza desde el borde de la pantalla, generalmente desde el lado izquierdo, y se usa para navegar por diferentes secciones de una aplicación. Es muy común en aplicaciones con varias pantallas o secciones.

En Jetpack Compose, puedes integrar un Drawer fácilmente con Scaffold. Aquí te explico cómo:

```
Scaffold(
    drawerContent = {
        // Aquí pones los elementos del Drawer, como botones, listas, etc.
        Text("Elemento 1")
        Text("Elemento 2")
```

```
        // Más elementos...
    },
    // Aquí van otros componentes como TopAppBar, FloatingActionButton, etc.
) {
    // El contenido principal de tu pantalla
}
```

94.7 Uso conjunto de TopAppBar y Navigation

Vamos a fijarnos en una implementación básica con `TopAppBar` y un componente de navegación como `NavHost` de la biblioteca de navegación de Jetpack:

1. Configurar gradle

```
dependencies {
    implementation "androidx.compose.ui:ui:<version>"
    implementation "androidx.navigation:navigation-compose:<version>"
}
```

2. Definir Navhost El NavHost es el componente que gestiona la navegación entre diferentes pantallas (composables) en tu aplicación. Define tus rutas de navegación y sus correspondientes composables:

NavHost necesita un NavController.

```
val navController = rememberNavController()
NavHost(navController = navController, startDestination = "home") {
    composable("home") { HomeScreen(navController) }
    composable("profile") { ProfileScreen(navController) }
    // Agrega más rutas y pantallas según sea necesario
}
```

3. Integar NavHost y TopAppBar

```
Scaffold(
    topBar = {
        TopAppBar(
            title = { Text("Mi Aplicación") },
            navigationIcon = {
                IconButton(onClick = { /* Manejar clic del menú o navegación */ }) {
                    Icon(Icons.Filled.Menu, contentDescription = "Menú")
                }
            }
        )
    }
) {
    NavHost(navController = navController, startDestination = "home") {
        // Tus rutas composable
    }
}
```

1. Gestionar la navegación En cada pantalla componible , puedes manejar la navegación respondiendo a eventos del usuario. Por ejemplo, en HomeScreen, podrías tener un botón que navegue a ProfileScreen:

```
@Composable
fun HomeScreen(navController: NavController) {
    Button(onClick = { navController.navigate("profile") }) {
        Text("Ir al Perfil")
    }
}
```

Consideraciones Adicionales:
* Actualización del Título y Acciones de TopAppBar: Puedes hacer que el título y las acciones en TopAppBar cambien según la pantalla en la que se encuentre el usuario. Esto se puede lograr pasando datos al TopAppBar desde cada pantalla o usando un ViewModel. * Integración con el Drawer y el Bottom Navigation: Si estás utilizando un Drawer o una Bottom Navigation Bar, puedes integrarlos de manera similar dentro del Scaffold y gestionar la

navegación con el mismo NavController. * Uso de Argumentos y Navegación Profunda: Jetpack Navigation permite pasar argumentos entre pantallas y configurar la navegación profunda para una integración más compleja.

94.8 Apendice

Enlaces:
* Barras de la aplicación
* Barra superior, en Material 3 * Barra inferior

Versión 0.5, 12-12-23 Versión 0.6 8-1-24