

Esmeraldas 18 de diciembre del 2025

Desarrollo Móvil

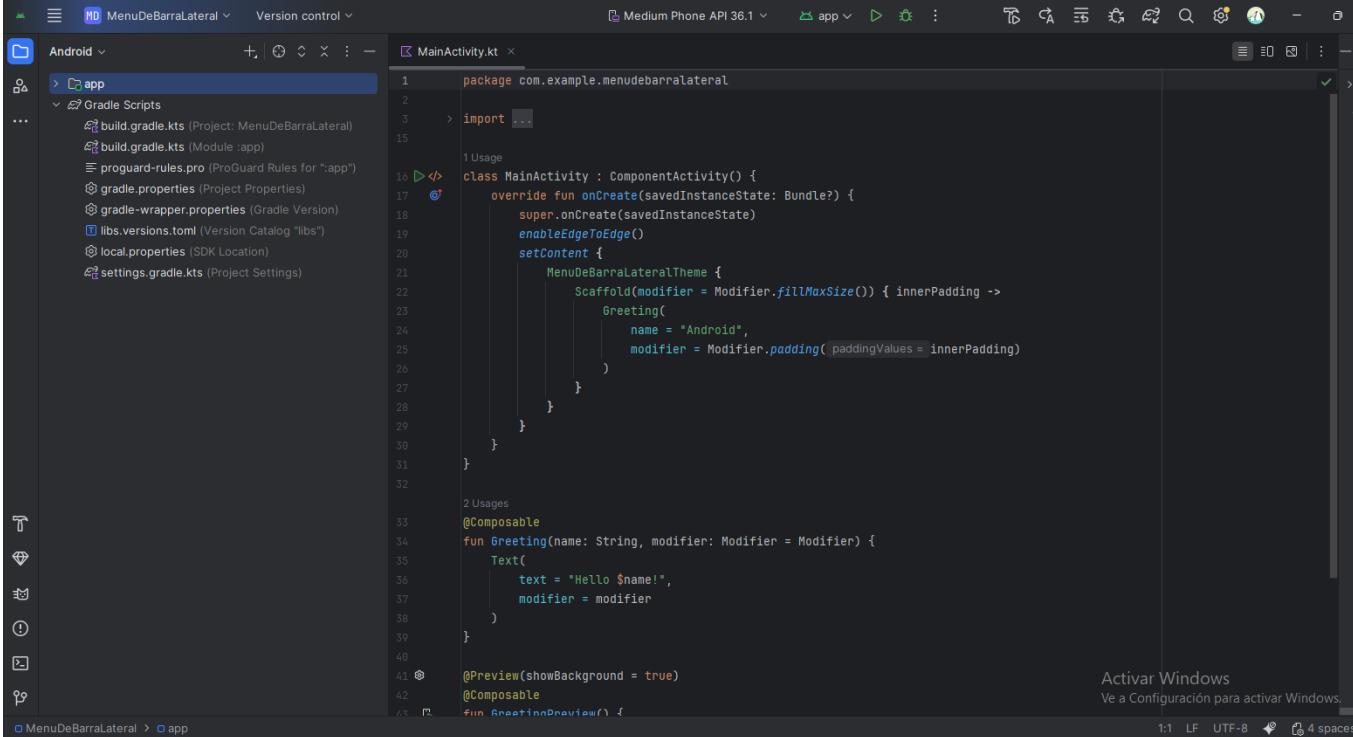
nombres: Zambrano palacios Jandry Steven

Maestro: Kleber Posligua

Tarea 2:

Menú Barra lateral con Jetpack Compose

Primer paso de la tarea es que se debe crear el proyecto menú barra lateral y ubicar en el proyecto en Empty Activity esperar a que cargue



The screenshot shows the Android Studio interface with the project 'MenuDeBarraLateral' open. The code editor displays 'MainActivity.kt' which contains the following Kotlin code:

```
1 package com.example.menuidebarralateral
2
3 > import ...
15
16 Usage
17 <> class MainActivity : ComponentActivity() {
18     override fun onCreate(savedInstanceState: Bundle?) {
19         super.onCreate(savedInstanceState)
20         enableEdgeToEdge()
21         setContent {
22             MenuDeBarraLateralTheme {
23                 Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
24                     Greeting(
25                         name = "Android",
26                         modifier = Modifier.padding(paddingValues = innerPadding)
27                     )
28                 }
29             }
30         }
31     }
32
33 2 Usages
34 @Composable
35 fun Greeting(name: String, modifier: Modifier = Modifier) {
36     Text(
37         text = "Hello $name!",
38         modifier = modifier
39     )
40
41     @Preview(showBackground = true)
42     @Composable
43     fun GreetingPreview() {
```

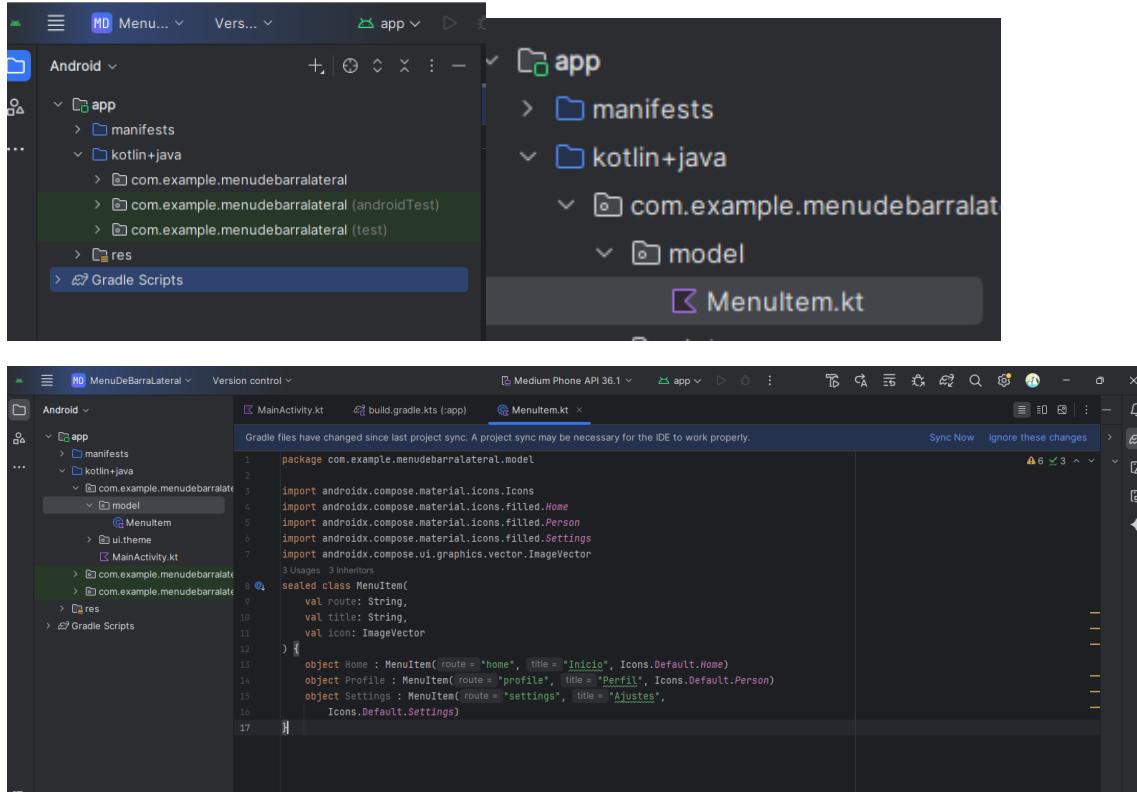
The code implements a Jetpack Compose application with a main activity that features a scaffold and a greeting text component. The 'Greeting' component is annotated with @Composable and takes a 'name' parameter and a 'modifier' parameter.

2 paso cuando está el proyecto creado tenemos que ir a buil.gradle: modulo app y ubicar las dependencias

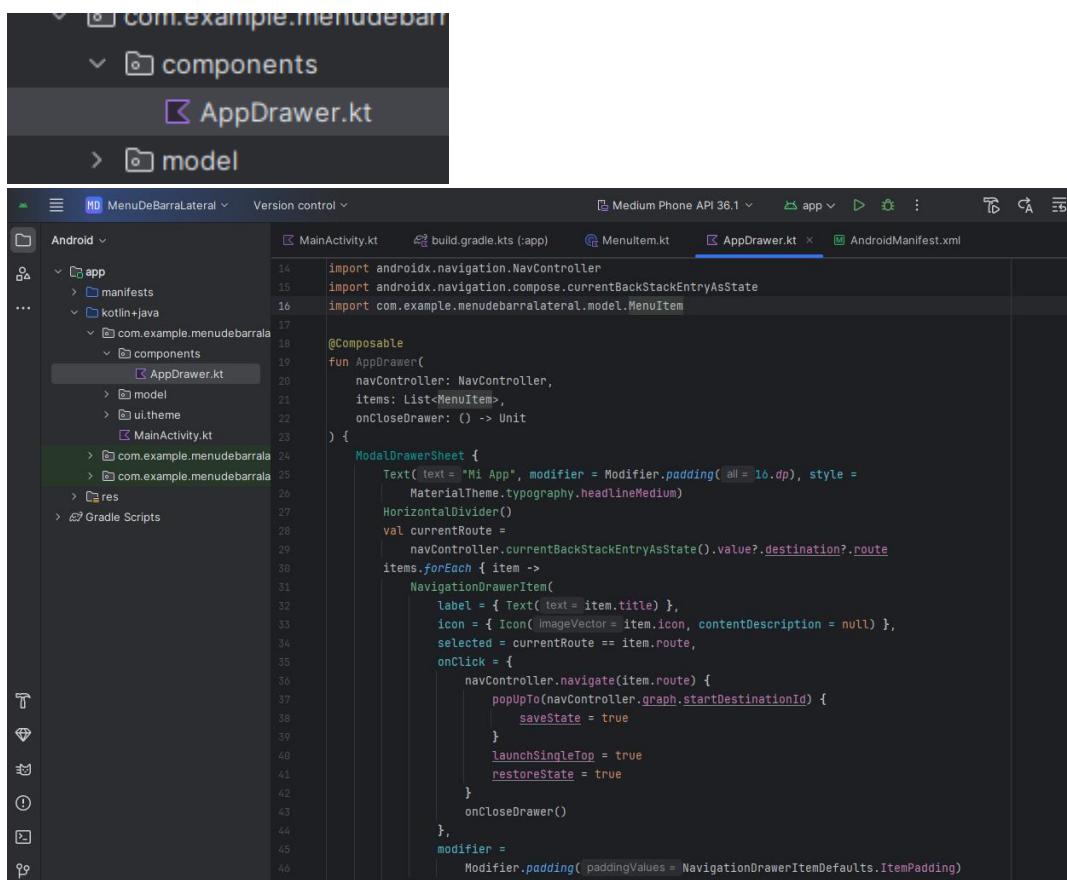
implementation("androidx.navigation:navigation-compose:2.7.7") //o versión más reciente

implementation("androidx.compose.material:material-icons-extended:1.6.0")

paso 3 crear un nuevo paquete llamado: model en class kotlin class file MenuItem



Paso 4 crear un package Compose y crear un kotlin poniendo nombre de AppDrawer.kt

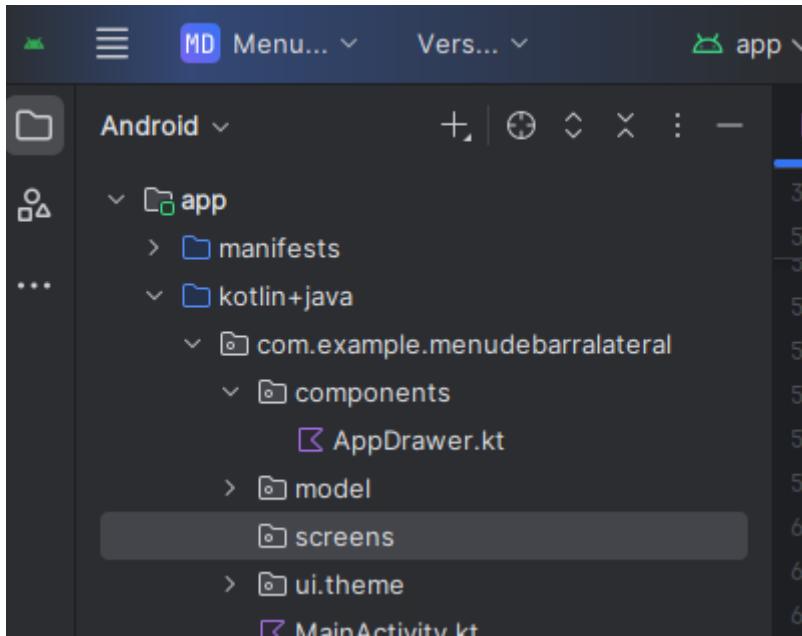


The screenshot shows the Android Studio interface with the code editor open to `MainActivity.kt`. The code implements a side menu using Jetpack Compose. It defines a `TopAppBar` with a title and a navigation icon. The main content is a `NavHost` with three possible destinations: `MenuItem.Home.route`, `MenuItem.Profile.route`, and `MenuItem.Settings.route`. The code uses `Modifier.padding` to add inner padding to the `NavHost`.

```
33     fun MainAppScreen() {
34         // 2. EL SCAFFOLD CON LA DRAWA SUPERIOR
35         Scaffold(
36             topBar = {
37                 TopAppBar(
38                     title = { Text(text = "Menú Lateral Demo") },
39                     navigationIcon = {
40                         IconButton(onClick = {
41                             scope.launch { drawerState.open() }
42                         })
43                     }
44                 )
45             }
46         ) { innerPadding ->
47             // 3. EL NAVHOST CAMBIA EL CONTENIDO CENTRAL
48             NavHost(
49                 navController = navController,
50                 startDestination = MenuItem.Home.route,
51                 modifier = Modifier.padding(paddingValues = innerPadding)
52             ) {
53                 composable(MenuItem.Home.route) { ScreenContent(title = "Pantalla de Inicio") }
54                 composable(MenuItem.Profile.route) { ScreenContent(title = "Pantalla de Perfil") }
55                 composable(MenuItem.Settings.route) { ScreenContent(title = "Pantalla de Ajustes") }
56             }
57         }
58     }
59 }
```



Paso 5 creamos un package Screens



```
fileScreen.kt SettingsScreen.kt build.gradle.kts (app) Menutem.kt AppDrawer.kt AndroidManifest.xml
1 package com.example.menudebarralateral.screens
2
3 import androidx.compose.foundation.background
4 import androidx.compose.foundation.layout.Box
5 import androidx.compose.foundation.layout.fillMaxSize
6 import androidx.compose.material3.MaterialTheme
7 import androidx.compose.material3.Text
8 import androidx.compose.runtime.Composable
9 import androidx.compose.ui.Alignment
10 import androidx.compose.ui.Modifier
11 import androidx.compose.ui.graphics.Color
12 @Composable
13 fun HomeScreen() {
14     Box(
15         modifier = Modifier
16             .fillMaxSize()
17             .background(color = 0xFFE0F7FA), // Un azul muy clarito
18         contentAlignment = Alignment.Center
19     ) {
20         Text(
21             text = "Bienvenido al Inicio", style =
22                 MaterialTheme.typography.displaySmall
23     )
24 }
```

```
ProfileScreen.kt SettingsScreen.kt build.gradle.kts (app) Menutem.kt AppDrawer.kt AndroidManifest.xml
1 package com.example.menudebarralateral.screens
2
3 import androidx.compose.foundation.background
4 import androidx.compose.foundation.layout.Box
5 import androidx.compose.foundation.layout.fillMaxSize
6 import androidx.compose.material3.MaterialTheme
7 import androidx.compose.material3.Text
8 import androidx.compose.runtime.Composable
9 import androidx.compose.ui.Alignment
10 import androidx.compose.ui.Modifier
11 import androidx.compose.ui.graphics.Color
12 @Composable
13 fun ProfileScreen() {
14     Box(
15         modifier = Modifier
16             .fillMaxSize()
17             .background(color = 0xFFFF33FF), // Un morado muy clarito
18         contentAlignment = Alignment.Center
19     ) {
20         Text(
21             text = "Perfil de Usuario", style =
22                 MaterialTheme.typography.displaySmall
23     )
24 }
```

4:52



≡ Menú Lateral Demo

Pantalla de Perfil

Activar Windows

Ve a Configuración para activar Windows.





Una **Sealed Class** (clase sellada) es una clase que restringe qué otras clases pueden heredarla, definiendo un conjunto **cerrado y limitado de subclases conocidas en tiempo de compilación**. Se recomienda usarla cuando necesitas representar un conjunto fijo y predecible de estados o tipos relacionados, como estados de UI (cargando, éxito, error), tipos de mensajes o resultados de operaciones, para **asegurar el manejo exhaustivo** de todos los casos posibles y evitar extensiones no deseadas, siendo una alternativa más potente que los enum.

