**Navigation Drawer en Android (Java) – Guía Completa**

Un **Navigation Drawer** (o “cajón de navegación”) es un patrón de interfaz muy común en apps Android que proporciona un menú deslizable desde el lateral para acceder a las secciones principales de la aplicación. Al deslizar el dedo desde el borde izquierdo (o tocar el icono “hamburguesa”) aparece este panel. Google explica que este panel muestra *“el menú de navegación principal de la app”*. De hecho, muchas apps de Google usan este esquema para organizar su contenido; como señala la documentación, el Navigation Drawer “se desliza desde la izquierda para mostrar su contenido”.

Este patrón es útil cuando la aplicación tiene múltiples destinos de navegación de nivel superior. Sin embargo, si solo hay pocos ítems (5 o menos), Google recomienda usar un menú inferior (Bottom Navigation) en lugar del drawer. En resumen: use el Navigation Drawer para un menú lateral que contenga varias opciones de navegación global, dejando más espacio en pantalla para el contenido principal.

**Pautas de diseño (Material Design)**

Google provee pautas específicas de Material Design para el Navigation Drawer. Algunos puntos clave son:

* **Ubicación y comportamiento:** siempre aparece desde el **borde de inicio** de la pantalla (izquierdo en idiomas LTR, derecho en RTL) y cubre la altura completa de la pantalla. Suele estar por debajo de la barra de estado y se oscurece el resto al abrirlo.
* **Ancho:** el ancho sugerido es igual a la anchura de pantalla menos la altura de la *Action Bar* (aprox. 56 dp a cada lado), con un máximo de **320 dp**. En móviles usar márgenes horizontales de 16 dp y en tablets de 24 dp.
* **Iconos y márgenes:** los iconos de cada ítem se sitúan a 16 dp de los bordes izquierdo/derecho del drawer. El contenido asociado (texto, avatar) se alinea a 72 dp desde el borde izquierdo.
* **Tipografía y colores:** típicamente se usa la fuente Roboto (Regular o Medium) a 14 sp para los ítems. El color de texto suele ser #000000 al 87% para ítems normales, y #000000 100% o el ColorPrimary para el ítem seleccionado. El *ripple* táctil también resalta el ítem tocado.
* **Espaciado:** se recomienda añadir 8 dp de padding arriba/abajo de cada grupo de ítems. No es necesario padding adicional si se usa un *subheader* (ya lo incorpora).
* **Elevación:** el drawer se muestra con una sombra de elevación sobre el resto del contenido para indicar jerarquía.
* **Encabezado (opcional):** a menudo se incluye un layout de cabecera con foto/usuario en la parte superior usando el atributo app:headerLayout.

Estas pautas buscan coherencia con Material Design. Por ejemplo, el sitio oficial indica que el drawer “se desliza desde la izquierda” y describe su diseño de márgenes, tipografía y comportamiento de selección. Respetando estas especificaciones (colores, espaciados, iconografía) lograremos una UI pulida y accesible.

**Componentes esenciales**

Para implementar un Navigation Drawer en Android con Java usamos principalmente estos componentes:

* **Drawer Layout:** es el contenedor raíz (de androidx.drawerlayout.widget.DrawerLayout) que aloja todo el diseño. Dentro del DrawerLayout hay dos vistas hijas: una para el contenido principal y otra para el menú. Google señala que “declare un DrawerLayout como vista raíz. Dentro agregue un layout para el contenido principal y otro para el contenido del drawer”. Es decir, el DrawerLayout envuelve todo, permitiendo que el drawer se deslice sobre la UI.
* **NavigationView:** del paquete Material Components (com.google.android.material.navigation.NavigationView) es la vista que representa el menú lateral. Simplifica el drawer al cargar un recurso de menú XML y manejar el diseño de los ítems. Generalmente se configura dentro del DrawerLayout con android:layout\_gravity="start" para indicar que aparece desde el inicio de la pantalla. En este NavigationView se enlaza el archivo de menú con app:menu="@menu/tu\_menu". También puede incluir un encabezado con app:headerLayout="@layout/encabezado\_drawer".
* **Toolbar (ActionBar):** la barra superior de la app. Usamos un androidx.appcompat.widget.Toolbar como *ActionBar* para mostrar el título y el icono hamburguesa. En el layout principal se coloca dentro del contenido principal (no dentro del drawer). Luego en el código Java hacemos setSupportActionBar(toolbar) para usarlo.
* **Archivo de menú XML:** en res/menu/ definimos un recurso XML con los ítems del drawer. Cada ítem debe tener un android:id único, un android:icon (opcional) y un android:title. Por ejemplo:

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item

android:id="@+id/nav\_account"

android:title="Mi Cuenta"

android:icon="@drawable/ic\_user" />

<item

android:id="@+id/nav\_settings"

android:title="Configuración"

android:icon="@drawable/ic\_settings" />

<!-- Más ítems -->

</menu>

Estos ítems se mostrarán automáticamente en el NavigationView cuando usemos app:menu="@menu/tu\_menu".

En resumen, la estructura típica en XML (sin código Java) sería: un DrawerLayout raíz, dentro un layout para la UI principal (que contiene la Toolbar y un contenedor de fragmentos), y luego la NavigationView con el menú. Por ejemplo:

<?xml version="1.0" encoding="utf-8"?>

<androidx.drawerlayout.widget.DrawerLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<!-- Contenido principal: puede ser un LinearLayout o similar -->

<LinearLayout

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<!-- Toolbar como ActionBar -->

<androidx.appcompat.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

android:background="?attr/colorPrimary"

app:title="Título de la App" />

<!-- Contenedor de contenido (por ejemplo para Fragments) -->

<FrameLayout

android:id="@+id/fragment\_container"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

</LinearLayout>

<!-- NavigationView para el menú lateral -->

<com.google.android.material.navigation.NavigationView

android:id="@+id/nav\_view"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

app:headerLayout="@layout/drawer\_header" <!-- opcional: layout de encabezado -->

app:menu="@menu/navigation\_menu" /> <!-- tu menú de ítems -->

</androidx.drawerlayout.widget.DrawerLayout>

Este ejemplo concuerda con las guías de Android: el DrawerLayout como raíz con dos hijos (contenido y drawer). Observa android:layout\_gravity="start" en el NavigationView (para que aparezca por el borde izquierdo). El FrameLayout con id fragment\_container es donde cargaremos dinámicamente los contenidos de cada sección.

**Estructura de carpetas y archivos**

Un proyecto típico en Android Studio con Navigation Drawer (Java) tendría al menos estos archivos:

* **Carpeta app/java/…:** contine la clase MainActivity.java (o la *Activity* principal) donde implementamos la lógica del drawer.
* **Carpeta app/res/layout:** incluye el layout principal (activity\_main.xml) con el DrawerLayout mostrado arriba. También puede haber layouts adicionales, por ejemplo un archivo de layout para el encabezado del drawer (drawer\_header.xml).
* **Carpeta app/res/menu:** contiene el XML del menú lateral (por ejemplo navigation\_menu.xml) con los ítems del drawer.
* **Carpeta app/res/values:** aquí colocamos recursos auxiliares:
  + strings.xml (títulos de ítems, y cadenas *Open*/ *Close* para el toggle del drawer).
  + colors.xml y styles.xml (para definir colores del tema, colorPrimary, tipografía, etc.).
  + Otros recursos como imágenes de iconos en drawable/.

Por ejemplo, [13†L124-L132] enumera archivos como MainActivity.java, activity\_main.xml, navigation\_drawer\_header.xml, navigation\_drawer\_menu.xml, y recursos de estilo/colores. También es importante declarar en strings.xml las cadenas nav\_open y nav\_close, que el ActionBarDrawerToggle usará para la accesibilidad (estas describen el estado de abrir/cerrar el menú lateral).

**Ejemplo de implementación en Java**

En la actividad principal (por ejemplo MainActivity.java), debemos **configurar el DrawerLayout, el Toolbar y el NavigationView**. A continuación un esquema de código clave (en la carpeta app/java/com/tuapp/):

public class MainActivity extends AppCompatActivity

implements NavigationView.OnNavigationItemSelectedListener {

private DrawerLayout drawerLayout;

private NavigationView navigationView;

private Toolbar toolbar;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main); // usa el layout con DrawerLayout

// Referencias a las vistas

drawerLayout = findViewById(R.id.drawer\_layout);

toolbar = findViewById(R.id.toolbar);

navigationView = findViewById(R.id.nav\_view);

// Configurar Toolbar como ActionBar

setSupportActionBar(toolbar);

getSupportActionBar().setTitle("Mi App");

// Crear el toggle para el Drawer (icono "hamburguesa")

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(

this, drawerLayout, toolbar,

R.string.nav\_open, // "Abrir menú" (contentDescription para accesibilidad)

R.string.nav\_close // "Cerrar menú"

);

drawerLayout.addDrawerListener(toggle);

toggle.syncState(); // muestra el icono en el Toolbar

// Manejar selección de ítems en el NavigationView

navigationView.setNavigationItemSelectedListener(this);

}

// Este método se llama cuando se selecciona un ítem del menú lateral

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem item) {

// Identificar el ítem por su ID

int id = item.getItemId();

if (id == R.id.nav\_account) {

// Navegar a la sección Cuenta

// Ejemplo: reemplazar fragmento

getSupportFragmentManager().beginTransaction()

.replace(R.id.fragment\_container, new CuentaFragment())

.commit();

} else if (id == R.id.nav\_settings) {

// Navegar a Configuraciones

getSupportFragmentManager().beginTransaction()

.replace(R.id.fragment\_container, new SettingsFragment())

.commit();

}

// ... manejar otros ítems ...

// Cerrar el Drawer tras la selección

drawerLayout.closeDrawer(GravityCompat.START);

return true;

}

// Si presionan "atrás" con el drawer abierto, cerrarlo primero

@Override

public void onBackPressed() {

if (drawerLayout.isDrawerOpen(GravityCompat.START)) {

drawerLayout.closeDrawer(GravityCompat.START);

} else {

super.onBackPressed();

}

}

}

Este código ejemplifica lo siguiente:

* En onCreate, después de setContentView, obtenemos referencias a DrawerLayout, Toolbar y NavigationView. Luego llamamos setSupportActionBar(toolbar) para usar la toolbar como ActionBar.
* **ActionBarDrawerToggle:** creamos un objeto ActionBarDrawerToggle que enlaza el DrawerLayout con la toolbar. Le pasamos los recursos R.string.nav\_open y R.string.nav\_close para describir la acción (accesibilidad). Luego lo añadimos al Drawer con addDrawerListener y sincronizamos el estado con syncState(). Esto muestra automáticamente el icono de “hamburguesa” y lo convierte en flecha de retroceso cuando sea necesario.
* **NavigationView.OnNavigationItemSelectedListener:** implementamos esta interfaz para responder a los clics de menú. En el método onNavigationItemSelected, comprobamos item.getItemId() y realizamos la acción correspondiente (por ejemplo, reemplazar un fragmento en fragment\_container). Al final cerramos el drawer con drawerLayout.closeDrawer(GravityCompat.START).
* **onBackPressed:** opcionalmente, sobrecargamos este método para cerrar el drawer cuando esté abierto antes de cerrar la actividad. Esto mejora la usabilidad.

El código comentado anterior está basado en ejemplos comunes. Notar que AndroidX recomienda ahora el uso de Navigation Component para una gestión más sencilla del menú, pero el enfoque tradicional con ActionBarDrawerToggle y fragmentos sigue siendo válido.

**Buenas prácticas**

Para un Navigation Drawer robusto y accesible, considere lo siguiente:

* **Accesibilidad:** use cadenas descriptivas (nav\_open y nav\_close) para el toggle del drawer. Esto permite que TalkBack anuncie correctamente *“Abrir menú”* o *“Cerrar menú”* al interactuar con el icono. Asegúrese de que los ítems del menú tengan texto legible y, si es apropiado, texto alternativo para iconos. Los márgenes y tamaños recomendados (14 sp para textos, 48x48 dp para botones, etc.) ya favorecen la accesibilidad visual según Material Design.
* **Uso de Fragments:** es común que cada opción del menú lateral cargue un fragmento en el FrameLayout del contenido principal. Esto mantiene una sola actividad *master* y facilita el manejo del ciclo de vida. En el ejemplo anterior, los fragmentos CuentaFragment, SettingsFragment, etc., se reemplazan según la selección. Recuerde usar getSupportFragmentManager() (AndroidX) para gestionar estas transacciones.
* **Compatibilidad con orientación y dispositivos grandes:** en tablets o pantallas anchas se puede optar por un drawer permanente (siempre abierto) usando layouts en layout-sw600dp/ o layout-land/. Si no, al rotar el dispositivo preserve el estado del drawer (puede verificar en onCreate si estaba abierto con savedInstanceState). Android gestionará en gran medida la rotación, pero vale la pena probar que el fragmento actual y el estado del menú se mantengan. También asegúrese de usar android:layout\_gravity="start", de modo que en idiomas RTL el drawer se muestre por el lado derecho automáticamente (Android lo maneja).
* **Compatibilidad de versiones:** use la librería Material Components (com.google.android.material:material) para el NavigationView, como recomienda Google. Ya no se usan las viejas Support Libraries para esto. Agregue en el build.gradle la dependencia de Material (por ejemplo implementation 'com.google.android.material:material:1.x.x').

* **Otros consejos:** si su app combina menú inferior y lateral, recuerde respetar las pautas de Material. Evite redundancias de navegación (no tenga el mismo destino accesible desde dos menús diferentes). Y cuide que los ítems de menú estén en res/menu y sean simples, sin lógica compleja; la lógica va en la Activity o ViewModel.

En resumen, **Google enfatiza el diseño universal y la coherencia visual**. El Navigation Drawer debe implementarse siguiendo las especificaciones de espaciado, tipografía y colores de Material Design. Además, usar fragmentos para manejar el contenido garantiza una arquitectura modular y facilita la orientación de pantalla. Siguiendo estas pautas –y el código de ejemplo anterior– obtendrá un drawer funcional, accesible y bien integrado en Android Studio con Java.

**Fuentes:** Android Developers y material oficial, tutoriales reconocidos y guías prácticas, y blogs especializados (todos citados arriba) proporcionan los lineamientos y ejemplos aquí presentados.