Navigation Drawer dans Android Studio avec Kotlin

Introduction

Le **Navigation Drawer** est un menu latéral coulissant utilisé pour la navigation dans une application Android. Il permet d'afficher différentes sections ou fragments en fonction des choix de l'utilisateur.

1. L'architecture d'un Navigation Drawer repose sur plusieurs composants principaux :

📌 1.1. Structure globale

L'architecture d'un Navigation Drawer comprend généralement :

- **DrawerLayout : Conteneur principal qui englobe toute l'interface et permet d'afficher le tiroir de navigation.**
- **NavigationView : Vue qui contient le menu du tiroir de navigation.**
- **Toolbar : Barre d'outils (ActionBar) pour afficher le titre et l'icône du menu.**
- **FragmentContainer : Zone de contenu qui change dynamiquement en fonction de la navigation.**
- **Menu Items : Liste des options de navigation sous forme de menu.**
- **Header : En-tête du menu contenant des informations supplémentaires comme un logo, une image de profil, un nom d'utilisateur, etc.**

---

2. Composants graphiques du Navigation Drawer

Les principaux éléments graphiques qui composent un Navigation Drawer sont :

📌 2.1. DrawerLayout (Conteneur principal)

Il est l'élément racine de l'interface et contient tous les autres composants (*activity_main.xml*). Il permet d'afficher ou de masquer le menu de navigation.

- **XML :**

```
<androidx.drawerlayout.widget.DrawerLayout
        android:id="@+id/drawer_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true">
</androidx.drawerlayout.widget.DrawerLayout>
```

📌 2.2. Toolbar (Barre d'outils)

La Toolbar remplace l'ActionBar et affiche le bouton de menu pour ouvrir/fermer le tiroir.

- **XML :**

```
<androidx.appcompat.widget.Toolbar
   android:id="@+id/toolbar"
   android:layout_width="match_parent"
   android:layout_height="56dp"
   android:background="@color/lavender"
   android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>
```

📌 2.3. NavigationView (Vue de navigation)

Il contient le menu qui affiche les éléments de navigation sous forme de liste.

- **XML :**

```
<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/nav_header"
        app:menu="@menu/nav_menu"/>
```

📌 2.4. nav_menu.xml (Menu de navigation)

C'est un fichier de menu qui contient les options de navigation sous forme d'éléments.

- **XML :**

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <group android:checkableBehavior="single">
          <item
            android:id="@+id/nav_home"
            android:icon="@drawable/nav_home"
            android:title="Home"/>
             ...
        </group>
</menu>
```

📌 2.5. nav_header.xml (En-tête du Navigation Drawer)

Il contient des informations supplémentaires, comme une image de profil, un logo ou un email.

- **XML :**

```
<LinearLayout
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="176dp"
   android:gravity="bottom"
   android:padding="16dp"
   android:background="@drawable/headerbkg">
        ....
</LinearLayout>
```

📌 2.6. Fragments pour le contenu

Chaque option du menu ouvre un Fragment correspondant pour afficher le bon contenu.

Exemple d'un HomeFragment :

- **Kotlin :**

class HomeFragment : Fragment(R.layout.fragment_home)

---

3. Fonctionnement global

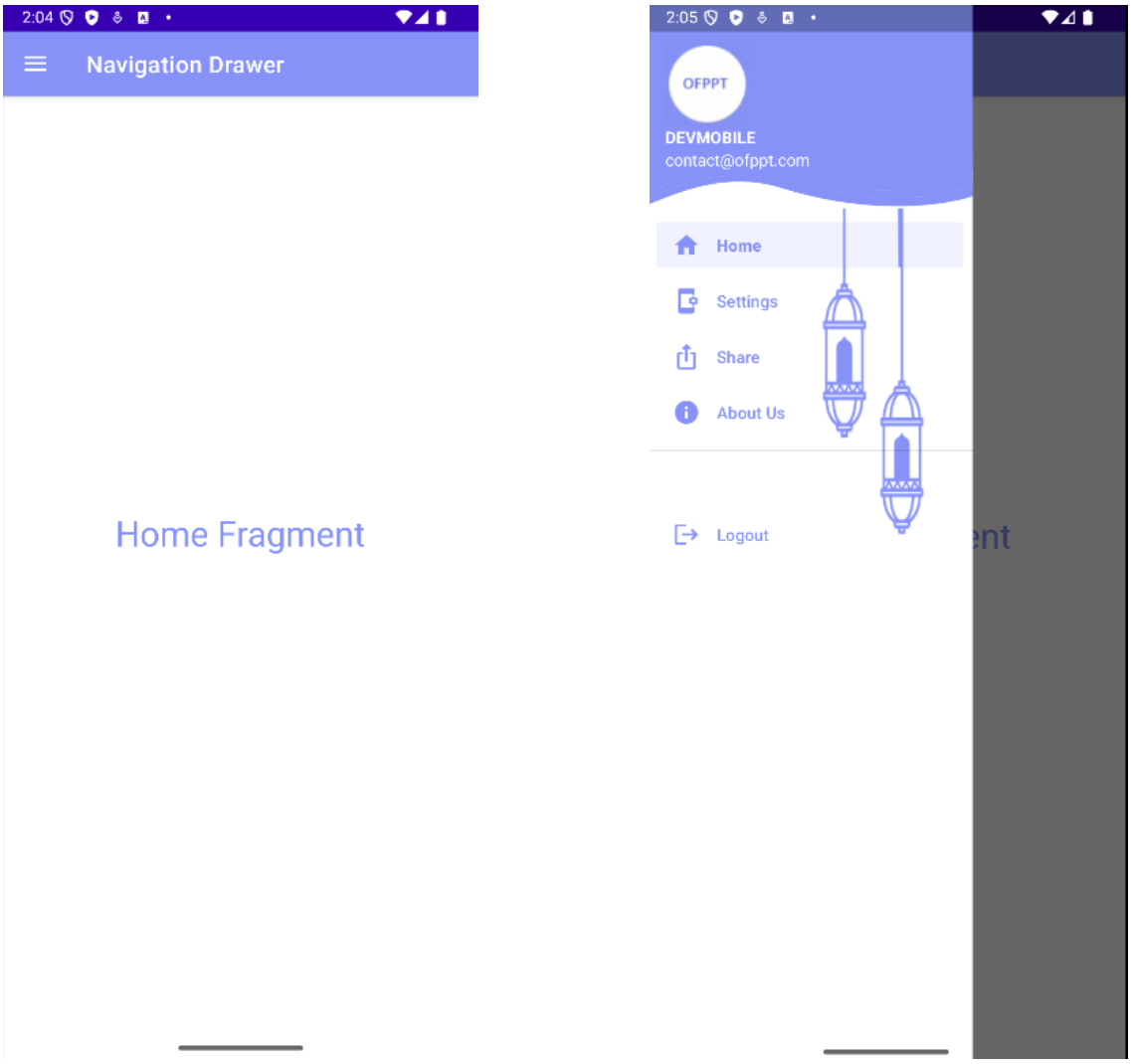
1. L'utilisateur ouvre le menu : en balayant l'écran ou en appuyant sur le bouton du menu.
2. Le menu affiche une liste d'options : Home, Settings, Share, About Us, Logout.
3. L'utilisateur sélectionne une option :
    - Si c'est une option de navigation, le contenu principal change pour afficher le fragment correspondant.
    - Si c'est Logout, un message de déconnexion apparaît.
4. Le menu se ferme automatiquement après la sélection.

L'architecture du Navigation Drawer repose sur DrawerLayout pour la structure, NavigationView pour le menu et Fragments pour afficher le contenu dynamique. Ce composant est très utilisé pour créer une navigation fluide et intuitive dans les applications Android.

TP Navigation Drawer sous Android :

Guide Complet

Dans ce guide, nous allons détailler les **étapes de mise en place** d'un Navigation Drawer avec **Kotlin**, en utilisant **Android Studio** et le **Material Design**.



---

1. Configuration des ressources

a) Modifier le fichier colors.xml

Dans le dossier res/values/colors.xml, nous définissons des couleurs personnalisées :

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="lavender">#8692f7</color>
</resources>
```

b) Modifier le fichier strings.xml

Dans res/values/strings.xml, définissons les chaînes de texte :

```xml
<resources>
    <string name="app_name">Navigation Drawer</string>
    <string name="open_nav">Open Navigation Drawer</string>
    <string name="close_nav">Close Navigation Drawer</string>
    <!-- TODO: Remove or change this placeholder text -->
    <string name="hello_blank_fragment">Hello blank fragment</string>
</resources>
```

c) Modifier le fichier themes.xml

Ce fichier contient **deux styles principaux** :

1. **Theme.NavigationDrawer** : Thème pour les écrans qui utilisent une **ActionBar**.
2. **Theme.MainActivity** : Thème pour MainActivity, qui **n'a pas d'ActionBar**.

Chaque thème hérite d'un **thème de base** fourni par **Material Components**, ce qui garantit une compatibilité avec le **Material Design**.

Ajoutez le thème personnalisé dans res/values/themes.xml :

```xml
<resources xmlns:tools="http://schemas.android.com/tools">
   <!-- Base application theme. -->
   <style name="Theme.NavigationDrawer"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
      <!-- Primary brand color. -->
      <item name="colorPrimary">@color/lavender</item>
      <item name="colorPrimaryVariant">@color/lavender</item>
      <item name="colorOnPrimary">@color/white</item>
      <!-- Secondary brand color. -->
      <item name="colorSecondary">@color/teal_200</item>
      <item name="colorSecondaryVariant">@color/teal_700</item>
      <item name="colorOnSecondary">@color/black</item>
```

```
    <!-- Status bar color. -->
    <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
    <!-- Customize your theme here. -->
  </style>
  <style name="Theme.MainActivity" parent="Theme.MaterialComponents.DayNight.NoActionBar">
    <!-- Primary brand color. -->
    <item name="colorPrimary">@color/lavender</item>
    <item name="colorPrimaryVariant">@color/lavender</item>
    <item name="colorOnPrimary">@color/white</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/teal_200</item>
    <item name="colorSecondaryVariant">@color/teal_700</item>
    <item name="colorOnSecondary">@color/black</item>
    <!-- Status bar color. -->
    <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
    <!-- Customize your theme here. -->
  </style>
</resources>
```

Élément	Explication
Theme.NavigationDrawer	Thème avec **ActionBar** basé sur Material Design
Theme.MainActivity	Thème **sans ActionBar** (utile si on utilise une Toolbar personnalisée)
colorPrimary	Couleur principale (lavande)
colorPrimaryVariant	Variante de colorPrimary
colorOnPrimary	Couleur du texte/icônes sur colorPrimary
colorSecondary	Couleur secondaire pour les accents (teal)
colorSecondaryVariant	Variante de colorSecondary
colorOnSecondary	Couleur du texte sur colorSecondary
android:statusBarColor	Couleur de la **barre de statut**

d) Modifier le fichier AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Navigation Drawer"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.NavigationDrawer"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:theme="@style/Theme.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />


                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>

</manifest>
```

---

2. Ajouter des icônes Vector Assets

Ajoutez des **Vector Assets** dans res/drawable/ :

1. Faites **un clic droit** sur res/drawable → New → Vector Asset.
2. Ajoutez les icônes home, settings, logout, info, share, avec le code couleur : ***8692F7***.
3. Copier les image dans drawable.

---

3. Création du menu de navigation (nav_menu.xml)

Dans res/menu/, créez nav_menu.xml :

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    tools:showIn="navigation_view">

    <group
        android:checkableBehavior="single">
        <item
            android:id="@+id/nav_home"
            android:icon="@drawable/baseline_home_24"
            android:title="Home"/>
        <item
            android:id="@+id/nav_settings"
            android:icon="@drawable/baseline_app_settings_alt_24"
            android:title="Settings"/>
        <item
            android:id="@+id/nav_share"
            android:icon="@drawable/baseline_ios_share_24"
            android:title="Share"/>
        <item
            android:id="@+id/nav_about"
            android:icon="@drawable/baseline_info_24"
            android:title="About Us"/>
    </group>
    <item
        android:title="">
        <menu>
            <item
                android:id="@+id/nav_logout"
                android:icon="@drawable/baseline_logout_24"
                android:title="Logout"/>
        </menu>
    </item>
</menu>
```

- **<menu>** → Conteneur principal pour définir les éléments du menu.
- **xmlns:android** → Namespace nécessaire pour utiliser les attributs Android.
- **xmlns:tools et tools:showIn="navigation_view"** → Indique que ce menu est destiné à être affiché dans un **NavigationView**.

---

4. Créer l'en-tête du menu (nav_header.xml)

Dans res/layout/, créez nav_header.xml - (RootElement = LinearLayout):

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="176dp"
    android:gravity="bottom"
    android:padding="16dp"
    android:background="@drawable/headerbkg"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/aklogowhite" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DEVMOBILE"
        android:textColor="@color/white"
        android:textStyle="bold"/>

    <TextView
        android:layout_width="134dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:text="contact@ofppt.com"
        android:textColor="@color/white"
        android:textSize="14sp" />
</LinearLayout>
```

---

5. Modifier activity_main.xml

Dans res/layout/activity_main.xml, utilisez un **DrawerLayout** :

```xml
<?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"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:id="@+id/drawer_layout"
   android:fitsSystemWindows="true"
   tools:openDrawer="start"
   tools:context=".MainActivity">

   <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical">
```

```
    <androidx.appcompat.widget.Toolbar
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:id="@+id/toolbar"
        android:elevation="4dp"
        android:background="@color/lavender"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        android:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/fragment_container"/>
    <androidx.coordinatorlayout.widget.CoordinatorLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <!--<com.google.android.material.bottomappbar.BottomAppBar
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/bottomAppBar"
            android:layout_gravity="bottom"
            android:backgroundTint="@color/white"
            app:fabCradleMargin="10dp"
            app:fabCradleRoundedCornerRadius="50dp">

            <com.google.android.material.bottomnavigation.BottomNavigationView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/bottom_navigation"
                app:labelVisibilityMode="labeled"
                app:menu="@menu/bottom_menu"
                >
            </com.google.android.material.bottomnavigation.BottomNavigationView>
        </com.google.android.material.bottomappbar.BottomAppBar>
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="@color/white"
        android:src="@drawable/baseline_add_24"
        app:layout_anchor="@id/bottomAppBar"
        app:maxImageSize="40dp"
        android:id="@+id/fab"/>-->


    </androidx.coordinatorlayout.widget.CoordinatorLayout>

</RelativeLayout>
```

```
    <com.google.android.material.navigation.NavigationView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/nav_view"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header"
        app:menu="@menu/nav_menu"
        android:background="@drawable/menubkg"
        app:itemIconTint="@color/lavender"
        app:itemTextColor="@color/lavender"/>

</androidx.drawerlayout.widget.DrawerLayout>
```

Ce code XML définit l'interface principale de l'application en utilisant **DrawerLayout**, un **Navigation Drawer** avec un **Toolbar** et un **LinearLayout** pour afficher des fragments.

---

6. Créer les Fragments

Ajoutez ces **Fragments** :

1. HomeFragment
2. SettingsFragment
3. ShareFragment
4. AboutFragment

---

7. Modifier MainActivity.kt

Dans MainActivity.kt, implémentez le NavigationView.OnNavigationItemSelectedListener :

```
class MainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener
{
    private lateinit var drawerLayout: DrawerLayout

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        //Récupération des vues
        drawerLayout = findViewById<DrawerLayout>(R.id.drawer_layout)
        val toolbar = findViewById<Toolbar>(R.id.toolbar)
        //Définit la Toolbar comme la barre d'action principale
         setSupportActionBar(toolbar)
         val navigationView = findViewById<NavigationView>(R.id.nav_view)
```

```
//Écoute les clics sur les éléments du menu et appelle onNavigationItemSelected() (défini plus bas)
 navigationView.setNavigationItemSelectedListener(this)

    // Ajoute un bouton (≡) à la Toolbar pour ouvrir/fermer le menu
    //ActionBarDrawerToggle : helper qui lie le DrawerLayout à la Toolbar et affiche l'icône du menu hamburger.
    //toolbar : La Toolbar qui remplace la ActionBar dans l'application.
    //R.string.open_nav & R.string.close_nav : Ce sont des chaînes de caractères définies
    // dans res/values/strings.xml utilisées pour l'accessibilité (par exemple, pour les lecteurs d'écran).
    val toggle = ActionBarDrawerToggle(this, drawerLayout, toolbar, R.string.open_nav, R.string.close_nav)
    // Connecte le bouton à l'animation du DrawerLayout
    //ajoute l'objet toggle comme listener au drawerLayout
    //Cela signifie que toggle va écouter les événements d'ouverture et de fermeture du menu latéral.
     drawerLayout.addDrawerListener(toggle)
    // Met à jour l'état du bouton pour refléter l'ouverture/fermeture du menu.
    //Cette méthode synchronise l'icône du menu hamburger avec l'état actuel du DrawerLayout.
     toggle.syncState()
    // Charge le HomeFragment par défaut si aucune instance précédente n'existe.
        if (savedInstanceState == null) {
            supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, HomeFragment()).commit()
    // Marque l'élément nav_home comme sélectionné dans le menu.
            navigationView.setCheckedItem(R.id.nav_home)
        }

    }

    override fun onNavigationItemSelected(item: MenuItem): Boolean {
    // Vérifie quel élément du menu a été cliqué.
        when (item.itemId) {
    // Remplace le fragment affiché dans fragment_container
            R.id.nav_home -> supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, HomeFragment()).commit()
            R.id.nav_settings -> supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, SettingsFragment()).commit()
            R.id.nav_share -> supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, ShareFragment()).commit()
            R.id.nav_about -> supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, AboutFragment()).commit()
    // Cas particulier : R.id.nav_logout affiche juste un Toast ("Logout!") sans changer de fragment.
            R.id.nav_logout -> Toast.makeText(this, "Logout!",
```

```
Toast.LENGTH_SHORT).show()
        }
        // Ferme le menu après la sélection
        drawerLayout.closeDrawer(GravityCompat.START)
        return true
    }
    override fun onBackPressed() {
        super.onBackPressed()
        if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawer(GravityCompat.START)
        } else {
            onBackPressedDispatcher.onBackPressed()
        }
    }
```

Ce code définit la classe **MainActivity**, qui implémente un **Navigation Drawer** avec gestion des clics sur les éléments du menu.

Conclusion

Nous avons mis en place un **Navigation Drawer** complet avec des **fragments**, un **menu**, et une **interface Material Design**.

Ce menu permet une navigation fluide et structurée au sein de l'application. Vous pouvez désormais personnaliser votre **drawer** et y ajouter d'autres fonctionnalités selon vos besoins.

Vous pensez avoir terminé ? Détrompez-vous ! Il reste encore du défi pour les plus forts… 💪 🔥 👇

De la même manière que le menu de navigation (**Navigation Drawer**), ajoutez un menu inférieur (**Bottom Navigation**) contenant les éléments de votre choix. Ensuite, associez un écouteur d'événements (**listener**) à chaque élément du menu afin d'afficher un message Toast correspondant lorsqu'un utilisateur clique dessus. Cela permettra d'assurer une interaction fluide et réactive avec l'interface. 🚀👍