Navigation Drawer

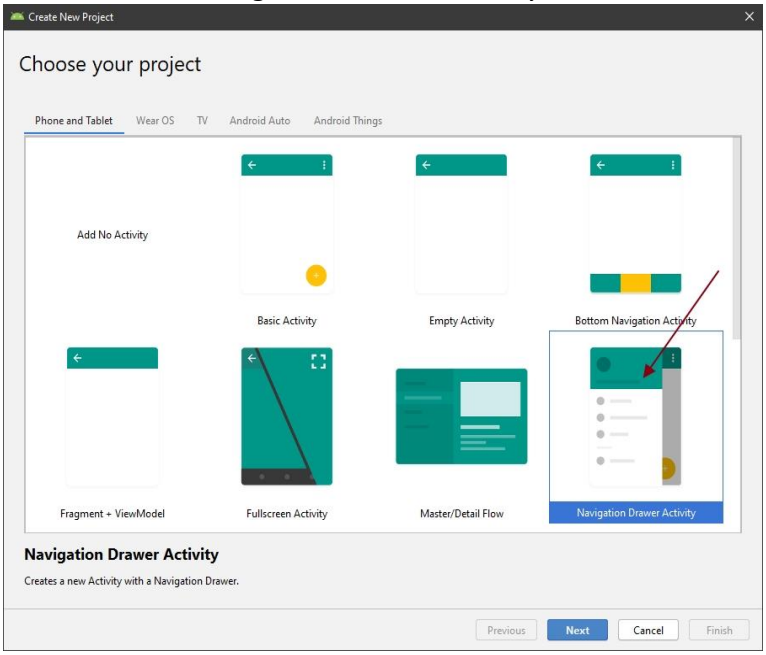
Contenidos

Navigation Drawer (Cajón de navegación)

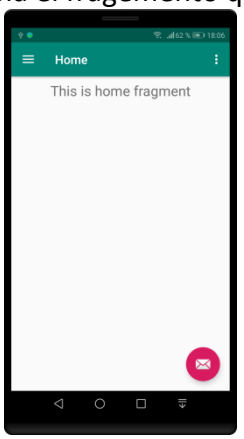
El menú lateral deslizante, o *Navigation Drawer*, es el que aparece en muchas aplicaciones al deslizar el dedo desde el borde izquierdo de la pantalla hacia el lado opuesto (también puede aparecer en el lado derecho, pero es menos frecuente).
El menú lateral debe cumplir en todo lo posible las directrices marcadas por las guías de diseño de *Material Desig:*

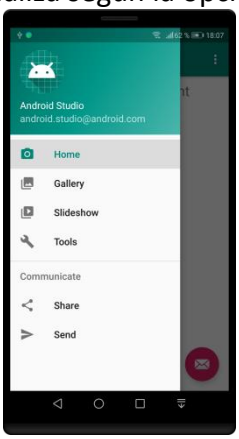
Proyecto NavigationDrawer

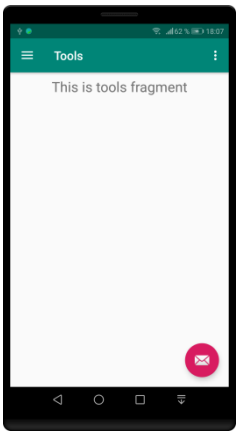
Para empezar a usar el menú lateral, podemos aprovechar las facilidades de AS y crear un proyecto llamado P_60_NavigationDrawer cuya *MainActivity* use la plantilla suministrada con el nombre *Navigation Drawer Activity:*



Si ejecutamos la aplicación observamos que cuenta con un cajón de navegación que cambia el fragemento que se visualiza según la opción elegida:



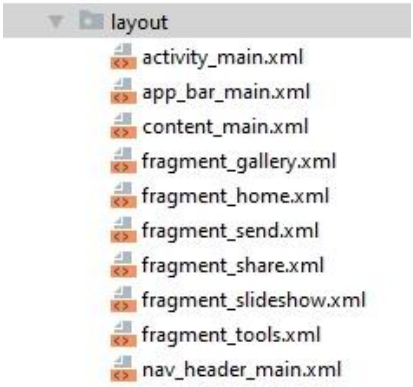




Vamos a estudiar los elementos suministrados por la citada plantilla.

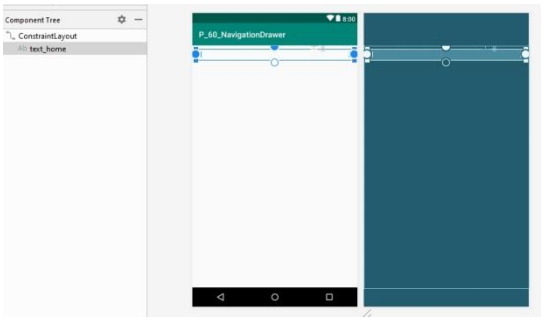
Interfaz gráfica

Observamos que el asistente nos ha creado múltiples *layouts*:



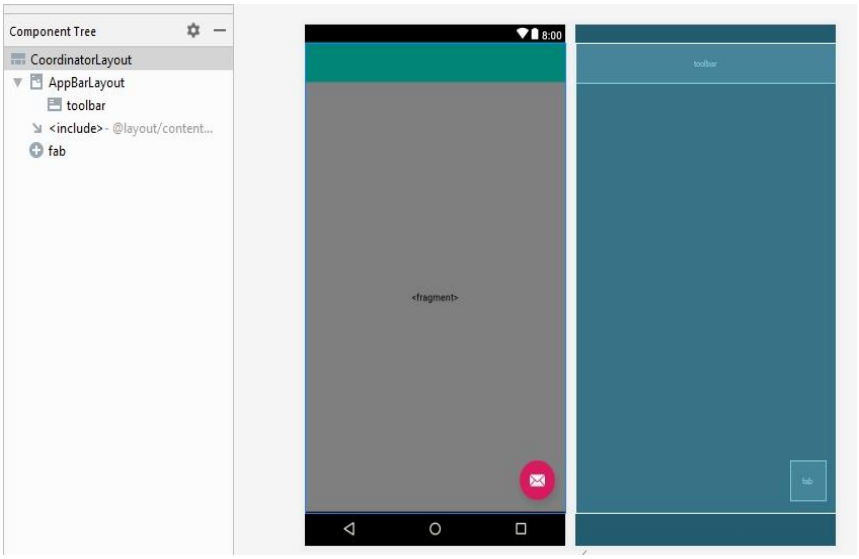
Layout de los fragmentos

Los que comienzan por fragment son los relativos a los diferentes fragmentos que se utilizan para cada una de las opciones. La plantilla crea para todos ellos un ConstraintLayout con un TextView dentro.



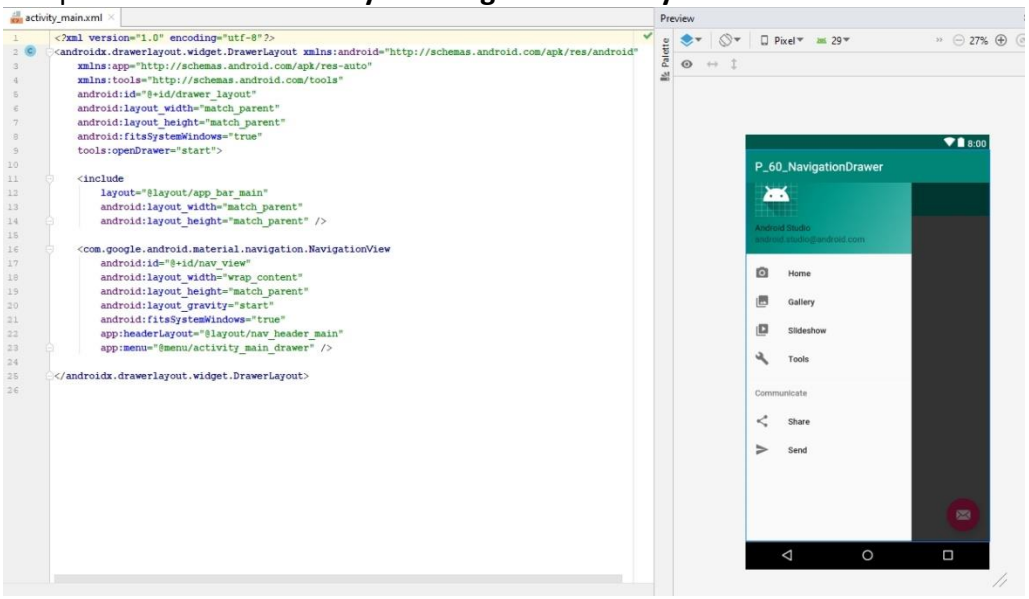
Layout app_bar_main

Incluye la *AppBar* y el *layout* content_main. En anteriores proyectos era nuestro "viejo conocido" *layout* activty_main.xml que ahora pasa a llamarse app_bar_main.xml y que deberá estar incluido junto al cajón de navegación en el layout activity_main



Layout de la actividad: activity_main

Para añadir el *Navigation Drawer* a una actividad el elemento raíz de su *layout* debe ser del tipo **<androidx.drawerlayout.widget.DrawerLayout>**.



Y dentro de este elemento se colocan únicamente 2 componentes (en el orden indicado):

- El *layout* real de la actividad que se añade en forma de <include>:

```
<include
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

- El *layout* del menú lateral, que entre otras cosas hará las veces de contenedor de las distintas opciones del 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"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
```

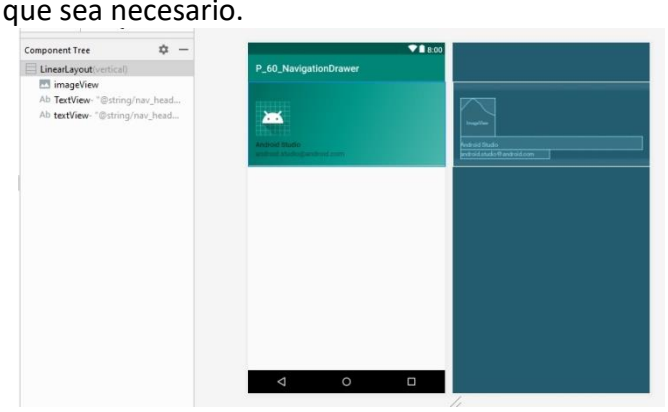
Usa la clase *NavigationView* para crear un Navigation Drawer basado en *Material Design.*

Propiedades más relevantes del nuevo componente:

- **android:fitsSystemWindows**: para conseguir el deslizamiento del menú por debajo de la Status Bar.
- **android:layout_gravity**: determina el lado de la pantalla por el que aparecerá el menú deslizante ("start" para que aparezca por la izquierda o "end" por la derecha).
- **app:headerLayout**: (opcional) asignamos al menú lateral el layout XML de su cabecera, es decir, de la zona que queda por encima de la lista de opciones del menú.
- **app:menu**: indicamos el recurso de menú que mostraremos en el Navigation Drawer. El componente utiliza el sistema de menús habitual de Android ya conocido.

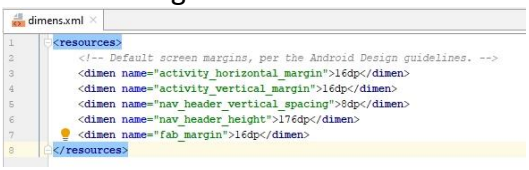
Layout cabecera: nav_header_main

La plantilla proporciona un *layout* con un *ImageView* y dos *TextView* pero puede ser todo lo complejo que sea necesario.



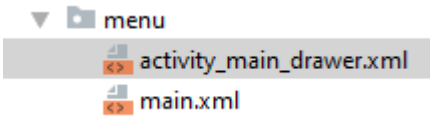
Dimensiones

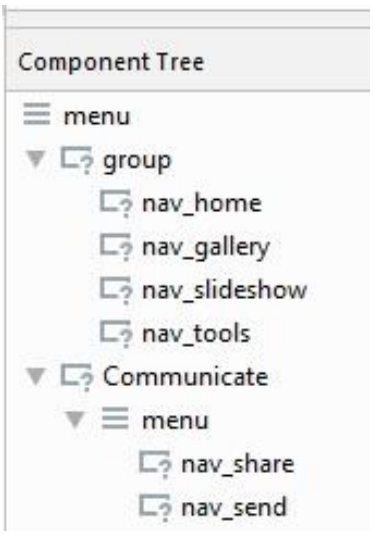
La plantilla también nos proporciona las dimensiones necesarias teniendo en cuenta las especificaciones de Material Design:



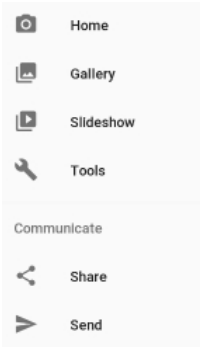
Opciones del menú

Dentro de la carpeta menu, el asistente de AS ha creado el fichero activity_main_drawer.xml con las opciones del menú lateral:





Nota AS 3.5.2:



No visualiza el menú en la pestaña Diseño del editor.

Encontrarás soluciones como quitar la línea tolos:showIn (pero entonces lo visualiza como si fuera un menú de la AppBar) o que cambies la versión de la librería material en el fichero build.gradle. Yo todavía no he encontrado la solución buena ☹!

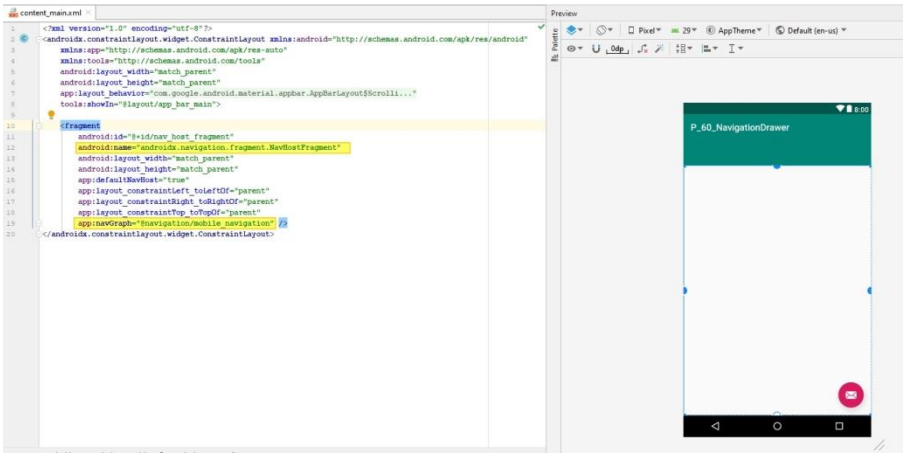
Hasta que lo solucionen siempre lo podemos ver en el layout activity_main.

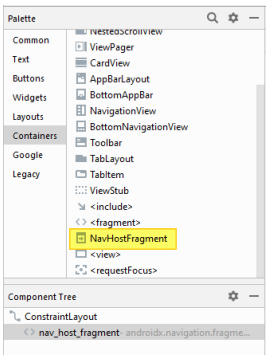
Es un menú de opciones típico en el que todas tienen su id (android:id), su icono (android:icon) y su título (android:title).

Introducción al componente de Android JetPack Navigation

Layout content_main

El fichero tiene un "fragmento especial" con unas propiedades que hacen uso del componente Navigation de Android JetPack:



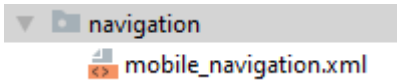


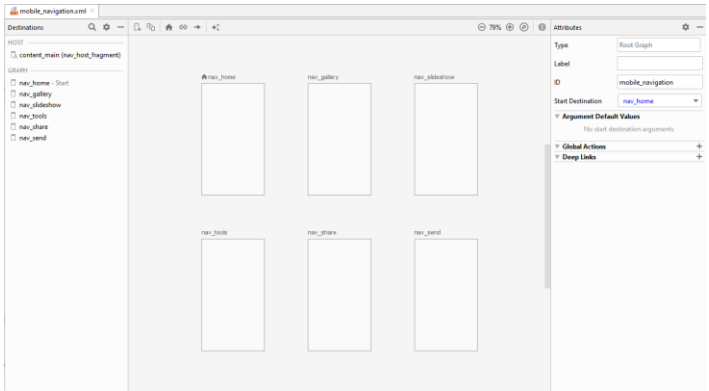
El componente Navigation consta de tres partes claves:

Gráfico de navegación: Es un recurso XML que contiene toda la información relacionada con la navegación por las partes de la aplicación.
NavHost: Es un contenedor vacío para los destinos del gráfico de navegación.
NavController: Es un objeto que administra la navegación de la app dentro de un NavHost.

Gráfico de navegación

Es un fichero xml, guardado en la carpeta res/navigation. Al abrirlo desde el editor de Navegación podemos visualizarlo en modo texto o en modo diseño.

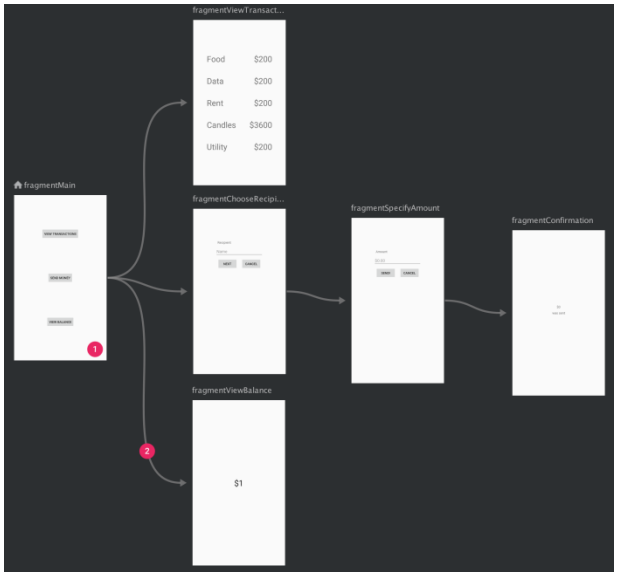




La navegación se produce entre los destinos de la aplicación, es decir, en cualquier parte de la aplicación a la que los usuarios puedan navegar. Estos destinos están conectados a través de acciones.
Un gráfico de navegación es un archivo de recursos que contiene todos sus destinos y acciones. El gráfico representa todas las rutas de navegación de su aplicación.

Cada destino se representa por una miniatura de vista previa y las acciones de conexión están representadas por flechas que muestran cómo los usuarios pueden navegar de un destino a otro.

Ej:



Editor de navegación:

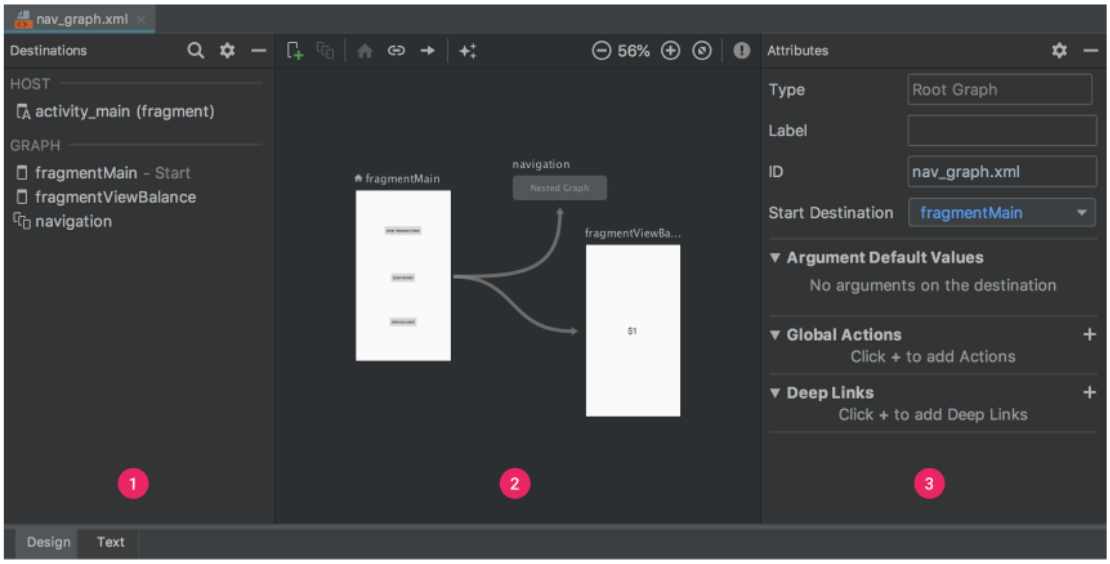


Figura 2. El editor de navegación

1. **Panel de destinos** : enumera su host de navegación y todos los destinos actualmente en el **editor de gráficos** .
2. **Editor de gráficos** : contiene una representación visual de su gráfico de navegación. Puede cambiar entre la vista de **diseño** y la representación XML subyacente en la vista de **texto** .
3. **Atributos** : muestra los atributos para el elemento seleccionado actualmente en el gráfico de navegación.

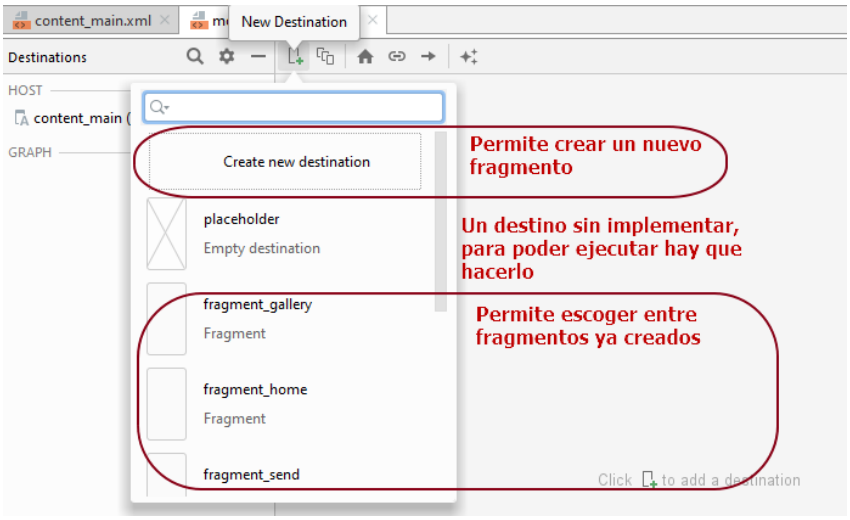
NavHost

Una de las partes centrales del componente de navegación es el host de navegación. El host de navegación es un contenedor vacío donde los destinos se intercambian dentro y fuera a medida que un usuario navega por su aplicación.

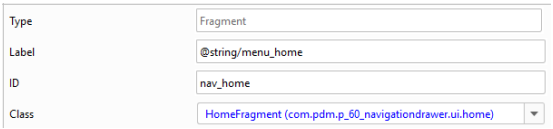
El componente de navegación está diseñado para aplicaciones que tienen una actividad principal con múltiples fragmentos destinos. La actividad principal está asociada con un gráfico de navegación y contiene uno NavHostFragment que es responsable de intercambiar destinos según sea necesario (por tanto, para que el editor de navegación pueda trabajar es necesario que algún layout lo tenga, en nuestro caso content_main.xml).

En una aplicación con múltiples actividades, cada actividad tiene su propio gráfico de navegación.

Agregar destinos



Cada destino tiene los siguientes atributos:



- **Tipo**: indica si el destino se implementa como un fragmento, actividad u otra clase personalizada en el código fuente.
- **Label**: contiene el título del destino.
- **ID**: identificador para referirse al destino en código, **debe coincidir con el que tenga en el menú**.
- **Class**: el nombre de la clase asociada con el destino.

Y si el destino es un diálogo que no tiene layout? Hay que añadirlo por texto XML.

Más adelante veremos más características del Editor de navegación, para añadir acciones, animaciones, etc.

Destino inicial

El destino de inicio es la primera pantalla que los usuarios ven al abrir su aplicación, y es la última pantalla que ven los usuarios al salir de su aplicación. El editor de navegación utiliza un icono para indicar el destino de inicio.

Entiendiendo el código

DrawerLayout

Obtenemos la referencia al cajón de navegación:

```
DrawerLayout drawer = findViewById(R.id.drawer_layout);
```

La clase DrawerLayout tiene varios métodos importantes:

- El método DrawerLayout.openDrawer() abre el menú y recibe un parámetro GravityCompat.START para indicar que se desplegará desde el lado izquierdo.
- Similarmente se cierra el drawer con DrawerLayout.closeDrawer() justo en el momento que se procesa la selección del ítem.
- También se puede comprobar el estado del menú con el método DrawerLayout.isDrawerOpen().

NavController

La navegación a un destino se realiza desde el código mediante objetos de la clase NavController, que administran la navegación de la aplicación dentro de un NavHost. Cada uno NavHosttiene su propio correspondiente NavController.

Puede usar un NavController mediante uno de los siguientes métodos:

- NavHostFragment.findNavController(Fragment)
- Navigation.findNavController(Activity, @IdRes int viewId)
- Navigation.findNavController(View)

En nuestro código de MainActivity:

```
NavController navController = Navigation.findNavController(this, R.id.nav_host_fragment);
```

NavigationUI

El componente de navegación incluye una clase NavigationUI. Esta clase contiene métodos estáticos que administran la navegación con la AppBar, el cajón de navegación y la navegación inferior.

NavigationUI usa un objeto AppBarConfiguration para administrar el comportamiento del botón de navegación de la esquina superior izquierda del área de visualización de la aplicación. De manera predeterminada, el botón de Navegación está oculto cuando un usuario se encuentra en un destino de nivel superior de un gráfico de navegación y aparece como un botón Up en cualquier otro destino. Si se trata de un cajón de navegación siempre aparece el icono hamburguesa.

Para utilizar el destino inicial del gráfico de navegación como el único destino de nivel superior, se crea un objeto AppBarConfiguration y se le pasa el gráfico de navegación correspondiente:

```
AppBarConfiguration appBarConfiguration = new
AppBarConfiguration.Builder(navController.getGraph()).build();
```

Para personalizar destinos que se consideran de nivel superior, se puede pasar un conjunto de ID de destino al constructor:

```
AppBarConfiguration appBarConfiguration = new
AppBarConfiguration.Builder(R.id.main, R.id.android).build();
```

Añadir ToolBar

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    setContentView(R.layout.activity_main);
    ...

    NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
    AppBarConfiguration appBarConfiguration =
            new AppBarConfiguration.Builder(navController.getGraph()).build();
    Toolbar toolbar = findViewById(R.id.toolbar);
    NavigationUI.setupWithNavController(toolbar, navController);
}
```

Para agregar soporte de navegación a la barra de acción predeterminada, hay que llamar al método setupActionBarWithNavController() en onCreate() de la actividad y también sobreescribir el método onSupportNavigateUp():

```
AppBarConfiguration appBarConfiguration;

...

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...

    NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
    appBarConfiguration = new
AppBarConfiguration.Builder(navController.getGraph()).build();
    NavigationUI.setupActionBarWithNavController(this, navController,
appBarConfiguration);
}
…
@Override
public boolean onSupportNavigateUp() {
    NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
    return NavigationUI.navigateUp(navController, appBarConfiguration)
            || super.onSupportNavigateUp();
}
```

Añadir cajón de navegación

Podemos considerar que el cajón de navegación como un conjunto de destinos todos considerados de nivel superior que hemos citado antes, solo hay que conectarle el DrawerLayer correspondiente.

Observa nuestra MainActivity:

```
DrawerLayout drawer = findViewById(R.id.drawer_layout);
NavigationView navigationView = findViewById(R.id.nav_view);
// Passing each menu ID as a set of Ids because each
// menu should be considered as top level destinations.
mAppBarConfiguration = new AppBarConfiguration.Builder(
        R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow,
        R.id.nav_tools, R.id.nav_share, R.id.nav_send)
        .setDrawerLayout(drawer)
        .build();
```

A continuación

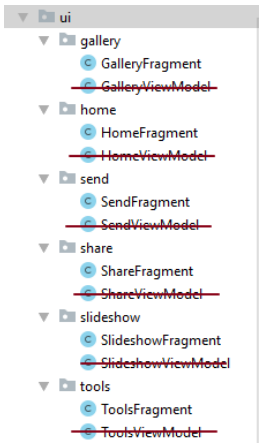
```
NavigationUI.setupWithNavController(navigationView, navController);
```

Observación: si nuestro menú tiene iconos propios con color para que no salgan con la "tintura" gris recomendada por Material Design hay que añadir:

```
navigationView.setItemIconTintList(null);
```

Componentes de arquitectura ViewModel y LifeData

La plantilla suministrada por AS, también hace uso de estos componentes de JetPack. Nosotros no los vamos a ver ahora (es mucha novedad para un solo tema ☺!).



Así que vamos a trabajar con los fragmentos como lo hacíamos hasta ahora; por lo tanto, vamos a borrar las clases que los utilizan.

En las clases de los fragmentos tendremos que borrar las referencias a los objetos de las clases borradas.

Y ya que estamos aquí, observa la buena organización de las clases en carpetas.

Práctica propuesta

Realiza los ejercicios planteados en Ejercicios_14_NavigationDrawer