UI Avanzada

Profesor: Ana Isabel Vegas

INDICE

1.- PESTAÑAS

Para crear una interfaz de usuario con pestañas, es necesario utilizar FragmentTabHost y TabWidget. FragmentTabHost debe ser el nodo raíz para el diseño, que contendrá tanto el TabWidget para la visualización de las pestañas, como un FrameLayout para mostrar el contenido.

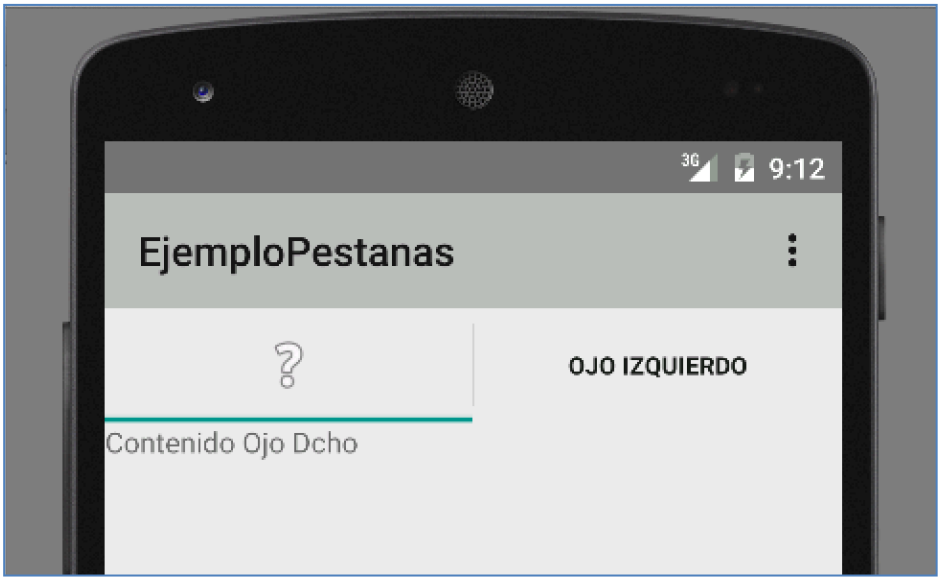

Gráfico 1. Diseño con pestañas

COMPONENTES DEL LAYOUT

Los elementos a utilizar en el layout son los siguientes:

TabHost: Es el nodo raíz que contiene el TabWidget y un FrameLayout. Su id debe ser @android:id/tabhost obligatoriamente.

TabWidget: Es un componente necesario para la visualización de las pestañas. En este caso el id también debe ser @android:id/tabs y no se puede cambiar.

FrameLayout: Aquí se define el contenido de cada pestaña.

```xml
<TabHost
    android:id="@android:id/tabhost"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <LinearLayout
                android:id="@+id/contenido1"
                android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent">

                <TextView
                    android:text="Contenido Ojo Dcho"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"/>

            </LinearLayout>
            <LinearLayout
                android:id="@+id/contenido2"
                android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent">
```

Gráfico 2. Layout con pestañas

Desde la actividad creamos las pestañas:

```java
// Crear las pestañas
TabHost tabs = (TabHost)
        findViewById(android.R.id.tabhost);
tabs.setup();
```

Gráfico 3. Crear las pestañas desde MainActivity

A continuación se añade el contenido de cada pestaña:

```
// Crear el contenido de la segunda
spec = tabs.newTabSpec("ojoIzdo");
spec.setContent(R.id.contenido2);
spec.setIndicator("OJO Izquierdo", null);
tabs.addTab(spec);
```

Gráfico 4. Añadir el contenido de la pestaña

Seleccionamos la pestaña a mostrar:

```
// Seleccionamos la pestaña a mostrar
tabs.setCurrentTab(0);
```

Gráfico 5. Seleccionar la pestaña a mostrar

Añadir el listener, este detectara cuando el usuario pulsa en la pestaña a mostrar.

```
// Creamos un listener para detectar el cambio de pestaña
tabs.setOnTabChangedListener(new TabHost.OnTabChangeListener() {

    @Override
    public void onTabChanged(String tabId) {
        Toast.makeText(getApplicationContext(),
                "HAs pulsado " + tabId,
                Toast.LENGTH_LONG).show();

    }
});
```

Gráfico 6. Añadir un listener

MOSTRAR IMÁGENES EN LA PESTAÑA

Como titulo de la pestaña puede aparecer texto o una imagen. A continuación vemos como hacerlo.

```
// Crear el contenido de la primera
TabHost.TabSpec spec = tabs.newTabSpec("ojoDcho");
spec.setContent(R.id.contenido1);
spec.setIndicator("", getResources().getDrawable(
        android.R.drawable.ic_menu_help));
tabs.addTab(spec);
```

Gráfico 7. Añadir imagen en la pestaña

2.- FRAGMENTS

Un fragment es un comportamiento (si el fragment es no visible) o una porción de la interfaz gráfica (si el fragment es visible). Es decir, es un módulo que puede unirse a otros.

Surgieron para que una misma aplicación pudiera ser multipantalla. Por este motivo, su mayor virtud es la reutilización de código.

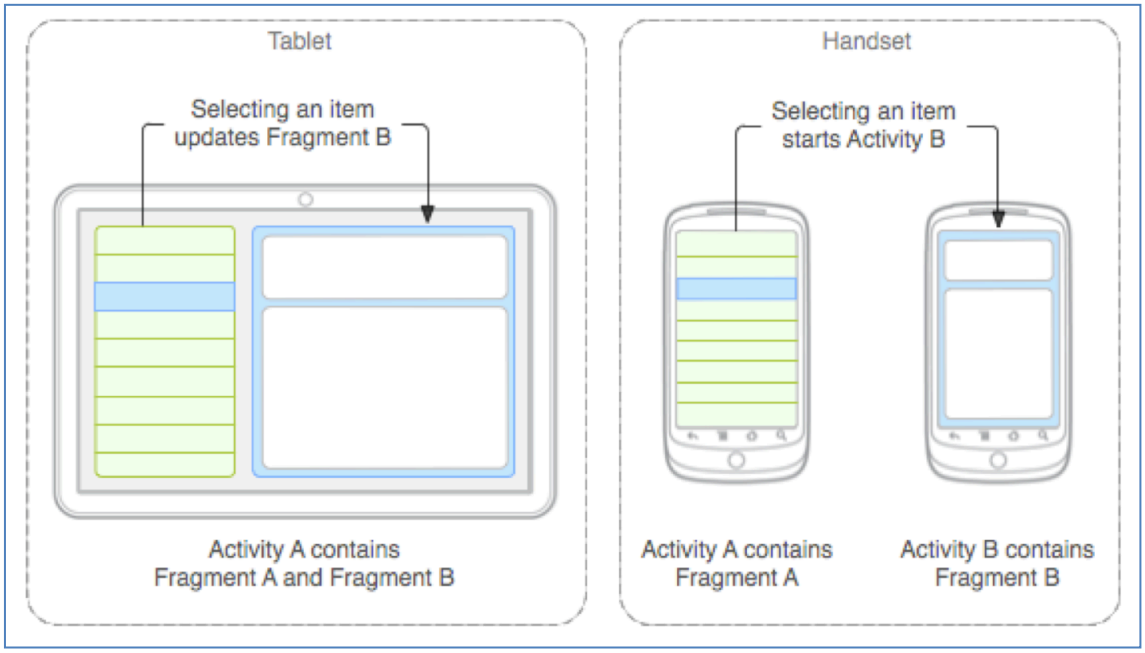

Gráfico 8. Fragments

Supongamos que tenemos ciertas funcionalidades en varios Fragments. Para las diferentes Activitys, dependiendo del dispositivo, podremos acomodarlos de diferente manera. Hay que atender a que si nos pasamos poniendo Fragments para cierto tamaño de pantalla, se verá el contenido de cada Fragment demasiado pequeño y podría ser difícil de manejar; por el contrario, quedarnos supondría desaprovechar la pantalla.

Por ejemplo podríamos tener diferente números de fragments en función de la pantalla del dispositivo.

- Smartphone: en el que cabe 1 Fragment por Activity

Gráfico 9. 1 fragment por Activity

- Tablet: en el que caben 2 Fragments por Activity

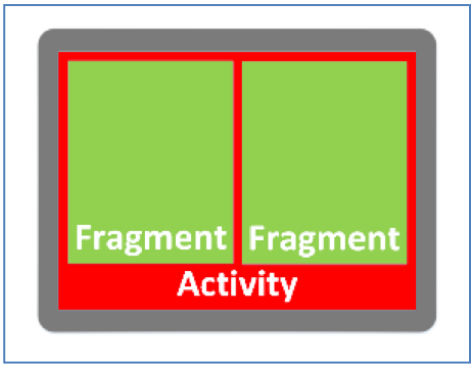

Gráfico 10. 2 fragment por Activity

- SmartTV: en el que caben 4 Fragments por Activity

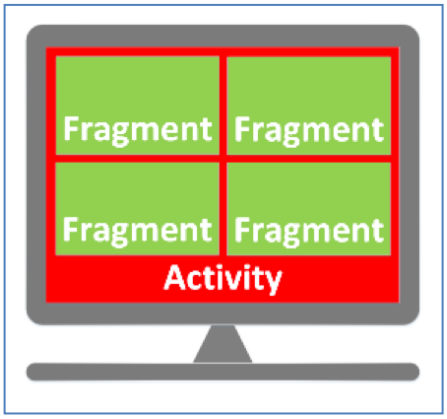

Gráfico 11. 4 fragment por Activity

VENTAJAS DE USO

Utilizar fragments nos aporta las siguientes ventajas:

- Modularidad: Se pueden poner donde queramos
- Reusabilidad: Se pueden reutilizar tantas veces como se necesiten

ACTIVITY CONTENEDORA

Una Activity contenedora es la Activity que contiene uno o más Fragments.

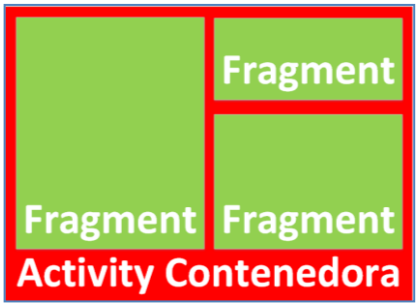

Gráfico 12. Activity Contenedora

En una aplicación puede haber varias Activities contenedoras. Y un mismo Fragment puede estar sobre varias Activities contenedoras.

Un Fragment siempre tiene que estar sobre un Activity, por lo que todo Fragment tendrá al menos una Activity contenedora.

Cada hueco reservado para un Fragment sobre una Activity contenedora podrá contener a la vez un único Fragment. Si es estático, definido en el diseño de la Activity contenedora como <Fragment/>, no se podrá intercambiar por otro Fragment. Si es dinámico, definido en el diseño por algún ViewGroup, como <FrameLayout/>, desde Java podrá sustituirse un Fragment por otro.

CICLO DE VIDA DE LOS FRAGMENTS

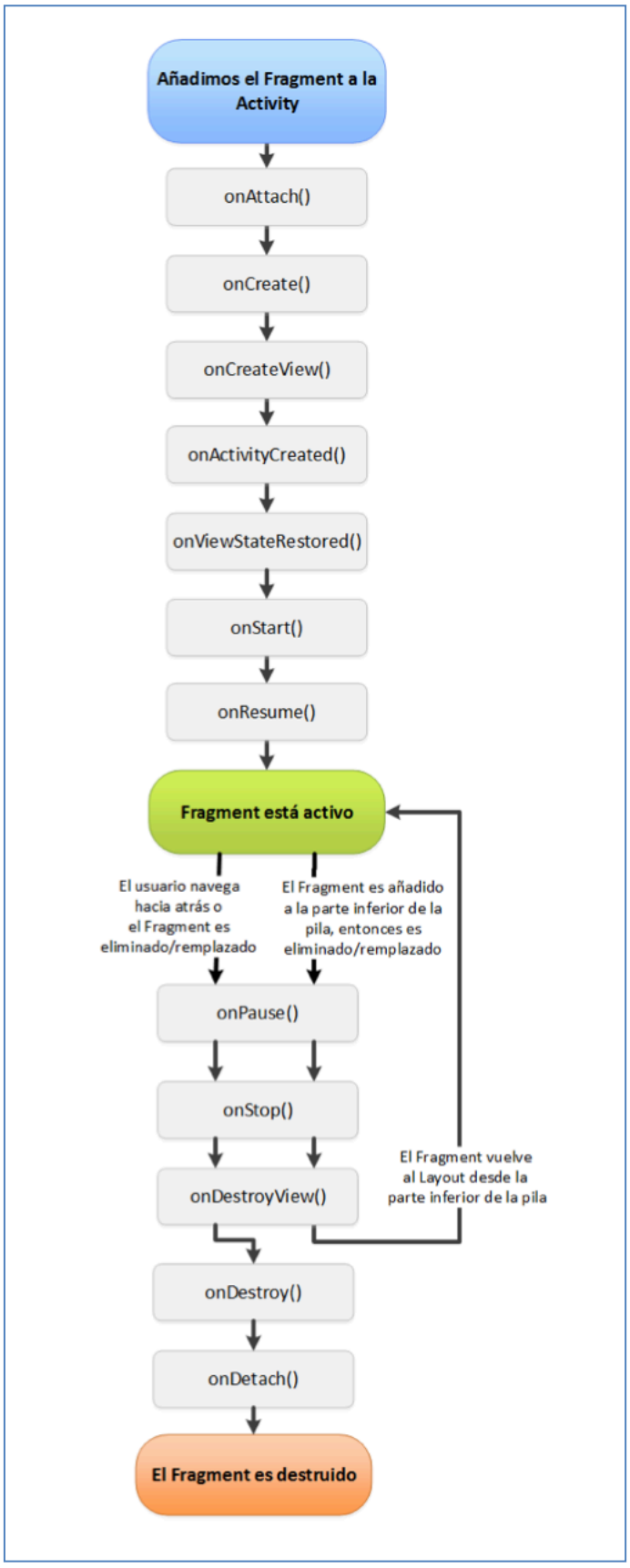

Gráfico 13. Ciclo de vida de los Fragments

onAttach(); Justo después de adjuntar el Fragment a la Activity contenedora

onCreate(); Al crearse el Fragment

onCreateView(); Llamado en el momento que se necesite pintar la interfaz del usuario por primera vez

onActivityCreated(); Justo después de completarse el onCreate de la Activity contenedora

onViewStateRestored(); Cuanto se restaura todo el estado que había sido guardado en la jerarquía de las Views del Fragment

onStart(); Al hacerse visible para el usuario

onResume(); Al comenzar la iteración con el usuario

onPause(); Se llama con la primera indicación de que el usuario abandona el Fragment (no siempre significa que el Fragment esté siendo destruido)

onStop(); Al no ser visible para el usuario

onDestroyView(); Cuando la View que fue previamente creada con onCreateView() sea desajuntada del Fragment

onDestroy(); Cuando ya no se va a utilizar

onDetach(); Justo antes de que el Fragment deje de estar asociado a su activity

Estados del Fragment que coinciden con los de Activity

- Reanudado (Resumed): Fragment visible
- Pausado (Paused): Otra Activity se ha puesto en primer plano y ha sombreado o se ve parcielmente la Activity contenedora del Fragment
- Detenido (Stopped): El Fragment no es visible, pero su estado e información están retenidos por Android. Se puede deber a que la Activity contenedora se haya detenido o que el Fragment se haya eliminado pero añadido a la pila de atrás. El Fragment será matado si lo es la Activity contenedora

Métodos coordinados con el ciclo de vida de la Activity contenedora

- onAttach(): cuando el Fragment se asocia a la Activity contenedora
- onCreateView(): Al crear la jerarquía de Views asociada al Fragment
- onActivityCreated(): al acabar de ejecutarse el método onCreate() de la Activity contenedora
- onDestroyView(): cuando se elimina la jerarquía de Views asociada al Fragment
- onDetach(): en el momento en que el Fragment se desasocia de la Activity contenedora Se podrá eliminar y añadir Fragments libremente cuando nos encontremos en el estado de Reanudado de la Activity contenedora

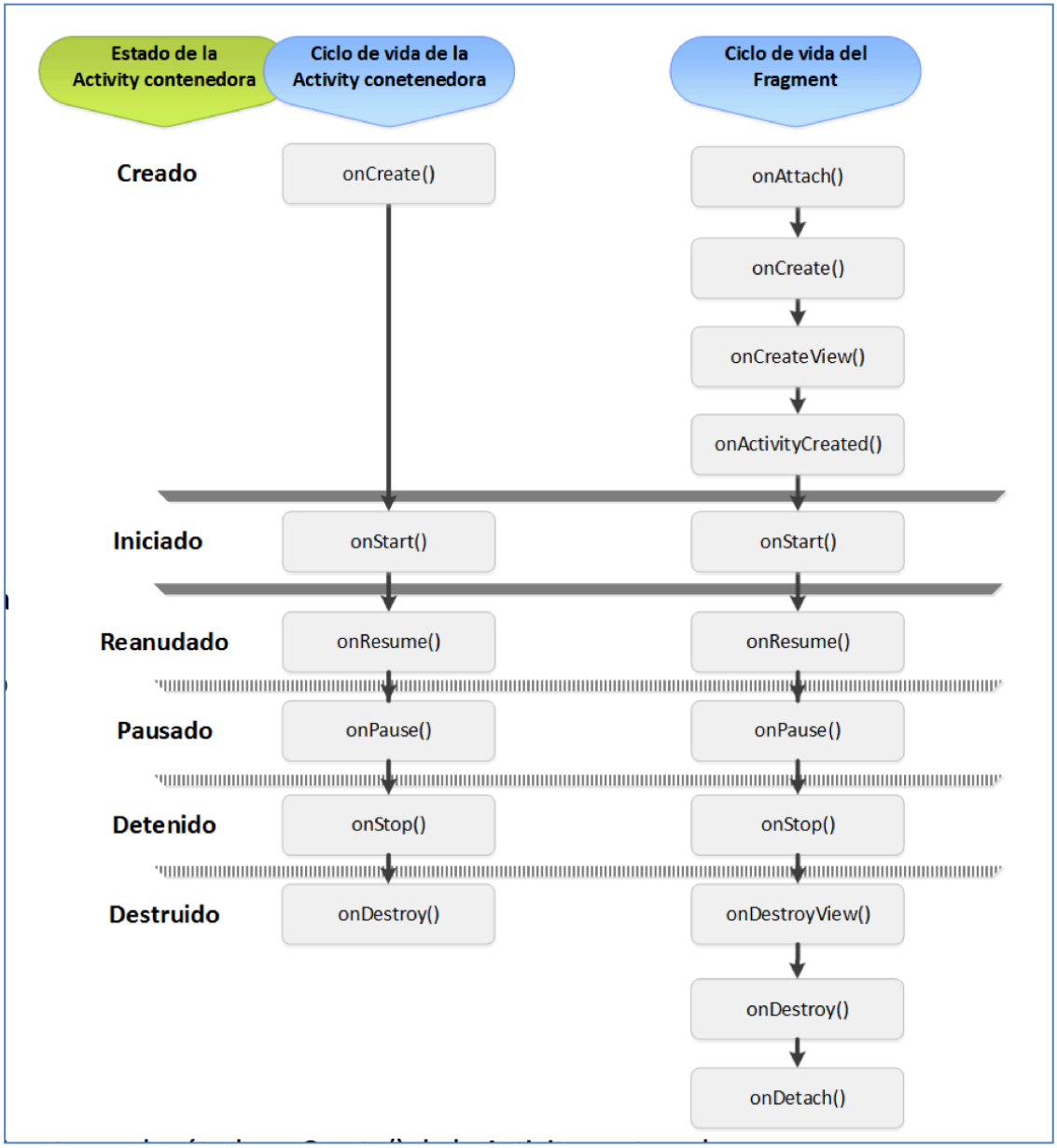

Gráfico 14. Ciclo de vida del Fragment vinculado con el ciclo de vida de la Activity contenedora

CREAR LA CLASE FRAGMENT

```
public class ItemDetailFragment extends Fragment {

    public static final String ARG_ITEM_ID = "item_id";
    private DummyContent.DummyItem mItem;


    public ItemDetailFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (getArguments().containsKey(ARG_ITEM_ID)) {
            mItem = DummyContent.ITEM_MAP.get(getArguments().getString(
                    ARG_ITEM_ID));
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_item_detail,
                container, false);

        if (mItem != null) {
            ((TextView) rootView.findViewById(R.id.item_detail))
                    .setText(mItem.content);
        }

        return rootView;
    }
}
```

Gráfico 15. Creación del Fragment

AGREGAR EL FRAGMENT A LA ACTIVITY CONTENEDORA

```java
public class ItemDetailActivity extends FragmentActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_item_detail);

        getActionBar().setDisplayHomeAsUpEnabled(true);

        if (savedInstanceState == null) {
            Bundle arguments = new Bundle();
            arguments.putString(ItemDetailFragment.ARG_ITEM_ID, getIntent()
                    .getStringExtra(ItemDetailFragment.ARG_ITEM_ID));
            ItemDetailFragment fragment = new ItemDetailFragment();
            fragment.setArguments(arguments);
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.item_detail_container, fragment).commit();
        }
    }
```

Gráfica 16. Agregar el fragment a la activity contenedora

3.- NAVIGATION DRAWER

El objeto Navigation Drawer nos muestra un botón en la barra de navegación que al pulsarlo se despliega el menú con las opciones.

Gráfico 17. Botón Navigation Drawer

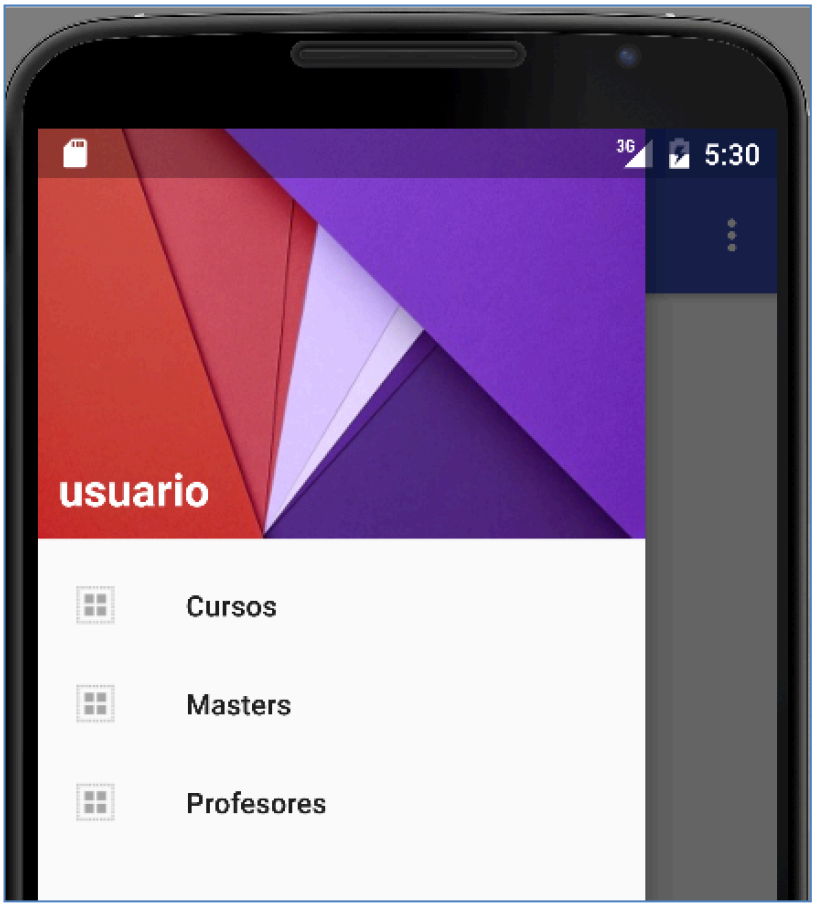

Gráfico 18. Menú de navigation drawer

Veamos como hacerlo.

Nuestro layout principal se define a través de un DrawerLayout, aquí incorporamos el objeto NavigationView el cual contiene el menú con las opciones a mostrar.

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".MainActivity">

    <!-- Contenido de la actividad -->
    <include layout="@layout/content_main" />

    <!-- Navigation View -->
    <android.support.design.widget.NavigationView
        android:id="@+id/navview"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        android:layout_gravity="start"
        app:headerLayout="@layout/header_navview"
        app:menu="@menu/menu_navview" />

</android.support.v4.widget.DrawerLayout>
```

Gráfico 19. Layout principal

En la siguiente imagen vemos como diseñar la cabecera.

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:src="@drawable/navheader"
        android:scaleType="centerCrop" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="usuario"
        android:textAppearance="@style/TextAppearance.AppCompat.Large.Inverse"
        android:textStyle="bold"
        android:layout_gravity="bottom"
        android:layout_marginBottom="10dp"
        android:layout_marginLeft="10dp" />
</FrameLayout>
```

Gráfico 20. Header_navview.xml

Este es el layout con las opciones del menú

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/menu_seccion_1"
            android:icon="@drawable/ic_menu"
            android:title="Cursos"/>
        <item
            android:id="@+id/menu_seccion_2"
            android:icon="@drawable/ic_menu"
            android:title="Masters"/>
        <item
            android:id="@+id/menu_seccion_3"
            android:icon="@drawable/ic_menu"
            android:title="Profesores"/>
    </group>

</menu>
```

Gráfico 21. menú_navview.xml

En el MainActivity creamos los siguiente atributos:

```java
public class MainActivity extends AppCompatActivity {

    private DrawerLayout drawerLayout;
    private NavigationView navView;
```

Gráfico 22. MainActivity

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.appbar);
    setSupportActionBar(toolbar);

    getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_nav_menu);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);

    drawerLayout = (DrawerLayout)findViewById(R.id.drawer_layout);
    navView = (NavigationView)findViewById(R.id.navview);
    navView.setNavigationItemSelectedListener(
            (menuItem) -> {

                    boolean fragmentTransaction = false;
                    Fragment fragment = null;

                    switch (menuItem.getItemId()) {
                        case R.id.menu_seccion_1:
                            fragment = new CursosFragment();
                            fragmentTransaction = true;
                            break;
                        case R.id.menu_seccion_2:
                            fragment = new MasterFragment();
                            fragmentTransaction = true;
                            break;
                        case R.id.menu_seccion_3:
                            fragment = new ProfesoresFragment();
                            fragmentTransaction = true;
                            break;
                    }

                    if(fragmentTransaction) {
                        getSupportFragmentManager().beginTransaction()
                                .replace(R.id.content_frame, fragment)
                                .commit();

                        menuItem.setChecked(true);
                        getSupportActionBar().setTitle(menuItem.getTitle());
                    }

                    drawerLayout.closeDrawers();

                    return true;
            });
```

Gráfico 23. Método onCreate()

En el método onCreate() que vemos en la imagen anterior se captura el evento para que al pulsar sobre las opciones del menú se muestre el fragment correspondiente.

ÍNDICE DE GRÁFICOS