Laboratorio 6.1

Apellidos: Moreno Vera
Nombres: Felipe Adrian
Código: 20120354I
Asignatura: Programación en Dispositivos Móviles (CC481)

2016 - I

Indice

Actividad 1

1. Creamos un proyecto nuevo.
2. Añadimos un tema en el fichero res/values/styles.xml.¿Que realiza este fichero?
Realiza cambios de los estilos definidos, o puedes crear estilos según te parezca.
<resources>
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
</style>
</resources>
Basta comentar el resto . . .

```
<style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
  <item name="colorPrimary">@color/colorPrimary</item>
  <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
  <item name="colorAccent">@color/colorAccent</item>
</style>
```

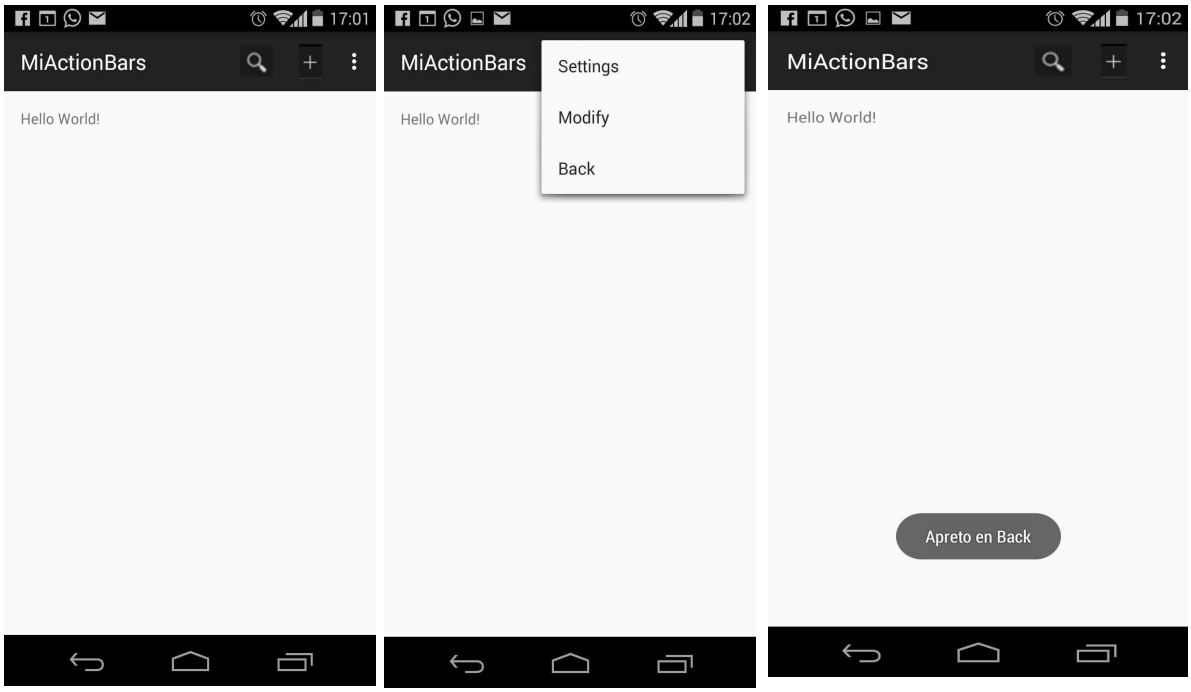
Donde en el archivo colors.xml se cambio los colores.

```
<color name="colorPrimaryDark">#303F9F</color>
<color name="colorPrimary">#1B1B1B</color>
<color name="colorAccent">#FF4081</color>
```

3. Con lo visto en la clase de teoría (punto 1.3) incluye el código y describa la salida.
Insertamos el código en menu.xml, añadimos también las imagenes de la carpeta drawable
Aparecen en el Action Bar los menus con las imagenes puestas.
El color de actionbar es plomo muy oscuro.

4. Ponga 3 opciones al menú overflow y que lance un Toast para cada una.
5. Ahora inserta los recursos (punto 1.4 de teoría) y vea la salida
Cambia los colores de salida(Es más oscura)

La vista esta al lado ...



echado, se muestra el mensaje “new” que añadimos.

Actividad 2

1. Primero modificamos el XML de MainActivity

1. Borramos el Relative Layout y vamos a crear un Linear Layout horizontal con un weightSum de 1.

2. Vamos a crear dos Fragment uno con id izquierdo que estará a la izquierda y el otro con id derecho que estará a la derecha. Los dos tendrán la propiedad layout_height a match_parent y layout_width a “0dp”. Este “0dp” lo que hace es priorizar el peso y no el ancho.

3. Posteriormente tendremos que indicar el peso de los Fragments. Para ello le agregamos la propiedad layout_weight a 0.5 en cada Fragment, que hará que cada Fragment tenga el 50% de la pantalla.

4. Si vamos a la parte de diseño veremos como hemos creado nuestro contenedor.

2. Vamos a crear los Fragments:

1. Creamos los dos fichero XML en la carpeta /res/layout los fragmentos izquierda.xml y derecha.xml.

2. En el XML de izquierda será un LinearLayout igual que vimos en teoría

1. Creamos un TextView igual que el de teoría, pero:

1. Con la propiedad android:layout_height="match_parent".

2. Crea una propiedad String con el texto a mostrar y agrégale algunas propiedades (especifica cuáles).

```
<EditText
  android:layout_width="wrap_content"
  android:layout_height="match_parent"
  android:hint="@string/str_left"
  android:id="@+id/editTextLeft"
  android:textAlignment="center"
  android:textColor="#b1330731"
  android:textSize="12dp"
  android:textStyle="bold" />
```

Le dimos color de texto, estilo blod, alineamiento y tamaño.

2. Crea un Button con un texto.

3. En el XML de derecha será un LinearLayout igual que vimos en teoría

3. Trabajamos el código Java de los Fragments:

1. Creamos el Java del Fragment de la derecha con nombre “Derecha.java”

1. Vamos a crear nuestra variable View ya que es la vista que

vamos a estar manejando. También creamos el TextView.
View rootView;
TextView txt;
2. Creamos el método onCreateView como se ha visto en clase de teoría, aunque todavía no lo implementamos
3. Colocamos la vista a utilizar, explica este aspecto
rootView= inflater.inflate (R.layout.derecha, container, false);
txt = (TextView) rootView.findViewById(R.id.txt);
return rootView;

2. Creamos "Izquierda.java".

1. Utilizamos la Vista, el Button y EditText.
View rootView;
Button boton
EditText campo;
2. Creamos el método onCreateView con la siguiente implementación.
rootView= inflater.inflate (R.layout.izquierda, container, false);
campo = (EditText) rootView.findViewById(R.id.campotxt);
boton= (Button) rootView.findViewById(R.id.boton);
return rootView;

4. A continuación vamos a crear una interfaz que es la clave de comunicación entre Fragments.

1. Creamos una interface Java con el nombre "enviarMensaje"
2. Creamos un método "enviarDatos" con un parámetro tipo String "mensaje".

5. Por último vamos a añadir la comunicación entre fragments. Para ello se va a especificar qué se va a cambiar en el mensaje de la derecha (nuestra clase derecha).

1. Creamos el método "obtenerDatos" en la clase Java "Derecha" que recibirá un String llamado "mensaje".
2. Lo único que hará nuestro método será cambiar el texto de nuestro TextView. Por tanto: txt.setText(mensaje);

6. El botón será la que dispare el cambio del texto. Dicho botón se encuentra definido en la clase izquierda.

1. Como variable de la clase (debajo de la declaración de nuestra variable "campo"), creamos la variable de nuestra interfaz: EnviarMensaje EM;
2. A continuación implementamos el evento onClick.
3. En dicho evento tendremos un "mensaje" tipo String que tendrá el valor de texto de la variable "campo" y el envío de del mensaje a través de nuestra interfaz:
String mensaje;
mensaje = campo.getText().toString();
EM.enviarDatos(mensaje);

4. Creamos el método onAttach() que permite ligar el contenido con el Activity cuando hablamos de Fragments. Explique dicho código

```
Public void onAttach(Context context){
        super.onAttach(activity);
        try{
                EM = (EnviarMensaje) activity;
        } catch (ClassCastException e) {
                throw new ClassCastException("Necesitas implementar mensaje");
        }
}
```

attach, llama al metodo, el cual el sistema lanza cuando llamamos al fragment de la actividad, instanciando una instancia de enviarMensaje

7. Ya que la actividad principal es la que implementará el Fragment vamos a modificarla.

1. Implementar la interfaz que hemos realizado.

2. Implementamos el método "enviarDatos". Tendrá el siguiente código, explíquelo:

```
public void enviarDatos (String mensaje){
        Derecha derecha = (Derecha) getSupportFragmentManager().
        findFragmentById(R.id.derecha);
        derecha.ObtenerDatos(mensaje);
}
```

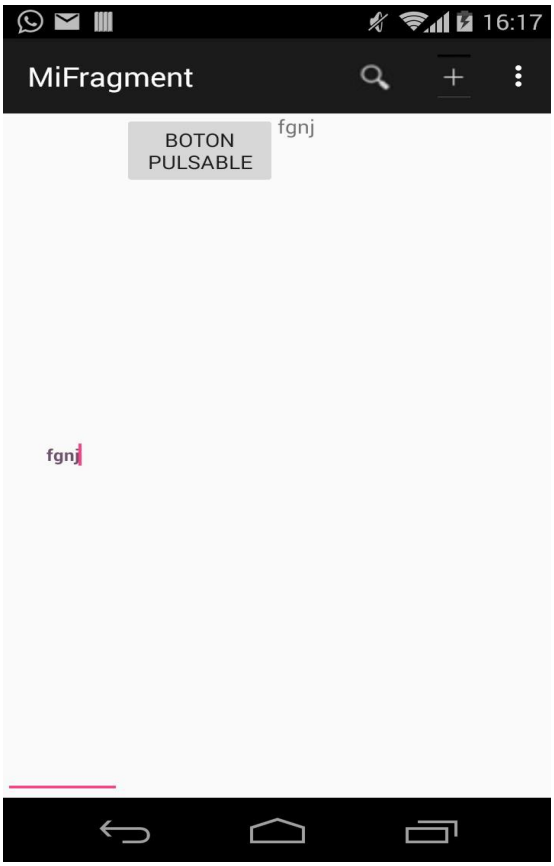
instancia la clase Derecha, y la convierte en un Fragment jalando el fragment del layout, luego ejecuta el metédo de enviar datos

8. A continuación deberemos indicar la clase en el fichero MainActivity. Es decir en cada etiqueta "fragment" deberemos establecer la propiedad "class". Guíese de la teoría.

```
<fragment class="com.example.jbot.mifragment.Izquierda"
  android:id="@+id/izquierdo"
  android:layout_weight="0.5"
  android:layout_width="0dp"
  android:layout_height="match_parent" />
<fragment class="com.example.jbot.mifragment.Derecha"
  android:id="@+id/derecho"
  android:layout_weight="0.5"
  android:layout_width="0dp"
  android:layout_height="match_parent" />
```

9. Por último deberemos declarar en el Manifest nuestros Fragments. Guíese de la teoría. Esta parte no es necesaria.

Solución:



Actividad 3

1. Vamos a comenzar creando un spinner como page filter en nuestro ActionBar.

1. Para ello creamos nuestro fichero /res/layout/toolbar.xml con la siguiente información que posteriormente incluiremos en nuestro layout principal

```
<android.support.v7.widget.Toolbar
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_height="?attr/actionBarSize"
        android:layout_width="match_parent"
        android:minHeight="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light" >
        <Spinner android:id="@+id/CmbToolbar"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </android.support.v7.widget.Toolbar>
```

2. Vamos a personalizar también nuestro layout. Lo primero será crear nuestro fichero /res/layout/appbar_filter_title.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
        style="@style/TextAppearance.AppCompat.Widget.ActionBar.Title"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

3. Ahora modificaremos la visualización de la lista despeglable. Creamos el fichero /appbar_filter_list.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
        style="?android:attr/spinnerDropDownItemStyle"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="@color/background_material_light"
        android:textColor="@color/primary_text_default_material_light" />
```

4. Como ya vimos en las colecciones de datos para utilizar el spinner se tiene que crear un Adapter en el método onCreate.

```
//Appbar
Toolbar toolbar = (Toolbar) findViewById(R.id.appbar);
setSupportActionBar(toolbar);
getSupportActionBar().setDisplayShowTitleEnabled(false);
//Appbar page filter
Spinner cmbToolbar = (Spinner) findViewById(R.id.CmbToolbar);
ArrayAdapter<String> adapter = new ArrayAdapter<>(
```

```
getSupportActionBar().getThemedContext(),
R.layout.appbar_filter_title,
new String[]{"Opción 1 ", "Opción 2 ", "Opción 3 "});
adapter.setDropDownViewResource(R.layout.appbar_filter_list);
cmbToolbar.setAdapter(adapter);
cmbToolbar.setOnItemSelectedListener(
new AdapterView.OnItemSelectedListener() {
@Override
public void onItemSelected(AdapterView<?> adapterView, View view,
int i, long l) {
//... Acciones al seleccionar una opción de la lista
Log.i("Toolbar 3", "Seleccionada opción " + i);
}
@Override
public void onNothingSelected(AdapterView<?> adapterView) {
//... Acciones al no existir ningún elemento seleccionado
}
});
```

5. Relacione los estilos definidos anteriormente con este último punto.

6. Compilamos la aplicación y verificamos lo que nos ha proporcionado dar los estilos.

2. Pasemos a ver ahora las tabs.

1. Lo primero es utilizar la librería de diseño de Google (Design Support Library 1) ya que vamos a utilizar TabLayout.

```
dependencies {
//...
compile 'com.android.support:design:22.2.0'
}
```

2. Una vez realizado escribimos en nuestro layout principal el código XML. De este código, ¿que significa la etiqueta include? Explique también que significa TabLayout y ViewPager. (se añade a content_main)

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".MainActivity">
        <include layout="@layout/toolbar"
                android:id="@+id/appbar" />
        <android.support.design.widget.TabLayout
                android:id="@+id/appbartabs"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" />
        <FrameLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:foreground="@drawable/header_shadow">
        <android.support.v4.view.ViewPager
                android:id="@+id/viewpager"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="@android:color/white" />
        </FrameLayout>
</LinearLayout>
```

TabLayout es un Layout horizontal para display tabs.

ViewPager es un gestionador de ciclo de vida de cada pagina abierta en un fragment

La etiqueta significa que de un fichero externo, trae información o características

3. Vamos a nuestro código Java veremos como añadir pestañas y contenido a cada Tab. En el ejemplo propuesto solamente se han creado dos Fragments pero tenga en cuenta que cada Tab tendría su propio contenido Fragment. Explique la clase padre FragmentPager y el código de nuestra nueva clase.

```
public class MiFragmentPagerAdapter extends FragmentPagerAdapter {
        final int PAGE_COUNT = 6;
        private String tabTitles[] =
        new String[] { "Tab Uno", "Tab Dos", "Tab Tres", "Tab Cuatro", "Tab
        Cinco", "Tab Seis"};
        public MiFragmentPagerAdapter(FragmentManager fm) {
        super(fm);
        }
        @Override
        public int getCount() {
        return PAGE_COUNT;
        }
        @Override
        public Fragment getItem(int position) {
        Fragment f = null;
        switch(position) {
        case 0:
        case 2:
        case 4:
        f = Fragment1.newInstance();
        break;
        case 1:
        case 3:
        case 5:
        f = Fragment2.newInstance();
        break;
}
return f;
}
}
@Override
public CharSequence getPageTitle(int position) {
        // Generate title based on item position
        return tabTitles[position];
}
}
```

4. Como se ha indicado deberemos de crear dos Fragment por tanto

1. El código Java del primero

```
public class Fragment1 extends Fragment {
public static Fragment1 newInstance() {
Fragment1 fragment = new Fragment1();
return fragment;
}
public Fragment1() {
// Required empty public constructor
}
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
return inflater.inflate(R.layout.fragment_1, container, false);
}
}
```

1. El código Java del segundo

```
public class Fragment2 extends Fragment {
        public static Fragment2 newInstance() {
        Fragment2 fragment = new Fragment2();
        return fragment;
        }
        public Fragment2() {
        // Required empty public constructor
        }
        @Override
        public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        }@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
return inflater.inflate(R.layout.fragment_2, container, false);
}
}
```

2. Su códgio XML.

fragment_1.xml

```
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="16dp"
        tools:context="net.sgoliver.android.toolbar.Fragment1">
        <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="@string/fragment1" />
</FrameLayout>
```

fragment_2.xml

```
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="16dp"
        tools:context="net.sgoliver.android.toolbar.Fragment2">
        <TextView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="@string/fragment2" />
</FrameLayout>
```

3. Como ya hemos visto deberemos hay que asociar el adaptador al componente viewpager. Para ello añadimos el siguiente código en el método onCreate de la actividad principal. Explique su funcionalidad.

```
Toolbar toolbar = (Toolbar) findViewById(R.id.appbar);
setSupportActionBar(toolbar);
ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
viewPager.setAdapter(new MiFragmentPagerAdapter(
getSupportFragmentManager()));
```

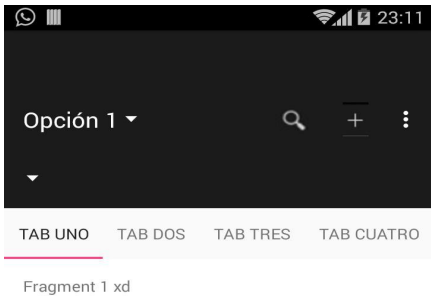
Funciona de tal manera que gestiona el desplazamiento de los Fragment.

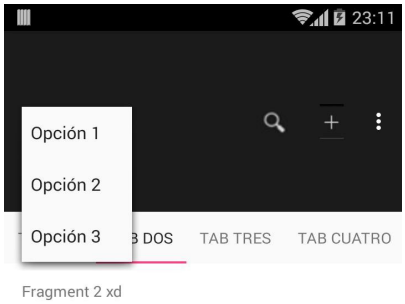
4. Añadimos justo debajo de lo anterior el siguiente código que deberá de explicar.Cambie el valor de MODE_SCROLLABLE por MODE_FIXED y explique su salida.

TabLayout tabLayout = (TabLayout) findViewById(R.id.appbartabs);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
tabLayout.setupWithViewPager(viewPager);

Lo que hizo fue añadir tabs, como si fueran múltiples pestañas eligiendo un fragment por cada tab.

5. Ejecute y vea que las Tabs no tienen el mismo estilo que el ActionBar.





Vamos a cambiar este aspecto incluyendo Toolbar y TabLayout en el elemento AppBarLayout otro componente de la librería de diseño de nuestro layout de la actividad principal.
NOTA: las propiedad de LinarLayout y la etiqueta FrameLayout se quedarán como vimos. Lo demás sustitúyelo. (se añade a content_main)

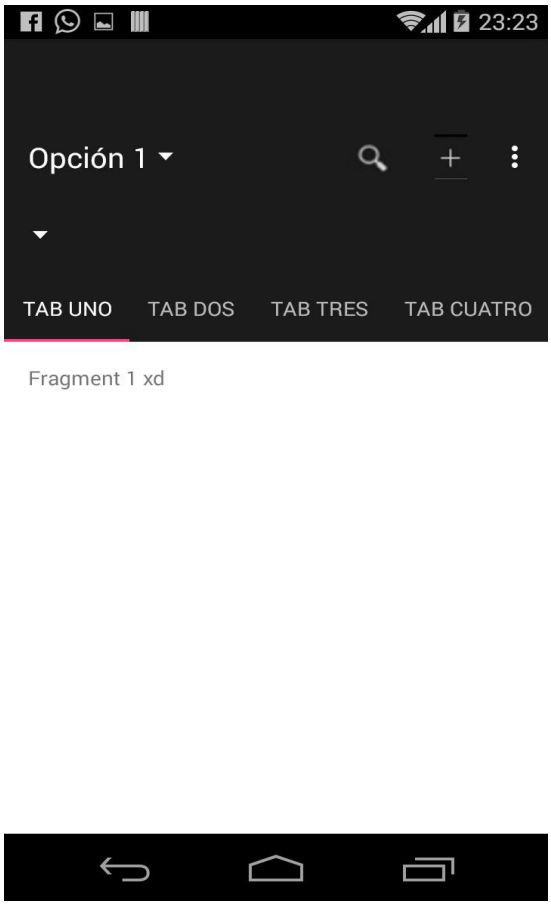
```
<android.support.design.widget.AppBarLayout
        android:id="@+id/appbarlayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" >
        <include layout="@layout/toolbar"
                android:id="@+id/appbar" />
```

```
<android.support.design.widget.TabLayout
        android:id="@+id/appbartabs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</android.support.design.widget.AppBarLayout>
```

3. Ejecute la aplicación y vea el resultado.

Solución final:

Vemos que ya están del mismo style.



Actividad 4

1. Creamos un proyecto nuevo
2. En nuestro Layout principal.

1. Tendremos que definir nuestro DrawerLayout con su propiedad fitsSystemWindows=”true” que evita superposiciones.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:id="@+id/navigation_drawer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="@bool/fitsSystemWindows">
<-- código puesto en los siguientes puntos -->
</android.support.v4.widget.DrawerLayout>
```

2. Dentro del DrawerLayout primeramente definiremos el LinearLayout.

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
<FrameLayout
        android:layout_width="match_parent"
        android:layout_height="@dimen/status_bar_height"
        android:background="?colorPrimary"/>
</LinearLayout>
```

3. Un FrameLayout. En este caso explique las diferentes opciones.

```
<FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/status_bar_height">
<android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?actionBarSize"
        android:background="?attr/colorPrimary"
        android:elevation="4dp"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:theme="@style/ToolbarTheme" />
<TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
```

```
        android:text="Inbox"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        android:textColor="@color/md_text" />
</FrameLayout>
```

4. Por último definiremos el más importante, el NavegationView.
Recuerde estas opciones vistas en teoría.

```
<android.support.design.widget.NavigationView
        android:id="@+id/navigation_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="@bool/fitsSystemWindows"
        app:headerLayout="@layout/navigation_header"
        app:menu="@menu/navigation_menu"
        app:theme="@style/NavigationViewTheme" />
```

3. Si queremos personalizar nuestro NavetationView (parte superior) con unos colores que nos guste y textos deberemos de crear un fichero XML nuevo. Podremos llamarlo /res/layout/navegation_header (especificado en la propiedad headerLayout en el XML principal).
Explique las diferentes propiedades y recuerde establecer las dos imágenes dadas para la práctica.

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="192dp"
android:gravity="bottom"
android:theme="@style/ThemeOverlay.AppCompat.Dark">
<ImageView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:scaleType="centerCrop"
android:background="@drawable/header" />
<ImageView
android:layout_marginLeft="16dp"
android:layout_marginTop="@dimen/navigation_margin"
android:layout_width="64dp"
android:layout_height="64dp"
android:scaleType="centerCrop"
android:src="@drawable/logo" />
<LinearLayout
android:layout_width="match_parent"
android:layout_height="52dp"
android:layout_gravity="left|bottom"
android:gravity="center"
```

```
android:orientation="vertical"
android:paddingBottom="8dp"
android:paddingLeft="16dp"
android:paddingRight="16dp">
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:gravity="center_vertical"
android:text="Programación en Dispositivos Móviles"
android:textAppearance="@style/TextAppearance.AppCompat.Body2"/>
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:gravity="center_vertical"
android:text="Navegation Drawer"
android:textAppearance="@style/TextAppearance.AppCompat.Body1"/>
</LinearLayout>
</FrameLayout>
```

4. Llegados a este punto deberemos especificar las diferentes opciones de nuestro Navigation. Para ello creamos nuestro fichero /app/menu/navigation_menu.xml. Antes de verlo aquí es donde entra Android Design Support Library ya que nos simplifica cada información de los items. Explique sus componentes

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<group android:checkableBehavior="single">
<item
android:checked="true"
android:id="@+id/item_navigation_inbox"
android:icon="@drawable/ic_inbox_black_24dp"
android:title="Inbox" />
<item
android:id="@+id/item_navigation_starred"
android:icon="@drawable/ic_action_toggle_star"
android:title="Starred" />
<item
android:id="@+id/item_navigation_sent_mail"
android:icon="@drawable/ic_action_content_send"
android:title="Sent mail" />
<item
android:id="@+id/item_navigation_drafts"
android:icon="@drawable/ic_action_content_drafts"
android:title="Drafts" />
</group>
```

```
<item android:title="Subheader">
<menu>
<item
android:id="@+id/item_navigation_settings"
android:icon="@drawable/ic_action_content_mail"
android:title="Settings" />
<item
android:id="@+id/item_navigation_help_and_feedback"
android:icon="@drawable/ic_action_action_delete"
android:title="Help and feedback" />
</menu>
</item>
</menu>
```

5. En cuanto al fichero styles.xml, attrs.xml y colors.xml descárguelo de la plataforma y sustitúyalo en el proyecto por el original. Explique su contenido.

6. En nuestro Java de actividad principal.

1. Declaramos fuera de onCreate los elementos necesarios.

```
DrawerLayout drawerLayout;
Toolbar toolbar;
ActionBar actionBar;
TextView textView;
```

2. En nuestro onCreate añadiremos el icono de tres barras a nuestro ActionBar y daremos soporte al mismo.

```
toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
actionBar = getSupportActionBar();
actionBar.setHomeAsUpIndicator(R.drawable.ic_menu_white_24dp);
actionBar.setDisplayHomeAsUpEnabled(true);
```

3. Finalmente en nuestro onCreate indicamos el código que hace funcionar a nuestro NavegationView. Con la finalidad de estructurar nuestro código creamos el método setupNavegation, que pasa la instancia NavetationView.

```
drawerLayout = (DrawerLayout)
findViewById(R.id.navigation_layout);
NavigationView navigationView = (NavigationView)
findViewById(R.id.navigation_view);
if (navigationView != null) {
setupNavigation(navigationView);
}
setupNavigation(navigationView);
```

4. El método setupNavegation que contendrá todas nuestras opciones. Explique que realiza cada opción antes de ejecutar la aplicación.

```
private void setupNavigation(NavigationView navigationView) {
navigationView.setNavigationItemSelectedListener(
new NavigationView.OnNavigationItemSelectedListener() {
@Override
public boolean onNavigationItemSelected(MenuItem menuItem){
textView = (TextView) findViewById(R.id.textView);
switch (menuItem.getItemId()) {
case R.id.item_navigation_inbox:
menuItem.setChecked(true);
textView.setText(menuItem.getTitle());
drawerLayout.closeDrawer(GravityCompat.START);
return true;
case R.id.item_navigation_starred:
menuItem.setChecked(true);
textView.setText(menuItem.getTitle());
drawerLayout.closeDrawer(GravityCompat.START);
return true;
case R.id.item_navigation_sent_mail:
menuItem.setChecked(true);
textView.setText(menuItem.getTitle());
drawerLayout.closeDrawer(GravityCompat.START);
return true;
case R.id.item_navigation_drafts:
menuItem.setChecked(true);
textView.setText(menuItem.getTitle());
drawerLayout.closeDrawer(GravityCompat.START);
return true;
case R.id.item_navigation_settings:
menuItem.setChecked(true);
textView.setText(menuItem.getTitle());
Toast.makeText(MainActivity.this, "Launching " +
menuItem.getTitle().toString(),
Toast.LENGTH_SHORT).show();
drawerLayout.closeDrawer(GravityCompat.START);
Intent intent = new Intent(
MainActivity.this, SettingsActivity.class);
startActivity(intent);
return true;
case R.id.item_navigation_help_and_feedback:
menuItem.setChecked(true);
Toast.makeText(MainActivity.this,
menuItem.getTitle().toString(),
```

```
Toast.LENGTH_SHORT).show();
drawerLayout.closeDrawer(GravityCompat.START);
return true;
}
return true
}
});
}
```

5. Por último, establecemos en nuestro Toolbar que si pulsa las tres barras se abra nuestro componente.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case android.R.id.home:
drawerLayout.openDrawer(GravityCompat.START);
return true;
}
return super.onOptionsItemSelected(item);
}
```

7. Como vimos anteriormente se llamaba a otra Activity por lo que tendremos que crearla.

1. En onCreate deberemos de establecer el ActionBar.

```
toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
TypedValue typedValueColorPrimaryDark = new TypedValue();
SettingsActivity.this.getTheme().resolveAttribute(R.attr.colorPrimaryDark,
typedValueColorPrimaryDark, true);
```

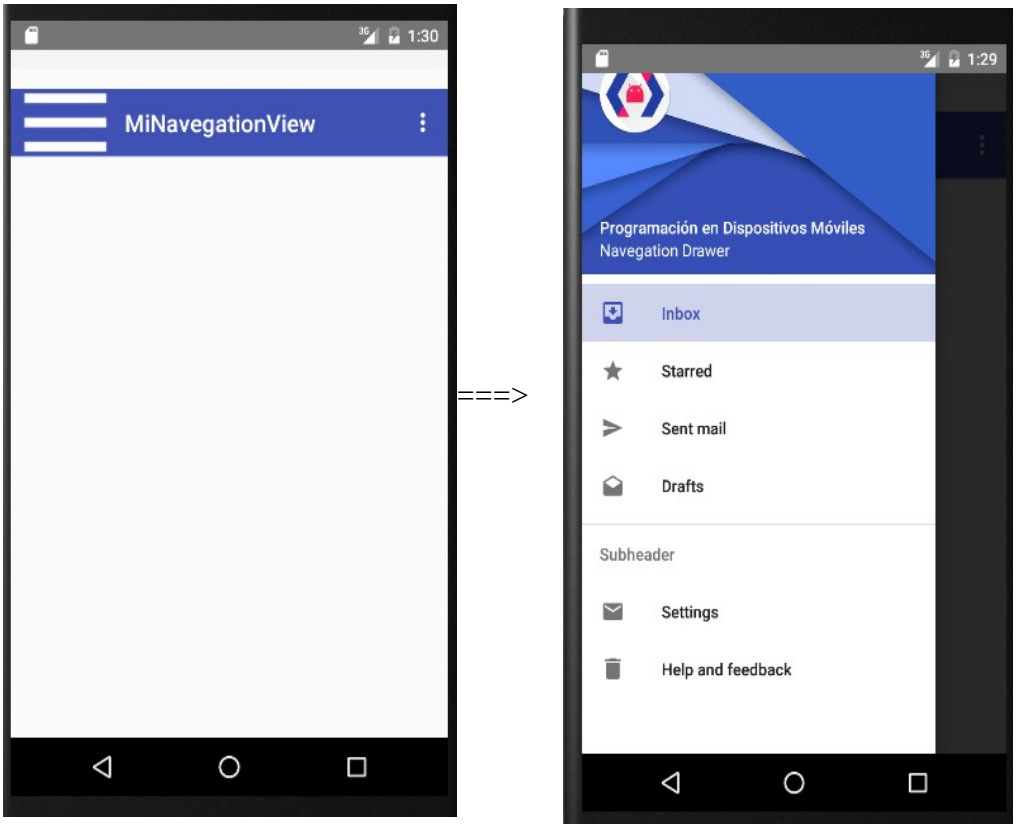
2. En el XML el FrameLayout correspondiente.

```
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
<android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?actionBarSize"
        android:background="?attr/colorPrimary"
        android:elevation="4dp"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:theme="@style/ToolbarTheme" />
<TextView
```

```
            android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Settings"
            android:textColor="@color/md_text" />
</FrameLayout>
```

Solución:

Córrelo desde la máquina virtual, lo hice desde mi celular, pero los 24p eran demasiado grande. :D



Link del github con los códigos del laboratorio:

https://github.com/Jenazad/PDM/tree/master/Laboratorio_6

Referencias

http://stackoverflow.com/questions/15492791/how-do-i-show-overflow-menu-items-to-action-bar-in-android
http://developer.android.com/intl/es/guide/components/fragments.html
https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html
http://developer.android.com/intl/es/guide/components/fragments.html
http://developer.android.com/intl/es/reference/android/support/design/widget/TabLayout.html
http://developer.android.com/intl/es/reference/android/support/v4/view/ViewPager.html