UTILIZACIÓN DE FRAGMENTS EN ANDROID

MariLuz Sevilla Aparicio

Contenido

1. Introducción

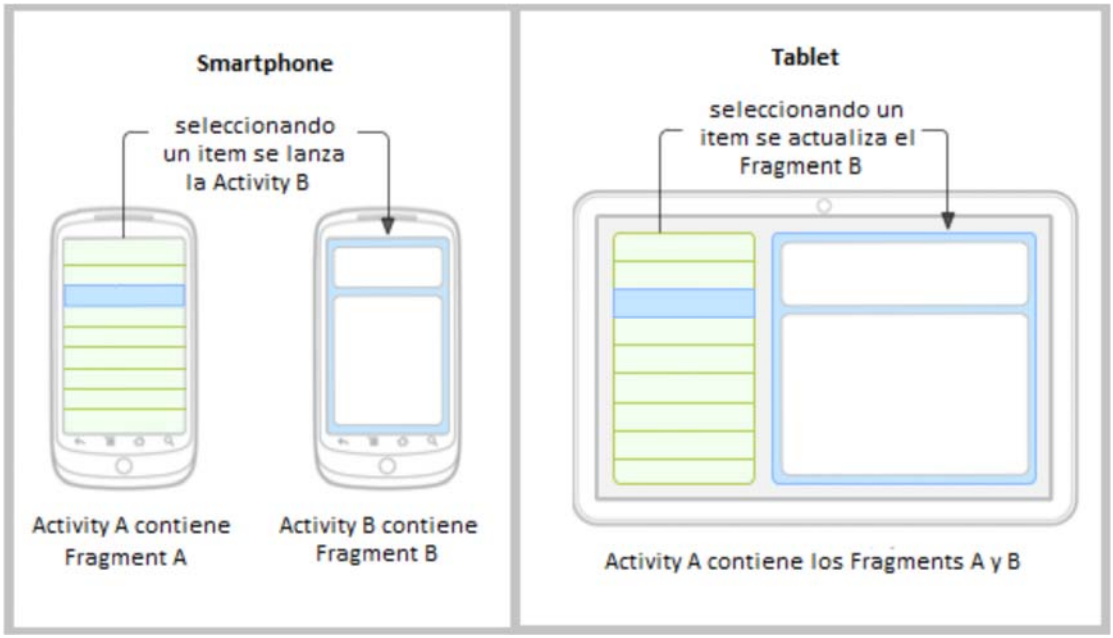
Cuando empezaron a aparecer dispositivos de gran tamaño tipo Tablet, el equipo de Android tuvo que solucionar el problema de la adaptación de la interfaz gráfica de las aplicaciones a ese nuevo tipo de pantallas. Una interfaz de usuario diseñada para un teléfono móvil no se adaptaba fácilmente a una pantalla 4 o 5 pulgadas mayores. La solución a esto vino en forma de un nuevo tipo de componente llamado Fragment.

Android introduce fragmentos en Android 3.0 (nivel de API 11), principalmente para apoyar diseños de interfaz de usuario más dinámicas y flexibles en las pantallas grandes, como las Tablet.

Por ejemplo, en un teléfono, puede que sea necesario separar los fragmentos para proporcionar una interfaz de usuario de un solo panel, cuando más de uno no puede caber dentro de la misma actividad.

Un fragment funciona como una Activity, ya que contienen layouts al igual que las Activitys, pero la diferencia está en que los fragments están dentro de una Activity.

En la imagen vemos la diferencia entre usar fragments en un Smartphone o en una Tablet.



2. Definición de Fragment

Un fragmento es una pieza de software en la que definimos un diseño visual y un comportamiento determinado, y que además, tiene la ventaja de que puede trabajar con otros fragments en un mismo Activity.

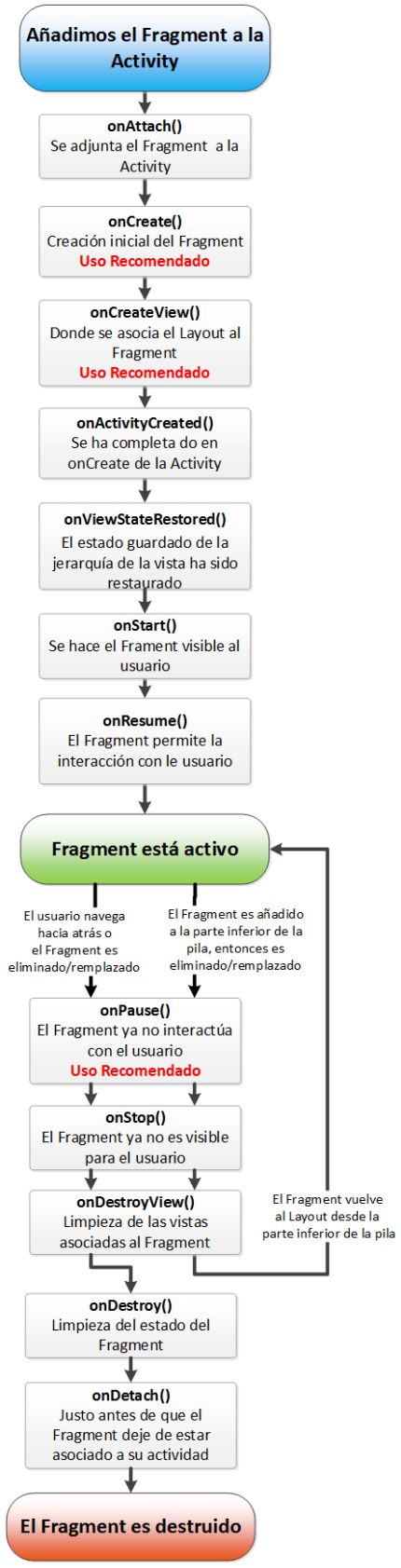
Un fragment podría definirse como una porción de la interfaz de usuario que puede añadirse o eliminarse de una interfaz de forma independiente al resto de elementos de la actividad, y que por supuesto puede reutilizarse en otras actividades. Esto, aunque en principio puede parecer algo trivial, nos va a permitir poder dividir nuestra interfaz en varias porciones de forma que podamos diseñar diversas configuraciones de pantalla, dependiendo de su tamaño y orientación, sin tener que duplicar código en ningún momento, sino tan sólo utilizando o no los distintos fragmentos para cada una de las posibles configuraciones.

3. Características de los Fragments

- Un Fragment tiene su propio Layout y su propio ciclo de vida.
- Las clases java asociadas a los fragment heredan o tienen como superclase a Fragment (extends Fragment).
- Un fragment necesita de una Activity para que esta actúe de contenedor y para intercambiar datos con otros fragment. Esta Activity tiene como superclase a FragmentActivity (extends Fragment Activity).
- Cada uno de los fragment que se declaren constará de su propio código y actuará de forma independiente al resto.

4. Ciclo de vida de un Fragment

En la siguiente imagen se detallan los métodos del ciclo de vida de un fragment.


Añadimos el Fragment a la Activity
onAttach()
Se adjunta el Fragment a la Activity
onCreate()
Creación inicial del Fragment
Uso Recomendado
onCreateView()
Donde se asocia el Layout al Fragment
Uso Recomendado
onActivityCreated()
Se ha completa do en onCreate de la Activity
onViewStateRestored()
El estado guardado de la jerarquía de la vista ha sido restaurado
onStart()
Se hace el Frament visible al usuario
onResume()
El Fragment permite la interacción con le usuario
Fragment está activo
El usuario navega hacia atrás o el Fragment es eliminado/remplazado
El Fragment es añadido a la parte inferior de la pila, entonces es eliminado/remplazado
onPause()
El Fragment ya no interactúa con el usuario
Uso Recomendado
onStop()
El Fragment ya no es visible para el usuario
onDestroyView()
Limpieza de las vistas asociadas al Fragment
El Fragment vuelve al Layout desde la parte inferior de la pila
onDestroy()
Limpieza del estado del Fragment
onDetach()
Justo antes de que el Fragment deje de estar asociado a su actividad
El Fragment es destruido

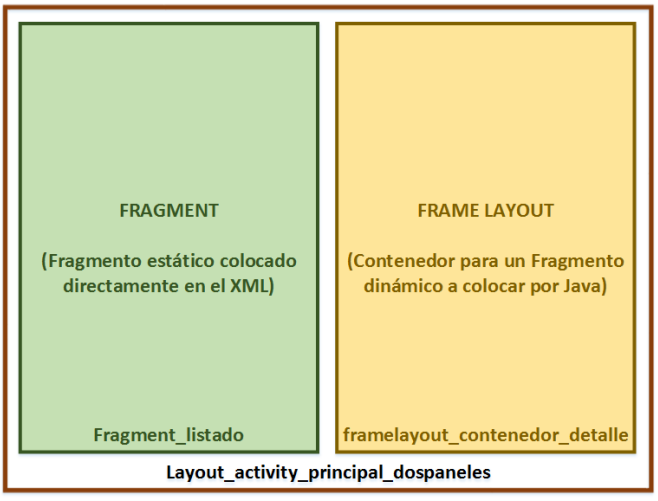
Generalmente a la hora de crear un fragmento se deben implementar al menos los siguientes métodos:

- onCreate() - El sistema llama a este método a la hora de crear el fragmento, normalmente iniciaremos los componentes esenciales del fragmento.
- onCreateView() - El sistema llamara al método cuando sea la hora de crear la interface de usuario o vista, normalmente se devuelve la view del fragmento.
- onPause() - El sistema llamara a este método en el momento que el usuario abandone el fragmento, por lo tanto es un buen momento para guardar información.

5. Tipos de Fragments

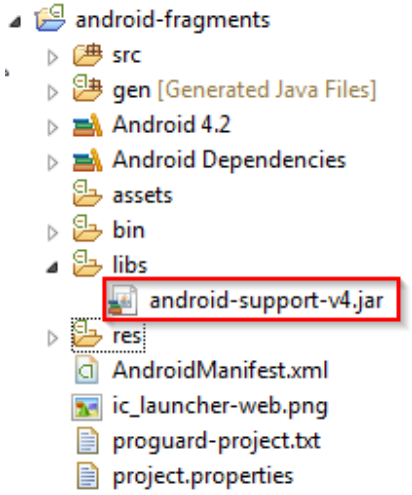
Fragments estáticos o finales y dinámicos:

- Un **Fragment estático o final** es aquel que se declara en el fichero XML de la carpeta Layout directamente. Este Fragment tendrá la cualidad de no ser eliminado o sustituido por nada -De lo contrario tendremos errores.
- Un **Fragment dinámico** es el que se crea desde código Java y se asocia a un ViewGroup (Se recomienda el uso de FrameLayout). Éste sí que se podrá eliminar o sustituir por otro Fragment u otro contenido.

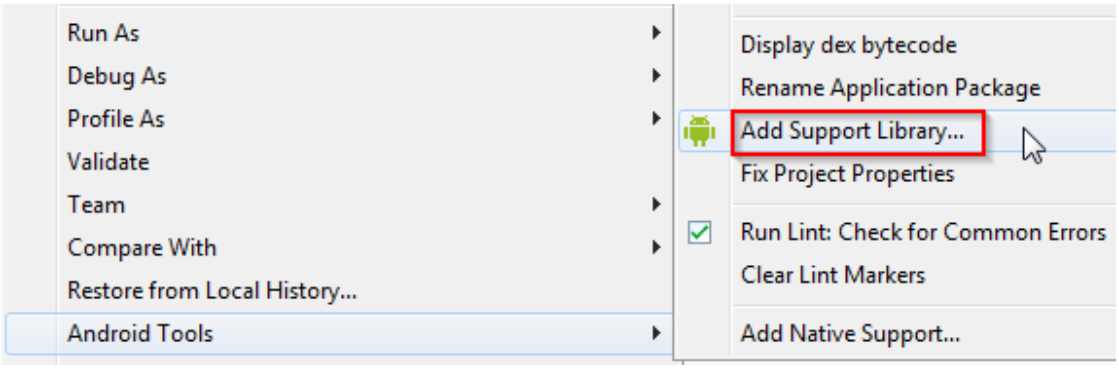


6. Librerías necesarias

Como hemos dicho, todo fragment debe tener asociada, además del layout, su propia clase java, que en este caso debe extender de la clase `Fragment`. Y aquí viene el primer contratiempo. Los fragment aparecieron con la versión 3 de Android, por lo que en principio no estarían disponibles para versiones anteriores. Sin embargo, Google pensó en todos y proporcionó esta característica también como parte de la librería de compatibilidad `android-support`, que en versiones recientes del plugin de Android para Eclipse se añade por defecto a todos los proyectos creados. Como se aprecia en la siguiente imagen:



Si no fuera así, también puede incluirse manualmente en el proyecto mediante la opción "Add Support Library…" del menú contextual del proyecto.



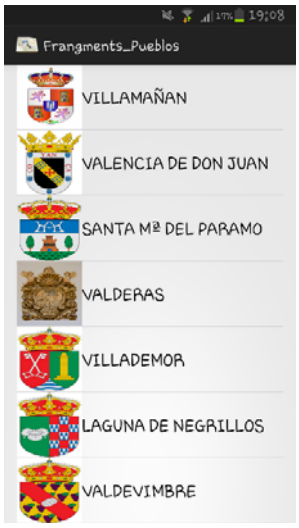
Hecho esto, ya no habría ningún problema para utilizar la clase `Fragment`, y otras que comentaremos más adelante, para utilizar fragmentos compatibles con la mayoría de versiones de Android.

7. Aplicación desarrollada

Se trata de una aplicación con dos fragments, uno con un listado, y el otro donde aparecen los detalles según el ítem seleccionado en el listado. Dependiendo en que dispositivo se visualice la aplicación tendrá una visualización u otra.

Así se vería la aplicación en un Smartphone:

La primera captura nos muestra la lista que está en un fragment, y al pulsar en uno de los ítems de la lista se abrirá otra activity con un fragmente detallando lo seleccionado.



Vertical Lista

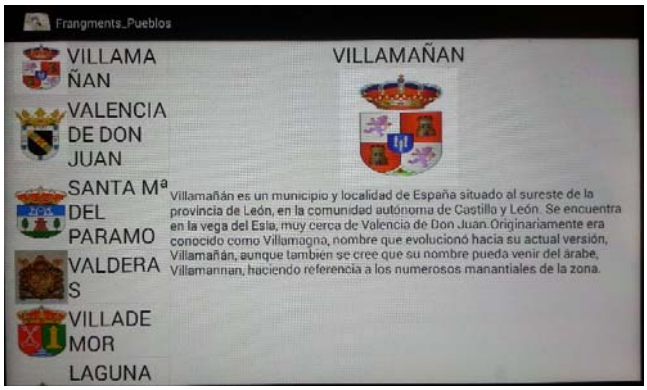
Vertical Detalle

Horizontal Lista

Horizontal Detalle

En las siguientes imágenes se muestra como se ve utilizando una Tablet. Aparece de forma diferente ya que han ser la pantalla de mayor tamaño nos muestra los dos fragment en la misma Activity. (El de la lista y el del detalle).

Vertical

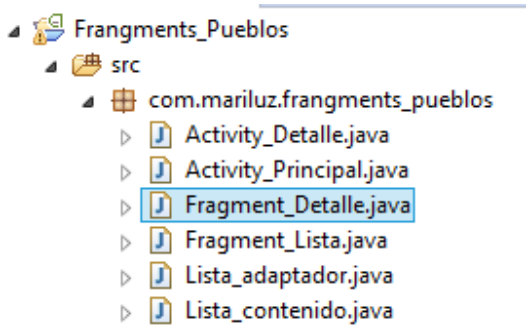


Horizontal

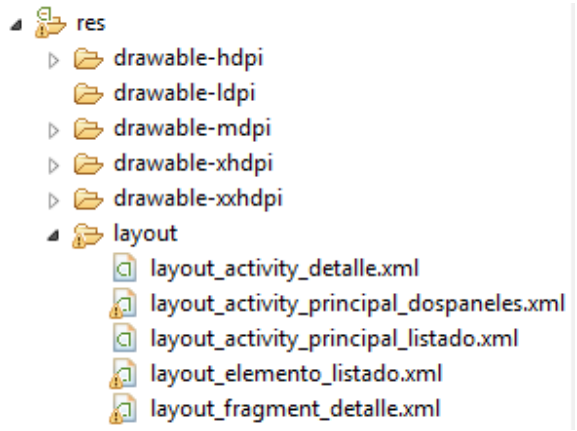
No mostraré todo el código de la aplicación, pero si haré referencia a algunas de sus partes. La aplicación se adjuntará con este pdf, para poder ver el código en su totalidad.

A continuación se muestra la estructura del proyecto.

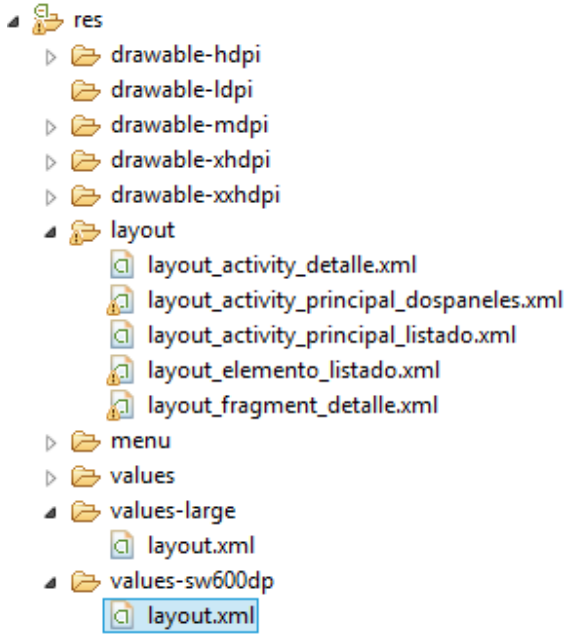
Clases creadas:



Layouts (carpeta /res/layout):



Dentro de la carpeta /res, se han creado dos carpetas, values-large y values-sw600dp. Con sus respectivos layouts.



Tendremos 3 carpetas “values”:

- values-large: para pantallas grandes; es decir, para Tablets (Necesario para que funcione en la versión 3 de Android). Crearemos en esta carpeta un nuevo fichero XML llamado “layout.xml” (lo explico un poco más adelante).
- values-sw600dp: para pantallas con un ancho mayor a 600dp (independientemente de la rotación de la pantalla); para Tablets de este tamaño o más. Este valor lo podemos cambiar, pero 600 nos va a funcionar muy bien para nuestros propósitos. Crearemos en esta carpeta otro nuevo fichero XML también llamado “layout.xml” (lo explico un poco más adelante).

- values: carpeta de valores para el resto de dispositivos que no cumplan las condiciones anteriores. Lo usaran los Smartphones, o las Tablets pequeñas que no cumplan el tamaño mínimo que hemos puesto. No tocamos ni añadimos nada en esta carpeta.

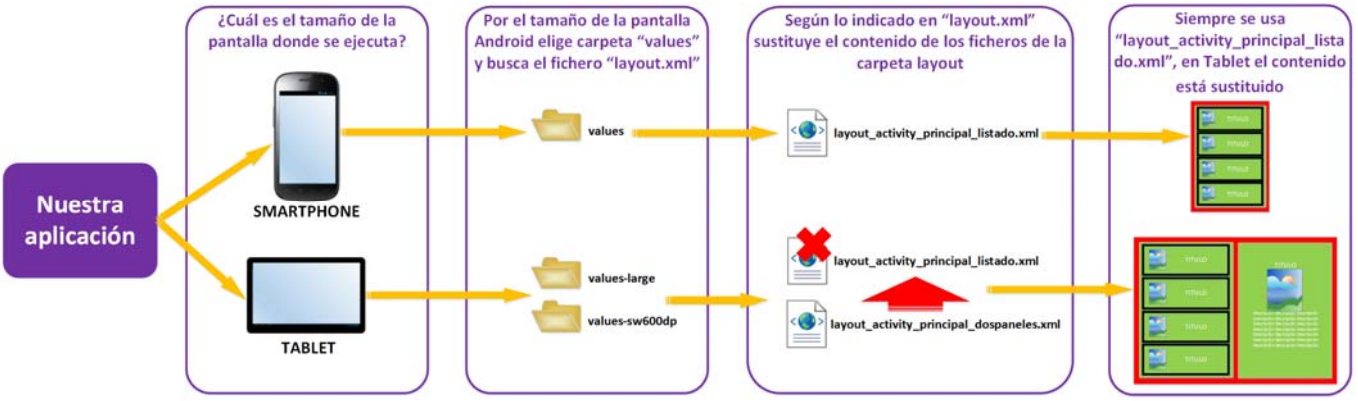
Hemos creado dos ficheros llamados “layout.xml”. Ambos contendrán el mismo contenido que es el siguiente:

```
<resources>

    <!--
    Alias del Layout que reemplaza la versión de solo listado con la de dos paneles en pantallas grandes
    -->
    <item name="layout_activity_principal_listado" type="layout">
        @layout/layout_activity_principal_dospaneles</item>

</resources>
```

Podríamos haber creado varias carpetas “layout” pero tendríamos código duplicado con lo mismo; con el alias evitamos repetir código innecesariamente. Este **alias lo único que hace es sustituir el contenido de un XML por otro**. Así, si la pantalla es pequeña cargará directamente el XML “layout_activity_principal_listado.xml” en el que solo se muestra un listado; y si es grande, el contenido de este fichero será sustituido por el contenido del fichero “layout_acitivity_principal_dospaneles.xml” que dispone un Fragment a la izquierda para el listado y un FrameLayout a la derecha para el Fragment dinámico del detalle. Para mayor claridad, echa un vistazo a la imagen siguiente:



También podemos usar el layout_activity_principal_dospaneles.xml para el Smartphone, lo único que se vería todo un poco amontonado ya que la pantalla es más pequeña que la de una Tablet, pero se vería con los dos fragment igual que una Tablet.

Empecemos por el código primero en ejecutarse, que es **Activity_Principal.java**. Esta es una Activity, pero para tratar con Fragments necesitamos un tipo de Activity especial llamada “FragmentActivity” de la que extenderemos la clase.

```
public class Activity_Principal extends FragmentActivity implements Fragment_Lista.Callbacks {
```

Pero antes vamos a explicar en esencia lo que hace, que son dos cosas:

Diferenciar la cantidad de elementos que maneja para cada tipo de dispositivo: En el "onCreate" le asociaremos el Layout "layout_activity_principal_listado.xml" –recordemos que su contenido tendrá únicamente un listado, o un listado y un detalle, dependiendo del tamaño de la pantalla. Para saber si estamos trabajando con dos Fragments en la misma Acitivity, es tan sencillo como preguntar si existe el detalle, sino existe estaremos trabajando con un solo Fragment que será sin lugar a dudas el listado.

Comunicar a los Fragments: Programaremos los Fragments de manera completamente modular; es decir, un Fragment no se comunica con otro directamente sino a través de interfaces. Funciona del siguiente modo:

1) Se pulsa sobre un elemento del Fragment que contiene al listado. A través de un método de Callback "onEntradaSelecionada" (esto es programación por eventos, en este artículo explicamos su comportamiento, pero resumimos al explicar el código por no ser el tema principal del artículo) comunica el id de la entrada seleccionada a la Activity que lo engloba (de ahí que implemente a "Fragment_Lista.Callbacks").

2) La Activity recoge el dato. Si:

   Es Tablet: crea un nuevo Fragment con el detalle, le envía el id para que cargue el contenido apropiado y lo coloca en el FrameLayout, remplazando cualquier otro Fragment con el detalle que pueda existir.

   Es Smartphone: Crea una nueva Activity al que le envía el id. La nueva Activity cargará el Fragment con el detalle sobre sí misma.

3) Se muestra el Fragment con el detalle al usuario.

Fragment_Lista.java. Esta clase es un Fragment, pero es un Fragment especial que extenderá de ListFragment, que es el listado preparado para ser Fragment (Aquí no asociamos ningún Layout, pues al extender de ListFragment ya lo trae puesto). Callback: Tendrá implementado el Callback que notifique a la Activity de que elemento del listado se haya pulsado. Explico uno a uno los métodos aquí usados:

Constructor: Seguro que te fijarás en el constructor vacío, es necesario para el correcto funcionamiento de los Fragments, no quitarlo.

onCreate: como ya hicimos en el artículo de listado asignamos el contenido de cada entrada a cada elemento del listado.

onAttach: Simplemente asegura que el desarrollador haya implementado el Callback, en la clase que use a este Fragment.

onDetach: Limpia el Callback.

onListItemClick: al extender de ListFragment, es necesario sobrescribir esta clase, que es la que escucha la pulsación sobre un elemento de la lista. Y así usaremos el Callback para notificar a la Actividad del id pulsado.

```
public class Fragment_Lista extends ListFragment {

    /**
     * El callback del fragmento que notificará de pulsaciones en la lista
     */
    private Callbacks mCallbacks = CallbacksVacios;
```

Hasta aquí ya tenemos lo que hace funcionar la estructura de los Tablets.

Continuemos para el caso de Smartphone. Necesitamos un segundo Activity para contener al Fragment del detalle.

Activity_Detalle. Extiende de FragmentActivity. Solo tiene un método onCreate que comprobará si ya ha creado al Fragment del detalle para cargarlo, o no para crearlo. La creación del Fragment del detalle es igual que en el caso de Activity_Principal.java.

```
public class Activity_Detalle extends FragmentActivity {

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

        // Comprobamos que previamente no hayamos entrado en esta actividad
        //(por ejemplo, al rotar el dispositivo). Si es así se añade el fragmento al contenedor
        if (savedInstanceState == null) {
            // Crea el fragmento del detalle de la entrada y lo añade a la actividad
            Bundle arguments = new Bundle();
            arguments.putString(Fragment_Detalle.ARG_ID_ENTRADA_SELECIONADA,
                    getIntent().getStringExtra(Fragment_Detalle.ARG_ID_ENTRADA_SELECIONADA));
            Fragment_Detalle fragment = new Fragment_Detalle();
            fragment.setArguments(arguments);
            getSupportFragmentManager().beginTransaction().add(R.id.framelayout_contenedor_detalle, fragment)
            .commit();
        }
```

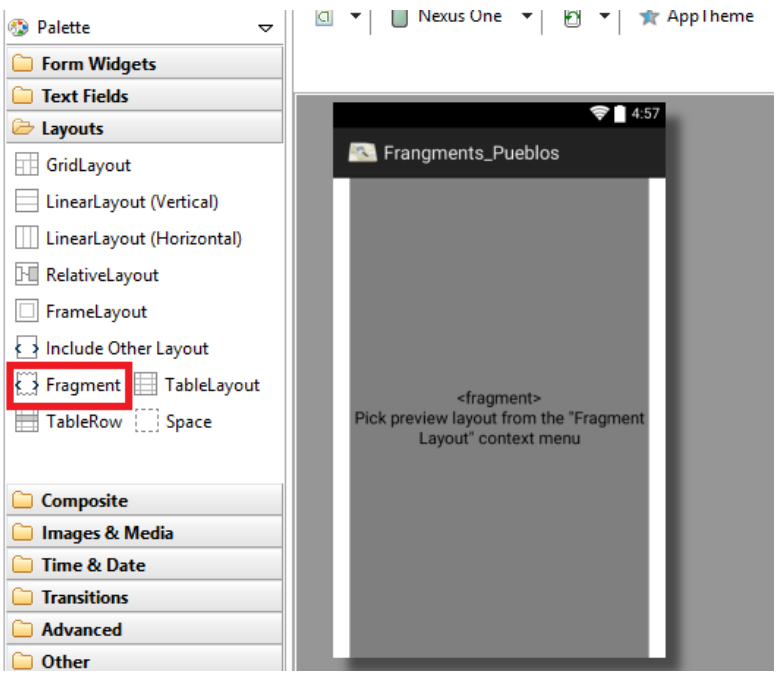
Lista_adaptador.java. Esta clase crea el adaptador para la lista.

Lista_contenido.java, que contiene a la clase Lista_entrada del artículo del listado. Lo hemos hecho así por simplificar el código. Ya que si los datos no van a cambiar, así es muy fácil hacer una estructura estática que de elementos que siempre esté visible por todas las clases sea el momento que sea de la ejecución. Simplemente guardamos unos listados con el contenido de las entradas estáticamente, esto se hace al inicio de la aplicación.

No se nos olvide añadir al **AndroidManifest.xml** la Actividad que no es la principal, es decir Activity_Detalle.

```
<activity
    android:name="com.mariluz.frangments_pueblos.Activity_Detalle"
    android:label="@string/titulo_detalle_entrada"
    android:parentActivityName=".Activity_Principal" >
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".ItemListActivity" />
</activity>
```

Para agregar fragments a los layouts, lo haremos desde Palette/Layouts/Fragment :



Cuando introducimos un componente fragment debemos seleccionar la clase a la que va asociado. Nos saldrá una pantalla en la que seleccionaremos la clase deseada.

En lo referente a los layouts:

El código de **layout_activity_principal_dospaneles.xml** será:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:baselineAligned="false"
    android:divider="?android:attr/dividerHorizontal"
    android:orientation="horizontal"
    android:showDividers="middle" >
    <!--
    Este fragmento estático contiene al listado
    -->
    <fragment
        android:id="@+id/fragment_listado"
        android:name="com.mariluz.frangments_pueblos.Fragment_Lista"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3" />
    <!--
    Este FrameLayout contendrá un fragmento dinámico con el contenido del elemento pulsado del listado
    -->
    <FrameLayout
        android:id="@+id/framelayout_contenedor_detalle"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3" />

</LinearLayout>
```

Para móvil, tenemos por separado el listado con el Fragment estático. Cuyo código de **layout_activity_principal_listado.xml** es:

```
<!--
Vista que contiene 1 panel para pantallas pequeñas.
-->

<!--
Este fragmento estático contiene al listado
-->
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_listado"
    android:name="com.mariluz.frangments_pueblos.Fragment_Lista"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp" />
```

Y por otra parte, de manera separada pero con la misma lógica al que hemos explicado de la Tablet, tenemos el Fragment dinámico para el detalle.

Para **layout_activity_detalle.xml**:

```
<!--
Vista que contiene 1 panel para pantallas pequeñas.
-->

<!--
Este FrameLayout contendrá un fragmento dinámico con el contenido del elemento pulsado del listado
-->
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/framelayout_contenedor_detalle"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

También tenemos la estructura simple de la vista de cada elemento del listado en **layout_elemento_listado.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center|left"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/imageView_imagen_miniatura"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:adjustViewBounds="true"
        android:scaleType="fitXY"
        android:contentDescription="Descripción del contenido de la imagen"
        android:src="@android:drawable/ic_menu_gallery" />

    <TextView
        android:id="@+id/textView_titulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</LinearLayout>
```

Y el del detalle, en **layout_fragment_detalle.xml**:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center|top"
    android:orientation="vertical" >

    <ScrollView
        android:id="@+id/scrollView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

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

            <TextView
                android:id="@+id/textView_superior"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="Large Text"
                android:textAppearance="?android:attr/textAppearanceLarge" />

            <ImageView
                android:id="@+id/imageView_imagen"
                android:layout_width="178dp"
                android:layout_height="178dp"
                android:layout_gravity="center"
                android:adjustViewBounds="true"
                android:contentDescription="Descripción del contenido de la imagen"
                android:scaleType="fitXY"
                android:src="@android:drawable/ic_menu_gallery" />

            <TextView
                android:id="@+id/textView_inferior"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="Small Text"
                android:textAppearance="?android:attr/textAppearanceSmall" />
        </LinearLayout>

    </ScrollView>

</LinearLayout>
```

8. Bibliografía

- http://jarroba.com/fragments-fragmentos-en-android/
- http://jarroba.com/programar-fragments-fragmentos-en-android/
- http://elbauldeandroid.blogspot.com.es/2013/12/fragments.html
- http://fsvelectronicainformatica.blogspot.com.es/2013/01/que-son-los-fragments-fragmentos-en.html
- http://www.sgoliver.net/blog/fragments-en-android/
- https://sekthdroid.wordpress.com/2013/02/01/introduccion-fragments-en-android/