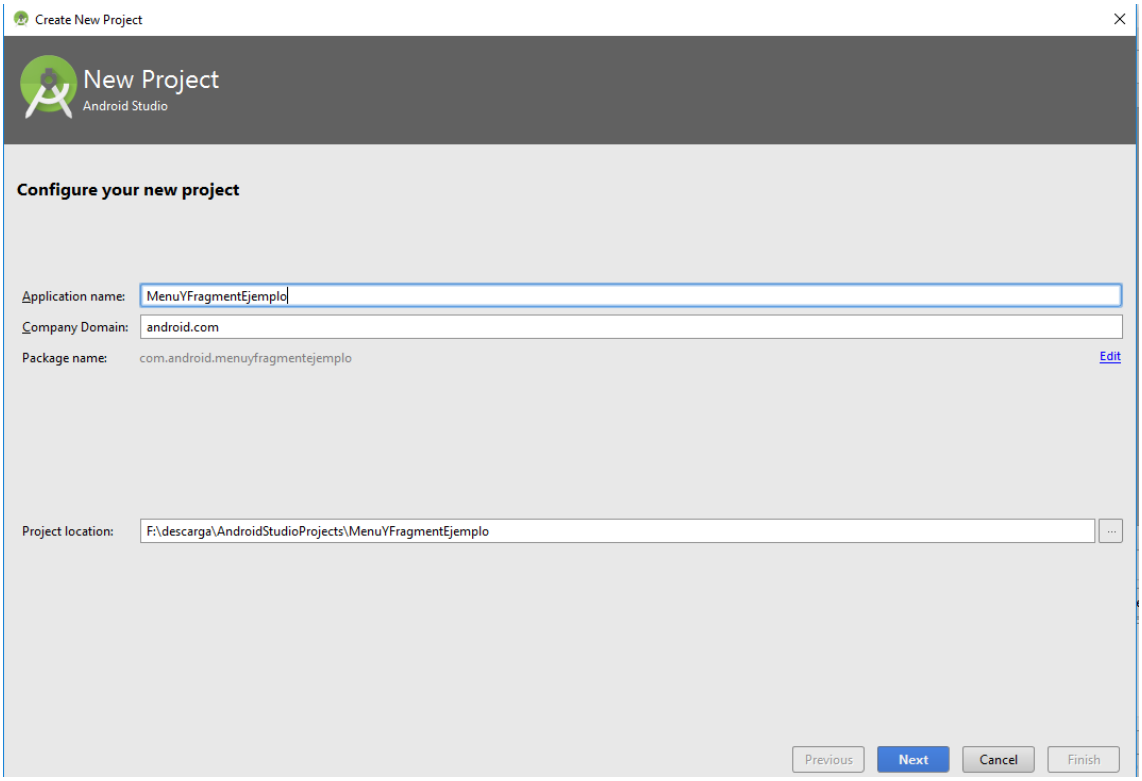
Ejemplo usando Menú y Fragmento en Android

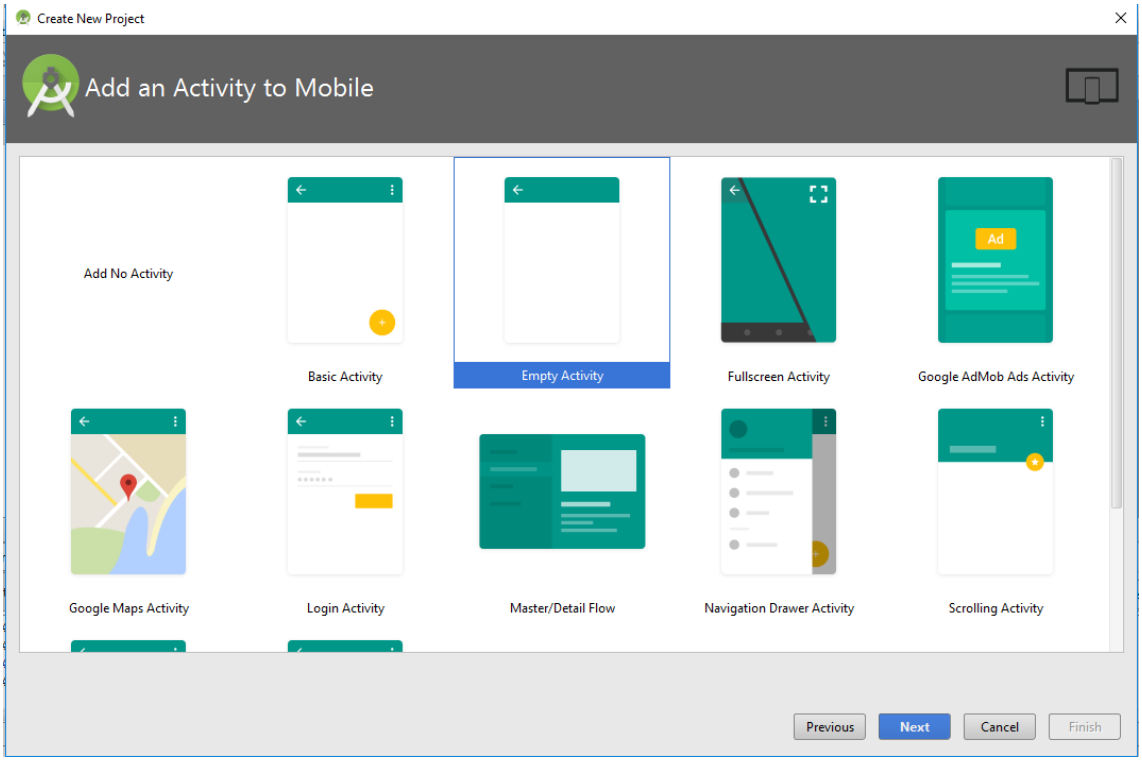
Juan Antonio Japón de la Torre

06/09/2016

Creación del Proyecto



Seleccionamos la opción Empty Activity para tener una vista vacia



Layout Activity (Main Activity)

Para poder cargar fragmentos encima de nuestro Activity una vez pulsemos en una u otra opción de nuestro menu modificaremos nuestro main layout y lo dejaremos como un FrameLayout. Este tipo de layout permite añadir encima

```
<?xml version="1.0" encoding="utf-8"?>
<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:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:id="@+id/contenedorLayouts"></FrameLayout>
```

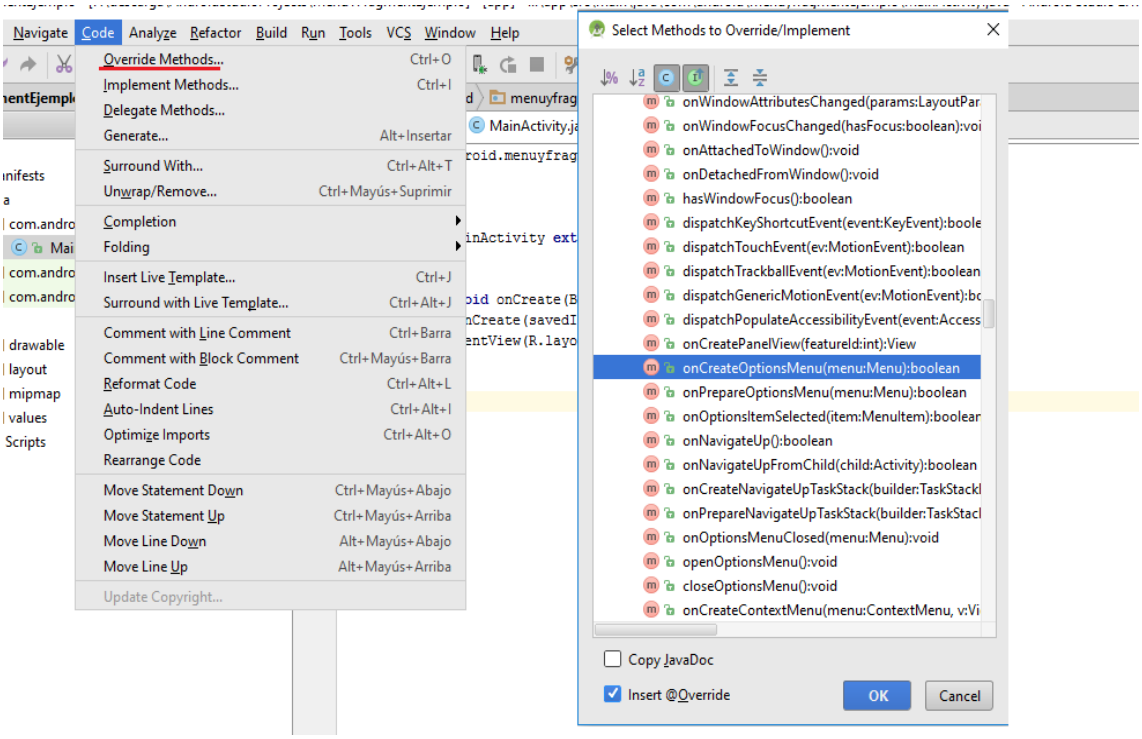
2. Activity o Controlador

Se nos habrá creado una actividad vacía:

```
package com.android.menuyfragmentejemplo;

import ...

public class MainActivity extends AppCompatActivity {

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

En dicho activity implementaremos un menú para cambiar de vista pulsando en sus opciones. Para ello implementaremos los métodos **OnCreateOptionsMenu** y **OnOptionsItemSelected** que es el que se encarga de cargar el menú y el segundo gestionara los eventos de las opciones.

Para importar este método debemos pulsar en Code/Override Methods y buscar el método **OnCreateOptionsMenu** y el **OnOptionsItemSelected**

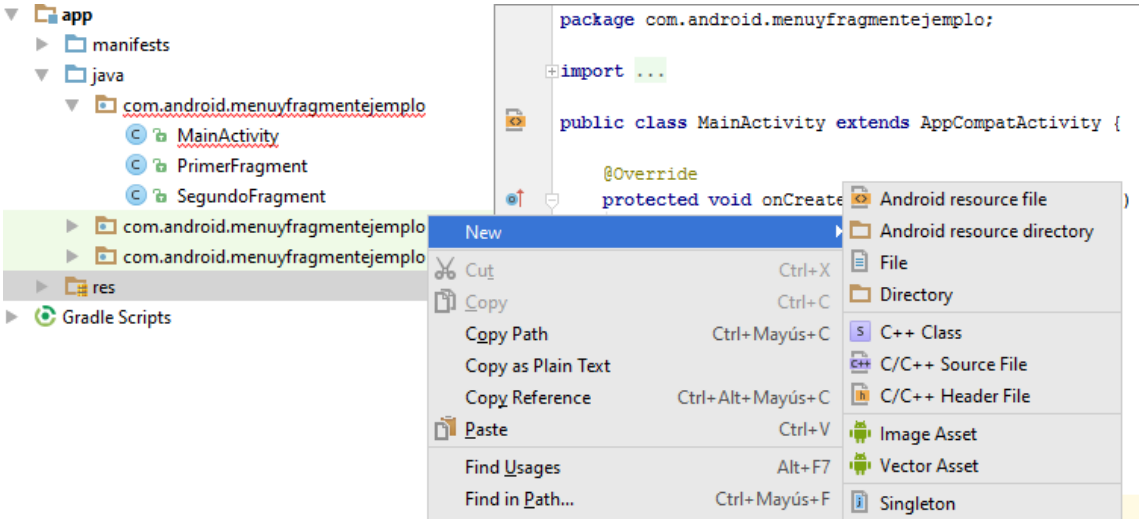


Creación y configuración de nuestro menú

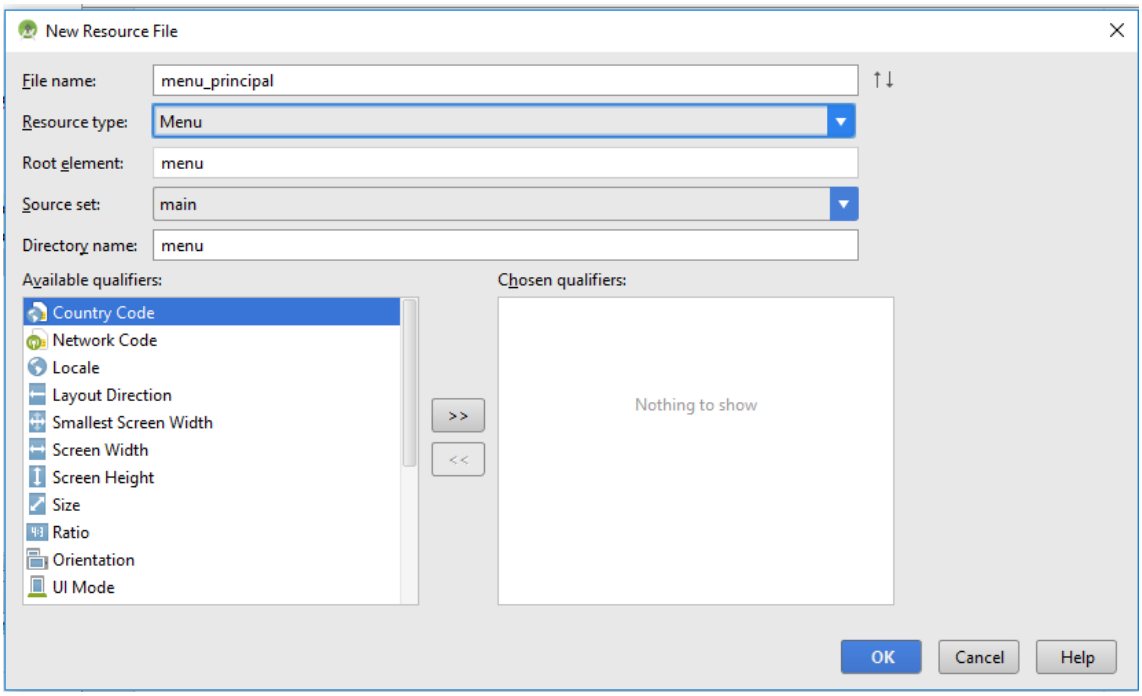
Con el paso realizado anteriormente tendremos el método OncreateOptionsMenu ya creado en nuestro Activity Main que lo dejaremos de la siguiente manera:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_principal,menu);
    return super.onCreateOptionsMenu(menu);
}
```

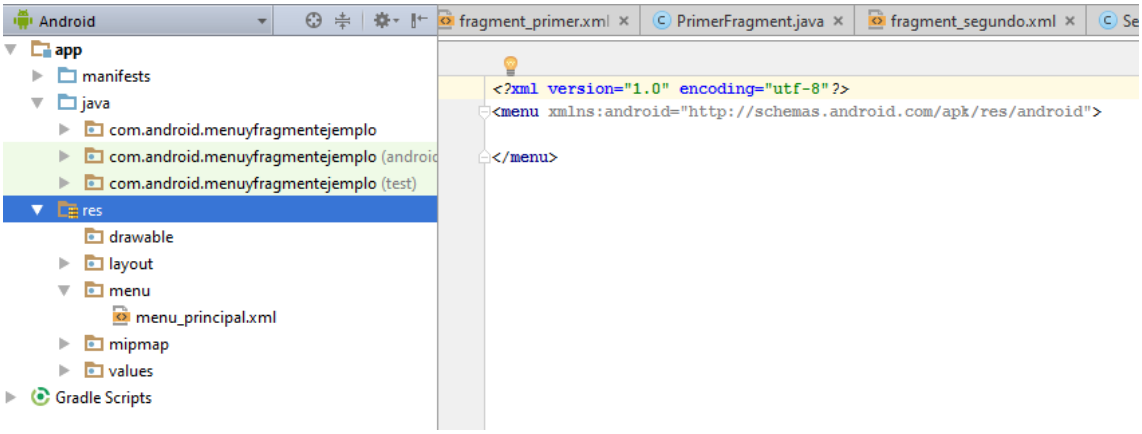
El archivo R.menu.menu_principal hace referencia a un menu que crearemos en la carpeta **res/menu/menu_principal.xml.** Para crearlo pulsaremos en la carpeta res, le daremos a **new/Android resource file**



Introducimos el nombre que va a tener y seleccionamos en resource type la opción menu



De esta manera se nos habrá creado en nuestra carpeta res un directorio llamado menu que contendrá nuestro archivo xml con nuestro menú vacio de momento



Para añadirle opciones a nuestro menu debemos añadir ítems como podemos ver justo debajo:

```
<item android:title="Cambiar vista"
android:icon="@android:drawable/ic_menu_rotate"></item>
```

Quedando de la siguiente manera

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

```
     //Item de ejemplo
    <item android:id="@+id/opcion1" android:title="Cambiar vista"
MenuYFragmentEjemplo:showAsAction="always"
android:icon="@android:drawable/ic_menu_rotate" />
</menu>
```

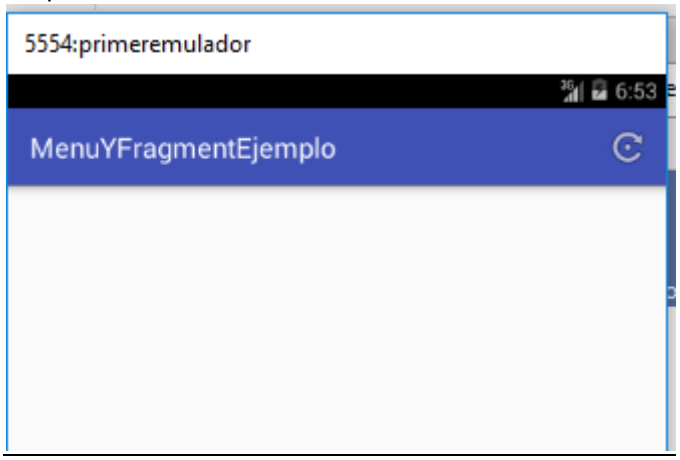
Para poder llamar a showAsAction (encargado de mostrar el icono en el menu de forma visible) debemos importar en la etiqueta menu

xmlns:nombreApp="http://schemas.android.com/apk/res-auto"
Ejemplo: **xmlns:MenuYFragmentEjemplo=http://schemas.android.com/apk/res-auto**

Y en los ítems utilizar la abreviatura que hemos usado con anterioridad para poder trabajar con showAsAction que es:

nombreApp:showAsAction="always"
Ejemplo: **MenuYFragmentEjemplo:showAsAction="always"**

Una vez realizado el cambio si ejecutamos la app nos aparecera en nuestro menu el icono con la opción



Preparando las opciones de nuestro menu en el Activity

Una vez tenemos esto en nuestra actividad principal implementaremos el método **OnOptionsItemSelected** que será el que gestionará los eventos pulsados en el menú:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    return super.onOptionsItemSelected(item);
}
```

Una vez importado crearemos un switch para gestionar las opciones del menu:

```
switch(item.getItemId()){
    case R.id.opcion1:
        //Evento a realizar
        break;
    default:
        return super.onOptionsItemSelected(item);
}
```

Como solo tenemos una opción creada solo tendrá un caso. El método completo quedará asi hasta que añadamos los eventos:

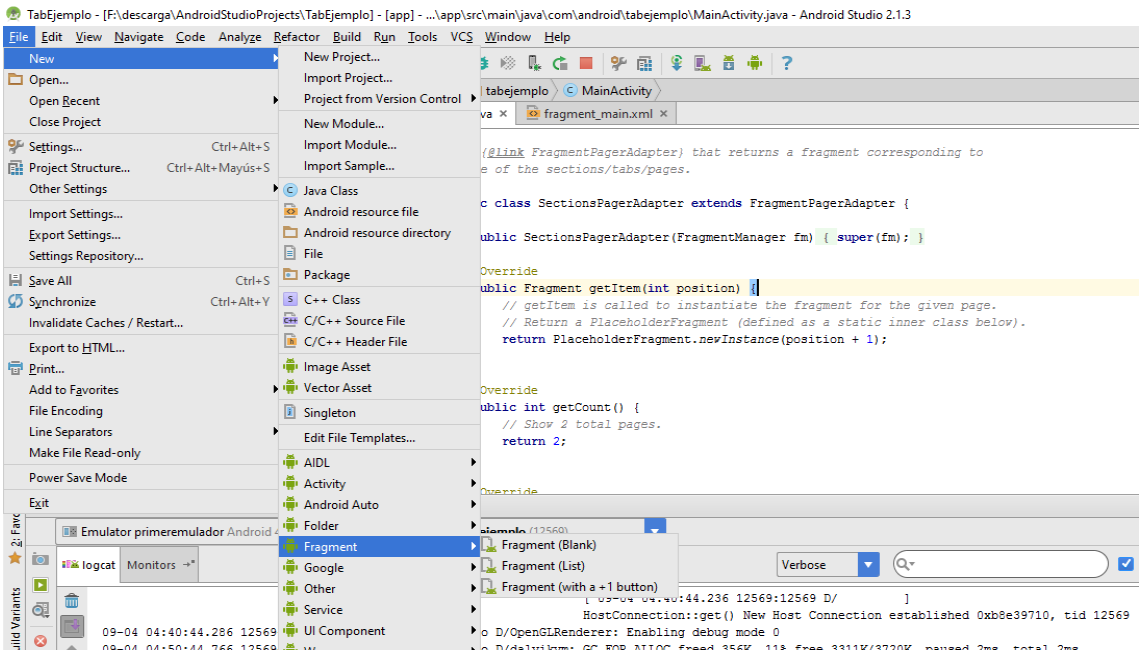
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()){
        case R.id.opcion1:
            //Evento a realizar
            break;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

3. Fragments

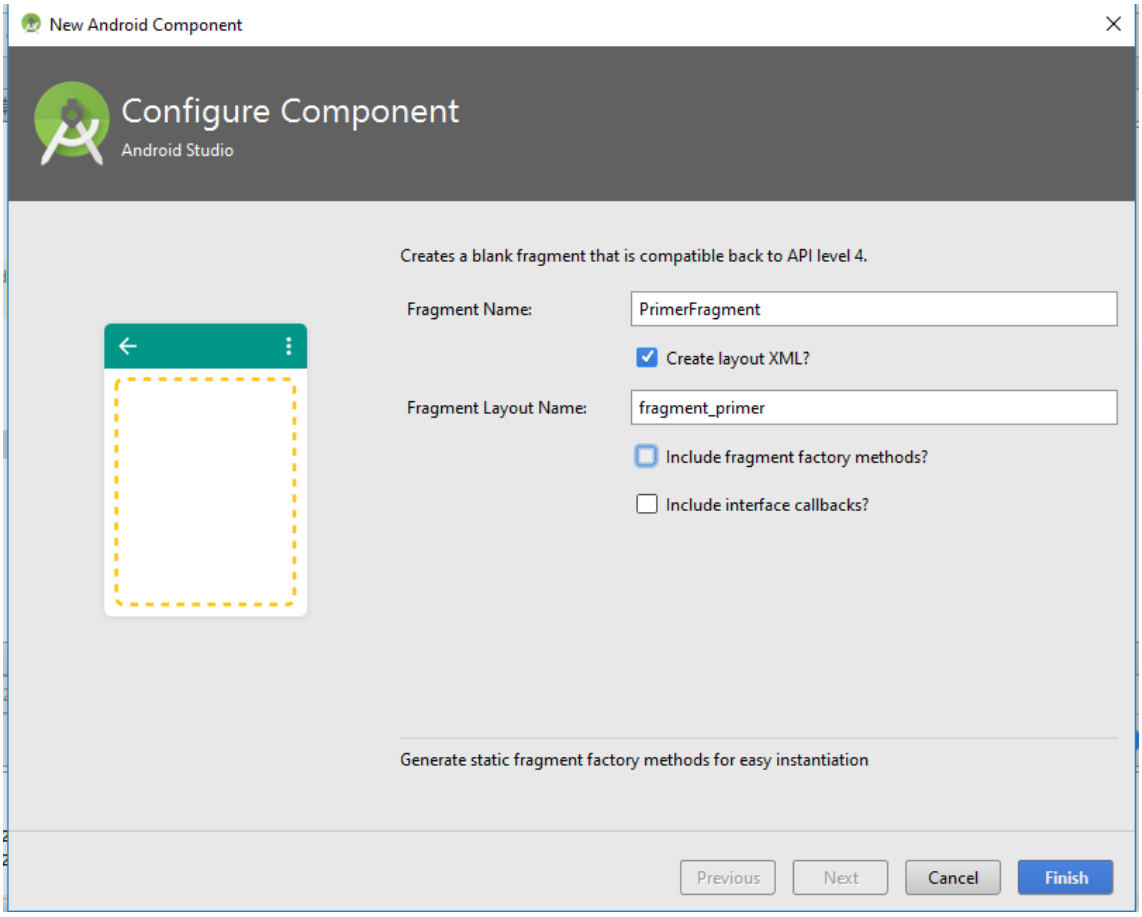
Los fragments son las vistas que serán cargadas en nuestro Tab. Difiere de los Activity en que podemos cargar varios fragmentos en una misma activity como si tuviéramos varios ejecutándose a la vez.

3.1 Creando un Fragment

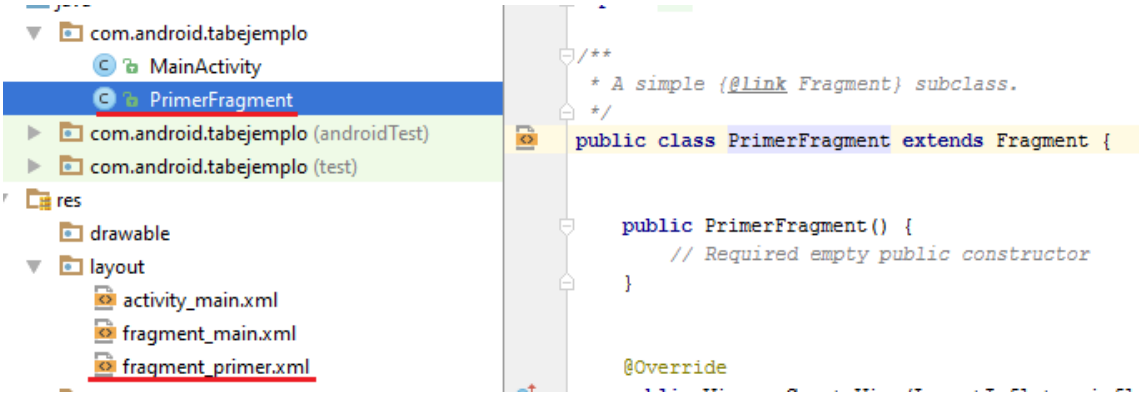
Para crear un Fragment debemos pulsar en la casilla de File/New/Fragment/Fragment(Blank) para que nos cree un Fragment en Blanco sin nada



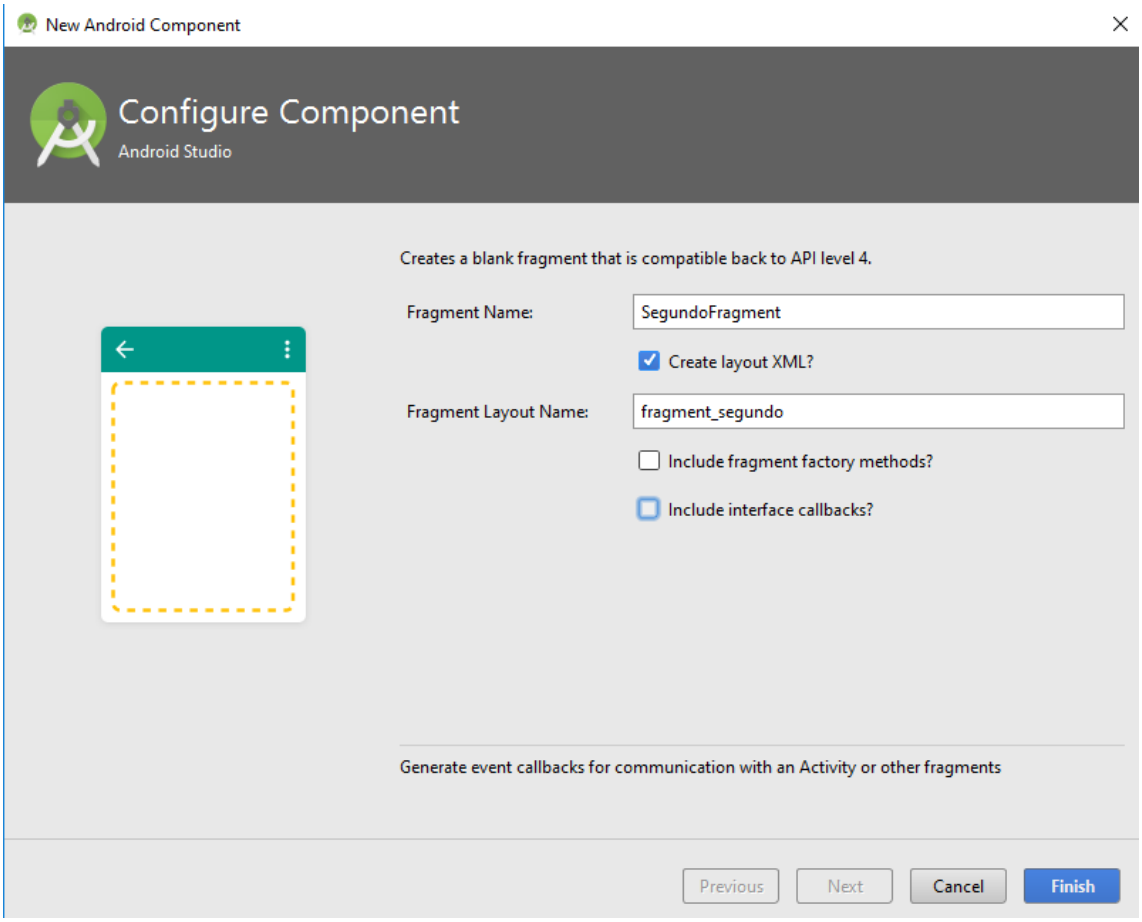
Le daremos un nombre a la actividad que hará de fragment, le daremos un nombre a la vista del Fragment y desmarcaremos las opciones de abajo para tener un código limpio



De esta manera se nos habrá creado un Activity y un XML con la vista



De la misma manera creamos un Segundo Fragment ya que nuestra App tenía 2 opciones



Gestión de eventos en el método OnOptionsItemSelected

Por último controlaremos el fragmento que se cargará en el evento. De forma previa crearemos 4 variables globales:

- Variable de tipo Framelayout para cargar una vista u otra
- Variable de tipo booleana para controlar el cambio de una vista a otra
- 2 variables que instancien los 2 fragmentos creados anteriormente que será los que se muestren en nuestra vista principal

```
FrameLayout contenedorLayouts;
boolean changeOption = true;
PrimerFragment primerF;
SegundoFragment segundoF;
```

Una vez creadas todas las variables. Inicializaremos en el método OnCreate el layout principal y asignarle un fragmento de inicio;

```
//Inicializamos el contenedor y un primer fragmento que se cargara de
inicio
contenedorLayouts = (FrameLayout)
findViewById(R.id.contenedorLayouts);
primerF = new PrimerFragment();
segundoF = new SegundoFragment();
//Realizamos una transaccion para cargar el fragmento que deseamos d
forma inicial
getSupportFragmentManager().beginTransaction().add(R.id.contenedorLayo
uts,primerF).commit();
```

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

    contenedorLayouts = (FrameLayout) findViewById(R.id.contenedorLayouts);
    primerF = new PrimerFragment();

    getSupportFragmentManager().beginTransaction().add(R.id.contenedorLayouts,primerF).commit();
}
```

Esto solo valdría para mostrar un fragmento por primera vez. Si queremos hacer un evento de cambio de fragment debemos irnos al método **onOptionsItemSelected** y hacer una condición para gestionar el cambio de un fragmento a otro usando la variable booleana que creamos al principio:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()){
        case R.id.opcion1:
            //Evento a realizar
            if(changeOption==false){

getSupportFragmentManager().beginTransaction().replace(R.id.contenedor
Layouts,new SegundoFragment()).commit();
                changeOption = true;
            }else{

getSupportFragmentManager().beginTransaction().replace(R.id.contenedor
Layouts,new SegundoFragment()).commit();
                changeOption = false;
            }
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Como resultado la aplicación quedaría asi: