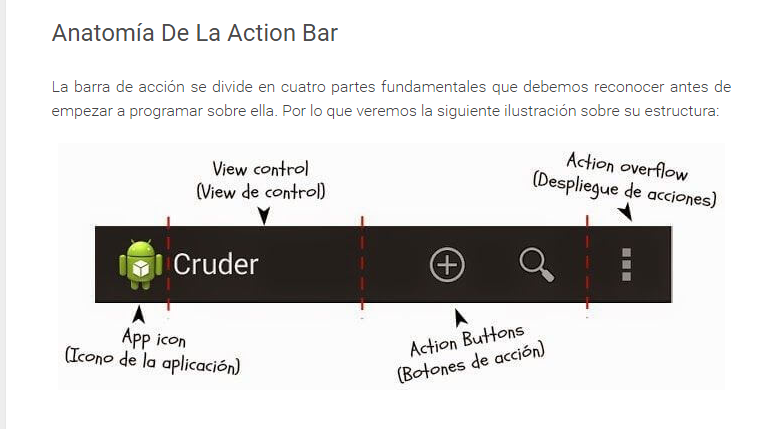
<http://www.hermosaprogramacion.com/2014/09/android-action-bar/>

<http://www.hermosaprogramacion.com/2016/02/controles-tutorial-botones-android/>

<http://www.hermosaprogramacion.com/2014/11/android-navigation-drawer-tutorial/>



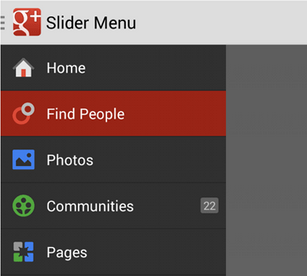
MENU

Éstas dimensiones pueden ser el margen que queda visible, tamaño de la sombra, etc. Un ejemplo sería:

[?](http://androcode.es/2013/01/implementacion-del-menu-lateral/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <com.slidingmenu.lib.SlidingMenu      xmlns:sliding="http://schemas.android.com/apk/res-auto"      android:id="@+id/slidingmenulayout"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      sliding:viewAbove="@layout/YOUR\_ABOVE\_VIEW"      sliding:viewBehind="@layout/YOUR\_BEHIND\_BEHIND"      sliding:touchModeAbove="margin|fullscreen"      sliding:behindOffset="@dimen/YOUR\_OFFSET"      sliding:behindWidth="@dimen/YOUR\_WIDTH"      sliding:behindScrollScale="@dimen/YOUR\_SCALE"      sliding:shadowDrawable="@drawable/YOUR\_SHADOW"      sliding:shadowWidth="@dimen/YOUR\_SHADOW\_WIDTH"      sliding:fadeEnabled="true|false"      sliding:fadeDegree="float"      sliding:selectorEnabled="true|false"      sliding:selectorDrawable="@drawable/YOUR\_SELECTOR"/> |

Menú lateral desplegable DrawerLayout android (Slider)

[](http://3.bp.blogspot.com/-5dXukCD_5cM/U53jyTxzzLI/AAAAAAAAAus/2Y16PmtPgPg/s1600/ffff.PNG)

Buenos días!  
  
Hoy vamos a ver como crear un menú desplegable del tipo Google Maps o google plus +.  Parece algo complicado pero he intentado hacerlo lo más simple posible! eso si, siempre se puede perfeccionar muchísimo subiendo así el nivel de dificultad.  
Para que funcione el nivel desplegable es necesario API 11 y si queremos desplearlo con un botón en vez de arrastrando con el dedo ará falta la versión API 14.  
  
  
  
  
  
Como siempre he creado el código totalmente comentado. Para empezar mostraré el Xml, basado en  android.support.v4.widget.DrawerLayout, simplemente he añadido un listview.

**<android.support.v4.widget.DrawerLayout**

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"**>**

**<FrameLayout**

android:id="@+id/content\_frame"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" **>**

**</FrameLayout>**

Como siempre he creado el código totalmente comentado. Para empezar mostraré el Xml, basado en  android.support.v4.widget.DrawerLayout, simplemente he añadido un listview.

**<android.support.v4.widget.DrawerLayout**

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"**>**

**<FrameLayout**

android:id="@+id/content\_frame"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" **>**

**</FrameLayout>**

**<ListView** android:id="@+id/left\_drawer"

android:layout\_width="240dp"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

android:choiceMode="singleChoice"

android:divider="@android:color/transparent"

android:dividerHeight="0dp"

android:background="#ff11"**/>**

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

Una vez visto el Xml a continuación el Java:

**package** com.tywors.menulateral.app;

**import** **android.content.res.Configuration**;

**import** **android.support.v4.app.ActionBarDrawerToggle**;

**import** **android.support.v4.widget.DrawerLayout**;

**import** **android.support.v7.app.ActionBarActivity**;

**import** **android.os.Bundle**;

**import** **android.view.MenuItem**;

**import** **android.view.View**;

**import** **android.widget.AdapterView**;

**import** **android.widget.ArrayAdapter**;

**import** **android.widget.ListView**;

**import** **android.widget.Toast**;

**public** **class** **MainActivity** **extends** ActionBarActivity {

*//El drawerLayout esn el que se desplega y*

*//contiene dentro el menú, normalmente un listview*

**private** DrawerLayout mDrawerLayout;

*//Declaremos el ListView*

**private** ListView mDrawerList;

*//ActionBarDrawerToggle es donde aparecerá el boton*

*//para desplegar el menú*

**private** ActionBarDrawerToggle mDrawerToggle;

**@Override**

**protected** void onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

*//relacionamos ocn el XML*

mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer\_layout);

mDrawerList = (ListView) findViewById(R.id.left\_drawer);

*//Configuramos el Boton que desplegará el menú*

mDrawerToggle = **new** ActionBarDrawerToggle(

**this**, *//la actividad*

mDrawerLayout, *//el drawerLayout que desplegará*

R.drawable.ic\_launcher, *//el icono que mostraremos*

R.string.app\_name, *//descripción al abrir*

R.string.app\_name *//descripción al cerrar*

) { };

*//Creamos nuestro menú*

**final** String[] opciones = {"Inicio", "Descargas", "Salir"};

*//rellenamos la List view*

mDrawerList.setAdapter(**new** ArrayAdapter<String>(**this**,

android.R.layout.simple\_list\_item\_1,

android.R.id.text1, opciones));

*//Añadimos la acción que haga en cada fila del*

*//list view. en este caso solo mostraremos un Toast con un mensaje*

mDrawerList.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {

**@Override**

**public** void onItemClick(AdapterView<?> arg0, View arg1,

int arg2, long arg3) {

Toast.makeText(MainActivity.this, "id: " + opciones[arg2],

Toast.LENGTH\_SHORT).show();

*//Se cierra el menú*

mDrawerLayout.closeDrawers();

}

});

*//Mostramos el botón en la barra de la aplicación*

getActionBar().setDisplayHomeAsUpEnabled(**true**);

*//Activamso el click en el icono de la aplicación*

getActionBar().setHomeButtonEnabled(**true**);

}

*//Que el botón de desplegar siempre este sincronizado*

**@Override**

**protected** void onPostCreate(Bundle savedInstanceState) {

**super**.onPostCreate(savedInstanceState);

mDrawerToggle.syncState();

}

*//Igual con la configuración*

**@Override**

**public** void onConfigurationChanged(Configuration newConfig) {

**super**.onConfigurationChanged(newConfig);

mDrawerToggle.onConfigurationChanged(newConfig);

}

*//Activamos el click paradesplegar*

**@Override**

**public** boolean onOptionsItemSelected(MenuItem item) {

**if** (mDrawerToggle.onOptionsItemSelected(item)) {

**return** **true**;

}

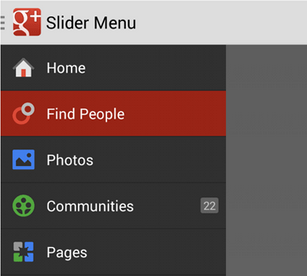
**return** **super**.onOptionsItemSelected(item);

}

}

## domingo, 15 de junio de 2014

### Menú lateral desplegable DrawerLayout android (Slider)

[](http://3.bp.blogspot.com/-5dXukCD_5cM/U53jyTxzzLI/AAAAAAAAAus/2Y16PmtPgPg/s1600/ffff.PNG)

Buenos días!  
  
Hoy vamos a ver como crear un menú desplegable del tipo Google Maps o google plus +.  Parece algo complicado pero he intentado hacerlo lo más simple posible! eso si, siempre se puede perfeccionar muchísimo subiendo así el nivel de dificultad.  
Para que funcione el nivel desplegable es necesario API 11 y si queremos desplearlo con un botón en vez de arrastrando con el dedo ará falta la versión API 14.  
  
  
  
  
  
Como siempre he creado el código totalmente comentado. Para empezar mostraré el Xml, basado en  android.support.v4.widget.DrawerLayout, simplemente he añadido un listview.

**<android.support.v4.widget.DrawerLayout**

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"**>**

**<FrameLayout**

android:id="@+id/content\_frame"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" **>**

**</FrameLayout>**

**<ListView** android:id="@+id/left\_drawer"

android:layout\_width="240dp"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

android:choiceMode="singleChoice"

android:divider="@android:color/transparent"

android:dividerHeight="0dp"

android:background="#ff11"**/>**

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

Una vez visto el Xml a continuación el Java:

**package** com.tywors.menulateral.app;

**import** **android.content.res.Configuration**;

**import** **android.support.v4.app.ActionBarDrawerToggle**;

**import** **android.support.v4.widget.DrawerLayout**;

**import** **android.support.v7.app.ActionBarActivity**;

**import** **android.os.Bundle**;

**import** **android.view.MenuItem**;

**import** **android.view.View**;

**import** **android.widget.AdapterView**;

**import** **android.widget.ArrayAdapter**;

**import** **android.widget.ListView**;

**import** **android.widget.Toast**;

**public** **class** **MainActivity** **extends** ActionBarActivity {

*//El drawerLayout esn el que se desplega y*

*//contiene dentro el menú, normalmente un listview*

**private** DrawerLayout mDrawerLayout;

*//Declaremos el ListView*

**private** ListView mDrawerList;

*//ActionBarDrawerToggle es donde aparecerá el boton*

*//para desplegar el menú*

**private** ActionBarDrawerToggle mDrawerToggle;

**@Override**

**protected** void onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

*//relacionamos ocn el XML*

mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer\_layout);

mDrawerList = (ListView) findViewById(R.id.left\_drawer);

*//Configuramos el Boton que desplegará el menú*

mDrawerToggle = **new** ActionBarDrawerToggle(

**this**, *//la actividad*

mDrawerLayout, *//el drawerLayout que desplegará*

R.drawable.ic\_launcher, *//el icono que mostraremos*

R.string.app\_name, *//descripción al abrir*

R.string.app\_name *//descripción al cerrar*

) { };

*//Creamos nuestro menú*

**final** String[] opciones = {"Inicio", "Descargas", "Salir"};

*//rellenamos la List view*

mDrawerList.setAdapter(**new** ArrayAdapter<String>(**this**,

android.R.layout.simple\_list\_item\_1,

android.R.id.text1, opciones));

*//Añadimos la acción que haga en cada fila del*

*//list view. en este caso solo mostraremos un Toast con un mensaje*

mDrawerList.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {

**@Override**

**public** void onItemClick(AdapterView<?> arg0, View arg1,

int arg2, long arg3) {

Toast.makeText(MainActivity.this, "id: " + opciones[arg2],

Toast.LENGTH\_SHORT).show();

*//Se cierra el menú*

mDrawerLayout.closeDrawers();

}

});

*//Mostramos el botón en la barra de la aplicación*

getActionBar().setDisplayHomeAsUpEnabled(**true**);

*//Activamso el click en el icono de la aplicación*

getActionBar().setHomeButtonEnabled(**true**);

}

*//Que el botón de desplegar siempre este sincronizado*

**@Override**

**protected** void onPostCreate(Bundle savedInstanceState) {

**super**.onPostCreate(savedInstanceState);

mDrawerToggle.syncState();

}

*//Igual con la configuración*

**@Override**

**public** void onConfigurationChanged(Configuration newConfig) {

**super**.onConfigurationChanged(newConfig);

mDrawerToggle.onConfigurationChanged(newConfig);

}

*//Activamos el click paradesplegar*

**@Override**

**public** boolean onOptionsItemSelected(MenuItem item) {

**if** (mDrawerToggle.onOptionsItemSelected(item)) {

**return** **true**;

}

**return** **super**.onOptionsItemSelected(item);

}

}

### Tutorial Android: Crear degradado en botón o layout

[](http://2.bp.blogspot.com/-PtjkoQNRVao/UYENdysHZiI/AAAAAAAAAeI/07QE0BfkfxY/s1600/linea-y-esferas-de-energia-en-texto-1.jpg)

En este tutorial vamos a aprender como crear un degradado en un botón o en una capa ya que el procedimiento es igual. Para el ejemplo utilizaremos un Button.  
  
El procedimiento inicial es el mismo que en un tutorial anterior de cómo añadir un botón en Android. Visitar el siguiente link para releerlo:  
<http://www.tutorialeshtml5.com/2012/02/tutorial-basico-android-agregar-boton.html>  
Una vez tengamos el Button creado el degradado lo tenemos que crear en un .xml en este ejemplo será degradado.xml el código es el siguiente, como siempre totalmente comentado:

<?xml version="1.0" encoding="utf-8"?>

*<!-- Creamos la forma que tendrá el degradado, en este caso un retangulo -->*

**<shape** xmlns:android="http://schemas.android.com/apk/res/android"

android:shape="rectangle"**>**

*<!-- Creamos el degradado añadiendo cada*

*color hexagesimal dondo orresponda*

*también se puede añadir un ángulo -->*

**<gradient** android:startColor="#954e00"

android:centerColor="#ead228"

android:endColor="#954e00"

android:angle="-90" **/>**

**</shape>**

El archivo .xml lo añadimos a la carpeta res/drawable/ de nuestro eclipse. Ahora para añadirlo a nuestro botón ya creado simplemente necesitaríamos la siguiente línea:

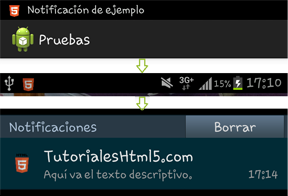
bt\_click.setBackgroundResource(R.drawable.gradient\_bronce);

De esa manera añadimos el degradado al botón, el resultado será el siguiente:

[](http://2.bp.blogspot.com/-eXz5c3MSVUg/UYEQf4yHrLI/AAAAAAAAAeY/9Oblsdn1lY0/s1600/Sin+t%C3%ADtuloxxxx.png)

NOTIFICACION

### Tutorial Android: Lanzar una notificación en barra de notificaciones de manera sencilla

[](http://4.bp.blogspot.com/-MmlISbMd_oY/UN3KF3rdB0I/AAAAAAAAAVg/sHLDQyFXnd8/s1600/tttt.png)

En este tutorial vamos a ver un ejemplo de cómo hacer saltar una notificación en la barra de notificaciones de nuestro dispositivo Android. Repito que es un simple ejemplo ya que esta parte puede ser muy amplia y compleja. para ver todo su contenido y perfecta implementación recomiendo visitar el post oficial: [http://goo.gl/yB9g5](https://developer.android.com/guide/topics/ui/notifiers/notifications.html) ya que hay un abanico de posibilidades.  
Aunque este ejemplo que he hecho sirve perfectamente para aprender su funcionamiento y depende del tipo de aplicación dónde se implementa ya bastará. En la imagen que tenéis a la izquierda podéis ver el resultado final.  
  
El código como siempre lo muestro a continuación completamente comentado. Para dudas o cualquier cosa que queráis no olvidéis escribir un comentario.

**public** void LanzamosNotificacion(){

*//Recogemos el servicio NOTIFICATION\_SERVICE encargado de mostrar*

*//objetos en la barra de notificación*

NotificationManager notificationManager = (NotificationManager)

getSystemService(Context.NOTIFICATION\_SERVICE);

*//Creamos los datos para la notificacion*

*//Icono a mostrar:*

int icon = R.drawable.logo;

*//Texto a mostrar en segundo plano*

CharSequence Text = "Aquí va el texto descriptivo.";

*//Texto que solo se mostrara al lanzar la notificacion*

CharSequence texto\_1 = "Notificación de ejemplo";

*//Texto a mostrar en primer plano*

CharSequence texto\_titulo = "TutorialesHtml5.com";

*//Que se muestre justo ahora la notificacion*

long when = System.currentTimeMillis();

*//Creamos el objeto de notificación*

Notification notification = **new** Notification(icon, texto\_1, when);

*//Creamos el Intent que lanzara cuando hagamos click en la notificacion*

Intent notificationIntent = **new** Intent(getApplicationContext()

,MainActivity.class);

*//Agregamos el Intent a un Penmding Intent*

*//Sobre esto deberéis leer algo más*

PendingIntent contentIntent = PendingIntent.getActivity(**this**,

0, notificationIntent, 0);

*//Añadimos los demás textos a la notificación*

notification.setLatestEventInfo(getApplicationContext(), texto\_titulo,

Text, contentIntent);

*//Lanzamos, el 10 es el identificador que le queramos*

*//poner a esta notificacion*

notificationManager.notify(10, notification);

}

El contenido de home\_menu.xml tiene la siguiente estructura:

***<?xml version="1.0" encoding="utf-8"?>***

***<menu xmlns:android="http://schemas.android.com/apk/res/android">***

***<item***

***android:id="@+id/btConfig"***

***android:title="Configuración"***

***android:icon="@drawable/config">***

***</item>***

***<item***

***android:id="@+id/btInfo"***

***android:title="Acerca de"***

***android:icon="@drawable/info">***

***</item>***

***</menu>***

**3. Añadir las opciones de menú dentro del Activity deseado**  
  
Este paso es bastante sencillo, sólo**debemos sobreescribir el método onCreateOptionsMenu**dentro del Activity que implemente el menú e indicarle el documento XML que contiene dichas opciones.

// Añadiendo las opciones de menú

      @Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

          MenuInflater inflater = getMenuInflater();

          inflater.inflate(R.menu.*home\_menu*, menu);

**return** **true**;

      }

|  |
| --- |
| Android: Desarrollo de opciones de menú en layouts (Activity) - Resultado final |
| *Vista final de la pantalla cuando se pulsa el botón MENÚ* |

**4. Dando funcionalidad a cada una de las opciones de menú**  
  
El código que muestra a continuación muestra un mensaje de texto diferente según que opción se pulsa, se deberá cambiar el contenido de cada case según la funcionalidad que queremos dar a la aplicación: Acceder a la base de datos para modificar algún registro, cambiar de Activity, etc.

// Añadiendo funcionalidad a las opciones de menú

@Override

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**switch** (item.getItemId()) {

**case** R.id.*btConfig*:

           Toast.*makeText*(getApplicationContext(), "Has pulsadoConfig",

                 Toast.*LENGTH\_SHORT*).show();

**return** **true**;

**case** R.id.*btInfo*:

           Toast.*makeText*(getApplicationContext(), "Has pulsado la opción

                 Info", Toast.*LENGTH\_SHORT*).show();

**return** **true**;

**default**:

**return** **super**.onOptionsItemSelected(item);

    }

}

# **Cómo admitir varios tamaños de pantalla**

[Anterior](https://developer.android.com/training/multiscreen/index.html)[Siguiente](https://developer.android.com/training/multiscreen/screendensities.html)

## **En esta sección puedes aprender:**

1. [Cómo utilizar los valores "wrap\_content" y "match\_parent"](https://developer.android.com/training/multiscreen/screensizes.html#TaskUseWrapMatchPar)
2. [Cómo utilizar RelativeLayout](https://developer.android.com/training/multiscreen/screensizes.html#TaskUseRelativeLayout)
3. [Cómo utilizar calificadores de tamaño](https://developer.android.com/training/multiscreen/screensizes.html#TaskUseSizeQuali)
4. [Cómo utilizar el calificador de ancho más pequeño](https://developer.android.com/training/multiscreen/screensizes.html#TaskUseSWQuali)
5. [Cómo utilizar alias de diseño](https://developer.android.com/training/multiscreen/screensizes.html#TaskUseAliasFilters)
6. [Cómo utilizar calificadores de orientación](https://developer.android.com/training/multiscreen/screensizes.html#TaskUseOriQuali)
7. [Cómo utilizar mapas de bits de la clase NinePatch](https://developer.android.com/training/multiscreen/screensizes.html#TaskUse9Patch)

## **También puedes consultar:**

* [Cómo admitir varias pantallas](https://developer.android.com/guide/practices/screens_support.html)

## **¡Pruébalo!**

[DESCARGAR LA APLICACIÓN DE EJEMPLO](http://developer.android.com/shareables/training/NewsReader.zip)

NewsReader.zip

En esta sección se explica cómo admitir varios tamaños de pantalla. Para ello, sigue estos pasos:

* Asegúrate de que el diseño se haya ajustado correctamente al tamaño de la pantalla.
* Configura la pantalla con el diseño de interfaz adecuado.
* Asegúrate de aplicar el diseño adecuado a la pantalla correspondiente.
* Utiliza el mapa de bits con la escala adecuada.

## **Cómo utilizar los valores "wrap\_content" y "match\_parent"**

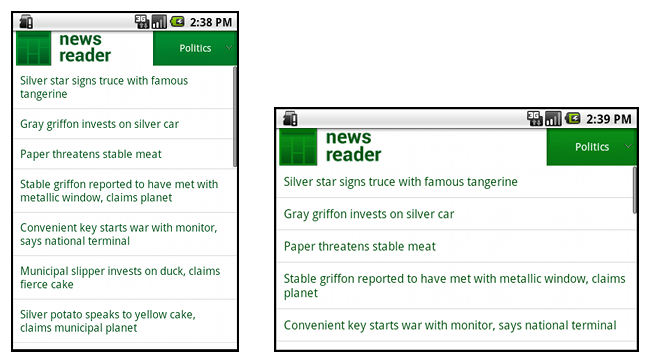
Para garantizar que el diseño es flexible y que se adapta a varios tamaños de pantalla, debes utilizar los valores "wrap\_content" y "match\_parent" para la altura y el ancho de algunos componentes de la vista. Si utilizas "wrap\_content", el ancho o la altura de la vista se establece en el tamaño mínimo necesario para adaptar el contenido a esta vista, mientras que "match\_parent" (también conocido como "fill\_parent" antes del nivel 8 del API) provoca que el componente se amplíe hasta coincidir con el tamaño de la vista principal.

Al utilizar los valores de tamaño "wrap\_content" y "match\_parent" en lugar de los tamaños predefinidos, las vistas pueden utilizar únicamente el espacio requerido para esa vista o ampliarse hasta rellenar el espacio disponible respectivamente. Por ejemplo:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <LinearLayout android:layout\_width="match\_parent"   
                  android:id="@+id/linearLayout1"    
                  android:gravity="center"  
                  android:layout\_height="50dp">  
        <ImageView android:id="@+id/imageView1"   
                   android:layout\_height="wrap\_content"  
                   android:layout\_width="wrap\_content"  
                   android:src="@drawable/logo"  
                   android:paddingRight="30dp"  
                   android:layout\_gravity="left"  
                   android:layout\_weight="0" />  
        <View android:layout\_height="wrap\_content"   
              android:id="@+id/view1"  
              android:layout\_width="wrap\_content"  
              android:layout\_weight="1" />  
        <Button android:id="@+id/categorybutton"  
                android:background="@drawable/button\_bg"  
                android:layout\_height="match\_parent"  
                android:layout\_weight="0"  
                android:layout\_width="120dp"  
                style="@style/CategoryButtonStyle"/>  
    </LinearLayout>  
  
    <fragment android:id="@+id/headlines"   
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="match\_parent" />  
</LinearLayout>

Observa cómo se utilizan en el ejemplo los valores "wrap\_content" y "match\_parent" para los tamaños de los componentes en lugar de dimensiones específicas. Esto permite que el diseño se adapte correctamente a diferentes tamaños y orientaciones de la pantalla.

Por ejemplo, esta es la apariencia del diseño en modo horizontal y vertical. Ten en cuenta que los tamaños de los componentes se adaptan automáticamente a la altura y al ancho:



**Figura 1.** La aplicación de ejemplo News Reader en modo vertical (izquierda) y horizontal (derecha)

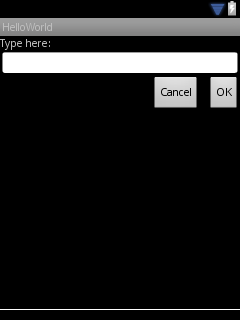
## **Cómo utilizar RelativeLayout**

Puedes crear diseños de un cierto nivel de complejidad con instancias anidadas de [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) y combinaciones de los valores de tamaño"wrap\_content" y "match\_parent". Sin embargo, [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) no te permite controlar con precisión las relaciones espaciales de las vistas secundarias; las vistas de [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html) simplemente se alinean en paralelo. Si quieres orientar las vistas secundarias de una forma que no sea una línea recta, a menudo la mejor solución es utilizar [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html)que te permite especificar el diseño según las relaciones espaciales entre los componentes. Por ejemplo, puedes alinear una vista secundaria en el lateral izquierdo y otra vista en el lateral derecho de la pantalla.

Por ejemplo:

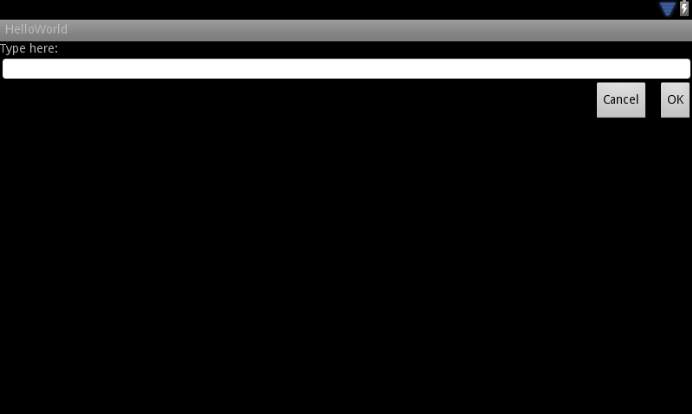
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <TextView  
        android:id="@+id/label"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:text="Type here:"/>  
    <EditText  
        android:id="@+id/entry"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:layout\_below="@id/label"/>  
    <Button  
        android:id="@+id/ok"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_below="@id/entry"  
        android:layout\_alignParentRight="true"  
        android:layout\_marginLeft="10dp"  
        android:text="OK" />  
    <Button  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_toLeftOf="@id/ok"  
        android:layout\_alignTop="@id/ok"  
        android:text="Cancel" />  
</RelativeLayout>

La figura 2 indica cómo se muestra este diseño en una pantalla QVGA.



**Figura 2.** Captura de pantalla de una pantalla QVGA (pantalla pequeña)

La figura 3 indica cómo se muestra este diseño en una pantalla más grande.



**Figura 3.** Captura de pantalla de una pantalla WSVGA (pantalla grande)

Ten en cuenta que aunque el tamaño de los componentes es diferente, las relaciones espaciales se mantienen según se ha especificado con[RelativeLayout.LayoutParams](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html).

## **Cómo utilizar calificadores de tamaño**

Hay mucha diferencia entre un diseño flexible y un diseño relativo como el que se ha utilizado en las secciones anteriores. Aunque ambos diseños se adaptan a diferentes pantalla estirando el espacio dentro de los componentes y alrededor de los mismos, puede que no ofrezcan la mejor experiencia de usuario para cada tamaño de pantalla. Por tanto, tu aplicación no solo debe implementar los diseños flexibles, sino que también debe ofrecer varios diseños alternativos para diferentes configuraciones de pantalla. Para ello, se utilizan [calificadores de configuración](http://developer.android.com/guide/practices/screens_support.html#qualifiers), que permiten que el tiempo de ejecución seleccione el recurso adecuado en función de la configuración actual del dispositivo (por ejemplo, un diseño diferente para diferentes tamaños de pantalla).

Por ejemplo, muchas aplicaciones implementan el patrón de "panel dual" para pantallas grandes (la aplicación mostraría una lista de elementos en un panel y el contenido en otro panel). Aunque los tablets y las televisiones son lo suficientemente grandes como para que los dos paneles aparezcan simultáneamente en la pantalla, las pantallas de los teléfonos tienen que mostrarlos por separado. Para implementar estos diseños, puedes utilizar los siguientes archivos:

* res/layout/main.xml, diseño de panel único (predeterminado):

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
  
    <fragment android:id="@+id/headlines"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="match\_parent" />  
</LinearLayout>

* res/layout-large/main.xml, diseño de panel dual:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent"  
    android:orientation="horizontal">  
    <fragment android:id="@+id/headlines"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="400dp"  
              android:layout\_marginRight="10dp"/>  
    <fragment android:id="@+id/article"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.ArticleFragment"  
              android:layout\_width="fill\_parent" />  
</LinearLayout>

Observa el calificador large en el nombre de directorio del segundo diseño. Este diseño se seleccionará en dispositivos con pantallas clasificadas como grandes (por ejemplo, tablets de 7" y superiores). El otro diseño (sin calificadores) se seleccionará en el caso de dispositivos más pequeños.

## **Cómo utilizar el calificador de ancho mínimo**

Una de las dificultades a las que se enfrentaron los desarrolladores con los dispositivos Android anteriores a la versión 3.2 fue el contenedor de tamaño de pantalla "grande". Algunos ejemplos de este tipo de dispositivo son el tablet Dell Streak, el tablet Galaxy Tab original y los tablets de 7" en general. Sin embargo, es posible que muchas aplicaciones quieran mostrar varios diseños para diferentes dispositivos de esta categoría (por ejemplo, para dispositivos de 5" y de 7"), aunque todos estos dispositivos se consideren de pantalla grande. Por esta razón, Android introdujo el calificador de "ancho mínimo" (entre otros) en Android 3.2.

Este calificador te permite mostrar contenido en pantallas que tengan un ancho mínimo determinado en píxeles independientes de la densidad. Por ejemplo, el tablet típico de 7" tiene un ancho mínimo de 600 dp, por lo que si quieres que la interfaz de usuario sea de panel dual en esta pantalla (y una única lista en pantallas más pequeñas), puedes utilizar los mismos dos diseños de la sección anterior para los diseños de panel único y de panel dual, solo que en lugar de utilizar el calificador de tamaño large, debes utilizar sw600dp para indicar que el diseño de panel dual se utiliza con las pantallas cuyo ancho mínimo sea de 600 dp:

* res/layout/main.xml, diseño de panel único (predeterminado):

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
  
    <fragment android:id="@+id/headlines"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="match\_parent" />  
</LinearLayout>

* res/layout-sw600dp/main.xml, diseño de panel dual:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent"  
    android:orientation="horizontal">  
    <fragment android:id="@+id/headlines"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="400dp"  
              android:layout\_marginRight="10dp"/>  
    <fragment android:id="@+id/article"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.ArticleFragment"  
              android:layout\_width="fill\_parent" />  
</LinearLayout>

Esto significa que los dispositivos cuyo ancho mínimo sea igual o superior a 600 dp utilizarán el diseño layout-sw600dp/main.xml (panel dual), mientras que las pantallas más pequeñas utilizarán el diseño layout/main.xml (panel único).

No obstante, esto no funcionará en los dispositivos anteriores a Android 3.2 porque no reconocen sw600dp como calificador de tamaño, por lo que también tendrás que seguir utilizando el calificador large. Por tanto, debes tener un archivo con el nombre res/layout-large/main.xml idéntico ares/layout-sw600dp/main.xml. En la siguiente sección, obtendrás información sobre una técnica que te permite evitar que se dupliquen los archivos de diseños.

## **Cómo utilizar alias de diseño**

El calificador de ancho mínimo solo está disponible en Android 3.2 o superior. Por tanto, tendrás que seguir utilizando los contenedores de tamaño abstractos (pequeño, normal, grande y extragrande) para que sean compatibles con versiones anteriores. Por ejemplo, si quieres que tu interfaz de usuario sea de panel único en teléfonos pero multipanel en tablets de 7", televisiones y otros dispositivos grandes, tendrás que utilizar los siguientes archivos:

* res/layout/main.xml: diseño de panel único,
* res/layout-large: diseño multipanel,
* res/layout-sw600dp: diseño multipanel.

Los dos últimos archivos son idénticos porque uno de ellos se utilizará con dispositivos Android 3.2 y el otro con tablets y televisiones que utilicen versiones anteriores de Android.

Para evitar la duplicación del mismo archivo para tablets y televisiones (así como todos los problemas que esto conlleva), puedes utilizar archivos alias. Por ejemplo, puedes establecer los siguientes diseños:

* res/layout/main.xml: diseño de panel único,
* res/layout/main\_twopanes.xml: diseño de panel dual.

Añade estos dos archivos:

* res/values-large/layout.xml:

<resources>  
    <item name="main" type="layout">@layout/main\_twopanes</item>  
</resources>

* res/values-sw600dp/layout.xml:

<resources>  
    <item name="main" type="layout">@layout/main\_twopanes</item>  
</resources>

Estos dos últimos archivos tienen el mismo contenido, pero en realidad no definen el diseño. Solo configuran main para que sea un alias demain\_twopanes. Como los archivos tienen selectores large y sw600dp, se aplican a tablets y a televisiones independientemente de la versión de Android (las televisiones y los tablets anteriores a la versión 3.2 utilizarán largey las televisiones y los tablets posteriores a la versión 3.2 utilizarán sw600dp).

## **Cómo utilizar calificadores de orientación**

Aunque algunos diseños se pueden utilizar tanto en modo horizontal como vertical, la mayoría de ellos pueden beneficiarse de los ajustes. A continuación, se indica cómo se comporta el diseño según cada tamaño y orientación de la pantalla en la aplicación de ejemplo News Reader:

* **pantalla pequeña, vertical:** panel único con logotipo,
* **pantalla pequeña, horizontal:** panel único con logotipo,
* **tablet de 7", vertical:** panel único con barra de acciones,
* **tablet de 7", horizontal:** panel dual ancho con barra de acciones,
* **tablet de 10", vertical:** panel dual estrecho con barra de acciones,
* **tablet de 10", horizontal:** panel dual ancho con barra de acciones,
* **televisión, horizontal:** panel dual ancho con barra de acciones.

Cada uno de estos diseños se establecen en un archivo XML en el directorio res/layout/. Para definir posteriormente las diferentes configuraciones de pantalla, la aplicación utiliza alias de diseño para asignarlos a cada configuración:

res/layout/onepane.xml:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
  
    <fragment android:id="@+id/headlines"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="match\_parent" />  
</LinearLayout>

res/layout/onepane\_with\_bar.xml:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <LinearLayout android:layout\_width="match\_parent"   
                  android:id="@+id/linearLayout1"    
                  android:gravity="center"  
                  android:layout\_height="50dp">  
        <ImageView android:id="@+id/imageView1"   
                   android:layout\_height="wrap\_content"  
                   android:layout\_width="wrap\_content"  
                   android:src="@drawable/logo"  
                   android:paddingRight="30dp"  
                   android:layout\_gravity="left"  
                   android:layout\_weight="0" />  
        <View android:layout\_height="wrap\_content"   
              android:id="@+id/view1"  
              android:layout\_width="wrap\_content"  
              android:layout\_weight="1" />  
        <Button android:id="@+id/categorybutton"  
                android:background="@drawable/button\_bg"  
                android:layout\_height="match\_parent"  
                android:layout\_weight="0"  
                android:layout\_width="120dp"  
                style="@style/CategoryButtonStyle"/>  
    </LinearLayout>  
  
    <fragment android:id="@+id/headlines"   
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="match\_parent" />  
</LinearLayout>

res/layout/twopanes.xml:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent"  
    android:orientation="horizontal">  
    <fragment android:id="@+id/headlines"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="400dp"  
              android:layout\_marginRight="10dp"/>  
    <fragment android:id="@+id/article"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.ArticleFragment"  
              android:layout\_width="fill\_parent" />  
</LinearLayout>

res/layout/twopanes\_narrow.xml:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="fill\_parent"  
    android:orientation="horizontal">  
    <fragment android:id="@+id/headlines"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.HeadlinesFragment"  
              android:layout\_width="200dp"  
              android:layout\_marginRight="10dp"/>  
    <fragment android:id="@+id/article"  
              android:layout\_height="fill\_parent"  
              android:name="com.example.android.newsreader.ArticleFragment"  
              android:layout\_width="fill\_parent" />  
</LinearLayout>

Una vez que se hayan definido todos los diseños posibles, solo se debe asignar el diseño adecuado a cada configuración a través de calificadores de configuración. Ahora ya puedes utilizar la técnica de los alias de diseño:

res/values/layouts.xml:

<resources>  
    <item name="main\_layout" type="layout">@layout/onepane\_with\_bar</item>  
    <bool name="has\_two\_panes">false</bool>  
</resources>

res/values-sw600dp-land/layouts.xml:

<resources>  
    <item name="main\_layout" type="layout">@layout/twopanes</item>  
    <bool name="has\_two\_panes">true</bool>  
</resources>

res/values-sw600dp-port/layouts.xml:

<resources>  
    <item name="main\_layout" type="layout">@layout/onepane</item>  
    <bool name="has\_two\_panes">false</bool>  
</resources>

res/values-large-land/layouts.xml:

<resources>  
    <item name="main\_layout" type="layout">@layout/twopanes</item>  
    <bool name="has\_two\_panes">true</bool>  
</resources>

res/values-large-port/layouts.xml:

<resources>  
    <item name="main\_layout" type="layout">@layout/twopanes\_narrow</item>  
    <bool name="has\_two\_panes">true</bool>  
</resources>

## **Cómo utilizar mapas de bits de la clase NinePatch**

Admitir diferentes tamaños de pantalla normalmente implica que las fuentes de imagen también deben poder adaptarse a varios tamaños. Por ejemplo, un fondo de botón debe adaptarse a cualquier forma de botón a la que se aplique.

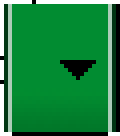
Si utilizas imágenes sencillas en componentes que pueden cambiar de tamaño, observarás rápidamente que los resultados no es que sean precisamente impresionantes, ya que las imágenes se estirarán o estrecharán. La solución es utilizar mapas de bits de la clase NinePatch, que son archivos PNG con un formato especial que indican las áreas que se pueden y no se pueden estirar.

Por tanto, al diseñar mapas de bits que se vayan a utilizar en componentes con tamaño variable, utiliza siempre mapas de bits de la clase NinePatch. Para convertir un mapa de bits en uno de la clase NinePatch, puedes empezar con una imagen normal (consulta la figura 4, que se ha ampliado cuatro veces para obtener una mayor claridad).



**Figura 4.** button.png

A continuación, puedes pasar a la utilidad draw9patch del SDK (que se localiza en el directorio tools/) en la que puedes marcar las áreas que se deben estirar dibujando píxeles a lo largo de los bordes superior e izquierdo. También puedes marcar el área que debe incluir el contenido dibujando píxeles a lo largo de los bordes inferior y derecho, como se muestra en la figura 5.

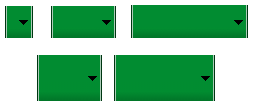


**Figura 5.** button.9.png

Observa los píxeles de color negro situados junto a los bordes. Los que aparecen en los bordes superior e izquierdo indican los lugares en los que se puede estirar la imagen, mientras que los que aparecen en los bordes inferior y derecho indican dónde se debe situar el contenido.

Además, observa la extensión .9.png. Debes utilizar esta extensión, ya que, de este modo, el marco detecta que se trata de una imagen de la clase NinePatch, en lugar de una imagen PNG normal.

Cuando aplicas este fondo a un componente (definiendo android:background="@drawable/button"), el marco estira la imagen de forma adecuada para adaptarla al botón, como se muestra en varios tamaños de la figura 6.



**Figura 6.** Botón que utiliza la clase NinePatch button.9.png en varios tamaños