# Cómo admitir varios tamaños de pantalla

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.

**EN ESTA SECCIÓN PUEDES APRENDER:**



**TAMBIÉN PUEDES CONSULTAR:**

- Cómo admitir varias pantallas

**¡PRUÉBALO!**

Descargar la aplicación de ejemplo

NewsReader.zip

## 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:

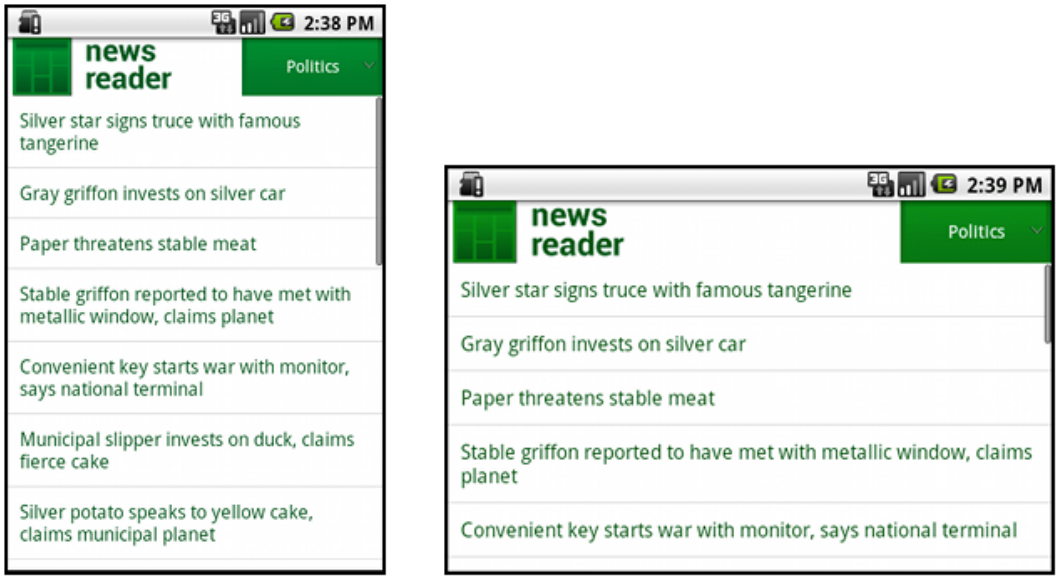


Figura 1. La aplicación de ejemplo News Reader en modo vertical (izquierda) y horizontal (derecha)

## Cómo utilizar RelativeLayout

Puedes crear diseños de un cierto nivel de complejidad con instancias anidadas de `LinearLayout (/reference/android/widget/LinearLayout.html)` y combinaciones de los valores de tamaño `"wrap_content"` y `"match_parent"`. Sin embargo, `LinearLayout (/reference/android/widget/LinearLayout.html)` no te permite controlar con precisión las relaciones espaciales de las vistas secundarias; las vistas de `LinearLayout (/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 (/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` (/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 a `res/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 de `main_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 `large`y 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.

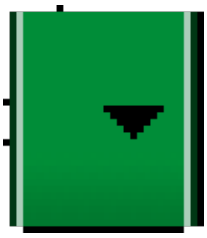

Figura 5. `button.9.png`

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.

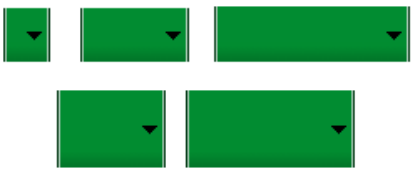

Figura 6. Botón que utiliza la clase NinePatch `button.9.png` en varios tamaños