14. Otros contenedores

Los contenedores determinan la estructura visual de las actividades pues cada uno tiene una forma concreta de colocar en su interior a las vistas que lo componen.

Los contenedores extienden directa o indirectamente la clase `ViewGroup`, que es una vista especial (`ViewGroup` extiende `View`) que puede contener a otras en su interior. Estos son algunos de los contenedores que tienes a tu disposición en Android:

- `AbsoluteLayout`: cada hijo es asignado a una posición fija dentro del contenedor.
- `AdapterView`: los hijos que pueblan este contenedor se obtienen dinámicamente de un adaptador. Por ejemplo, `ListView` muestra a sus hijos en una lista vertical desplazable.
- `DrawerLayout`: permite extraer a sus hijos (cajones) desde los lados de la ventana.
- `FrameLayout`: sirve para reservar una zona de la pantalla para un solo hijo.
- `GridLayout`: organiza a sus hijos dentro de una red rectangular de forma flexible, es decir, sin necesidad de especificar elementos intermedios como filas, por ejemplo.
- `LinearLayout`: organiza a sus hijos en una sola fila o columna.
- `RelativeLayout`: la posición de los hijos se especifica en relación a la de los demás o la del contenedor.
- `ScrollView`: permite desplazar (*scroll*) a su único hijo, que suele ser otro contenedor, y así poder mostrar información que no cabe en una sola pantalla.
- `TableLayout`: organiza a sus hijos en una tabla mediante elementos de tipo `TableRow`.
- `ViewPager`: organiza páginas de datos entre las que nos podemos desplazar a la izquierda o a la derecha.
- `WebView`: permite mostrar páginas web.

Aunque no es muy frecuente, dada la gran variedad de contenedores que existen, también tienes la posibilidad de desarrollar tus propios contenedores extendiendo la clase `ViewGroup`.

Esta unidad viene acompañada por cinco proyectos (`CCC14_1`, …, `CCC14_5`) que ilustran algunos de estos contenedores. Solo el segundo (`CCC14_2`) contiene código Java no trivial. El resto de los proyectos cuentan con un fichero Java como los utilizados hasta ahora, que básicamente infla la interfaz especificada en el fichero de diseño correspondiente (`activity_main.xml`).

14.1 AbsoluteLayout

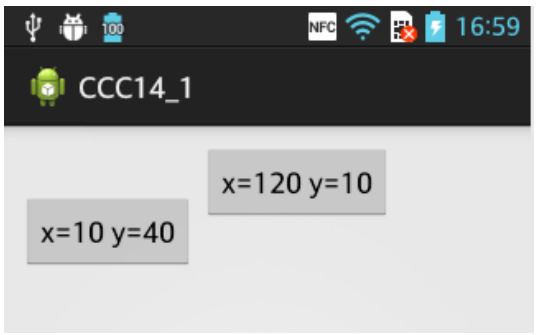
Los elementos de este contenedor se sitúan especificando sus posiciones X e Y exactas mediante los atributos `android:layout_x` y `android:layout_y`, respectivamente:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="10dip"
    android:layout_y="40dip"/>
```

Se trata de un contenedor no recomendable pues las interfaces no se adaptan adecuadamente al cambiar la resolución o el tamaño del dispositivo. Veamos un ejemplo sencillo:

`/res/layout/activity_main.xml`

```
<AbsoluteLayout 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"
    tools:context=".Main" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="10dip"
        android:layout_y="40dip"
        android:text="@string/x_10_y_40" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="120dip"
        android:layout_y="10dip"
        android:text="@string/x_120_y_10" />

</AbsoluteLayout>
```



14.2 FrameLayout

Este contenedor presenta un solo elemento en su lado superior izquierdo. Cualquier otra vista que añadamos en su interior se superpondrá a la anterior. Sin embargo, también se puede utilizar para alternar entre elementos de forma dinámica. Por ejemplo, el siguiente proyecto permite alternar entre dos imágenes sin mas que pulsar sobre ellas.

Necesitamos introducir una vista nueva de tipo `ImageView`, que se utiliza para mostrar imágenes en Android. El fichero con la imagen se debe colocar en las carpetas de recursos, `drawable`, y asociarlas al atributo `android:src` como a continuación:

```
<ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/img0169"
        android:scaleType="fitCenter" />
```

Colocaremos dos imágenes en la carpeta `drawable` de nombre `circulo.png` y `cuadrado.png`, respectivamente. En el siguiente fichero de diseño se añaden dos elementos de tipo `ImageView` al `FrameLayout`, uno por cada imagen:

`/res/layout/activity_main.xml`

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <ImageView
        android:id="@+id/circulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/circulo"
        android:onClick="circulo"
        android:scaleType="fitCenter"
        android:src="@drawable/circulo" />

    <ImageView
        android:id="@+id/cuadrado"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/cuadrado"
        android:onClick="cuadrado"
        android:scaleType="fitCenter"
        android:src="@drawable/cuadrado" />

</FrameLayout>
```

El atributo `android:scaleType` sirve para escalar la imagen dentro de la vista. Sus posibles valores incluyen:

- `fitXY`: escala independientemente cada dimensión para ajustarla al tamaño de la vista sin mantener la proporción original.
- `center`: centra la imagen en el contenedor sin escalarla.
- `centerInside`: escala la imagen proporcionalmente para ajustarla al tamaño de la vista.
- `fitCenter`: escala la imagen proporcionalmente de tal forma que o bien su anchura o su altura se ajustan al tamaño de la vista.

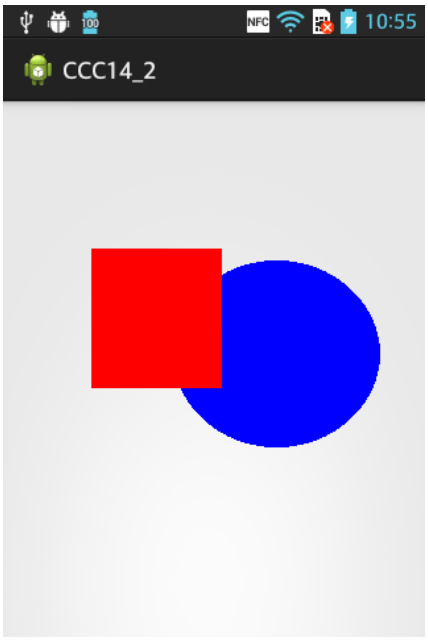
Los valores asignados al atributo `android:onClick` de cada vista garantizan que cuando pulsemos la primera imagen se ejecutará el método `circulo()`, y cuando pulsemos la segunda, se ejecutará el método `cuadrado()`. El código de ambos métodos se ha añadido en el fichero `MainActivity.java`:

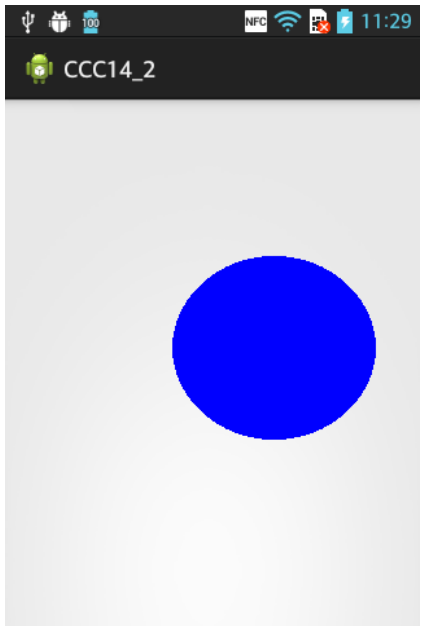
`/src/MainActivity.java`

```
package es.uam.eps.dadm.ccc14_2;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.ImageView;

public class MainActivity extends Activity {
     ImageView primera;
     ImageView segunda;

     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
         primera = (ImageView) findViewById(R.id.circulo);
```

```
        segunda = (ImageView) findViewById(R.id.cuadrado);
    }

    public void circulo (View view){
       segunda.setVisibility(View.VISIBLE);
       primera.setVisibility(View.GONE);
    }

    public void cuadrado (View view){
       primera.setVisibility(View.VISIBLE);
       segunda.setVisibility(View.GONE);
    }
}
```

En el método `onCreate()` se consiguen referencias a cada una de las vistas, para poder actuar sobre su visibilidad dentro de los métodos `circulo()` y `cuadrado()`. Inicialmente las dos imágenes se ven superpuestas pero, al pulsar por primera vez, se ejecuta el método `cuadrado()`, con lo que se verá solo el círculo azul (a la derecha):





14.3 GridLayout

Este contenedor permite organizar los elementos en filas y columnas sin más que especificar la fila y columna donde se desea colocar cada elemento. Por ejemplo, el siguiente elemento de tipo `EditText` se situará en la primera fila y segunda columna:

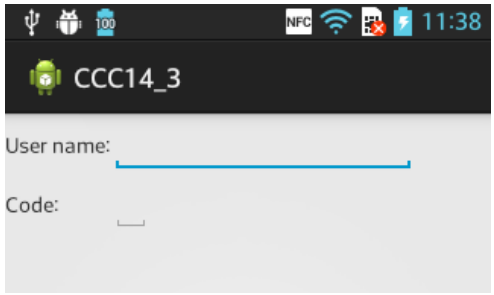
```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="0"
    android:layout_column="1"
    android:inputType="textPersonName"
```

```
        android:minWidth="200dip"/>
```

Los elementos de tipo `EditText` permiten que el usuario introduzca texto. El atributo `inputType` permite especificar el tipo de contenido básico del campo de texto. Los elementos de tipo `TextView` sirven para mostrar mensajes. El siguiente fichero de diseño genera una interfaz para que los usuarios tecleen su nombre y código:

/res/layout/activity_main.xml

```
<GridLayout 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"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_row="0"
        android:layout_column="0"
        android:text="@string/user_name" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_row="0"
        android:layout_column="1"
        android:inputType="textPersonName"
        android:minWidth="200dip"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_row="1"
        android:layout_column="0"
        android:text="@string/code" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_row="1"
        android:layout_column="1"
        android:inputType="textPassword"/>
</GridLayout>
```

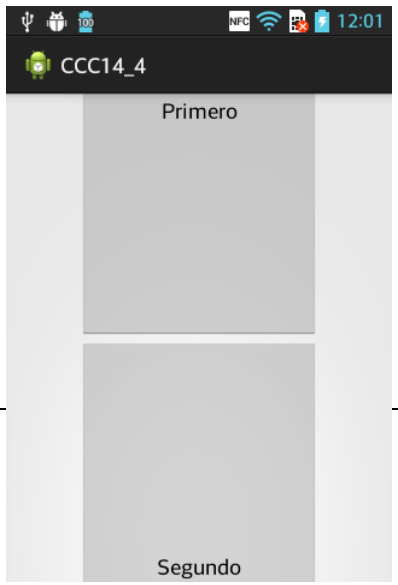


14.4 ScrollView

Este elemento permite mostrar contenido que no cabe en una sola pantalla mediante la utilización de una barra vertical de scroll. Solo puede contener un hijo, que suele ser otro contenedor como un `LinearLayout`, que a su vez contiene al resto de las vistas que deseamos acomodar en la pantalla. El siguiente ejemplo permite acomodar dos botones que se han dimensionado con una altura de 380dip, que no caben en una sola pantalla:

/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center">
        <Button
            android:layout_width="200dip"
            android:layout_height="400dip"
            android:text="Primero" />
        <Button
            android:layout_width="200dip"
            android:layout_height="400dip"
            android:text="Segundo" />
    </LinearLayout>
</ScrollView>
```



14.5 TableLayout

El contenedor `TableLayout` organiza a sus hijos en filas y columnas. Cada fila se define mediante un elemento `TableRow` que puede tener 0 o más elementos. El número de columnas del contenedor es el máximo número de elementos en una fila.

La anchura de una columna es igual a la del elemento más ancho de esa columna. Sin embargo, se puede utilizar el atributo `android:stretchColumns` para especificar que ciertas columnas aprovecharán el espacio libre disponible. Por ejemplo, la siguiente instrucción especifica que lo harán las columnas primera y tercera:

```
android:stretchColumns="0,2"
```

Si en lugar de especificar el número de las columnas ponemos un asterisco, todas las columnas se ensanchan:

```
android:stretchColumns="*"
```

A continuación se muestra un fichero de diseño que reproduce la interfaz de la sección 14.3 mediante un `TableLayout` en lugar de un `GridLayout`:

/res/layout/activity_main.xml

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/user_name" />
        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:minWidth="200dip"
            android:inputType="textPersonName"/>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/code" />
        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="textPassword"/>
    </TableRow>
</TableLayout>
```