8. La primera interfaz de usuario

Una aplicación Android es básicamente un conjunto de actividades. Cada actividad se implementa en una clase Java derivada de la clase base `Activity` de Android y, en general, mediante un fichero XML de diseño.

8.1 El fichero XML de diseño

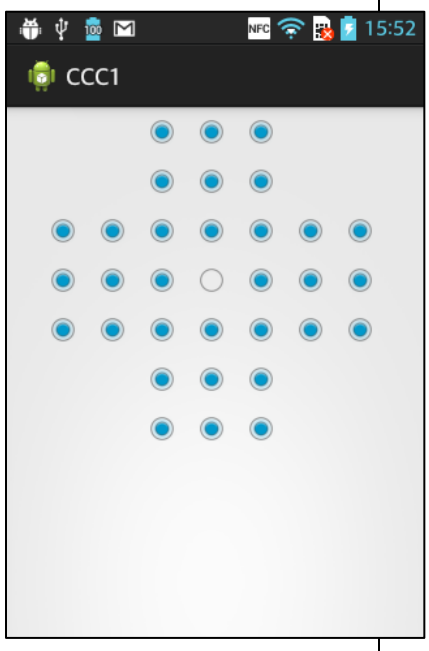
Las actividades interactúan con los usuarios a través de su interfaz de usuario. Esta interfaz se implementa con objetos de las clases `View` y `ViewGroup`, que podemos traducir como vistas y contenedores, respectivamente.

Por ejemplo, en la interfaz de nuestro juego vamos a utilizar vistas de tipo `RadioButton` y contenedores de tipo `LinearLayout`. Este tipo de contenedor organiza su contenido (los botones) en una sola fila o columna. Veremos mas adelante que existen otros contenedores como, por ejemplo, `TableLayout`, para organizar vistas de forma tabular.

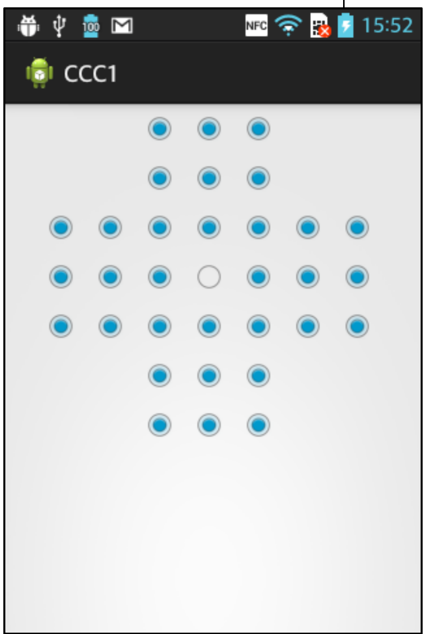
La implementación de la interfaz de usuario puede hacerse en Java o mediante un fichero XML. En nuestro primer programa, la interfaz se especifica en el siguiente fichero XML:

`/res/layout/activity_main.xml`

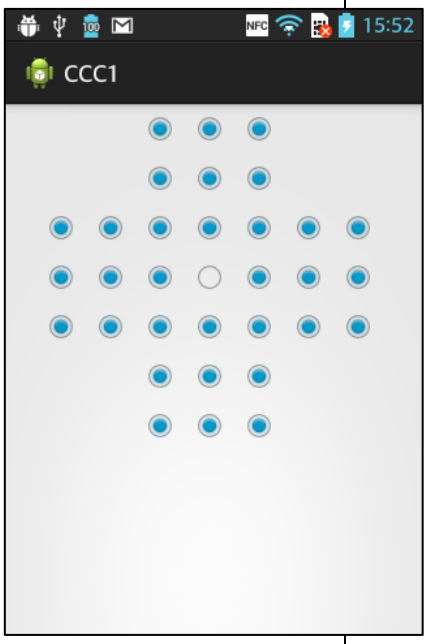
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center_horizontal">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
```



```
            android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="false"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <RadioButton
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
```



```
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
        <RadioButton
              android:checked="true"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
    </LinearLayout>
</LinearLayout>
```



La etiqueta `<LinearLayout>` especifica un elemento de tipo `LinearLayout`. Este elemento es un contenedor que organiza su contenido en una sola fila o en una sola columna. Sus atributos se estudian a continuación.

El atributo `android:layout_width` especifica la anchura de la vista. Su valor puede ser una dimensión (tal y como "40dip") o una de las siguientes constantes:

- `fill_parent`: La vista será tan ancha como el padre, menos el padding del padre si es que tiene. El padding se especifica mediante el atributo `android:padding` como veremos mas adelante.
- `match_parent`: La vista será tan ancha como el padre, menos el padding del padre si es que tiene. Este valor remplaza a `fill_parent` desde el nivel API 8.

- `wrap_content`: La vista será lo suficientemente ancha para contener su propio contenido más padding.

El atributo `android:layout_height` especifica la altura de la vista. Su valor puede ser una dimensión (tal y como "40dip") o una de las constantes de la tabla de arriba (donde se ha de sustituir anchura por altura).

Los atributos `android:layout_with` y `android:layout_height` se utilizan tanto en los contenedores `LinearLayout` como en las vistas `RadioButton`. Todos los elementos utilizan el valor `wrap_content` salvo el contenedor principal, que utiliza `match_parent`.

El atributo `android:orientation` tiene el valor `vertical`, es decir, los hijos se organizan en una columna. En este caso, el elemento `LinearLayout` principal contiene siete hijos de tipo `LinearLayout` dispuestos verticalmente, uno por cada fila del tablero de nuestro juego. A su vez, cada uno de estos hijos contiene botones en fila, ya que la orientación por defecto es `horizontal`. El resultado final es siete filas de botones, con tres o siete botones por fila dependiendo de la posición.

El atributo `android:gravity` del contenedor principal tiene el valor `center_horizontal`, lo cual garantiza que sus hijos, es decir, cada una de las filas del tablero, quedarán centradas en la pantalla.

La etiqueta `<RadioButton>` especifica un elemento de tipo `RadioButton`, que puede ser pulsado por el usuario. Estos elementos van a jugar el papel de las fichas de nuestro juego. El atributo `android:checked` especifica el estado inicial del botón, que puede ser pulsado (`true`) o no (`false`). El único botón sin pulsar es el que ocupa la posición central del tablero para simular la ausencia de ficha en esa posición al iniciar el juego.

Esto completa el análisis del fichero de diseño. Echemos un vistazo al fichero Java.

8.2 El fichero Java

El fichero `MainActivity.java` contiene la definición de la clase `MainActivity` que extiende la clase `Activity` de Android. En nuestro caso, solo se sobrescribe el método `onCreate()` de `Activity`. Este método es uno de los métodos del ciclo de vida de la actividad. No se llama explícitamente, como todos los métodos que empiezan por `on`, sino que se ejecuta automáticamente en un cierto momento de la vida de la actividad.

Concretamente, `onCreate()` se ejecuta una vez instanciada la actividad pero cuando todavía no se ha mostrado en la pantalla. El código es el siguiente:

/src/MainActivity.java

```
package es.uam.eps.android.ccc8;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {

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

}
```

La anotación `@override` hace que el compilador compruebe que el método que se está sobrescribiendo efectivamente existe en la superclase. De esta forma se evitan errores como el que ocurre si intentamos implementar el método:

```
protected void onCreates(Bundle savedInstanceState) {
        ...
}
```

Dentro del método, lo primero que hacemos es llamar a la versión de la superclase, lo cual es obligatorio. Además, ha de ser la primera instrucción del método sobrescrito:

```
super.onCreate(savedInstanceState);
```

Como ves, se pasa el objeto recibido de tipo `Bundle` al método de la superclase. El objeto de tipo `Bundle` contiene el estado de las vistas de la interfaz almacenado en forma de pares clave-valor. Este objeto sirve para que la actividad guarde y posteriormente recree su interfaz gráfica. Como veremos, el programador puede añadir información a este objeto en forma de pares clave-valor extra, con el objetivo de restaurar información adicional cuando se infle la interfaz.

A continuación llamamos al método `setContentView()`, que infla la interfaz especificada por su argumento, que es un identificador de recurso. El identificador `R.layout.activity_main` representa el fichero `activity_main.xml` que acabamos de estudiar en la sección anterior.

Recuerda que la clase `R` es una clase constante generada automáticamente y que contiene valores enteros organizados en subclases como, por ejemplo, `id` y `layout`. Así, dentro de `layout` se sitúan los identificadores de los ficheros XML de diseño como `activity_main.xml` y, en el interior de `id`, se encuentran los identificadores de las vistas de la interfaz.