Tema 5. Diseño de la interfaz

Prof. Manuel Castillo

Programación de Dispositivos Móviles

Escuela Profesional de Ciencias de la Computación

Facultad de Ciencias

Universidad Nacional de Ingeniería

Objetivos

- Conocer las vistas y objetos típicos de un formulario.
- Saber cómo implementar los controles sobre un formulario.
- Saber cómo utilizar un ListView.
- Conocer las mejoras sobre un ListActivity.

Índice de contenido

- Entendiendo la Interfaz de Usuario de Android
- Layouts Típicas
- Views y Eventos de Usuario
- Colecciones de datos y Views
- Menús
- Estilos y Temas

Introducción

- El interfaz de usuario es la parte de la Vista dentro del MVC.
- Toda clase está basada en *View*.
  - Debido a la herencia toda vista tiene características comunes.
- Toda Vista está asociada a una etiqueta del *XML*.
  - Las propiedades de la clase están asociadas a parámetro de la etiqueta *XML*.
- Los ficheros de Layout se encuentran en */res/layout*.

Conceptos previos I: Identificador

- Nombre único asignado
  - *android:id="@+id/nombre_identificador"*
    - android:id: nombre del atributo.
    - @+: indica la declaración de un nuevo identificador.
    - id: corresponde a la categoría del identificador.
    - nombre_identificador.
- Acceso al identificador:
  - Java: R.id.nombre_identificador
  - XML: @id/nombre_identificador

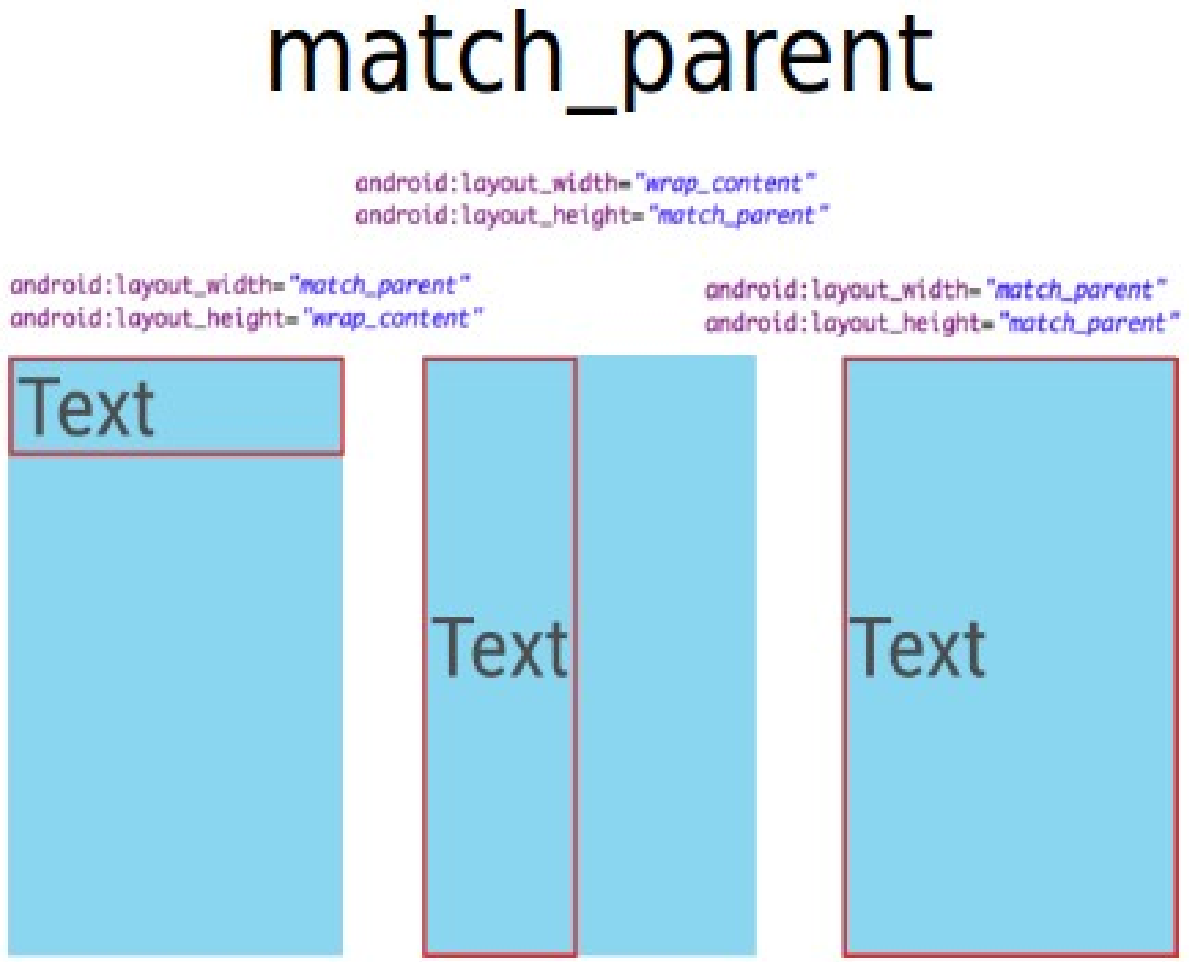
Conceptos previos II: Combinar actividades

- Heredar de la clase Java (*AppActivity*).
- Sobrecargar al menos al método *onCreate*.
- Enlazar la actividad con la interfaz mediante el método *setContentView*.
- Declarar nuevas “Activities” en el Manifesto.

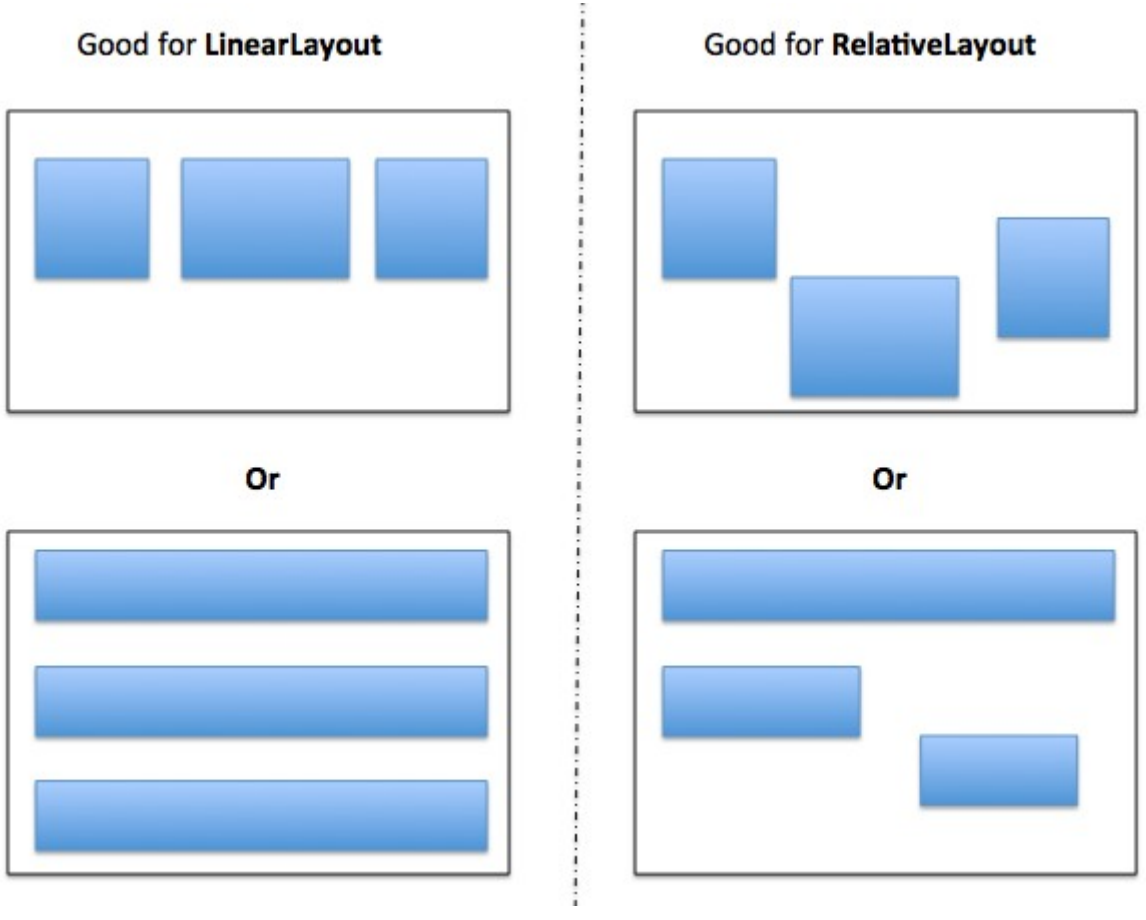
Conceptos previos III: Tamaño de elementos

- Todo *layout* tiene su:
  - Altura: *android:height*.
  - Anchura: *android:width*.
- Valores:
  - **match_parent** (anteriormente fill_parent): significa que el tamaño del elemento es igual al del elemento padre.
    - Por ejemplo, ocupará el mismo espacio que su contenedor.
  - **wrap_content**: significa que el tamaño del elemento es igual al de su contenedor.
    - Por ejemplo, tendrá el tamaño de la suma del tamaño de su contenido más el de los diferentes espacios internos.
  - Especificando un valor: puede definir el tamaño de un elemento con constantes.
- Hay que especificar el tamaño de los elementos en dp (density-independent pixels) y no en px.
  - dp conservan las mismas proporciones sea cual sea la densidad de pantalla.

Conceptos previos III: Tamaño de elementos (II)



1. Layouts típicas



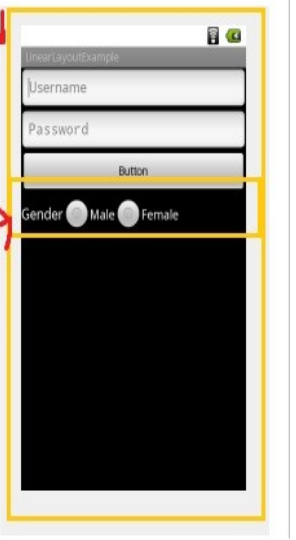
1.1. LinearLayout

- LinearLayout es el layout más utilizado en la práctica.
- Distribuye los elementos uno detrás de otro, bien de forma horizontal o vertical.
- Tiene una orientación vertical y horizontal.

```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" >
    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Username" >
        <requestFocus />
    </EditText>
    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Password"
        android:inputType="textPassword" />
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Gender"
            android:textAppearance="?android:attr/textAppearanceMedium" />
        <RadioButton
            android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male" />
        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female" />
    </LinearLayout>
</LinearLayout>
```

Vertical Layout

Horizontal Layout



1.1. Atributos (I): Orientación

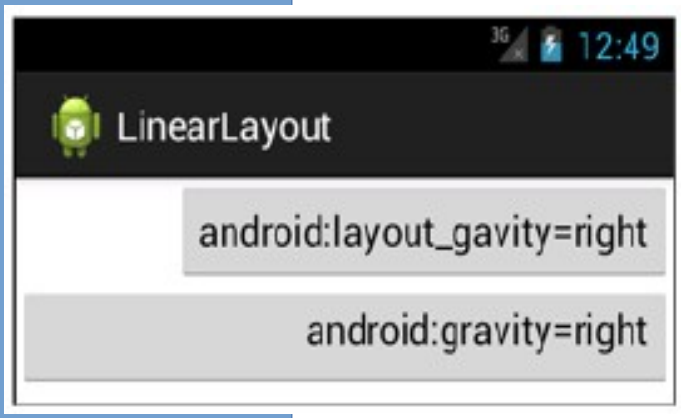
- Orientación: *android:orientation*
  - Verticalmente, los unos debajo de los otros, un único elemento por línea.
  - Horizontalmente, los unos a continuación de los otros, a la derecha del anterior.
- Por defecto, horizontal

1.1. Atributos (II): Gravedad

- Gravedad o posicionamiento:
  - ***layout_gravity***: especifica el posicionamiento de un elemento en su contenedor.
  - ***gravity***: especifica el posicionamiento del contenido de un elemento (por ejemplo, se puede especificar la posición de un texto en un botón).

1.1. Atributos (II): Gravedad (*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" >
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/layout_gravity"
        android:layout_gravity="right" />
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/gravity"
        android:gravity="right" />
</LinearLayout>
```



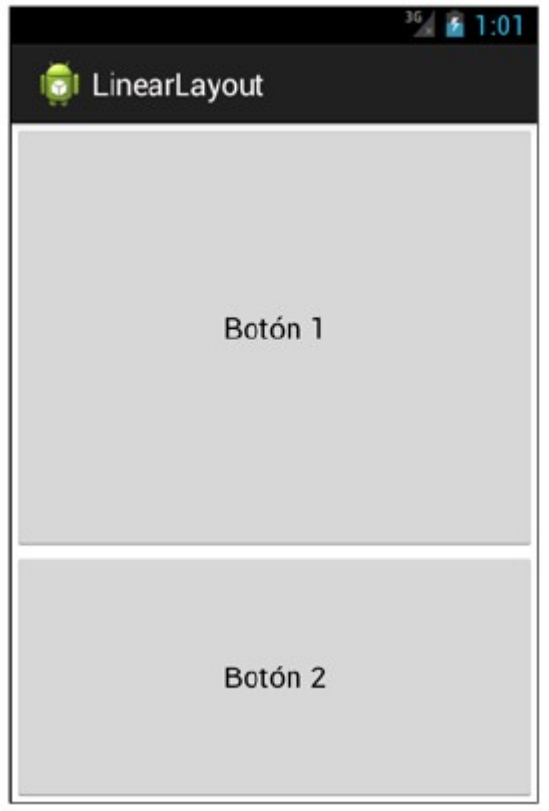
1.1. Atributos (II): Peso de los elementos

- Indicar a un elemento el espacio que puede ocupar.
- Cuanto mayor sea el peso de un elemento, más se podrá extender un componente y ocupar el espacio disponible.
- Por defecto 0

1.1. Atributos (II): Gravedad (*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" >
    <Button
        android:layout_width="0px"
        android:layout_height="match_parent"
        android:text="@string/btn1"
        android:layout_weight="2" />
    <Button
        aandroid:layout_width="0px"
        android:layout_height="match_parent"
        android:text="@string/btn2"
        android:layout_weight="1" />
</LinearLayout>
```

Botón 1 es el doble de 2



1.2. RelativeLayout

- Dispone las vistas en parámetros absolutos o relativos.
- Tiene más flexibilidad a la hora de colocar las vistas.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Enter email address" />
    <EditText
        android:id="@+id/inputEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label" />
    <Button
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/inputEmail"
        android:layout_marginRight="10px"
        android:text="Login" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/btnLogin"
        android:layout_toRightOf="@id/btnLogin"
        android:text="Cancel" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="Register" />
</RelativeLayout>
```

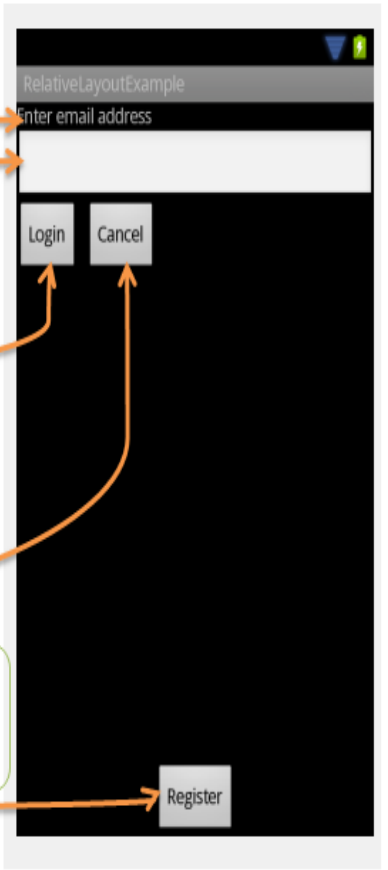
TextView with normal properties

EditView with normal properties

Button aligned left to the parent and also below the inputEmail EditView control

Button aligned at top of the parent and also right to the btnLogin

Button aligned at bottom of the parent and also center horizontally to the parent



1.2.1. Posicionamiento relativo al contenedor

- Puede posicionar un elemento en función de los bordes del contenedor.
- Utilizar uno o varios de los atributos siguientes con un valor booleano true/false (verdadero/falso).
  - *android:layout_alignParentTop*: alinear el elemento con el borde superior del contenedor.
  - *android:layout_alignParentBottom*: alinear el elemento con el borde inferior del contenedor.
  - *android:layout_alignParentLeft*: alinear el elemento con el borde izquierdo del contenedor.
  - *android:layout_alignParentRight*: alinear el elemento con el borde derecho del contenedor.
- Puede combinar varios valores de posicionamiento.

1.2.2. Posicionamiento relativo a otros elementos

- Dispone de nueve opciones de posicionamiento distintas:
  - *android:layout_above*: colocar el elemento encima del elemento referenciado.
  - *android:layout_below*: colocar el elemento debajo del elemento referenciado.
  - *android:layout_toLeftOf*: colocar el elemento a la izquierda del elemento referenciado.
  - *android:layout_toRightOf*: colocar el elemento a la derecha del elemento referenciado.
  - *android:layout_alignTop*: indica que el extremo superior de este elemento está alineado con el extremo superior del elemento referenciado.

1.2.2. Posicionamiento relativo a otros elementos

- *android:layout_alignBottom*: indica que el extremo inferior de este elemento está alineado con el extremo inferior del elemento referenciado.
- *android:layout_alignLeft*: indica que el extremo izquierdo de este elemento está alineado con el extremo izquierdo del elemento referenciado.
- *android:layout_alignRight*: indica que el extremo derecho de este elemento está alineado con el extremo derecho del elemento referenciado.
- *android:layout_alignBaseLine*: indica que las líneas de base de este elemento están alineadas con las del elemento referenciado.

- Puede centrar elementos en un RelativeLayout mediante las opciones:
  - android:layout_centerHorizontal: permite centrar un elemento horizontalmente.
  - android:layout_centerVertical: permite centrar un elemento verticalmente.
  - android:layout_centerInParent: permite centrar un elemento en el contenedor padre.

1.1.3. 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"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/nomEdit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:hint="@string/email" />
    <EditText
        android:id="@+id/prenomEdit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@id/nomEdit"
        android:hint="@string/pass" />
```

Alineación extremo superior y derecho

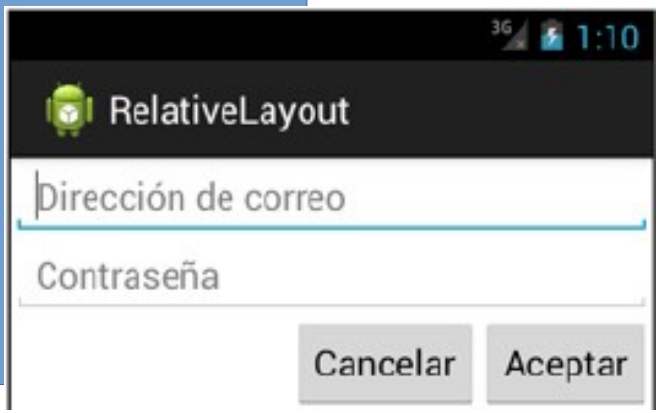
Alineado por su extremo derecho con el extremo derecho del contenedor y por debajo del elemento anterior.

1.1.3. Ejemplo

```
<Button
    android:id="@+id/valider"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/prenomEdit"
    android:layout_below="@id/prenomEdit"
    android:text="@string/ok" />
<Button
    android:id="@+id/annuler"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/valider"
    android:layout_toLeftOf="@id/valider"
    android:text="@string/cancel" />
</RelativeLayout>
```

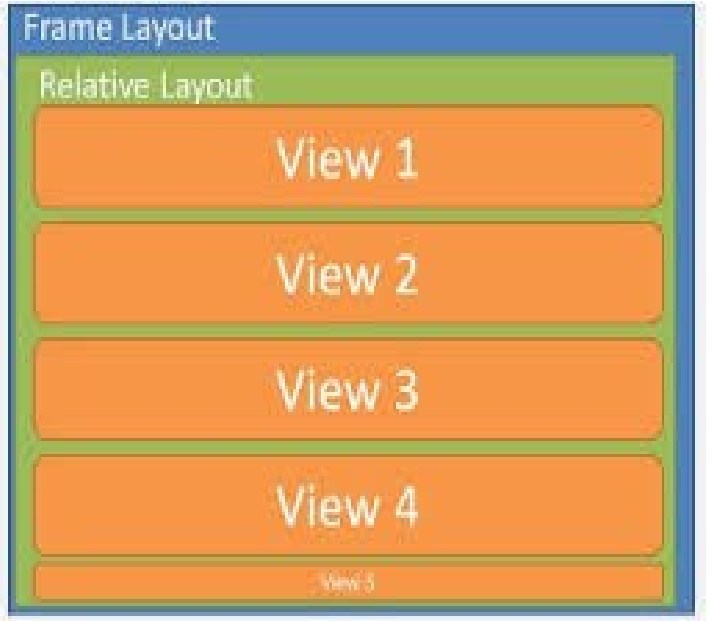
Situado debajo del segundo campo, alineado a su derecha

Alineado por arriba con el primer botón y se encuentra justo a su izquierda.



1.3. FrameLayout

- Posiciona elementos usando todo el contenedor.
- Normalmente cuando queremos que varios elementos ocupen el mismo lugar y uno solo visible.
  - Propiedad *visibility*.



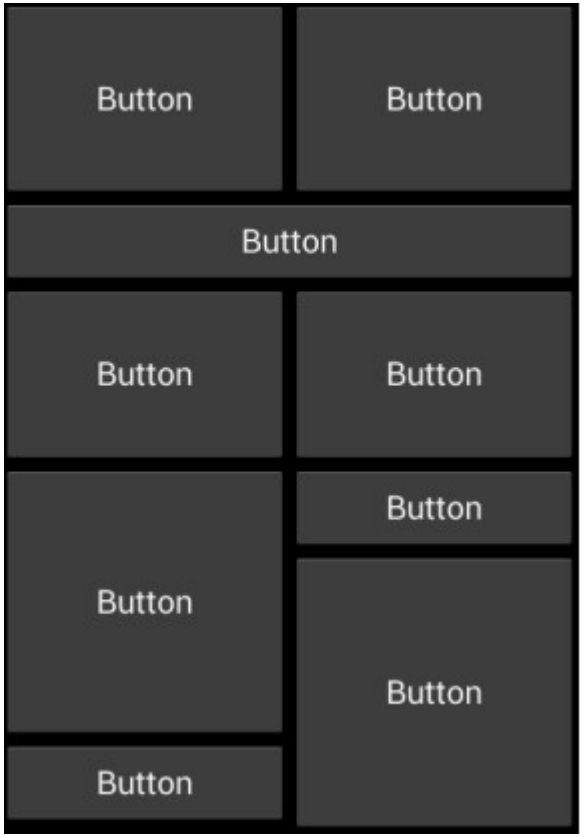
1.3. Ejemplo

```
<FrameLayout xmlns : android="http://schemas...
    android: layout_height="match_parent"
    android:layout_width="match_parent">
    <AnalogClock
        android:layout width= "wrap_content"
        android:layout_height= "wrap_content"
    <CheckBox
        android:layout_width= "wrap_content"
        android:layout_height= "wrap_content"
        android:text = "UncheckBox" />
    <Button
        android: layout width= "wrap_content"
        android:layout_height= "wrap_content"
        android:text = " Unbotón"
        android:visibi1ity= "invisible" / >
    <TextViev
        android:layout_width= ”wrap_coatent "
        android:layout_height = " wrap_content "
        android:text = "Untextocualquier a "
        android:visibi1ity= " invisible" />
    </FraraeLayout >
```

¿Que saldría?

1.4. GridLayout

- Permite dividir la pantalla en filas y columnas.
- El elemento ***spacer*** permite dejar una celda vacía y, de este modo, tener espacios vacíos en una interfaz gráfica.



1.4. Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="3" >

    <Space
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn5" />
```

Indica que es de 3 columnas

Primera fila, es un espacio vacío.

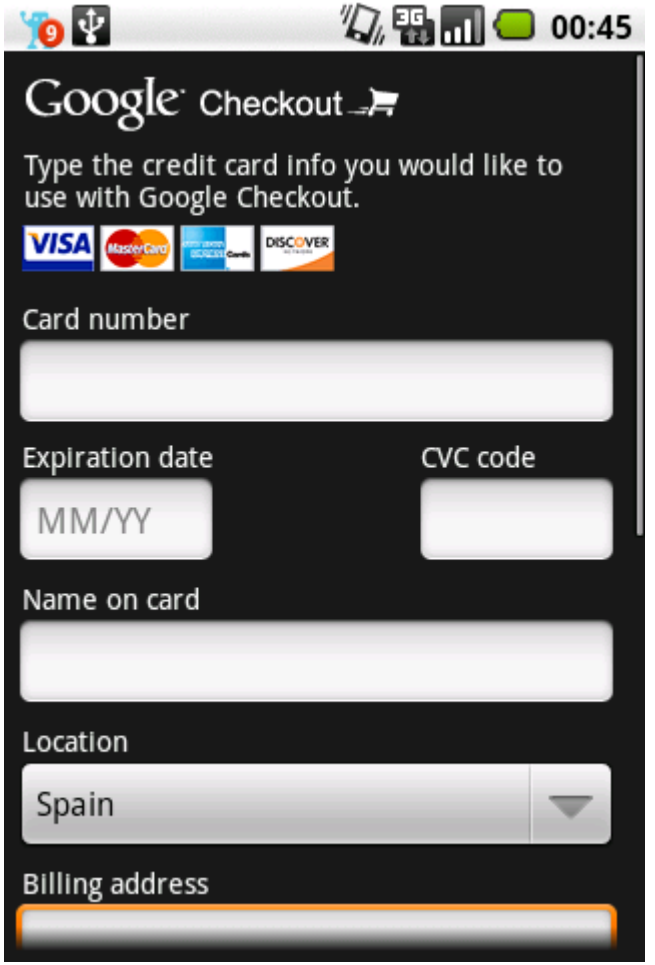
1.4. Ejemplo

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/btn6" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/btn7" />
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_columnSpan="2"
    android:text="@string/btn8" />
</GridLayout>
```

Ocupa dos columnas

2. Formularios

- *Button*: Botón
- *ToggleButton*: una especie de checkbox.
- *ImageButton*: botón con imagen.
- *ImageView*: Manejo de imágenes en pantalla.
- *TextView*: etiqueta no editable.
- *EditText*: Campo de texto editables.



2.1. TextView

- La especificación del texto que se desea mostrar se realiza mediante el atributo *android:text*

```
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/text" />
```

2.2. EditText

- *android:hint* permite indicar al usuario el tipo de texto esperado.
- *android:inputType:* mejorar la experiencia del usuario mostrando un teclado específico en función del tipo de campo. Valores:
  - *text* (valor por defecto): teclado normal.
  - *textCapCharacters*: teclado todo en mayúsculas.
  - textCapWords: primera letra automáticamente en mayúsculas.
  - *textAutoCorrect*: activa la corrección automática.
  - *textMultiLine*: texto en varias líneas.
  - *textNoSuggestions*: sin sugerencias de corrección.
  - *textUri*: permite introducir una URL web.
  - *textEmailAddress*: dirección de correo electrónico.
  - *textEmailSubject*: asunto de correo electrónico.
  - *textShortMessage*: activa el acceso directo a smiley en el teclado.
  - *textPersonName*: permite introducir el nombre de una persona (muestra speech to text en la parte inferior izquierda).
  - *textPostalAddress*: permite introducir una dirección postal (muestra speech to text en la parte inferior izquierda del teclado).
  - *textPassword*: entrada de una contraseña.
  - *textVisiblePassword*: entrada de una contraseña visible.
  - *number*/*numberSigned*/numberDecimal/*phone*/*datetime*/*date*/*time*: teclado numérico.

```
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/text" />
```

2.3. Button


```
<Button android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/btn" />
```

2.4. Checkbox

- El elemento Checkbox representa una simple casilla de selección, como las que se pueden activar en los formularios web:
- *android:checked*: Puede definir el estado inicial de una *checkbox* (*true* si activa, *false* en caso contrario)

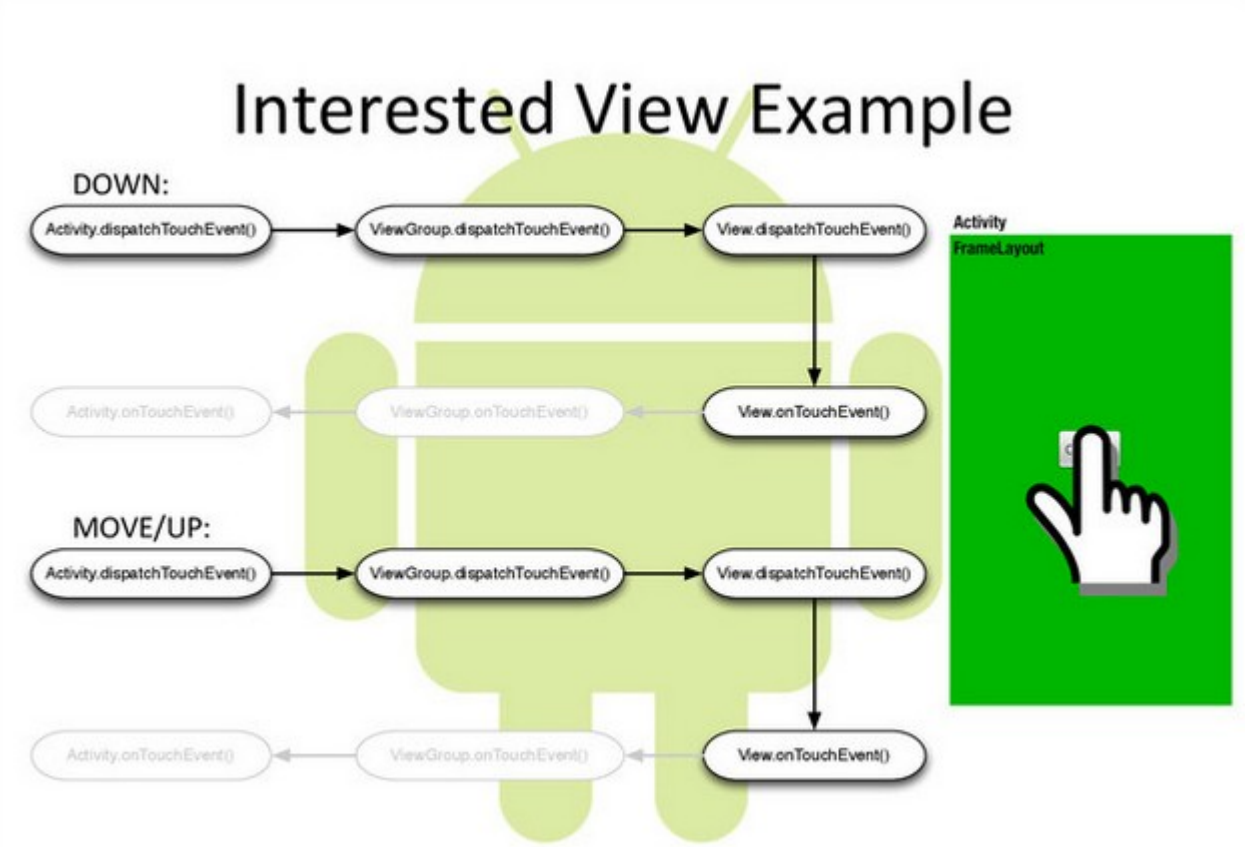
```
<CheckBox android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:checked="true"
android:text="@string/checkbox" />
```

2.5. ImageView

- Para añadir imágenes en una aplicación fácilmente.
- *android:src* permite especificar la imagen que se desea mostrar.
- *android:contentDescription* permite proporcionar una breve descripción de la imagen mostrada (utilizado para accesibilidad).

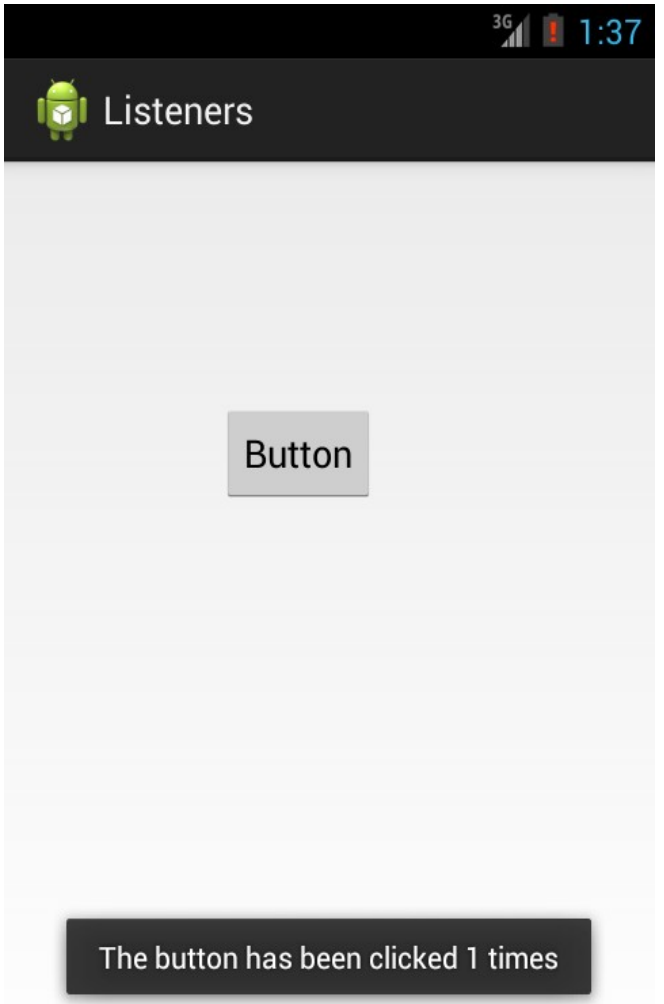
```
<ImageView
android:layout_width="fill_parent "
android:layout_height="wrap_content "
android:src="@drawable/ic_launcher "
android:contentDescription="@string/app_name "/>
```

2. Listeners


Interested View Example
DOWN:
Activity.dispatchTouchEvent()
ViewGroup.dispatchTouchEvent()
View.dispatchTouchEvent()
Activity.onTouchEvent()
ViewGroup.onTouchEvent()
View.onTouchEvent()
MOVE/UP:
Activity.dispatchTouchEvent()
ViewGroup.dispatchTouchEvent()
View.dispatchTouchEvent()
Activity.onTouchEvent()
ViewGroup.onTouchEvent()
View.onTouchEvent()
Activity
FrameLayout

2.1. Definición

- Manejadores de eventos en Android.
- Permiten gestionar las distintas acciones que realiza el usuario y asociarlas a un conjunto de funcionalidades.
- Normalmente vienen definidas con un Interfaz Java y deben implementarse.
- La gestión del clic puede hacerse de dos formas distintas:
  - Gestionar el clic en los botones, de forma separada.
  - Hacer que su actividad implemente la interfaz *onClickListener*.



2.1. Ejemplo: Gestionar el clic de manera separada


```
Button btn1 = (Button) findViewById(R.id.btn1);
btn1.setOnClickListener( new OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.v("ClickListener", "Interacción con el botón 1");
    }
});

Button btn2 = (Button) findViewById(R.id.btn2);
btn2.setOnClickListener( new OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.v("ClickListener", "Interacción con el botón 2");
    }
});
```

Iniciados mediante el método *findViewById*

Recibe por parámetro un nuevo *listener* que permite sobrecargar el método *onClick*.

Tratamiento realizado por el método *onClick*

2.2. Capturar eventos (I)

- *View.setOnClickListener:* se llama cuando el usuario selecciona un elemento. Se puede utilizar cualquier medio como la pantalla táctil, las teclas de navegación o eltrackball.
  - *onClick()*
- *View.setOnLongClickListener:* se llama cuando el usuario selecciona un elemento durante más de un segundo.
  - *onLongClick()*
- *View.setOnFocusChangeListener*: se llama cuando el usuario navega dentro o fuera de un elemento.
  - *onFocusChange()*
- *View.setOnKeyListener:* se llama cuando se pulsa o se suelta una tecla del dispositivo.
  - *onKey()*

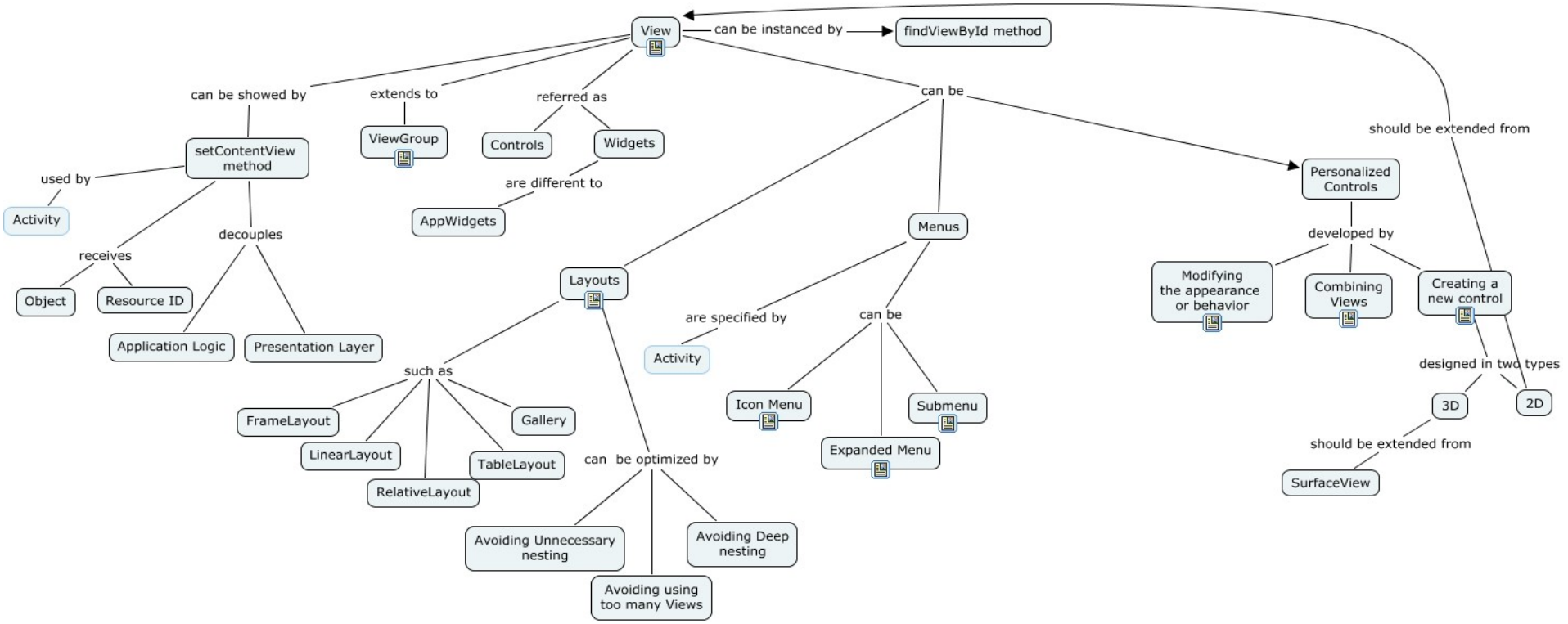
2.2. Capturar eventos (II)

- *View.setOnTouchListener: se llama cuando se pulsa o se suelta o se desplaza en la pantalla táctil.*
  - *onTouch()*
- *View.setOnCreateContextMenuListener:* se llama cuando se crea un menú de contexto.
  - *onCreateContextMenu()*
- *View.setOnDragListener:* Permite capturar y arrastar el componente.
- *View.setOnHoverListener:* Cada vez que nos desplazamos por él.

2.2. Ejemplo: OnLongClickListener

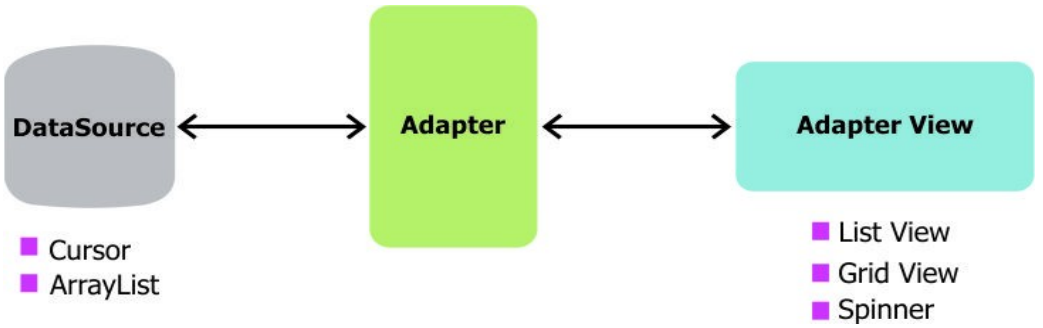

```
Button boton = (Button) findViewById(R.id.boton);
    boton.setOnLongClickListener(new OnLongClickListener() {
            @Override
        public boolean onLongClick(View v) {
                // TODO Auto-generated method stub
            Toast.makeText(v.getContext(), "Estas pulsando 1 seg",
                        Toast.LENGTH_SHORT).show();
             return true;
        }
});
```

3. Colecciones de datos y Views



3.1. Adapter

- Componente que relaciona la vista con los datos.
- Todos los controles de selección accederán a los datos que contienen a través de un adaptador.
- Android proporciona de serie varios tipos de adaptadores sencillos:
  - *ArrayAdapter*. Es el más sencillo de todos los adaptadores, y provee de datos a un control de selección a partir de un array de objetos de cualquier tipo.
  - *SimpleAdapter*. Se utiliza para mapear datos sobre los diferentes controles definidos en un fichero *XML* de layout.
  - *SimpleCursorAdapter*. Se utiliza para mapear las columnas de un cursor abierto sobre una base de datos sobre los diferentes elementos visuales contenidos en el control de selección.



3.1.1. Crear un dinámico *ArrayAdapter*


```
final String[] datos =
        new String[]{"Elem1","Elem2","Elem3","Elem4","Elem5"};

ArrayAdapter<String> adaptador =
        new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, datos);
```

pasamos el ID de un layout predefinido en Android, formado únicamente por un control *TextView*

Arreglo con los datos a mostrar

3.1.2. Crear un estático *ArrayAdapter*


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string-array name="valores_array">
      <item>Elem1</item>
      <item>Elem2</item>
      <item>Elem3</item>
      <item>Elem4</item>
      <item>Elem5</item>
   </string-array>
</resources>
```

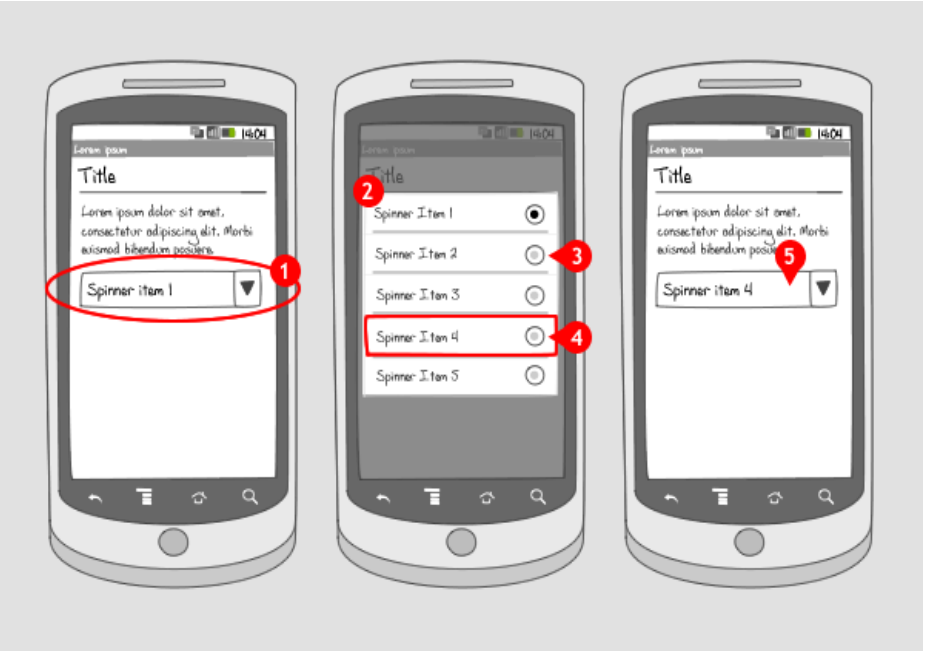
Recurso de tipo *string-array creado en un fichero xml en /res/values*

```
ArrayAdapter<CharSequence> adapter =
   ArrayAdapter.createFromResource(this,
      R.array.valores_array,
      android.R.layout.list_item_1);
```

Para hacer referencia a este array XML que acabamos de crear

3.2. Spinner

- El usuario selecciona la lista, se muestra una especie de lista emergente al usuario con todas las opciones disponibles y al seleccionarse una de ellas ésta queda fijada en el control.




```
<Spinner android:id="@+id/CmbOpciones"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

3.2.1. Código Java

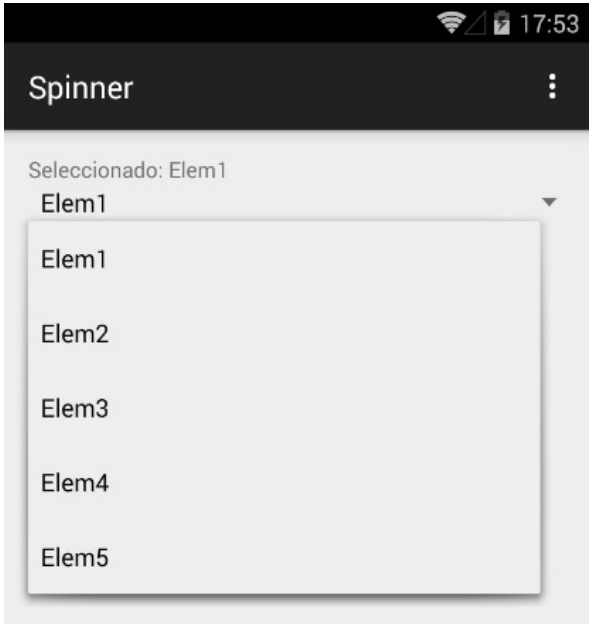
```
private Spinner cmbOpciones;
    //...
    cmbOpciones = (Spinner)findViewById(R.id.CmbOpciones);

    adaptador.setDropDownViewResource(
        android.R.layout.simple_spinner_dropdown_item);

    cmbOpciones.setAdapter(adaptador);
```

Muestra elementos en otra línea emergente con otro estilo

Asignamos el adaptador al control



3.2.2. Eventos lanzados por spinner


```
cmbOpciones.setOnItemSelectedListener(
      new AdapterView.OnItemSelectedListener() {
         public void onItemSelected(AdapterView<?> parent,
                        android.view.View v, int position, long id) {
            lblMensaje.setText("Seleccionado: " +
                  parent.getItemAtPosition(position));
         }

         public void onNothingSelected(AdapterView<?> parent) {
            lblMensaje.setText("");
         }
      });
```

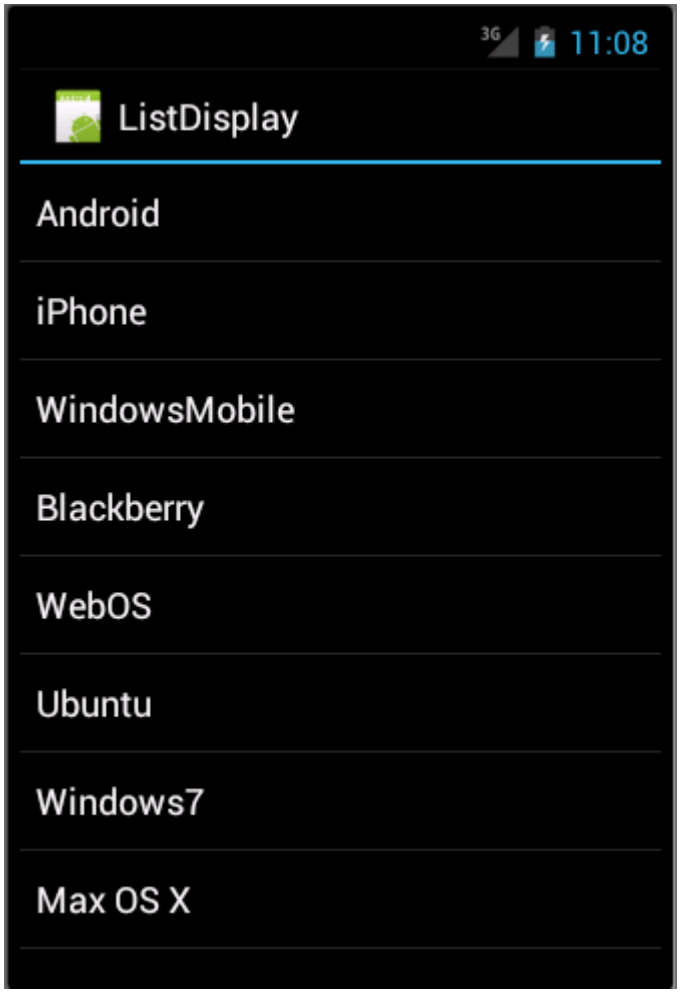
Para el item seleccionado

Cuando no hay ninguna opción seleccionada

3.3. ListView

- Permite la presentación de un listado de datos en pantalla en formato listado.

```
<ListView android:id="@+id/LstOpciones"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```



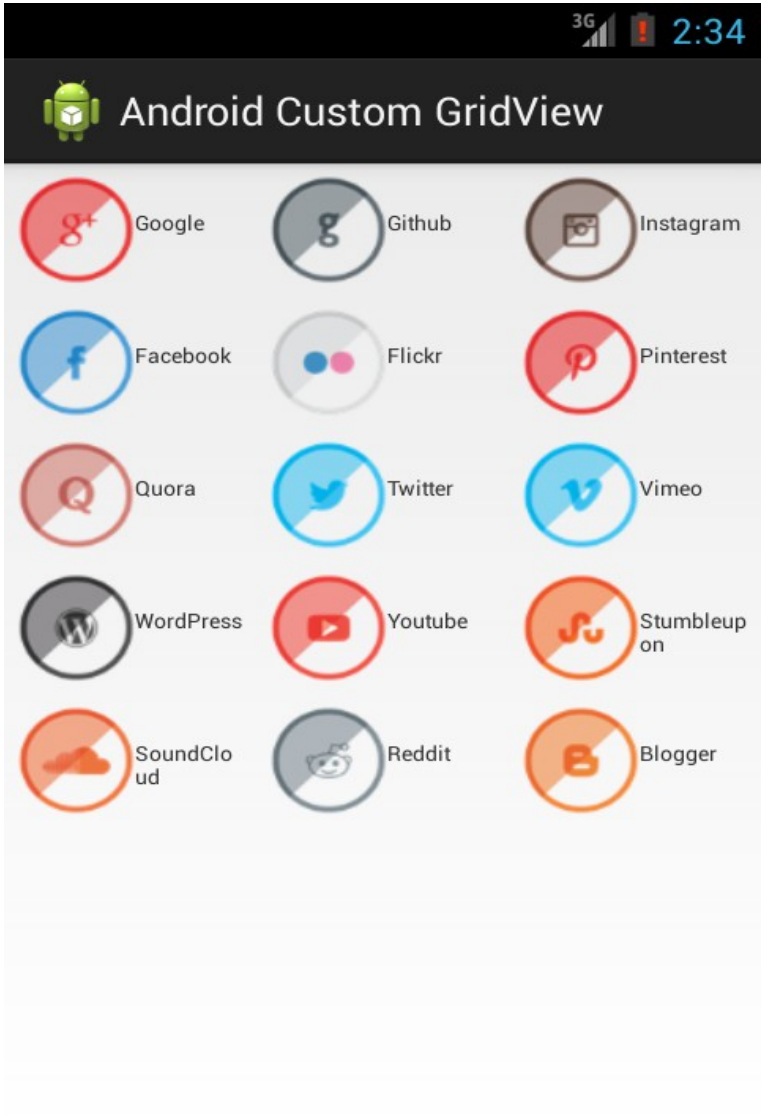
3.3.1. Controlador

```
lv = (ListView)findViewById(R.id.LstOpciones);
public static final String [] COMIDAS={
      "arroz con pollo","ají de gallina","lomo saltado",
     "tacu tacu", "ceviche", "hostia que hambre me ha dado"
};
ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,
                  R.layout.simple_list_item_1, COMIDAS);
      lv.setAdapter(adapter);
```

Se asocia el adaptador

3.4. GridView

- Similar al *ListView* pero con otra disposición, de rejilla.




```
<GridView android:id="@+id/GridOpciones"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:numColumns="auto_fit"
    android:columnWidth="80px"
    android:horizontalSpacing="5dp"
    android:verticalSpacing="10dp"
    android:stretchMode="columnWidth" />
```

3.4.1. Controlador

```
ArrayAdapter<String> adaptador =
        new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, datos);

grdOpciones = (GridView)findViewById(R.id.GridOpciones);

grdOpciones.setAdapter(adaptador);
```

3.2.2. Evento lanzado por GridView

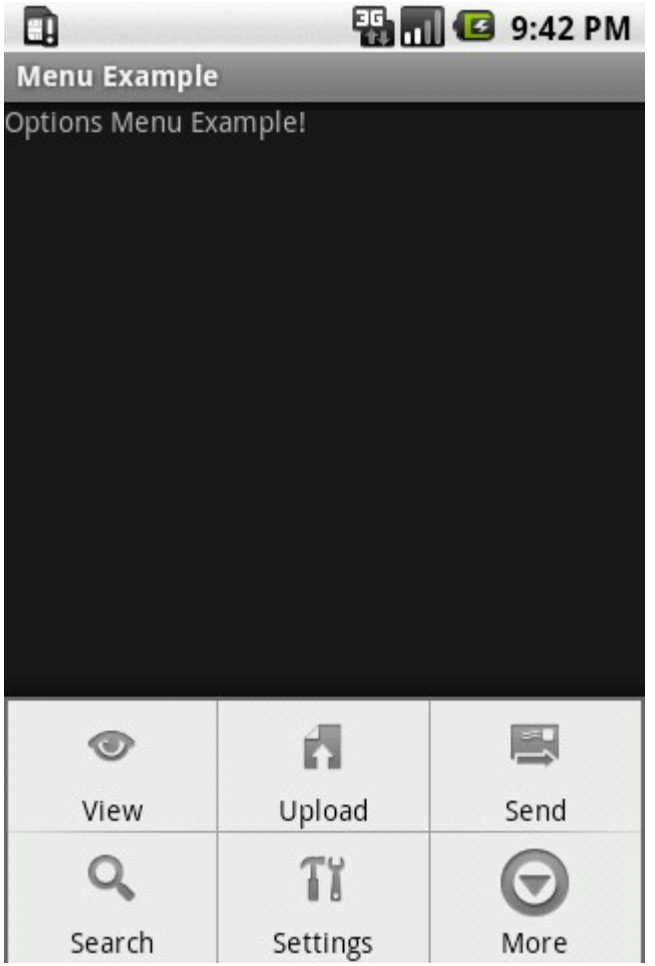

```
grdOpciones.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent,
            android.view.View v, int position, long id) {
                lblMensaje.setText("Opción seleccionada: "
                    + parent.getItemAtPosition(position));
            }
    });
```

4. Menús


10:16 AM
Vlingo
What would you like to do?
Messaging
Voice Dial
Restaurants
Businesses
Search
Speak it
10:48 AM
Vlingo InCar Beta
Call
Text
Navigate
SafeReader
Tap & Speak
10:20 AM
Jay Leno
mobile 12223334567
I am running late and I will see you in 10 minutes.
Cancel
Send

4.1. Definición

- Permite la introducción de acciones complementarias al usuario en una *Activity*.
- Está históricamente relacionado con el botón físico de Menú de los antiguos Android, en desuso a partir de 3.0.
- Ahora tiene una representación en las nuevas barras de acciones de Android 4.x.
- El fichero que define el menú es un *XML* normalmente en */res/menu*.
- Se pueden declarar por dos formas:
  - Vista *(xml)*.
  - Controlador(Java).



4.2. Menú lanzado en vista (*I*)


```
//código XML
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/MnuOpc1" android:title="Opcion1"
        android:icon="@android:drawable/ic_menu_preferences"></item>
    <item android:id="@+id/MnuOpc2" android:title="Opcion2"
        android:icon="@android:drawable/ic_menu_compass"></item>
</menu>

//Código Java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.activity_main, menu);
    return true;
}
```

4.2. Menú lanzado en vista (*II*)


```
@Override
public boolean onContextItemSelected(MenuItem item) {

    switch (item.getItemId()) {
        case R.id.CtxLblOpc1:
            lblMensaje.setText("Etiqueta: Opcion 1 pulsada!");
            return true;
        case R.id.CtxLblOpc2:
            lblMensaje.setText("Etiqueta: Opcion 2 pulsada!");
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

4.2. Menú lanzado en controlador (*I*)


```
private static final int MNU_OPC1 = 1;
private static final int MNU_OPC2 = 2;
private static final int MNU_OPC3 = 3;
//...

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    //Alternativa 2
    menu.add(Menu.NONE, MNU_OPC1, Menu.NONE, "Opcion1")
            .setIcon(android.R.drawable.ic_menu_preferences);
    menu.add(Menu.NONE, MNU_OPC2, Menu.NONE, "Opcion2")
            .setIcon(android.R.drawable.ic_menu_compass);
    menu.add(Menu.NONE, MNU_OPC3, Menu.NONE, "Opcion3")
            .setIcon(android.R.drawable.ic_menu_agenda);
    return true;
}
```

ID asociado a la acción (cuando tenemos varias opciones)

ID de la opción

Texto de la opción

Icono

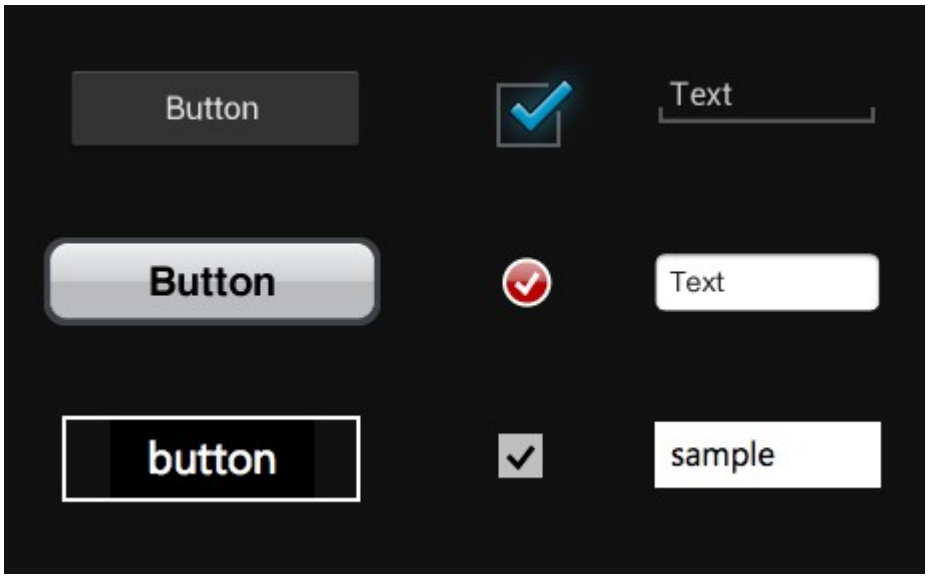
4.2. Menú lanzado en controlador (*II*)


```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.MnuOpc1:
            lblMensaje.setText("Opcion 1 pulsada!");
            return true;
        case R.id.MnuOpc2:
            lblMensaje.setText("Opcion 2 pulsada!");;
            return true;
        case R.id.MnuOpc3:
            lblMensaje.setText("Opcion 3 pulsada!");;
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

5. Estilos y temas

5.1. Estilos

- Definen características de las vistas, muy similar al concepto de *CSS*.



5.2. Temas

- Conjuntos de estilos predefinidos.
- A día de hoy los dos más actuales son *Holo* y *Material Design*.
- Se definen en el fichero */res/values/styles.xml*.