

Apuntes de Android

José A Montenegro Montes

15 de julio de 2016

Índice general

1. Interfaz Usuario	11
1.1. Layouts	11
1.1.1. FrameLayout	11
1.1.2. LinearLayout	13
1.1.3. TableLayout	16
1.1.4. GridLayout	20
1.1.5. RelativeLayout	22
1.2. Atributos	31
1.2.1. ViewGroup.LayoutParams	31
1.2.2. ViewGroup.MarginLayoutParams	31
1.3. Estilo y Temas	35
1.3.1. Estilo	35
1.3.2. Temas	38
1.4. Widgets	41
1.4.1. RadioButton y RadioGroup	41
1.4.2. CheckBox	45
1.4.3. Spinner	49
1.4.4. Switch y ToggleButton	56
1.4.5. ImageButton y ImageView	59
1.4.6. ProgressBar	61
1.4.7. ProgressDialog	67
1.4.8. SeekBar	69
1.4.9. RatingBar	72
1.4.10. WebView	74
1.5. Containers	81
1.5.1. ListView	81
1.5.2. GridView	83
1.5.3. ExpandableListView	91
1.5.4. TabHost	97
1.6. Date and Time	107

1.6.1. Chronometer	107
1.6.2. Pickers	112
2. Almacenamiento	119
2.1. Preferencias Compartidas	119
2.2. Almacenamiento Interno	121
2.3. Almacenamiento Externo	124
2.4. Base de Datos	132
2.4.1. Creación de la Base de datos	132
2.4.2. Inserción en la Base de datos	134
2.4.3. Borrado en la Base de datos	136
2.4.4. Actualizar en la base de datos	137
2.4.5. Consultas base de datos	138
2.4.6. Aplicación Ejemplo	144
2.4.7. SQL y Android	158
3. Comunicaciones	163
3.1. Conexiones HTTP	163
3.2. Conexiones HTTPS	170
3.3. Bluetooth	177
3.3.1. Configurando Bluetooth	178
3.3.2. Buscando dispositivos	180
3.3.3. Descubriendo nuevos dispositivos	181
3.3.4. Vinculación y Desvinculación	182

List of source codes

1.1.	XML FrameLayout	11
1.2.	XML LinearLayout	13
1.3.	XML TableLayout	16
1.4.	XML grid Layout	20
1.5.	XML Calculadora	25
1.6.	Atributos TextView	31
1.7.	TextView sin Style	35
1.8.	TextView con Style	35
1.9.	Style	35
1.10.	XML RadioButton	41
1.11.	Java RadioButton	43
1.12.	XML CheckBox	45
1.13.	Java CheckBox	46
1.14.	XML Spinner	49
1.15.	Resource Spinner	51
1.16.	Java Spiner Array String	53
1.17.	Java Spinner Resources	54
1.18.	XML ToggleButton	56
1.19.	Java ToggleButton	58
1.20.	XML ImageButton Button y ImageView	59
1.21.	XML Progress Bar	62
1.22.	Java Progress Bar	65
1.23.	XML SeekBar	69
1.24.	Java SeekBar	71
1.25.	XML RatingBar	72
1.26.	Java RatingBar	73
1.27.	XML WebView	75
1.28.	Java WebView	76
1.29.	Java ListViewActivity	81
1.30.	XML GridView	83
1.31.	Java GridViewActivity String Adapter	85

1.32. Java GridViewActivity Image Adapter	87
1.33. XML ExpandableList	91
1.34. XML ExpandableList	92
1.35. XML ExpandableList	92
1.36. Java ExpandableList	93
1.37. XML TabHost	97
1.38. Java TabHost	99
1.39. XML Chronometer	107
1.40. Java Chronometer	110
1.41. XML Picker	113
1.42. Java Picker	115
1. Lectura de un Recurso	123
2. Método Escritura en la SD	125
3. Método Lectura en la SD	126
2.1. Java AlmacenamientoActivity	126
2.2. XML AlmacenamientoActivity	130
4. Creación clase que hereda de SQLiteOpenHelper	133
5. Ejemplo creación tabla de datos Sqllite	133
6. Ejemplo eliminación tabla de datos Sqllite	134
7. Ejemplo inserción datos Sqllite	136
8. Ejemplo inserción tabla de datos Sqllite	136
9. Ejemplo de método delete	137
10. Ejemplo delete tabla de datos SQLite	137
11. Ejemplo del método update	137
12. Ejemplo actualizar tabla de datos SQLite	137
13. Ejemplo query tabla de datos SQLite	138
14. Ejemplo de consulta para obtener todos los usuarios BD	138
15. Ejemplo de consulta para obtener todos los usuarios BD	138
16. Ejemplo del Objeto Cursor	140
2.3. Java AdaptadorBD	140
2.4. Java AdaptadorBD	145
2.5. Java SQLActivity	149
2.6. XML SQLActivity	150
2.7. Java AddAlumnos	151
2.8. XML AddAlumnos	153
2.9. Java Consulta	154
2.10. XML Consulta	156
2.11. Java Alumnos	157
17. Método utilizado para verificar la conexión	164
18. Método Conectar url y lectura página web	164
3.1. Java Http	164

3.2.	XML Http	168
19.	Conexión https y obtención certificados	170
3.3.	JavaHttps	170
3.4.	Java Cert	175
3.5.	XMLHttps	176
20.	Dispositivo soporta Bluetooth	179
21.	Habilitar Bluetooth	179
22.	Registrar un BroadcastReceiver para recibir cambio de estados	180
23.	Solicitar los dispositivos vinculados.	180
24.	Registrar un BroadcastReceiver para recibir cambio de estados	181
25.	Registrar un BroadcastReceiver para recibir cambio de estados	182
26.	Vinculación y Desvinculación de Dispositivos	183
3.6.	Java Bluetooth	183
3.7.	XML Bluetooth	191

Intro

Este documento no pretende ser un libro sobre Android, simplemente un conjunto de apuntes que son modificados constantemente y sin una revisión a fondo.

Contendrá errores que serán modificados en revisiones posteriores.

Capítulo 1

Interfaz Usuario

1.1. Layouts

1.1.1. FrameLayout

```
https://developer.android.com/reference/android/widget/FrameLayout.html?hl=es
```

FrameLayout es diseñado para mostrar un único elemento. Es posible añadir varios elementos y controlar su posición mediante el atributo *android:layout_gravity*.

Listing 1.1: XML FrameLayout

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button13"
        android:layout_gravity="left|top" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button14"
        android:layout_gravity="right|top" />
```

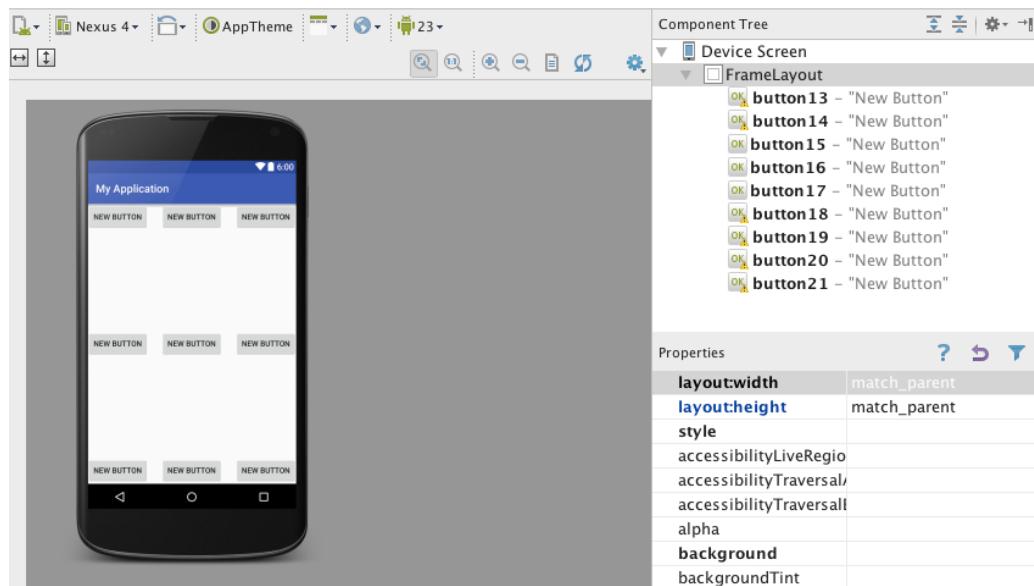


Figura 1.1: FrameLayout con varios elementos

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button15"
    android:layout_gravity="center_horizontal|bottom" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button16"
    android:layout_gravity="center_horizontal|top" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button17"
    android:layout_gravity="center" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button18"
    android:layout_gravity="center" />

```

```
        android:layout_gravity="left|center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button19"
        android:layout_gravity="right|center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button20"
        android:layout_gravity="left|bottom" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button21"
        android:layout_gravity="right|bottom" />
</FrameLayout>
```

1.1.2. LinearLayout

<https://developer.android.com/reference/android/widget/LinearLayout.html?hl=es>

Un Layout que ordena sus hijos en una columna (horizontal) o una fila (vertical).

Listing 1.2: XML LinearLayout

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

<LinearLayout android:orientation="vertical"
    android:layout_height="wrap_content"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
```

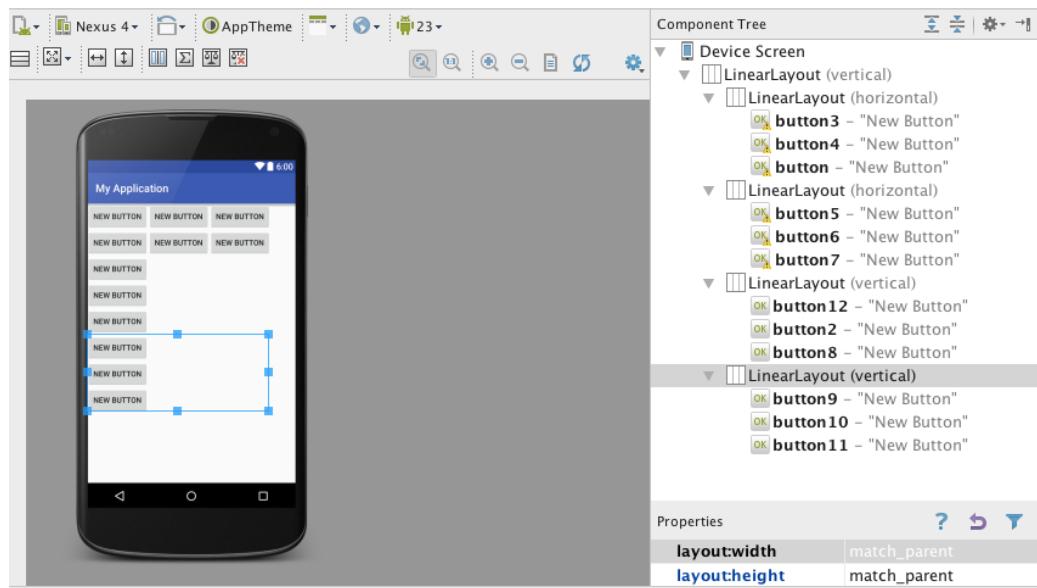


Figura 1.2: LinearLayout Horizontal y Vertical

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button3" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button4" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button" />

</LinearLayout>

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button

```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button5" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button6" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button7" />

</LinearLayout>

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button12" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button2" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button8" />
</LinearLayout>

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button9" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button10" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button11" />
</LinearLayout>

</LinearLayout>
```

1.1.3. TableLayout

<https://developer.android.com/guide/topics/ui/layout/grid.html>

<https://developer.android.com/reference/android/widget/TableRow.html>

Es un layout que establece las vistas en filas y columnas. Un *TableLayout* consiste de un número de objetos *TableRow*, cada uno definiendo una fila. Cada fila tendrá cero o más celdas y cada una contiene un objeto. Las celdas pueden ocupar más de una columna (span). El tamaño de una columna es definido por la columna con la celda con el mayor tamaño en esa columna.

Listing 1.3: XML TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res
    /android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
```

1.1. LAYOUTS

17

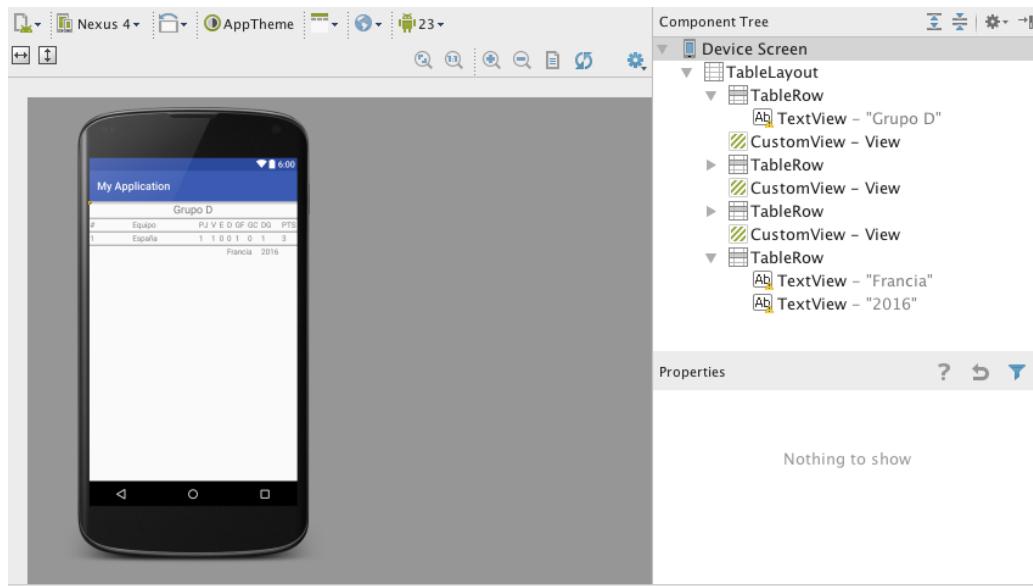


Figura 1.3: TableLayout

```
<TableRow android:gravity="center_horizontal">
    <TextView
        android:text="Grupo D"
        android:padding="3dip"
        android:textSize="20dp" />

</TableRow>

<View
    android:layout_height="2dip"
    android:background="#FF909090" />

<TableRow>
    <TextView
        android:text="#""
        android:padding="3dip"
        android:layout_column="0"
        android:gravity="left" />
    <TextView
        android:text="Equipo"
        android:layout_column="1"
        android:padding="3dip"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:text="PJ"
        android:layout_column="2"
```

```
        android:padding="3dip" />
<TextView
    android:text="V"
    android:layout_column="3"
    android:padding="3dip" />
<TextView
    android:text="E"
    android:layout_column="4"
    android:padding="3dip" />
<TextView
    android:text="D"
    android:layout_column="5"
    android:padding="3dip" />
<TextView
    android:text="GF"
    android:layout_column="6"
    android:padding="3dip" />
<TextView
    android:text="GC"
    android:layout_column="7"
    android:padding="3dip" />
<TextView
    android:text="DG"
    android:layout_column="8"
    android:padding="3dip" />
<TextView
    android:text="PTS"
    android:layout_column="9"
    android:padding="3dip" />
</TableRow>

<View
    android:layout_height="2dip"
    android:background="#FF909090" />

<TableRow android:baselineAligned="false">
    <TextView
        android:text="1"
        android:padding="3dip"
        android:gravity="left"
        android:layout_column="0"/>
    <TextView
        android:text="España"
        android:padding="3dip"
        android:layout_column="1"
        android:layout_gravity="top"
        android:layout_width="wrap_content" />
    <TextView
```

```
        android:text="1"
        android:layout_column="2"
        android:padding="3dip" />
    <TextView
        android:text="1"
        android:layout_column="3"
        android:padding="3dip" />
    <TextView
        android:text="0"
        android:layout_column="4"
        android:padding="3dip" />
    <TextView
        android:text="0"
        android:layout_column="5"
        android:padding="3dip" />
    <TextView
        android:text="1"
        android:layout_column="6"
        android:padding="3dip" />
    <TextView
        android:text="0"
        android:layout_column="7"
        android:padding="3dip" />
    <TextView
        android:text="1"
        android:layout_column="8"
        android:padding="3dip" />
    <TextView
        android:text="3"
        android:layout_column="9"
        android:padding="3dip" />
</TableRow>

<View
    android:layout_height="2dip"
    android:background="#FF909090" />
<TableRow>

    <TextView
        android:text="Francia"
        android:layout_column="5"
        android:padding="3dip"
        android:layout_span="3" />
    <TextView
        android:text="2016"
        android:layout_column="7"
        android:padding="3dip" />
```

```
</TableRow>
</TableLayout>
```

1.1.4. GridLayout

<https://developer.android.com/reference/android/widget/GridLayout.html>

Layout que establece las vistas en una cuadricula rectangular.

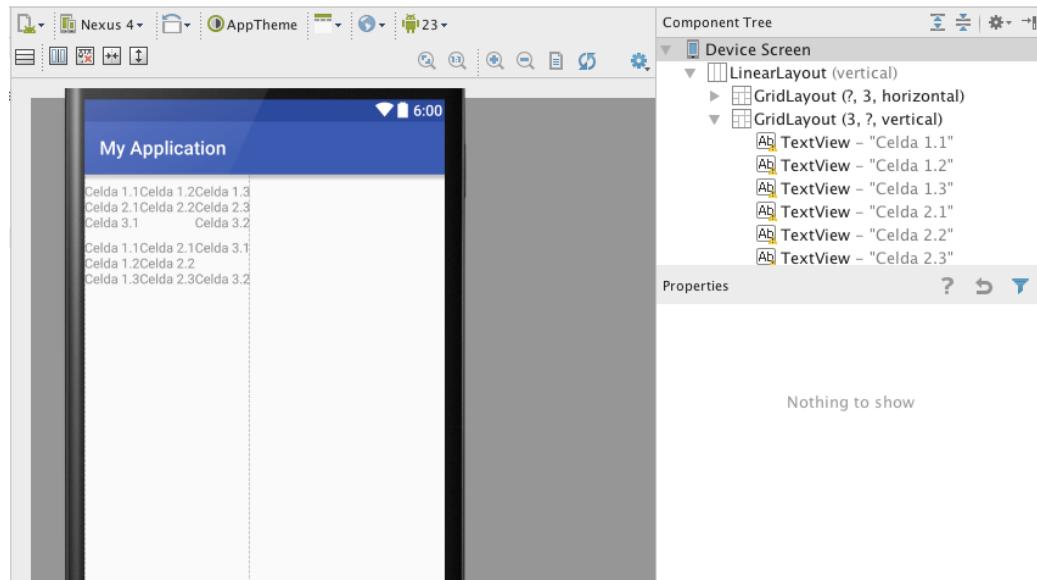


Figura 1.4: GridLayout

Observa cómo modifica el atributo horizontal y vertical al layout.

Listing 1.4: XML grid Layout

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

    <GridLayout android:orientation="horizontal"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
```

```
    android:columnCount="3"
    android:layout_marginTop="10dp">

        <TextView android:text="Celda 1.1" />
        <TextView android:text="Celda 1.2" />
        <TextView android:text="Celda 1.3" />

        <TextView android:text="Celda 2.1" />
        <TextView android:text="Celda 2.2" />
        <TextView android:text="Celda 2.3" />

        <TextView android:text="Celda 3.1"
            android:layout_columnSpan="2" />

        <TextView android:text="Celda 3.2" />

    </GridLayout>

<GridLayout android:orientation="vertical"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:rowCount="3"
    android:layout_marginTop="10dp">

    <TextView android:text="Celda 1.1" />
    <TextView android:text="Celda 1.2" />
    <TextView android:text="Celda 1.3" />

    <TextView android:text="Celda 2.1" />
    <TextView android:text="Celda 2.2" />
    <TextView android:text="Celda 2.3" />

    <TextView android:text="Celda 3.1"
        android:layout_rowSpan="2" />

    <TextView android:text="Celda 3.2" />

</GridLayout>
```

1.1.5. RelativeLayout

<https://developer.android.com/guide/topics/ui/layout/relative.html>

<https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html>

RelativeLayout permite mostrar vistas en posiciones relativas. La posición de cada vista puede ser especificada como relativa a otro elemento o al área del *RelativeLayout* (*align*).

Algunos parámetros (mostrados en las figuras 1.5 , 1.6) que podemos utilizar son los siguientes:

- *android:layout_above Positions* la esquina inferior de la vista estará sobre la vista dada por el ID.
- *android:layout_below Positions* la esquina superior de la vista estará por debajo de la vista dada por el ID.
- *android:layout_toRightOf Positions* Posición de la esquina derecha de la vista a la izquierda de la vista dada por el ID.
- *android:layout_toLeftOf Positions* Posición de la esquina izquierda de la vista a la derecha de la vista dada por el ID.

Otro conjunto de atributos son los siguientes (figura 1.7):

- *android:layout_alignBottom* Hace que la esquina inferior de la vista coincida con la esquina inferior de la vista referenciada por el ID.
- *android:layout_alignLeft* Hace que la esquina izquierda de la vista coincida con la esquina izquierda de la vista referenciada por el ID.
- *android:layout_alignRight* Hace que la esquina derecha de la vista coincida con la esquina derecha de la vista referenciada por el ID.
- *android:layout_alignTop* Hace que la esquina superior de la vista coincida con la esquina superior de la vista referenciada por el ID.
- *android:layout_alignBaseline* Iguala baseline de la vista con el baseline de la vista referenciada por el ID. (Ver figura 1.8)

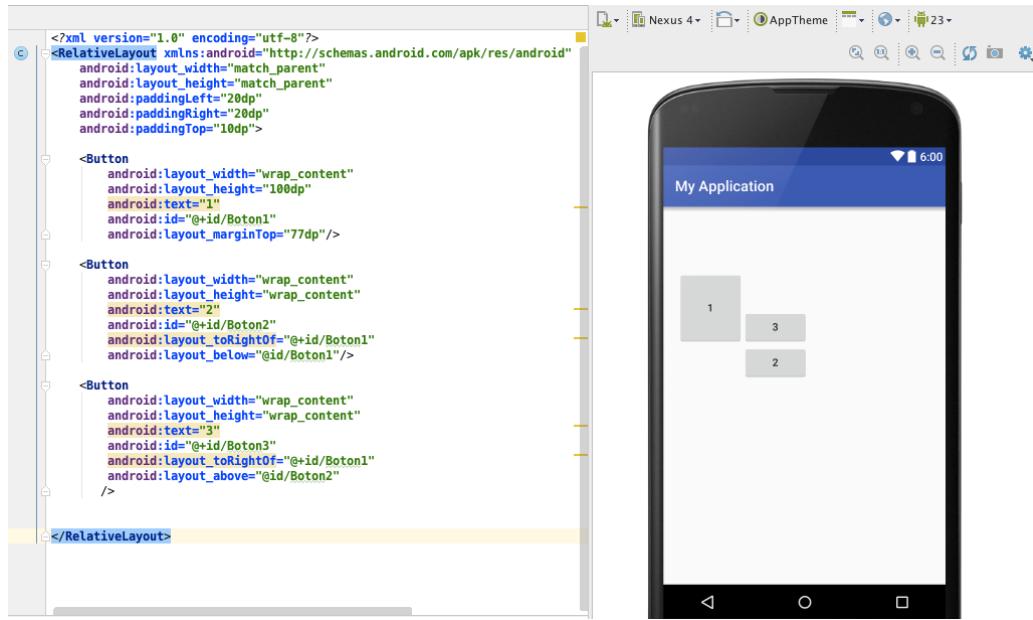


Figura 1.5: RelativeBelowAbove

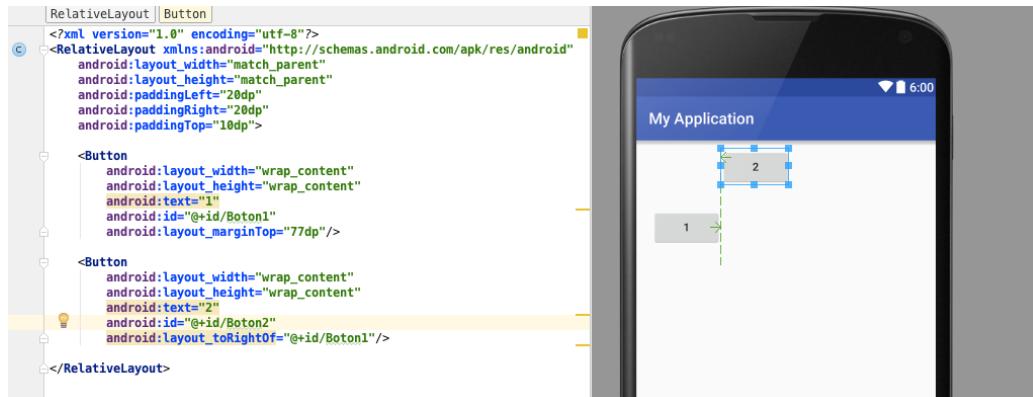


Figura 1.6: RelativeRightOf

Tenemos un grupo de atributos que relación con la vista padre (figura 1.8). Estos atributos tienen prioridad sobre los otros elementos de posición (figura 1.9).

- *android:layout_alignParentBottom* Si true, hace que la esquina inferior coincida con la esquina inferior del padre.
- *android:layout_alignParentLeft* Si true, hace que la esquina izquierda coincida con la esquina izquierda del padre.

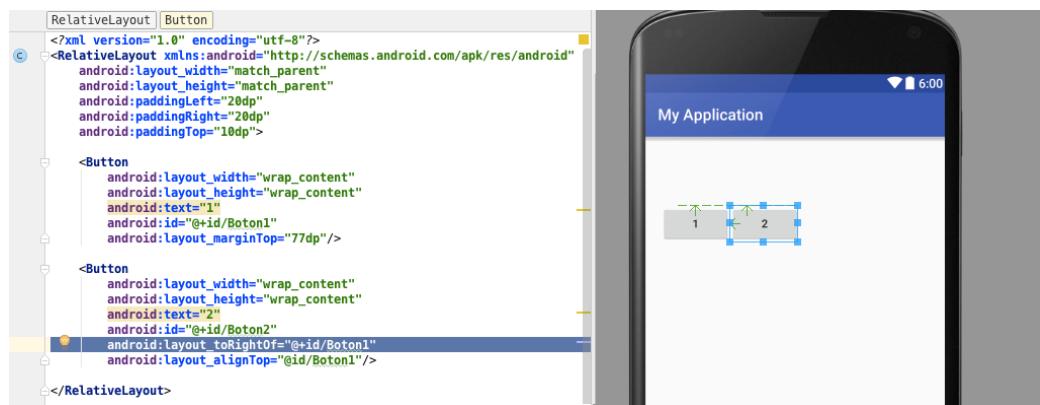


Figura 1.7: RelativeRightTop

- `android:layout_alignParentRight` Si true, hace que la esquina derecha coincida con la esquina derecha del padre.
- `android:layout_alignParentTop` Si true, hace que la esquina superior coincida con la esquina superior del padre.

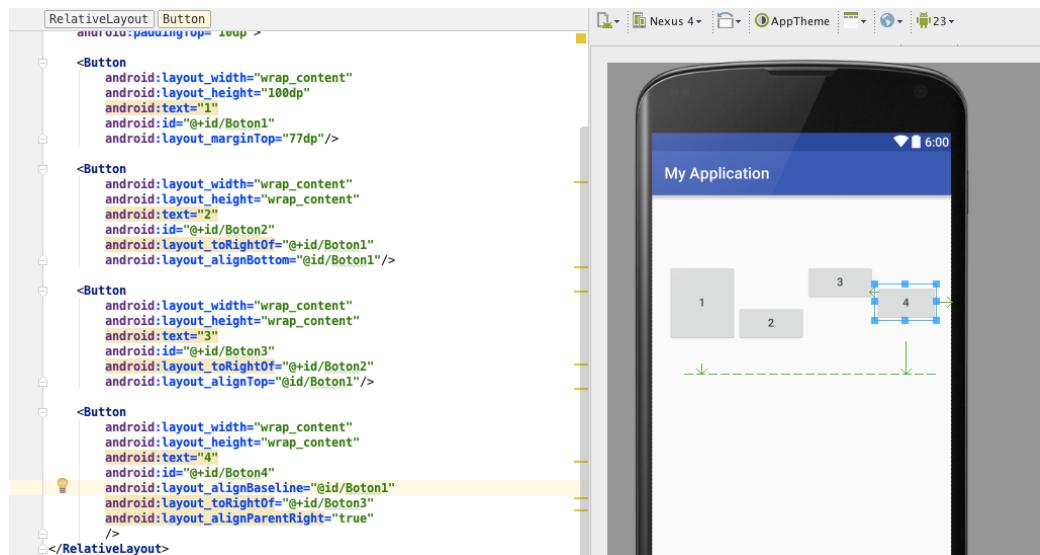


Figura 1.8: RelativeAlignParent

* Ejercicio 1 Creamos una calculadora con RelativeLayout y TableLayout

Creamos el interfaz visual de una calculadora utilizando RelativeLayout y TableLayout, tal y como muestra la figura 1.10.

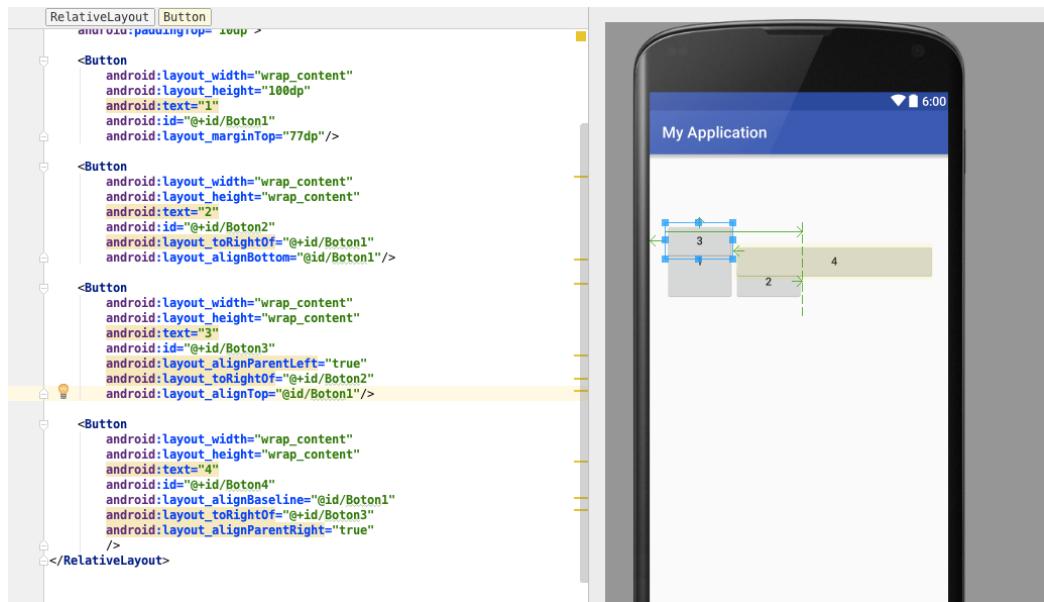


Figura 1.9: RelativePrioridad

Una posible solución al problema puede ser el listado 1.5.

Listing 1.5: XML Calculadora

```

<?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:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:paddingTop="10dp">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="100dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Display"
        android:id="@+id/textDisplay"
        android:textSize="30dp"
        android:textStyle="italic"
        android:typeface="monospace"
        android:gravity="bottom|right" />

    <TextView
        android:layout_width="fill_parent"

```

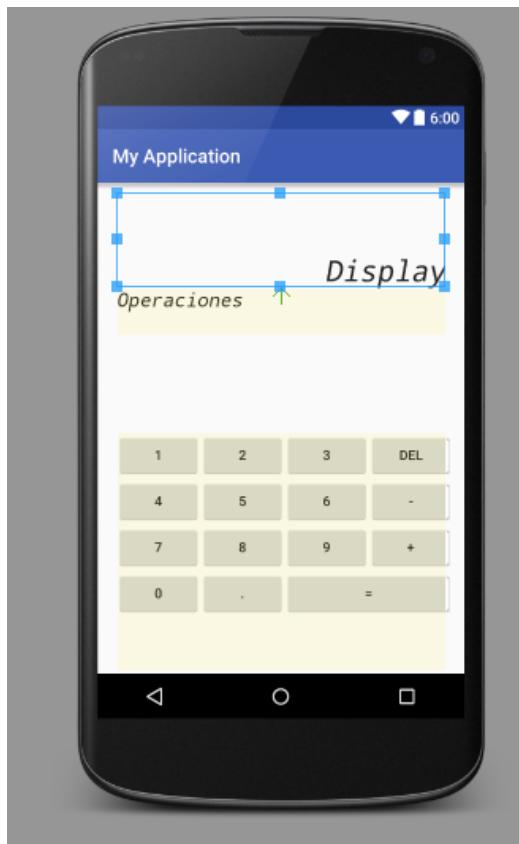


Figura 1.10: Calculadora con RelativeLayout y TableLayout

```
    android:layout_height="50dp"
    android:textAppearance="?android:attr/
        textAppearanceLarge"
    android:text="Operaciones"
    android:id="@+id/textView5"
    android:textSize="20dp"
    android:textStyle="italic"
    android:typeface="monospace"
    android:gravity="top|left"
    android:layout_below="@+id/textDisplay"
    android:layout_centerHorizontal="true" />

<TableLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@+id/textView5"
    android:layout_marginTop="100dp">

    <TableRow
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="1"
        android:id="@+id/button25" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="2"
        android:id="@+id/button26" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="3"
        android:id="@+id/button27" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Del"
        android:id="@+id/button30"
        android:layout_column="3" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="4"
        android:id="@+id/button22"
        android:layout_gravity="center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="5"
        android:id="@+id/button23"
        android:layout_gravity="center_vertical" />

    <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="6"
        android:id="@+id/button24"
        android:layout_gravity="center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="-"
        android:id="@+id/button34"
        android:layout_column="3" />

</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="7"
        android:id="@+id/button31"
        android:layout_gravity="center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="8"
        android:id="@+id/button32"
        android:layout_gravity="center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="9"
        android:id="@+id/button33"
        android:layout_gravity="center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="+"
        android:id="@+id/button36"
        android:layout_column="3" />

</TableRow>

<TableRow>
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:id="@+id/button28"
        android:layout_gravity="center_vertical" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=". "
        android:id="@+id/button35"
        android:layout_gravity="center_vertical"
        android:layout_column="1" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text ="="
        android:id="@+id/button29"
        android:layout_gravity="center_vertical"
        android:layout_span="2"
        android:layout_column="2" />

    </TableRow>
</TableLayout>

</RelativeLayout>
```

* Ejercicio 2 Modifique la calculadora con RelativeLayout y GridLayout

1.2. Atributos

1.2.1. ViewGroup.LayoutParams

```
https://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html
```

La clase LayoutParams describe las dimensiones de la vista:

- ***android:layout_height*** Especifica la altura de la vista
- ***android:layout_width*** Especifica la anchura de la vista

Los valores posibles son:

- ***FILL_PARENT* | *MATCH_PARENT*** la vista será tan grande como su padre.
- ***WRAP_CONTENT*** la vista será tan grande como su contenido.
- ***Numero*** Dimensión exacta.

Listing 1.6: Atributos TextView

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/hola"  
    android:id="@+id/textView" />
```

* Ejercicio 3 Etiquetas height y width

Podemos hacer lo mismo que realizamos con *TexView* con los demás componentes visuales de Android, por ejemplo *Button*.

1.2.2. ViewGroup.MarginLayoutParams

```
https://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html
```

- ***android:layout_marginBottom*** Especifica espacio parte inferior de la vista.



Figura 1.11: Atributos `layout_height` y `layout_width`.

- ***android:layout_marginTop*** Especifica espacio superior de la vista.
- ***android:layout_marginLeft*** Especifica espacio parte izquierda de la vista.
- ***android:layout_marginRight*** Especifica espacio parte derecha de la vista.
- ***android:layout_margin*** Especifica espacio en todas las posibilidades.

android:layout_marginStart y *android:layout_marginEnd* son equivalentes a *android:layout_marginLeft* y *android:layout_marginRight* para idiomas de izquierda a derecha.

1.2. ATRIBUTOS

33

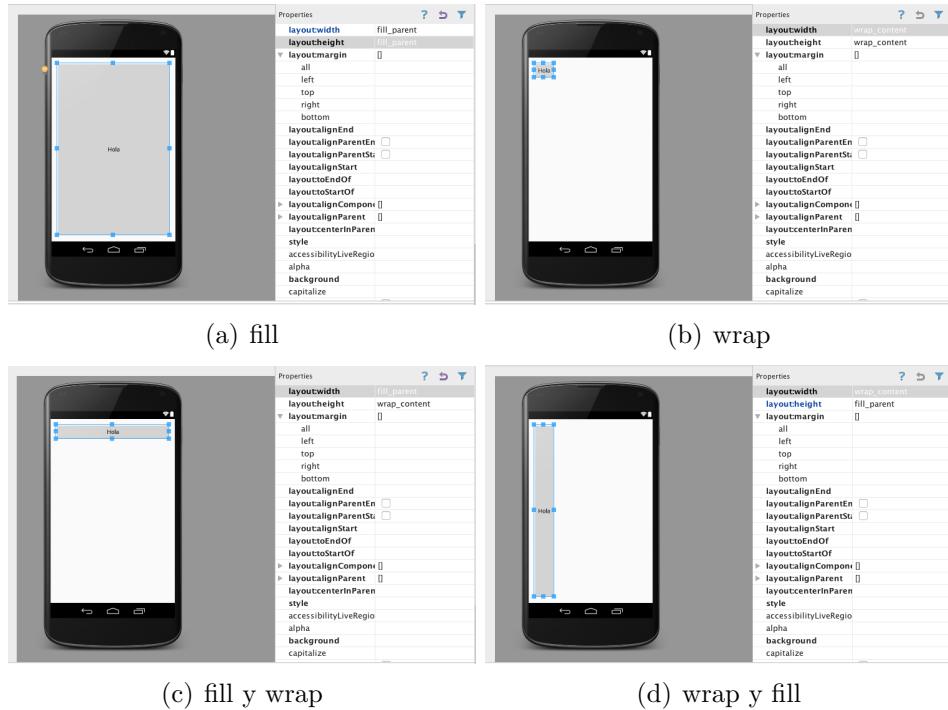


Figura 1.12: Atributos layout_height y layout_width.

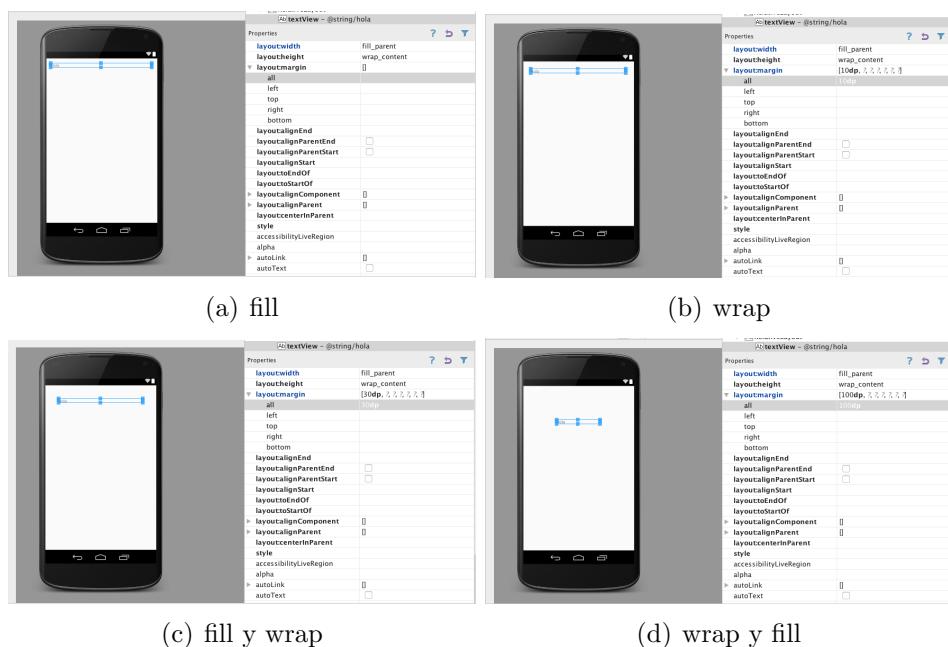


Figura 1.13: Atributos layout_margin.

1.3. Estilo y Temas

```
https://developer.android.com/guide/topics/ui/themes.html
```

1.3.1. Estilo

Un estilo (*Styles*) es una colección de propiedades que especifican cómo visualizaremos una Vista (View) o ventana.

Un estilo puede especificar propiedades como *height*, *padding*, *font color*, *font size*, *background color*, ... Es definido como un recurso XML y fuera del XML que especifica el *layout*.

Listing 1.7: TextView sin Style

```
<TextView  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:textColor="#00FF00"  
    android:typeface="monospace"  
    android:text="@string/hello" />
```

El código 1.7 muestra el componente TextView con sus propiedades. El código 1.8 tiene una representación visual equivalente al código 1.7 pero utilizando un estilo que es definido en el código 1.9.

Listing 1.8: TextView con Style

```
<TextView  
    style="@style/CodeFont"  
    android:text="@string/hello" />
```

Listing 1.9: Style

```
<resources>  
    <style name="CodeFont" parent="@android:style/  
        TextAppearance.Medium">  
        <item name="android:layout_width">fill_parent</item>  
        <item name="android:layout_height">wrap_content</item>  
        <item name="android:textColor">#00FF00</item>  
        <item name="android:typeface">monospace</item>  
    </style>  
</resources>
```

Vamos a ver el ejemplo en Android Studio. La figura 1.7 muestra el código 1.14 en el layout.

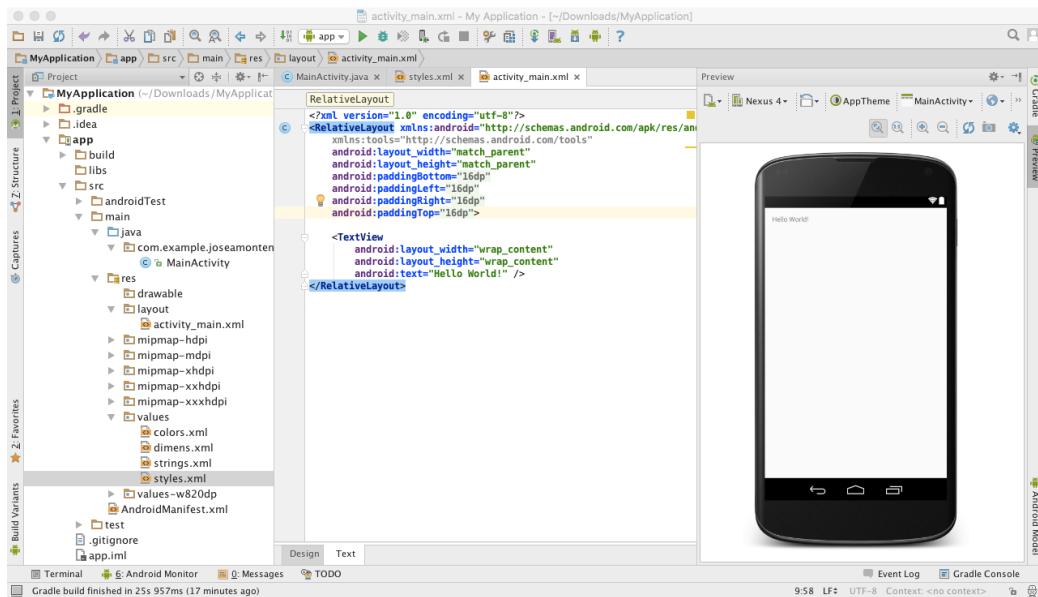


Figura 1.14: TextView Sin Style

Ahora añadimos el código 1.9 en el archivo *res/values/styles.xml*, tal y como muestra la figura 1.15.

Finalmente modificamos el TexView para incluir el estilo y quedará como muestra la figura 1.16.

Herencia

El atributo *parent* en el elemento `<style>` permite especificar un estilo del cual podemos heredar propiedades. Podemos hacer uso de esta característica para heredar las propiedades de un estilo existente y entonces definir solamente las propiedades que quieras modificar o añadir.

Por ejemplo si queremos modificar un estilo existente en la plataforma, lo haríamos así:

```

<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>

```

Por otro lado, si queremos heredar de estilos que hemos definido, no es necesario utilizar el atributo *parent*. En su lugar, pondremos el prefijo del estilo que queremos heredar como prefijo del estilo que queremos modificar, separados por un punto. Por ejemplo, para modificar el estilo *CodeFont* y añadir el color Red, podríamos hacer los siguiente:

1.3. ESTILO Y TEMAS

37

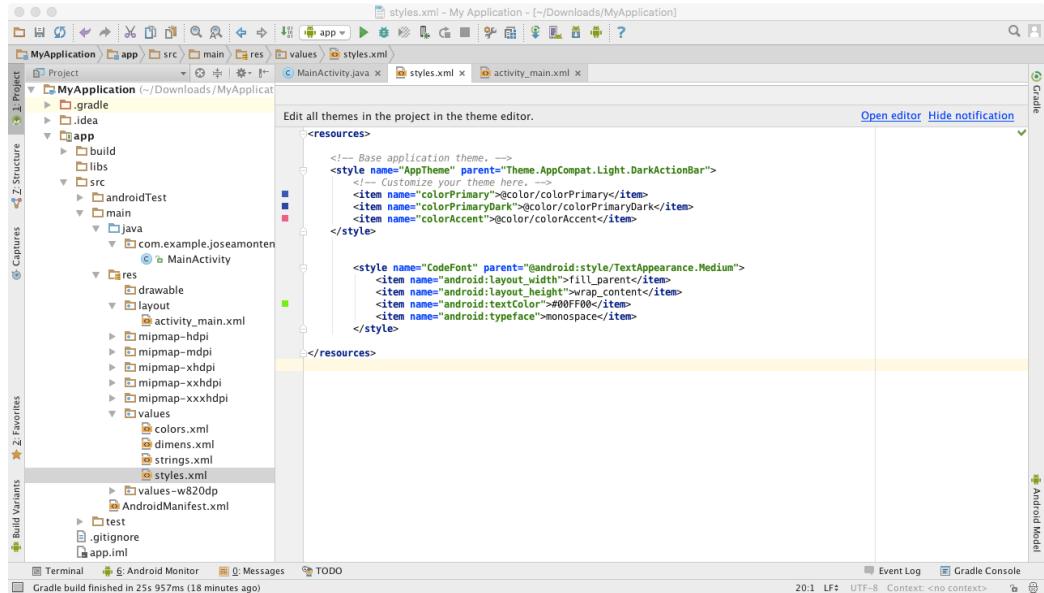


Figura 1.15: Style

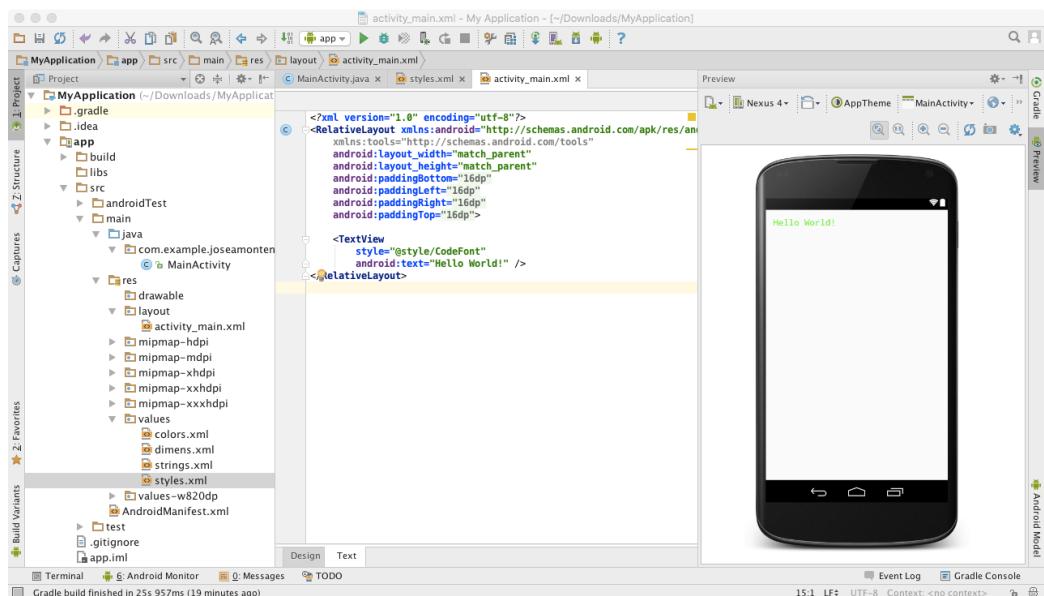


Figura 1.16: TextView Con Style

```

<style name="CodeFont.Red">
    <item name="android:textColor">#FF0000</item>
</style>

```

1.3.2. Temas

Un tema es un estilo aplicado a una Actividad o aplicación, más que a una vista individual. Cuando un estilo es aplicado como un tema, cada vista en la Actividad o aplicación aplicará cada propiedad del estilo que soporta.

Para aplicar un tema, es necesario especificar en el manifiesto Android de la aplicación el atributo *android:theme* al elemento *<activity>* o *<application>*.

```
<activity android:theme="@android:style/Theme.Translucent">
```

```
<application android:theme="@android:style/Theme.Translucent">
```

También es posible aplicar un estilo personalizado, modificando los existentes, por ejemplo:

```
<application android:theme="@style/CustomTheme">
```

Y añadiendo el estilo que queremos y que modifica a un estilo predefinido.

```
1 <color name="custom_theme_color">#b0b0ff</color>
2 <style name="CustomTheme" parent="android:Theme.Light">
3   <item name="android:windowBackground">@color/custom_theme_color</item>
4   <item name="android:colorBackground">@color/custom_theme_color</item>
5 </style>
```

El editor de AndroidStudio permite modificar los estilos para ver su apariencia antes de aplicarlos.

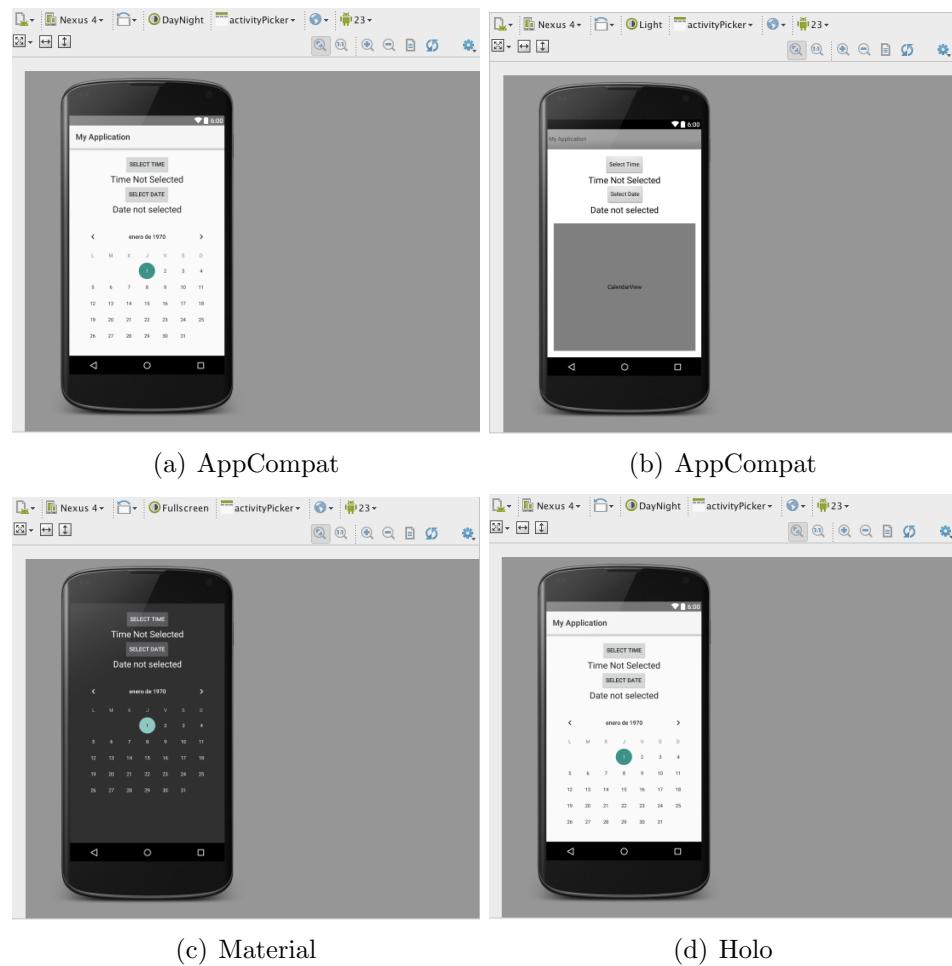


Figura 1.17: Temas en Android Studio

1.4. Widgets

1.4.1. RadioButton y RadioGroup

<https://developer.android.com/guide/topics/ui/controls/radiobutton.html?hl=es>

<https://developer.android.com/reference/android/widget/RadioButton.html?hl=es>

Radiobuttons permiten al usuario seleccionar una opción de un conjunto. Son normalmente usados cuando los elementos del conjunto son mutuamente exclusivos y el usuario necesita ver todas las opciones. Si no fuera necesario seleccionar todas las opciones podemos seleccionar un *spinner*.

Para que un grupo sea mutuamente exclusivo deben de ser agrupados en un *RadioGroup*. Agrupando todos los radiobuttons, solamente uno estará seleccionado a la vez.

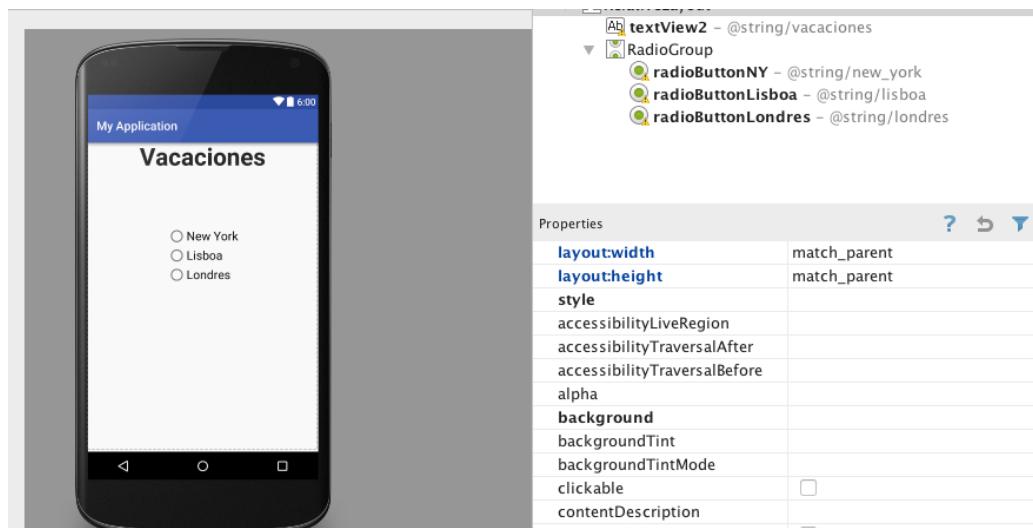


Figura 1.18: Ejemplo Vacaciones RadioButton

Listing 1.10: XML RadioButton

```

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Vacaciones" />

    <RadioGroup
        android:id="@+id/radioGroup1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <RadioButton
            android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New York" />

        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Lisboa" />

        <RadioButton
            android:id="@+id/radioButton3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Londres" />
    </RadioGroup>
</RelativeLayout>

```

```
        android:text="@string/vacaciones"
        android:id="@+id/textView2"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="40dp"
        android:textStyle="bold" />

    <RadioGroup
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true">

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/new_york"
            android:id="@+id radioButtonNY"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="91dp"
            android:textSize="20dp"
            android:checked="false"
            android:onClick="onRadioButtonClicked" />

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/lisboa"
            android:id="@+id radioButtonLisboa"
            android:textSize="20dp"
            android:checked="false"
            android:layout_below="@+id.radioButtonNY"
            android:layout_alignLeft="@+id radioButtonNY"
            android:layout_alignStart="@+id radioButtonNY"
            android:onClick="onRadioButtonClicked" />

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/londres"
            android:id="@+id radioButtonLondres"
            android:textSize="20dp"
            android:checked="false"
            android:layout_below="@+id radioButtonLisboa"
            android:layout_alignLeft="@+id radioButtonLisboa"
            android:layout_alignStart="@+id radioButtonLisboa"
            android:onClick="onRadioButtonClicked" />
    </RadioGroup>
```

```
</RelativeLayout>
```

Listing 1.11: Java RadioButton

```
package com.example.joseamontenegromontes.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.RadioButton;
import android.widget.Toast;

public class Main2Activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
    }

    public void onRadioButtonClicked(View view) {
        boolean checked = ((RadioButton) view).isChecked();
        StringBuffer ciudad=new StringBuffer();

        switch(view.getId()) {
            case R.id.radioButtonLisboa:
                ciudad.append("Lisboa");
                break;
            case R.id.radioButtonLondres:
                ciudad.append("Londres");
                break;
            case R.id.radioButtonNY:
                ciudad.append("New York");
                break;
        }

        ciudad.insert(0,"Nos vamos de vacaciones a: ");

        Log.d("Debug",ciudad.toString());
        Toast.makeText(this,ciudad,Toast.LENGTH_SHORT).show();
    }
}
```


1.4.2. CheckBox

Checkboxes permiten al usuario seleccionar una o más opciones de un conjunto.

```
https://developer.android.com/guide/topics/ui/controls/checkbox.html?hl=es
```

```
https://developer.android.com/reference/android/widget/CheckBox.html?hl=es
```

En el ejemplo crearemos tres checkbox para seleccionar los ingredientes de una hamburguesa, tal y como muestra la figura 1.19.

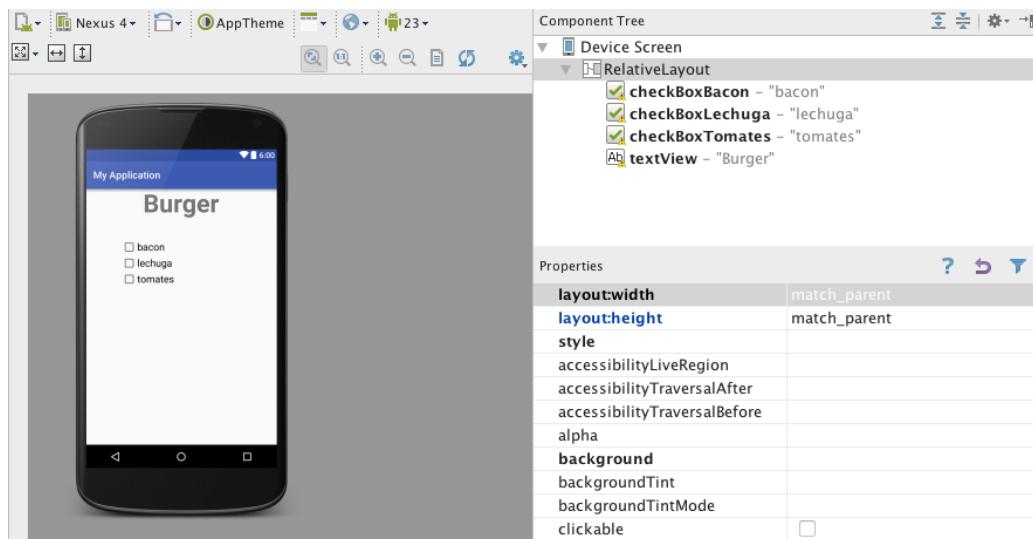


Figura 1.19: Ejemplo Hamburguesa CheckBox

El listado 1.12 muestra la creación de los tres CheckBox. Observase el evento *Onclick* en todos los elementos.

Listing 1.12: XML CheckBox

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

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="bacon"
        android:id="@+id/checkBoxBacon"
        android:layout_marginLeft="72dp"
        android:onClick="onCheckBoxClicked" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="lechuga"
        android:id="@+id/checkBoxLechuga"
        android:layout_marginLeft="72dp"
        android:onClick="onCheckBoxClicked" />

    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="tomates"
        android:id="@+id/checkBoxTomates"
        android:layout_marginLeft="72dp"
        android:onClick="onCheckBoxClicked" />

    <textView ...>
```

```
        android:layout_marginTop="100dp"
        android:checked="false"
        android:onClick="checkBox"
        android:textSize="20dp" />

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="lechuga"
    android:id="@+id/checkBoxLechuga"
    android:layout_below="@+id/checkBoxBacon"
    android:layout_alignLeft="@+id/checkBoxBacon"
    android:checked="false"
    android:onClick="checkBox"
    android:textSize="20dp" />

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="tomates"
    android:id="@+id/checkBoxTomates"
    android:layout_below="@+id/checkBoxLechuga"
    android:layout_alignLeft="@+id/checkBoxLechuga"
    android:checked="false"
    android:onClick="checkBox"
    android:textSize="20dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Burger"
    android:id="@+id/textView"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:textStyle="bold"
    android:textIsSelectable="false"
    android:textSize="50dp" />

</RelativeLayout>
```

El listado 1.13 contiene el método *CheckBox* encargado de recoger los eventos cada vez que un *CheckBox* es pulsado.

Listing 1.13: Java CheckBox

```
package com.example.joseamontenegromontes.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.View;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

    public void checkBox (View view){
        boolean checked = ((CheckBox) view).isChecked();
        StringBuffer Ingrediente= new StringBuffer();

        switch(view.getId()) {

            case R.id.checkBoxLechuga:
                Ingrediente.append("Lechuga");
                break;
            case R.id.checkBoxBacon:
                Ingrediente.append("Bacon");
                break;
            case R.id.checkBoxTomates:
                Ingrediente.append("Tomate");
                break;
        }

        if (checked) Ingrediente.insert(0,"Añade ");
        else Ingrediente.insert(0,"Quita ");

        Log.d("DEBUG",Ingrediente.toString());

        Toast.makeText(this,Ingrediente,Toast.LENGTH_SHORT).
        show();
    }
}
```

* Ejercicio 4 Modificar ejemplo hamburguesa

Modificaremos el ejemplo anterior para mostrar los ingredientes de la hamburguesa, cada vez que pulsamos un checkbox.

1.4.3. Spinner

<https://developer.android.com/guide/topics/ui/controls/spinner.html?hl=es>

<https://developer.android.com/reference/android/widget/Spinner.html>

Spinner proporcionan una forma rápida de seleccionar un elemento de un conjunto. Inicialmente, el spinner muestra un valor seleccionado. Tocando el spinner mostrará un menú desplegable con todos los valores disponibles, de los cuales el usuario podrá elegir uno.

La figura 1.20 muestra el resultado del listado 1.14. En este caso la apariencia final depende de la ejecución del código Java, debido a que los elementos del conjunto son establecidos en ejecución. Ejecutando el código 1.17 o 1.16 (explicaremos posteriormente las diferencias) el spinner es mostrado en el emulador (figura 1.21).

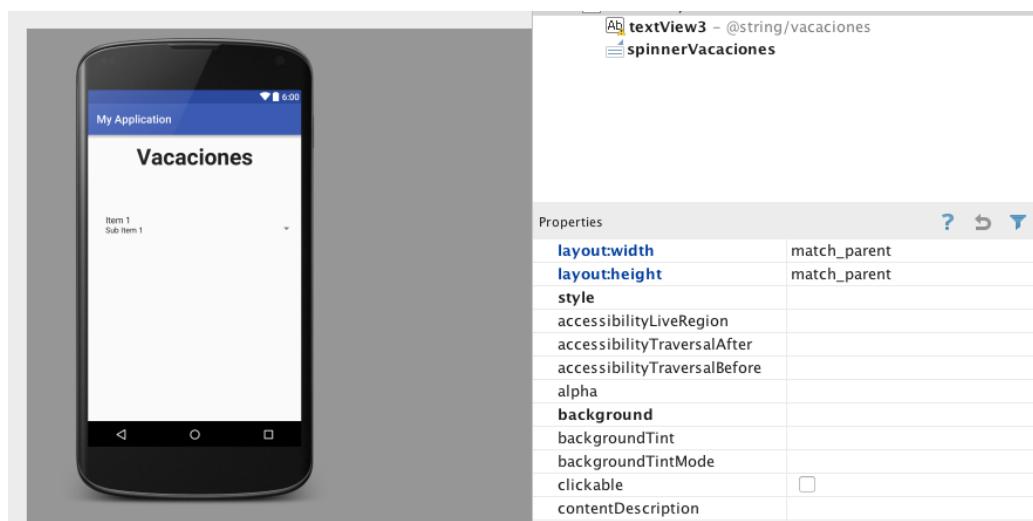


Figura 1.20: Ejemplo Vacaciones Spinner

Listing 1.14: XML Spinner

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegro.montenegro.MainActivity">
```

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/vacaciones"  
    android:id="@+id/textView3"  
    android:textSize="40dp"  
    android:textStyle="bold"  
    android:layout_alignParentTop="true"  
    android:layout_centerHorizontal="true" />  
  
<Spinner  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/spinnerVacaciones"  
    android:layout_alignParentTop="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="119dp" />  
  
</RelativeLayout>
```



Figura 1.21: Ejemplo Viaje Spinner en Emulador

Para establecer el contenido de un spinner es necesario crear un adaptador, para posteriormente asignarlo a un objeto java spinner.

Tenemos dos opciones para establecer el Adaptador. La primera opción es mediante un array de String. Creamos un adaptador de tipo Array (*ArrayAdapter*) y de tipos String (*iString*). En el constructor le pasamos el contexto, un recurso (definido por defecto por Android como *android.R.layout.simple_spinner_item*) y la lista de objetos.

```
https://developer.android.com/reference/android/widget/ArrayAdapter.html
```

Constructor

```
ArrayAdapter(Context context, int resource, T[] objects)
```

Creación del Adaptador mediante Array de String

```
final String[] viaje = new String[]{"New York", "Lisboa", "Madrid"};  
  
ArrayAdapter<String> adaptador =  
    new ArrayAdapter<String>(this, android.R.  
        layout.simple_spinner_item, viaje);
```

Otra opción es crear el adaptador utilizando un array de recursos. En este caso no utilizaremos el constructor, utilizaremos el método *createFromResource*. Los parámetros son similares al constructor, el contexto, y dos recursos. El primero de los recursos es el array que almacena los elementos del *spinner* (listado 1.15). El segundo recurso es idéntico al utilizado en el constructor para visualizar el *spinner*.

Método estático

```
static ArrayAdapter<CharSequence> createFromResource(  
    Context context, int textViewResourceId, int  
    textArrayResourceId)
```

Listing 1.15: Resource Spinner

```
<resources>  
    <string-array name="opcionesVacaciones">  
        <item>NewYork</item>  
        <item>Lisboa</item>  
        <item>Madrid</item>  
    </string-array>  
</resources>
```

La siguiente línea modifica el aspecto visual del spinner.

Establece el recurso visual para crear la vista del spinner

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

```
void setDropDownViewResource(int resource)
```

La figura 1.22 muestra la diferencia visual entre incluir o no el layout.



Figura 1.22: Efecto visual de android.R.layout.simple_spinner_dropdown_item.

Finalmente solamente nos queda obtener el objeto *Spinner* Java y asignarle el adaptador que hemos creado, independientemente de la opción escogida para crear el adaptador.

Asignación del Adaptador al spinner

```
Spinner vacaciones = (Spinner) findViewById(R.id.spinnerVacaciones);
vacaciones.setAdapter(adapter);
```

Hasta este momento hemos creado el elemento visual de un *spinner*. Ahora vamos a obtener los eventos cada vez que un elemento del *spinner* es seleccionado. Para ello utilizaremos el método *setOnItemSelectedListener* de la clase *AdapterView*. La clase *Spinner* hereda de la clase *AdapterView*.

<https://developer.android.com/reference/android/widget/AdapterView.html>

AdapterView

```
void setOnItemSelectedListener(AdapterView.OnItemSelectedListener listener)
```

En la definición podemos observar que necesitamos un objeto de tipo (*listener*) *AdapterView.OnItemSelectedListener*. Es necesario implementar los métodos. El método *onItemSelected* es el método invocado cada vez que se selecciona un elemento de la lista del *Spinner*.

```
https://developer.android.com/reference/android/widget/AdapterView.OnItemSelectedListener.html
```

Asignación del Adaptador al spinner

```
vacaciones.setOnItemSelectedListener(
    new AdapterView.OnItemSelectedListener() {

        public void onItemSelected(AdapterView<?> parent,
            android.view.View v, int position, long id) {
            Toast.makeText(contexto,"Nos vamos a: " + parent
                .getItemAtPosition(position), Toast.
                LENGTH_SHORT).show();
        }

        public void onNothingSelected(AdapterView<?> parent) {
            Toast.makeText(contexto,"Nos quedamos en casa."
                ,Toast.LENGTH_SHORT).show();
        }
    );
}
```

Los listados 1.16 y 1.17 muestran como quedaría el código Java completo con las dos opciones para establecer el contenido del *Spinner*.

Listing 1.16: Java Spiner Array String

```
package com.example.joseamontenegromontes.myapplication;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class Main3Activity extends AppCompatActivity {

    Context contexto;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
```

```

contexto= getBaseContext();

final String[] viaje = new String[]{"New York", "Lisboa
", "Madrid"};

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

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

Spinner vacaciones = (Spinner) findViewById(R.id.
    spinnerVacaciones);
vacaciones.setAdapter(adaptador);

vacaciones.setOnItemSelectedListener(
    new AdapterView.OnItemSelectedListener() {

        public void onItemSelected(AdapterView<?>
            parent, android.view.View v, int
            position, long id) {
            Toast.makeText(contexto,"Nos vamos a:
                " + parent.getItemAtPosition(
                    position),Toast.LENGTH_SHORT).show
                    ();
        }

        public void onNothingSelected(AdapterView<?>
            parent) {
            Toast.makeText(contexto,"Nos quedamos
                en casa." ,Toast.LENGTH_SHORT).
                show();
        }
    });
}

```

Listing 1.17: Java Spinner Resources

```

package com.example.joseamontenegromontes.myapplication;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;

```

```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class Main3Activity extends AppCompatActivity {

    Context contexto;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);

        contexto= getBaseContext();

        ArrayAdapter<CharSequence> adapter =
            ArrayAdapter.createFromResource(this, R.array.
                opcionesVacaciones, android.R.layout.
                simple_spinner_item);

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

        Spinner vacaciones = (Spinner) findViewById(R.id.
            spinnerVacaciones);
        vacaciones.setAdapter(adaptador);

        vacaciones.setOnItemSelectedListener(
            new AdapterView.OnItemSelectedListener() {

                public void onItemSelected(AdapterView<?>
                    parent, android.view.View v, int
                    position, long id) {
                    Toast.makeText(contexto,"Nos vamos a:
                        " + parent.getItemAtPosition(
                        position),Toast.LENGTH_SHORT).show
                        ();
                }

                public void onNothingSelected(AdapterView
                    <?> parent) {
                    Toast.makeText(contexto,"Nos quedamos
                        en casa.",Toast.LENGTH_SHORT).
                    show();
                }
            });
}
```

```

    }
}

```

* Ejercicio 5 Modificar ejemplo Viaje

Introduzca más opciones en el viaje.

Añada código Java necesario para que no se visualice el primer *Toast* hasta que el usuario no seleccione una opción.

Otra posibilidad es añadir un elemento inicial “Escoge una opción” que no sea válido en la selección.

1.4.4. Switch y ToggleButton

<https://developer.android.com/guide/topics/ui/controls/togglebutton.html>

<https://developer.android.com/reference/android/widget/Switch.html>

Switch y *ToggleButton* permite al usuario cambiar entre dos estados.

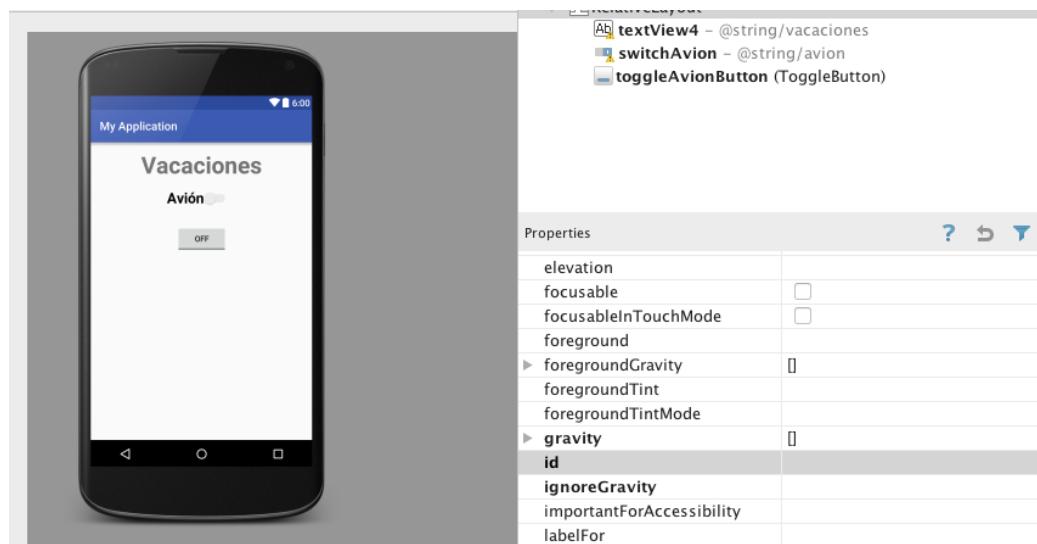


Figura 1.23: Ejemplo Vacaciones ToggleButton

Listing 1.18: XML ToggleButton

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/
res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

```



Figura 1.24: Ejemplo Vacaciones ToggleButton en emulador

```
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/vacaciones"
        android:id="@+id/textView4"
        android:textSize="40dp"
        android:textStyle="bold"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
```

```

<Switch
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/avion"
    android:id="@+id/switchAvion"
    android:layout_marginTop="65dp"
    android:checked="false"
    android:textStyle="bold"
    android:textSize="25dp"
    android:layout_alignRight="@+id/toggleAvionButton"
    android:layout_alignEnd="@+id/toggleAvionButton" />

<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/toggleAvionButton"
    android:layout_below="@+id/switchAvion"
    android:centerHorizontal="true"
    android:layout_marginTop="31dp" />

</RelativeLayout>

```

Listing 1.19: Java ToggleButton

```

package com.example.joseamontenegromontes.myapplication;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.CompoundButton;
import android.widget.Switch;
import android.widget.Toast;
import android.widget.ToggleButton;

public class Main4Activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);

        final Context context= getApplicationContext();
        Switch mySwitch = (Switch) findViewById(R.id.
            switchAvion);

        mySwitch.setOnCheckedChangeListener(new CompoundButton
            .OnCheckedChangeListener() {
            @Override

```

```
        public void onCheckedChanged(CompoundButton
            compoundButton, boolean b) {
                if (b) Toast.makeText(context, "Nos vamos en
                    avión", Toast.LENGTH_SHORT).show();
                else    Toast.makeText(context, "Nos vamos en
                    coche", Toast.LENGTH_SHORT).show();
            }
        });

ToggleButton toggle = (ToggleButton) findViewById(R.id
    .toggleAvionButton);
toggle.setOnCheckedChangeListener(new CompoundButton.
    OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton
        compoundButton, boolean b) {
        if (b) Toast.makeText(context, "Nos vamos en
            avión", Toast.LENGTH_SHORT).show();
        else    Toast.makeText(context, "Nos vamos en
            coche", Toast.LENGTH_SHORT).show();
    }
});
```

* Ejercicio 6 Modificar ejemplo Vacaciones

Modifique los atributos `android:textOff` y `android:textOn` para cambiar el mensaje de los estados. También es posible modificar los mensajes con los métodos `setTextOff` y `setTextOn`.

1.4.5. ImageButton y ImageView

<https://developer.android.com/reference/android/widget/ImageButton.html?hl=es>

<https://developer.android.com/guide/topics/ui/controls/button.html?hl=es>

ImageView muestra una imagen. *ImagenButton* muestra un botón con una imagen (en vez de texto) que puede ser pulsada por el usuario, al igual que un botón. Si queremos que un botón tenga una imagen y un texto deberemos utilizar un objeto *Button* con los atributos *drawableLeft*, *drawableRight*, *drawableTop*, *drawableBottom*, para establecer la imagen a la izquierda, derecha, arriba o abajo del texto.

En el listado 1.20 el botón tiene texto e imagen utilizando el atributo `drawableTop`.

Listing 1.20: XML ImageButton Button y ImageView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

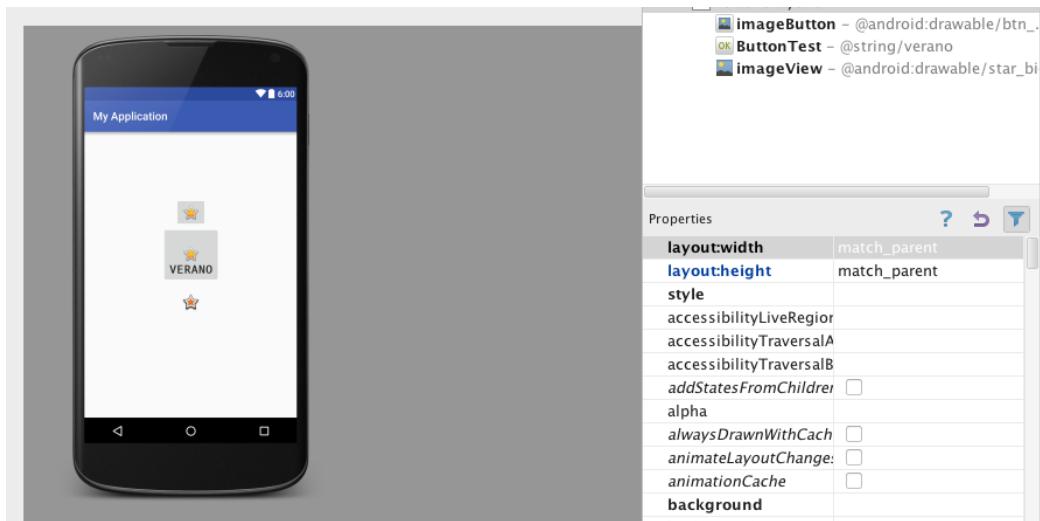


Figura 1.25: Ejemplo Vacaciones ImageButton y ImageView

```

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegro.montes.
        myapplication.MainActivity">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="102dp"
        android:src="@android:drawable/btn_star_big_on"
        android:contentDescription="@string/estrella" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawableTop="@android:drawable/btn_star_big_on
        "
        android:id="@+id/ButtonTest"
        android:paddingTop="32sp"
        android:text="@string/verano"
        android:layout_below="@+id/imageButton"
        android:layout_centerHorizontal="true"

```

```

        android:textStyle="bold"
        android:typeface="monospace"
        android:textSize="20sp" />

<ImageView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/imageView"
    android:src="@android:drawable/star_big_on"
    android:layout_below="@+id/imageButton"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="120dp"
    android:contentDescription="@string/estrella" />
</RelativeLayout>

```

1.4.6. ProgressBar

<https://developer.android.com/reference/android/widget/ProgressBar.html?hl=es>

Indicador visual del progreso de una operación. En uno de sus modos, muestra una barra que indica cuánto ha progresado la tarea que se está realizando.

Puede configurarse cómo indeterminado. En el modo indeterminado, la barra de progreso muestra una animación cíclica sin indicador de progreso. Este modo es usado por aplicaciones que desconocen la longitud de la tarea.

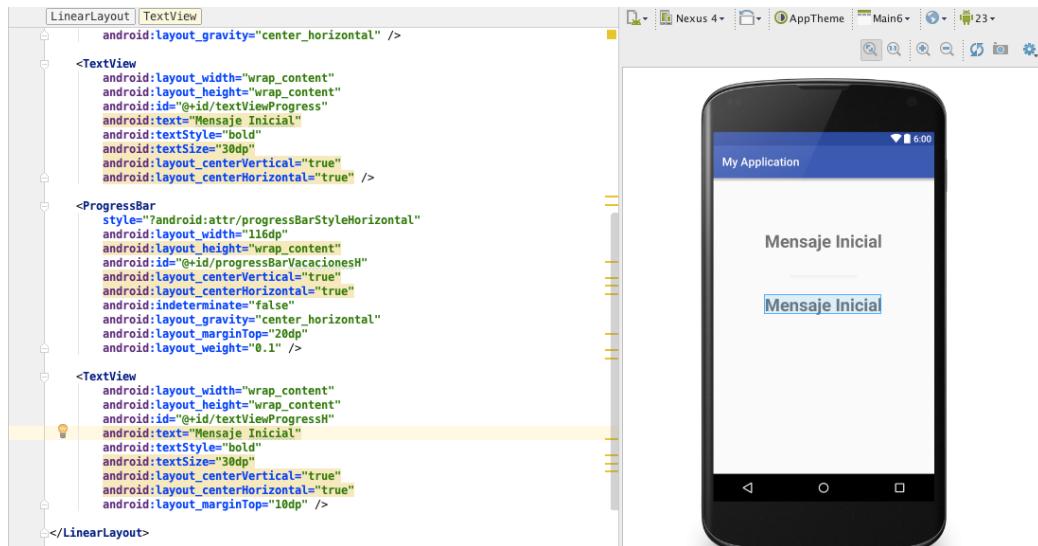


Figura 1.26: Ejemplo Vacaciones Progress

El ejemplo de la figura 1.26 muestra dos *ProgressBar* (uno horizontal y otro circular) y dos *Textview* para mostrar el progreso con un valor numérico. El listado ?? muestra el código de la interfaz gráfica.

Listing 1.21: XML Progress Bar

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegro.
        myapplication.MainActivity"
    android:gravity="top|center"
    android:weightSum="1">

    <ProgressBar
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/progressBarVacaciones"
        android:layout_above="@+id/linearLayout2"
        android:layout_centerHorizontal="true"
        android:layout_gravity="center_horizontal" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textViewProgress"
        android:text="Mensaje Inicial"
        android:textStyle="bold"
        android:textSize="30dp"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <ProgressBar
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="116dp"
        android:layout_height="wrap_content"
        android:id="@+id/progressBarVacacionesH"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:indeterminate="false"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="20dp"
        android:layout_weight="0.1" />
```

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textViewProgressH"
    android:text="Mensaje Inicial"
    android:textStyle="bold"
    android:textSize="30dp"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="10dp" />

</LinearLayout>

```

Normalmente las tareas que hacen uso de un *ProgressBar* son ejecutadas en una hebra distinta a la principal. Las hebras pueden ser creadas mediante *Thread* o *AsyncTask*. En nuestro caso hemos realizado un ejemplo con cada una de ellas.

<https://developer.android.com/guide/components/processes-and-threads.html>

En el siguiente listado observamos la creación de una hebra. El método *run* es la ejecución principal. Este método llama a *doWork* que realiza la tarea (en este caso ponemos la hebra a dormir un tiempo para simular un trabajo). La hebra UI o principal no puede accederse desde una hebra secundaria. Una forma de acceder a ella es implementar el método *post* de la clase *View* con un objeto *Runnable* y a su vez implementar el método *run*. En el método *run* es donde podemos acceder a los componentes visuales. Si el indicar de progreso no ha llegado al máximo actualizaremos el valor y en caso que lleguemos al máximo, hacemos que desaparezca el *ProgressBar* y cambiamos el mensaje del *TextView*.

Creación de una hebra para el ProgressBar

```

new Thread(new Runnable() {

    public void doWork() {
        try {
            Thread.sleep(sleepHebra);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        mProgressStatus++;
    }

    public void run() {
        while (mProgressStatus < maximo) {
            doWork();

            mHandler.post(new Runnable() {
                public void run() {
                    if (mProgressStatus<maximo) {
                        mProgress.setProgress(mProgressStatus);
                        textViewProgress.setText(mProgressStatus
                            + "/" + mProgress.getMax());
                    }
                }
            });
        }
    }
});

```

```

        }
    else{
        textViewProgress.setText(R.string.
            Enhorabuena);
        mProgress.setVisibility(View.GONE);
    }
}
);
}
}
)).start();

```

Si por cualquier motivo queremos realizar el seguimiento del ProgressBar con un *AsynTask*, el siguiente listado muestra cómo sería. *AsynTask* tiene cuatro métodos:

- *onPreExecute*: Método ejecutado una única ocasión, cuando inicia.
- *doInBackground*: Método que realiza la tarea.
- *onPostExecute*: Método ejecutado una única ocasión, cuando finaliza.
- *onProgressUpdate*: Método ejecuta cuando *doInBackground* ejecuta el método *publishProgress*.

Los parámetros utilizados por *AsynTask* son:

1. Tipo pasado al método *doInBackground*
2. Tipo pasado al método *onProgressUpdate*
3. Tipo pasado al método *onPostExecute* y devuelto por el método *doInBackground*

Creación de un *AsynTask* para el *ProgressBar*

```

class ProgressTask extends AsyncTask <Integer, Integer, Void>{

    protected void onPreExecute() {
        mProgressH.setVisibility(View.VISIBLE);
    }

    protected Void doInBackground(Integer... params) {
        for (; mProgressStatusH <= params[0];
            mProgressStatusH++) {
            try {
                Thread.sleep(sleepHebra);
                publishProgress(mProgressStatusH);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
}

```

```
protected void onPostExecute(Void result) {  
    mProgressH.setVisibility(View.GONE);  
    textViewProgressH.setText(R.string.Vacaciones);  
}  
  
protected void onProgressUpdate(Integer... values) {  
    String countdown= Integer.toString(100-values[0]);  
    textViewProgressH.setText(countdown);  
    mProgressH.setProgress(values[0]);  
}  
}
```

El código java al completo de los dos widgets *ProgressBar* y *Texview* es mostrado en el listado 1.22.

Listing 1.22: Java Progress Bar

```
package com.example.joseamontenegromontes.myapplication;  
  
import android.os.AsyncTask;  
import android.os.Handler;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.ProgressBar;  
import android.widget.TextView;  
  
public class Main6Activity extends AppCompatActivity {  
  
    private ProgressBar mProgress,mProgressH;  
    private TextView textViewProgress,textViewProgressH;  
    private int mProgressStatus = 0, mProgressStatusH=0;  
    int sleepHebra=50;  
    int maximo = 100;  
  
    private Handler mHandler = new Handler();  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main6);  
  
        mProgress = (ProgressBar) findViewById(R.id.  
            progressBarVacaciones);  
        mProgressH = (ProgressBar) findViewById(R.id.  
            progressBarVacacionesH);  
        textViewProgress = (TextView) findViewById(R.id.  
            textViewProgress);
```

```

textViewProgressH = (TextView) findViewById(R.id.
    textViewProgressH);

mProgress.setMax(100);

new ProgressTask().execute(maximo);

new Thread(new Runnable() {

    public void doWork(){
        try {
            Thread.sleep(sleepHebra);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    mProgressStatus++;

}

public void run() {
    while (mProgressStatus < maximo) {
        doWork();

        mHandler.post(new Runnable() {
            public void run() {

                if (mProgressStatus<maximo) {
                    mProgress.setProgress(
                        mProgressStatus);
                    textViewProgress.setText(
                        mProgressStatus + "/"
                        + mProgress.getMax());
                }
                else{
                    textViewProgress.setText(R
                        .string.Enhorabuena);
                    mProgress.setVisibility(
                        View.GONE);
                }
            }
        });
    }
}
}).start();
}

class ProgressTask extends AsyncTask <Integer, Integer, Void

```

```

> {
    @Override
    protected void onPreExecute() {
        mProgressH.setVisibility(View.VISIBLE);
    }

    @Override
    protected Void doInBackground(Integer... params) {
        for (; mProgressStatusH <= params[0];
            mProgressStatusH++) {
            try {
                Thread.sleep(sleepHebra);
                publishProgress(mProgressStatusH);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        mProgressH.setVisibility(View.GONE);
        textViewProgressH.setText(R.string.Vacaciones);
    }

    protected void onProgressUpdate(Integer... values) {
        String countdown= Integer.toString(100-values[0]);
        textViewProgressH.setText(countdown);
        mProgressH.setProgress(values[0]);
    }
}
}

```

1.4.7. ProgressDialog

<https://developer.android.com/reference/android/app/ProgressDialog.html?hl=es>

Otra posibilidad es crear un Diálogo que muestre la información de progreso. La figura 1.27 muestra un ejemplo.

En el siguiente listado mostramos cómo realizar un diálogo de progreso con una Thread, de la figura 1.27.

Creación de un ProgressDialog

```
final ProgressDialog progress = new ProgressDialog(this);
progress.setMessage("Vacaciones in Progress");
```

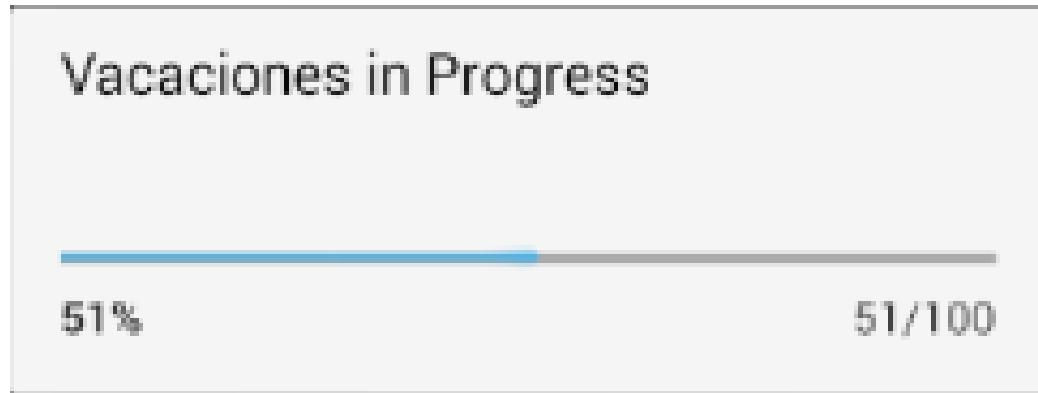


Figura 1.27: Ejemplo Vacaciones in Progress

```

progress.setProgressStyle(ProgressDialog.
    STYLE_HORIZONTAL);
progress.setProgress(0);
progress.show();

final int totalProgressTime = 100;

final Thread hebra = new Thread(new Runnable() {
    public void run() {

        while (jumpTime < totalProgressTime) {
            try {
                Thread.sleep(100);
                jumpTime += 1;
            progress.setProgress(jumpTime);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});

hebra.start();

```

* Ejercicio 7 Modificaciones ProgressDialog

Una vez que finalice el progreso, deberíamos hacer desaparecer el diálogo con el método *dismiss*. Ojo a posibles errores.

* Ejercicio 8 Modificaciones ProgressDialog con AsyncTask

Cree el mismo problema con una AsyncTask.

1.4.8. SeekBar

```
https://developer.android.com/reference/android/widget/SeekBar.html
```

```
https://developer.android.com/reference/android/widget/SeekBar.OnSeekBarChangeListener.html
```

SeekBar es una extensión de *ProgressBar* que añade un elemento que es posible desplazar. El usuario puede desplazar el elemento a la izquierda o derecha para establecer un valor de progreso.

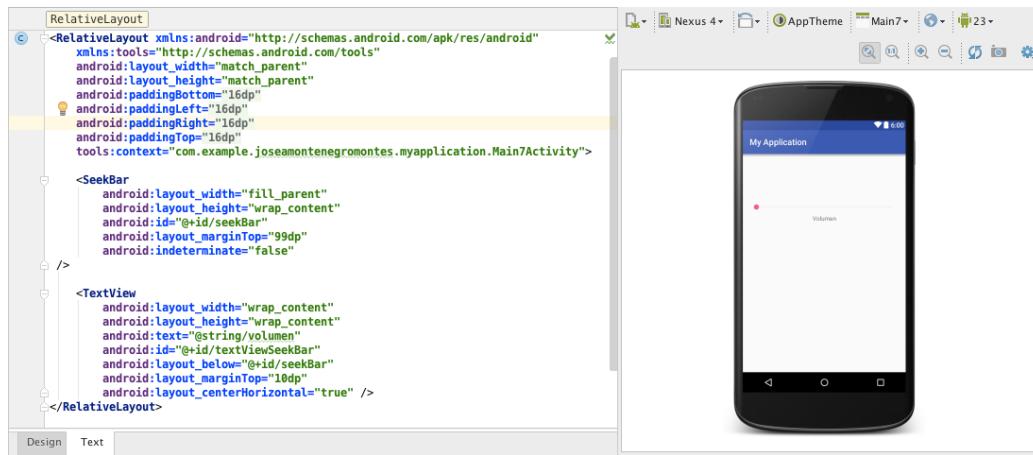


Figura 1.28: Ejemplo SeekBar

El listado 1.23 muestra el listado de la representación de la imagen 1.28 y la figura 1.29 en el emulador.

Listing 1.23: XML SeekBar

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegro.myapplication.MainActivity">

    <SeekBar
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/seekBar"
        android:layout_marginTop="99dp"
        android:indeterminate="false" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/volume"
        android:id="@+id/textViewSeekBar"
        android:layout_below="@+id/seekBar"
        android:layout_marginTop="10dp"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

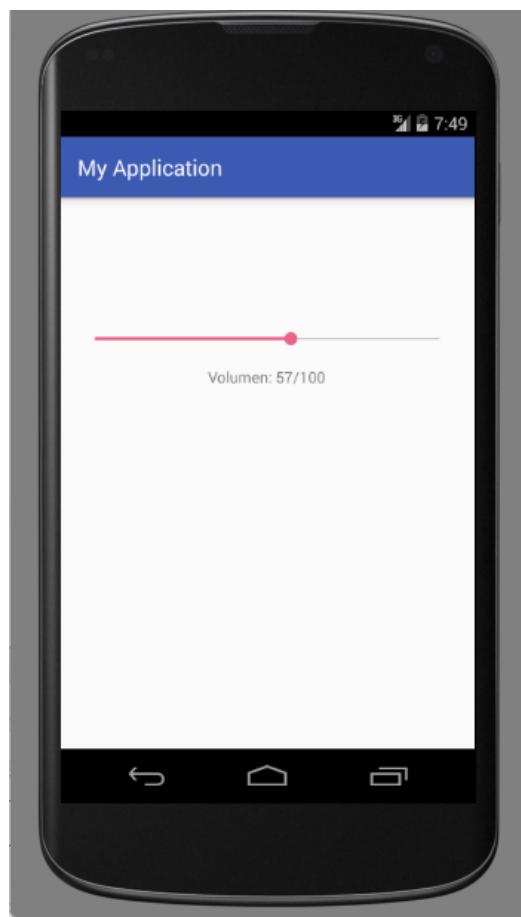


Figura 1.29: EjemploSeekBarEmulador

```
        android:indeterminate="false"
    />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/volumen"
        android:id="@+id/textViewSeekBar"
        android:layout_below="@+id/seekBar"
        android:layout_marginTop="10dp"
        android:layout_centerHorizontal="true"  />
</RelativeLayout>
```

En este caso el listado 1.24 muestra la implementación del Listener del SeekBar, con sus tres métodos.

- ***onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)***. Notifica que el nivel de progreso ha sido modificado. La variable *int progress* indica el nivel actual.

- ***onStartTrackingTouch(SeekBar seekBar)***. Notifica que el usuario ha comenzado la interacción con la vista.
- ***onStopTrackingTouch(SeekBar seekBar)***. Notifica que el usuario ha dejado de interactuar con la vista.

Listing 1.24: Java SeekBar

```
package com.example.joseamontenegromontes.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

public class Main7Activity extends AppCompatActivity {

    private SeekBar seekBar;
    private TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main7);

        seekBar = (SeekBar) findViewById(R.id.seekBar);
        textView = (TextView) findViewById(R.id.
                textViewSeekBar);

        seekBar.setOnSeekBarChangeListener(new SeekBar.
                OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
                    i, boolean b) {
                textView.setText("Volumen: " + i + "/" +
                        seekBar.getMax());
            }
        });

        @Override
        public void onStartTrackingTouch(SeekBar seekBar)
        {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
```

```

        }
    });

}
}

```

1.4.9. RatingBar

<https://developer.android.com/reference/android/widget/RatingBar.html>

RatingBar es una extensión de *SeekBar* y *ProgressBar* que muestra una valoración con estrellas. El usuario puede cambiar la valoración señalando más o menos estrellas.

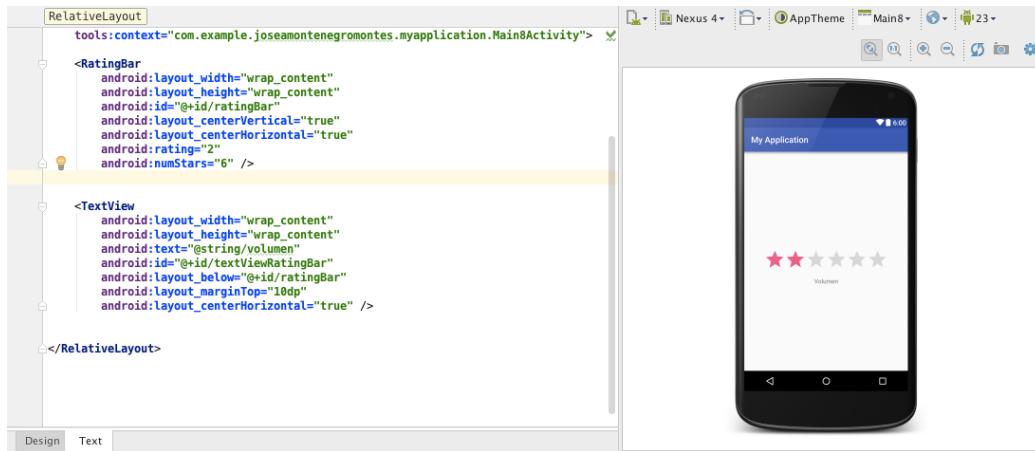


Figura 1.30: Ejemplo RatingBar

Podemos configurar el widget y establecer el número de estrellas. Por ejemplo en el listado 1.25, establecemos seis estrellas (`android:numStars="6"`) y una valoración de dos (`android:rating="2"`).

Listing 1.25: XML RatingBar

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegro.myapplication.MainActivity">

```

```
<RatingBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/ratingBar"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:rating="2"
    android:numStars="6" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/volumen"
    android:id="@+id/textViewRatingBar"
    android:layout_below="@+id/ratingBar"
    android:layout_marginTop="10dp"
    android:layout_centerHorizontal="true" />

</RelativeLayout>
```

En el código en Java 1.26 establecemos en el listener `setOnRatingBarChangeListener`, el método `onRatingChanged` que se ejecutará cuando se modifica el `RatingBar`.

Listing 1.26: Java RatingBar

```
package com.example.joseamontenegromontes.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.RatingBar;
import android.widget.TextView;

public class Main8Activity extends AppCompatActivity {
    private RatingBar ratingbar;
    private TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main8);
        textView = (TextView) findViewById(R.id.
                textViewRatingBar);
        ratingbar = (RatingBar) findViewById(R.id.ratingBar);

        ratingbar.setOnRatingBarChangeListener(new RatingBar.
            OnRatingBarChangeListener() {
            @Override
```

```

        public void onRatingChanged(RatingBar ratingBar,
            float v, boolean b) {
            textView.setText("Valoración: " + v );
        }
    );
}

}

```

1.4.10. WebView

<https://developer.android.com/reference/android/webkit/WebView.html>

WebView es una vista que muestra página web. Permite crear un navegador web o mostrar contenido html en una actividad.

Es necesario incluir en el manifiesto el siguiente permiso:

```
<uses-permission android:name="android.permission.
INTERNET" />
```

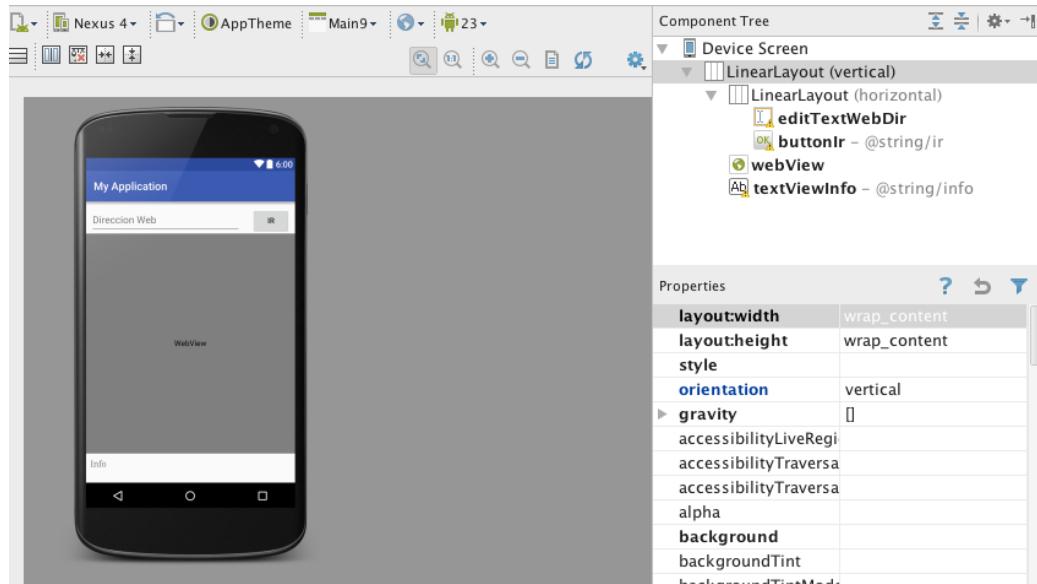


Figura 1.31: Ejemplo webView

El listado ?? muestra un ejemplo con *webView*, donde el usuario introduce una URI en un *EditText* y tras pulsar un botón es mostrada en el *webView*. Información sobre el tiempo de carga de la página es mostrada en un *TextView*.

Listing 1.27: XML WebView

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

<LinearLayout android:orientation="vertical"
    android:layout_height="wrap_content"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <EditText
            android:layout_width="274dp"
            android:layout_height="fill_parent"
            android:id="@+id/editTextWebDir"
            android:autoLink="web"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp"
            android:layout_marginRight="20dp"
            android:inputType="textUri"
            android:hint="Direccion Web" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/ir"
            android:id="@+id/buttonIr"
            android:layout_marginTop="10dp"
            android:layout_marginRight="10dp"
            android:onClick="irDir" />

    </LinearLayout>

    <WebView
        android:id="@+id/webView"
        android:layout_height="400dp"
        android:layout_width="fill_parent" />

    <TextView
        android:layout_width="fill_parent"
        android:text="@string/info"
        android:id="@+id/textViewInfo"
        android:layout_marginBottom="20dp"
        android:layout_marginTop="10dp"
        android:layout_height="fill_parent"
```

```
    android:typeface="serif"
    android:textSize="15sp"
    android:layout_marginLeft="10dp" />
</LinearLayout>
```

Podemos modificar el comportamiento de *WebView* de varias formas:

- Creando y estableciendo una subclase de *WebChromeClient*. Trata las información sobre el progreso de las páginas y las aletas de JavaScript.
- Creando y estableciendo una subclase de *WebViewClient*. Trata eventos y errores sobre la página.
- Modifica *WebSettings*.

En el listado ?? establecemos inicialmente que la web se ajuste al tamaño del *webView*, concretamente con el siguiente código.

Modificación de Ajustes

```
WebSettings webSettings = webView.getSettings();
webSettings.setLoadWithOverviewMode(true);
webSettings.setUseWideViewPort(true);
```

Además hemos establecido una clase de *WebViewClient* para implementar los métodos:

- *onPageStarted*. Ejecuta cuando comienza la petición de la página.
- *onPageFinished*. Ejecuta cuando finaliza la petición de la página.
- *onReceivedError*. Ejecuta cuando se produce un error en la petición de la página.

Modificación de Ajustes

```
webView.setWebViewClient(new MyWebViewClient());
```

<https://developer.android.com/reference/android/webkit/WebViewClient.html>

<https://developer.android.com/reference/android/webkit/WebChromeClient.html>

Más detalle en el funcionamiento de la aplicación puede verse en el código 1.28.

Listing 1.28: Java WebView

```
package com.example.joseamontenegromontes.widget;

import android.graphics.Bitmap;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
```

```
import android.webkit.URLUtil;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class Main9Activity extends AppCompatActivity {

    WebView webView;
    EditText webDir;
    TextView dir;
    boolean errorLoad=false;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main9);

        webView = (WebView)         findViewById(R.id.webView);
        webDir   = (EditText)       findViewById(R.id.
                editTextWebDir);
        dir      = (TextView)       findViewById(R.id.textViewInfo
            );
        webView.setWebViewClient(new MyWebViewClient());

        WebSettings webSettings = webView.getSettings();
        webSettings.setLoadWithOverviewMode(true);
        webSettings.setUseWideViewPort(true);

    }

    private class MyWebViewClient extends WebViewClient {

        private long loadTime;

        @Override
        public void onPageStarted(WebView view, String url,
            Bitmap favicon) {
            super.onPageStarted(view, url, favicon);
            Log.d("DEBUG onPageStarted",url);

            if (!errorLoad){
                loadTime = System.currentTimeMillis();
            }
        }
    }
}
```

```

        dir.setText(url+" has started loading...");  

    }  
  

}  
  

@Override  

public void onPageFinished(WebView view, String url) {  

    super.onPageFinished(view, url);  
  

    Log.d("DEBUG onPageFinished",url);  
  

    if (!errorLoad){  

        loadTime = System.currentTimeMillis() - this.  

            loadTime;  
  

        String time = new SimpleDateFormat("mm:ss:SSS",  

            Locale.getDefault())  

            .format(new Date(this.loadTime));  

        dir.setText(url+" has finished loading in " +  

            time);  

    }  
  

}  
  

public void onReceivedError(WebView view, int  

    errorCode, String description, String failingUrl){  

    String summary = "<html><body>" +  

        "<br><br><center><H1>Upps!!! </H1>" +  

        "<H2> Error "+description+" </H2>" +  

        "<br><br><H3>" +failingUrl+" </H3></center  

        >" +  

        "</body></html>";  
  

    webView.loadData(summary, "text/html", null);  

    dir.setText("Error in "+ failingUrl);  

    errorLoad= true;  

}
}  
  

public void irDir( View v){  
  

    String dirStr= webDir.getText().toString();  
  

    if (URLUtil.isValidUrl(dirStr)) {  

        errorLoad=false;  

        webView.loadUrl(dirStr);  

    }  

    else {  

        Toast.makeText(getApplicationContext(), "URL No vá

```

```
    lida", Toast.LENGTH_LONG).show();
    dir.setText(dirStr+" URL No válida");
}
Log.d("DEBUG irDir",dirStr);
}
}
```

1.5. Containers

1.5.1. ListView

<https://developer.android.com/guide/topics/ui/layout/listview.html>

ListView es una vista de grupo que muestra una lista de elementos que puede aplicarle un scroll. La lista de elementos son insertados automáticamente utilizando un Adaptador.

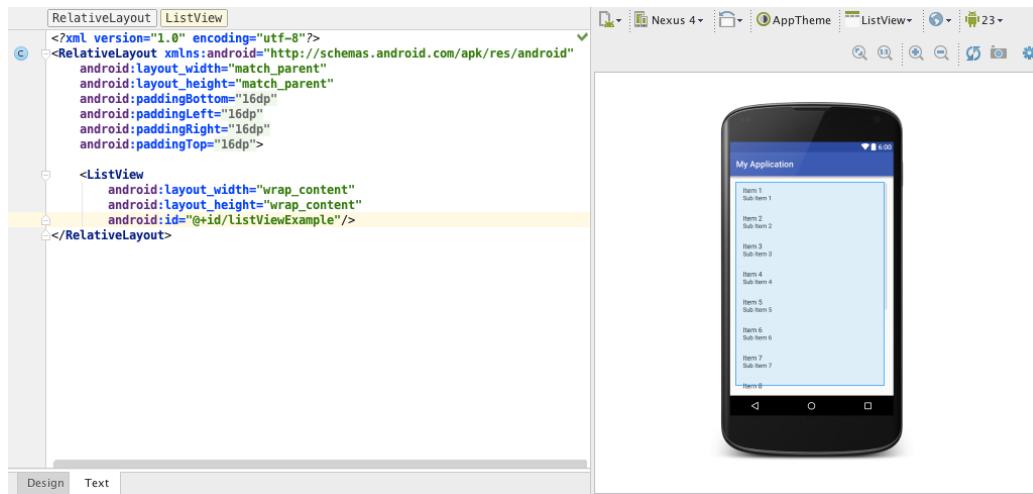


Figura 1.32: Ejemplo ListView

El diseño visual no incluye información relevante, tal y como muestra la figura 1.32. Sin embargo en el código java (listado 1.29) observamos que inicialmente obtenemos el objeto visual *listview* mediante el método *findViewById*. Lo más destacable del código es la utilización del adaptador de Cadena (*ArrayAdapter*), que nos proporciona por defecto Android, de la misma forma que hacíamos con el *Spinner*.

Listing 1.29: Java ListViewActivity

```

package com.example.joseamontenegromontes.widget;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

public class ListViewActivity extends AppCompatActivity {

```

```

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

    ListView listview = (ListView) findViewById(R.id.
        listViewExample);

    String[] values = new String[] {
        "Albacete", "Alicante", "Almería", "Álava", "Asturias", "Á
        vila", "Badajoz", "Balears, Illes",
        "Barcelona", "Bizkaia", "Burgos", "Cáceres", "Cá
        diz", "Cantabria", "Castellón",
        "Ciudad Real", "Córdoba", "Coruña, A", "Cuenca
        ", "Gipuzkoa", "Girona", "Granada",
        "Guadalajara", "Huelva", "Huesca", "Jaén", "León
        ", "Lleida", "Lugo", "Madrid", "Málaga",
        "Murcia", "Navarra", "Ourense", "Palencia",
        "Palmas, Las", "Pontevedra", "Rioja, La",
        "Salamanca", "Santa Cruz de Tenerife", "Segovia
        ", "Sevilla", "Soria", "Tarragona",
        "Teruel", "Toledo", "Valencia", "Valladolid",
        "Zamora", "Zaragoza", "Ceuta", "Melilla"};
    }

    final ArrayList<String> list = new ArrayList<>();
    for (int i = 0; i < values.length; ++i) {
        list.add(values[i]);
    }
    final ArrayAdapter adapter = new ArrayAdapter(this,
        android.R.layout.simple_list_item_1, list);
    listview.setAdapter(adapter);

    listview.setOnItemClickListener(new AdapterView.
        OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView
                , View view, int i, long l) {
                Toast.makeText(getApplicationContext(),
                    "Pulsado posición "+i+" provincia: "+
                    adapterView.getItemAtPosition(i),
                    Toast.LENGTH_LONG).show();
            }
        });
}
}

```

1.5.2. GridView

<https://developer.android.com/guide/topics/ui/layout/gridview.html>

GridView es una vista de grupo que muestra una cuadrícula bidimensional de elementos que puede aplicarle un scroll. La lista de elementos son insertados automáticamente utilizando un Adaptador de la misma forma que hicimos con el *Spinner* y *ListView*.

Asignación del Adaptador al gridView

```
final ArrayAdapter adapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, list);
gridview.setAdapter(adapter);
```

En el ejemplo 1.33 hemos creado un interfaz que incluye un *GridView* y cuatro botones que permiten modificar la configuración del *Gridview*, modificando su atributo *numColumns* para mostrar la información en 1, 2, 3 o 4 columnas (figura 1.34). La información mostrada en el *GridView* es la misma que la mostrada en el *ListView*.

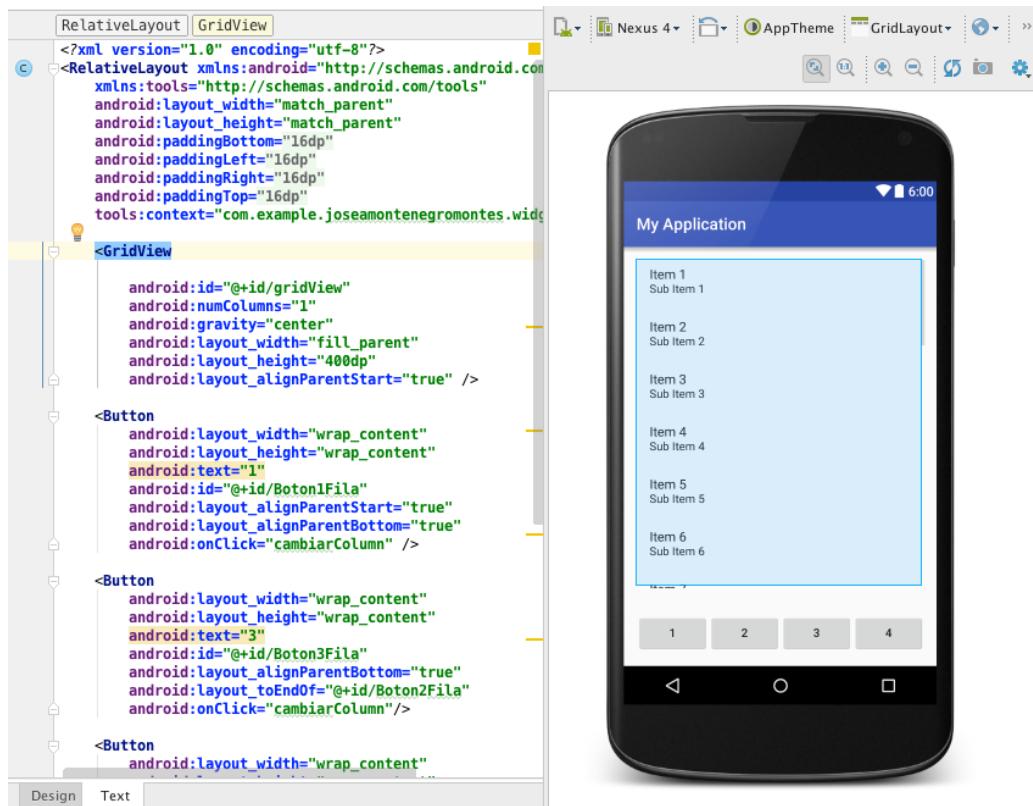


Figura 1.33: Ejemplo GridView

Listing 1.30: XML GridView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".GridView">

    <GridView

        android:id="@+id/gridView"
        android:numColumns="1"
        android:gravity="center"
        android:layout_width="fill_parent"
        android:layout_height="400dp"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="1"
        android:id="@+id/Boton1Fila"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"
        android:onClick="cambiarColumn" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="3"
        android:id="@+id/Boton3Fila"
        android:layout_alignParentBottom="true"
        android:layout_toEndOf="@+id/Boton2Fila"
        android:onClick="cambiarColumn"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="2"
        android:id="@+id/Boton2Fila"
        android:layout_alignParentBottom="true"
        android:layout_toEndOf="@+id/Boton1Fila"
        android:onClick="cambiarColumn"/>

    <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="4"
        android:id="@+id/Boton4Fila"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:onClick="cambiarColumn"/>
</RelativeLayout>
```

Cómo mencionamos anteriormente, en el código 1.31 observamos que el adaptador que utilizamos es de nuevo un *ArrayAdapter* que nos proporciona el sistema.

Listing 1.31: Java GridViewActivity String Adapter

```
package com.example.joseamontenegromontes.widget;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AbsListView;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;

import java.util.ArrayList;

public class GridViewActivity extends AppCompatActivity {
    android.widget.GridView gridview;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_grid_view);

        gridview = (android.widget.GridView) findViewById(R.id
                .gridView);

        String[] values = new String[] { "Albacete", "Alicante",
                ", "Almería", " Álava", "Asturias", "Ávila",
                "Badajoz", "Balears, Illes", "Barcelona", " Bizkaia",
                "Burgos", "Cáceres", "Cádiz",
                "Cantabria", "Castellón", "Ciudad Real", "Có
                rdoba", "Coruña, A", "Cuenca ",
                "Gipuzkoa", "Girona", "Granada", "Guadalajara", "
                Huelva", "Huesca", "Jaén", "León",
                "Lleida", "Lugo", "Madrid", "Málaga", "Murcia", "
```

```

        Navarra", "Ourense", "Palencia", "Palmas,
        Las",
        "Pontevedra", "Rioja, La", "Salamanca", "Santa
        Cruz de Tenerife",
        "Segovia", "Sevilla", "Soria", "Tarragona", "
        Teruel", "Toledo", "Valencia",
        "Valladolid", "Zamora", "Zaragoza", "Ceuta", "
        Melilla"};
```

```

    final ArrayList<String> list = new ArrayList<>();
    for (int i = 0; i < values.length; ++i) {
        list.add(values[i]);
    }
    final ArrayAdapter adapter = new ArrayAdapter(this,
        android.R.layout.simple_list_item_1, list);
    gridview.setAdapter(adapter);
}
```

```

public void cambiarColumn (View v){

    switch(v.getId()){
        case R.id.Boton1Fila: gridview.setNumColumns(1);
            break;
        case R.id.Boton2Fila: gridview.setNumColumns(2);
            break;
        case R.id.Boton3Fila: gridview.setNumColumns(3);
            break;
        case R.id.Boton4Fila: gridview.setNumColumns(4);
            break;
    }
}
}
```

En este caso vamos a modificar la clase Java para mostrar elementos distintos a cadenas de texto, por ejemplo, imágenes. Para ello es necesario crear un nuevo Adaptador que hereda de *BaseAdapter*.

<https://developer.android.com/reference/android/widget/BaseAdapter.html>

Creación de un Adaptador específico

```
private class ImageAdapter extends BaseAdapter
```

En la creación del nuevo adaptador es necesario la implementación de algunos métodos.

- **getCount**. Establece el número de elementos que contiene la vista, necesario para saber cuántos elementos pintar.

- **getItem**. Devuelve el objeto especificado en la posición que pasamos por parámetro. En nuestro ejemplo no lo hemos utilizado.
 - **getItemId**. Similar al anterior pero devuelve el id del objeto seleccionado. Tampoco es utilizado en el ejemplo.
 - **getView**. Este método crea una vista (*View*) para cada imagen que añadimos al adaptador. Uno de los parámetros que tiene es una vista, la primera vez que es invocado cada vista no estará creada (null) con lo cual será necesario crear un objeto específico, en este caso un objeto *ImageView* (ver descripción del objeto *ImageView*).
- En el caso que el objeto ya hubiera sido creado devolvemos el objeto reciclado y ponemos la imagen correspondiente.

Método getView que muestra cada objeto del GridView

```
public View getView(int position, View convertView, ViewGroup
parent) {
    ImageView imageView;
    if (convertView == null) { //Nuevo es necesario
        crear
        imageView = new ImageView(mContext);
        imageView.setLayoutParams(new AbsListView.
            LayoutParams(85, 85));
        imageView.setScaleType(ImageView.ScaleType.
            CENTER_CROP);
        imageView.setPadding(8, 8, 8, 8);
    } else { //Reciclado, ya creado
        imageView = (ImageView) convertView;
    }

    imageView.setImageResource(mThumbIds[position]);
    return imageView;
}
```

Asignamos a gridview el Adaptador específico

```
gridview.setAdapter(new ImageAdapter(this));
```

El resultado es el mostrado ahora lo podemos ver en la figura 1.35.

El código Java completo es mostrado en el listado 1.32.

Listing 1.32: Java GridViewActivity Image Adapter

```
package com.example.joseamontenegromontes.widget;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AbsListView;
```

```
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;

public class GridView extends AppCompatActivity {
    android.widget.GridView gridview;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_grid_view);

        gridview = (android.widget.GridView) findViewById(R.id
                .gridView);

        gridview.setAdapter(new ImageAdapter(this));

        gridview.setOnItemClickListener(new AdapterView.
                OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent,
                    View v,
                    int position, long id) {
                Toast.makeText(getApplicationContext(), "" +
                        position,
                        Toast.LENGTH_SHORT).show();
            }
        });
    }

    public void cambiarColumnn (View v){

        switch(v.getId()){
            case R.id.Boton1Fila: gridview.setNumColumns(1);
                break;
            case R.id.Boton2Fila: gridview.setNumColumns(2);
                break;
            case R.id.Boton3Fila: gridview.setNumColumns(3);
                break;
            case R.id.Boton4Fila: gridview.setNumColumns(4);
                break;
        }
    }

    private class ImageAdapter extends BaseAdapter {
        private Context mContext;
```

```
public ImageAdapter(Context c) { //Constructor
    mContext = c;
}

public int getCount() {
    return mThumbIds.length; //Numero de banderas,
                           modificar
}

public Object getItem(int position) { //No utilizado
    return null;
}

public long getItemId(int position) { //No utilizado
    return 0;
}

public View getView(int position, View convertView,
                    ViewGroup parent) {
    ImageView imageView;
    if (convertView == null) { //Nuevo es necesario
        imageView = new ImageView(mContext);
        imageView.setLayoutParams(new AbsListView.
                               LayoutParams(85, 85));
        imageView.setScaleType(ImageView.ScaleType.
                               CENTER_CROP);
        imageView.setPadding(8, 8, 8, 8);
    } else { //Reciclado, ya creado
        imageView = (ImageView) convertView;
    }

    imageView.setImageResource(mThumbIds[position]);
    return imageView;
}

// references to our images
private Integer[] mThumbIds = {
    R.drawable.brazil, R.drawable.canada,
    R.drawable.china, R.drawable.france,
    R.drawable.germany, R.drawable.india,
    R.drawable.italy, R.drawable.japan,
    R.drawable.korea, R.drawable.mexico,
    R.drawable.netherlands, R.drawable.portugal,
    R.drawable.spain, R.drawable.turkey,
    R.drawable.united_kingdom, R.drawable.
                               united_states
};
```

```

    }
}

```

Si queremos que cada elemento del gridView tenga varios elementos (figura 1.36), podemos crear un layout que contenga varios elementos.

Layout de cada elemento del gridView

```

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

    <ImageView
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:id="@+id/grid_image" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/grid_text"
        android:layout_marginTop="1dp" />
</LinearLayout>

```

Ahora debemos modificar el método *getView* del Adaptador para que muestre el layout almacenado en el XML. Esta acción la podemos realizar mediante la clase *LayoutInflater*. Tal y como recomienda la web de Android crearemos una instancia mediante la llamada al sistema. Finalmente, una vez que tenemos una instancia del *LayoutInflater*, llamamos al método *inflate*, al que le indicamos el id donde está el xml del layout y no utilizaremos el segundo parámetro (null).

Una vez establecido el layout de cada elemento del grid, necesitamos establecer sus componentes, en este caso un *TextView* y un *ImageView*.

<https://developer.android.com/reference/android/view/LayoutInflator.html>

Método *getView* que muestra cada objeto del GridView

```

public View getView(int position, View convertView, ViewGroup
    parent) {
    View gridElement=null;

    LayoutInflater inflater = (LayoutInflater) mContext
        .getSystemService(Context.
            LAYOUT_INFLATER_SERVICE);

    if (convertView == null) { //Nuevo es necesario
        crear
        gridElement = inflater.inflate(R.layout.
            grid_element, null);
    }
}

```

```

        TextView textView = (TextView) gridElement.
            findViewById(R.id.grid_text);
        ImageView image = (ImageView)gridElement.
            findViewById(R.id.grid_image);
        textView.setText(mThumbDes[position]);
        image.setImageResource(mThumbIds[position]);
    } else { //Reciclado, ya creado
        gridElement = (View) convertView;
    }

    return gridElement;
}

```

1.5.3. ExpandableListView

<https://developer.android.com/reference/android/widget/ExpandableListView.html>

Una vista que muestra los elementos en una lista de dos niveles (padres e hijos). Este objeto difiere de *ListView* permitiendo dos niveles: grupos los cuales puede ser individualmente expandidos para mostrar sus hijos.

El diseño en el editor visual de Android es muy similar, tal y como muestra la figura 1.37. El código del interfaz visual es detallado en el listado 1.33.

Visualmente en el emulador observamos la ejecución de la aplicación en la figura 1.38. Para que el componente visual tenga esa apariencia es necesario crear un adaptador propietario que hereda de *BaseExpandableListAdapter*. La implementación del adaptador requiere de la implementación de varios métodos:

- *getGroupCount* y *getChildrenCount*: Establece el número de elementos del nivel 1 y nivel 2 respectivamente.
- *getGroup* y *getChild*: Obtiene el elemento de nivel 1 o 2 especificado por el identificador.
- *getGroupId* y *getChildId*: Obtiene el identificador de nivel 1 o 2 especificado por el identificador.
- *getGroupView* y *getChildView*: Son los dos métodos encargados de pintar los elementos. En nuestro caso cada uno hace referencia a los archivos XML que referencian a los elementos de nivel 1 y 2. En nuestro caso, los listados 1.34 y 1.35 muestran como serán representados los elementos.
- *isChildSelectable* y *hasStableIds*: Ambos métodos vienen del interfaz *ExpandableListAdapter*. El primer método permite si el podemos seleccionar el elemento de segundo nivel. Establece si el conjunto de datos que utilizamos en el adaptador es estable, es decir, si modificamos los datos que son visualizadas. Es necesario para actualizar la vista y reflejar los cambios.

Listing 1.33: XML ExpandableList

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegromontes.widget.
        ExpandableListViewActivity">

<ExpandableListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/expandableListView"
    android:layout_centerVertical="true"
    android:layout_alignParentStart="true" />
</RelativeLayout>

```

Listing 1.34: XML ExpandableList

```

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

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/
        textAppearanceLarge"
    android:text="Large Text"
    android:id="@+id/grupo"
    android:paddingLeft="25dp" />
</LinearLayout>

```

Listing 1.35: XML ExpandableList

```

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

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```
        android:textAppearance="?android:attr/  
            textAppearanceLarge"  
        android:text="Large Text"  
        android:id="@+id/item"  
        android:paddingLeft="25dp" />  
</LinearLayout>
```

Finalmente el código 1.36 muestra el manejo de los siguientes eventos:

- *setOnGroupClickListener* y *setOnChildClickListener*: Eventos cuándo seleccionamos un nodo de nivel 1 o 2 respectivamente.
- *setOnGroupExpandListener*: Evento activa cuando el grupo se expande.
- *setOnGroupCollapseListener*: Evento activa cuando el grupo se contrae.

Listing 1.36: Java ExpandableList

```
package com.example.joseamontenegromontes.widget;  
  
import android.app.Activity;  
import android.content.Context;  
import android.graphics.Typeface;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.Gravity;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AbsListView;  
import android.widget.AdapterView;  
import android.widget.BaseExpandableListAdapter;  
import android.widget.ExpandableListView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
public class ExpandableListViewActivity extends  
    AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_expandable_list_view)  
        ;  
  
        ExpandableListView expandableListView = (  
            ExpandableListView) findViewById(R.id.  
            expandableListView);  
        AdapterExpandableListAdapter adapter = new  
            AdapterExpandableListAdapter(this);
```

```
expandableListView.setAdapter(adapter);

expandableListView.setOnChildClickListener(new
    ExpandableListView.OnChildClickListener() {
        @Override
        public boolean onChildClick(ExpandableListView
            expandableListView, View view, int i, int il,
            long l) {
            Toast.makeText(getApplicationContext(), "
                Pulsado padre: "+i+" hijo: "+il,Toast.
                LENGTH_SHORT).show();
            return false;
        }
    });
}

expandableListView.setOnGroupClickListener(new
    ExpandableListView.OnGroupClickListener() {
        @Override
        public boolean onGroupClick(ExpandableListView
            expandableListView, View view, int i, long l)
        {
            Toast.makeText(getApplicationContext(), "
                Pulsado padre: "+i,Toast.LENGTH_SHORT).
                show();
            return false;
        }
    });
}

expandableListView.setOnGroupCollapseListener(new
    ExpandableListView.OnGroupCollapseListener() {
        @Override
        public void onGroupCollapse(int i) {
            Toast.makeText(getApplicationContext(), "
                Collapse: "+i,Toast.LENGTH_SHORT).show();
        }
    });
}

expandableListView.setOnGroupExpandListener(new
    ExpandableListView.OnGroupExpandListener() {
        @Override
        public void onGroupExpand(int i) {
            Toast.makeText(getApplicationContext(), "Expand
                : "+i,Toast.LENGTH_SHORT).show();
        }
    });
}
```

```
private class AdapterExpandableListAdapter extends
    BaseExpandableListAdapter {

    private String[] groups = { "Parent1", "Parent2"};
    private String[][] children = { { "Child1", "Child2" },
        { "Child3", "Child4", "Child5" } };
    Context context;

    AdapterExpandableListAdapter (Context contextp) {
        context=contextp;
    }
    @Override
    public int getGroupCount() {
        return groups.length;
    }

    @Override
    public int getChildrenCount(int i) {
        int count = 0;
        count = children[i].length;
        return count;
    }

    @Override
    public Object getGroup(int i) {
        return groups[i];
    }

    @Override
    public Object getChild(int i, int il) { //Grupo, Hijo
        return children[i][il];
    }

    @Override
    public long getGroupId(int i) {
        return i;
    }

    @Override
    public long getChildId(int i, int il) {
        return il;
    }

    @Override
    public boolean hasStableIds() {
        return true;
    }
```

```
@Override
public View getGroupView(int i, boolean b, View view,
    ViewGroup viewGroup) {

    String Name = (String) getGroup(i);
    if (view == null) {
        LayoutInflator infalInflater = (LayoutInflator
            ) context
            .getSystemService(Context.
                LAYOUT_INFLATER_SERVICE);

        view = infalInflater.inflate(R.layout.
            group_expandable,null);
    }

    TextView item = (TextView) view.findViewById(R.id.
        grupo);
    item.setTypeface(null, Typeface.BOLD);
    item.setText(Name);
    return view;
}

@Override
public View getChildView(int i, int il, boolean b,
    View view, ViewGroup viewGroup) {
    final String NameHijo = (String) getChild(i, il);

    if (view == null) {
        LayoutInflator infalInflater = (LayoutInflator
            ) context
            .getSystemService(Context.
                LAYOUT_INFLATER_SERVICE);
        view = infalInflater.inflate(R.layout.
            item_expandable, null);
    }

    TextView item = (TextView) view.findViewById(R.id.
        item);
    item.setText(NameHijo);
    return view;
}

@Override
public boolean isChildSelectable(int i, int il) {
    return true;
}
```

```
    }  
}
```

1.5.4. TabHost

<https://developer.android.com/reference/android/widget/TabWidget.html>

Contenedor para una vista con pestañas. El objeto tiene dos hijos: un conjunto de etiquetas de las pestañas que el usuario puede seleccionar y un *FrameLayout* que muestra el contenido de la página.

Una vez que seleccionamos el objeto *TabHost* de la paleta de componentes, tal y como muestra la figura 1.32, una serie de elementos son incluidos en la definición del interfaz.

Dentro del elemento *TabHost* android incluye un elemento *LinearLayout* con orientación vertical, que incluye un elemento *TabWidget*, encargado de establecer las pestañas y un objeto *FrameLayout* que almacenara cada una de las lengüetas en su correspondiente *LinearLayout*. En nuestro caso cada lengüeta solo contiene un *TextView*.

Listing 1.37: XML TabHost

```
<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin">  
  
<TabHost  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:id="@+id/tabHost"  
    android:layout_alignParentStart="true"  
    android:layout_alignParentTop="true">  
  
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
  
<TabWidget  
    android:id="@+id/tabs"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"></TabWidget>
```

```
<FrameLayout
    android:id="@+id/tabcontent"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/linearTab1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Tab 1"
            android:id="@+id/textView6" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearTab2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tab 2"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:id="@+id/textView7" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearTab3"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Tab 3"
            android:id="@+id/textView8" />
    </LinearLayout>

```

```
        android:id="@+id/textView8" />
    </LinearLayout>
</FrameLayout>
</LinearLayout>
</TabHost>
</RelativeLayout>
```

Por otro lado, el código java es bastante simple, tal y como observamos en el listado 1.38. Inicialmente obtenemos el objeto *TabHost* mediante el método *findViewById* para después llamar al método *setup*. Si no llamamos a este método se producirá un error *NullPointerException*.

Posteriormente incluiremos cada una de las pestañas disponibles, con su identificador del *LinearLayout* correspondiente. Al finalizar el código observamos que hemos implementado el *Listener* que permite obtener el evento cada vez que pulsamos una pestaña.

Listing 1.38: Java TabHost

```
package com.example.joseamontenegromontes.widget;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TabHost;

public class TabHostActivity extends AppCompatActivity {

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

        TabHost tabs=(TabHost)findViewById(R.id.tabHost);
        tabs.setup();

        TabHost.TabSpec spec1=tabs.newTabSpec("Tab 1");
        spec1.setContent(R.id.linearTab1);
        spec1.setIndicator("TAB1");
        tabs.addTab(spec1);

        TabHost.TabSpec spec2=tabs.newTabSpec("Tab 2");
        spec2.setContent(R.id.linearTab2);
        spec2.setIndicator("TAB2");
        tabs.addTab(spec2);

        TabHost.TabSpec spec3=tabs.newTabSpec("Tab 3");
        spec3.setContent(R.id.linearTab3);
        spec3.setIndicator("TAB3");
        tabs.addTab(spec3);
        tabs.setCurrentTab(0);

        tabs.setOnTabChangedListener(new TabHost.OnTabChangeListener() {
```

```
    @Override  
    public void onTabChanged(String tabId) {  
        Log.i("Tabs", "Pulsada pestaña: " + tabId);  
    }  
});  
  
}  
}
```



Figura 1.34: GridView con distintas configuraciones de columnas.



Figura 1.35: GridView con distintas configuraciones de columnas.



Figura 1.36: GridView con distintas configuraciones de columnas.

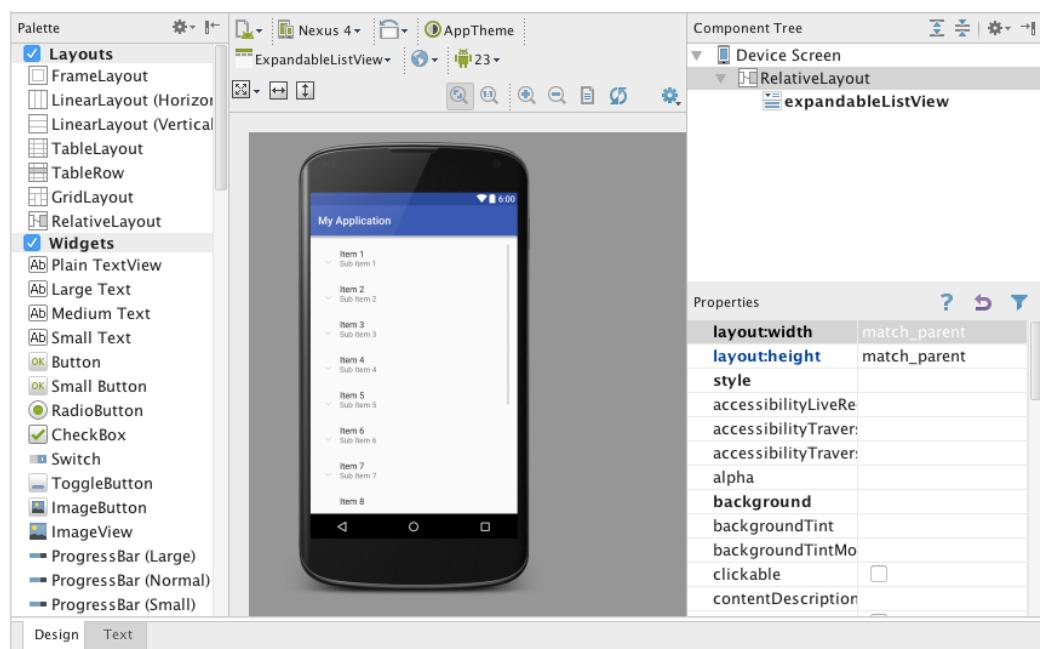


Figura 1.37: Ejemplo ExpandableListView



Figura 1.38: Ejemplo ExpandableListViewEmulador

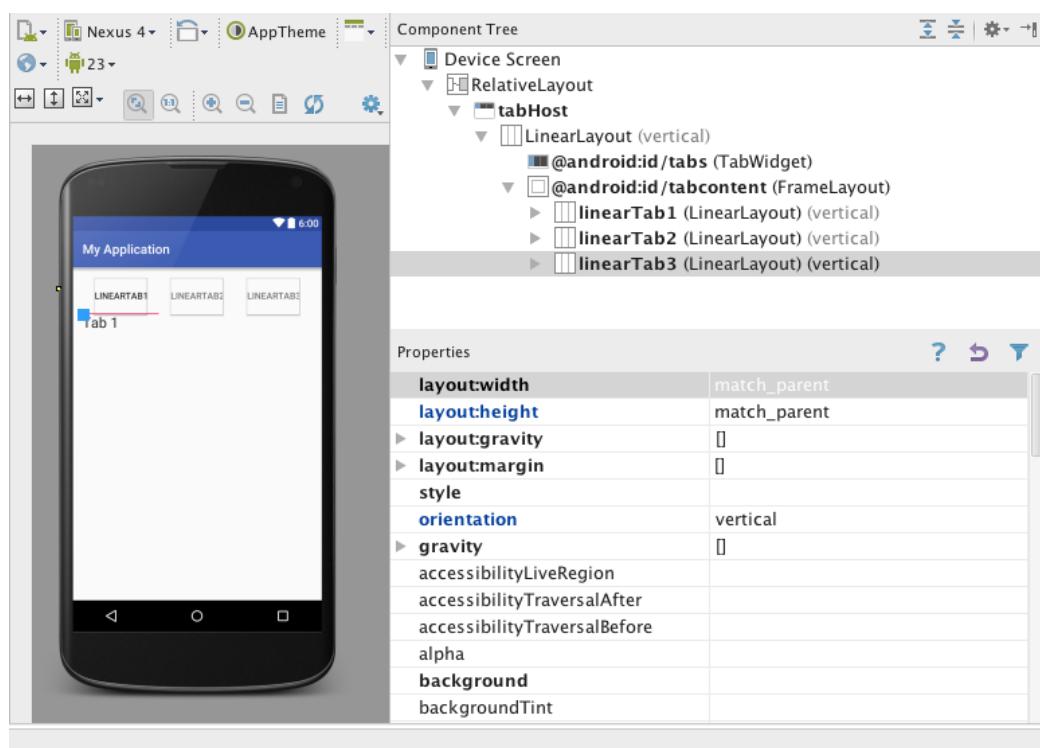


Figura 1.39: Ejemplo TabHost

1.6. Date and Time

1.6.1. Chronometer

<https://developer.android.com/reference/android/widget/Chronometer.html>

Vista que implementa un cronómetro. En la figura 1.40 diseñamos un cronómetro con varios botones que controlan el estado del cronómetro. En la figura 1.41 mostramos la ejecución de la aplicación.

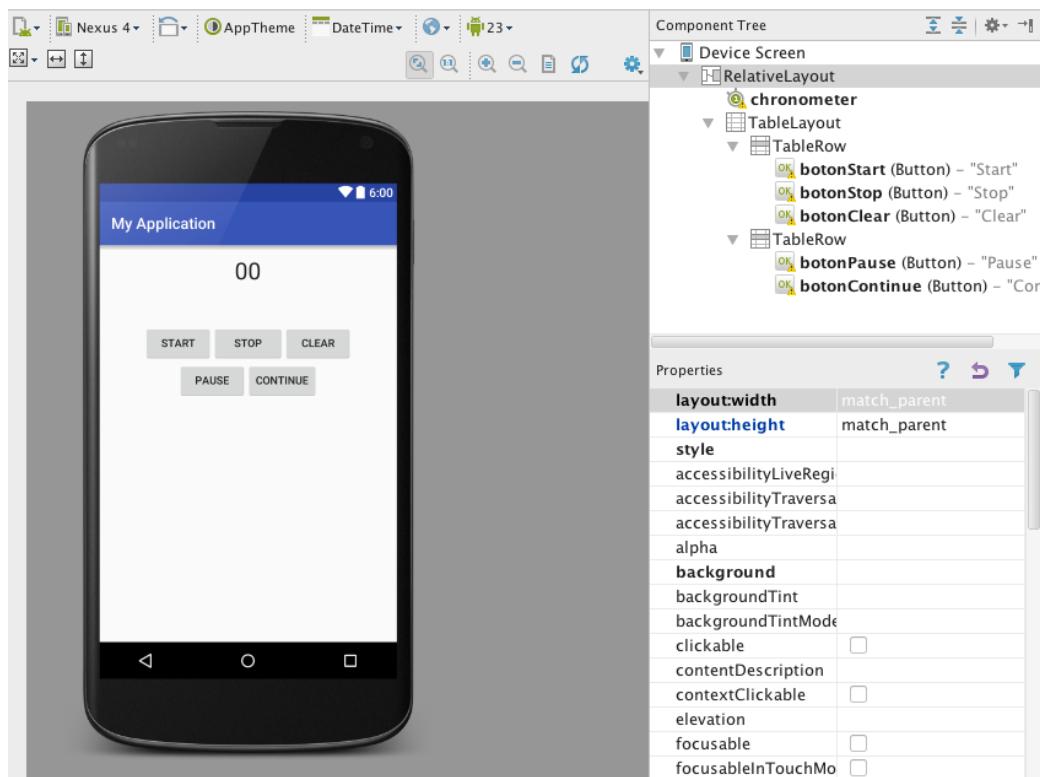


Figura 1.40: Ejemplo Chronometer

El listado ?? muestra el diseño visual de la aplicación donde lo único a resaltar es la utilización de un *TableLayout*.

Listing 1.39: XML Chronometer

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
```



Figura 1.41: Ejemplo Chronometer en el emulador

```
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegromontes.widget.
        DateTime">

    <Chronometer
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/chronometer"
        android:gravity="center_horizontal"
        android:textSize="30dp"
        android:layout_alignParentEnd="true" />

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="51dp"
```

```
    android:layout_alignParentEnd="true"
    android:layout_centerVertical="true"
    android:layout_below="@+id/chronometer">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Start"
            android:id="@+id/botonStart"
            android:onClick="onClick" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Stop"
            android:id="@+id/botonStop"
            android:onClick="onClick" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Clear"
            android:id="@+id/botonClear"
            android:onClick="onClick" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_horizontal" >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Pause"
            android:id="@+id/botonPause"
            android:onClick="onClick" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Continue"
            android:id="@+id/botonContinue"
            android:onClick="onClick" />
```

```
</TableRow>
</TableLayout>

</RelativeLayout>
```

Por otro lado en el listado ?? establecemos los métodos necesarios para controlar los estados del cronómetro.

Listing 1.40: Java Chronometer

```
package com.example.joseamontenegromontes.widget;

import android.os.SystemClock;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Chronometer;
import android.widget.Toast;

import java.util.Calendar;
import java.util.TimeZone;

public class DateTime extends AppCompatActivity {

    long timeWhenStopped = 0;
    Chronometer cronometro;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_date_time);

        cronometro = (Chronometer) findViewById(R.id.
                chronometer);

        cronometro.setOnChronometerTickListener(new
            Chronometer.OnChronometerTickListener() {
            @Override
            public void onChronometerTick(Chronometer
                chronometer) {

                Calendar calendar = Calendar.getInstance(
                    TimeZone.getTimeZone("GMT"));
                calendar.setTimeInMillis(SystemClock.
                    elapsedRealtime() - chronometer.getBase())
                ;
                int hours = calendar.get(Calendar.HOUR_OF_DAY)
                ;
                int minutes = calendar.get(Calendar.MINUTE);
```

```
        int seconds = calendar.get(Calendar.SECOND);

        Log.d("DEBUG", hours+" H "+minutes+" M "+
              seconds+" S ");
    }
}

public void onClick(View v){
    switch(v.getId()){

        case R.id.botonClear:
            Log.d("DEBUG", "CLEAR");
            cronometro.setBase(SystemClock.elapsedRealtime
                ());
            cronometro.stop();
            timeWhenStopped = 0;
            break;
        case R.id.botonStart:
            cronometro.setBase(SystemClock.elapsedRealtime
                ());
            Log.d("DEBUG", "START");
            cronometro.start();
            timeWhenStopped = 0;
            break;
        case R.id.botonStop:
            Log.d("DEBUG", "STOP");
            cronometro.stop();
            timeWhenStopped = 0;
            break;
        case R.id.botonContinue:
            cronometro.setBase(SystemClock.elapsedRealtime
                () + timeWhenStopped);
            cronometro.start();
            Log.d("DEBUG", "Continue");
            break;
        case R.id.botonPause:
            timeWhenStopped = cronometro.getBase() -
                SystemClock.elapsedRealtime();
            cronometro.stop();
            Log.d("DEBUG", "Pause");
            break;
    }
}
}
```

* Ejercicio 9 Modificar el cronómetro

La aplicación del cronómetro contiene, al menos, un error en su funcionamiento, concretamente en el tratamiento del botón *continue*.

Ejercicio 10 Añadir un botón lap.

Añada un botón *lap* que almacene el tiempo de un evento (vuelta) y lo muestre en la pantalla.

1.6.2. Pickers

<https://developer.android.com/guide/topics/ui/controls/pickers.html>

<https://developer.android.com/reference/android/widget/CalendarView.html>

En esta sección veremos *DatePicker*, *TimePicker* y *CalendarView*, aunque esté último no sea un pickers propiamente dicho.

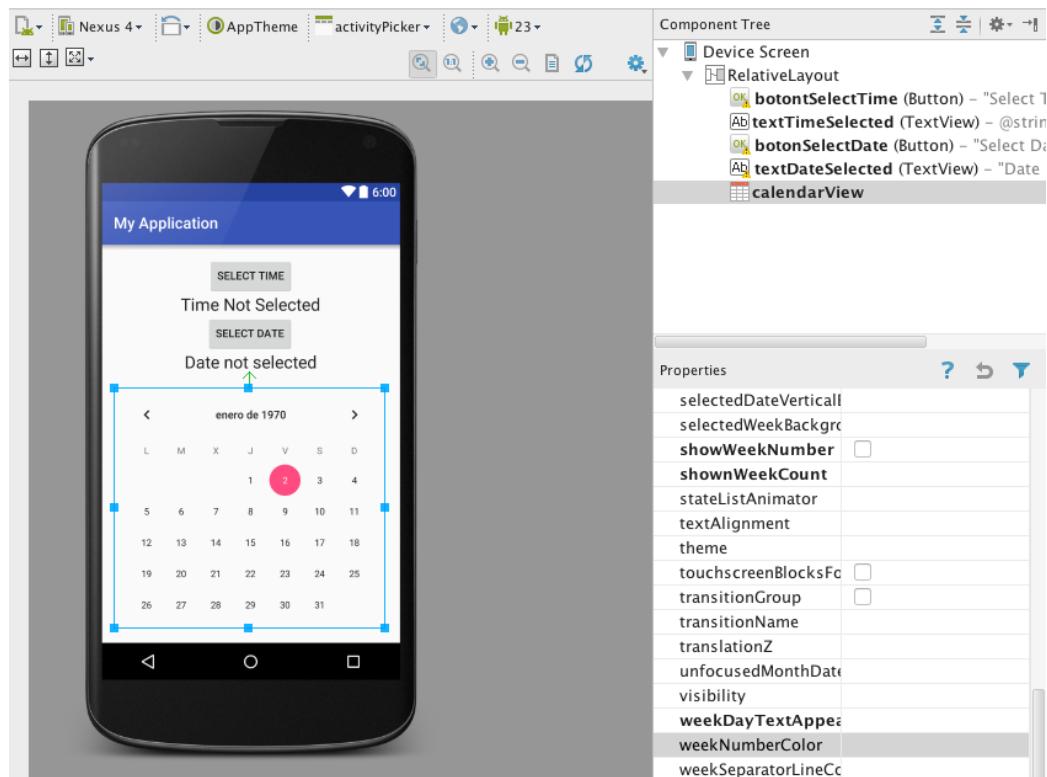


Figura 1.42: Ejemplo Pickers

En la aplicación diseñada mostrada en la figura 1.42 solamente establecemos una vista nueva, *CalendarView*, tal y como mostramos en el código 1.41. Los demás elementos son mostrados como un diálogo que es mostrado cuándo pulsamos el botón correspondientes, tal y como muestra la figura 1.42.

Listing 1.41: XML Picker

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegromontes.widget.
        activityPicker">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Select Time"
        android:id="@+id/botontSelectTime"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:onClick="showPickerDialog" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/
            textAppearanceLarge"
        android:text="@string/time_not_selected"
        android:id="@+id/textTimeSelected"
        android:layout_below="@+id/botontSelectTime"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Select Date"
        android:id="@+id/botonSelectDate"
        android:onClick="showPickerDialog"
        android:layout_below="@+id/textTimeSelected"
        android:layout_alignEnd="@+id/botontSelectTime" />

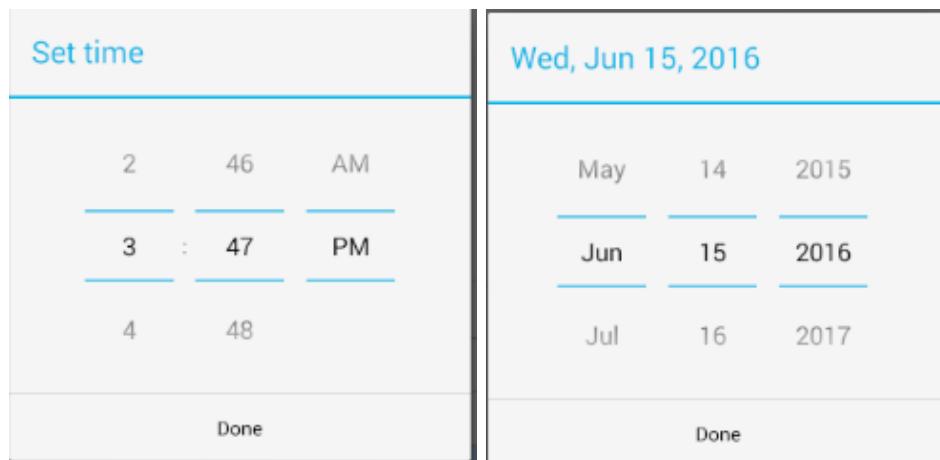
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/
            textAppearanceLarge"
        android:text="Date not selected"
```

```

    android:id="@+id/textDateSelected"
    android:layout_below="@+id/botonSelectDate"
    android:layout_centerHorizontal="true" />

<CalendarView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/calendarView"
    android:layout_alignParentEnd="true"
    android:layout_below="@+id/textDateSelected"
    android:layout_marginTop="20dp"
    android:firstDayOfWeek="0" />
</RelativeLayout>

```



(c) Calendar

Figura 1.43: Pickers y CalendarView

Los eventos de los botones son establecidos en el método `showPickerDialog` donde creamos dos frag-

mentos de diálogo según si queremos seleccionar una fecha o una hora. Para tal fin creamos dos clases privadas *TimePickerFragment* y *DatePickerFragment* que heredan de *DialogFragment*. Ambas clases crean un Diálogo que se encarga de gestionar los respectivos *pickers*. Una vez seleccionado y finalizado el diálogo, la selección es establecida en los *TexView* correspondiente y en el caso de la fecha también es seleccionado en el *CalendarView*. El listado 1.42 muestra el código completo de la aplicación.

Listing 1.42: Java Picker

```
package com.example.joseamontenegromontes.widget;

import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.support.v4.app.DialogFragment;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.format.DateFormat;
import android.view.View;
import android.widget.CalendarView;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;

import java.util.Calendar;

public class activityPicker extends AppCompatActivity {

    TextView timePicker,datePicker;
    CalendarView calendar;

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

        timePicker = (TextView) findViewById(R.id.
                textTimeSelected);
        datePicker = (TextView) findViewById(R.id.
                textDateSelected);
        calendar = (CalendarView) findViewById(R.id.
                calendarView);

        calendar.setFirstDayOfWeek(2); //lunes

        calendar.setOnDateChangeListener(new CalendarView.
                OnDateChangeListener() {
            @Override
            public void onSelectedDayChange(CalendarView
                    calendarView, int i, int i1, int i2) {
                datePicker.setText("Selected: "+i+" y "+i1+" m
                }
        });
    }
}
```

```
        "+i2+" d");
    }
});
```

```
}
```

```
public void showPickerDialog(View v) {
```

```
    DialogFragment newFragment;
```

```
    switch (v.getId()) {
        case R.id.botontSelectTime:
            newFragment = new TimePickerFragment();
            newFragment.show(getSupportFragmentManager(),
                "timePicker");
            break;

        case R.id.botonSelectDate:
            newFragment = new DatePickerFragment();
            newFragment.show(getSupportFragmentManager(),
                "datePicker");
            break;
    }
}
```

```
private class TimePickerFragment extends DialogFragment
    implements TimePickerDialog.OnTimeSetListener {
```

```
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState)
    {
        final Calendar c = Calendar.getInstance();
        int hour = c.get(Calendar.HOUR_OF_DAY);
        int minute = c.get(Calendar.MINUTE);

        return new TimePickerDialog(getActivity(), this,
            hour, minute,
            DateFormat.is24HourFormat(getActivity()));
    }

    public void onTimeSet(TimePicker view, int hourOfDay,
        int minute) {
        timePicker.setText("Selected: "+hourOfDay+" h"+
            minute+" m");
    }
}
```

```
}

private class DatePickerFragment extends DialogFragment
    implements DatePickerDialog.OnDateSetListener {

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        final Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);

        return new DatePickerDialog(getActivity(), this,
            year, month, day);
    }

    public void onDateSet(DatePicker view, int year, int
        month, int day) {

        final Calendar c = Calendar.getInstance();
        c.set(year,month,day);
        long timeSelected= c.getTimeInMillis();
        calendar.setDate(timeSelected);
        datePicker.setText("Selected: "+year+" y "+month+
            " m "+day+" d");
    }
}
```

Capítulo 2

Almacenamiento

Android proporciona varias opciones para almacenar los datos de la aplicación de forma persistente. La solución escogida depende de las necesidades específicas, así como si los datos deben ser privados a la aplicación o accesible a otras aplicaciones y el espacio que necesite los datos.

Preferencias Compartida : Almacena datos primitivos en pares clave-valor.

Almacenamiento Interno : Almacena datos privados en la memoria del dispositivo.

Almacenamiento Externo : Almacena datos públicos en almacenamiento externo.

Base de datos SQLite : Almacena datos estructurados en una base de datos.

2.1. Preferencias Compartidas

```
https://developer.android.com/reference/android/  
content/SharedPreferences.html
```

```
https://developer.android.com/guide/topics/data/  
data-storage.html#pref
```

La clase *SharedPreferences* proporciona un marco de trabajo que permite almacenar y recuperar pares de clave-valor de datos primitivos. Podemos utilizar *SharedPreferences* para almacenar cualquier dato primitivo: booleans, floats, ints, longs, y strings.

Para obtener una instancia del objeto *SharedPreferences* para la aplicación utilizaremos uno de los dos siguientes métodos:

- *getSharedPreferences*. Utilizamos este método si necesitamos múltiples archivos de preferencias identificados por un nombre, el cual debe ser especificado con el primer parámetro.
- *getPreferences*. Utilizar este método si solamente necesitamos una archivo de preferencias para la Actividad. Debido a que será el único archivo de preferencias para tu actividad, no será necesario especificar un nombre.

Para escribir los valores:

1. Llamar *edit()* para obtener un *SharedPreferences.Editor*.
2. Añadir valores con los métodos cómo *putBoolean()* y *putString()*.
3. Almacenar los nuevos valores con *commit()*

Para leer los valores, utilizaremos los métodos tales como *getBoolean()* y *getString()*.

El siguiente ejemplo almacena un entero que representa el número de ocasiones que el usuario ha accedido al recurso. Mediante el método *getSharedPreferences* obtenemos el contenido de las preferencias del archivo indicado. Solamente una instancia es creada, así que solamente se puede editar uno tras otro, en el caso que existan varias invocaciones. Si el archivo de las preferencias no existe, será creado. El segundo parámetro es el modo de operación. Estableceremos 0 (MODE_PRIVATE) para la operación por defecto. Los modos existentes son:

- *MODE_PRIVATE*: Modo por defecto, el archivo creado solamente es accesible por la aplicación que la invoca.
- *MODE_WORLD_READABLE*: Permite otras aplicaciones tener acceso de lectura al archivo de preferencias.
- *MODE_WORLD_WRITEABLE* : Permite todas las aplicaciones tener acceso de escritura al archivo creado.

Finalmente recuperamos el dato con etiqueta *lecturaRecurso*. El segundo parámetro es el valor por defecto que obtendrá la variable en el caso que la etiqueta no exista.

```
public static final String PREFS_NAME = "Almacenamiento";  
  
SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);  
recurso = settings.getInt("lecturaRecurso", 0);
```

El siguiente listado muestra el almacenamiento del entero cuando finaliza la aplicación. De la misma forma que realizamos la lectura estableceremos la escritura mediante el método *getSharedPreferences*. Una vez establecido las preferencias, obtenemos un objeto editor que nos permite almacenar el entero con el método *putInt*. Finalmente, almacenamos las variables mediante el método *commit*.

```
protected void onStop() {  
    super.onStop();  
  
    SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);  
    SharedPreferences.Editor editor = settings.edit();  
    editor.putInt("lecturaRecurso", recurso);  
  
    editor.commit();  
}
```

El listado completo de la aplicación es mostrado en XXXXX.

2.2. Almacenamiento Interno

```
https://developer.android.com/reference/android/content/Context.html#openFileOutput\(java.lang.String,int\)
```

Es posible almacenar archivos en el almacenamiento interno del dispositivo. Por defecto, los archivos almacenados en el almacenamiento interno son privados a la aplicación y otras aplicaciones no pueden acceder a ellos. Cuando la aplicación es desinstalada, los archivos son eliminados.

Para crear y escribir un archivo privado en el almacenamiento interno:

1. Llamar a *openFileOutput()* con el nombre del archivo y el modo de operación. Esto devuelve un objeto *OutputStream*.
2. Escribir el archivo con *write()*.
3. Cerrar el fichero con *close()*.

Para realizar una lectura en el almacenamiento interno:

1. Llamar a *openFileInput()* con el nombre del archivo a leer. Esto devuelve un objeto *FileInputStream*.
 2. Leer bytes del archivo con *read()*.
 3. Cerrar el fichero con *close()*.
- *getFilesDir()*. Obtiene el path absoluto al directorio donde los archivos son almacenados.
 - *getDir()*. Crea o abre un directorio existente dentro del espacio de almacenaje interno.
 - *deleteFile()*. Borra un archivo almacenado en el almacenamiento externo.
 - *fileList()*. Devuelve un array de archivos almacenados por la aplicación.

Además de imágenes e iconos podemos almacenar archivos en el directorio de recursos de la aplicación. A continuación mostraremos un ejemplo como es posible leer la información de un archivo de texto almacenado en los recursos. Obtenemos acceso a los recursos mediante el siguiente método.

```
InputStream is = this.getResources().openRawResource(R.raw.lapepa);
```

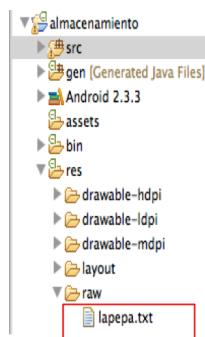


Figura 2.1: Ruta del archivo a leer.

El listado 1 muestra como podemos realizar la lectura del archivo que hemos almacenado dentro de nuestra aplicación en la carpeta raw. La localización del archivo es realizada mediante el método *openRawResource* (línea

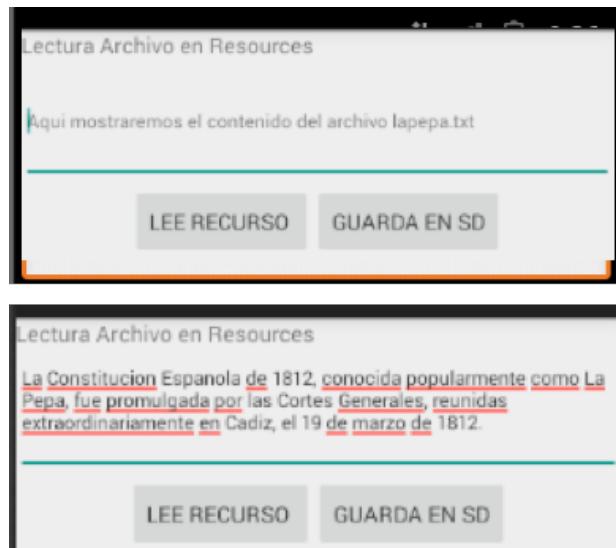


Figura 2.2: Aplicación Almacenamiento.

4 del listado 1) El método es ejecutado cuando pulsamos el botón “Leer Recurso” de la Aplicación Almacenamiento (figura 2.2)

```

1 public StringBuffer MostrarArchivo() throws IOException{
2     String str="";
3     StringBuffer buf = new StringBuffer();
4     InputStream is = this.getResources().openRawResource(R.raw.lapepa);
5
6     BufferedReader reader = new BufferedReader(new InputStreamReader(is));
7
8     if (is!=null)
9         while ((str = reader.readLine()) != null)
10            buf.append(str + "\n");
11
12     is.close();
13     return buf;
14 }
```

Listing 1: Lectura de un Recurso

2.3. Almacenamiento Externo

<https://developer.android.com/reference/android/os/Environment.html>

Todos los dispositivos Android soportan almacenamiento externo que puede ser usados para almacenar ficheros, bien puede ser almacenamiento removible (tarjeta SD) o no. Los archivos almacenados en un almacenamiento externo pueden ser leidos por todos e incluso modificados cuando el dispositivo es conectado a un ordenador. Cómo su nombre indica el almacenamiento externo puede estar no disponible si el usuario elimina el dispositivo de almacenamiento.

El almacenamiento externo necesita establecer permisos en el manifiesto:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

El permiso *WRITE_EXTERNAL_STORAGE* permite la lectura y escritura de los archivos.

Previamente a realizar cualquier trabajo en el almacenamiento externo, debemos verificar que el dispositivo está disponible mediante el método *getExternalStorageState()*.

```
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Podemos obtener la ruta absoluta para saber donde almacenamos mediante la siguiente sentencia:

```
String spPath = Environment.getExternalStorageDirectory().getAbsolutePath();
```

Una vez obtenido la ruta del archivo podemos hacer uso de la clase estándar IO de Java (Java.io.file), listado 3 y 2 .

Mediante DDMS podemos observar el contenido del sistema de archivos, incluido la tarjeta SD (figura 2.3).

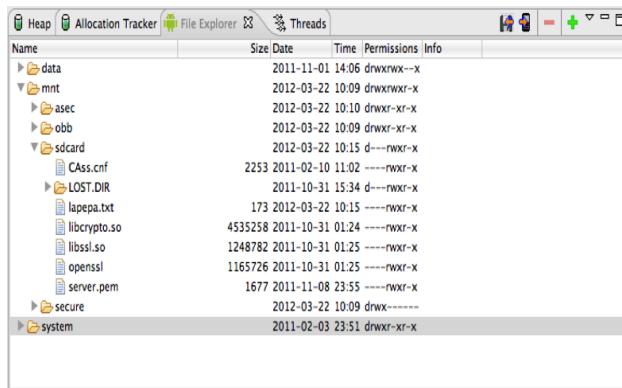


Figura 2.3: Aplicación DDMS para explorar archivos dispositivos

```

1 public void escribirSD(String info) throws IOException {
2     String sdPath = Environment.getExternalStorageDirectory().getAbsolutePath() ;
3     File myFile = new File(sdPath + "/lapepa.txt");
4     myFile.createNewFile();
5
6     FileOutputStream fOut = new FileOutputStream(myFile);
7     OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
8     myOutWriter.append(info);
9     myOutWriter.close();
10    fOut.close();
11    Toast.makeText(getApplicationContext(),"Escritura en SD correcto.",
12            Toast.LENGTH_SHORT).show();
13 }

```

Listing 2: Método Escritura en la SD

```

1 public String leerSD() throws IOException {
2     String sdPath = Environment.getExternalStorageDirectory().getAbsolutePath() ;
3     File myFile = new File(sdPath+ "/lapepa.txt");
4
5     FileInputStream fIn = new FileInputStream(myFile);
6     BufferedReader myReader = new BufferedReader(new InputStreamReader(fIn));
7     String aDataRow = "", aBuffer = "";
8     while ((aDataRow = myReader.readLine()) != null)
9         aBuffer += aDataRow + "\n";
10    myReader.close();
11
12    Toast.makeText(getApplicationContext(), "Lectura SD correcto",
13        Toast.LENGTH_SHORT).show();
14    return aBuffer;
15 }
```

Listing 3: Método Lectura en la SD

La aplicación Almacenamiento hace uso del Almacenamiento Interno y Externo. Hace uso de los métodos de los listados 1 ,2 y 3.



Figura 2.4: Aplicación Almacenamiento.

Listing 2.1: Java AlmacenamientoActivity

```

package es.uma.almacenamiento;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AlmacenamientoActivity extends AppCompatActivity
{
    /** Called when the activity is first created. */
    public static final String PREFS_NAME = "Almacenamiento";
    int recurso;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final EditText EdTxt= (EditText) findViewById(R.id.
            editText1);
        Button botonLecRes= (Button) findViewById(R.id.
            button1);
        Button botonGuardaSD= (Button) findViewById(R.id.
            botonGuardaSD);
        Button botonLeerSD= (Button) findViewById(R.id.
            botonLeerSD);
        final EditText editTextLeerSD= (EditText)
            findViewById(R.id.editTextLeerSD);

        SharedPreferences settings = getSharedPreferences(
            PREFS_NAME, 0);
        recurso = settings.getInt("lecturaRecurso", 0);

        botonLecRes.setOnClickListener(new View.
            OnClickListener() {
            public void onClick(View v) {
                try {
                    StringBuffer txt= MostrarArchivo() ;
                    EdTxt.setText(txt);
                    recurso++;
                    Toast.makeText(getApplicationContext(),"Lecturas:

```



```

        }
        else{
            Toast.makeText(getApplicationContext(),"
                Almacenamiento Externo no preparado.
                ", Toast.LENGTH_SHORT).show();
        }
    });
}

public StringBuffer MostrarArchivo() throws IOException {
    String str="";
    StringBuffer buf = new StringBuffer();
    InputStream is = this.getResources().openRawResource(R.
        raw.lapepa);

    BufferedReader reader = new BufferedReader(new
        InputStreamReader(is));

    if (is!=null)
        while ((str = reader.readLine()) != null) buf.append(
            str + "\n");
    is.close();

    return buf;
}

protected void onStop(){
    super.onStop();

    SharedPreferences settings = getSharedPreferences(
        PREFS_NAME, 0);
    SharedPreferences.Editor editor = settings.edit();
    editor.putInt("lecturaRecurso", recurso);

    editor.commit();
}

public void escribirSD(String info) throws IOException {
    String sdPath = Environment.
        getExternalStorageDirectory().getAbsolutePath() ;
    File myFile = new File(sdPath+"/lapepa.txt");
    myFile.createNewFile();
    FileOutputStream fOut = new FileOutputStream(myFile);
    OutputStreamWriter myOutWriter = new
        OutputStreamWriter(fOut);
    myOutWriter.append(info);
    myOutWriter.close();
}

```

```

fOut.close();

Toast.makeText(getApplicationContext(),"Escritura en SD
correcto.", Toast.LENGTH_SHORT).show();

}

public String leerSD() throws IOException {
    String sdPath = Environment.getExternalStorageDirectory
        ().getAbsolutePath() ;
    File myFile = new File(sdPath+"/lapepa.txt");

    FileInputStream fIn = new FileInputStream(myFile);
    BufferedReader myReader = new BufferedReader(new
        InputStreamReader(fIn));
    String aDataRow = ""; String aBuffer = "";
    while ((aDataRow = myReader.readLine()) != null)
        aBuffer += aDataRow + "\n";

    myReader.close();

    Toast.makeText(getApplicationContext(),"Lectura SD correcto",
        Toast.LENGTH_SHORT).show();
    return aBuffer;
}

public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(
            state)) {
        return true;
    }
    return false;
}
}

```

Listing 2.2: XML AlmacenamientoActivity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    tools:context="es.uma.almacenamiento.MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/text" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:hint="@string/hintText"
        android:inputType="textMultiLine"
        android:textSize="12dp" >

        <requestFocus />
    </EditText>

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center" >

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="@string/buttonText" />

        <Button
            android:id="@+id/botonGuardaSD"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/SaveSDB" />

    </LinearLayout>

    <TextView
        android:id="@+id/textView2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/TxtSD" />

    <EditText
        android:id="@+id/editTextLeerSD"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:hint="@string/HintLecSD"
        android:inputType="textMultiLine"
        android:textSize="12dp" />

    <Button
        android:id="@+id/botonLeerSD"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/BotonLeerSD" />

</LinearLayout>
```

2.4. Base de Datos

<https://developer.android.com/guide/topics/data/data-storage.html#db>

<https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html>

Android proporciona soporte completo para base de datos *SQLite*. Proporciona características de una base de datos relacional como sintaxis SQL, transacciones . . . , con un costo de memoria muy pequeño en ejecución. Cualquier base de datos que creas podrá ser accesible por la aplicación pero no fuera de la aplicación.

Los tipos de datos que podemos utilizar son TEXT (similar a String en Java), INTEGER (similar a long en Java) y REAL (similar a double en Java). Más información en <http://www.sqlite.org>.

2.4.1. Creación de la Base de datos

Android administra automáticamente la base de datos, solamente es necesario definir las sentencias para la creación y actualización. El acceso a la

base de datos implica un acceso al sistema de fichero, por lo que puede ser lento. El método recomendado para crear una base de datos SQLite es crear una subclase de *SQLiteOpenHelper* y sobreescribir el método *onCreate()* y *onUpgrade()*.

- *onCreate()* es invocado por el sistema si la base de datos no existe.
- *onUpgrade()* es invocado si la versión de la base de datos ha sido incrementada.

El código 4 muestra la creación de la base de datos heredando de la clase *SQLiteOpenHelper*, como mencionamos anteriormente. Los métodos *onCreate* y *onUpgrade* hacen uso del método *ExecSQL* de la clase *SQLiteDatabase* que permite ejecutar comandos SQL directamente.

```

1 private static class BaseDatosHelper extends SQLiteOpenHelper{
2     BaseDatosHelper(Context context) {
3         super(context, DATABASE_NAME, null, DATABASE_VERSION);
4     }
5
6     public void onCreate(SQLiteDatabase db) {
7         db.execSQL(DATABASE_TABLE_CREATE);
8     }
9
10    public void onUpgrade(SQLiteDatabase db, int oldVersion,int newVersion) {
11        Log.w(TAG, "Actualizando base de datos de la version " + oldVersion+
12                " a "+ newVersion + ", borraremos todos los datos");
13        db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
14        onCreate(db);
15    }
16 }
```

Listing 4: Creación clase que hereda de SQLiteOpenHelper

En el caso de *onCreate*, la sentencia SQL que ejecutaremos para nuestro ejemplo sería la mostrada en el listado 5. Nuestra base de datos solamente contiene una tabla.

```

1 CREATE TABLE Infoalumno (
2     _id      INTEGER PRIMARY KEY AUTOINCREMENT,
3     alumno   TEXT    NOT NULL,
4     curso    TEXT    NOT NULL,
5     telefono TEXT    NOT NULL
6 );
```

Listing 5: Ejemplo creación tabla de datos Sqllite

En el caso de *onUpdate*, la sentencia SQL que ejecutaremos para nuestro ejemplo sería la mostrada en el listado 9.

```
1 DROP TABLE IF EXISTS
```

Listing 6: Ejemplo eliminación tabla de datos Sqllite

Tanto si queremos probar las sentencias o verificar la base de datos podemos hacer uso de la herramienta *SQLite Studio*, disponible en la web <http://sqlitestudio.pl/>. Por ejemplo, en la figura 2.5 mostramos el editor de sentencias SQL y la figura 2.6 muestra el resultado de la creación de la tabla del listado 5.

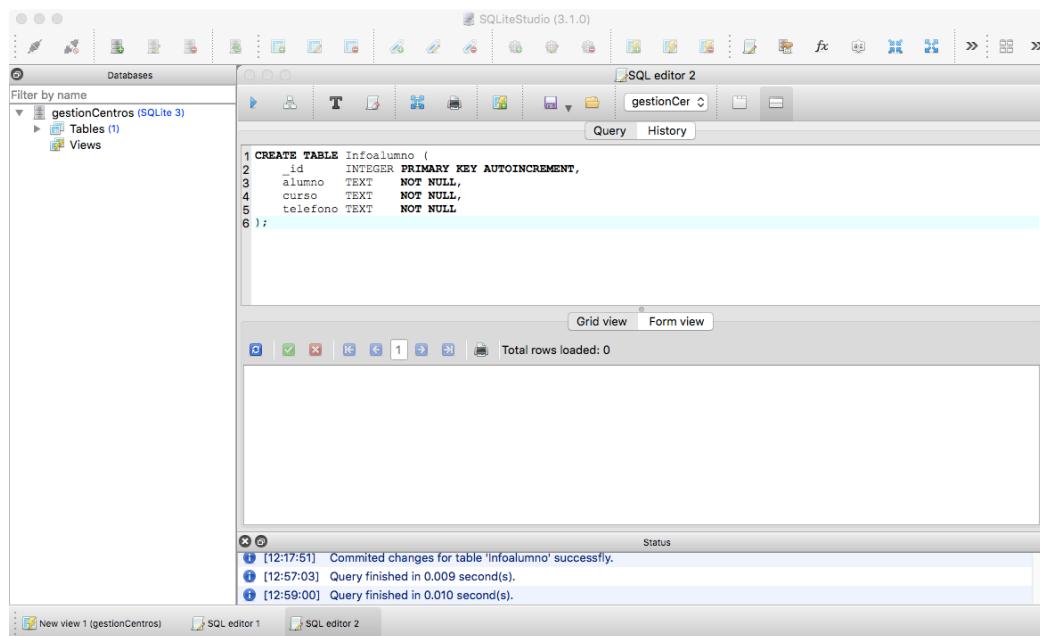


Figura 2.5: Create Table Editor SQLiteStudio

2.4.2. Inserción en la Base de datos

Para la escritura y lectura de la base de datos la clase que herede de *SQLiteOpenHelper* proporciona los métodos *getReadableDatabase()* y *getWritableDatabase()* para acceder en modo lectura o escritura, respectivamente.

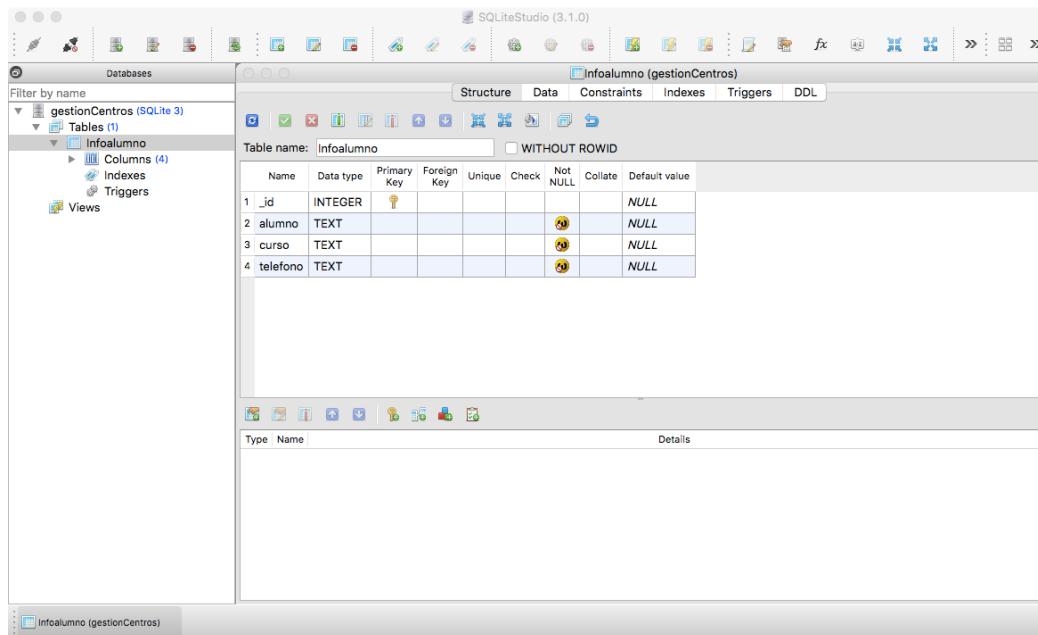


Figura 2.6: Resultado Create Table SQLiteStudio

Para insertar elementos en una tabla podemos hacer uso del método *insert* de la clase *SQLiteDatabase*. Los parámetros son los establecidos en la siguiente tabla:

Parámetro	Comentario
String table	Tabla para insertar los datos.
String nullColumnHack	Opcional, puede ser null. En caso de no ser null, inserta un NUL
ContentValues values	Pares de datos a insertar (claves, valores)

El método devolverá el identificador de la fila insertada o -1 si la operación es errónea. El listado 7 muestra un método para insertar alumnos en la base de datos utilizando el método *insert*.

```

1 public long insertarAlumno(Alumnos alumno){
2     ContentValues initialValues = new ContentValues();
3     initialValues.put(KEY_ALUMNO, alumno.getNombre());
4     initialValues.put(KEY_CURSO, alumno.getCurso());
5     initialValues.put(KEY_TELEFONO, alumno.getCurso());
6
7     return bsSql.insert(DATABASE_TABLE, null, initialValues);
8 }
```

Listing 7: Ejemplo inserción datos Sqllite

```
1 INSERT INTO Infoalumno (alumno,curso,telefono) VALUES ("Pepe Lopez","2","95167689")
```

Listing 8: Ejemplo inserción tabla de datos Sqllite

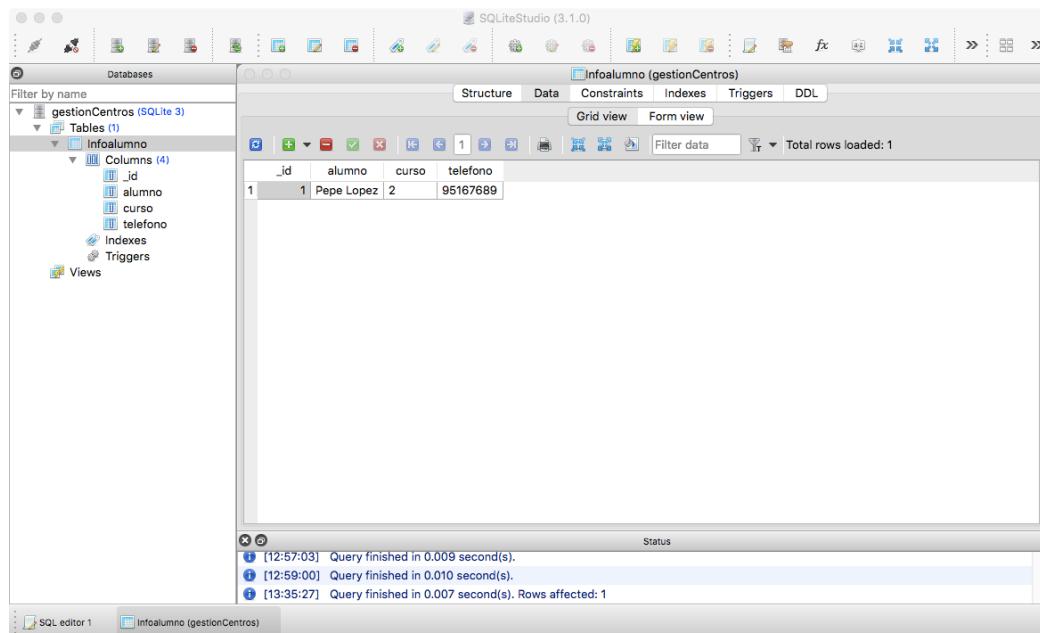


Figura 2.7: Insercción datos SQLiteStudio

2.4.3. Borrado en la Base de datos

De la misma forma que podemos insertar datos en la base de datos, es posible eliminar elementos de la tabla. Para tal fin podemos hacer uso del método *delete*.

Parámetro	Comentario
String table	Tabla para eliminar los datos.
String whereClause	Opcional, si es null borrará todas las filas. En caso de no ser null, borrará los elementos que se aplica en la cláusula.
String whereArgs	Valores utilizados por el segundo parámetro.

El método devolverá el identificador de la fila borrada o 0 si la operación es errónea. El listado 9 muestra un método para eliminar alumnos en la base de datos utilizando el método *delete*.

```

1 public boolean borrarAlumno(long expediente) {
2     return bsSql.delete(DATABASE_TABLE, KEY_ROWID + "=" + expediente, null) > 0;
3 }
```

Listing 9: Ejemplo de método delete

```

1 DELETE FROM Infoalumno WHERE _id='1';
```

Listing 10: Ejemplo delete tabla de datos SQLite

2.4.4. Actualizar en la base de datos

Si queremos actualizar algún dato de los elementos existentes de la base de datos, podemos hacer uso del método *update*.

Parámetro	Comentario
String table	Tabla para eliminar los datos.
ContentValues values	Pares de datos a modificar (claves, valores)
String whereClause	Opcional, si es null borrará todas las filas. En caso de no ser null, borrará los elementos que se aplica en la cláusula.
String whereArgs	Valores utilizados por el segundo parámetro.

```

1 public boolean actualizarAlumno(int expediente, String alumno, String curso, String telefono) {
2     ContentValues args = new ContentValues();
3     args.put(KEY_ALUMNO, alumno);
4     args.put(KEY_CURSO, curso);
5     args.put(KEY_TELEFONO, telefono);
6     return bsSql.update(DATABASE_TABLE, args, KEY_ROWID + "=" + expediente, null) > 0;
7 }
```

Listing 11: Ejemplo del método update

```

1 UPDATE Infoalumno SET alumno = "Antonio Lopez" WHERE _id = '1';
```

Listing 12: Ejemplo actualizar tabla de datos SQLite

2.4.5. Consultas base de datos

Las consultas pueden ser creadas mediante dos métodos:

- `RawQuery()`: Acepta una sentencia SQL como entrada.
- `Query()`: Proporciona un interfaz estructurado para una consulta SQL.

La sentencia sql mostrada por el listado 13, es realizada en Android en el código 14 y 15.

```
1 SELECT * FROM Infoalumno Where _id=1
```

Listing 13: Ejemplo query tabla de datos SQLite

```
1 private String[] todasColumnas =
2     new String[] {KEY_ROWID, KEY_ALUMNO, KEY_CURSO, KEY_TELEFONO};
3
4 Cursor mCursor = bsSql.query(true, DATABASE_TABLE, todasColumnas,
5     KEY_ROWID + " = " + expediente, null, null, null, null);
```

Listing 14: Ejemplo de consulta para obtener todos los usuarios BD

```
1 Cursor cursor = db.rawQuery("SELECT * FROM "+DATABASE_TABLE +
2     " WHERE "+KEY_ROWID+ " = " + expediente, null);
3
4 String[] param = new String[1];
5 param[0] = Integer.toString(expediente, 10);
6
7 Cursor cursor = db.rawQuery("SELECT * FROM "+DATABASE_TABLE +
8     " WHERE "+KEY_ROWID+ " = ? ", param);
```

Listing 15: Ejemplo de consulta para obtener todos los usuarios BD

El significado de los parámetros del método `query` son los siguientes:

Parámetro	Comentario
String NombreBD	Base de dato a realizar la consulta
Int [] nombreColumnas	Lista de columnas que devuelve la consulta
String clausulaWhere	Clausula Where, para filtrar selección datos
String [] argumentosSeleccion	Argumentos pasamos a clausulaWhere
String[] groupBy	Filtro declara como agrupar filas.
String[] having	Filtro para los grupos
String[] orderBy	Columnas usadas para ordenar los datos.

El significado de los parámetros del método *rawQuery* son los siguientes:

Parámetro	Comentario
String sql	Consulta sql.
String [] selectionArgs	Argumentos pasamos a la consulta que sustituye ?s

En el listado ?? observamos que la consulta devuelve un objeto tipo *Cursor*. Este objeto representa el resultado de una consulta de forma eficiente. Alguno de los métodos que proporciona son:

- GetCount: Número de elementos obtenidos de la consulta (filas).
- MoveToFirst, moveToNext: Permite moverse entre las filas obtenidas en la consulta.
- IsAfterLast: Verifica si estamos en el último resultado de la consulta.

Una vez establecidos en la fila deseada de la consulta, tenemos métodos para obtener los datos de las columnas para el elemento de la fila que estamos procesando:

- getLong(int indiceColumna)
- getString(indiceColumna)

El listado 16 muestra una consulta que obtiene todos los alumnos.

```

1 public List<Alumnos> getAllAlumnos() {
2     List<Alumnos> listaAlumnos = new ArrayList<Alumnos>();
3     Cursor cursor = this.getTodosAlumnos(); cursor.moveToFirst();
4
5     while (!cursor.isAfterLast()) {
6         Alumnos comment = cursorToAlumnos(cursor);
7         listaAlumnos.add(comment);
8         cursor.moveToNext();
9     }
10
11     cursor.close(); return listaAlumnos;
12 }
```

Listing 16: Ejemplo del Objeto Cursor

El listado 2.4 muestra el listado completo del adaptador de la base de datos.

Listing 2.3: Java AdaptadorBD

```

package es.uma.SQL;

import java.util.List;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import java.util.ArrayList;

public class AdaptadorBD {

    public static final String KEY_ROWID = "_id";
    public static final String KEY_ALUMNO = "alumno";
    public static final String KEY_CURSO = "curso";
    public static final String KEY_TELEFONO = "telefono";

    private static final String TAG = "AdaptadorBD";

    private static final String DATABASE_NAME = "gestionCentros
    ";
    private static final String DATABASE_TABLE = "Infoalumno";
    private static final int DATABASE_VERSION = 1;

    private static final String DATABASE_TABLE_CREATE =
    "create table "+DATABASE_TABLE+
    "("+KEY_ROWID+" integer primary key autoincrement, "
```

```
+KEY_ALUMNO+" text not null, "
+KEY_CURSO+" text not null, "
+KEY_TELEFONO+" text not null);";

private final Context context;

private BaseDatosHelper BDHelper;
private SQLiteDatabase bsSql;
private String[] todasColumnas =new String[] {KEY_ROWID,
    KEY_ALUMNO,KEY_CURSO,KEY_TELEFONO};

public AdaptadorBD(Context ctx) {
    this.context = ctx;
    BDHelper = new BaseDatosHelper(context);
}

public AdaptadorBD openEscritura() throws SQLException{
    bsSql = BDHelper.getWritableDatabase();
    return this;
}

public AdaptadorBD openLectura() throws SQLException{
    bsSql = BDHelper.getReadableDatabase();
    return this;
}

public void close(){
    BDHelper.close();
}

public long insertarAlumno(String alumno, String curso,
    String telefono){
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_ALUMNO, alumno);
    initialValues.put(KEY_CURSO, curso);
    initialValues.put(KEY_TELEFONO, telefono);
    return bsSql.insert(DATABASE_TABLE, null, initialValues);
}

public long insertarAlumno(Alumnos alumno){
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_ALUMNO, alumno.getNombre());
    initialValues.put(KEY_CURSO, alumno.getCurso());
    initialValues.put(KEY_TELEFONO, alumno.getCurso());
    return bsSql.insert(DATABASE_TABLE, null, initialValues);
}
```

```
public boolean borrarAlumno(long expediente){  
    return bsSql.delete(DATABASE_TABLE, KEY_ROWID + "=" +  
        expediente, null) > 0;  
}  
  
public Cursor getTodosAlumnos() {  
  
    return bsSql.query(DATABASE_TABLE, todasColumnas,null,null  
        ,null,null,null);  
}  
  
public Cursor getAlumno(long expediente) throws SQLException  
{  
  
    Cursor mCursor = bsSql.query(true, DATABASE_TABLE,  
        todasColumnas,  
        KEY_ROWID + "=" + expediente,null,null,null,null,  
        null);  
  
    if (mCursor != null)  mCursor.moveToFirst();  
  
    return mCursor;  
}  
  
  
public boolean actualizarAlumno(int expediente, String  
    alumno, String curso, String telefono){  
    ContentValues args = new ContentValues();  
    args.put(KEY_ALUMNO, alumno);  
    args.put(KEY_CURSO, curso);  
    args.put(KEY_TELEFONO, telefono);  
    return bsSql.update(DATABASE_TABLE, args,KEY_ROWID + "=" +  
        expediente, null) > 0;  
}  
  
public boolean actualizarAlumno(Alumnos alumno){  
    ContentValues args = new ContentValues();  
    args.put(KEY_ALUMNO, alumno.getNombre());  
    args.put(KEY_CURSO, alumno.getCurso());  
    args.put(KEY_TELEFONO, alumno.getTelefono());  
    return bsSql.update(DATABASE_TABLE, args,KEY_ROWID + "=" +  
        alumno.getExpediente(), null) > 0;
```

```
}

public String MostrarAlumno(long expediente) {
    String cadena=null;

    Cursor c = getAlumno(expediente);

    if (c.moveToFirst()) {

        cadena=
            "EXPEDIENTE: " + c.getString(0) + "\n" +
            "ALUMNO: " + c.getString(1) + "\n" +
            "CURSO: " + c.getString(2) + "\n" +
            "TELEFONO: " + c.getString(3);
    }

    return cadena;
}

public String MostrarAlumno(Cursor c) {
    String cadena=null;

    cadena=
        "EXPEDIENTE: " + c.getString(0) + "\n" +
        "ALUMNO: " + c.getString(1) + "\n" +
        "CURSO: " + c.getString(2) + "\n" +
        "TELEFONO: " + c.getString(3);

    return cadena;
}

public List<Alumnos> getAllAlumnos() {

    List<Alumnos> listaAlumnos = new ArrayList<Alumnos>();
    Cursor cursor = this.getTodosAlumnos();

    cursor.moveToFirst();

    while (!cursor.isAfterLast()) {
        Alumnos comment = cursorToAlumnos(cursor);
        listaAlumnos.add(comment);
        cursor.moveToNext();
    }
    cursor.close();
    return listaAlumnos;
}

private Alumnos cursorToAlumnos(Cursor cursor) {
```

```

Alumnos alumno = new Alumnos();
alumno.setExpediente(cursor.getLong(0));
alumno.setNombre(cursor.getString(1));
alumno.setCurso(cursor.getString(2));
alumno.setTelefono(cursor.getString(3));

return alumno;
}

//***** CLASE PRIVADA ***

private static class BaseDatosHelper extends
    SQLiteOpenHelper{
    BaseDatosHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DATABASE_TABLE_CREATE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
        int newVersion){
        Log.w(TAG, "Actualizando base de datos de la version "
            + oldVersion
            + " a "
            + newVersion + ", borraremos todos los datos");
        db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
        onCreate(db);
    }
}
}

```

2.4.6. Aplicación Ejemplo

La aplicación SQL gestiona la información de los alumnos (expediente, nombre, curso y teléfono) mediante una base de datos SQLite.

La pantalla principal (figura 2.8:a) muestra las opciones para insertar (figura 2.8:b) y consulta (figura 2.8:c) de alumnos. La consulta de alumnos tiene añadida la funcionalidad de realizar una llamada por teléfono con la información obtenida de la base de datos (figura 2.8:d).

El spinner en la consulta es actualizado automáticamente mediante *SimpleCursorAdapter*:

```
1 SimpleCursorAdapter adapter2 =
2     new SimpleCursorAdapter (this, android.R.layout.simple_spinner_item, cursor,
3     new String[] {"_id"}, new int[] {android.R.id.text1});
```

El código completo para gestionar la base de datos lo tenemos en el listado 2.4.

Listing 2.4: Java AdaptadorBD

```
package es.uma.SQL;

import java.util.List;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import java.util.ArrayList;

public class AdaptadorBD {

    public static final String KEY_ROWID = "_id";
    public static final String KEY_ALUMNO = "alumno";
    public static final String KEY_CURSO = "curso";
    public static final String KEY_TELEFONO = "telefono";

    private static final String TAG = "AdaptadorBD";

    private static final String DATABASE_NAME = "gestionCentros
        ";
    private static final String DATABASE_TABLE = "Infoalumno";
    private static final int DATABASE_VERSION = 1;

    private static final String DATABASE_TABLE_CREATE =
        "create table "+DATABASE_TABLE+
        "("+KEY_ROWID+" integer primary key autoincrement, "
        "+KEY_ALUMNO+" text not null, "
        "+KEY_CURSO+" text not null, "
        "+KEY_TELEFONO+" text not null);";

    private final Context context;

    private BaseDatosHelper BDHelper;
    private SQLiteDatabase bsSql;
    private String[] todasColumnas =new String[] {KEY_ROWID,
        KEY_ALUMNO,KEY_CURSO,KEY_TELEFONO};
```

```
public AdaptadorBD(Context ctx) {
    this.context = ctx;
    BDHelper = new BaseDatosHelper(context);
}

public AdaptadorBD openEscritura() throws SQLException{
    bsSql = BDHelper.getWritableDatabase();
    return this;
}

public AdaptadorBD openLectura() throws SQLException{
    bsSql = BDHelper.getReadableDatabase();
    return this;
}

public void close(){
    BDHelper.close();
}

public long insertarAlumno(String alumno, String curso,
                           String telefono){
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_ALUMNO, alumno);
    initialValues.put(KEY_CURSO, curso);
    initialValues.put(KEY_TELEFONO, telefono);
    return bsSql.insert(DATABASE_TABLE, null, initialValues);
}

public long insertarAlumno(Alumnos alumno){
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_ALUMNO, alumno.getNombre());
    initialValues.put(KEY_CURSO, alumno.getCurso());
    initialValues.put(KEY_TELEFONO, alumno.getCurso());
    return bsSql.insert(DATABASE_TABLE, null, initialValues);
}

public boolean borrarAlumno(long expediente){
    return bsSql.delete(DATABASE_TABLE, KEY_ROWID + "=" +
                        expediente, null) > 0;
}

public Cursor getTodosAlumnos() {
```

```
        return bsSql.query(DATABASE_TABLE, todasColumnas,null,null
                ,null,null,null);
}

public Cursor getAlumno(long expediente) throws SQLException
{
    Cursor mCursor = bsSql.query(true, DATABASE_TABLE,
            todasColumnas,
            KEY_ROWID + "=" + expediente,null,null,null,null,
            null);

    if (mCursor != null)  mCursor.moveToFirst();

    return mCursor;
}

public boolean actualizarAlumno(int expediente, String
        alumno, String curso, String telefono){
    ContentValues args = new ContentValues();
    args.put(KEY_ALUMNO, alumno);
    args.put(KEY_CURSO, curso);
    args.put(KEY_TELEFONO, telefono);
    return bsSql.update(DATABASE_TABLE, args,KEY_ROWID + "=" +
            expediente, null) > 0;
}

public boolean actualizarAlumno(Alumnos alumno){
    ContentValues args = new ContentValues();
    args.put(KEY_ALUMNO, alumno.getNombre());
    args.put(KEY_CURSO, alumno.getCurso());
    args.put(KEY_TELEFONO, alumno.getTelefono());
    return bsSql.update(DATABASE_TABLE, args,KEY_ROWID + "=" +
            alumno.getExpediente(), null) > 0;
}

public String MostrarAlumno(long expediente){
    String cadena=null;

    Cursor c = getAlumno(expediente);

    if (c.moveToFirst()){

        cadena=
        "EXPEDIENTE: " + c.getString(0) + "\n" +
```

```

"ALUMNO: " + c.getString(1) + "\n" +
"CURSO: " + c.getString(2) + "\n" +
"TELEFONO: " + c.getString(3);
}

return cadena;
}

public String MostrarAlumno(Cursor c) {
String cadena=null;

cadena=
"EXPEDIENTE: " + c.getString(0) + "\n" +
"ALUMNO: " + c.getString(1) + "\n" +
"CURSO: " + c.getString(2) + "\n" +
"TELEFONO: " + c.getString(3);

return cadena;
}

public List<Alumnos> getAllAlumnos() {

List<Alumnos> listaAlumnos = new ArrayList<Alumnos>();
Cursor cursor = this.getTodosAlumnos();

cursor.moveToFirst();

while (!cursor.isAfterLast()) {
    Alumnos comment = cursorToAlumnos(cursor);
    listaAlumnos.add(comment);
    cursor.moveToNext();
}
cursor.close();
return listaAlumnos;
}

private Alumnos cursorToAlumnos(Cursor cursor) {
Alumnos alumno = new Alumnos();
alumno.setExpediente(cursor.getLong(0));
alumno.setNombre(cursor.getString(1));
alumno.setCurso(cursor.getString(2));
alumno.setTelefono(cursor.getString(3));

return alumno;
}

//***** CLASE PRIVADA ***

```

```
private static class BaseDatosHelper extends  
    SQLiteOpenHelper{  
    BaseDatosHelper(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL(DATABASE_TABLE_CREATE);  
    }  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion,  
        int newVersion){  
        Log.w(TAG, "Actualizando base de datos de la version "  
            + oldVersion  
            + " a "  
            + newVersion + ", borraremos todos los datos");  
        db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);  
        onCreate(db);  
    }  
}  
}
```

La actividad principal está detallada en el listado 2.5 y 2.6.

Listing 2.5: Java SQLActivity

```
package es.uma.SQL;  
  
import android.support.v7.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.ImageButton;  
  
public class SQLActivity extends AppCompatActivity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        final Button botonAdd= (Button) findViewById(R.id.  
            button2);  
        final Button botonConsulta= (Button) findViewById(R.id.
```

```
        .button1);

    final Intent IntentConsulta = new Intent(SQLActivity.
        this, Consulta.class);
    final Intent IntentAdd = new Intent(SQLActivity.this,
        AddAlumnos.class);

    botonAdd.setOnClickListener(new ImageButton.
        OnClickListener() {
        public void onClick(View v) {
            startActivity(IntentConsulta);
        }
    });

    botonConsulta.setOnClickListener(new ImageButton.
        OnClickListener() {
        public void onClick(View v) {
            startActivity(IntentAdd);
        }
    });

}

}
```

Listing 2.6: XML SQLActivity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="120dp"

        android:gravity="center"
        android:text="@string/Bienvenida"
        android:textSize="40dp"
        android:layout_gravity="center" />
```

```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:text="@string/BotonAddAlumno" />  
  
<Button  
    android:id="@+id/button2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:text="@string/BotonConsulta" />  
  
<Button  
    android:id="@+id/button3"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:text="@string/BotonBorrarBD" />  
  
</LinearLayout>
```

La actividad para añadir alumnos es detallada en el listado 2.7 y 2.8.

Listing 2.7: Java AddAlumnos

```
package es.uma.SQL;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.Spinner;  
import android.widget.Toast;  
  
public class AddAlumnos extends Activity {  
    EditText edAlumno;  
    EditText edTlf;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.addalumnos);  
  
        final Spinner s = (Spinner) findViewById(R.id.  
            spinnerAdd);
```

```

edAlumno = (EditText) findViewById(R.id.
    editTextAlumno);
edTlf = (EditText) findViewById(R.id.
    editTextTelefono);

ArrayAdapter adapter = ArrayAdapter.
    createFromResource(
        this, R.array.cursos, android.R.layout.
        simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.
    simple_spinner_dropdown_item);
s.setAdapter(adapter);

final Button botonAdd= (Button) findViewById(R.id.
    AddAlumnoBoton);

botonAdd.setOnClickListener(new ImageButton.
    OnClickListener() {
    public void onClick(View v) {

        String alumno = edAlumno.getText().toString();
        String tlf = edTlf.getText().toString();
        String curso = (String) s.getSelectedItem();

        addBDAlumno (alumno,tlf,curso);

    }
});}

public void addBDAlumno (String alumno, String tlf, String
    curso) {

    if ((alumno.length()!=0) & (tlf.length()!=0)) {
        AdaptadorBD db = new AdaptadorBD (this);
        db.openEscritura();
        db.insertarAlumno(alumno,curso,tlf);
        db.close();
        Toast.makeText (this,"Almacenamiento base de datos
            correcta",Toast.LENGTH_SHORT).show();
        edAlumno.setText("");
        edTlf.setText("");
    }
}

```

```
        else Toast.makeText (this,"Todos los campos son  
        obligatorios",Toast.LENGTH_SHORT).show();  
    }  
  
}
```

Listing 2.8: XML AddAlumnos

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/  
    res/android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:orientation="vertical" >  
  
    <TextView  
        android:id="@+id/textView1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/alumnoTxt" />  
  
    <EditText  
        android:id="@+id/editTextAlumno"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:inputType="textPersonName" />  
  
    <TextView  
        android:id="@+id/textView2"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/CursoTxt" />  
  
    <Spinner  
        android:id="@+id/spinnerAdd"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" />  
  
    <TextView  
        android:id="@+id/textView3"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/telefonoTxt" />  
  
    <EditText  
        android:id="@+id/editTextTelefono"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"
```

```
        android:inputType="phone" />

    <Button
        android:id="@+id/AddAlumnoBoton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/BotonAddAlu" />

</LinearLayout>
```

La actividad para realizar consultas sobre los alumnos es detallada en el listado 2.9 y 2.10.

Listing 2.9: Java Consulta

```
package es.uma.SQL;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.SimpleCursorAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class Consulta extends Activity {
    EditText alumno,tlf;
    Spinner s;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.consulta);

        s = (Spinner) findViewById(R.id.spinnerConsulta);
        alumno = (EditText) findViewById(R.id.
            editTextAlumnoConsulta);
        tlf = (EditText) findViewById(R.id.
            editTextTlfConsulta);
        final Button botonConsulta= (Button) findViewById(R.id
            .consultaBoton);
        final Button botonTlf= (Button) findViewById(R.id.
            BotonLlamaTlf);
```

```
AdaptadorBD db = new AdaptadorBD (this);
db.openLectura();

Cursor cursor=db.getTodosAlumnos();
SimpleCursorAdapter adapter2 =
    new SimpleCursorAdapter(this,android.R.layout.
        simple_spinner_item,cursor,
        new String[] {"_id"}, new int[] {android.R.id
            .text1},0);
adapter2.setDropDownViewResource(android.R.layout.
    simple_spinner_dropdown_item);
s.setAdapter(adapter2);

int numElementos=cursor.getCount();
if (numElementos==0){
    botonConsulta.setEnabled(false);
    s.setEnabled(false);
    alumno.setEnabled(false);
    tlf.setEnabled(false);
    Toast.makeText(this, "Base de Datos Vacía.
        Inserte alumnos en Menu Principal", Toast.
        LENGTH_LONG).show();
}
db.close();

botonConsulta.setOnClickListener(new ImageButton.
    OnClickListener() {
    public void onClick(View v) {

        Imprime();
        botonTlf.setEnabled(true);
    }
});;

botonTlf.setOnClickListener(new ImageButton.
    OnClickListener() {
    public void onClick(View v) {

        String tlfS=tlf.getText().toString();
        Intent intent =new Intent(Intent.ACTION_CALL,
            Uri.parse("tel:" + tlfS));
        startActivity(intent);
    }
});;

}
```

```
public void Imprime () {  
  
    Cursor expe=(Cursor)s.getSelectedItem();  
    alumno.setText(expe.getString(1));  
    tlf.setText(expe.getString(3));  
  
}  
}
```

Listing 2.10: XML Consulta

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/  
    res/android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:orientation="vertical" >  
  
    <TextView  
        android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:text="@string/expedienteTxt" />  
  
    <Spinner  
        android:id="@+id/spinnerConsulta"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" />  
  
    <Button  
        android:id="@+id/consultaBoton"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_gravity="center"  
        android:text="@string/BotonConsulta" />  
  
    <TextView  
        android:id="@+id/textView1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/alumnoTxt" />  
  
    <EditText  
        android:id="@+id/editTextAlumnoConsulta"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"
```

```
        android:editable="false" >  
  
        <requestFocus />  
</EditText>  
  
<TextView  
    android:id="@+id/textView2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/telefonoTxt" />  
  
<EditText  
    android:id="@+id/editTextTlfConsulta"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:editable="false" />  
  
<Button  
    android:id="@+id/BotonLlamaTlf"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:enabled="false"  
    android:text="@string/BotonLlamarTlf" />  
  
</LinearLayout>
```

Finalmente hemos creado una clase Alumno para simplificar la gestión de los datos de los alumnos (listado 2.11).

Listing 2.11: Java Alumnos

```
package es.uma.SQL;  
  
public class Alumnos {  
  
    long expediente;  
    String Nombre;  
    String telefono;  
    String curso;  
  
    Alumnos (long expediente, String Nombre, String telefono,  
             String curso) {  
        expediente=expediente;  
        Nombre=Nombre;  
        telefono=telefono;  
        curso=curso;  
    }  
  
    Alumnos (String Nombre, String telefono, String curso) {
```

```
Nombre=Nombred;
telefono=telefonod;
curso=cursod;
}

Alumnos () {
    expediente=0;
    Nombre=null;
    telefono=null;
    curso=null;
}
public void setNombre(String Nombred) {
    Nombre=Nombred;
}

public void setExpediente (long expediente) {
    expediente=expediente;
}

public void setTelefono (String telefonod) {
    telefono=telefonod;
}

public void setCurso(String cursod) {
    curso=cursod;
}

public String getNombre() {
    return Nombre;
}

public long getExpediente () {
    return expediente;
}

public String getTelefono () {
    return telefono;
}

public String getCurso() {
    return curso;
}
}
```

2.4.7. SQL y Android

Para verificar las acciones que estamos realizando con la base de datos podemos obtener la base de datos y abrirla en el ordenador. La figura 2.9 muestra la localización de la base de datos en nuestra aplicación.

Tal como muestra la figura 2.9, la aplicación es almacenada en DATA/data/APP_NAME/databases/Filename donde:

DATA lo obtenemos mediante Environment.getDataDirectory()

APP_NAME es el nombre de la aplicación y

FILENAME es el nombre de la base de datos.

Además es posible gestionar desde el adb la base de datos, tal y como muestra la figura 2.10.

```
1 ./adb shell
2 # sqlite3 /data/data/es.uma.SQL/databases/gestionCentros
3 sqlite>.tables
4 sqlite> select * from Infoalumno;
5 Sqlite>.exit
```

```
1 ./help
```

* Ejercicio 11 Actualice la aplicación SQL

1. Implementar borrar base de datos desde el principal. Incluir un AlertDialog para confirmar el borrado.
2. Añadir un botón e implementar un método para realizar actualizaciones de la información de los alumnos almacenados en la BD.
3. Modificar las consultas utilizando nombre de alumno en vez de expediente (spinner).

** Ejercicio 12 Mejore la aplicación SQL

1. Añadir una tabla de profesores, con su nombre, teléfono y antigüedad.
2. Añadir una tabla de cursos, con su ID, descripción del curso y profesor que lo imparte.
3. Crearemos una tabla que permita identificar los cursos a los que están matriculados los alumnos.



(a) Pantalla inicial



(b) Añadir Alumnos



(c) Consulta Alumnos



(d) Intent Call

Figura 2.8: Aplicación ejemplo SQLite.

Name	Size	Date	Time	Permissions	Info
data		2011-11-01	14:06	drwxrwx--x	
anr		2012-03-13	22:25	drwxrwxr-x	
app		2012-03-23	07:40	drwxrwx--x	
app-private		2011-10-31	15:32	drwxrwx--x	
backup		2012-03-23	07:39	drwx-----	
dalvik-cache		2012-03-23	07:40	drwxrwx--x	
data		2012-03-22	12:40	drwxrwx--x	
Modificacion.Aplicacion		2012-03-23	07:39	drwxr-x--x	
android.tts		2011-10-31	15:33	drwxr-x--x	
cert.paquete		2012-03-23	07:39	drwxr-x---x	
chat.criptado		2012-03-23	07:39	drwxr-x--x	
ciclos.vida		2012-02-18	00:03	drwxr-x--x	
cifrado.simetrico		2012-02-18	00:03	drwxr-x--x	
com.android.browser		2011-11-08	23:56	drwxr-x--x	
com.android.calculator2		2011-10-31	15:33	drwxr-x--x	
com.android.camera		2011-10-31	15:33	drwxr-x--x	
com.android.certinst...		2011-10-31	15:33	drwxr-x--x	
com.android.contacts		2012-02-18	00:32	drwxr-x--x	
com.android.customm...		2011-10-31	15:33	drwxr-x--x	
com.android.defcont...		2011-10-31	15:33	drwxr-x--x	
com.android.deskclock		2011-10-31	15:34	drwxr-x--x	
com.android.develop...		2011-10-31	15:33	drwxr-x--x	
com.android.email		2011-10-31	15:34	drwxr-x--x	
com.android.fallback		2011-10-31	15:33	drwxr-x--x	
com.android.gallery		2011-10-31	15:33	drwxr-x--x	
com.android.gesture....		2012-03-23	07:39	drwxr-x--x	
com.android.htmlviewer		2011-10-31	15:33	drwxr-x--x	
com.android.inputme...		2011-10-31	15:33	drwxr-x--x	
com.android.inputme...		2011-10-31	15:33	drwxr-x--x	
com.android.launcher		2011-10-31	15:34	drwxr-x--x	
com.android.mms		2011-10-31	15:34	drwxr-x--x	
com.android.music		2011-10-31	15:29	drwxr-x--x	
es.uma.SQL		2012-03-23	07:40	drwxr-x--x	
databases		2012-03-22	22:28	drwxrwx--x	
gestionCentros	4096	2012-03-22	16:22	-rw-rw----	

Figura 2.9: Localización base de datos.

```
macmonte:platform-tools monte$ ./adb shell
# sqlite3 /data/data/es.uma.SQL/databases/gestionCentros
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"

sqlite> .tables
Infoalumno      android_metadata
sqlite> select * from Infoalumno;
1|pepe lopez|Primero|952132898
2|jose luis de la borbolla|Primero|6541231678
sqlite> .exit
#
```

Figura 2.10: Ejecución SQLite3 con adb shell.

```
macmonte:platform-tools monte$ ./adb shell
# sqlite3 /data/data/es.uma.SQL/databases/gestionCentros
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"

sqlite> .tables
Infoalumno      android_metadata
sqlite> select * from Infoalumno;
1|pepe lopez|Primero|952132898
2|jose luis de la borbolla|Primero|6541231678
sqlite> .exit
#
```

Figura 2.11: Ejecución SQLite3 con adb shell.

Capítulo 3

Comunicaciones

3.1. Conexiones HTTP

```
https://developer.android.com/reference/java/net/  
HttpURLConnection.html
```

```
https://developer.android.com/training/basics/  
network-ops/connecting.html
```

Android proporciona el paquete estándar de red (java.net) conexiones http (httpClient). El acceso a Internet desde la aplicación requiere establecer el permiso en el manifiesto

```
android.permission.INTERNET
```

En caso que verifiquemos la conexión de nuestro sistema (listado 17) deberemos añadir además el permiso

```
android.permission.ACCESS_NETWORK_STATE
```

El acceso a la web desde la hebra principal no es una buena práctica y desde la versión 3 de Android obtendremos un error a menos que especifiquemos los siguientes comandos en el método *onCreate* de la actividad principal:

```
1 StrictMode.ThreadPolicy policy = new StrictMode.  
2 ThreadPolicy.Builder().permitAll().build();  
3 StrictMode.setThreadPolicy(policy);
```

```

1 private boolean isNetworkAvailable() {
2     ConnectivityManager cm = (ConnectivityManager)
3         getSystemService(Context.CONNECTIVITY_SERVICE);
4     NetworkInfo networkInfo = cm.getActiveNetworkInfo();
5
6     if (networkInfo != null && networkInfo.isConnected())
7         return true;
8     else return false;
9 }
```

Listing 17: Método utilizado para verificar la conexión

Esta es la clase estándar proporcionada por Java, por lo cual permite reutilizar código escrito en otras plataformas. El listado 18 muestra la conexión y la lectura mediante un stream de lectura (figura 3.1). De la misma forma podemos crear un stream de escritura.

```

1 private String Conectar (String urls){
2     String pagina=null;
3     try {
4         URL url = new URL(urls);
5         HttpURLConnection con = (HttpURLConnection) url.openConnection();
6         pagina = readStream(con.getInputStream());
7     }
8     catch (Exception e) {
9         e.printStackTrace();
10    }
11    return pagina;
12 }
```

Listing 18: Método Conectar url y lectura página web

Listing 3.1: Java Http

```

package com.example.joseamontenegromontes.httpcomm;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

import android.content.Context;
```



Figura 3.1: Aplicación http utilizando HttpURLConnection

```
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.net.Uri.Builder;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

public class HttpActivity extends AppCompatActivity {
    WebView webview;
    Button conectarB;
    EditText url, codeHttp;
    TextView info;

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

```
/*      StrictMode.ThreadPolicy policy = new StrictMode.
         ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);*/  
  
webview = (WebView) findViewById(R.id.webView1);
conectarB = (Button) findViewById(R.id.button1);
url = (EditText) findViewById(R.id.editText1);
info = (TextView) findViewById(R.id.textInfo);
codeHttp = (EditText) findViewById(R.id.
    editTextCodeHtml);  
  
webview.getSettings().setJavaScriptEnabled(true);
webview.getSettings().setDefaultTextEncodingName("utf
-8");
webview.getSettings().setLoadWithOverviewMode(true);
webview.getSettings().setUseWideViewPort(true);  
  
  
conectarB.setOnClickListener(new ImageButton.
    OnClickListener() {
    public void onClick(View v) {  
  
        String urlS=url.getText().toString();  
  
        if (!urlS.contains("http")){
            Uri urlB= new Uri.Builder();
            urlB.scheme("http");urlB.authority(urlS);
            urlS=urlB.build().toString();
        }  
  
        ConnectivityManager connMgr = (
            ConnectivityManager) getSystemService(
            Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.
            getActiveNetworkInfo();  
  
        if (networkInfo != null && networkInfo.
            isConnected()) {
            new HttpTask().execute(urlS);
        } else {
            Toast.makeText(getApplicationContext(),"No
                hay conexión a internet",Toast.
                LENGTH_SHORT).show();
        }  
    }  
}
```

```
        }
    });
}

class HttpTask extends AsyncTask<String, Void, String> {

    long ini, fin;

    protected String doInBackground(String... urls) {
        String pagina = null;
        String newUrls = urls[0];

        URL url = null;
        HttpURLConnection con = null;
        ini = System.currentTimeMillis();

        try {
            url = new URL(newUrls);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }

        try {
            con = (HttpURLConnection) url.openConnection();
            ;
            pagina = readStream(con.getInputStream());
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            con.disconnect();
        }

        fin = System.currentTimeMillis();

        return pagina;
    }

    protected void onPostExecute(String pagina) {
        webview.loadDataWithBaseURL(null, pagina, "text/
            html", "UTF-8", null);
        info.setText("Descargado en " + (fin - ini) + " "
            + milisegundos.);
        codeHttp.setText(pagina);
    }
}
```

```

        }

    private String readStream(InputStream in) {
        BufferedReader reader = null;
        StringBuffer pagina = new StringBuffer();
        try {
            reader = new BufferedReader(new
                InputStreamReader(in));
            String line = "";
            while ((line = reader.readLine()) != null) {
                //Log.d("http",line);
                pagina.append(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return pagina.toString();
    }

}

```

}

Listing 3.2: XML Http

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:weightSum="1">

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

```

```
<LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="230dp"
        android:layout_height="50dp"
        android:hint="@string/httpphint"
        android:inputType="textUri"
        android:singleLine="true" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="135dp"
        android:layout_height="wrap_content"
        android:text="@string/botonConectar" />

</LinearLayout>

<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="223dp" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:inputType="textAutoCorrect|none|textMultiLine"
        android:ems="10"
        android:id="@+id/editTextCodeHtml"
        android:layout_gravity="center_horizontal"
        android:layout_weight="0.87"
        android:editable="false" />

    <TextView
        android:id="@+id/textInfo"
        android:layout_width="fill_parent"
        android:layout_height="25dp"
        android:textSize="12dp" />

</LinearLayout>
```

3.2. Conexiones HTTPS

Mediante el paquete Net también podemos realizar conexiones seguras con la clase *httpsURLConnection* tal y como muestra el listado 19.

El protocolo SSL (https) además de permitir tener un canal de comunicación seguro, nos permite autenticar al servidor de la conexión mediante la utilización de certificados.

```

1 Certificate[] peerCertificates;
2 SSLSocketFactory factory = HttpsURLConnection.getDefaultSSLSocketFactory();
3
4 /* Conecta SSL al servidor y puerto (443 normalmente) */
5 SSLSocket socket = (SSLSocket)factory.createSocket(host, port);
6 sslSocket.startHandshake();
7
8 /* Obtenemos la cadena de Certificados del Servidor */
9 peerCertificates = sslSocket.getSession().getPeerCertificates();

```

Listing 19: Conexión https y obtención certificados

La aplicación SSLCertProyecto realiza la conexión a un servidor web seguro. Una vez establecida la conexión nos muestra los certificados X.509 utilizados en la conexión.

La aplicación muestra en la pantalla varios campos del certificado como: Usuario, Emisor, Número de Serie y Periodo de Validez. La información completa del certificado (incluyendo claves públicas) es mostrado en el Log.

Listing 3.3: Java Https

```

package es.uma.SSL;

import java.io.IOException;
import java.security.cert.Certificate;
import java.security.cert.X509Certificate;
import java.util.LinkedList;

import android.app.Activity;
import android.content.Context;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;

```

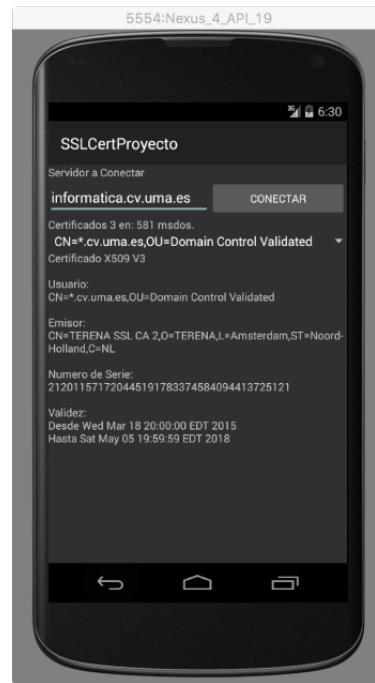


Figura 3.2: Aplicación SSL

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocket;
import javax.net.ssl.SSLSocketFactory;

public class SSLCertProyectoActivity extends AppCompatActivity
{
    String host;
    Context context;
    Spinner spinnerCert;
    EditText ServerName;
    TextView Info,Certs;
    @Override

    public void onCreate(Bundle savedInstanceState) {
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.main);
context = getApplicationContext();

Button ConectarB= (Button) findViewById(R.id.
    ConectarBoton);
ServerName= (EditText) findViewById(R.id.serverName);
Info= (TextView) findViewById(R.id.statusText);
Certs= (TextView) findViewById(R.id.textResultado);

spinnerCert= (Spinner) findViewById(R.id.spinner1);

spinnerCert.setVisibility(View.INVISIBLE);

ConectarB.setOnClickListener(new Button.
    OnClickListener() {
    public void onClick(View v) {

        if(ServerName.getText().length()==0){
            Toast.makeText(context, "Longitud del Servidor
                no valida", Toast.LENGTH_LONG).show();
        }
        else{
            host= ServerName.getText().toString();
            Info.setText(host);

            new SSLTask().execute();
        }
    }
}
);
}

private class SSLTask extends AsyncTask<Void,Void(Void> {

    Certificate[] peerCertificates; //Certificados que
        obtengo conexiOn
    public X509Certificate[] serverCertificates=null;

    int numCert=0; //Numeros Certificados Obtenidos
    private int port=443;
    long init,fin;

    protected Void doInBackground(Void... voids) {
        try {
            init=System.currentTimeMillis();
            SSLSocketFactory factory = HttpsURLConnection.

```

```
        getDefaultSSLSocketFactory();
        SSLSocket socket = (SSLSocket) factory.
            createSocket(host, port);

        HandshakeyCertificacion(socket);
        socket.close();
        fin=System.currentTimeMillis();
    } catch (IOException e) {
        Log.i("SSL", "Certificados "+e);
    }

    return null;
}

@Override
protected void onPostExecute(Void aVoid) {

    Info.setText("Certificados "+numCert+ " en: "+(fin
        -init)+" msdos. ");

    if (numCert>0){
        spinnerCert.setVisibility(View.VISIBLE);

        final LinkedList<X509Cert> certificados = new
            LinkedList<X509Cert>();

        for (int i = 0; i < numCert; ++i) {
            certificados.add(new X509Cert(
                serverCertificates[i]));
        }

        ArrayAdapter<X509Cert> spinner_adapter =
            new ArrayAdapter<X509Cert> (
                SSLCertProyectoActivity.this,
                android.R.layout.
                    simple_spinner_item, certificados)
            ;
        spinner_adapter.setDropDownViewResource(
            android.R.layout.
                simple_spinner_dropdown_item);
        spinnerCert.setAdapter(spinner_adapter);

        spinnerCert.setOnItemSelectedListener(
            new AdapterView.OnItemSelectedListener
            () {
                public void onItemSelected(
                    AdapterView<?> parent,
                    android
```

```
        .
        view
        .
        View
        v,
        int
        position
        ,
        long
        id
        ) {
        Certs.setText(certificados.get
        (position).Imprimir());
    }

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

public void HandshakeyCertificacion(SSLocket
sslSocket)
throws IOException {

// Hacemos el Handshake
try {
    sslSocket.startHandshake();
} catch (IOException e) {
    Log.e("SSLHandshake", e.getMessage());
}

SSLSession sesionSSL=sslSocket.getSession();
if (sesionSSL!=null) {

    // obtener los certificados
    peerCertificates = sesionSSL.
        getPeerCertificates();

    if (peerCertificates != null) {

        numCert = peerCertificates.length;

        if (numCert > 0) {
```

Listing 3.4: Java Cert

```
package es.uma.SSL;

import java.security.cert.X509Certificate;

public class X509Cert {

    X509Certificate cert;

    X509Cert (X509Certificate crt) {
        cert=crt;
    }
}
```

```

public String toString(){
    return cert.getSubjectDN().getName();
}

public String Imprimir(){
    String certs="Certificado X509 V"+ cert.getVersion() +"\n\
    n";
    certs=certs.concat("Usuario: \n"+cert.getSubjectDN() .
        getName()+" \n\n");
    certs=certs.concat("Emisor: \n"+cert.getIssuerDN().getName
        ()+" \n\n");
    certs=certs.concat("Número de Serie: \n"+cert.
        getSerialNumber()+" \n\n");
    certs=certs.concat("Validez: \nDesde "+cert.getNotBefore()+
        "\n"+ "Hasta "+cert.getNotAfter()));

    return certs;
}

}

```

Listing 3.5: XML Https

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/ServerInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/serverText" />

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <EditText
            android:id="@+id/serverName"
            android:layout_width="155dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.10"

```

```
        android:autoText="false"
        android:cursorVisible="false"
        android:freezesText="true"
        android:inputType="textNoSuggestions|
            textShortMessage"
        android:lines="1"
        android:singleLine="true" />

    <Button
        android:id="@+id/ConectarBoton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.15"
        android:text="@string/ConnectBoton" />
</LinearLayout>

<TextView
    android:id="@+id/statusText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textNoSuggestions"
    android:singleLine="true" />

<Spinner
    android:id="@+id/spinner1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/textResultado"
    android:layout_width="match_parent"
    android:layout_height="320dp"/>

</LinearLayout>
```

3.3. Bluetooth

<https://developer.android.com/guide/topics/connectivity/bluetooth.html>

Android incluye soporte para la pila Bluetooth, la cuál permite a un dispositivo intercambiar información con otro dispositivo.

En esta sección vamos a describir las tareas básicas para inicializar Bluetooth y buscar dispositivos cercanos y realizar la vinculación entre los dispositivos cercanos. En el enlace de ayuda de Android, podemos obtener el

código para la transmisión de información.

De las clases e interfaces de la api disponible, utilizaremos las siguientes:

- **BluetoothAdapter** Representa el adaptador local Bluetooth. Permite, descubrir otros dispositivos Bluetooth, obtener la lista de dispositivos vinculados y crear los elementos necesarios para las comunicaciones entre dispositivos.
- **BluetoothDevice** Representa un dispositivo Bluetooth remoto. Utilizamos para comunicarnos con un dispositivo remoto o para obtener información sobre él, tal como su nombre, dirección, y estado de vinculación.

3.3.1. Configurando Bluetooth

Inicialmente necesitamos verificar que el dispositivo soporta Bluetooth y verificar que está habilitado.

Si Bluetooth no está soportado, entonces no incluiríamos las capacidades de Bluetooth. Si Bluetooth está soportado, pero deshabilitado, la aplicación puede pedir al usuario que habilite Bluetooth dentro de la aplicación (figura 3.3). El listado 20 muestra el código para comprobar si el dispositivo soporta Bluetooth.

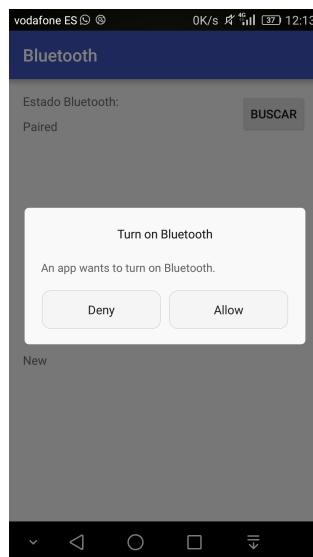


Figura 3.3: Petición de activar Bluetooth

```

1 BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
2 if (mBluetoothAdapter == null) {
3     // Dispositivo no soporta Bluetooth
4 }
```

Listing 20: Dispositivo soporta Bluetooth

El listado 21 muestra el código necesario para enviar un Intent al sistema que permita activar el interfaz Bluetooth. Hemos utilizado el método *startActivityForResult*, con lo cuál es necesario implementar el método *onActivityResult* para comprobar las acciones del usuario.

```

1 if (!mBluetoothAdapter.isEnabled()) {
2     Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
3     startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
4 }
5
6 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
7
8     if (requestCode == REQUEST_ENABLE_BT) {
9
10         if (resultCode == RESULT_OK) {
11             estadoB.setText("Conectado");
12             Log.v("Bluetooth", "Usuario activo el Bluetooth");
13             paired();
14         }
15         else {
16             estadoB.setText("No Conectado");
17             Log.v("Bluetooth", "Usuario NO activo el Bluetooth");
18         }
19     }
20 }
```

Listing 21: Habilitar Bluetooth

Es posible que el usuario desconecte el interfaz de Bluetooth durante la ejecución de la aplicación, para conocer cualquier cambio en la interfaz de Bluetooth, será necesario registrar *BroadcastReceiver* para el intent *BluetoothAdapter.ACTION_STATE_CHANGED*.

```

1      IntentFilter filter = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
2          registerReceiver(mReceiver, filter);
3
4
5  private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
6      public void onReceive(Context context, Intent intent) {
7          String action = intent.getAction();
8
9          if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)) {
10              final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
11                                              BluetoothAdapter.ERROR);
12
13              switch (state) {
14                  case BluetoothAdapter.STATE_OFF:
15                      adapterPaired.clear();
16                      adapterNew.clear();
17                      estadoB.setText("Bluetooth off");
18                      break;
19                  case BluetoothAdapter.STATE_TURNING_OFF:
20                      estadoB.setText("Turning Bluetooth off...");
21                      break;
22                  case BluetoothAdapter.STATE_ON:
23                      estadoB.setText("Bluetooth on"); paired();
24                      break;
25                  case BluetoothAdapter.STATE_TURNING_ON:
26                      estadoB.setText("Turning Bluetooth on...");
27                      break;
28              }
29          }
30      }
}

```

Listing 22: Registrar un BroadcastReceiver para recibir cambio de estados

3.3.2. Buscando dispositivos

Utilizando *BluetoothAdapter*, podemos encontrar dispositivos remotos o ver la lista de dispositivos vinculados previamente. El listado 23 realiza la petición de los dispositivos vinculados.

```

1  private void paired() {
2      Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
3
4      if (pairedDevices.size() > 0) {
5          for (BluetoothDevice device : pairedDevices) {
6              adapterPaired.add(device.getName() + "\n" + device.getAddress());
7              arrayListPairedBluetoothDevices.add(device);
8          }
9      }
10 }

```

Listing 23: Solicitar los dispositivos vinculados.

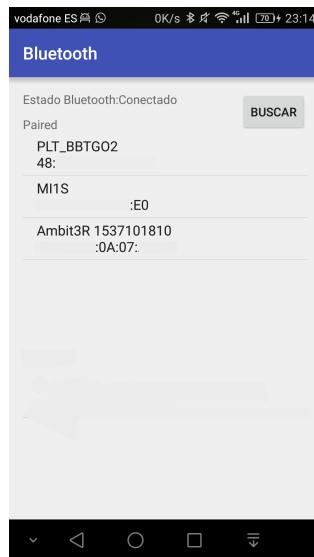


Figura 3.4: Ejemplo de lista de elementos vinculados

3.3.3. Descubriendo nuevos dispositivos

Para descubrir nuevos dispositivos realizaremos el código 24

```
1  public void onClick(View view) {
2      final boolean b = mBluetoothAdapter.startDiscovery();
3      Log.d("Bluetoooh", "Discovery resultado: "+b);
4  }
```

Listing 24: Registrar un BroadcastReceiver para recibir cambio de estados

La ejecución del método *startDiscovery* hace necesario registrar un BroadcastReceiever para obtener el resultado de la ejecución del método.

```

1  private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
2      public void onReceive(Context context, Intent intent) {
3          String action = intent.getAction();
4          if (BluetoothDevice.ACTION_FOUND.equals(action)) {
5              BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
6              adapterNew.add(device.getName() + "\n" + device.getAddress());
7              arrayListBluetoothDevices.add(device);
8          }
9      }
10 };
11
12 IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
13 registerReceiver(mReceiver, filter);

```

Listing 25: Registrar un BroadcastReceiver para recibir cambio de estados

Una vez pulsado el botón *Buscar*, los dispositivos disponibles serán incluidos en una nueva lista de dispositivos no vinculados.

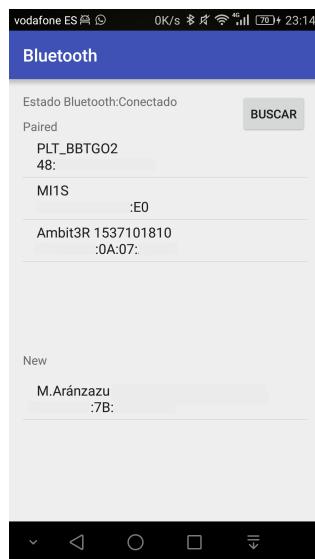


Figura 3.5: Nuevos dispositivos

3.3.4. Vinculación y Desvinculación

En la aplicación tenemos dos listas, los dispositivos vinculados y los no vinculados abajo. Los dispositivos vinculados pueden ser desvinculados y los no vinculados pasar a ser vinculados con el código 26.

```

1 private void pairDevice(BluetoothDevice device) {
2     try {
3         Method method = device.getClass().getMethod("createBond", (Class[]) null);
4         method.invoke(device, (Object[]) null);
5     } catch (Exception e) {
6         e.printStackTrace();
7     }
8 }
9
10
11 private void unpairDevice(BluetoothDevice device) {
12     try {
13         Method method = device.getClass().getMethod("removeBond", (Class[]) null);
14         method.invoke(device, (Object[]) null);
15     } catch (Exception e) {
16         e.printStackTrace();
17     }
18 }
19 }
```

Listing 26: Vinculación y Desvinculación de Dispositivos

La figura 3.6 muestra el mensaje previo para la vinculación de dos dispositivos. Una vez vinculados, el dispositivo pasa a la lista de vinculados (figura 3.7). Finalmente la figura 3.8 muestra la desvinculación de un dispositivo al pulsar en la lista de dispositivos vinculados.

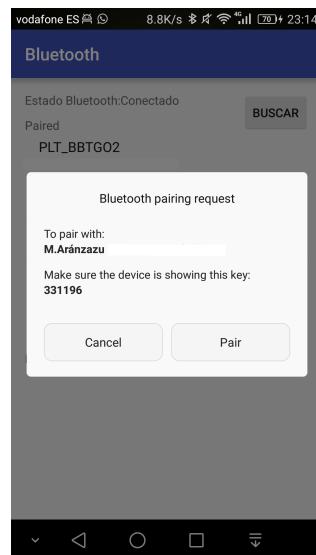


Figura 3.6: Solicitud de vinculación

Listing 3.6: Java Bluetooth

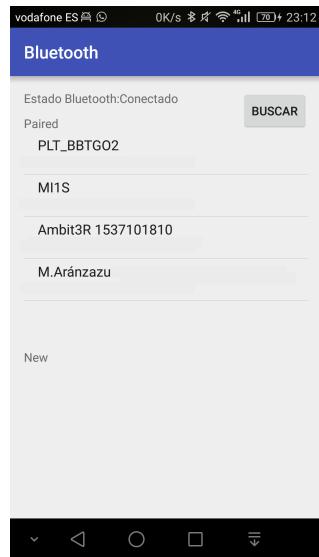


Figura 3.7: Dispositivo vinculado

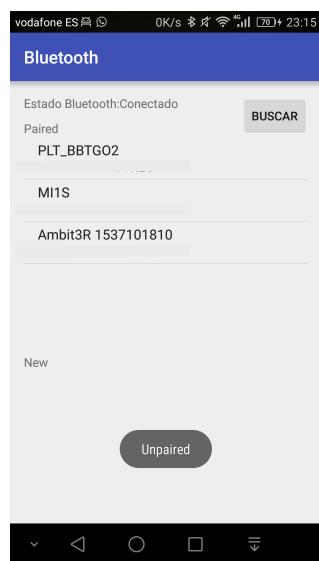


Figura 3.8: Dispositivo desvinculado

```
package com.example.joseamontenegromontes.bluetooth;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Set;

public class BluetoothActivity extends AppCompatActivity {
    int REQUEST_ENABLE_BT=1;
    BluetoothAdapter mBluetoothAdapter;

    TextView estadoB;
    ListView paired,newDiscover;
    ArrayAdapter adapterPaired,adapterNew;

    ArrayList<BluetoothDevice> arrayListPairedBluetoothDevices
            = null;
    ArrayList<BluetoothDevice> arrayListBluetoothDevices =
            null;
    boolean broadcastRegister=false;

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

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter
                ();
        if (mBluetoothAdapter == null) {
            Toast.makeText(this,"Dispositivo no soporta
                    bluetooth",Toast.LENGTH_LONG).show();
        }else {
```

```
estadoB = (TextView) findViewById(R.id.estadoText)
        ;
paired = (ListView) findViewById(R.id.listView);
newDiscover = (ListView) findViewById(R.id.
        listView2);
arrayListPairedBluetoothDevices = new ArrayList<
        BluetoothDevice>();
arrayListBluetoothDevices = new ArrayList<
        BluetoothDevice>();

final ArrayList<String> list = new ArrayList<>();
adapterPaired = new ArrayAdapter(this, android.R.
        layout.simple_list_item_1, list);
paired.setAdapter(adapterPaired);

final ArrayList<String> listNew = new ArrayList
        <>();
adapterNew = new ArrayAdapter(this, android.R.
        layout.simple_list_item_1, listNew);
newDiscover.setAdapter(adapterNew);

paired.setOnItemClickListener(new AdapterView.
        OnItemClickListener() {
    @Override

        public void onItemClick(AdapterView<?>
                adapterView, View view, int i, long l) {

            Toast.makeText(getApplicationContext(),
                    "Pulsado " + i + " " + adapterView
                    .getItemAtPosition(i), Toast.
                    LENGTH_LONG).show();

            BluetoothDevice device =
                    arrayListPairedBluetoothDevices.get(i)
                    ;

            unpairDevice(device);
            arrayListPairedBluetoothDevices.remove(i);

            adapterPaired.clear();

            for (BluetoothDevice newDevice :
                    arrayListPairedBluetoothDevices) {
                adapterPaired.add(newDevice.getName()
                        + "\n" + newDevice.getAddress());
            }
        }
    }
```

```
        }
    });

newDiscover.setOnItemClickListener(new AdapterView
    .OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?>
        adapterView, View view, int i, long l) {

        Toast.makeText(getApplicationContext(),
            "Pulsado " + i + " " + adapterView
                .getItemAtPosition(i), Toast.
                LENGTH_LONG).show();

        BluetoothDevice device =
            arrayListBluetoothDevices.get(i);
        pairDevice(device);

        arrayListBluetoothDevices.remove(i);
        adapterNew.clear();

        for (BluetoothDevice newDevice :
            arrayListBluetoothDevices) {
            adapterNew.add(newDevice.getName() +
                "\n" + newDevice.getAddress());
        }

    }
});

if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(
        BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent,
        REQUEST_ENABLE_BT);
} else {
    estadoB.setText("Conectado");
    paired();
}

// Registro BroadCastReceiver
IntentFilter filter = new IntentFilter(
    BluetoothDevice.ACTION_FOUND);
filter.addAction(BluetoothAdapter.
    ACTION_STATE_CHANGED);
filter.addAction(BluetoothDevice.
```

```

        ACTION_BOND_STATE_CHANGED);
registerReceiver(mReceiver, filter);

        broadcastRegister = true; // Para saber onDestroy
    }
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (broadcastRegister) unregisterReceiver(mReceiver);
}

protected void onActivityResult(int requestCode, int
    resultCode, Intent data) {

    if (requestCode == REQUEST_ENABLE_BT) {

        if (resultCode == RESULT_OK) {
            estadoB.setText("Conectado");
            Log.v("Bluetooth", "Usuario activo el
                Bluetooth");
            paired();
        }
        else {
            estadoB.setText("No Conectado");
            Log.v("Bluetooth", "Usuario NO activo el
                Bluetooth");
        }
    }
}

private void paired(){
    Set<BluetoothDevice> pairedDevices = mBluetoothAdapter
        .getBondedDevices();

    if (pairedDevices.size() > 0) {
        for (BluetoothDevice device : pairedDevices) {
            adapterPaired.add(device.getName() + "\n" +
                device.getAddress());
            arrayListPairedBluetoothDevices.add(device);
        }
    }
}

///
private final BroadcastReceiver mReceiver = new
    BroadcastReceiver() {

```

```
public void onReceive(Context context, Intent intent)
{
    String action = intent.getAction();

    if (action.equals(BluetoothAdapter.
ACTION_STATE_CHANGED)) {
        final int state = intent.getIntExtra(
            BluetoothAdapter.EXTRA_STATE,
            BluetoothAdapter.ERROR);
        switch (state) {
            case BluetoothAdapter.STATE_OFF:
                adapterPaired.clear();
                adapterNew.clear();
                estadoB.setText("Bluetooth off");
                break;
            case BluetoothAdapter.STATE_TURNING_OFF:
                estadoB.setText("Turning Bluetooth off
                    ...");
                break;
            case BluetoothAdapter.STATE_ON:
                estadoB.setText("Bluetooth on");
                paired();
                break;
            case BluetoothAdapter.STATE_TURNING_ON:
                estadoB.setText("Turning Bluetooth on
                    ...");
                break;
        }
    }

    if (BluetoothDevice.ACTION_FOUND.equals(action)) {
        BluetoothDevice device = intent.
            getParcelableExtra(BluetoothDevice.
                EXTRA_DEVICE);
        adapterNew.add(device.getName() + "\n" +
            device.getAddress());
        arrayListBluetoothDevices.add(device);
    }

    if (BluetoothDevice.ACTION_BOND_STATE_CHANGED.
equals(action)) {
        final int state          = intent.getIntExtra(
            BluetoothDevice.EXTRA_BOND_STATE,
            BluetoothDevice.ERROR);
        final int prevState      = intent.getIntExtra(
            BluetoothDevice.EXTRA_PREVIOUS_BOND_STATE,
            BluetoothDevice.ERROR);

        if (state == BluetoothDevice.BOND_BONDED &&
```

```

prevState == BluetoothDevice.BOND_BONDING)
{
    Toast.makeText(getApplicationContext(),"
        Paired",Toast.LENGTH_LONG).show();
    BluetoothDevice device = intent.
        getParcelableExtra(BluetoothDevice.
            EXTRA_DEVICE);
    adapterPaired.add(device.getName() + "\n"
        + device.getAddress());
    arrayListPairedBluetoothDevices.add(device
        );
}

} else if (state == BluetoothDevice.BOND_NONE
    && prevState == BluetoothDevice.
    BOND_BONDED) {
    Toast.makeText(getApplicationContext(),"
        Unpaired",Toast.LENGTH_LONG).show();

}

}
};

public void onClick(View view) {
    final boolean b = mBluetoothAdapter.startDiscovery();
    Log.d("Bluetoooh","Discovery resultado: "+b);
}

private void pairDevice(BluetoothDevice device) {
    try {
        Method method = device.getClass().getMethod(""
            createBond", (Class[]) null);
        method.invoke(device, (Object[]) null);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void unpairDevice(BluetoothDevice device) {
    try {
        Method method = device.getClass().getMethod(""
            removeBond", (Class[]) null);
        method.invoke(device, (Object[]) null);
    } catch (Exception e) {

```

```
        e.printStackTrace();
    }
}

}
```

Listing 3.7: XML Bluetooth

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.joseamontenegroMontes.bluetooth
        .BluetoothActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/blueState"
        android:id="@+id/textView" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/textView"
        android:layout_toEndOf="@+id/textView"
        android:id="@+id/estadoText" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="250dp"
        android:id="@+id/listView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/textView"
        android:layout_marginTop="30dp" />

    <Button
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:text="Buscar"
    android:id="@+id/buttonDis"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:onClick="onClick" />

<ListView
    android:layout_width="wrap_content"
    android:layout_height="150dp"
    android:id="@+id/listView2"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_alignParentEnd="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New"
    android:id="@+id/textView2"
    android:layout_below="@+id/listView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Paired"
    android:id="@+id/textView3"
    android:layout_alignBottom="@+id/buttonDis"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
</RelativeLayout>
```
