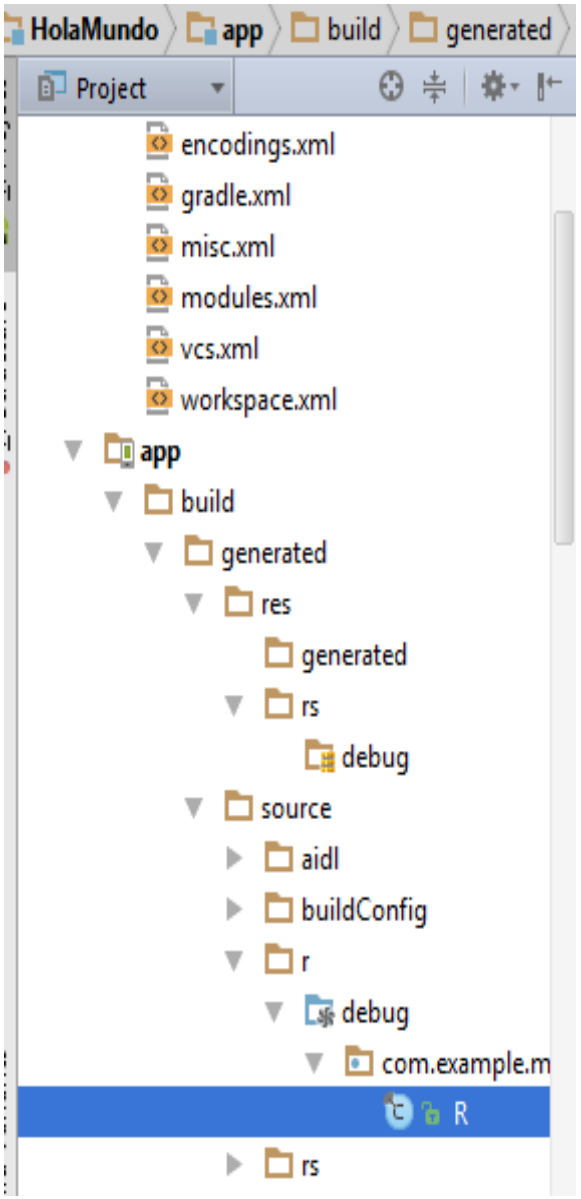
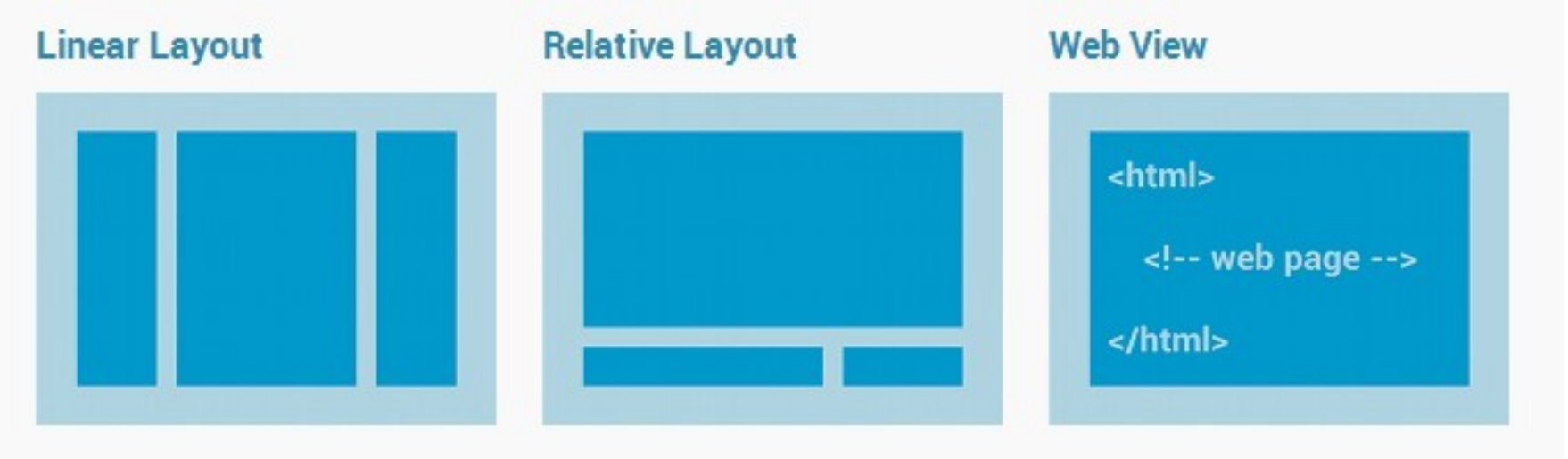
Creación de una interfaz de usuario usando ficheros XML

Layout


HolaMundo
app
build
generated
Project
encodings.xml
gradle.xml
misc.xml
modules.xml
vcs.xml
workspace.xml
app
build
generated
res
generated
rs
debug
source
aidl
buildConfig
r
debug
com.example.m
R
rs

Layout

- Android proporciona una alternativa para el diseño de interfaces de usuario. Se trata de los ficheros de diseño basados en XML. Vamos a ver uno de estos ficheros.
- Para ello accede al fichero res/layout/main.xml de uno de nuestros proyectos.
- Hay varios tipos de Layouts, cada uno tiene unas ventajas respecto a los otros, pero se pueden combinar todos dentro de nuestro diseño.



Layout

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

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

- Se introduce un elemento de tipo LinearLayout., su función es contener otros elementos de tipo View. Este LinearLayout tiene cuatro atributos.
- El primero, **xmlns:android**, es una declaración de un espacio de nombres de XML que utilizaremos en Android (Este parámetro solo es necesario especificarlo en el primerelemento).
- El atributo **android:orientation** indica que los elementos se van a distribuir de forma vertical.
- Los otros dos permiten definir el ancho y alto de la vista. En el ejemplo se ocupará todo el espacio disponible en la pantalla puesto que hemos utilizado los valores fill_parent. También tenemos el valor wrap_content para que el tamaño dependa del valor que haya dentro del elemento.

Propiedades en el fichero /res/layout/*.xml

android:id.
Se trata de un número entero que sirve para identificar cada objeto view de forma única dentro de nuestro programa, cuando lo declaramos a través de un xml de resource podemos hacer referencia a la clase de recursos R usando una @, esto es imprescindible, ya que si no no podremos identificar nuestros elementos en nuestro programa para después usarlos y/o modificarlos, veamos algunos ejemplos:

- android:id=”@id/boton”. Hace referencia a un id ya existente asociado a la etiqueta “boton”, esto se usa para cuando usamos los Layout Relativos, ya que para ubicar los elementos, lo hacemos indicando por ejemplo que un botón lo insertamos a la derecha de otro, pues bien ese otro se pone así.
- android:id=”@+id/boton2”. Esto crea una nueva etiqueta en la clase R llamada “boton2”.

atributos height, width (Altura y Ancho)
Otra propiedad importante es el Alto y el Ancho de los controles y Layouts, ya que para que Android sepa dibujar un objeto View debemos proveerle estos datos, y podemos hacerlo de 3 formas:

- android:layout_width="40dp". Indicando un número exacto que definamos, usaremos 40dp como unidad de medida, dp significa: Densidad de píxeles independientes, una unidad abstracta que se basa en la densidad física de la pantalla. Esta unidad es perfecta para buscar la compatibilidad con TODAS las pantallas de móvil o tables, ya que es una medida proporcional.

Propiedades en el fichero /res/layout/*.xml

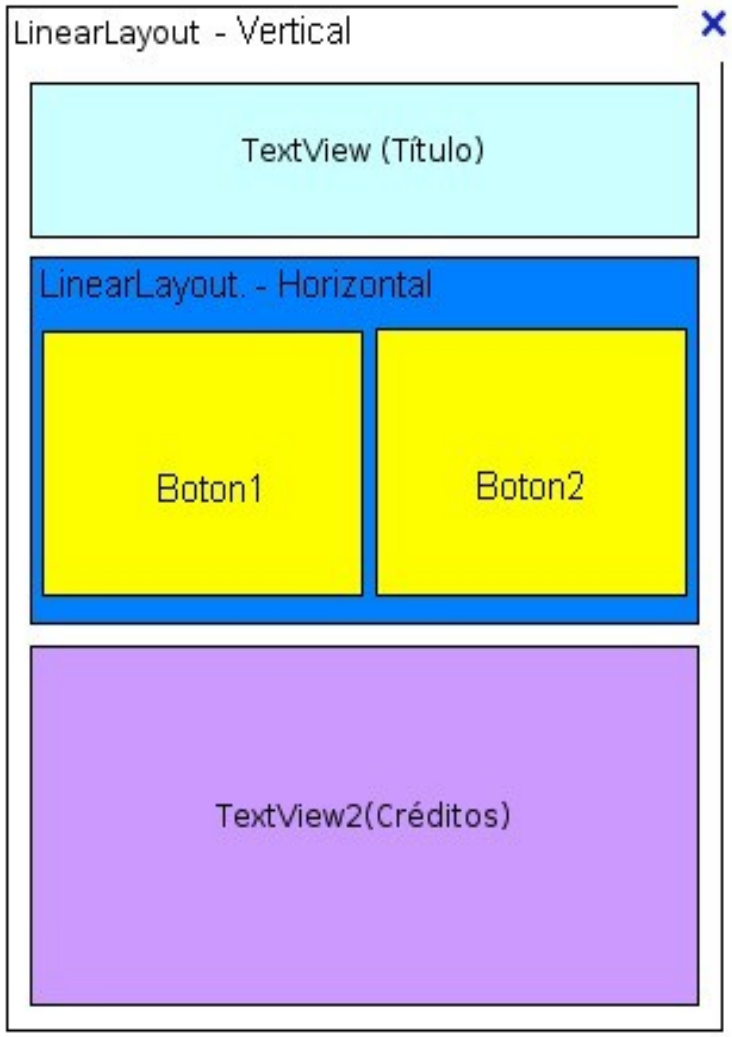
- La constante FILL_PARENT que indica que la vista intentará ser tan grande como su padre (menos el padding), es decir se ajusta a tope!
- La constante WRAP_CONTENT que indica que la vista intentará ser lo suficientemente grande para mostrar su contenido (más el padding).
- **android:layout_weight. P**ermitir dar a los elementos contenidos en el layout unas dimensiones proporcionales entre ellas. Si incluimos en un LinearLayout vertical dos cuadros de texto (EditText) y a uno de ellos le establecemos un layout_weight=”1” y al otro un layout_weight=”2” conseguiremos como efecto que toda la superficie del layout quede ocupada por los dos cuadros de texto y que además el segundo sea el doble (relación entre sus propiedades weight) de alto que el primero.

- **android:layout_gravity="center"**. Se usa para centrar, es la 'gravedad' además las puedes combinar, es decir, Center_Horizontal|Top. <-- Esto te lo centra horizontal y lo ajusta en vertical arriba.

- Para el resto de Atributos, cada elemento tendrá los propios, basta con poner el cursor dentro de la etiqueta del Layout que estemos colocando y pulsar las teclas Ctrl+Espacio para que Android Studiote recomiende las propiedades del elemento que estamos insertando

Ejemplos: LinearLayout

- Este tipo de layout apila uno tras otro todos sus elementos hijos de forma horizontal o vertical según se establezca su propiedad android:orientation="vertical" y android:orientation="Horizontal" .




```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/an
droid"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/etiquetaPrueba"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="90dp"
        android:layout_marginTop="80dp"
        android:text="Etiqueta de prueba." />

<LinearLayout>
    <Button
        android:id="@+id/"boton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text= "Boton1" />
    <Button
        android:id="@+id/boton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Boton2" />
 <LinearLayout>

     <TextView
        android:id="@+id/etiquetaPrueba"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight=1
       android:text="Etiqueta de prueba." />

</LinearLayout>
```

Ejemplos: LinearLayout

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/re
s/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <AnalogClock
        android:id="@+id/reloj"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="90dp"
        android:layout_marginTop="25dp"/>

    <CheckBox
        android:id="@+id/checkPrueba"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ChekBox de prueba" />

    <Button
        android:id="@+id/botonPrueba"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Boton de prueba" />

    <TextView
        android:id="@+id/etiquetaPrueba"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="90dp"
        android:layout_marginTop="80dp"
        android:text="Etiqueta de prueba." />

</LinearLayout>
```

Ejemplos: TableLayout


```
<TableLayout
xmlns:android="http://schemas.android.com/apk/re
s/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

 <TableRow>
    <AnalogClock
       android:id="@+id/reloj"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"/>
    <CheckBox
        android:id="@+id/checkPrueba"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ChekBox de prueba" />
 </TableRow>

 <TableRow>
    <Button
        android:id="@+id/botonPrueba"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Boton Prueba" />
    <TextView
        android:id="@+id/etiquetaPrueba"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Etiqueta de prueba." />
    </TableRow>

</TableLayout>
```

Ejemplos: RelativeLayout

- Este tipo de layout es útil cuando queremos hacer cosas un poco mas complejas y vistosas.
- Este layout permite especificar la posición de cada elemento de forma relativa a su elemento padre o a cualquier otro elemento incluido en el propio layout. De esta forma, al incluir un nuevo elemento A podremos indicar por ejemplo que debe colocarse debajo del elemento B y alineado a la derecha del layout padre.
- Veamos esto en el ejemplo siguiente:

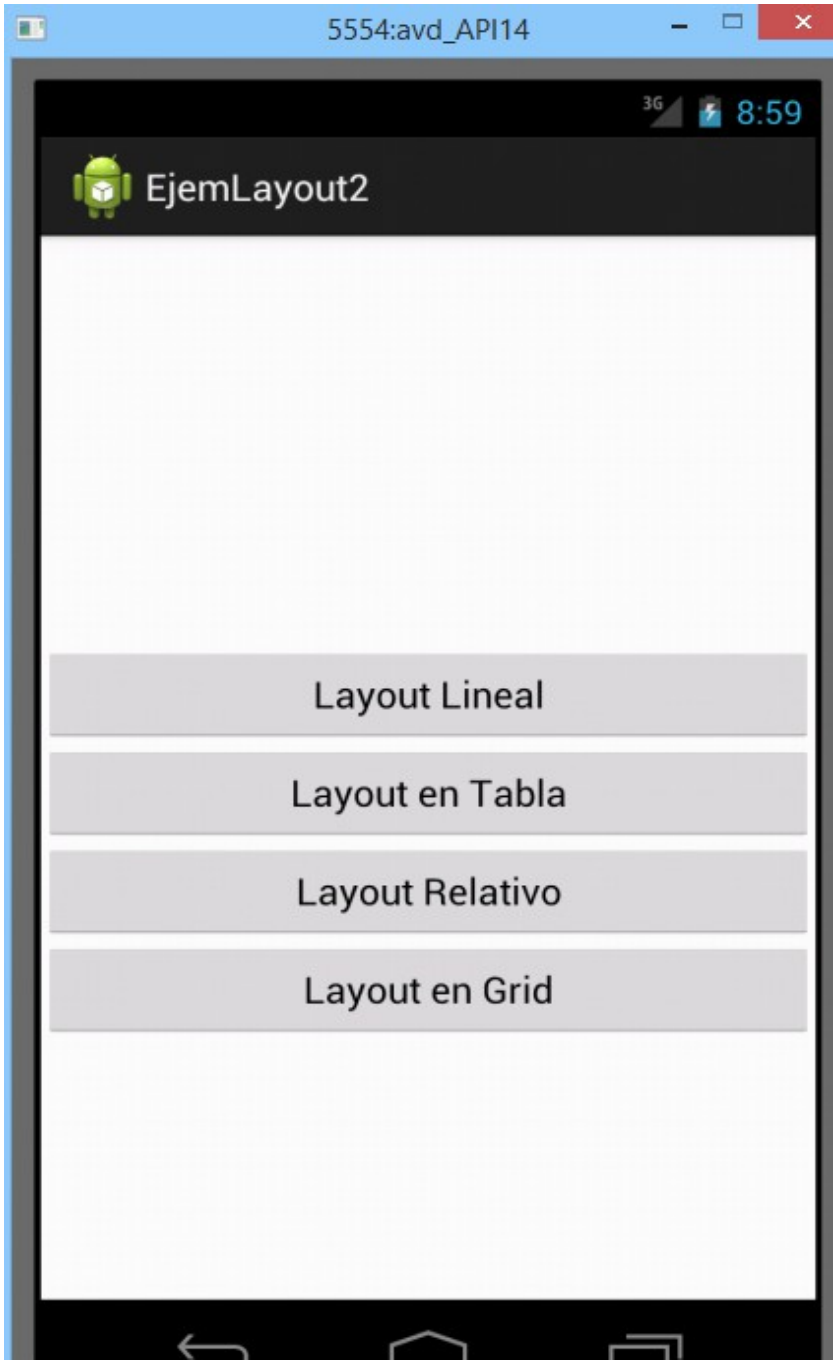
Ejemplos: RelativeLayout

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

<AnalogClock
    android:id="@+id/reloj"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

<CheckBox
    android:id="@+id/checkPrueba"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ChekBox de prueba"
    android:layout_below="@+id/reloj"/>

<Button
    android:id="@+id/botonPrueba"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Boton Prueba"
    android:layout_alignParentRight="true"/>

<TextView
    android:id="@+id/etiquetaPrueba"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Etiqueta de prueba."
    android:layout_alignParentBottom="true"/>
</RelativeLayout>
```

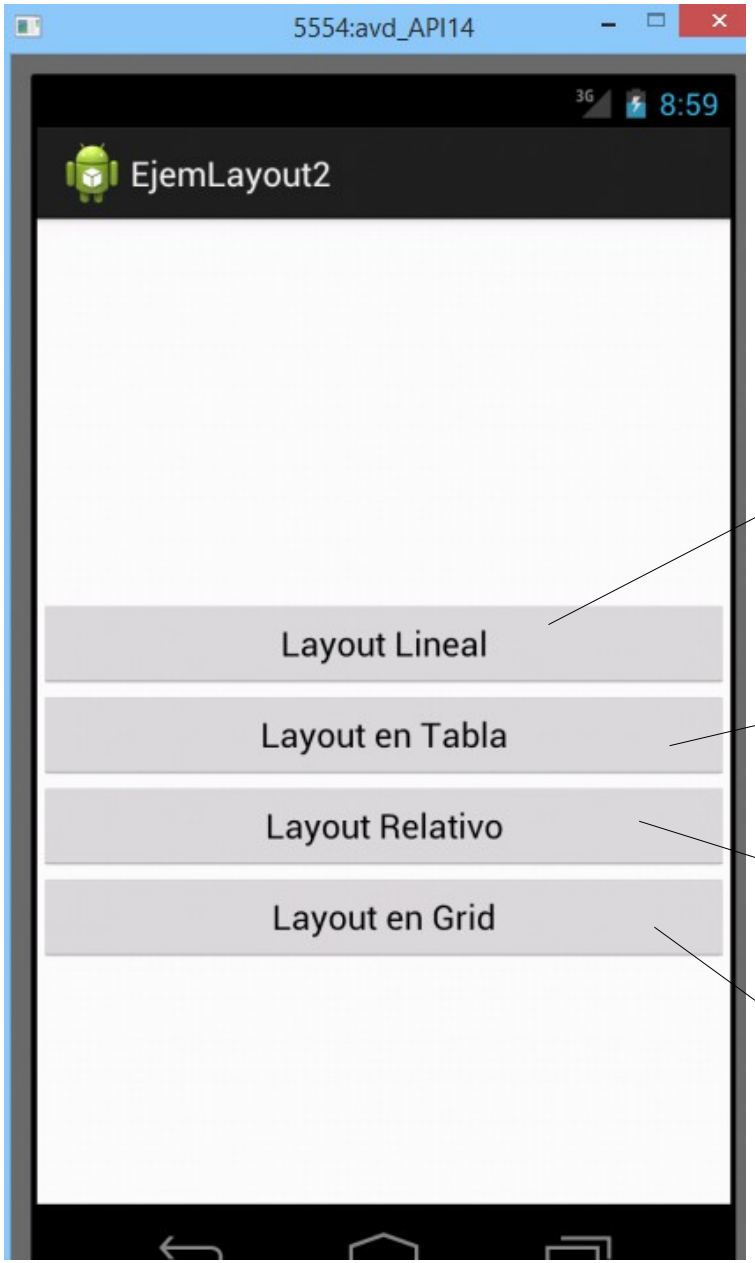
Ejercicios recopilatorios

1.-Completar: Coloca todas las componentes de los ejemplos anteriores también sobre un GridLayout

2.-Une todos los ejemplos en una sola aplicación cuya actividad principal tenga cuatro botones. El evento onClick(View v) sobre cada botón invoca a distintas actividades cuya IU visualizan los Layout vistos en los ejemplo de las transparencias



Ejercicios recopilatorios



A resolver....