Ejercicio 3

i) Constraint Layout: Es un diseño en Android que le brinda formas adaptables y flexibles para crear vistas para sus aplicaciones.

La ventaja que ofrece este Layout a la hora de diseñarlo es que puedes poner una relación de cada elemento con otros elementos o con el mismo Layout haciéndolo totalmente responsive, teniendo la seguridad de que el control o elemento se mantendrá en su posición relativa dentro de la pantalla sea cual sea su tamaño. Es un versión mejorada de RelativeLayout, que permite una edición visual desde el editor.

Los componentes se pueden añadir en cualquiera de los cuatro lados del contenedor e ir añadiendo mas elementos pegados a estos.

Ejemplo:

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

<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

   android:layout_width="match_parent"

   android:layout_height="match_parent">

   <TextView

      android:id="@+id/textView"

      android:layout_width="match_parent"

      android:layout_height="wrap_content"

      android:gravity="center"

      android:text="Ejemplo ConstraintLayout" />

   <TextView

      android:id="@+id/textView"

      android:layout_width="299dp"

      android:layout_height="33dp"

      android:layout_marginEnd="40dp"

      android:layout_marginStart="8dp"

      android:text="TextView"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:layout_editor_absoluteY="41dp" />
<EditText
    android:id="@+id/editText"
    android:layout_width="148dp"
    android:layout_height="52dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="20dp"
    android:ems="5"
    android:inputType="textPersonName"
    android:text="Name"
    app:layout_constraintBottom_toTopOf="@+id/button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.881"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintVertical_bias="0.464" />
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginTop="168dp"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
```

```
    tools:layout_editor_absoluteX="0dp" />

</android.support.constraint.ConstraintLayout>
```

ii) Relative Layout: Este Layout posiciona los controles o elementos hijos de forma relativa entre unos y otros, por ejemplo: debajo de, o ala izquierda de, es decir, cada elemento depende de uno o varios elementos para estructurarse, de esta manera, si el tamaño de la pantalla aumenta, se reordena para quedar alineado con los bordes o con los elementos de los que es dependiente.

Ejemplo:

```
<RelativeLayout

  xmlns:android="http://schemas...

  android:layout_height="match_parent"

  android:layout_width="match_parent">

 <AnalogClock

  android:id="@+id/AnalogClock01"

  android:layout_width="wrap_content"

  android:layout_height="wrap_content"

  android:layout_alignParentTop="true"/>

 <CheckBox

  android:id="@+id/CheckBox01"

  android:layout_width="wrap_content"

  android:layout_height="wrap_content"

  android:layout_below="@id/AnalogClock01"

  android:text="Un checkBox"/>

 <Button

  android:id="@+id/Button01"

  android:layout_width="wrap_content"

  android:layout_height="wrap_content"

  android:text="Un botón"

  android:layout_below="@id/CheckBox01"/>

 <TextView

  android:id="@+id/TextView01"
```

```
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentBottom="true"
  android:text="Un texto cualquiera"/>
</RelativeLayout>
```

iii) Linear Layout: Este Layout distribuye los elementos uno detrás de otro, bien de forma horizontal o vertical. Esto es muy útil para la creación de formularios, ya que deja una estructura ordenada y adaptada para todos los tamaños de pantallas.

Ejemplo Vertical:

```
<LinearLayout  xmlns:android="http://...
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation ="vertical">
 <AnalogClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
 <CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Un checkBox"/>
 <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Un botón"/>
 <TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Un texto cualquiera"/>
</LinearLayout>
```

Ejemplo Horizontal:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Ejemplo Horizontal" />
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />
    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="5"
        android:inputType="textPersonName"
        android:text="Name" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:text="Button" />
</LinearLayout>
```

iv) Frame Layout: Este Layout posiciona las vistas usando todo el contenedor, sin distribuirlas espacialmente. Suele utilizarse cuando queremos que varias vistas ocupen un mismo lugar. Podemos hacer que solo una sea visible, o superponerlas.Para modificar la visibilidad de un elemento utilizaremos la propiedad visibility. Además, este Layout se estructura en marcos que se pueden ordenar para dar prioridad a las capas que deseemos.

Ejemplo:

```
<FrameLayout xmlns:android="http://schemas...
 android:layout_height="mach_parent"
 android:layout_width="match_parent">
 <AnalogClock
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
 <CheckBox
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Un checkBox"/>
 <Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Un botón"
   android:visibility="invisible"/>
 <TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Un texto cualquiera"
   android:visibility="invisible"/>
</FrameLayout>
```

v) Table Layout: Este Layout se estructura en forma de tabla, con celdas, de forma que se puede seleccionar el número de filas y de columnas que deseemos para nuestro layout. Cada componente se añade en una de las celdas. Dentro de la etiqueta de éste Layout para cada fila debemos poner la etiqueta TableRow.

Ejemplo:

```
<TableLayout xmlns:android=”http://...
      android:layout_height="match_parent"
      android:layout_width="match_parent">
 <TableRow>
   <AnalogClock
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"/>
   <CheckBox
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Un checkBox"/>
 </TableRow>
 <TableRow>
   <Button
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Un botón"/>
   <TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Un texto cualquiera"/>
 </TableRow>
</TableLayout>
```

vi) Grid Layout: Este Layout es básicamente como un ListView, cuyos elementos están organizados en una cuadrícula estricta. Está conectado a un adaptador y recupera vistas del adaptador cuando el usuario se desplaza por él. Todos los elementos en la cuadrícula deben ser del mismo tamaño. El usuario puede mover un selector visible a través de cada elemento. El objetivo de un GridLayout es mostrar los datos de un

adaptador y permitir al usuario navegar y seleccionar cada uno de los elementos mostrados. La única diferencia con ListView es que los elementos se colocan en una cuadrícula en lugar de en una lista vertical.

Ejemplo:

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/GridLayout1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:columnCount="2"
   android:rowCount="2"
   android:orientation="horizontal"
   tools:context=".GridXMLActivity" >
   <Button
      android:id="@+id/button1"
      android:layout_gravity="left|top"
      android:text="Button" />
   <Button
      android:id="@+id/button2"
      android:layout_gravity="left|top"
      android:text="Button" />
   <Button
      android:id="@+id/button3"
      android:layout_gravity="left|top"
      android:text="Button" />
   <Button
      android:id="@+id/button4"
      android:layout_gravity="left|top"
      android:text="Button" />
</GridLayout>
```

Ejercicio 4

La carpeta app es el módulo del proyecto que contendrá todo el software de la aplicación de ejemplo.
La carpeta principal del módulo es la carpeta main dentro de la carpeta src. Dentro de main nos encontramos el fichero AndroidManifest.xml, este fichero contiene la definición XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación, sus componentes ol os permisos necesarios para su ejecución.
Dentro de main tambien nos encontraremos con la carpeta java. Esta carpeta contendrá todo el código fuente de la aplicación, clases auxiliares, etc. También es donde encontraremos el MainActivity.java que es el código base de la pantalla principal de la aplicación.

Junto a la carpeta java tambien encontraremos la carpeta res que contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, layouts, cadenas de texto, etc. Los recursos se dividen en distintos tipos agrupados por carpetas:

- grawable: Contiene imagenes y otros elementos gráficos usados por la aplicación. Esta carpeta a su vez se divide en otras subcarpetas dependiendo de la densidad de la pantalla del dispositivo.

    1. drawable: Recursos independientes de la densidad.
    2. drawable-ldpi: densidad baja.
    3. drawable-mdpi: densidad media.
    4. drawable-hdpi: densidad alta.
    5. drawable-xhdpi: densidad muy alta.
    6. drawable-xxhdpi: densidad muy muy alta.

- mipmap: Contiene los iconos de lanzamiento de la aplicación para las distintas densidades de pantalla existentes. Esta carpeta también se divide en subcarpetas.

    7. mipmap-mdpi.
    8. mipmap-hdpi.
    9. etc.

- layout: Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Los layout se dividen sugún la orientación.

    10. layout: vertical.
    11. layout-land: horizontal.

- anim y animator: Contienen la definición de las animaciones utilizadas por la aplicación.
- color: Contiene ficheros XML de definición de listas de colores según estado.
- menu: Contiene la definición XML de los menús de la aplicación.

- xml: Contiene otros ficheros XML de datos utilizados por la aplicación.
- raw: Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetos de recursos.
- values: Contiene los ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (string.xml), estilos (styles.xml), colores (colors.xml), arrays de valores (arrays.xml), tamaños (dimens.xml), etc.

En app tambien encontramos build.grandle que contiene la información necesaria para la compilación del proyecto. En un proyecto pueden existir varios ficheros build.grandle para definir determinados parámetros a distintos niveles.