INTERFACES.-LAYOUTS-Y-CONTROLES.pdf

ammandiitaa13

Programación multimedia y dispositivos móviles

1º Desarrollo de Aplicaciones Multiplataforma

Estudios España

Accede al documento original

INTERFACES. LAYOUTS Y CONTROLES

1. Introducción a los layouts

Objetos contenedores para elementos de las vistas de las interfaz de usuario. No visuales, controlan la distribución, posición y dimensiones de los elementos de su interior. Pueden contener otros. Gran variedad, especializados en diferentes distribuciones de los elementos.

- LinearLayout
- FrameLayout
- AbsoluteLayout
- RelativeLayout
- TableLayout
- GridLayout
- ConstraintLayout

2. Clase View y ViewGroup

Todos los elementos que se pueden distribuir en los layouts heredan de la clase View. Además se pueden agrupar en un grupo (ViewGroup). Un elemento tipo ViewGroup es RecyclerView, permite construir listados de objetos complejos. Los objetos View van a tener una serie de atributos comunes para el dimensionamiento y posicionamiento en el layout. A cada uno de estos atributos se les pueden asignar medidas en diferentes unidades:

- Pulgadas (in)
- Milímetros (mm)
- Puntos (pt)
- Píxeles (px)
- Píxeles independientes de la densidad (dp)
- Píxeles independientes de la escala (sp)

Los atributos que puede tomar un objeto View pueden ser:

- De posicionamiento: pueden tomar valores relativos con wrap_content (se ajusta el contenedor al objeto, con paddings) y match_parent (se toma el máximo tamaño que permite el contenedor).
  - layout_width. Ancho. Puede tomar valores absolutos o relativos.
  - layout_height. Alto. Puede tomar valores absolutos o relativos.
  - layout_gravity. Para centrar o justificar la vista.
  - layout_weight. Distribuir el espacio entre diferentes objetos según un peso.
- Para los márgenes:
  - layout_margin
  - layout_marginBottom, layout_marginTop, layout_marginLeft, layout_marginRight
- Para el espaciado:
  - padding. Espaciado igual en las 4 direcciones
  - paddingTop, paddingBottom.
  - paddingLeft, paddingRight.
- Otros:
  - background. Permite establecer un color de fondo por defecto.
  - Identificación de objetos. @id+/nombre -> en el fichero XML. Kotlin: var/val nombreElemento = findViewById<TipoObjeto>(R.id.nombre)

3. Tipos de layouts

- LinearLayout. sitúa los elementos hijos uno tras otro, de forma horizontal o vertical, dependiendo del valor indicado en el atributo orientation.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#ff8000"
android:orientation="vertical">
</LinearLayout>
```

Se puede distribuir varios elementos con distintos pesos con layout_weight.

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:background="#ff8800"
        android:layout_weight="1">
</LinearLayout>
```

- FrameLayout. coloca los elementos internos por defecto en la esquina superior izquierda. Se puede modificar la posición de cada uno con layout_gravity, con left, right, bottom y top. Permite modificar la visibilidad de los elementos internos con visibility.

```
<?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"
        android:id=”@+id/framelayout1”>
        <FrameLayout
                android:layout_width=”100dp”
                android:layout_height=”100dp”
                android:layout_gravity=”left|bottom”
                android:background=”#ff0000”>
        </FrameLayout>
```

- AbsoluteLayout: permite situar los elementos con coordenadas absolutas, con layout_x y layout_y. No es recomendable su uso y está obsoleto (deprecated).

```
<?xml version=”1.0” encoding=”utf-8”?>
<AbsoluteLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id=”@+id/Absolutelayout1”>
        <AbsoluteLayout
                android:layout_width=”100dp”
                android:layout_height=”100dp”
                android:layout_x=”200dp”
```

```
                android:layout_y=”300dp”
                android:background=”#ff0000”>
        </AbsoluteLayout>
```

- RelativeLayout: los elementos van colocados en una posición relativa de unos sobre otros o con referencia al contenedor padre. Existen opciones de posicionamiento diferentes si la referencia es otro elemento (control u otro) o es el layout padre.
    - Posicionado y alineado relativo a un control.
        - android: layout_above, android: layout_below,
        - android:layout_toLeftOf, android:layout_toRightOf.
        - android:layout_alignLeft, android:layout_alignRight
    - Posicionado y alineado relativo a layout padre:
        - android:layout_alignParentLeft,
        - android:layout_alignParentRight

```
<?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:id="@+id/LayoutPadre">
        <RelativeLayout
                android: id="@+id/hijo1"
                android:layout_width="100dp"
                android: layout_height="100dp"
                android:layout_alignParentTop = "true"
                android: layout_alignParentRight = "true"
                android:background="#ff0000">
        </RelativeLayout>
        <RelativeLayout
                android:layout_width="100dp"
                android:layout_height="100dp"
                android:layout_alignParentTop = "true"
                android:Layout_toLeftOf="@id/hijo1"
                android:background="#ff8000">
        </RelativeLayout>
```

- TableLayout: distribuye los elementos en forma de tabla, definiendo filas y columnas y la posición de cada elemento dentro de la tabla. Las filas se añaden con TableRow.

```
        <?xml version="1.0" encoding="utf-8"?>
        <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:background="#000000" >
                <TableRow>
                        <Button
                        android:id="@+id/button3"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="3dp"
                        android:text="BOTON 1" />
                        <Button
                        android:id="@+id/button5"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_margin="3dp"
                android:text="Button" />
        </TableRow>
        <!—MAS FILAS-->
    </TableLayout>
```

- GridLayout: similar al anterior. Se indica el nº de filas y columnas con rowCount y columnCount.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:rowCount=”3”
        android:columnCount=”3”
        android:orientation=”horizontal”
        android:id=”@+id/LayoutPadre”>
            <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="3dp"
            android:background=”#FF2600" />
```

- ConstraintLayout: para cambiar la forma de hacer las vistas. Evitar el uso de layouts anidados. Útil para vistas complejas con muchos elementos. Recomendable usar diseño gráfico en primer lugar. Se pueden realizar ajustes posteriores con el fichero XML. Contenedor muy flexible y heredero de otros: RelativeLayout. Crea un marco exterior de referencia sobre el que se pueden imponer una serie de restricciones para organizar los elementos. Los elementos se pueden anclar entre sí.

4. Introducción a los controles

Incluidos por defecto y accesibles. Clasificados en categorías. Se incluyen también por defecto algunos componentes de Google, de gestión de anuncios y de mapas. Existen muchas librerías de terceros que se pueden integrar en proyectos. La visualización de la interfaz gráfica en modo split permite realizar acciones como replicar elementos fácilmente y con un control más fino. Si se copian y pegan elementos en el XML inicialmente aparecen superpuestos en la parte gráfica. Conveniente indicar el id de cada control y layout.
Con Constraint Widget se pueden fijar los muelles e incluir zonas rígidas. Deben ir anclados unos a otros mediante enlaces, que pueden tener una parte de espaciado fijo y otra flexible (en forma de muelle). Algunos elementos irán anclados a los bordes del contenedor o layout. Habitual que haya una relación entre campos de texto y sus etiquetas descriptivas. Aquí se realiza un anclaje a nivel del texto, no de los controles.
Para realizar el anclaje de 2 elementos que tienen texto, hay que activar Show baseline y conectar los dos baselines

5. Tipos de controles

- TextView. mostrar texto en la pantalla. Parámetros más importantes:
    - android:text. Es el texto que se mostrará
    - android:background: El color de fondo

    - android:textColor: El color del texto
    - android:textSize: El tamaño del texto. Se recomienda usar unidades sp
    - android:textStyle: normal, bold o italic
    - android:typeface: Tipo de letra.

  Se puede definir en el XML de la vista y después ser manipulado por código.
- Button: crear botones. Hereda de TextView y tiene atributos y métodos adicionales. La gestión de la acción del botón se puede hacer de varias formas:
    - Definir en el XML de la vista y en el Button correspondiente un atributo android:onClick="nombreFuncion". El método debe definirse en código.
    - Asociarle un manejador de eventos desde código Kotlin y definiendo la acción correspondiente.
    - Hacer que la Activity donde se halla haga de manejador de eventos. Para ello la activity debe implementar la interfaz View.onClickListener (si se va a trabajar con el evento onClick).

      ```
      val texto = findViewById<TextView>(R.id.textoDAM)
      val boton1 = findViewById<Button>(R.id.button1)
      boton1.setOnClickListener(){
            Toast.makeText(this, "BANZAI!", Toast.LENGTH_LONG).show()
            texto.setText("Abogado?")
            texto.setTextSize(3, 28f)
            texto.setTextColor(Color.BLUE)
      }
      ```
- ToggleButton: botón con 3 estados visibles, pulsado y no pulsado. Se puede variar su estilo, incluido el texto, dependiendo de su estado.
- ImageButton: botón que muestra una imagen, pero no texto. Hereda de ImageView. La imagen debe estar incluida en res/drawable
- EditText: tipo más simple para la entrada de texto. Hereda de la clase TextView. Por defecto va a leer texto plano y se usará el teclado textual del móvil. Existen muchas opciones gracias a inputType:
    - textPassword. Para contraseña
    - Number. Entrada numérica
    - Date. Texto tratado como fecha
    - Time. Texto tratado como hora
    - textCapSentences. Pone en mayúscula la primera letra de cada frase.
    - textCapWords. Pone en mayúscula la primera letra de cada palabra.

  Existen más atributos que añaden más funcionalidad:
    - android:hint. Para poner un texto predefinido que aparecerá en gris.
    - android:maxLength. Para limitar el número de caracteres.
    - android:lines. Para limitar el número de líneas
    - android:digits. Para limitar el número de dígitos.
- AutoCompleteTextView: variante de EditText, da sugerencias de autocompletado.
- RadioButton: Control de selección excluyente para elegir una opción en un conjunto. De tipo circular. Para que sean excluyentes (sólo se puede seleccionar una opción) hay que incluir varios RadioButtons en un RadioGroup. En primer lugar hay que colocar un RadioGroup y después se pueden ir añadiendo RadioButtons, que se irán posicionando, por defecto, en modo vertical.
- Spinner. Muestra una lista desplegable para seleccionar un único elemento.
- Checkbox. Para marcar/desmarcar opciones en una aplicación
- Switch, SeekBar, RatingBar, ProgressBar