Desarrollo de Aplicaciones para Dispositivos Móviles

Ingeniero Walter Medina
walter.medina@correounivalle.edu.co

Componentes Layouts:

Los layouts son estructuras que definen la disposición y organización de los elementos de la interfaz en una aplicación. Los layouts determinan cómo se colocan y distribuyen los elementos, como botones, texto, imágenes, etc., en la pantalla del dispositivo.

Algunos tipos de Layouts:

- **LinearLayout**
- **RelativeLayout**
- **ConstraintLayout**
- **TableLayout**
- **GridLayout**

LinearLayout

Un LinearLayout es un tipo de vista contenedor que organiza sus elementos secundarios en una única dirección: horizontalmente o verticalmente. Es uno de los tipos más básicos de layouts disponibles en Android y se utiliza para colocar vistas (como botones, texto, imágenes, etc.) en una secuencia lineal.

```
10    <LinearLayout
11        android:layout_width="match_parent"
12        android:layout_height="match_parent"
13        tools:context=".MainActivity"
14        android:padding="16dp"
15        android:orientation="vertical">
16
17        <Button...>
23
24        <Button...>
30
31        <Button...>
37        <Button...>
43        <View...>
48
49        <LinearLayout...>
66
67        <LinearLayout...>
92
93        <LinearLayout...>
118
119   </LinearLayout>
```

RelativeLayout

Es un tipo de vista de contenedor que permite organizar los elementos de la interfaz en relación unos con otros o con el contenedor padre. Esto significa que puedes posicionar los elementos en función de otros elementos, como alinearlos a la derecha, a la izquierda, arriba o abajo de otro elemento, o centrar un elemento en relación con su padre.

```
<RelativeLayout
    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"
    tools:context=".RelativeLayout">

    <ImageView...>

    <View...>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:padding="5dp"
        android:text="EditText:"
        android:layout_below="@+id/vSuperior"
        android:layout_toLeftOf="@+id/etNombre"/>

    <EditText...>
    <Button...>
    <TextView...>
    <View...>

</RelativeLayout>
```

ConstraintLayout

Permite crear interfaces de usuario flexibles y dinámicas mediante la disposición de vistas en relación entre sí o con el contenedor padre. Mucho más potente y flexible que RelativeLayout o LinearLayout.

Ventajas de Usar ConstraintLayout

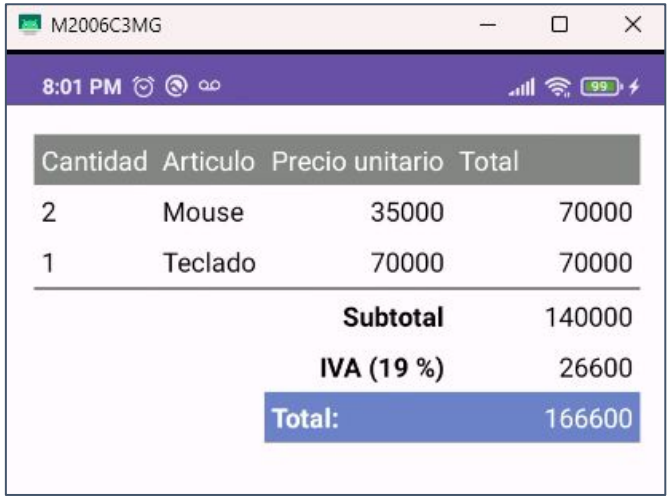
- **Rendimiento Mejorado**: Como puedes reducir la cantidad de layouts anidados, la jerarquía de vistas se simplifica, mejorando el rendimiento.
- **Flexibilidad en el Diseño**: Permite crear interfaces de usuario altamente personalizadas y complejas sin necesidad de múltiples *layouts* anidados.
- **Facilidad de Mantenimiento**: Una jerarquía de vistas más simple hace que el código sea más fácil de entender y mantener.
- **Responsividad**: Facilita la creación de diseños que se adaptan de manera eficiente a diferentes tamaños y orientaciones de pantalla.

```
<androidx.constraintlayout.widget.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"
    tools:context=".ConstraintLayout">
    <!-- Elementos sobre otros elementos-->
    <ImageView
        android:id="@+id/vSmall"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginTop="32dp"
        android:src="@drawable/ic_user"
        android:translationZ="1dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.3"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button...>

    <androidx.constraintlayout.widget.Guideline...>

    <View...>


</androidx.constraintlayout.widget.ConstraintLayout>
```

TableLayout

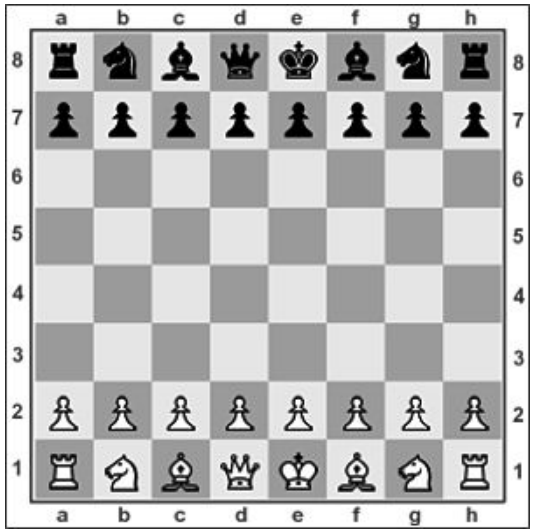
Es un tipo de vista que organiza sus hijos en una cuadrícula de filas y columnas, similar a una tabla HTML. Cada fila dentro de un TableLayout es una instancia de la clase TableRow, y dentro de estas filas se pueden agregar otras vistas, como botones, textos, imágenes, etc.

```xml
<TableLayout
    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:stretchColumns="3"
    android:padding="16dp"
    tools:context=".TableLayout">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/gris">

        <TextView
            android:layout_column="0"
            android:padding="4dp"
            android:text="Cantidad"
            android:textSize="18sp"
            android:textColor="@color/white"/>

        <TextView...>
        <TextView...>
    </TableRow>

    <TableRow...>

</TableLayout>
```



GridLayout

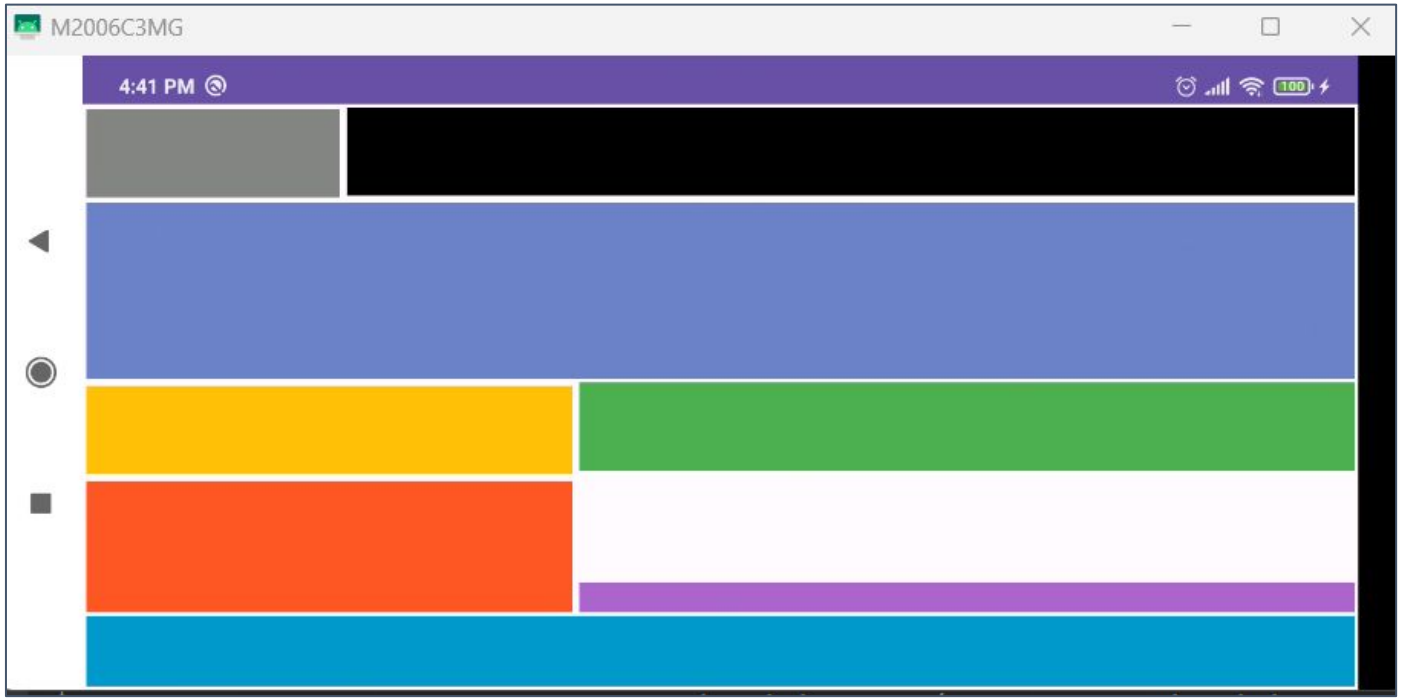
Organiza sus elementos hijos en una cuadrícula (grid) con filas y columnas. Este tipo de layout es útil cuando necesitas disponer elementos en una estructura de rejilla o tablero.

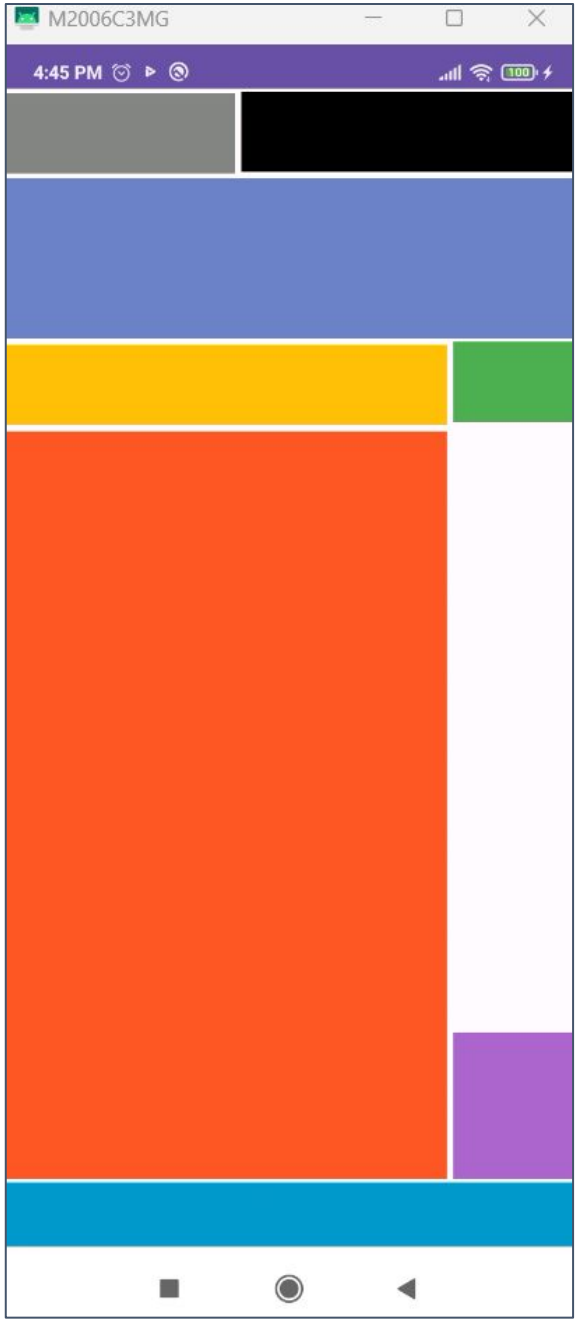



```xml
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:columnCount="4"
    android:rowCount="3"
    tools:context=".GridLayout">

    <View
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="@color/black"
        android:layout_column="0"
        android:layout_row="0" />

    <View...>
    <View...>

    <View...>

</GridLayout>
```

Ejercicio en clase

Realizar el siguiente diseño, únicamente usando ConstraintLayout, tenga en cuenta que al girar la pantalla del celular se debe adaptar los componentes.





GRACIAS !!

“El esfuerzo vence al talento cuando el talento no se está esforzando”.
Anónimo

Bibliografía:

- https://developer.android.com/docs?hl=es-419