| **SIMULACIONES**  **IV + IoT + DL**  **Industria 5.0**  **Copyright 2026 para:**  **Diego L. Aristizábal Ramírez**  **Profesor, Facultad de Ciencias, Departamento de Física**  **Universidad Nacional de Colombia**  **Medellín** | **LA GUI** | 8 |
| --- | --- | --- |

| Temas |
| --- |
| * INTRODUCCIÓN * MI DECIMA SEXTA APP: UNA PRIMERA GUI * AGREGANDO BOTONES A MI DECIMO SEXTA APP * MI DECIMA SEPTIMA APP: MI PRIMERA GUI COMPLEJA * MI DECIMA OCTAVA APP: MIS PROPIOS BOTONES * TALLER * REFERENCIAS |

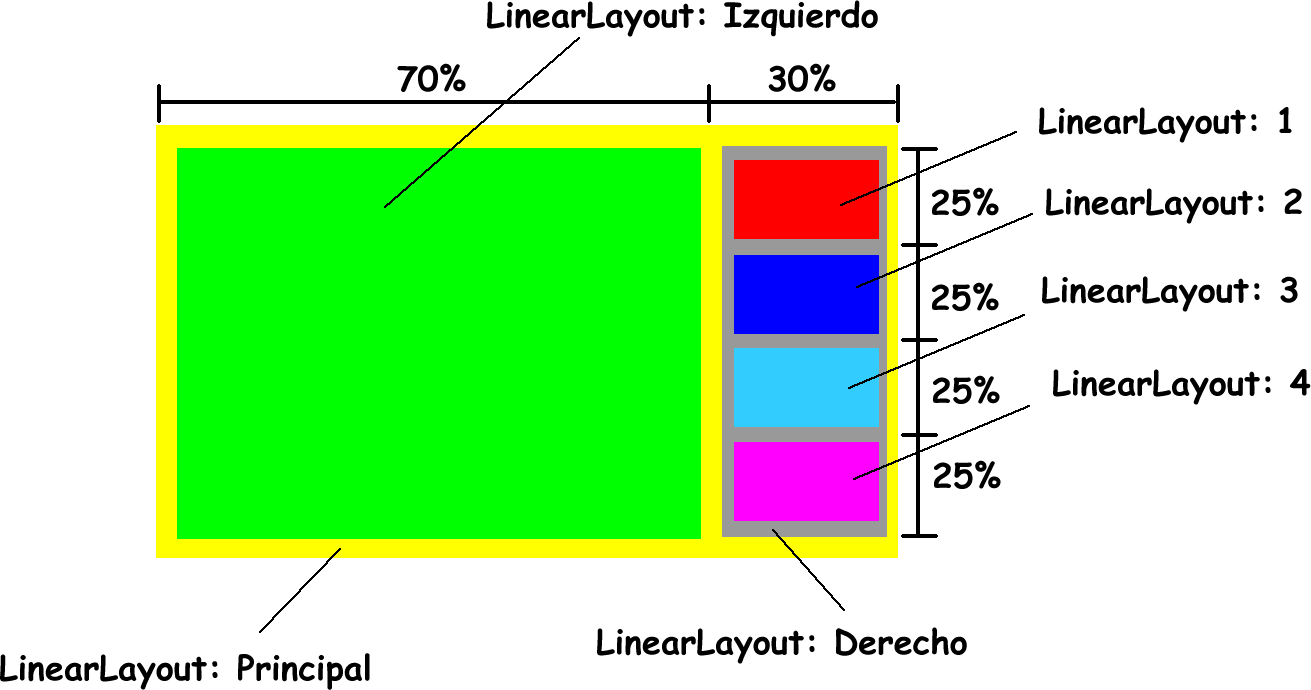
**INTRODUCCIÓN**

En este módulo se desarrollaran tres apps que ilustran conceptos básicos de cómo desarrollar GUIs (**GUI**: **G**raphical **U**ser **I**nterface) con buen aspecto e independientes de la resolución y tamaño de las pantallas de los dispositivos móviles. Ya en los módulos anteriores se elaboraron GUIs que eran muy simples. Pero es realmente en este módulo donde se aborda con detalle este tema.

Es importante señalar que las GUI se pueden diseñar usando código XML, o incluso apoyándose en el IDE Android Studio mediante la acción de “arrastrar” y “pegar” componentes GUI. **Sin embargo desde el inicio de este curso se advirtió que en la medida de lo posible todos los desarrollos se harán con código Java implementado por el programador**.

**MI DECIMA SEXTA APP: UNA PRIMERA GUI**

En esta aplicación en sus primeros pasos se desplegará la GUI de la Figura 1 en cuanto a los **contenedores**. Al final se agregarán botones a los cuatro contenedores de la derecha.



**Figura 1**

**¿Qué es un Layout?**

Desde la propia guía de desarrollador de Android Studio, se define un **Layout** como una estructura visual para una interfaz de usuario, es decir, aquello que hace de intermediario entre el terminal móvil y el usuario. A través de ellos se diseña la distribución de los componentes gráficos (botones, campos de texto y en general todas las vistas -View-) en la pantalla del dispositivo.  Un **Layout**es un **contenedor** de una o más vistas y controla su comportamiento y posición

 La siguiente lista describe los **Layout** más utilizados en Android:

**LinearLayout**: Dispone los elementos en una fila o en una columna.  
**TableLayout**: Distribuye los elementos de forma tabular.  
**RelativeLayout**: Dispone los elementos en relación a otro o al padre.  
**AbsoluteLayout**: Posiciona los elementos de forma absoluta.  
**FrameLayout**: Permite el cambio dinámico de los elementos que contiene.  
**ConstraintLayout**: Versión mejorada de **RelativeLayout**.

El autor de estas notas prefiere usar el administrador **LinearLayout** debido a su facilidad de uso y que adicionalmente le ha dado muy buenos resultados. Para el caso de la GUI de la Figura 1 se usarán 7 **LinearLayout**.

Una vez realizada las anteriores aclaraciones se procederá a desarrollar MiDecimaSextaApp, la cual sólo tendrá una actividad.

**Paso 1: Primera versión de la GUI**

Con **Android Studio** proceder a crear un proyecto denominado **MiDecimaSextaApp** (hacerlo tal cual se han venido desarrollando los proyectos en el curso). Crear ahora la clase **ActividadPrincipalMiDecimaSextaApp** la cual debe heredar de Activity, Figura 2. De los métodos correspondientes al ciclo de vida de una aplicación Android sólo se usarán dos: **onCreate( )** y **onDestroy( )**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

**Figura 2**

Agregar el siguiente código a la clase.

public class ActividadPrincipalMiDecimaSextaApp extends Activity {  
  
  
 public void onCreate(Bundle savedInstanceState){  
 super.onCreate(savedInstanceState);  
  
  
  
  
 }*//fin del método onCreate* protected void onDestroy() {  
  
 this.finish();  
  
 super.onDestroy();  
  
 }*//fin del método onDestroy*}

Se implementará un método privado para que se encargue del diseño de la GUI y que se denominará **crearGUI( )**. En este método se **devolverá (return) un objeto tipo LinearLayout** el cual se pegará como vista definitiva a la actividad. Este método tiene el siguiente código (observar que se tuvo que adicionar código a **onCreate()**. Por ejemplo, se tuvo que crear un **objeto tipo LayoutParams** que aporta la información de cómo pegar el **LinearLayout** a la **Activity**. El parámetro ***MATCH\_PARENT*** indica que el **LinearLayout** ocupará toda la dimensión respectiva de donde se pegará).

public class ActividadPrincipalMiDecimaSextaApp extends Activity {  
  
  
 public void onCreate(Bundle savedInstanceState){  
 super.onCreate(savedInstanceState);  
  
 */\*  
 Para informar cómo se debe pegar el administrador de  
 diseño LinearLayout obtenido con el método crearGui()  
 \*/* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
  
  
 }*//fin del método onCreate  
  
  
  
 /\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI(){  
  
 */\*  
 Creación del LinearLayout principal. Su color  
 es amarillo  
 \*/* LinearLayout linearPrincipal = new LinearLayout(this);  
 linearPrincipal.setBackgroundColor(Color.*YELLOW*);  
  
  
 return linearPrincipal;  
 }*//fin método crearGUI* protected void onDestroy() {  
  
 this.finish();  
  
 super.onDestroy();  
  
 }*//fin del método onDestroy*}

Declarar ahora la. actividad en el **archivo manifiesto**. Incluir el código que obliga a que la GUI de la app se despligue **horizontalmente**

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.curso\_simulaciones.midecimasextaapp">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MiDecimaSextaApp"  
 tools:targetApi="31" >  
  
 *<!--Declarar la actividad que lanzara la aplicación-->* <activity android:name=".ActividadPrincipalMiDecimaSextaApp"  
 android:screenOrientation="landscape"  
 android:exported="true"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
 </activity>  
  
 </application>  
  
</manifest>

Compilar y ejecutar. Se despliega la GUI de la Figura 3.



**Figura 3**

**Paso 2: segunda versión de la GUI**

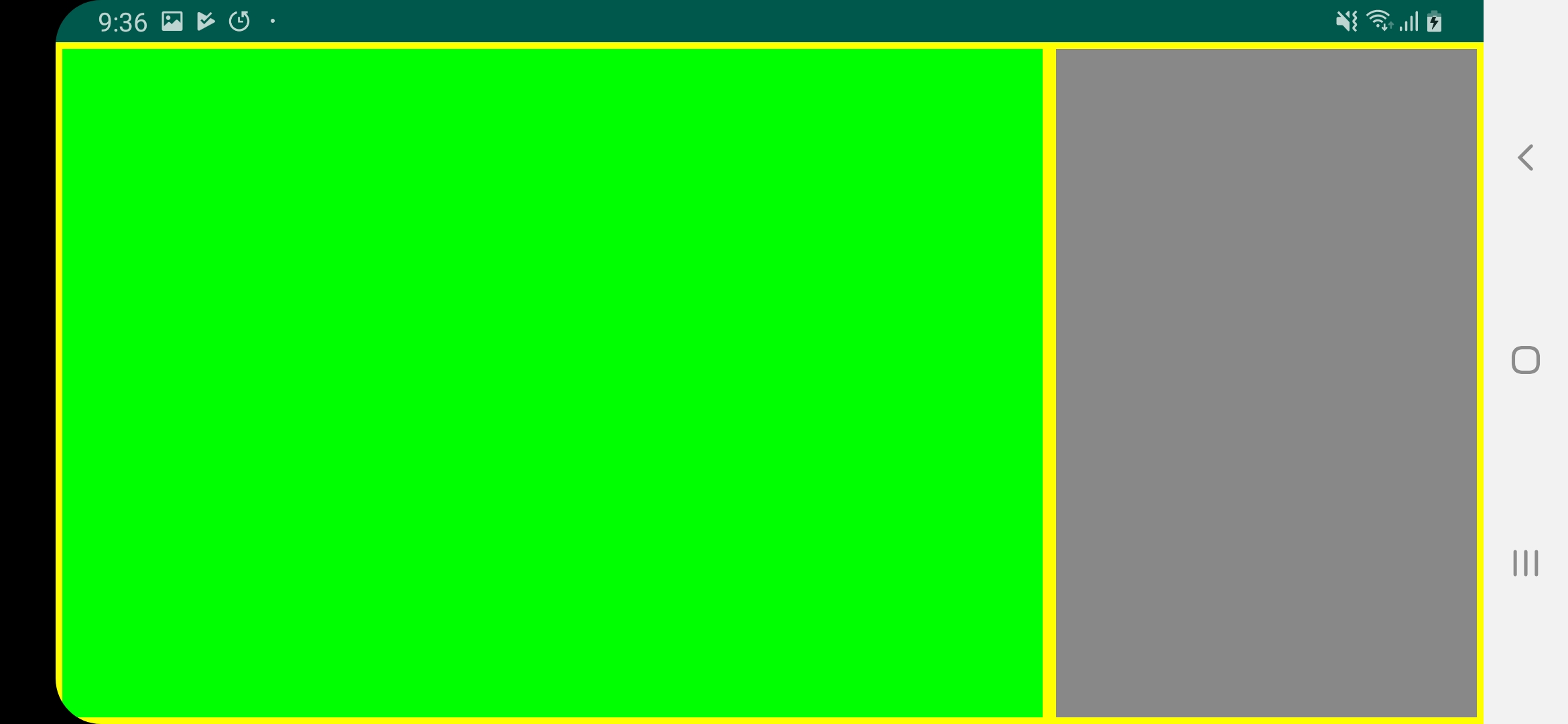
Se agregará código al método **crearGUI( )** que cree los **LinearLayout** **izquierdo** y **derecho** de la Figura 1. Observar que si se toma el **LinearLayout** principal como 100%, se necesita que el **LinearLayout** izquierdo ocupe el 70% y el derecho el 30%. Esto se logra asignando “pesos”. **Vamos despacio**.

Con el método **setWeightSum(float weightSum)** de **LinearLayout** se asigna un número que indica el total (100%) en “espacio” de este contenedor. Antes se debe declarar si los componentes se pegarán en dirección vertical u horizontal al **LinearLayout** principal. En esta app se pegan en dirección horizontal y la suma de los “pesos” se le asigna 10. Más adelante se pegara el componente **LinearLayout** izquierdo con un “peso” de 7 y el derecho con un “peso” de 3. Observar que la suma debe dar 10. Se podrían usar otros números pero deben guardar la proporción 100%, 70% y 30%.

Agregar el código al método **crearGUI( )** para que realice las tareas descritas en el párrafo anterior.

*/\*método responsable de administrar el diseño de la GUI\*/*private LinearLayout crearGUI() {  
  
 */\*  
 Creación del LinearLayout principal. Su color  
 es amarillo  
 \*/* LinearLayout linearPrincipal = new LinearLayout(this);  
 linearPrincipal.setBackgroundColor(Color.*YELLOW*);  
  
  
 *//los componentes se agregarán horizontalamente* linearPrincipal.setOrientation(LinearLayout.*HORIZONTAL*);  
 */\*  
 Definir la suma los pesos de los componentes que se pegarán a  
 LinearLayout principal  
 \*/* linearPrincipal.setWeightSum(10.0f);  
  
  
 */\*  
 Creación del LinearLayout izquierdo. Su color  
 es verde  
 \*/* LinearLayout linearIzquierdo = new LinearLayout(this);  
 linearIzquierdo.setBackgroundColor(Color.*GREEN*);  
  
 */\*  
 Creación del LinearLayout derecho. Su color  
 es gris  
 \*/* LinearLayout linearDerecho = new LinearLayout(this);  
 linearDerecho.setBackgroundColor(Color.*GRAY*);  
  
 */\*  
 Atributos con que se pega el LinearLayout  
 izquierdo a LinearLayout principal.  
 Como el pegado es horizontal el  
 CERO debe ir de primero  
 \*/* LinearLayout.LayoutParams parametros\_pegado\_linear\_izquierdo = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 *//peso con que se pega* parametros\_pegado\_linear\_izquierdo.weight = 7.0f;  
 *//se dejará un espacio de 10 pixeles en los bordes* parametros\_pegado\_linear\_izquierdo.setMargins(10, 10, 10, 10);  
  
  
 */\*  
 Atributos con que se pega el LinearLayout  
 derecho a LinearLayout principal.  
 Como el pegado es horizontal el  
 CERO debe ir de primero  
 \*/* LinearLayout.LayoutParams parametros\_pegado\_linear\_derecho = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_linear\_derecho.weight = 3.0f;  
 *//se dejará un espacio de 10 pixeles en los bordes* parametros\_pegado\_linear\_derecho.setMargins(10, 10, 10, 10);  
  
  
  
 */\*  
 Pegar ahora los dos LinearLayout (izquerdo  
 y el derecho al LinearLayout principal.  
 \*/* linearPrincipal.addView(linearIzquierdo, parametros\_pegado\_linear\_izquierdo);  
 linearPrincipal.addView(linearDerecho, parametros\_pegado\_linear\_derecho);  
  
  
 return linearPrincipal;  
}//fin método crearGUI()

Compilar y ejecutar. Se despliega la GUI de la Figura 4.



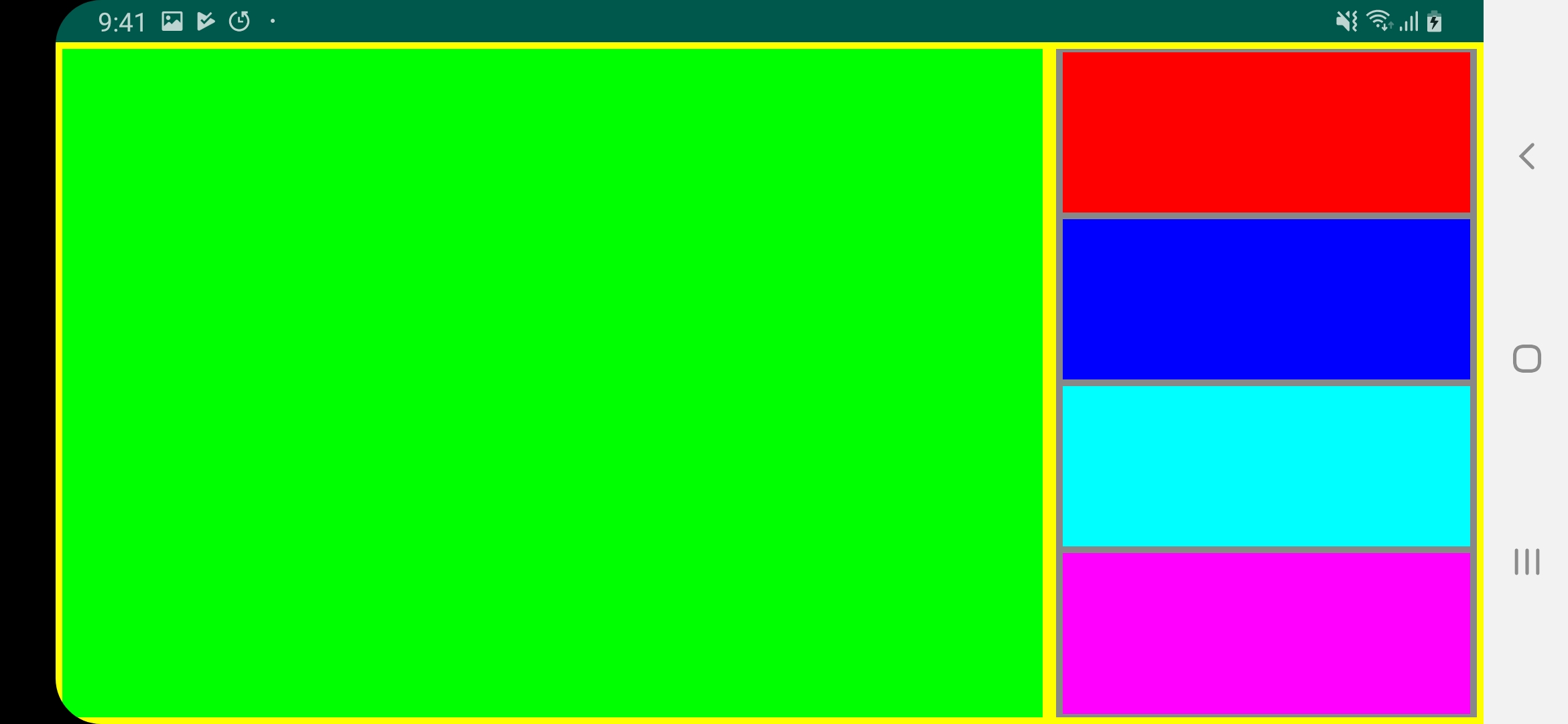
**Figura 4**

**Paso 3: tercera versión de la GUI**

A continuación se incluirá el código que permita agregar los **LinearLayaut** 1, 2, 3 y 4 de la Figura 1 con los pesos de 25% cada uno en el **LinearLayout** derecho.

*/\*método responsable de administrar el diseño de la GUI\*/*private LinearLayout crearGUI() {  
  
 */\*  
 Creación del LinearLayout principal. Su color  
 es amarillo  
 \*/* LinearLayout linearPrincipal = new LinearLayout(this);  
 linearPrincipal.setBackgroundColor(Color.*YELLOW*);  
  
  
 *//los componentes se agregarán horizontalamente* linearPrincipal.setOrientation(LinearLayout.*HORIZONTAL*);  
 */\*  
 Definir la suma los pesos de los componentes que se pegarán a  
 LinearLayout principal  
 \*/* linearPrincipal.setWeightSum(10.0f);  
  
  
 */\*  
 Creación del LinearLayout izquierdo. Su color  
 es verde  
 \*/* LinearLayout linearIzquierdo = new LinearLayout(this);  
 linearIzquierdo.setBackgroundColor(Color.*GREEN*);  
  
 */\*  
 Creación del LinearLayout derecho. Su color  
 es gris  
 \*/* LinearLayout linearDerecho = new LinearLayout(this);  
 linearDerecho.setBackgroundColor(Color.*GRAY*);  
  
 */\*  
 Atributos con que se pega el LinearLayout  
 izquierdo a LinearLayout principal.  
 Como el pegado es horizontal el  
 CERO debe ir de primero  
 \*/* LinearLayout.LayoutParams parametros\_pegado\_linear\_izquierdo = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 *//peso con que se pega* parametros\_pegado\_linear\_izquierdo.weight = 7.0f;  
 *//se dejará un espacio de 10 pixeles en los bordes* parametros\_pegado\_linear\_izquierdo.setMargins(10, 10, 10, 10);  
  
  
 */\*  
 Atributos con que se pega el LinearLayout  
 derecho a LinearLayout principal.  
 Como el pegado es horizontal el  
 CERO debe ir de primero  
 \*/* LinearLayout.LayoutParams parametros\_pegado\_linear\_derecho = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_linear\_derecho.weight = 3.0f;  
 *//se dejará un espacio de 10 pixeles en los bordes* parametros\_pegado\_linear\_derecho.setMargins(10, 10, 10, 10);  
  
  
  
 */\*  
 Pegar ahora los dos LinearLayout (izquerdo  
 y el derecho al LinearLayout principal.  
 \*/* linearPrincipal.addView(linearIzquierdo, parametros\_pegado\_linear\_izquierdo);  
 linearPrincipal.addView(linearDerecho, parametros\_pegado\_linear\_derecho);  
  
 */\*  
 Creación del LinearLayout uno. Su color  
 es rojo  
 \*/* LinearLayout linearUno = new LinearLayout(this);  
 linearUno.setBackgroundColor(Color.*RED*);  
  
 */\*  
 Creación del LinearLayout dos. Su color  
 es azul  
 \*/* LinearLayout linearDos = new LinearLayout(this);  
 linearDos.setBackgroundColor(Color.*BLUE*);  
  
 */\*  
 Creación del LinearLayout tres. Su color  
 es cyan  
 \*/* LinearLayout linearTres = new LinearLayout(this);  
 linearTres.setBackgroundColor(Color.*CYAN*);  
  
 */\*  
 Creación del LinearLayout cuatro. Su color  
 es magenta  
 \*/* LinearLayout linearCuatro = new LinearLayout(this);  
 linearCuatro.setBackgroundColor(Color.*MAGENTA*);  
  
  
 */\*  
 Para definir la suma de los pesos  
 de los componentes que se pegarán a  
 LinearLayout derecho  
 \*/* linearDerecho.setWeightSum(4.0f);  
 *//los componentes se agregarán verticalmente* linearDerecho.setOrientation(LinearLayout.*VERTICAL*);  
  
 */\*  
 Atributos con que se pegan los LinearLayout  
 uno, dos, tres y cuatro a LinearLayout dereccho.  
 En este caso Todos se pegan con los  
 mismos atributos. El CERO debe ir de segundo  
 \*/* LinearLayout.LayoutParams parametros\_pegado\_linear\_a\_derecho = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_pegado\_linear\_a\_derecho.weight = 1.0f;  
 *//se dejará un espacio de 10 pixeles en los costados  
 //y 5 pixeles arriba y abajo* parametros\_pegado\_linear\_a\_derecho.setMargins(10, 5, 10, 5);  
  
  
 */\*  
 Pegar ahora los cuatro LinearLayout (1,2, 3 y 4)  
 al LinearLayout derecho.  
 \*/* linearDerecho.addView(linearUno, parametros\_pegado\_linear\_a\_derecho);  
 linearDerecho.addView(linearDos, parametros\_pegado\_linear\_a\_derecho);  
 linearDerecho.addView(linearTres, parametros\_pegado\_linear\_a\_derecho);  
 linearDerecho.addView(linearCuatro, parametros\_pegado\_linear\_a\_derecho);  
  
  
 return linearPrincipal;  
}//fin método crearGUI()

Compilar y ejecutar. Se despliega la GUI de la Figura 5.



**Figura 5**

**AGREGANDO BOTONES A MI DECIMO SEXTA APP**

A continuación se agregarán los LinearLayout de la derecha botones (al de arriba 2 botones y a los otros tres de a un botón), Figura 6. A su vez cada botón activará un evento cuando se haga clic sobre él. El evento será cambiar el color del LinearLayout izquierdo.



Figura 6

**Nota importante**: Para agregar las etiquetas a los botones es necesario advertir que se presenta **un primer inconveniente con el tamaño y resolución de pantalla de los innumerables dispositivos móviles ANDROID y es en el tamaño de la letra (de la cadena que etiqueta los botones)**. En la literatura se encuentran diferentes indicaciones para manejar este problema. El autor de estas notas gestiona esta situación en un método que denominó **gestionarResolucion()**. En este método se calcula el tamaño de las letras con base a un porcentaje de la dimensión más pequeña entre el largo y el ancho de la pantalla del dispositivo móvil. Esta manera de proceder le ha dado muy buen resultado al autor garantizando que en todos los dispositivos móviles se despliegue en forma adecuada la GUI (para ser más cautos, digamos que en su mayoría). En otras palabras que la GUI vaya adquiriendo la propiedad de “**responsividad**”.

**Paso 1: Gestionar resolución de letras**

Agregar a **ActividadPrincipalMiDecimaSextaApp** la siguiente variable de instancia (hay que recordar que lo mejor es ubicar estas variables al comienzo de la clase),

**private int tamanoLetraResolucionIncluida**;

y el un método privado que se denominará **gestionarResolucion( )** con el siguiente código. Por orden, ubicarlo después del méton onCreate().

private void gestionarResolucion(){  
  
 *//independencia de la resolución de la pantalla* DisplayMetrics displayMetrics = this.getApplicationContext().getResources().getDisplayMetrics();  
 int alto = displayMetrics.heightPixels;  
 int ancho = displayMetrics.widthPixels;  
 int dimensionReferencia;  
  
 *//tomar el menor valor entre alto y ancho de pantalla* if (alto > ancho) {  
  
 dimensionReferencia = ancho;  
 } else {  
  
 dimensionReferencia = alto;  
 }  
  
  
 *//una estimación de un buen tamaño* int tamanoLetra = dimensionReferencia / 20;  
  
  
 *//tamano de letra para usar acomodado a la resolución de pantalla* tamanoLetraResolucionIncluida = (int) (tamanoLetra / displayMetrics.scaledDensity);

}

Adicionalmente en el método **onCreate()** agregar después de la línea de código

**super**.onCreate(savedInstanceState);

la línea de código que invoca el método anterior.

gestionarResolucion();

**Paso 2: Crear los botones**

Adicionar a la clase **ActividadPrincipalMiDecimaSextaApp** las siguientes variables de instancia (recordar unbicarlas al principio de la clase).

**private** Button **botonUno**, **botonDos**, **botonTres**, **botonCuatro**, **botonCinco**;

Agregar ahora un método privado que se denominará **crearElementosGui()** con el siguiente código. Por orden ubicarlo después del método **gestionarResolucion()**.

*/\*método responsable de la creación de los elementos de la GUI\*/*private void crearElementosGUI() {  
  
  
 */\*  
 1. El tamaño a usar de la letra tiene corrección de  
 resolución y tamaño de pantalla.  
\*/* botonUno = new Button(this);  
 botonUno.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonUno.setText("UNO");  
 botonUno.setBackgroundColor(Color.*rgb*(220, 156, 80));  
  
 botonDos = new Button(this);  
 botonDos.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonDos.setText("DOS");  
 botonDos.setBackgroundColor(Color.*rgb*(220, 156, 80));  
  
 botonTres = new Button(this);  
 botonTres.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonTres.setText("TRES");  
 botonTres.setBackgroundColor(Color.*rgb*(220, 156, 80));  
  
 botonCuatro = new Button(this);  
 botonCuatro.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonCuatro.setText("CUATRO");  
 botonCuatro.setBackgroundColor(Color.*rgb*(220, 156, 80));  
  
 botonCinco = new Button(this);  
 botonCinco.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonCinco.setText("CINCO");  
 botonCinco.setBackgroundColor(Color.*rgb*(220, 156, 80));

}*//fin método crearElementosGUI*

Adicionalmente en el método onCreate() agregar después de la línea de código

| gestionarResolucion(); |
| --- |

la línea de código que invoca el método anterior.

*//para crear elementos de la GUI*crearElementosGUI();

**Paso 3: Pegar los botones**

En el método **crearGUI()** agregar el siguiente código antes de la línea de código,

**return** linearPrincipal;

*//pegar los botones*linearUno.setWeightSum(2f);  
linearUno.setOrientation(LinearLayout.*HORIZONTAL*);  
linearDos.setWeightSum(1f);  
linearDos.setOrientation(LinearLayout.*HORIZONTAL*);  
linearTres.setWeightSum(1f);  
linearTres.setOrientation(LinearLayout.*HORIZONTAL*);  
linearCuatro.setWeightSum(1f);  
linearCuatro.setOrientation(LinearLayout.*HORIZONTAL*);  
  
LinearLayout.LayoutParams parametros\_pegado\_botones = new LinearLayout.LayoutParams(0,android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
parametros\_pegado\_botones.weight = 1.0f;  
parametros\_pegado\_botones.setMargins(10,5,10,5);  
linearUno.addView(botonUno, parametros\_pegado\_botones);  
linearUno.addView(botonDos, parametros\_pegado\_botones);  
linearDos.addView(botonTres, parametros\_pegado\_botones);  
linearTres.addView(botonCuatro, parametros\_pegado\_botones);  
linearCuatro.addView(botonCinco, parametros\_pegado\_botones);

Compilar y ejecutar. Se despliega la GUI de la Figura 7.



**Figura 7**

**Paso 4: Agregar los eventos a los botones.**

Hay que recordar que el objetivo de esta app ejemplo es que al hacer clic en los botones cambie el color del **LinearLayout** izquierdo (ejemplo de cómo disparan eventos los botones). Para esto es necesario que la variable que hace referencia a este objeto **sea de instancia**. Agregar la siguiente variable de instancia (hay que recordar que se ubicará al principio de la clase),

**private** LinearLayout **linearIzquierdo**;

**Para que no quede oculta esta variable**, es necesario cambiar en el método **crearGUI( )** la siguiente línea de código

LinearLayout linearIzquierdo = **new** LinearLayout(**this**);

por

linearIzquierdo = **new** LinearLayout(**this**);

A continuación se implementa el método privado **eventos()** con el siguiente código. Por orden agregarlo después del método **crearGUI()**.

*/\*Administra los eventos de la GUI\*/*private void eventos() {  
  
 *//evento del boton con etiqueta UNO* botonUno.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 linearIzquierdo.setBackgroundColor(Color.*BLACK*);  
 }  
 });  
  
 *//evento del boton con etiqueta DOS* botonDos.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 linearIzquierdo.setBackgroundColor(Color.*WHITE*);  
 }  
 });  
  
 *//evento del boton con etiqueta TRES* botonTres.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 linearIzquierdo.setBackgroundColor(Color.*RED*);  
 }  
 });  
  
 *//evento del boton con etiqueta CUATRO* botonCuatro.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 linearIzquierdo.setBackgroundColor(Color.*rgb*(200,200,50));  
 }  
 });  
  
 *//evento del boton con etiqueta CINCO* botonCinco.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 linearIzquierdo.setBackgroundColor(Color.*GREEN*);  
 }  
 });

}

Por último al final del método **onCreate()** agregar la línea de código que invoca el método anterior.

eventos();

**Compilar** y **ejecutar**. Hacer clic en los botones y observar el resultado.

**MI DECIMA SEPTIMA APP: MI PRIMERA GUI COMPLEJA**

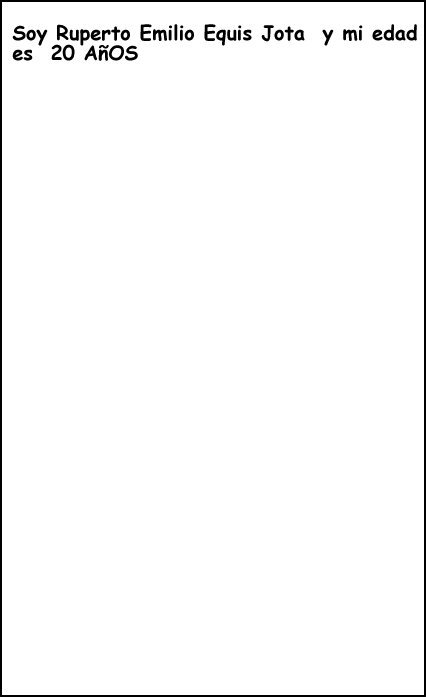
**Los requisitos y el diseño**

Esta aplicación la conforman cuatro actividades: la actividad principal (**ActividadPrincipalMiDecimaSeptimaApp**) y tres actividades secundarias. La actividad principal tiene una GUI como la de la Figura 8 izquierda, que se compone de 12 botones ubicados en dos columnas de a seis por columna. Sólo los botones UNO y DOS estarán inicialmente activados. Al botón UNO se le asignará un evento que lanzará una actividad secundaria (**ActividadSecundaria\_1**) la cual se compone de una GUI cuyo fondo es una imagen, Figura 8 centro. Al botón DOS se le asignará un evento que lanzará otra actividad secundaria (**ActividadSecundaria\_2**) la cual se compone de una GUI, Figura 8 derecha, que contiene dos columnas. La columna de la izquierda tiene cuatro etiquetas (**EditText**) y la de la derecha 4 campos de texto (**TextView**). En estos últimos se ingresaran cadenas (**String**). También tiene una imagen en la parte inferior. Al salir de esta actividad estos datos deberán quedar almacenados en una clase cuyas variables de almacenamiento son estáticas (la clase se denominará **AlmacenDatosRAM**) y deberá activarse el botón TRES de la aplicación principal.

|  |  |  |
| --- | --- | --- |

**Figura 8**

Al hacer clic en el botón 3 de la actividad principal (ya activado) lanzará una tercera actividad secundaria (**ActividadSecundaria\_3**) la cual desplegará en una etiqueta (**EditText**) un mensaje que contendrá la información que se ingresó en la **ActividadSecundaria\_2**, Figura 9.



**Figura 9**

Finalmente se invocará el método **finish()** cuando se salga de la actividad principal. Esto liberará los recursos. Se recomienda también al salir de la aplicación principal dejar las variables de **AlmacenDatosRAM** en su estado por defecto.

Por orden se organizará la aplicación en **paquetes**. Un paquete denominado **actividades\_secundarias** en donde estarán contenidas las tres actividades secundarias. Otro denominado **datos** que contendrá la clase **AlmacenDatosRAM**.

**La implementación**

**Paso 1: Crear las clases y los paquetes**

Con Android Studio crear el proyecto denominado **MiDecimaSeptimaApp**. Crear los paquetes y clases de la jerarquía de la Figura 10.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Figura 10**

**Paso 2: Agregar el esqueleto más básico de las actividades**

Como las cuatro actividades (**Activity**) presentes en esta aplicación tienen **GUI** comenzarán con el mismo esqueleto de código. A continuación se ilustra el código de la clase **ActividadPrincipalMiDecimaSeptimaApp**:

public class ActividadPrincipalMiDecimaSeptimaApp extends Activity {  
  
  
  
 public void onCreate(Bundle savedInstanceState){  
 super.onCreate(savedInstanceState);  
  
 *//para crear elementos de la GUI* crearElementosGUI();  
  
 *//para informar cómo se debe pegar el adminitrador de  
 //diseño obtenido con el método GUI* ViewGroup.LayoutParams parametro\_layout\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
  
  
 *//pegar el contenedor con la GUI* this.setContentView(crearGUI(), parametro\_layout\_principal);  
  
 *//para administrar los eventos* eventos();  
  
  
  
 } *//fin del método onCreate  
  
  
  
 /\*método responsable de la creación de los elementos de la GUI\*/* private void crearElementosGUI(){  
  
  
  
 }*//fin método crearElementosGUI  
  
  
 /\*método responsable de administrar el diseño de la GUI\*/* private LinearLayout crearGUI(){  
  
 LinearLayout linearPrincipal = new LinearLayout(this);  
  
  
 return linearPrincipal;  
  
 }*//fin método crearGUI  
  
  
 /\*Administra los eventos de la GUI\*/* private void eventos(){  
  
  
 }*//fin método eventos*}*//fin Actividad*

Agregar este mismo código a las otras tres actividades: **ActividadSecundaria\_1**, **ActividadSecundaria\_2** y **ActividadSecundaria\_3**.No olvidar la herencia de **Activity** (**extends Activity**)

**Paso 3: ActividadPrincipalMiDecimaSeptimaApp**

Se comenzará creando los elementos de la GUI. Para esto se declaran las variables de los objetos **Button** a usar (los puntos suspensivos indican que antes o después hay código ya escrito por lo que el lector que esté reproduciendo las notas sólo deberá agregar el código nuevo)

...

**public class** ActividadPrincipalMiDecimoSeptimaApp **extends** Activity {

**private** Button **botonUno**, **botonDos**, **botonTres**, **botonCuatro**,

**botonCinco**,**botonSeis**,**botonSiete**,**botonOcho**,

**botonNueve**,**botonDiez**,**botonOnce**,**botonDoce**;

**public void** onCreate(Bundle savedInstanceState){

...

}

...

*}//fin Actividad*

Se continúa con la creación de los objetos de tipo **Button** con las propiedades iniciales que se les asignarán (color, etiqueta,...). Primero se gestionará el asunto del tamaño de las fuentes a través de un método que se denominó **gestionarResolucion()**. Esto para el asunto de una **GUI RESPONSIVA**. Adicionalmente se observa remarcado en azul que el valor calculado del tamaño de letra se lleva a la clase **AlmacenDatosRAM**:esta clase hace el papel de una muy pequeña y sencilla “base de datos” pero en RAM (es decir, no es persistente la información contenida allí y por ende si se apaga el dispositivo móvil o se termina la ejecución de la aplicación se borran estos datos) que se puede acceder desde cualquier clase de la aplicación de forma muy fácil (es aquí donde radica su utilidad) debido a que las variables de almacenamiento de esta clase se declaran **estáticas**. Agregar el siguiente código a la clase **AlmacenDatosRAM**.

public class AlmacenDatosRAM {  
  
 public static int *tamanoLetraResolucionIncluida*;

public AlmacenDatosRAM() {  
  
 }  
  
}

y agregar el siguiente código al clase **ActividadPrincipalMiDecimaSeptimaApp**.

...

**private int tamanoLetraResolucionIncluida**;

**public void** onCreate(Bundle savedInstanceState){  
 **super**.onCreate(savedInstanceState);  
  
 gestionarResolucion();

...

private void gestionarResolucion() {  
  
 *//independencia de la resolución de la pantalla* DisplayMetrics displayMetrics = this.getApplicationContext().getResources().getDisplayMetrics();  
 int alto = displayMetrics.heightPixels;  
 int ancho = displayMetrics.widthPixels;  
 int dimensionReferencia;  
  
 *//tomar el menor valor entre alto y ancho de pantalla* if (alto > ancho) {  
  
 dimensionReferencia = ancho;  
 } else {  
  
 dimensionReferencia = alto;  
 }  
  
  
 *//una estimación de un buen tamaño* int tamanoLetra = dimensionReferencia / 20;  
  
  
 *//tamano de letra para usar acomodado a la resolución de pantalla* tamanoLetraResolucionIncluida = (int) (tamanoLetra / displayMetrics.scaledDensity);  
  
 *//guardar en el almacen de datos para que otras clases la accedan fácilmente* AlmacenDatosRAM.*tamanoLetraResolucionIncluida* = tamanoLetraResolucionIncluida;  
  
  
}*//fin método gestionarResolucion()*

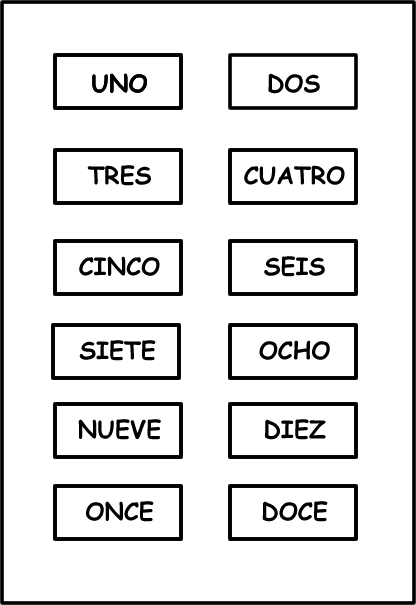
Ahora sí proceder a agregar el código del método **crearElementosGUI( )** para la creación de los botones.

*/\*método responsable de la creación de los elementos de la GUI\*/*private void crearElementosGUI() {  
  
  
 */\*  
 1. El tamaño a usar de la letra tiene corrección de  
 resolución y tamaño de pantalla.  
  
 2. Se usa un diseño de color de botón especial  
 PorterDuff.Mode.MULTIPLY. Podría usarse uno más sencillo  
 Como por ejemplo boton.setBackgroundColor(Color.RED)  
 \*/* botonUno = new Button(this);  
 botonUno.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*,tamanoLetraResolucionIncluida);  
 botonUno.setText("UNO");  
 botonUno.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
  
 botonDos = new Button(this);  
 botonDos.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonDos.setText("DOS");  
 botonDos.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
  
 botonTres = new Button(this);  
 botonTres.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*,tamanoLetraResolucionIncluida);  
 botonTres.setText("TRES");  
 botonTres.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonTres.setEnabled(false);  
  
 botonCuatro = new Button(this);  
 botonCuatro.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonCuatro.setText("CUATRO");  
 botonCuatro.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonCuatro.setEnabled(false);  
  
 botonCinco = new Button(this);  
 botonCinco.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonCinco.setText("CINCO");  
 botonCinco.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonCinco.setEnabled(false);  
  
 botonSeis = new Button(this);  
 botonSeis.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonSeis.setText("SEIS");  
 botonSeis.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonSeis.setEnabled(false);  
  
 botonSiete = new Button(this);  
 botonSiete.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonSiete.setText("SIETE");  
 botonSiete.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonSiete.setEnabled(false);  
  
 botonOcho = new Button(this);  
 botonOcho.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonOcho.setText("OCHO");  
 botonOcho.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonOcho.setEnabled(false);  
  
 botonNueve = new Button(this);  
 botonNueve.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonNueve.setText("NUEVE");  
 botonNueve.getBackground().setColorFilter(Color.*rgb*(220, 156,  
 80), PorterDuff.Mode.*MULTIPLY*);  
 botonNueve.setEnabled(false);  
  
 botonDiez = new Button(this);  
 botonDiez.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonDiez.setText("DIEZ");  
 botonDiez.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonDiez.setEnabled(false);  
  
 botonOnce = new Button(this);  
 botonOnce.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonOnce.setText("ONCE");  
 botonOnce.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonOnce.setEnabled(false);  
  
 botonDoce = new Button(this);  
 botonDoce.setTextSize(TypedValue.*COMPLEX\_UNIT\_SP*, tamanoLetraResolucionIncluida);  
 botonDoce.setText("DOCE");  
 botonDoce.getBackground().setColorFilter(Color.*rgb*(220, 156, 80), PorterDuff.Mode.*MULTIPLY*);  
 botonDoce.setEnabled(false);  
  
}*//fin método crearElementosGUI*

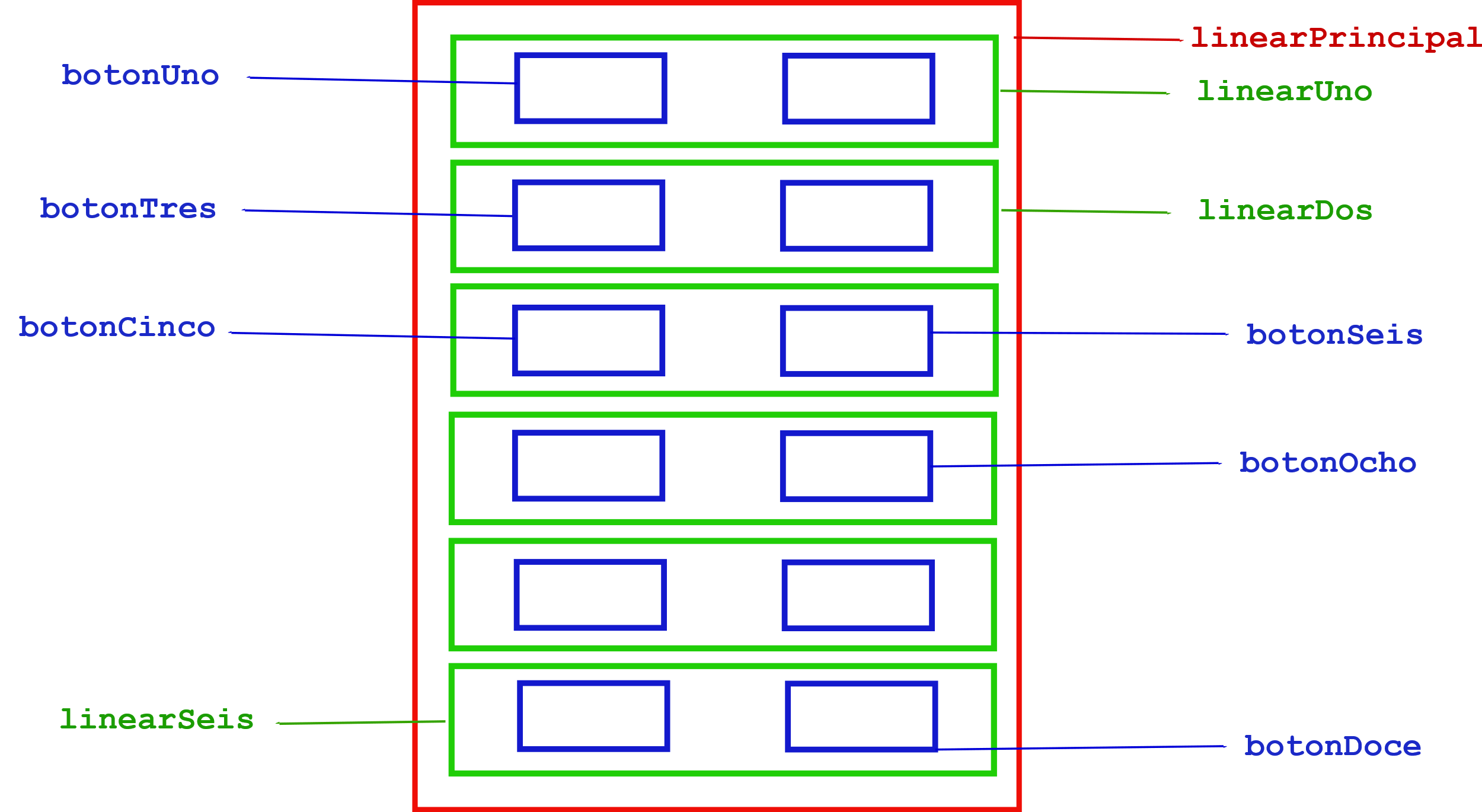
Proceder a agregar el código del método que administra el diseño y el pegado de los elementos de la GUI. Como se dijo en la anterior app, el autor de estas notas prefiere usar el administrador **LinearLayout** debido a su facilidad de uso y que adicionalmente le ha dado muy buenos resultados. Aquí se asigna **la orientación con la que se agregarán los componentes**, **la suma de “pesos”** y **otras propiedades**. En cuanto al asunto de “pesos”, el método **setWeightSum(float weightSum)** de **LinearLayout** asigna un número que indica el total (100%) en “espacio” de este contenedor.

**LinearLayout.LayoutParams** es la clase que administra la forma de pegar el componente en el contenedor. Por ejemplo el “peso” **weight** que ocupa dentro de éste. La suma de los pesos de los componentes que se agreguen al contenedor debe ser igual a **weightSum**. **LinearLayout.LayoutParams** también administra márgenes y otras propiedades.

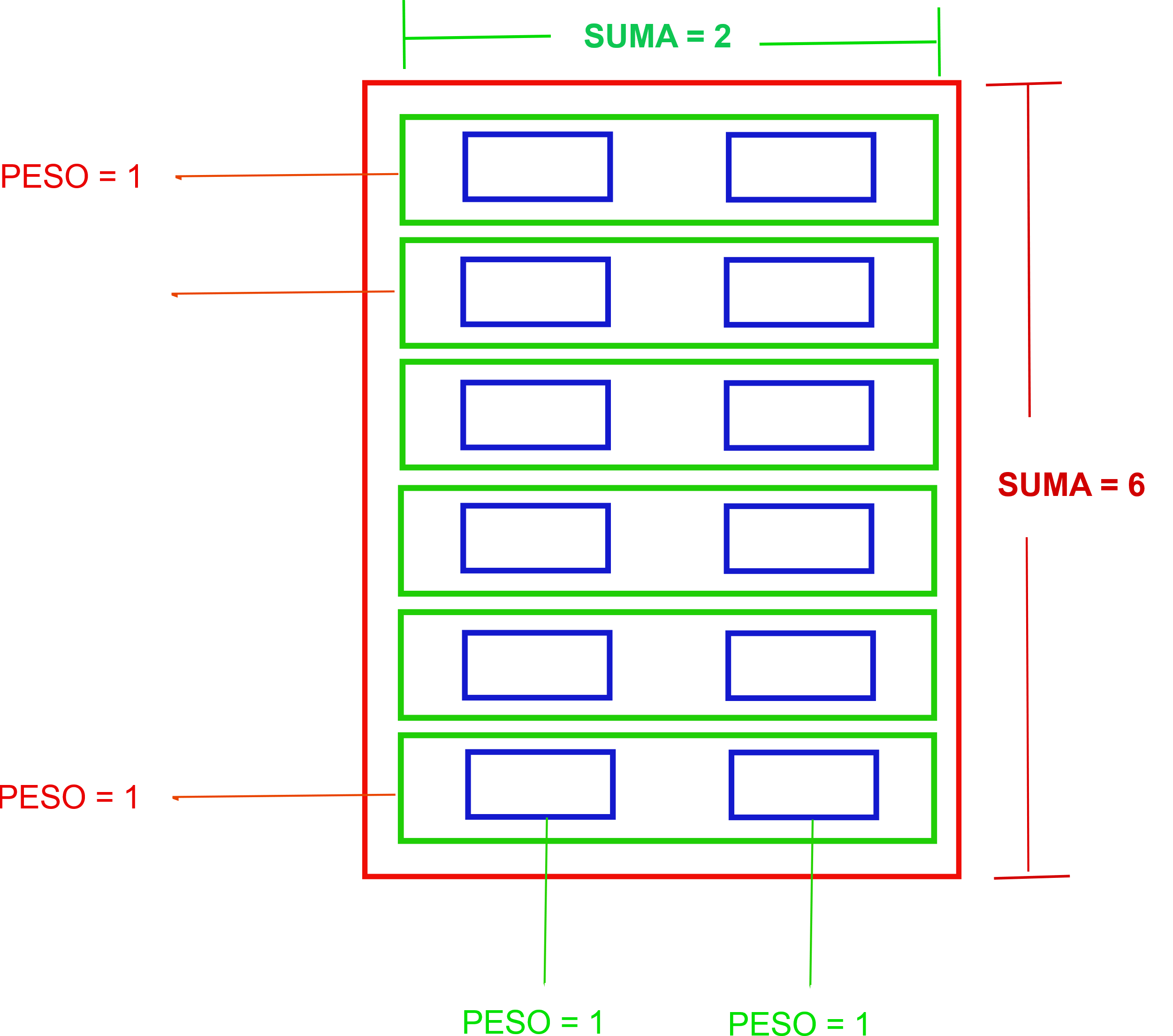
En las Figuras 11, 12 y 13 se ilustra la forma del diseño que se debe lograr para la GUI de la clase **ActividadPrincipalMiDecimaSeptimaApp**. En la Figura 11 el resultado que se debe obtener, en la Figura 12 los **LinaerLayout** que se usarán y la ubicación de los botones y en la Figura 13 los **weightSum** de los **LinearLayout** para agregar los componentes y los “pesos” con los que se agregan estos.



**Figura 11**



**Figura 12**



**Figura 13**

Proceder a completar el código del método **crearGUI( )**. En amarillo está remarcado el código que se debe agregar.

*/\*método responsable de administrar el diseño de la GUI\*/*private LinearLayout crearGUI() {  
  
 LinearLayout linearPrincipal = new LinearLayout(this);  
  
 *//los componentes se agregarán verticalmente* linearPrincipal.setOrientation(LinearLayout.*VERTICAL*);  
 *//para definir los pesos de las filas que se agregaran* linearPrincipal.setWeightSum(6.0f);  
  
 *//las seis filas* LinearLayout linearUno = new LinearLayout(this);  
 *//los componentes se agregarán horizontalmente* linearUno.setOrientation(LinearLayout.*HORIZONTAL*);  
 *//para definir el peso de los botones que se agregaran esta fila* linearUno.setWeightSum(2.0f);  
  
 LinearLayout linearDos = new LinearLayout(this);  
 *//los componentes se agregarán horizontalmente* linearDos.setOrientation(LinearLayout.*HORIZONTAL*);  
 *//para definir el peso de los botones que se agregaran esta fila* linearDos.setWeightSum(2.0f);  
  
 LinearLayout linearTres = new LinearLayout(this);  
 *//los componentes se agregarán horizontalmente* linearTres.setOrientation(LinearLayout.*HORIZONTAL*);  
 *//para definir el peso de los botones que se agregaran esta fila* linearTres.setWeightSum(2.0f);  
  
 LinearLayout linearCuatro = new LinearLayout(this);  
 *//los componentes se agregarán horizontalmente* linearCuatro.setOrientation(LinearLayout.*HORIZONTAL*);  
 *//para definir el peso de los botones que se agregaran esta fila* linearCuatro.setWeightSum(2.0f);  
  
 LinearLayout linearCinco = new LinearLayout(this);  
 *//los componentes se agregarán horizontalmente* linearCinco.setOrientation(LinearLayout.*HORIZONTAL*);  
 *//para definir el peso de los botones que se agregaran esta fila* linearCinco.setWeightSum(2.0f);  
  
 LinearLayout linearSeis = new LinearLayout(this);  
 *//los componentes se agregarán horizontalmente* linearSeis.setOrientation(LinearLayout.*HORIZONTAL*);  
 *//para definir el peso de los botones que se agregaran esta fila* linearSeis.setWeightSum(2.0f);  
  
  
 *//pegado de filas a linear principal  
 /\*  
 Pegado vetical (el ancho se acomoda en todo el contenedor  
 cada fila se pega con un peso de 1.0. Son seis filas,  
 la suma da 6.0.  
 Como el pegado es vertical el 0 debe ir como segundo argumento  
 \*/* LinearLayout.LayoutParams parametros\_pegado\_filas = new LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);  
 parametros\_pegado\_filas.weight = 1.0f;  
 linearPrincipal.addView(linearUno, parametros\_pegado\_filas);  
 linearPrincipal.addView(linearDos, parametros\_pegado\_filas);  
 linearPrincipal.addView(linearTres, parametros\_pegado\_filas);  
 linearPrincipal.addView(linearCuatro, parametros\_pegado\_filas);  
 linearPrincipal.addView(linearCinco, parametros\_pegado\_filas);  
 linearPrincipal.addView(linearSeis, parametros\_pegado\_filas);  
  
 *//pegado de botones a cada fila  
 /\*  
 Pegado horizontal (el alto se acomoda en todo el contenedor  
 se pegan dos botones por fila con un peso de 1.0,  
 la suma por fila da 2.0.  
 Como el pegado es vertical el 0 debe ir como primer argumento  
 \*/* LinearLayout.LayoutParams parametros\_pegado\_botones = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_botones.weight = 1.0f;  
 linearUno.addView(botonUno, parametros\_pegado\_botones);  
 linearUno.addView(botonDos, parametros\_pegado\_botones);  
 linearDos.addView(botonTres, parametros\_pegado\_botones);  
 linearDos.addView(botonCuatro, parametros\_pegado\_botones);  
 linearTres.addView(botonCinco, parametros\_pegado\_botones);  
 linearTres.addView(botonSeis, parametros\_pegado\_botones);  
 linearCuatro.addView(botonSiete, parametros\_pegado\_botones);  
 linearCuatro.addView(botonOcho, parametros\_pegado\_botones);  
 linearCinco.addView(botonNueve, parametros\_pegado\_botones);  
 linearCinco.addView(botonDiez, parametros\_pegado\_botones);  
 linearSeis.addView(botonOnce, parametros\_pegado\_botones);  
 linearSeis.addView(botonDoce, parametros\_pegado\_botones);  
  
  
  
 return linearPrincipal;  
  
}*//fin método crearGUI*

En el archivo manifiesto hay que declarar la actividad (**Activity**) que lanzará la aplicación y también el resto de actividades (**todas las actividades**). Se quiere que la GUI de la **ActividadPrincipalMiDecimoSextaApp** se mantenga **vertical** (respecto al dispositivo móvil) aunque se rote el dispositivo. Para esto se agregar el siguiente código en el archivo manifiesto (marcado en amarillo). Se quiere también que la GUI ocupe la **pantalla completa**, es decir, que no aparezca el título en una barra superior. Para esto debe agregarse en el archivo manifiesto la instrucción **Theme.NoTitleBar.Fullscreen** en la declaración de la actividad respectiva (código marcado en azul).

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.curso\_simulaciones.midecimaseptimaapp">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MiDecimaSeptimaApp"  
 tools:targetApi="31" >  
  
 *<!--Declarar la actividad que lanzara la aplicación-->* <activity android:name=".ActividadPrincipalMiDecimaSeptimaApp"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
 </activity>  
  
  
  
 *<!--declarando otras actividades-->* <activity android:name=".actividades\_secundarias.ActividadSecundaria\_1"></activity>  
  
 <activity android:name=".actividades\_secundarias.ActividadSecundaria\_2"></activity>  
  
 <activity android:name=".actividades\_secundarias.ActividadSecundaria\_3"></activity>  
  
 </application>  
  
  
</manifest>

Según las especificaciones de la aplicación los botones con las etiquetas UNO, DOS y TRES tienen eventos asociados. A los demás botones no se les asignará eventos en esta aplicación (solo están allí por razones del ejercicio de aprendizaje que se está llevando). Los eventos de esos tres botones se encargarán de lanzar las aplicaciones secundarias. Todo esto se agrega en el método al **eventos()** de la **ActividadPrincipalMiDecimaSeptimaApp** al cual se responsabilizó en esta actividad para gestionar los eventos.

*...*

*/\*Administra los eventos de la GUI\*/*private void eventos() {  
  
 *//evento del boton con etiqueta UNO* botonUno.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadSecundaria\_1();  
 }  
 });  
  
  
 *//evento del boton con etiqueta UNO* botonDos.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadSecundaria\_2();  
 }  
 });  
  
 *//evento del boton con etiqueta UNO* botonTres.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadSecundaria\_3();  
 }  
 });  
  
  
}*//fin método eventos  
  
//métodos que lanzar las actividades secundarias*private void lanzarActividadSecundaria\_1(){  
  
  
 Intent intent = new Intent(this, ActividadSecundaria\_1.class);  
 startActivity(intent);  
  
}  
  
  
private void lanzarActividadSecundaria\_2(){  
  
  
 Intent intent = new Intent(this, ActividadSecundaria\_2.class);  
 startActivity(intent);  
  
}  
  
private void lanzarActividadSecundaria\_3(){  
  
  
 Intent intent = new Intent(this, ActividadSecundaria\_3.class);  
 startActivity(intent);  
}

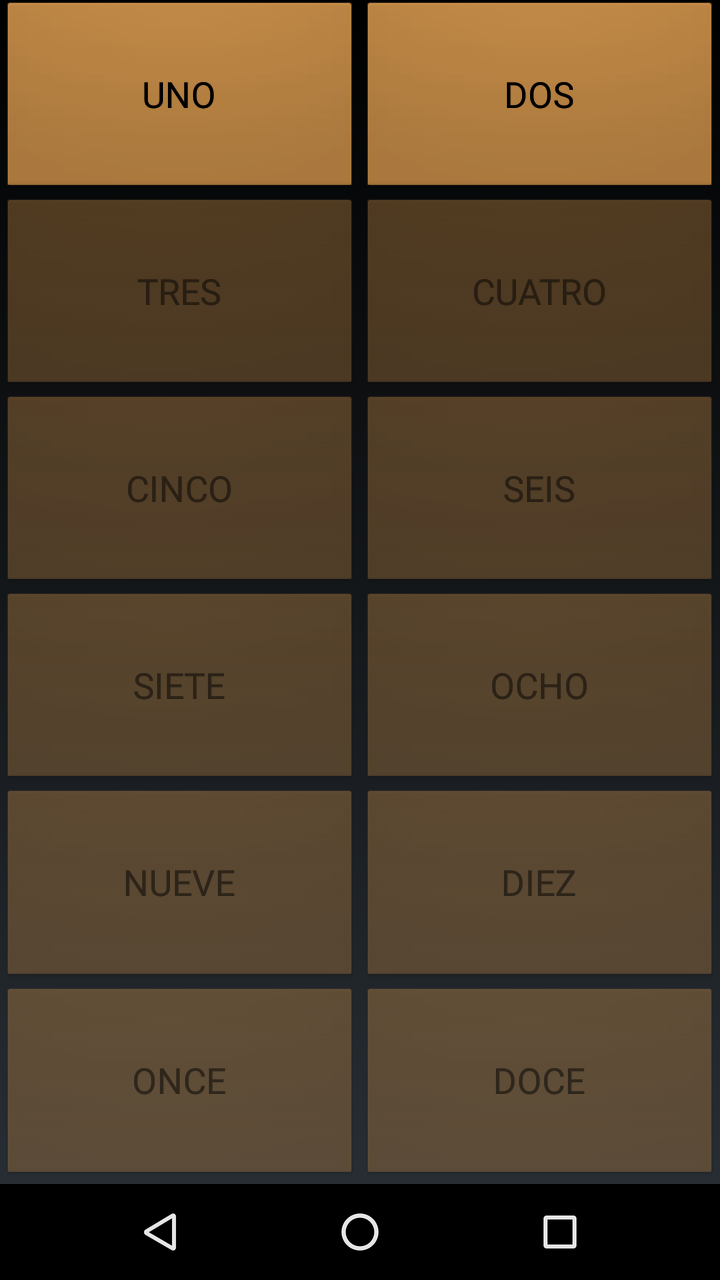
...

}*//fin Actividad*

Por último implementar el método **onDestroy()** en la **ActividadPrincipalMiDecimaSptimaApp**, para cuando se salga de ésta aplicación éste proceda a liberar los recursos usados y a finalizar correctamente la aplicación. Más adelante también se le agregará la responsabilidad de dejar los datos de la clase **AlmacenDatosRAM** con sus valores por defecto. Recordar que este método es automático debido a que pertenece al denominado ciclo de vida de la actividad (tema tratado en el módulo 7).

*/\* Métodos automático\*/*protected void onDestroy() {  
 this.finish();  
 super.onDestroy();  
}

Proceder a compilar y ejecutar la aplicación para saber cómo está quedando. Debería desplegarse la GUI de la Figura 14



**Figura 14**

**Paso 4: Agregar el código a la ActividadSecundaria\_1**

No se quiere que rote la GUI cuando rote el dispositivo móvil. También se quiere también que la GUI de esta actividad ocupe la pantalla completa, es decir, que no aparezca el título en una barra superior. Para esto debe agregarse en el archivo manifiesto el código marcado en amarillo.

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.curso\_simulaciones.midecimaseptimaapp">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MiDecimaSeptimaApp"  
 tools:targetApi="31" >  
  
 *<!--Declarar la actividad que lanzara la aplicación-->* <activity android:name=".ActividadPrincipalMiDecimaSeptimaApp"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
 </activity>  
  
  
  
 *<!--declarando otras actividades-->* <activity android:name=".actividades\_secundarias.ActividadSecundaria\_1"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait">  
 </activity>  
  
 <activity android:name=".actividades\_secundarias.ActividadSecundaria\_2"></activity>  
  
 <activity android:name=".actividades\_secundarias.ActividadSecundaria\_3"></activity>  
  
 </application>  
  
  
</manifest>

Como se va a usar una imagen (**gatico.png**)[[1]](#footnote-0) ésta deberá estar depositada en el directorio de la aplicación en la carpeta **app>src>main>res>drawable>**. Proceder ahora a agregar el código al método **crearGUI**() correspondiente a la administración del diseño a la actividad **ActividadSecundaria\_1**.

...

*/\*método responsable de administrar el diseño de la GUI\*/*

**private** LinearLayout crearGUI(){

...

linearPrincipal.setOrientation(LinearLayout.***VERTICAL***);  
linearPrincipal.setGravity(Gravity.***CENTER\_HORIZONTAL***);  
linearPrincipal.setGravity(Gravity.***FILL***);  
linearPrincipal.setBackgroundColor(Color.*rgb*(255, 252, 131));  
  
*//fondo*Drawable fondo = getResources().getDrawable(R.drawable.***gatico***);  
linearPrincipal.setBackgroundDrawable(fondo);

**return** linearPrincipal;  
}*//fin método crearGUI*

...

Proceder a compilar y ejecutar la aplicación para saber cómo está quedando. Debería desplegarse la GUI de la Figura 15 izquierda y luego de hacer clic en el botón UNO se despliega la GUI de la Figura 15 derecha.

|  |  |
| --- | --- |

**Figura 15**

**Paso 5: Agregar el código a la clase AlmacenDatosRAM**

Proceder a agregar el código remarcado con amarillo a la clase **AlmacenDatosRAM**. Las variables agregadas serán alimentadas por la clase **ActividadSecundarias\_2**.

public class AlmacenDatosRAM {  
  
 public static int *tamanoLetraResolucionIncluida*;  
  
 public static String *nombres*="Xxxx Xxxx";  
 public static String *apellidoUno*="Yyyy";  
 public static String *apellidoDos*="Zzzz";  
 public static int *edad*=0 ;  
 public static boolean *habilitar\_boton\_tres*=false;  
  
  
 public AlmacenDatosRAM() {  
  
 }  
  
}

**Paso 6: Agregar el código a la ActividadSecundaria\_2**

Se quiere también que la GUI de esta actividad ocupe la pantalla completa, es decir, que no aparezca el título en una barra superior. Tampoco se quiere dejar rotar la GUI cuando rote la pantalla. Para esto agregar en el **archvo manifiesto** el siguiente código.

...

<**activity  
 android:name=".actividades\_secundarias.ActividadSecundaria\_2"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait"**>  
  
</**activity**>

...

Como también se usará texto (en campos de texto y otros) se implementará un método que gestione los tamaños de las fuentes para garantizar que su aspecto se verá bien en los innumerables dispositivos móviles (resoluciones y tamaños de pantalla diferentes). Para esto agregar el siguiente código a la actividad **ActividadSecundaria\_2.**

**...**

**public class** ActividadSecundaria\_2 **extends** Activity {

**private int tamanoLetraResolucionIncluida**;  
**private int margenesResolucionIncluida**;

**public void** onCreate(Bundle savedInstanceState){  
 **super**.onCreate(savedInstanceState);  
  
 gestionarResolucion();

...

} *//fin del método onCreate*

...

**private void** gestionarResolucion(){  
  
  
tamanoLetraResolucionIncluida = (int)(0.8f\*AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
margenesResolucionIncluida = (int)(1.2f\*AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);

}*//fin método gestionarResolucion()*

...

}*//fin ActividadSecundaria\_2*

Agregar ahora a la **ActividadSecundaria\_2** el código que crea los componentes que conformarán la GUI. Como se va a usar una imagen (**identidad.png**)[[2]](#footnote-1) ésta deberá estar depositada en el directorio de la aplicación en la carpeta **app>src>main>res>drawable>**.

...

**public class** ActividadSecundaria\_2 **extends** Activity {

...

**private** TextView **textNombre**, **textPrimerApellido**,**textSegundoApellido**,  
 **textEdad**;  
  
**private** EditText **editNombres**, **editPrimerApellido**, **editSegundoApellido**, **editEdad**;  
  
**private** ImageView **imagen**;

...

*/\*método responsable de la creación de los elementos de la GUI\*/***private void** crearElementosGUI(){  
  
   
**textNombre** = **new** TextView(**this**);  
**textNombre**.setGravity(Gravity.***FILL\_VERTICAL***);  
**textNombre**.setBackgroundColor(Color.***GREEN***);  
**textNombre**.setTextSize(**tamanoLetraResolucionIncluida**);  
**textNombre**.setPadding(**margenesResolucionIncluida**, 0, 0, 0);  
**textNombre**.setText(**"NOMBRES"**);  
**textNombre**.setTextColor(Color.***BLACK***);  
  
**textPrimerApellido** = **new** TextView(**this**);  
**textPrimerApellido**.setGravity(Gravity.***FILL\_VERTICAL***);  
**textPrimerApellido**.setBackgroundColor(Color.***YELLOW***);  
**textPrimerApellido**.setTextSize(**tamanoLetraResolucionIncluida**);  
**textPrimerApellido**.setPadding(**margenesResolucionIncluida**, 0, 0, 0);  
**textPrimerApellido**.setText(**"PRIMER APELLIDO"**);  
**textPrimerApellido**.setTextColor(Color.***BLACK***);  
  
**textSegundoApellido** = **new** TextView(**this**);  
**textSegundoApellido**.setGravity(Gravity.***FILL\_VERTICAL***);  
**textSegundoApellido**.setBackgroundColor(Color.***GREEN***);  
**textSegundoApellido**.setTextSize(**tamanoLetraResolucionIncluida**);  
**textSegundoApellido**.setPadding(**margenesResolucionIncluida**, 0, 0, 0);  
**textSegundoApellido**.setText(**"SEGUNDO APELLIDO"**);  
**textSegundoApellido**.setTextColor(Color.***BLACK***);  
  
**textEdad** = **new** TextView(**this**);  
**textEdad**.setGravity(Gravity.***FILL\_VERTICAL***);  
**textEdad**.setBackgroundColor(Color.***YELLOW***);  
**textEdad**.setTextSize(**tamanoLetraResolucionIncluida**);  
**textEdad**.setPadding(**margenesResolucionIncluida**, 0, 0, 0);  
**textEdad**.setText(**"EDAD"**);  
**textEdad**.setTextColor(Color.***BLACK***);  
  
  
**editNombres** = **new** EditText(**this**);  
**editNombres**.setTextSize(**tamanoLetraResolucionIncluida**);  
**editNombres**.setText(AlmacenDatosRAM.*nombres*);  
  
**editPrimerApellido** = **new** EditText(**this**);  
**editPrimerApellido**.setTextSize(**tamanoLetraResolucionIncluida**);  
**editPrimerApellido**.setText(AlmacenDatosRAM.*apellidoUno*);  
  
  
**editSegundoApellido** = **new** EditText(**this**);  
**editSegundoApellido**.setTextSize(**tamanoLetraResolucionIncluida**);  
**editSegundoApellido**.setText(AlmacenDatosRAM.*apellidoDos*);  
  
  
**editEdad** = **new** EditText(**this**);  
*//despliega solo el teclado para entrar números enteros positivos***editEdad**.setKeyListener(DigitsKeyListener.*getInstance*(**false**, **false**));  
**editEdad**.setTextSize(**tamanoLetraResolucionIncluida**);  
**editEdad**.setText(**""** + AlmacenDatosRAM.*edad*);  
  
  
**imagen**=**new** ImageView(**this**);  
**imagen**.setImageResource(R.drawable.***identidad***);

}*//fin método crearElementosGUI*

...

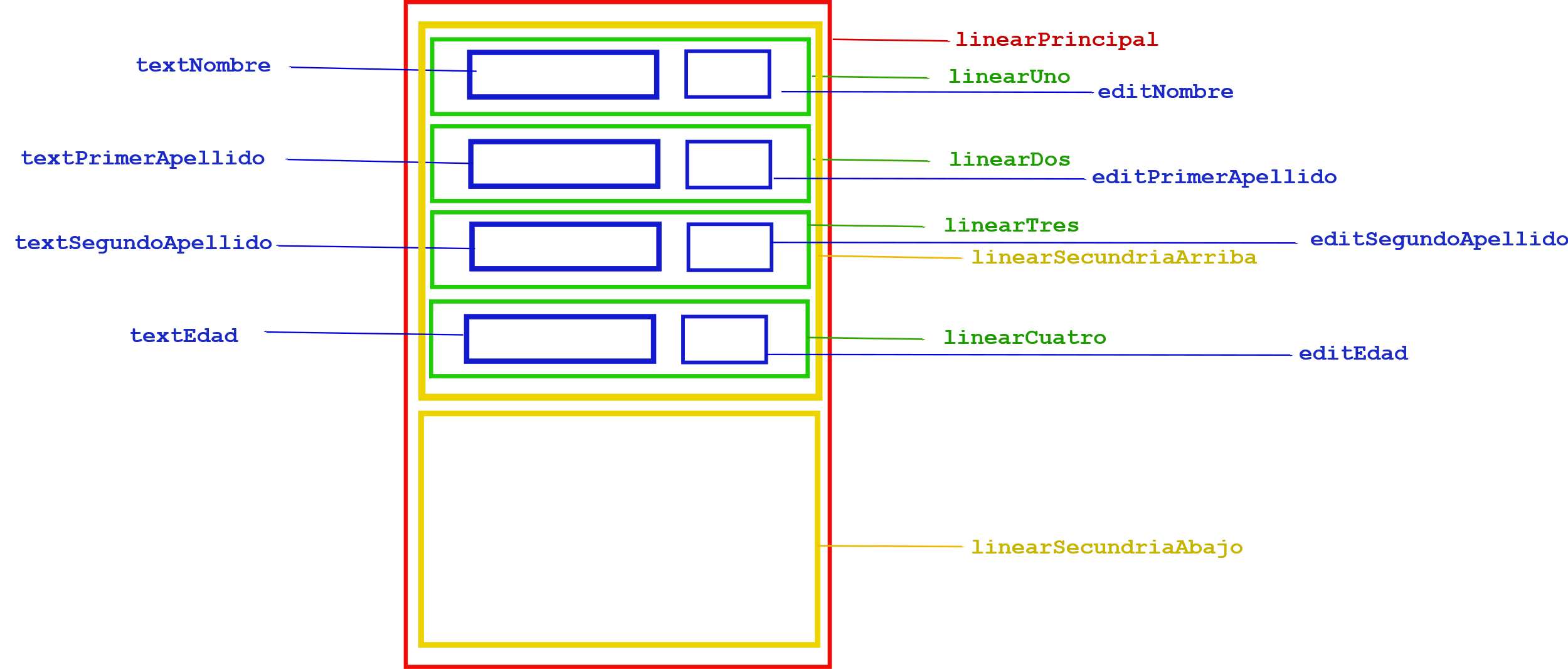
}*//fin ActividadSecundariaDos*

Ahora se procederá a agregar el código responsable de administrar el diseño de la GUI de la actividad **ActividadSecundaria\_2**. Se hará muy paulatinamente para comprender mejor este código. Este código irá encapsulado en el método **LinearLayout crearGUI()**. Primero se crearán dos **LinearLayout** secundarios, uno para pegar en la parte superior al **linearPrincipal** con un “peso”, **weight**, igual a 6.0 y el otro para pegarlo en la parte inferior con un peso igual a 4.0. Si se tiene en cuenta que al **linearPrincipal** se le asignó que la suma total de “pesos” es igual 10.0, **setWeightSum(10.0f)**, significa entonces que como éste ocupa el 100% de la pantalla, estos **LinearLayout** secundarios ocuparán el 60% y el 40 %. A su vez como estos dos **LinearLayout** contendrán componentes también se les asigna **setWeightSum**, al uno 4.0 y al otro 1.0. Esto es porque el de arriba contendrá 4 **LinearLayout** que se desempeñan como filas, cada una de las cuales se agrega con un peso de 1.0 y el de abajo contendrá una imagen que se pegará a éste con un peso de 1.0. El procedimiento definido en este párrafo se resume en las Figuras 16, 17 y 18.

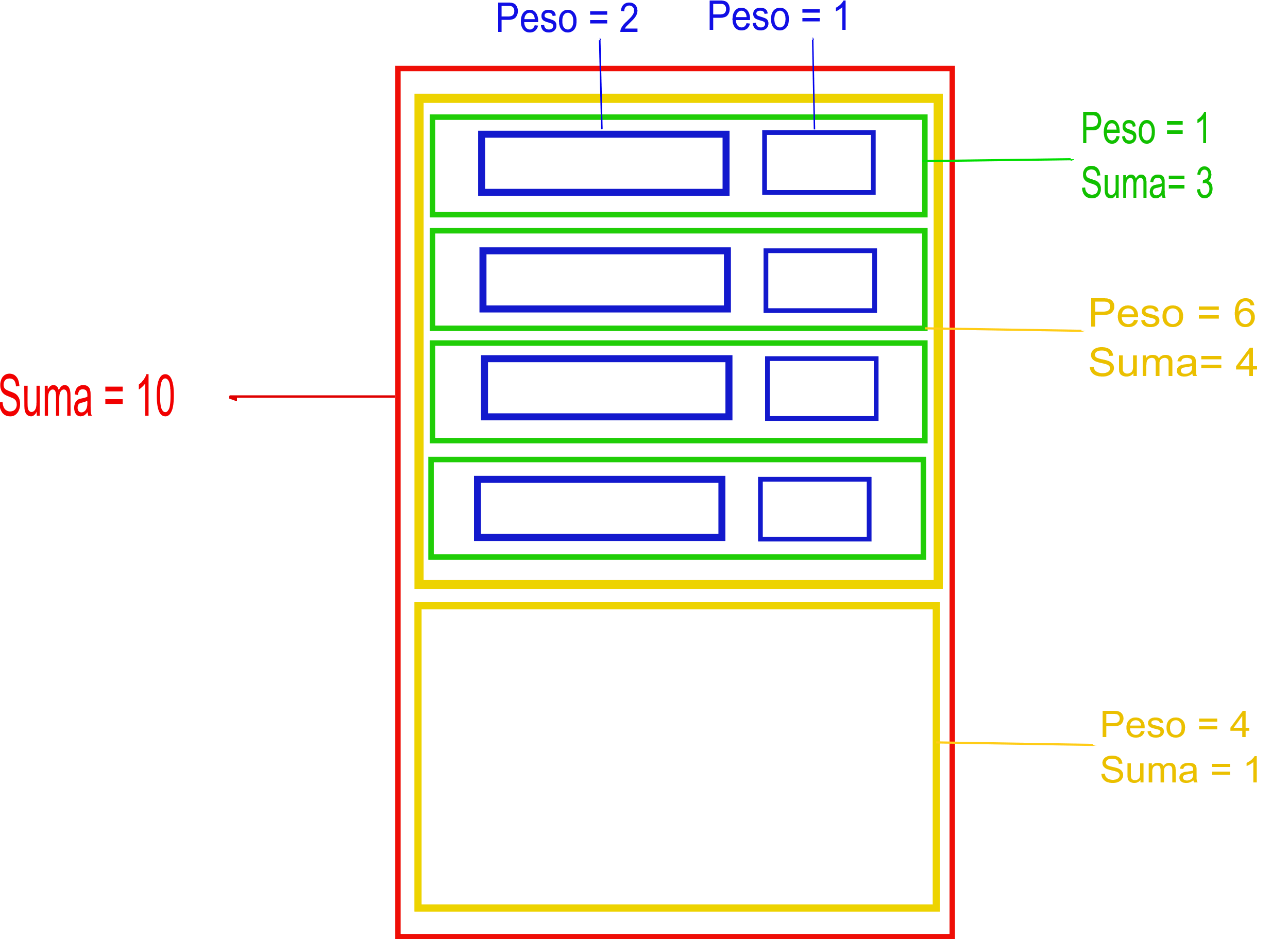
Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**Figura 16**



**Figura 17**



**Figura 18**

El código se describe a continuación. Irlo aregando a **ActividadSecundaria\_2**.

...

*/\*método responsable de administrar el diseño de la GUI\*/***private** LinearLayout crearGUI(){

...

linearPrincipal.setOrientation(LinearLayout.***VERTICAL***);  
*//pera definir los pesos de las filas que se agregaran*linearPrincipal.setWeightSum(10.0f);  
  
*//LinearLyout secundaria: arriba*LinearLayout linearSecundariaArriba = **new** LinearLayout(**this**);  
*//los componentes se agregarán verticalmente*linearSecundariaArriba.setOrientation(LinearLayout.***VERTICAL***);  
*//para definir los pesos de las filas que se agregaran*linearSecundariaArriba.setWeightSum(4.0f);  
  
*//LinearLyout secundaria: abajo*LinearLayout linearSecundariaAbajo = **new** LinearLayout(**this**);  
*//los componentes se agregarán verticalmente*linearSecundariaAbajo.setOrientation(LinearLayout.***VERTICAL***);  
*//pera definir los pesos de las filas que se agregaran*linearSecundariaAbajo.setWeightSum(1.0f);

...

}*//fin método crearGUI*

Proceder ahora a pegar estos **LinearLayout** secundarios a **linearPrincipal**. Para este se agrega el siguiente código en el que se debe definir los parámetros de pegado usando objetos de la clase **LinearLayout.LayoutParams**.

...

*/\*método responsable de administrar el diseño de la GUI\*/***private** LinearLayout crearGUI(){

...

*/\*  
Pegado de las dos LinearLayout secundarias a linearPrincipal.  
Aqui se debe tener cuidado que el segundo parámetro es 0 indicando  
que se agregan verticalmente respetando unos pesos. El primer parametro  
es MATCH\_PARENT indicando que esos componentes llenaran* ***todo el ancho*** *del contenedor  
\*/  
  
//pegar linear\_secundaria\_arriba a linear\_principal*LinearLayout.LayoutParams parametros\_pegado\_linear\_secundaria\_arriba\_a\_linear\_principal = **new** LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***,0);  
*//ocupará el 60% de linear\_principal*parametros\_pegado\_linear\_secundaria\_arriba\_a\_linear\_principal .**weight** = 6.0f;  
linearPrincipal.addView(linearSecundariaArriba, parametros\_pegado\_linear\_secundaria\_arriba\_a\_linear\_principal);  
  
*//pegar linear\_secundaria\_abajo a linear\_principal*LinearLayout.LayoutParams parametros\_pegado\_linear\_secundaria\_abajo\_a\_linear\_principal = **new** LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***,0);  
*//ocupará el 40% de linear\_principal*parametros\_pegado\_linear\_secundaria\_abajo\_a\_linear\_principal.**weight** = 4.0f;  
linearPrincipal.addView(linearSecundariaAbajo, parametros\_pegado\_linear\_secundaria\_abajo\_a\_linear\_principal);

...

}*//fin método crearGUI*

...

A continuación se crearán 4 filas **LinearLayout** que se pegarán **linearSecundariaArriba** con “pesos” de 1.0.

...

*/\*método responsable de administrar el diseño de la GUI\*/***private** LinearLayout crearGUI(){

...

*//creación las cuatro filas para pegar a linear\_secundaria\_arriba*LinearLayout linearUno = **new** LinearLayout(**this**);

*//los componentes se agregarán horizontalmente*

linearUno.setOrientation(LinearLayout.***HORIZONTAL***);  
*//para definir el peso de los componentes que se agregaran en esta fila*linearUno.setWeightSum(3.0f);  
  
LinearLayout linearDos = **new** LinearLayout(**this**);

*//los componentes se agregarán horizontalmente*

linearDos.setOrientation(LinearLayout.***HORIZONTAL***);  
*//para definir el peso de los componentes que se agregaran en esta fila*linearDos.setWeightSum(3.0f);  
  
LinearLayout linearTres = **new** LinearLayout(**this**);

*//los componentes se agregarán horizontalmente*

linearTres.setOrientation(LinearLayout.***HORIZONTAL***);  
*//para definir el peso de los componentes que se agregaran esta fila*linearTres.setWeightSum(3.0f);  
  
LinearLayout linearCuatro = **new** LinearLayout(**this**);

*//los componentes se agregarán horizontalmente*

linearCuatro.setOrientation(LinearLayout.***HORIZONTAL***);  
*//para definir el peso de los componentes que se agregaran en esta fila*linearCuatro.setWeightSum(3.0f);  
  
*//pegado de las filas linear a linear\_secundaria\_arriba*LinearLayout.LayoutParams parametros\_pegado\_filas\_a\_linear\_secundaria\_arriba = **new** LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***,0);  
parametros\_pegado\_filas\_a\_linear\_secundaria\_arriba .**weight** = 1.0f;  
linearSecundariaArriba.addView(linearUno, parametros\_pegado\_filas\_a\_linear\_secundaria\_arriba );  
linearSecundariaArriba.addView(linearDos, parametros\_pegado\_filas\_a\_linear\_secundaria\_arriba );  
linearSecundariaArriba.addView(linearTres, parametros\_pegado\_filas\_a\_linear\_secundaria\_arriba );  
linearSecundariaArriba.addView(linearCuatro, parametros\_pegado\_filas\_a\_linear\_secundaria\_arriba );

...

}*//fin método crearGUI*

...

Pegar ahora los componentes a las filas que se pegaron a **linearSecundariaArriba**. Cada una tendrá dos y se pegarán con pesos igual a 1.0 del total de 2.0 que tiene disponible cada una de estas filas. Es decir cada componente ocupara el 50% de la fila correspondiente. Observar que ya LinearLayout.LayoutParams tiene es el primer parámetro con valor 0 indicando que se agregarán los componentes horizontalmente respetando unos “pesos” definidos. El segundo parámetro ya es ***MATCH\_PARENT*** indicando que verticalmente llenará todo su contenedor es decir la fila correspondiente.

...

*/\*método responsable de administrar el diseño de la GUI\*/***private** LinearLayout crearGUI(){

...

*/\*  
Pegado de las dos LinearLayout secundarias a linear\_principal.  
Aqui se debe tener cuidado que el primer parámetro es 0 indicando  
que se agregan horizontalmente respetando unos pesos. El segundo parametro  
es MATCH\_PARENT indicando que esos componentes llenaran* ***todo el alto*** *del contenedor  
\*/  
  
  
//pegado de componentes a las filas pegadas linear\_secundaria\_arriba*LinearLayout.LayoutParams parametros\_pegado\_text\_view\_a\_filas\_arriba = **new** LinearLayout.LayoutParams(0,android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***);  
parametros\_pegado\_text\_view\_a\_filas\_arriba.**weight** = 2.0f;  
**int** a=**margenesResolucionIncluida**;  
parametros\_pegado\_text\_view\_a\_filas\_arriba.setMargins(a,a,a,a);  
  
LinearLayout.LayoutParams parametros\_pegado\_edit\_text\_a\_filas\_arriba = **new** LinearLayout.LayoutParams(0,android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***);  
parametros\_pegado\_edit\_text\_a\_filas\_arriba.**weight** = 1.0f;  
parametros\_pegado\_edit\_text\_a\_filas\_arriba.setMargins(a,a,a,a);  
  
linearUno.addView(**textNombre**, parametros\_pegado\_text\_view\_a\_filas\_arriba);  
linearUno.addView(**editNombres**, parametros\_pegado\_edit\_text\_a\_filas\_arriba);  
linearDos.addView(**textPrimerApellido**, parametros\_pegado\_text\_view\_a\_filas\_arriba);  
linearDos.addView(**editPrimerApellido**, parametros\_pegado\_edit\_text\_a\_filas\_arriba);  
linearTres.addView(**textSegundoApellido**, parametros\_pegado\_text\_view\_a\_filas\_arriba);  
linearTres.addView(**editSegundoApellido**, parametros\_pegado\_edit\_text\_a\_filas\_arriba);  
linearCuatro.addView(**textEdad**, parametros\_pegado\_text\_view\_a\_filas\_arriba);  
linearCuatro.addView(**editEdad**, parametros\_pegado\_edit\_text\_a\_filas\_arriba);

...

}*//fin método crearGUI*

...

Ahora se procede a pegar la imagen al **linearSecundariaAbajo** la cual ocupará todo este contenedor.

...

*/\*método responsable de administrar el diseño de la GUI\*/***private** LinearLayout crearGUI(){

...

*//pegado de imagen a linear\_secundaria\_abajo*LinearLayout.LayoutParams parametros\_pegado\_imagen\_abajo = **new** LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***,0);  
parametros\_pegado\_imagen\_abajo.**weight** = 1.0f;  
linearSecundariaAbajo.addView(**imagen**,parametros\_pegado\_imagen\_abajo);

...

}*//fin método crearGUI*

...

Por último agregar el método automático **onPause()** con el código allí escrito. Este método perteneciente al ciclo de vida de la actividad se activa cuando se sale de la **ActividadSecundaria\_2** y con el código que se agregó allí garantiza la actualización de los datos de la clase **AlmacenDatosRAM**.

...

*/\* Este método es automático\*/***protected void** onPause() {  
  
 String nombres = **editNombres**.getText().toString();  
 String primer\_apellido = **editPrimerApellido**.getText().toString();  
 String segundo\_apellido = **editSegundoApellido**.getText().toString();  
 String edad = **editEdad**.getText().toString();  
  
 AlmacenDatosRAM.*nombres* = nombres;  
 AlmacenDatosRAM.*apellidoUno* = primer\_apellido;  
 AlmacenDatosRAM.*apellidoDos* = segundo\_apellido;  
 *//pasar String a int* AlmacenDatosRAM.*edad* = Integer.*parseInt*(edad);  
  
 AlmacenDatosRAM.*habilitar\_boton\_tres*=**true**;  
  
 **super**.onPause();  
  
  
}

...

Proceder a compilar y ejecutar la aplicación para saber cómo está quedando. Debería desplegarse la GUI de la Figura 19 izquierda y luego de hacer clic en el botón DOS se despliega la GUI de la Figura 19 derecha. Hacer pruebas digitando información y regresando a la actividad principal y volver a hacer clic en el botón DOS. Salir de la aplicación y volver a ingresar y hacer clic en el botón DOS.

|  |  |
| --- | --- |

**Figura 19**

**Paso 7:**

Agregar más código a la **ActividadPrincipalMiDecimaSeptimaApp**.

El método **onDestroy()** de la clase principal **ActividadPrincipalMiDecimaSeptimaApp** debe responsabilizarse de llevar a valores iniciales los datos de la clase **AlmacenDatosRAM** cuando se salga de la aplicación. Agregar éste código en ese método de esa clase. El método **onResume()** debe responsabilizarse de actualizar el botón TRES.

...

*/\* Métodos automáticos\*/***protected void** onResume(){  
 **super**.onResume();  
  
 **if**(AlmacenDatosRAM.*habilitar\_boton\_tres*==**true**){  
 **botonTres**.setEnabled(**true**);  
 } **else** {  
  
 **botonTres**.setEnabled(**false**);  
  
 }  
}  
  
**protected void** onDestroy() {

*//Volver los valores de los daos a su estado por defecto*

AlmacenDatosRAM.*nombres* = **"Xxxx Xxxx"**;  
 AlmacenDatosRAM.*apellidoUno* = **"Yyyy"**;  
 AlmacenDatosRAM.*apellidoDos* = **"Zzzz"**;  
 AlmacenDatosRAM.*edad* = 0;

**this**.finish();

**super**.onDestroy();  
  
  
}

...

**Paso 8: Agregar el código a la ActividadSecundaria\_3**

Este es el código de la **ActividadSecundaria\_3.** Agregarlo.

**public class** ActividadSecundaria\_3 **extends** Activity {  
  
  
 **private int tamanoLetraResolucionIncluida**,**margenesResolucionIncluida**;  
 **private** TextView **aviso**;  
  
  
 **public void** onCreate(Bundle savedInstanceState){  
 **super**.onCreate(savedInstanceState);  
  
 gestionarResolucion();  
  
 *//para crear elementos de la GUI* crearElementosGUI();  
  
 *//para informar cómo se debe pegar el adminitrador de  
 //diseño obtenido con el método GUI* ViewGroup.LayoutParams parametro\_layout\_principal = **new** ViewGroup.LayoutParams(ViewGroup.LayoutParams.***MATCH\_PARENT***, ViewGroup.LayoutParams.***MATCH\_PARENT***);  
  
  
 *//pegar el contenedor con la GUI* **this**.setContentView(crearGUI(), parametro\_layout\_principal);  
  
 *//para administrar los eventos* eventos();  
  
  
  
 } *//fin del método onCreate*

**private void** gestionarResolucion(){  
  
tamanoLetraResolucionIncluida =

(int)(0.8f\*AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);  
margenesResolucionIncluida = (int)(1.2f\*AlmacenDatosRAM.*tamanoLetraResolucionIncluida*);

}*//fin método gestionandoResolucion()  
  
  
 /\*método responsable de la creación de los elementos de la GUI\*/* **private void** crearElementosGUI(){  
  
  
 **aviso** = **new** TextView(**this**);  
 **aviso**.setTextSize(**tamanoLetraResolucionIncluida**);  
 String nombres= AlmacenDatosRAM.*nombres*;  
 String primer\_apellido= AlmacenDatosRAM.*apellidoUno*;  
 String segundo\_apellido= AlmacenDatosRAM.*apellidoDos*;  
 String edad=**""** + AlmacenDatosRAM.*edad*;  
  
 String componer\_cadena= **"Soy "**+ nombres + **" "** + primer\_apellido + **" "** + segundo\_apellido +  
 **" y mi edad es "** + edad + **" años"**;  
  
  
 **aviso**.setText(componer\_cadena);  
  
  
  
 }*//fin método crearElementosGUI  
  
  
 /\*método responsable de administrar el diseño de la GUI\*/* **private** LinearLayout crearGUI(){  
  
 *//LinearLayoutPrincipal* LinearLayout linearPrincipal = **new** LinearLayout(**this**);  
 linearPrincipal.setOrientation(LinearLayout.***VERTICAL***);  
 linearPrincipal.setBackgroundColor(Color.***WHITE***);  
 linearPrincipal.setWeightSum(1.0f);  
  
 *//parámtero de pegado  
 //pegado de botones a cada fila* LinearLayout.LayoutParams parametros\_aviso = **new** LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***, 0);  
 parametros\_aviso.**weight**=1.0f;  
 parametros\_aviso.setMargins(**margenesResolucionIncluida**,0,0,0);  
 linearPrincipal.addView(**aviso**,parametros\_aviso);  
  
 **return** linearPrincipal;  
  
  
 }*//fin método crearGUI  
  
  
 /\*Administra los eventos de la GUI\*/* **private void** eventos(){  
  
  
 }*//fin método eventos*

*}//fin Actividad*

**Paso 9:**

Proceder a compilar y ejecutar la aplicación para saber cómo está quedando. Debería desplegarse la GUI de la Figura 20 izquierda y luego de hacer clic en el botón DOS se despliega la GUI de la Figura 14 centro. Hacer pruebas digitando información y regresando a la actividad principal y volver a hacer clic en el botón TRES. Salir de la aplicación y volver a ingresar y hacer clic en el botón TRES.

Instalar y ejecutar la aplicación en diferentes dispositivos móviles.

|  |  |  |
| --- | --- | --- |

**Figura 20**

**MI DECIMA OCTAVA APP: UNA APP CON MIS PROPIOS BOTONES**

En esta app se ilustra como diseñar nuestros propios botones para darle un aspecto a la GUI muy personal. Los tres botones que se usan en esta app tienen el aspecto de la Figura 21.

|  |  |  |
| --- | --- | --- |

**Figura 21**

Esas imágenes deben estar depositadas en el directorio de la aplicación en la carpeta **app>src>main>res>drawable>**.

**Paso 1: Creación del proyecto y su estructura**

Usando Android Studio proceder a crear un proyecto denominado **MiDecimaOctavaApp** con la estructura de la Figura 22.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Figura 22**

**Paso 2: Código de la clase MiBoton**

Agregar el siguiente código a la clase MiBoton.

public class MiBoton extends ImageView {  
  
 private Bitmap imagen;  
 private Bitmap imagen\_escalada = null;  
 private float tamamano\_letra;  
 private String cadena = "";  
  
  
 public MiBoton(Context context) {  
  
 super(context);  
  
  
 }  
  
  
 public void setText(String cadena) {  
  
 this.cadena = cadena;  
  
  
 }  
  
  
 public String getText() {  
  
 this.cadena = cadena;  
  
 return cadena;  
  
  
 }  
  
  
 public void setImagen(int imagen\_importada) {  
  
  
 imagen = BitmapFactory.*decodeResource*(getResources(), imagen\_importada);  
  
  
 }  
  
  
 protected void onDraw(Canvas canvas) {  
  
  
 Paint pincel = new Paint();  
 pincel.setAntiAlias(true);  
 pincel.setLinearText(true);  
  
 canvas.save();  
  
 float ancho=getWidth();  
 float alto=getHeight();  
 float escala=1;  
 if(ancho>alto){  
 escala=alto;  
 } else {escala=ancho;}  
  
 if (imagen != null) {  
  
 imagen\_escalada = escalarImagen(imagen, (int) (0.9f \* escala), (int) (0.9f \* escala));  
  
 }  
  
 int ex = (getWidth() - imagen\_escalada.getWidth()) / 2;  
 int ey = (getHeight() - imagen\_escalada.getHeight()) / 2;  
  
 tamamano\_letra = 0.08f \* getWidth();  
  
 if (imagen != null) {  
 canvas.drawBitmap(imagen\_escalada, ex, ey, null);*//dimension,dimension, null);* }  
  
  
 pincel.setTextSize(tamamano\_letra);  
 float anchoCadenaUnidades = pincel.measureText(cadena);  
 float posicion\_x\_letra = (getWidth() - anchoCadenaUnidades) / 2;  
 float posicion\_y\_letra = ey + imagen\_escalada.getHeight() + tamamano\_letra;  
  
 pincel.setColor(Color.*BLACK*);  
 canvas.drawText(cadena, posicion\_x\_letra, posicion\_y\_letra, pincel);  
  
  
 canvas.restore();  
 invalidate();  
  
 }  
  
 */\*  
 Se escala con Matrix para no dañar la resolución  
 \*/* public Bitmap escalarImagen(Bitmap bitmap, int newWidth, int newHeight) {  
 Bitmap scaledBitmap = Bitmap.*createBitmap*(newWidth, newHeight, bitmap.getConfig());  
  
 float scaleX = newWidth / (float) bitmap.getWidth();  
 float scaleY = newHeight / (float) bitmap.getHeight();  
  
 Matrix scaleMatrix = new Matrix();  
 scaleMatrix.setScale(scaleX, scaleY, 0, 0);  
  
 Canvas canvas = new Canvas(scaledBitmap);  
 canvas.setMatrix(scaleMatrix);  
 Paint paint = new Paint(Paint.*FILTER\_BITMAP\_FLAG*);  
 paint.setAntiAlias(true);  
 paint.setDither(true);  
 paint.setFilterBitmap(true);  
 canvas.drawBitmap(bitmap, 0, 0, paint);  
  
 return scaledBitmap;  
  
 }  
  
  
}

**Paso 3: Código de la clase GaugeSimple**

Agregar el siguiente código a la clase **GaugeSimple**.

public class GaugeSimple extends View {  
  
 private float largo;  
 private float minimo = 0;  
 private float maximo = 100f;  
 private float medida = 40.0f;*//tomar como medida inicial* private String unidades = "UNIDADES";  
 private int colorPrimerTercio = Color.*rgb*(200, 200, 0);  
 private int colorSegundoTercio = Color.*rgb*(0, 180, 0);  
 private int colorTercerTercio = Color.*RED*;  
 private int colorLineas = Color.*BLACK*;  
 private int colorFondo = Color.*WHITE*;  
 private int colorNumerosDesplieggue = Color.*BLACK*;  
 private int colorFranjaDinamica = Color.*rgb*(0, 0, 255);  
 private int angPrimertercio = 100;  
 private int angSegundoTercio = 100;  
 private int angTercerTercio = 50;  
  
  
 */\*\*  
 \* Constructor de GaugeSimple  
 \*/* public GaugeSimple(Context context) {  
  
 super(context);  
  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*HONEYCOMB*) {  
 this.setLayerType(View.*LAYER\_TYPE\_SOFTWARE*, null);  
  
 }  
 }  
  
 */\*\*  
 \* Modifica el rango de medicion  
 \* desde minimo hasta maximo  
 \*  
 \* @param minimo  
 \* @param maximo  
 \*/* public void setRango(float minimo, float maximo) {  
  
 this.minimo = minimo;  
 this.maximo = maximo;  
  
 }  
  
  
 */\*\*  
 \* Modifica el valor medido  
 \*  
 \* @param medida  
 \*/* public void setMedida(float medida) {  
  
 this.medida = medida;  
  
 }  
  
  
 */\*\*  
 \* Regresa el valor medido  
 \*  
 \* @return medida  
 \*/* public float getMedida() {  
  
 return medida;  
 }  
  
  
 */\*\*  
 \* Modifica las unidades del instrumento virtual  
 \*  
 \* @param unidades  
 \*/* public void setUnidades(String unidades) {  
  
 this.unidades = unidades;  
  
 }  
  
  
 */\*\*  
 \* Modifica los colores de los sectores circulares  
 \*  
 \* @param colorPrimerTercio  
 \* @param colorSegundoTercio  
 \* @param colorTercerTercio  
 \*/* public void setColorSectores(int colorPrimerTercio, int colorSegundoTercio, int colorTercerTercio) {  
  
 this.colorPrimerTercio = colorPrimerTercio;  
 this.colorSegundoTercio = colorSegundoTercio;  
 this.colorTercerTercio = colorTercerTercio;  
  
 }  
  
 */\*\*  
 \* Modifica los angulos de los sectores circulares  
 \* Deben sumar 250 grados  
 \*  
 \* @param angPrimerTercio  
 \* @param angSegundoTercio  
 \* @param angTercerTercio  
 \*/* public void setAngulosSectores(int angPrimerTercio, int angSegundoTercio, int angTercerTercio) {  
 this.angPrimertercio = angPrimerTercio;  
 this.angSegundoTercio = angSegundoTercio;  
 this.angTercerTercio = angTercerTercio;  
  
 }  
  
 public void setColorFranjaDinámica(int colorFranjaDinamica) {  
  
 this.colorFranjaDinamica = colorFranjaDinamica;  
  
  
 }  
  
 */\*\*  
 \* Modifica el color de fondo del tacometro  
 \*  
 \* @param colorFondo  
 \*/* public void setColorFondoTacometro(int colorFondo) {  
  
 this.colorFondo = colorFondo;  
  
  
 }  
  
  
 */\*\*  
 \* Modifica el color de las lineas del tacometro  
 \*  
 \* @param color\_lineas  
 \*/* public void setColorLineasTacometro(int color\_lineas) {  
  
 this.colorLineas = color\_lineas;  
  
  
 }  
  
  
 */\*\*  
 \* Modifica el color del numero que se despliega  
 \*  
 \* @param colorNumerosDesplieggue  
 \*/* public void setColorNumeroDespliegue(int colorNumerosDesplieggue) {  
  
 this.colorNumerosDesplieggue = colorNumerosDesplieggue;  
  
 }  
  
  
 */\*\*  
 \* @param canvas  
 \*/  
  
 //método para dibujar* protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
  
 */\*  
 se graba el estado actual del canvas  
 para al final restaurarlo  
 \*/* canvas.save();  
  
  
 */\*  
 La vista tendra las mismas dimensiones de su  
 contenedor  
 \*/* float ancho = this.getWidth();*//ancho de la vista* float alto = this.getHeight();*//alto de la vista  
  
 /\*  
 Se define la variable largo como el 80%  
 del menor valor entre alto y largo del  
 contenedor  
 \*/* if (ancho > alto) {  
  
 largo = 0.8f \* alto;  
  
 } else {  
  
 largo = 0.8f \* ancho;  
  
  
 }  
  
 */\*  
 se hace tralación del (0,0) al centro  
 del contenedor  
 \*/* canvas.translate(0.5f \* ancho, 0.5f \* alto);  
  
 *//configurando el pincel* Paint pincel = new Paint();  
 *//evita efecto sierra* pincel.setAntiAlias(true);  
 *//tamaño texto* pincel.setTextSize(0.05f \* largo);  
 *//para mejor manejo de la métrica de texto* pincel.setLinearText(true);  
 *//para efectos de buen escalado de bitmaps* pincel.setFilterBitmap(true);  
 *//para buen manejo de gradientes de color* pincel.setDither(true);  
  
  
 float esquinaSuperiorIzquierdaX = -0.5f \* largo;  
 float esquinaSuperiorIzquierdaY = -0.5f \* largo;  
 float esquinaInferiorDerechaX = 0.5f \* largo;  
 float esquinaInferiorDerechaY = 0.5f \* largo;  
  
 *//dibujar los tres segementos circulares* RectF rect = new RectF(esquinaSuperiorIzquierdaX, esquinaSuperiorIzquierdaY,  
 esquinaInferiorDerechaX, esquinaInferiorDerechaY);  
  
 pincel.setColor(colorPrimerTercio);  
 canvas.drawArc(rect, 145, angPrimertercio, true, pincel);  
 pincel.setColor(colorSegundoTercio);  
 canvas.drawArc(rect, 145 + angPrimertercio, angSegundoTercio, true, pincel);  
 pincel.setColor(colorTercerTercio);  
 canvas.drawArc(rect, 145 + angPrimertercio + angSegundoTercio, angTercerTercio, true, pincel);  
  
  
 float indent = (float) (0.05 \* largo);  
 float posicionY = (float) (0.5 \* largo);  
  
 *//franja dinámica  
 //calcular angulo para ubicar la aguja de acuerdo al valor medido* float angulo\_rotacion\_medida = 235 + (250f / (maximo - minimo)) \* (medida - minimo);  
 float a = (float) 0.01 \* largo;  
 rect = new RectF(esquinaSuperiorIzquierdaX + a, esquinaSuperiorIzquierdaY + a,  
 esquinaInferiorDerechaX - a, esquinaInferiorDerechaY - a);  
 pincel.setColor(colorFranjaDinamica);  
 canvas.drawArc(rect, 145, angulo\_rotacion\_medida - 235, true, pincel);  
  
 */\*  
 dibujar el tacometro sin la aguja  
 \*/  
 //aqui empieza el dibujo* float radio = (float) (0.48 \* largo);  
 pincel.setColor(colorFondo);  
 canvas.drawCircle(0, 0, radio, pincel);  
 pincel.setColor(colorLineas);  
 pincel.setStyle(Paint.Style.*STROKE*);  
 canvas.drawCircle(0, 0, radio, pincel);  
 pincel.setStrokeWidth(1f);  
 canvas.drawCircle(0, 0, 0.5f \* largo, pincel);  
  
 *//grosor de las líneas* pincel.setStrokeWidth(0.01f \* largo);  
  
  
  
 */\*  
 Divisiones grandes  
 Se dibuja primero la división vertical.  
 Luego se repite rotando de a 50 grados comenzando  
 en 235 grados.  
 \*/* pincel.setStyle(Paint.Style.*FILL*);  
 for (int i = 0; i < 6; i = i + 1) {  
 float anguloRotacion = 235 + 50 \* i;  
 canvas.rotate(anguloRotacion, 0, 0);  
 canvas.drawLine(0, -posicionY, 0, -posicionY + indent, pincel);  
 canvas.rotate(-anguloRotacion, 0, 0);  
  
 */\*  
 Dibujar los números  
 \*/* int valorIncrementoMarcas = (int) ((maximo - minimo) / 5f);  
 int valorMarca = (int) (minimo + valorIncrementoMarcas \* i);  
 String numero = "" + valorMarca;  
  
 *//ancho de la cadena del número* float anchoCadenaNumero = pincel.measureText(numero);  
  
 canvas.save();  
 *//dibuja números rotados* canvas.rotate(anguloRotacion, 0, 0);  
 *//endereza los números a orientación horizontal* canvas.rotate(-anguloRotacion, 0, -posicionY + 2.5f\*indent);  
 canvas.drawText(numero, -0.5f \* anchoCadenaNumero, -posicionY + 2.5f \* indent, pincel);  
 canvas.restore();  
  
 }  
  
 */\*  
 Divisiones pequeñas  
 Se dibuja primero la división vertical.  
 Luego se repite rotando de a 10 grados comenzando  
 en 235 grados.  
 \*/* pincel.setStyle(Paint.Style.*STROKE*);  
 *//grosor de las líneas* pincel.setStrokeWidth(0.005f \* largo);  
 for (int i = 0; i < 26; i = i + 1) {  
  
 float anguloRotacion = 235 + 10 \* i;  
 canvas.rotate(anguloRotacion, 0, 0);  
 canvas.drawLine(0, -posicionY, 0, -posicionY + (float) (0.6 \* indent), pincel);  
 canvas.rotate(-anguloRotacion, 0, 0);  
  
 }  
 *//aqui termina dibujo del tacometro sin aguja  
  
  
  
 /\*  
 dibujar la aguja  
 \*/  
 //aqui empieza dibujo de la aguja* pincel.setStrokeWidth(0.005f \* largo);  
 pincel.setColor(Color.*RED*);  
 canvas.rotate(angulo\_rotacion\_medida, 0, 0);  
 float b = (float) (1.5f\* indent);  
 canvas.drawLine(0, -posicionY, 0, b, pincel);  
 canvas.rotate(-angulo\_rotacion\_medida, 0, 0);  
 pincel.setStyle(Paint.Style.*FILL*);  
 pincel.setColor(colorFondo);  
 canvas.drawCircle(0, 0, (float) (0.4 \* indent), pincel);  
 pincel.setColor(Color.*RED*);  
 pincel.setStyle(Paint.Style.*STROKE*);  
 canvas.drawCircle(0, 0, (float) (0.4 \* indent), pincel);  
  
 *//dibujar circulo apoyo aguja  
  
 //aquí termina dibujo de la aguja  
  
 //Dibujar las unidades* pincel.setStyle(Paint.Style.*FILL*);  
 pincel.setColor(colorLineas);  
 pincel.setTextSize(0.08f \* largo);  
 float anchoCadenaUnidades = pincel.measureText(unidades);  
 canvas.drawText(unidades, -0.5f \* anchoCadenaUnidades,- 0.15f \* largo, pincel);  
 *//aqui termina dibujo de las unidades  
  
 //aqui despliegue de la medida* pincel.setTextSize(0.1f \* largo);  
 float anchoCadenaNumero = pincel.measureText("" + medida);  
 pincel.setColor(colorNumerosDesplieggue);  
 canvas.drawText("" + medida, -0.5f \* anchoCadenaNumero, 0.2f \* largo, pincel);  
  
 *//marcar empresa* String empresa = "IoT.PhysicsSensor";  
 pincel.setTextSize(0.05f \* largo);  
 float anchoCadenaNombreEmpresa = pincel.measureText(empresa);  
 canvas.drawText(empresa, -0.5f \* anchoCadenaNombreEmpresa, 0.35f \* largo, pincel);  
  
  
  
 *//se restaura el canvas al estado incial  
 //el que se garbó al principio de este método* canvas.restore();  
  
 *//para efectos de animación* invalidate();  
  
 }*//fin onDraw*}

**Paso 4: Código de la clase ActividadInstrumentos**

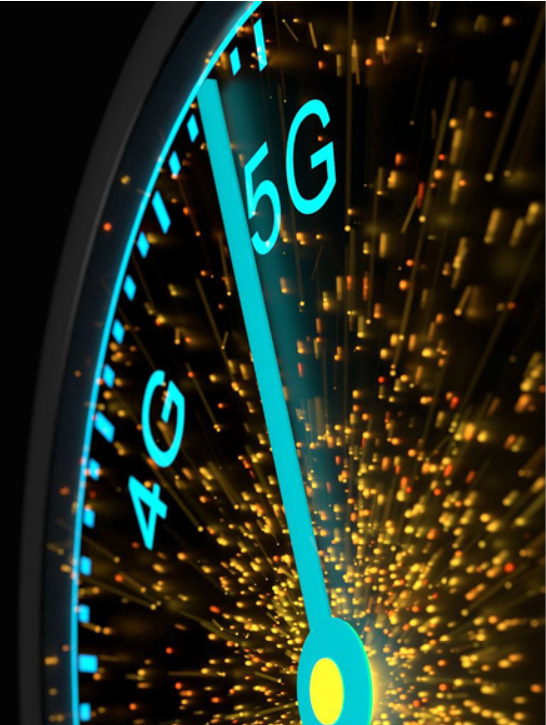
Agregar el siguiente código a la clase **ActividadInstrumentos**

**public class** ActividadInstrumentos **extends** Activity {  
  
 **private** GaugeSimple **temperatura**;  
 **private** GaugeSimple **humedad**;  
  
  
 */\*\*  
 \* Called when the activity is first created.  
 \*/* @Override  
 **public void** onCreate(Bundle icicle) {  
 **super**.onCreate(icicle);  
  
 crearElementosGUI();  
  
  
  
 ViewGroup.LayoutParams linear\_layou\_principal = **new** ViewGroup.LayoutParams(ViewGroup.LayoutParams.***MATCH\_PARENT***, ViewGroup.LayoutParams.***MATCH\_PARENT***);  
  
  
 **this**.setContentView(crearGUI(), linear\_layou\_principal);  
  
 }  
  
  
 **private void** crearElementosGUI(){  
  
 */\*  
 Gauge de temperatura  
 \*/* **temperatura**= **new** GaugeSimple(**this**);  
  
 *//rngos de la escala* **temperatura**.setRango(-10,40);  
 *//colores de la escala* **temperatura**.setColorSectores(Color.***CYAN***,Color.***GREEN***,Color.***RED***);  
 *//unidades* **temperatura**.setUnidades(**"Temperatura en C"**);  
 *//valor medido* **temperatura**.setMedida(25);  
  
 */\*  
 Gauge de humedad relativa  
 \*/* **humedad**= **new** GaugeSimple(**this**);  
  
 *//rngos de la escala* **humedad**.setRango(0,100);  
 *//colores de la escala* **humedad**.setColorSectores(Color.*rgb*(100, 250,80),Color.*rgb*(250, 150,80),Color.*rgb*(200, 250,80));  
 *//unidades* **humedad**.setUnidades(**"Humedad Relativa (%)"**);  
 *//valor medido* **humedad**.setMedida(70);  
  
  
 }  
  
  
 **private** LinearLayout crearGUI() {  
  
 LinearLayout linearPrincipal = **new** LinearLayout(**this**);  
 *//los componentes se agregarán verticalmente* linearPrincipal.setOrientation(LinearLayout.***VERTICAL***);  
 *//para definir los pesos de las filas que se agregaran* linearPrincipal.setWeightSum(2.0f);  
  
 *//las dos filas* LinearLayout linearArriba = **new** LinearLayout(**this**);  
 LinearLayout linearAbajo = **new** LinearLayout(**this**);  
   
  
  
 LinearLayout.LayoutParams parametros\_pegado\_a\_principal = **new** LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***, android.view.ViewGroup.LayoutParams.***MATCH\_PARENT***);  
 parametros\_pegado\_a\_principal.**weight** = 1.0f;  
 linearPrincipal.addView(linearArriba, parametros\_pegado\_a\_principal);  
 linearPrincipal.addView(linearAbajo, parametros\_pegado\_a\_principal);  
  
 *//pegar los gauges* linearArriba.addView(**temperatura**);  
 linearAbajo.addView(**humedad**);  
  
  
  
  
 **return** linearPrincipal;  
  
  
 }  
  
  
}

**Paso 5:**

**Código de la clase ActividadPrincipalMiDecimaOctavaApp**

Agregar el siguiente código a la clase **ActividadPrincipalMiDecimaOctav**a. Se usará la siguiente imagen, Figura 23, de fondo en la pantalla desplegada por esta actividad. Por esta razón deberá estar incluida en la carpeta **app>src>main>res>drawable>.**



**Figura 23**

public class ActividadPrincipalMiDecimaOctavaApp extends Activity {  
  
 *//private Button informar, actualizar,aceptar;* private MiBoton entrar, salir, actualizar;  
  
 */\*\*  
 \* Called when the activity is first created.  
 \*/* @Override  
 public void onCreate(Bundle icicle) {  
 super.onCreate(icicle);  
  
 *//Creación elementosGUI* creacionElementosGui();  
  
 eventos();  
  
 ViewGroup.LayoutParams linear\_layou\_principal = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 this.setContentView(crearGUI(), linear\_layou\_principal);  
  
 }  
  
  
 *//Creación elementosGUI* private void creacionElementosGui() {  
  
 entrar = new MiBoton(this);  
 entrar.setImagen(R.drawable.*entrar*);  
 entrar.setText("ENTRAR ");  
  
 salir = new MiBoton(this);  
 salir.setImagen(R.drawable.*salir*);  
 salir.setText("SALIR");  
  
 actualizar = new MiBoton(this);  
 actualizar.setImagen(R.drawable.*actualizar*);  
 actualizar.setText("ACTUALIZAR");  
  
  
  
 }  
  
  
 private LinearLayout crearGUI(){  
  
 LinearLayout linear\_layout\_principal =new LinearLayout(this);  
 linear\_layout\_principal.setOrientation(LinearLayout.*VERTICAL*);  
 linear\_layout\_principal.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
 linear\_layout\_principal.setGravity(Gravity.*FILL*);  
 linear\_layout\_principal.setBackgroundColor(Color.*WHITE*);  
 linear\_layout\_principal.setWeightSum(10);  
  
  
 *//LinearLayout primera fila* LinearLayout linear\_layout\_primera\_fila = new LinearLayout(this);  
 linear\_layout\_primera\_fila.setOrientation(LinearLayout.*HORIZONTAL*);  
 linear\_layout\_primera\_fila.setGravity(Gravity.*FILL*);  
 linear\_layout\_primera\_fila.setBackgroundColor(Color.*WHITE*);  
 LinearLayout.LayoutParams parametros\_primera\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, 0);*//ViewGroup.LayoutParams.WRAP\_CONTENT);* parametros\_primera\_fila.weight = 8.0f;  
 linear\_layout\_primera\_fila.setLayoutParams(parametros\_primera\_fila);  
  
  
 *//fondo primera fila* Drawable fondo = getResources().getDrawable(R.drawable.*cinco\_g*);  
 linear\_layout\_primera\_fila.setBackgroundDrawable(fondo);  
  
 *//LinearLayout segunda fila* LinearLayout linear\_layout\_segunda\_fila = new LinearLayout(this);  
 linear\_layout\_segunda\_fila.setOrientation(LinearLayout.*HORIZONTAL*);  
 linear\_layout\_segunda\_fila.setGravity(Gravity.*FILL*);  
 LinearLayout.LayoutParams parametros\_segunda\_fila = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*,0);*// ViewGroup.LayoutParams.WRAP\_CONTENT);* parametros\_segunda\_fila.weight = 2.0f;  
 linear\_layout\_segunda\_fila.setWeightSum(3.0f);  
 linear\_layout\_segunda\_fila.setLayoutParams(parametros\_segunda\_fila);  
  
  
 LinearLayout.LayoutParams parametros\_pegado\_boton = new LinearLayout.LayoutParams(0, android.view.ViewGroup.LayoutParams.*MATCH\_PARENT*);  
 parametros\_pegado\_boton.weight = 1.0f;  
 entrar.setLayoutParams(parametros\_pegado\_boton);  
 salir.setLayoutParams(parametros\_pegado\_boton);  
 actualizar.setLayoutParams(parametros\_pegado\_boton);  
 linear\_layout\_segunda\_fila.addView(entrar);  
 linear\_layout\_segunda\_fila.addView(salir);  
 linear\_layout\_segunda\_fila.addView(actualizar);  
  
  
 linear\_layout\_principal.addView(linear\_layout\_primera\_fila);  
 linear\_layout\_principal.addView(linear\_layout\_segunda\_fila);  
  
  
  
 return linear\_layout\_principal;  
  
 }  
  
 private void eventos(){  
  
 entrar.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 lanzarActividadInstrumentos();  
  
  
 }  
 });  
  
 salir.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
 finish();  
 }  
 });  
  
 actualizar.setOnClickListener(new View.OnClickListener() {  
  
 public void onClick(View v) {  
  
  
 }  
 });  
 }  
  
  
 private void lanzarActividadInstrumentos(){  
  
 Intent intent = new Intent(this, ActividadInstrumentos.class);  
 startActivity(intent);  
 }  
  
}

**Paso 6: Código del archivo manifiesto**

Agregar el siguiente código al archivo manifiesto.

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.curso\_simulaciones.midecimaoctavaapp">  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MiDecimaOctavaApp"  
 tools:targetApi="31" >  
  
 *<!--Declarar la actividad que lanzara la aplicación-->* <activity android:name=".ActividadPrincipalMiDecimaOctavaApp"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait"  
 android:exported="true">  
  
  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
  
 </activity>  
  
  
 *<!--declarando otras actividades-->* <activity  
 android:name=".actividades\_secundarias.ActividadInstrumentos"  
 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"  
 android:screenOrientation="portrait">  
 </activity>  
  
  
 </application>  
  
  
  
</manifest>

Compilar y ejecutar. Se despliega la GUI de la Figura 24 izquierda. Hacer clic en el botón ENTRAR y se despliega la GUI de la Figura 24 derecha. El botón ACTUALIZAR se dejó sin evento. El botón SALIR permite salir de la aplicación liberando los recursos.

|  | Imagen que contiene medidor, cd  Descripción generada automáticamente |
| --- | --- |

**Figura 24**

**TAREA**Desarrollar una aplicación denominada **MiDecimaNovenaApp** que cumpla los requisitos y diseño que se describen a continuación.

* La distribución de clases es igual a la de la aplicación MiDecimoSeptimaApp.

Son cuatro actividades (**Activity**) y una clase para almacenar datos en variables estáticas en RAM (**AlmacenDatosRAM**).

* La GUI de la actividad principal (**ActividadPrincipalMiDecimaNovenaApp**) es la de la Figura 25.

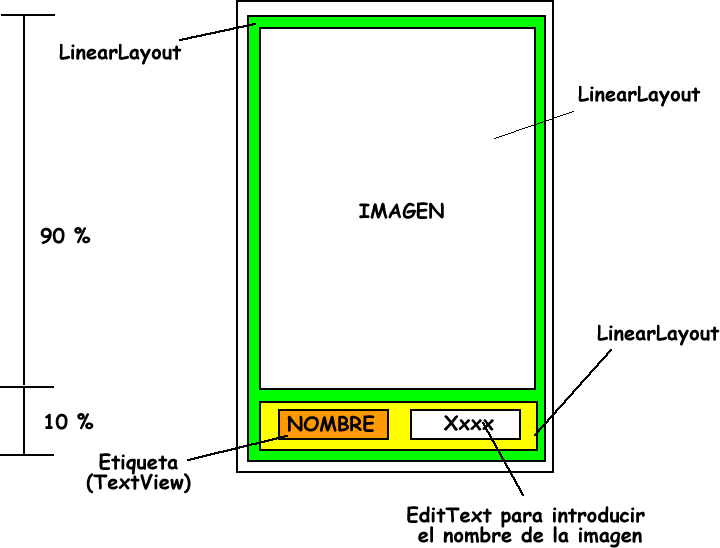
Gráfico de rectángulos

Descripción generada automáticamente

**Figura 25**

Los botones TRES y CUATRO en su estado inicial están desactivados. Los otros dos están activados.

* El botón UNO lanza **ActividadSecundaria\_1**, el botón DOS lanza **ActividadSecundarias\_2** y el botón TRES lanza **ActividadSecundarias\_3.** El botón CUATRO no hará nada y permanecerá inactivo.
* La **ActividadSecundaria\_1** tiene la GUI de la Figura 26.

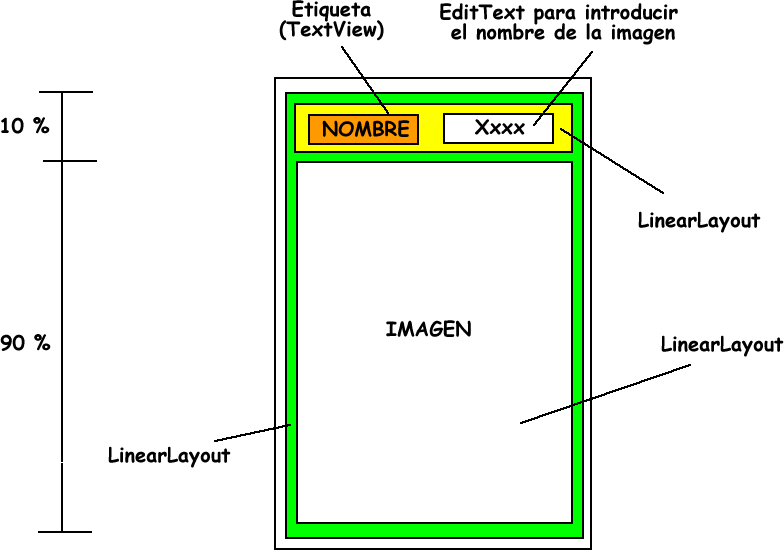


**Figura 26**

El nombre de la imagen que es introducido en esta GUI debe quedar almacenado en la clase **AlmacenDatosRAM**. Cuando se regrese de esta GUI a la GUI principal debe activarse el botón TRES.

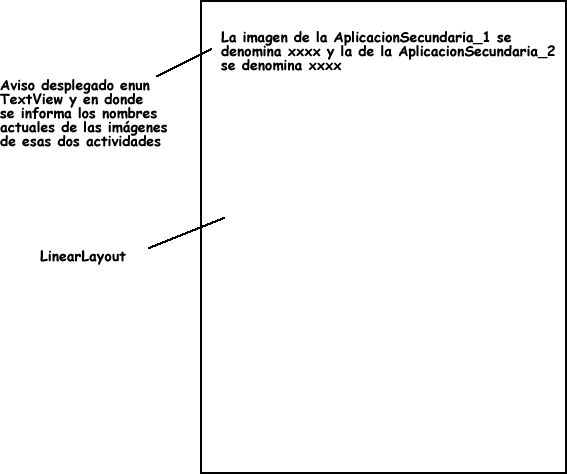
* La **ActividadSecundaria\_2** tiene la GUI de la Figura 27.

El nombre de la imagen que es introducido en esta GUI debe quedar almacenado en la clase **AlmacenDatosRAM**. Cuando se regrese de esta GUI a la GUI principal debe activarse el botón TRES en caso de no estar activado.



**Figura 27**

* Al hacer clic en el botón TRES se deberá desplegar la GUI de la **ActividadSecundaria\_3**  en donde se informa los nombres actuales de las imágenes de las otras dos actividades secundarias, Figura 28.



**Figura 28**

* Las GUIs no deben rotar cuando se rota la pantalla. Tampoco deben tener una barra superior donde se despliega el título.
* Cuando se salga de la aplicación se debe finalizar correctamente y se deberán llevar los valores almacenados en la clase **AlmacenDatosRAM** a sus valores por defecto.
* Instalar la aplicación con un **icono personal**.
* Ejecutar la aplicación en diferentes dispositivos móviles.

1. La imagen fue obtenida de: <http://www.risasinmas.com/wp-content/uploads/2012/06/gatito-paseando-rsm.jpg>. Se le cambio de nombre y de formato, aunque también se puede dejar en formato **jpg**. Si el lector desea puede cambiar la imagen a su gusto. [↑](#footnote-ref-0)
2. La imagen usada por el autor se puede obtener en https://www.profesionalnet.net/identidad-corporativa/. Se pasó a formato **png** pero igual se puede dejar en formato **jpg**. También se puede usar la imagen que dese el lector. [↑](#footnote-ref-1)