

PRACTICA 12.

OBJETIVO GENERAL.

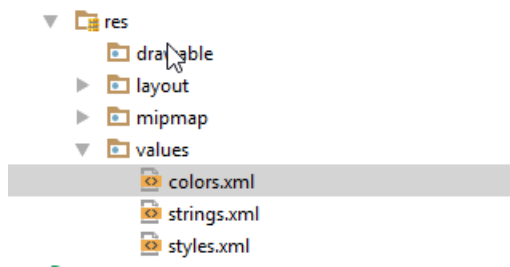
El alumno entenderá el uso adecuado de las carpetas de recursos de Android studio y aplicará cada uno de ellos en una aplicación.

OBJETIVOS ESPECÍFICOS:

- Utilizar LinearLayout como contenedor principal de la aplicación.
- Crear objetos Actividad:
 - Crear un TextView con texto

INTRODUCCION

Al iniciar un proyecto, Android Studio crea una serie de carpetas de las cuales, las que están en la carpeta res son las que usaremos en esta aplicación.



Comenzaremos con la carpeta de values, en ella estarán tres archivos xml como se ilustra en la figura anterior.

Vamos a crear una serie de colores que usaremos como recursos, la intención es que la aplicación tienda a ser de fácil mantenimiento.

El archivo colors.xml contiene en su inicio lo siguiente:

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <resources>
3      <color name="colorPrimary">#3F51B5</color>
4      <color name="colorPrimaryDark">#303F9F</color>
5      <color name="colorAccent">#FF4081</color>
6  </resources>
7
```

Son los colores básicos (pueden variar) dependiendo de la configuración de usted.

Programación de Dispositivos Móviles.

Delio Coss Camilo.

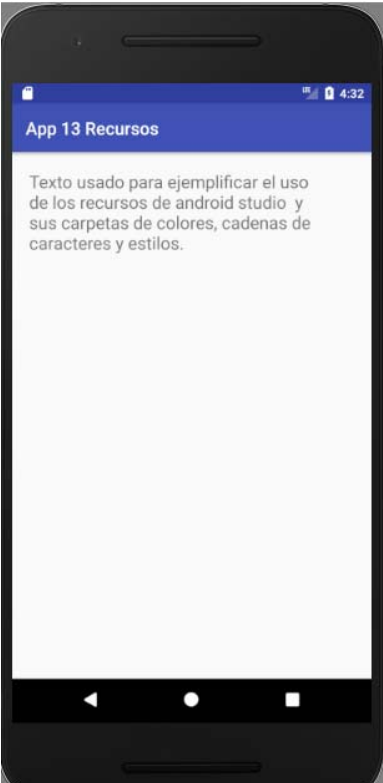
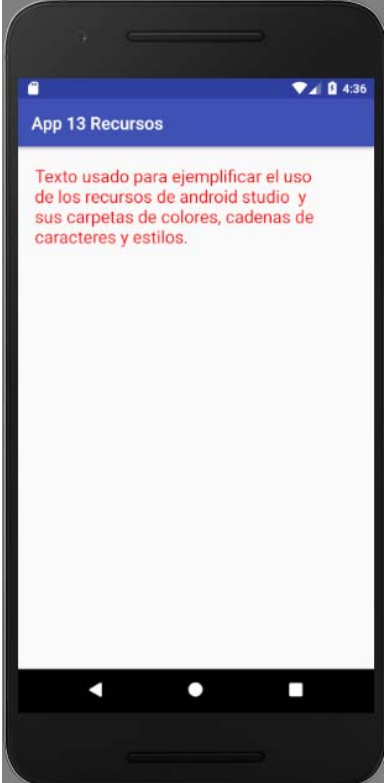
Vamos a crear tres colores, Rojo, Verde y Azul mediante el código:

```
// Esto son mis colores basicos
<color name="dccRojo">#f00</color>
<color name="dccVerde">#0f0</color>
<color name="dccAzul">#00f</color>
```

Ahora, vamos a cambiar el color del texto de un TextView aplicando este recurso.

```
<TextView
    android:id="@+id/txtMensaje"
    android:text="Texto usado para ejemplificar el uso de los recursos
de android studio
y sus carpetas de colores, cadenas de caracteres y estilos."
    android:layout_margin="20dp"
    android:textSize="20sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Si visualizamos la aplicación se verá:

Normal	Modificado
	

Podemos notar el color del texto en su color de origen, para cambiarlo debemos de aplicar el cambio al atributo correspondiente mediante

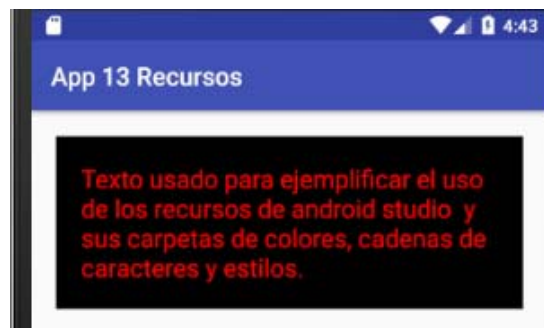
Programación de Dispositivos Móviles.
Delio Coss Camilo.

```
android:textColor="@color/dccRojo"
```

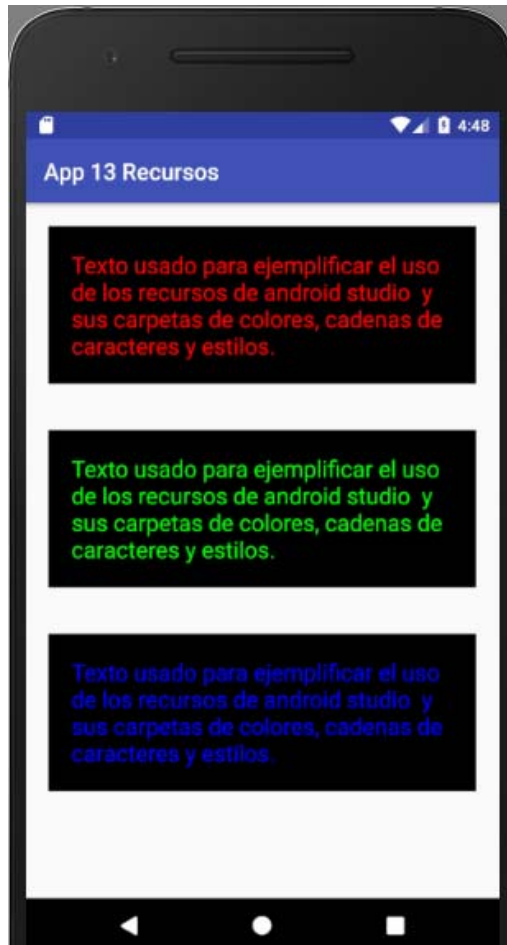
```
10 <TextView
11     android:id="@+id/txtMensaje"
12     android:text="Texto usado para ejemplificar el uso de los
13     y sus carpetas de colores, cadenas de caracteres y estilos."
14     android:layout_margin="20dp"
15     android:textSize="20sp"
16     android:layout_width="match_parent"
17     android:layout_height="wrap_content"
18     android:textColor="@color/dccRojo"
19 />
```

La definición del valor esta dado por el @color que le indica a Android que es un recurso de tipo color.

Genere el color negro mediante el siguiente código: #000 y luego genere un background al textview y de un padding de 20sp. Deberá de conseguir lo siguiente:



De esta forma, vamos creando los colores de nuestra aplicación y será mas simple su mantenimiento, pues si deseamos cambiar de colores nos dirigimos a este archivo XML y cambiamos el atributo. Agregue dos TextViews con colores verde y azul respectivamente con el mismo fondo negro, como se ilustra:

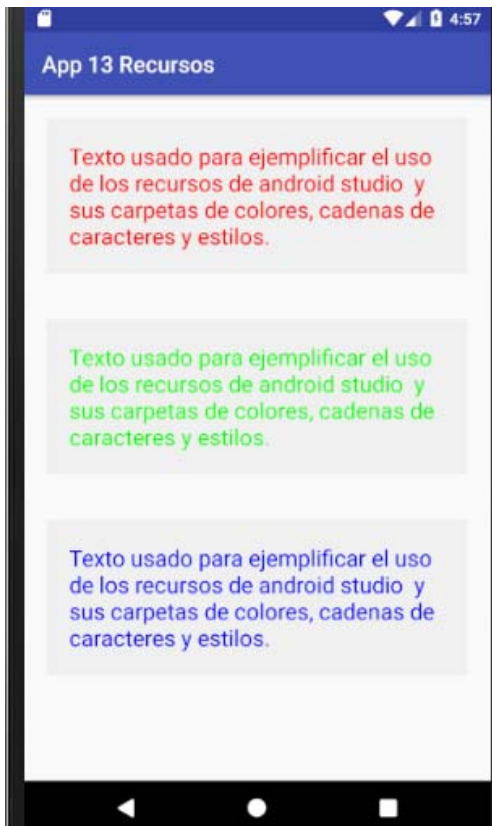


```
//Codigo para lograr lo anterior:
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

    // Esto son mis colores basicos
    <color name="dccRojo">#f00</color>
    <color name="dccVerde">#0f0</color>
    <color name="dccAzul">#00f</color>
    <color name="dccFondo">#000</color>

</resources>
```

Ahora para ver lo simple del mantenimiento, vamos a cambiar el color del background a gris, por lo tanto, creamos el color y luego solo cambiamos el atributo de #000 a #f0f0f0



Su código XML será:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color
name="colorPrimary">#3F51B5</color>
    <color
name="colorPrimaryDark">#303F9F</color>
    <color
name="colorAccent">#FF4081</color>

    // Esto son mis colores basicos
    <color name="dccRojo">#f00</color>
    <color name="dccVerde">#0f0</color>
    <color name="dccAzul">#00f</color>
    <color
name="dccFondo">#f0f0f0</color>
</resources>
```

Antes de crear una aplicación, será esencial definir los colores de cada elemento.

PRACTICA 12 Strings.

OBJETIVO GENERAL.

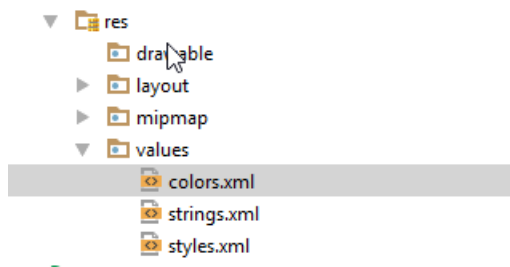
El alumno entenderá el uso adecuado de la carpeta de recursos values y archivo strings.xml de Android studio y aplicará su contenido a elementos de una aplicación.

OBJETIVOS ESPECÍFICOS:

- Utilizar LinearLayout como contenedor principal de la aplicación.
- Crear objetos Actividad:
 - Crear un TextView con texto

INTRODUCCION

Al iniciar un proyecto, Android Studio crea una serie de carpetas de las cuales, las que están en la carpeta res son las que usaremos en esta aplicación.



Comenzaremos con la carpeta de values, en ella estarán tres archivos xml como se ilustra en la figura anterior.

Vamos a crear una serie de colores que usaremos como recursos, la intención es que la aplicación tienda a ser de fácil mantenimiento.

El archivo strings.xml contiene en su inicio lo siguiente:

```
1 <resources>
2     <string name="app_name">App 13 Recursos</string>
3 </resources>
```

Son los textos de inicio de nuestra aplicación, este texto es el que se despliega en la barra de herramientas de nuestra aplicación.

Vamos a modificar la aplicación anterior, de tal forma que cambaremos la propiedad text de cada uno de los textviews por valores definidos en nuestro archivo strings.xml, como se indica:

```
<TextView
    android:id="@+id/txtMensaje"
    android:text="Texto usado para ejemplificar el uso de los recursos
de android studio
y sus carpetas de colores, cadenas de caracteres y estilos."
    android:layout_margin="20dp"
    android:textSize="20sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/dccRojo"
    android:padding="20dp"
    android:background="@color/dccFondo"
/>
```

Por

```
<TextView
    android:id="@+id/txtMensaje"
    android:text="@string/mensaje01"
    android:layout_margin="20dp"
    android:textSize="20sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/dccRojo"
    android:padding="20dp"
    android:background="@color/dccFondo"
/>
```

Y así con los otros dos, solo cambiando la propiedad:

`android:text="@string/mensaje02"` y `android:text="@string/mensaje02"`



Al igual, el mantenimiento de la aplicación se hace simple, pues solo modificaremos el valor del atributo de cada elemento.

Debemos de crear este tipo de práctica, pues si a futuro nuestra aplicación será en otros idiomas.

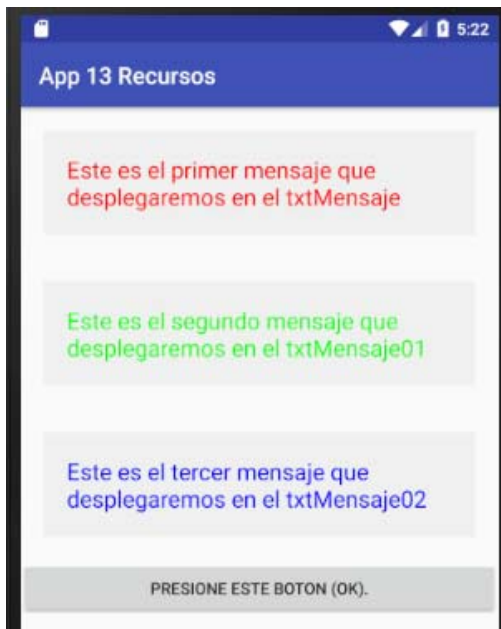
Todo elemento que tenga texto para ser desplegado, deberá el texto estar en estos archivos, por ejemplo, si tenemos un botón, tendríamos que crear el texto en el archivo strings.xml y luego asignarlo a al botón:

Archivo strings.xml agregamos:

```
<string name="btnMensaje">
    Presione este boton (OK).
</string>
```

En la aplicación agregamos:

```
<Button
    android:id="@+id/btnMensaje"
    android:text="@string/btnMensaje"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```



PRACTICA 12 Shapes.

OBJETIVO GENERAL.

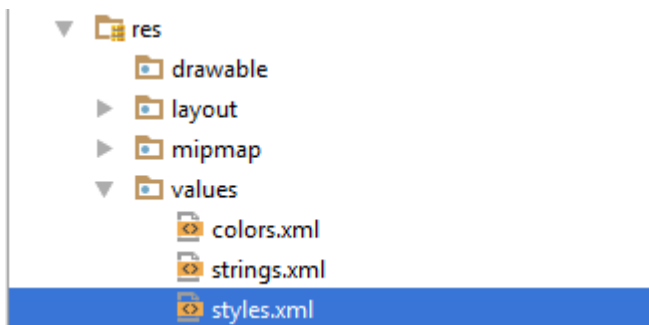
El alumno creará fondos personalizados para los elementos de su aplicación mediante el uso de archivos xml personalizados en la carpeta drawable

OBJETIVOS ESPECÍFICOS:

- Utilizar LinearLayout como contenedor principal de la aplicación.
- Crear objetos Actividad:
 - Crear dos TextViews con texto

INTRODUCCION

Al iniciar un proyecto, Android Studio crea una serie de carpetas de las cuales, las que están en la carpeta res son las que usaremos en esta aplicación.



Comenzaremos con la carpeta de values, en ella estarán tres archivos xml como se ilustra en la figura anterior.

Vamos a crear una serie de estilos que usaremos como recursos, la intención es que la aplicación tienda a ser de fácil mantenimiento.

El archivo styles.xml contiene en su inicio lo siguiente:

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

El diseño de la interfaz es la siguiente:

Son los textos de inicio de nuestra aplicación, este texto es el que se despliega en la barra de herramientas de nuestra aplicación.

Vamos a modificar la aplicación anterior, de tal forma que cambaremos la propiedad text de cada uno de los textviews por valores definidos en nuestro archivo strings.xml, como se indica:

```
<TextView
    android:id="@+id/txtMensaje"
    android:text="Texto usado para ejemplificar el uso de los recursos
de android studio
y sus carpetas de colores, cadenas de caracteres y estilos."
    android:layout_margin="20dp"
    android:textSize="20sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/dccRojo"
    android:padding="20dp"
    android:background="@color/dccFondo"
/>
```

Por

```
<TextView
    android:id="@+id/txtMensaje"
    android:text="@string/mensaje01"
    android:layout_margin="20dp"
    android:textSize="20sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/dccRojo"
    android:padding="20dp"
    android:background="@color/dccFondo"
/>
```

Y así con los otros dos, solo cambiando la propiedad:

```
android:text="@string/mensaje02" y android:text="@string/mensaje02"
```



Al igual, el mantenimiento de la aplicación se hace simple, pues solo modificaremos el valor del atributo de cada elemento.

Debemos de crear este tipo de práctica, pues si a futuro nuestra aplicación será en otros idiomas.

Todo elemento que tenga texto para ser desplegado, deberá el texto estar en estos archivos, por ejemplo, si tenemos un botón, tendríamos que crear el texto en el archivo strings.xml y luego asignarlo a al botón:

Archivo strings.xml agregamos:

```
<string name="btnMensaje">
    Presione este boton (OK).
</string>
```

En la aplicación agregamos:

```
<Button
    android:id="@+id/btnMensaje"
    android:text="@string/btnMensaje"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

