3. Elementos Visuales

Alberto Luengo

Elementos visuales

TextView: muestra texto al usuario

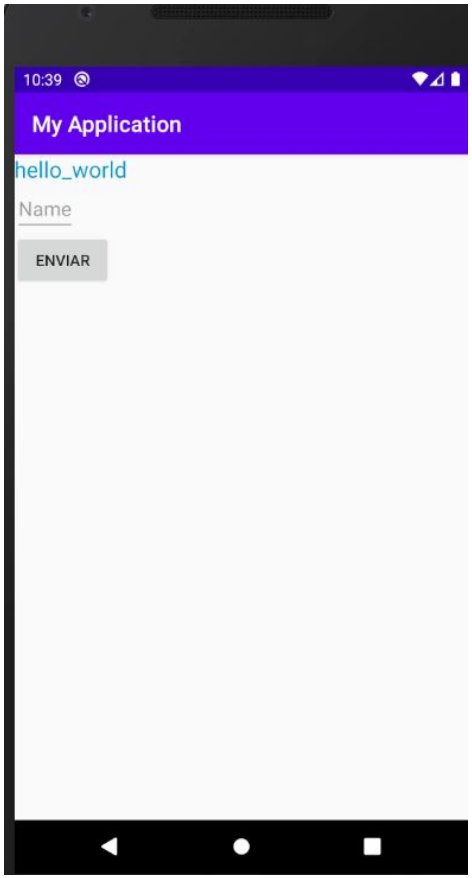
```
<TextView
    android:id="@+id/text_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="hello_world"
    android:textColor="@android:color/holo_blue_dark"
    android:textSize="20sp"/>
```

EditText: Permite al usuario editar un campo de texto

```
<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Name"
    android:inputType="textPersonName"/>
```

Button: control con texto o imagen que realiza una acción cuando el usuario lo presiona

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enviar" />
```



Layouts (Contenedores)

Si queremos combinar varios elementos de tipo vista tendremos que utilizar un objeto de tipo Layout. Es un contenedor que alberga varias vistas y controla su posición.

Hay que destacar que un Layout puede tener varios contenedores dentro y que es un descendiente de la clase ViewGroup.

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.
- **FrameLayout**: Permite el cambio dinámico de los elementos que contiene.
- **ConstraintLayout**: Versión mejorada de RelativeLayout, que permite una edición visual desde el editor y trabajar con porcentajes.

Contenedores: LinearLayout

LinearLayout: Alinea todos los campos secundarios en una única dirección, de manera vertical u horizontal. Puedes especificar la dirección del diseño con el atributo android:orientation.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
  <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Nombre" />
  <Button
      android:layout_width="100dp"
      android:layout_height="wrap_content"
      android:layout_gravity="right"
      android:text="Enviar" />
</LinearLayout>
```

Contenedores: RelativeLayout

RelativeLayout: Muestra vistas secundarias en posiciones relativas. La posición de cada vista puede especificarse como relativa a elementos hermanos (como a la izquierda o debajo de otra vista) o en posiciones relativas al área RelativeLayout principal (como alineada a la parte inferior, izquierda o centro).

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
  <EditText
      android:id="@+id/name"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Nombre" />
  <Button
      android:layout_width="96dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/name"
      android:layout_alignParentRight="true"
      android:text="OK" />
</RelativeLayout>
```

Contenedores: ConstraintLayout

ConstraintLayou: Permitirá simplificar las interfaces en anidamiento, para hacerlas lo más complejas posibles a nivel de diseño. Este layout, similar al RelativeLayout nos permitirá establecer relaciones entre todos los elementos y la propia vista padre, permitiendo así ser mucho más flexible que los demás.

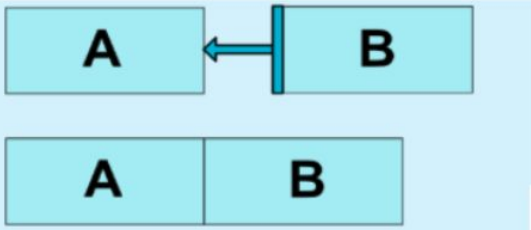


Fig. 1 - Relative Positioning Example

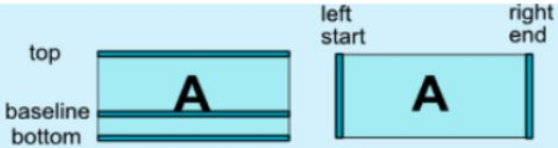


Fig. 2 - Relative Positioning Constraints

- `layout_constraintLeft_toLeftOf`
- `layout_constraintLeft_toRightOf`
- `layout_constraintRight_toLeftOf`
- `layout_constraintRight_toRightOf`
- `layout_constraintTop_toTopOf`
- `layout_constraintTop_toBottomOf`
- `layout_constraintBottom_toTopOf`
- `layout_constraintBottom_toBottomOf`
- `layout_constraintBaseline_toBaselineOf`
- `layout_constraintStart_toEndOf`
- `layout_constraintStart_toStartOf`
- `layout_constraintEnd_toStartOf`
- `layout_constraintEnd_toEndOf`

Scrolls

También podemos utilizar otras clases de Vistas, que son descritas a continuación:

- **ScrollView**: Visualiza una columna de elementos; cuando estos no caben en pantalla se permite un deslizamiento vertical.
- **NestedScrollView**: es lo mismo que el scrollview, pero soporta comportamientos anidados.
- **HorizontalScrollView**: Visualiza una fila de elementos; cuando estos no caben en pantalla se permite un deslizamiento horizontal.

Vinculación de vistas

1. Kotlin extensions
2. findViewById
3. Binding

Eventos y Referencias

Ids:

Para poder acceder en nuestro Activity a los elementos visuales que hemos creado en el layout(xml) todos tienen que tener un id referenciado de esta manera:

```
<Button
    android:id="@+id/btEnviar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Enviar" />
```

Ahora desde nuestro Activity podremos acceder a los datos que contiene o modificarlos:

Aquí podemos establecer lo que hará nuestro código una vez pulsado el botón con id BUTTON.

```
btEnviar.setOnClickListener {
//definir que haremos tras pulsar el boton
}
```

Eventos y Referencias

Acceder a elementos de las vistas:

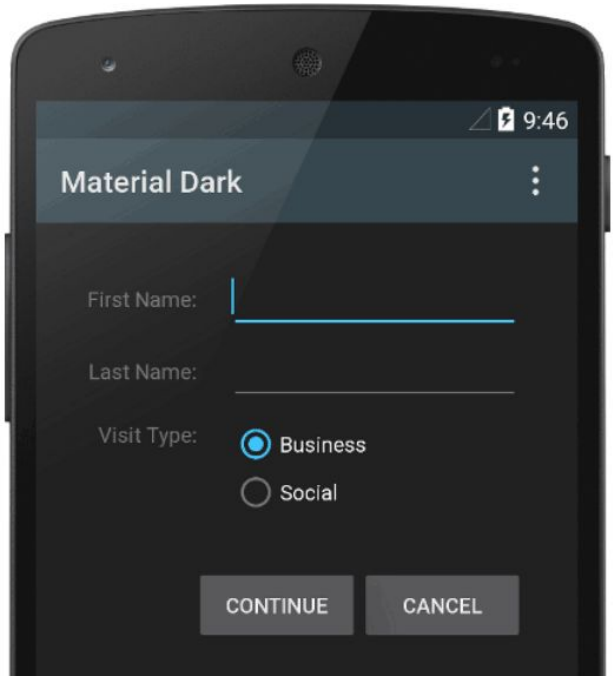
Si queremos acceder al texto introducido por el usuario en un EditText:

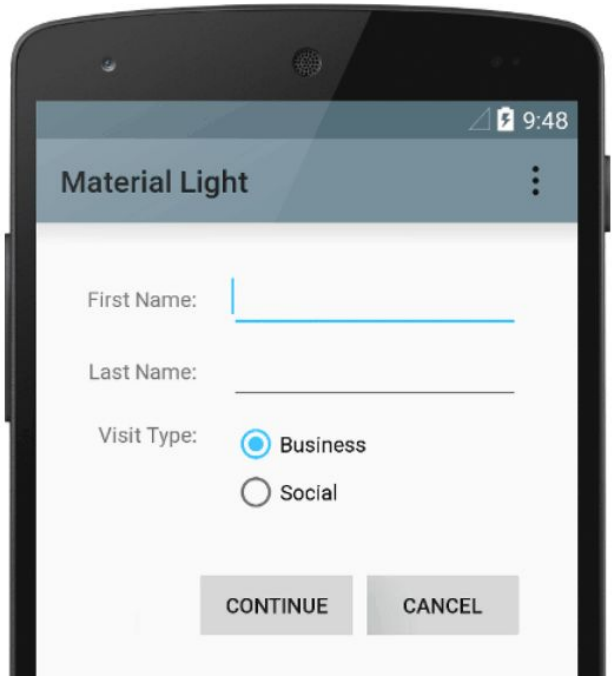
```
val texto = editText.text.toString()
```

Styles y Themes

Styles y Themes en android te permiten separar los detalles del diseño de tu app de la UI, similar a una hoja de estilos (CSS) en web.

Un style es una colección de atributos que especifican la apariencia de una vista. Un style puede cambiar los atributos de color, fuentes, tamaños, fondos y mucho más



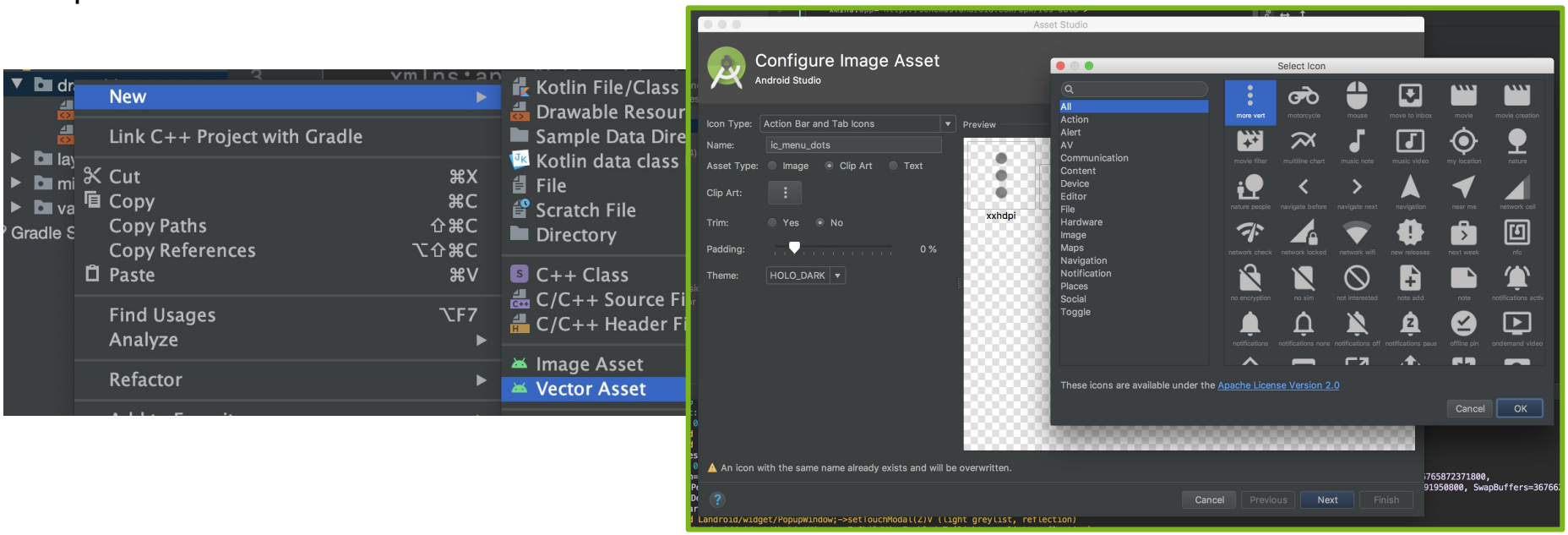


Drawables

Recursos de elementos de diseño

Un drawable es un tipo de recurso que puede ser dibujado en pantalla. Podremos utilizarlos para especificar el aspecto que van a tener los diferentes componentes de la interfaz, o partes de éstos. Estos drawables podrán ser definidos en XML o de forma programática.

En la carpeta drawable con el botón derecho podemos crear un icono para utilizar en la app o importarlo desde nuestro ordenador.

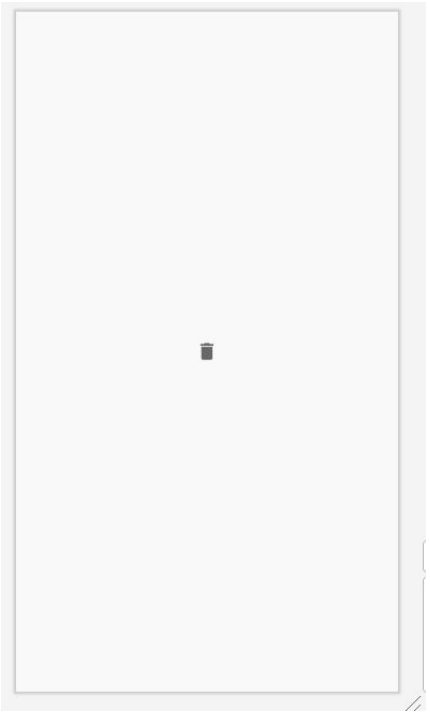


ImageView

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">

    <ImageView
        android:padding="8dp"
        android:clickable="true"
        android:background="?selectableItemBackgroundBorderless"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_baseline_delete_24" />

</LinearLayout>
```



Si queremos que se pueda pulsar usamos clickable = true y el background indicado en el xml

Weight

Podemos utilizar el peso en un componente para separarlo porcentualmente. Para hacer esto dentro de un LinearLayout utilizando el peso es así:

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="@color/colorPrimary"
        android:orientation="horizontal">

        <Button
            android:id="@+id/register"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:padding="10dip"
            android:text="Login" />

        <Button
            android:id="@+id/cancel"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:padding="10dip"
            android:text="Registro" />

    </LinearLayout>
```

Weight

Y jugando con los valores del peso obtenemos lo siguiente:

```
<LinearLayout
       android:layout_width="match_parent"
       android:layout_height="60dp"
       android:background="@color/colorPrimary"
       android:orientation="horizontal">

       <Button
           android:id="@+id/register"
           android:layout_width="0dp"
           android:layout_height="wrap_content"
           android:layout_weight="0.3"
           android:padding="10dip"
           android:text="Login" />

       <Button
           android:id="@+id/cancel"
           android:layout_width="0dp"
           android:layout_height="wrap_content"
           android:layout_weight="0.7"
           android:padding="10dip"
           android:text="Registro" />

   </LinearLayout>
```

Espacios

margin - Distancia entre los 4 lados

marginLeft - Distancia en el lado izquierdo

marginRight - Distancia en el lado derechpo

marginTop - Distancia en el lado de arriba

marginBottom - Distancia en el lado de abajo

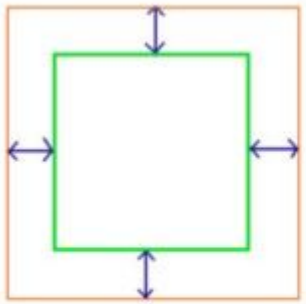
padding - Distancia entre los 4 lados desde dentro

paddingLeft - Distancia en el lado izquierdo desde dentro

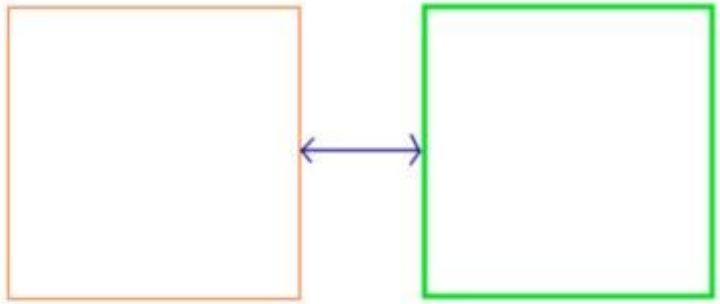
paddingRight - Distancia en el lado derechpo desde dentro

paddingTop - Distancia en el lado de arriba desde dentro

paddingBottom - Distancia en el lado de abajo desde dentro



Padding



Margin

Espacios

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:layout_margin="40dp"
        android:text="Text 1" />

    <LinearLayout
        android:orientation="vertical"
        android:id="@+id/layout1"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginBottom="10dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Margen en los 4 lados de **40dp**

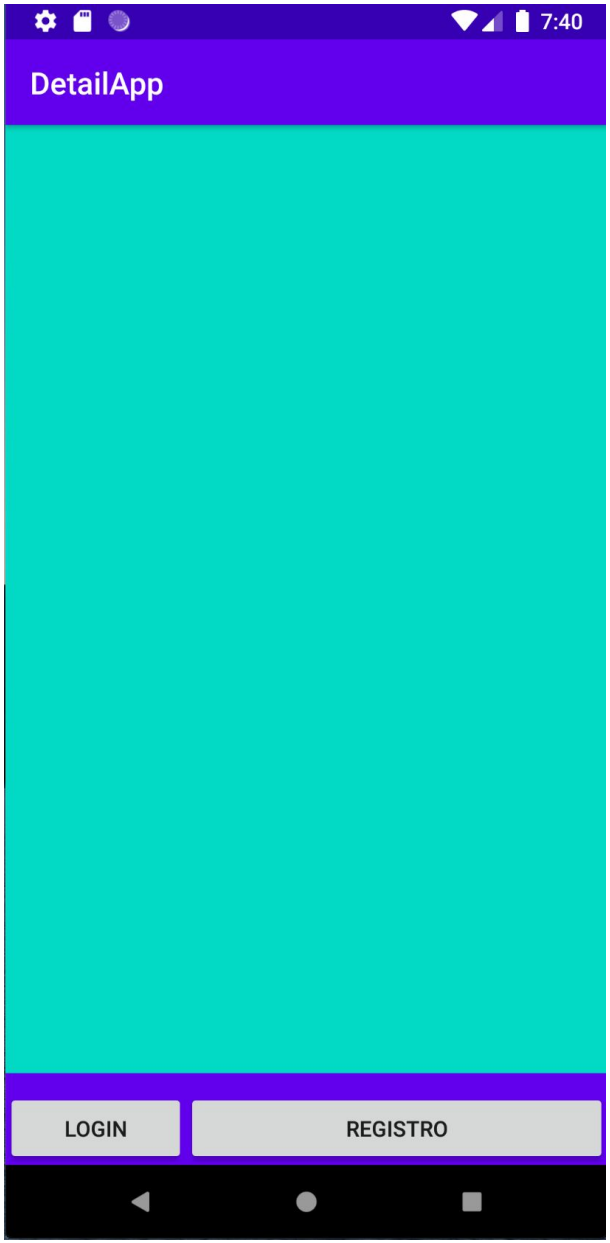
Margen izquierdo, derecha y abajo de **10dp**

Mover contenedores por la vista

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/colorAccent"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_alignParentBottom="true"
        android:background="@color/colorPrimary"
        android:gravity="bottom"
        android:orientation="horizontal">

        <Button
            android:id="@+id/register"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.3"
            android:padding="10dip"
            android:text="Login" />
        <Button
            android:id="@+id/cancel"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.7"
            android:padding="10dip"
            android:text="Registro" />

    </LinearLayout>
</RelativeLayout>
```



Resources

Los resources mas usados son los siguientes y se encuentran en la carpeta **values**

strings.xml / strings-es.xml

```
<string name="app_name">My Application</string>
<string name="title_activity_main2">MainActivity</string>
```

colors.xml

```
<color name="colorPrimary">#6200EE</color>
<color name="colorPrimaryDark">#3700B3</color>
<color name="colorAccent">#03DAC5</color>
```

dimens.xml

```
<dimen name="horizontal_margin">16dp</dimen>
```

styles.xml

```
<!-- Base application theme. -->
<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>
```

Botones de Material Design

También tenemos los recursos oficiales aquí, aunque pueden ser complicados de entender a la primera. Hay que importar siempre esta librería (con su correspondiente versión):

```
implementation 'com.google.android.material:material:1.6.0'
```

Y en styles.xml cambiamos el theme de la app:

```
<resources>

  <style name="AppTheme" parent="Theme.MaterialComponents.DayNight.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>
```

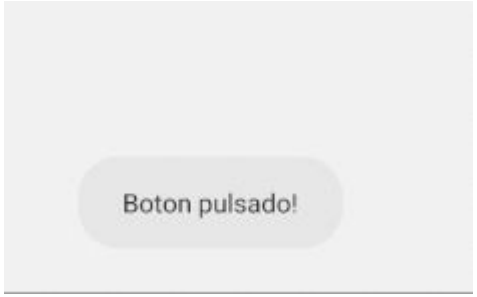

```
<Button
    android:id="@+id/textButton"
    android:layout_margin="30dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@android:color/white"
    android:backgroundTint="@color/colorPrimaryDark"
    android:text="Text button"
    style="@style/Widget.MaterialComponents.Button.TextButton"
    />
```

Mostrar mensajes al usuario

Toast:

Muestra feedback sobre alguna operación realizada. Idealmente debe mostrar mensajes cortos. Aparece por un periodo corto o largo.

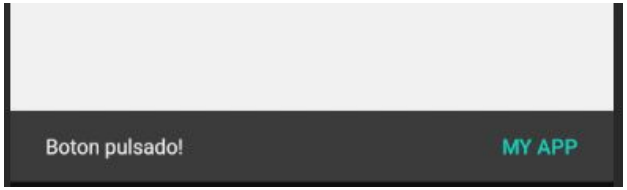
```
Toast.makeText(this, "Boton pulsado!", Toast.LENGTH_SHORT).show()
```



Snackbar:

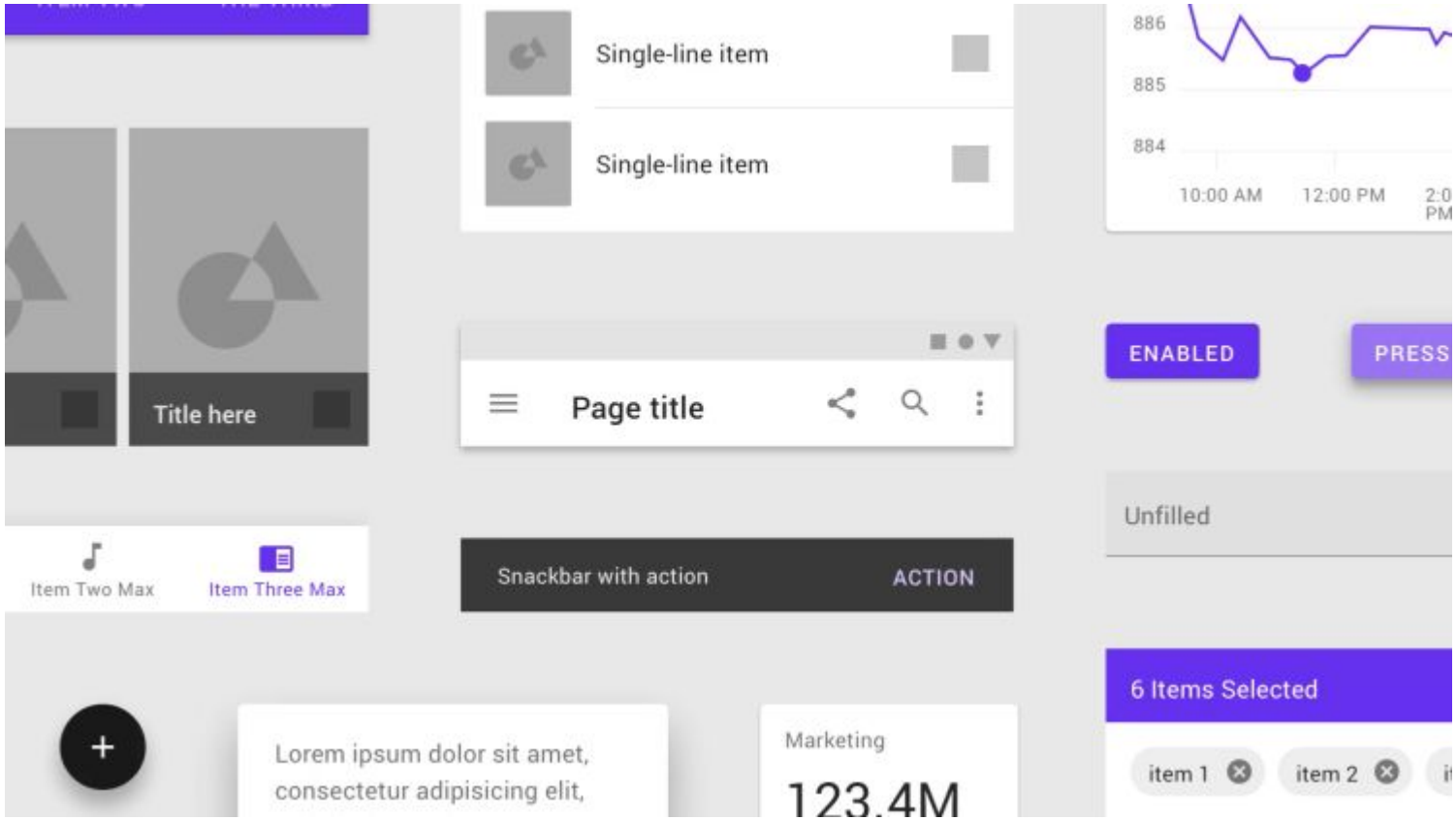
Muestra feedback sobre alguna operación realizada, y es similar a un toast aunque podemos hacer que no desaparezca tras un periodo de tiempo. Hay que usar una vista para poder llamarlo, puede ser el contenedor del activity o cualquier otra.
Puede contener un botón de texto de acción

```
Snackbar.make(view, "Boton pulsado!", Snackbar.LENGTH_SHORT).show()
```



Diseño en Android

Material design contiene la mayoría de componentes nativos de android con los que están hechas todas las apps y una guia de estilos.



Ejercicios

Ejercicio 1

En este ejercicio tenemos que:

1. añadir un botón (Button) debajo del TextView
2. asignar un id al TextView
3. asignar un id al Button
4. Al pulsar el botón cambiar el valor del texto

Recomendación:
Los mejores programadores aprenden a buscar las soluciones, no hace falta hacer nada de memoria ¡y menos al principio! 😄

Ejercicio 2

1. Con la misma aplicación, crear un contador, que cada vez que se pulse sobre el botón incremente su valor en n + 1 mostrandolo por un TextView.

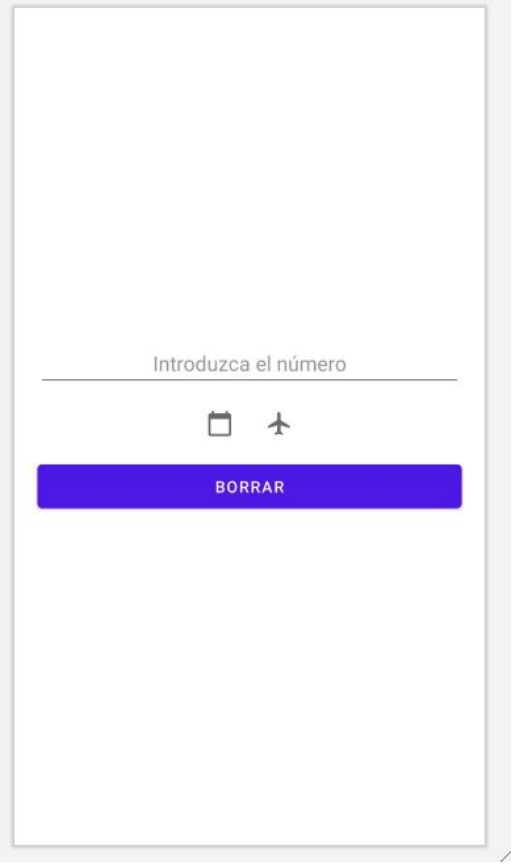
2. Añadir a la vista actual un EditText y un Button, y que cuando se pulse el Button, se actualice el TextView con el texto del EditText.

3. Crear otro TextView debajo del actual y que cada vez que se escriba se escriba en el EditText que se actualice con el mismo texto. Pista, el evento se llama **afterTextChanged**

Ejercicios

1. Añadir un EditText que solo acepte numeros con decimales
2. Añadir una imagen con el icono de un **calendario,** el area de pulsado debe ser mayor que el tamaño de la imagen
3. Añadir una imagen con el icono de un **avion,** el area de pulsado debe ser mayor que el tamaño de la imagen
4. Al pulsar en el calendario que muestre un Snackbar con el texto introducido en el EditText
5. Al pulsar en el avion que muestre un Toast con el texto introducido en el EditText
6. Añade un botón que borre el contenido del EditText

*Los textos, colores y dimensiones deberán estar en sus respectivos archivos de **res/values***




La excusa No.1 de los programadores para hacer de vagos:
HEY! VUELVAN A TRABAJAR!
Está compilando
Oh. Ok, continúen.

Gracias.