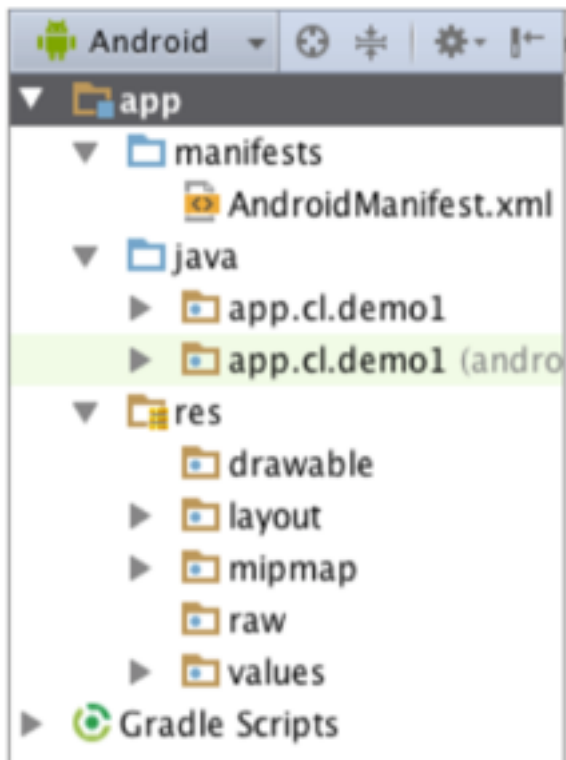


## RESUMEN DIA 01

### ANDROID ESTRUCTURA DEL PROYECTO



**AndroidManifest:** Es un archivo XML que contiene nodos descriptivos sobre las características de una aplicación Android. Algunas características son la declaración de permisos para ejecutar algunos servicios como la declaración de la versión del SDK,

**java/MainActivity:** Clase Java que representa la funcionalidad de una actividad, en donde se pueden programar los diferentes estados de una actividad, junto con la funcionalidad, eventos, e inicialización de componentes definidos en un layout.

**res/drawable:** Directorio en el cual se pueden almacenar objetos del tipo imagen, ya sea para definir una imagen de fondo o un icono de un botón, etc.

**res/raw:** Directorio en el cual se pueden almacenar objetos del tipo audio como por ejemplo un MP3.

**res/mipmap:** Directorio en el cual se pueden almacenar imágenes para establecer el icono de la aplicación, la cual es

definida en el archivo AndroidManifest.

**layout:** Archivo XML asociado a un MainActivity, el cual define la interfaz de usuario, posee diversos

controles, dentro de los cuales se destacan **layouts**, **controles de formulario**, etc.

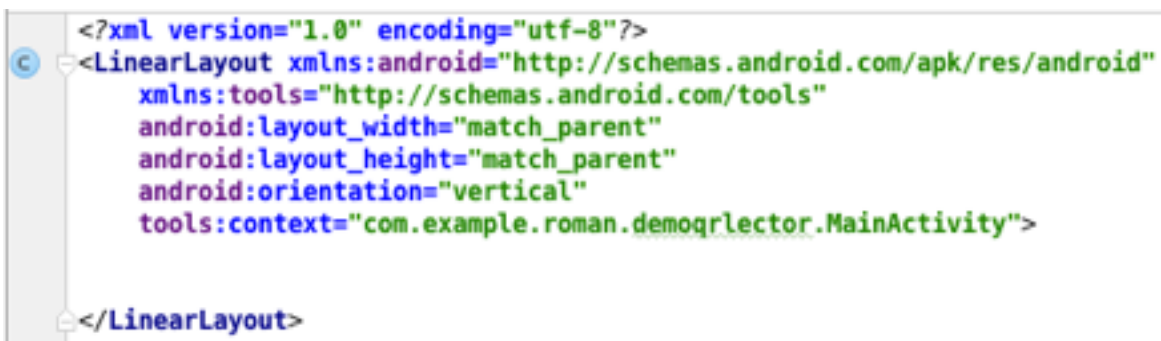
**values:** Directorio en donde se definen los recursos de tipo string, color, y styles de la aplicación.

## CONTROLES EN EL LAYOUT

El layout es un archivo con extensión xml en el cual se puede programar la interfaz de la aplicación móvil, en el cual se presentan los controles básicos junto con sus propiedades más importantes.

### LinearLayout

Es un control del tipo contenedor el cual permite distribuir los controles de un formulario de dos maneras vertical u horizontal. Un ejemplo se puede ver a continuación:



```
<?xml version="1.0" encoding="utf-8"?>
<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:orientation="vertical"
    tools:context="com.example.roman.demogrlector.MainActivity">

</LinearLayout>
```

Propiedades: todos los controles tienen propiedades que son importantes, dentro de las cuales para LinearLayout podemos definir.

**layout\_width:** corresponde al ancho del control que puede ser `wrap_content` (ajustado a la dimensión del control) o `match_parent` (ocupando todo el ancho de la pantalla)

**layout\_height:** corresponde al alto del control pudiendo ser `wrap_content` o `match_parent`.

**orientation:** corresponde a la orientación de como se van a distribuir los controles, pudiendo ser horizontal o vertical, si no se especifica la propiedad queda por defecto como horizontal. Como recomendación siempre un LinearLayout debería tener definida su orientación.

**context:** indica la clase a la cual está asociado el archivo xml, para poder identificar donde se debe programar los eventos de la aplicación. Esta propiedad se crea por defecto.

## TextView

Es un control que representa a una etiqueta o label, el cual puede servir como control informativo o para registrar algún mensaje como resultado.

```
<TextView
    android:id="@+id/scan_format"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="mensaje"
    android:textSize="13sp"
/>
```

algunas de sus propiedades básicas son:

**id:** representa el nombre que tendrá el control (nombre de variable), el cual es importante si se quiere utilizar dentro de la programación, el id siempre debe ir con la estructura del tipo "@+id/nombreVariable", y es opcional, salvo para los controles que capturan información dentro de la app.

**text:** propiedad opcional, la cual representa el texto que va a aparecer en la interfaz.

**textSize:** propiedad opcional que indica el tamaño del texto, la unidad recomendada para los textos en una aplicación móvil viene dada por sp (scale independent-pixel) que se ajusta a la densidad de la pantalla.

## EditText

Corresponde a un control de entrada de texto, el cual es utilizado para capturar datos de un formulario.

```
<EditText
    android:id="@+id/txt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="text"
    android:textSize="12sp"
    android:hint="ingrese nombre"
/>
```

algunas de sus propiedades básicas son:

**inputType:** corresponde al tipo de dato que acepta, y junto con esto también se desplegará un teclado acorde al tipo seleccionado pudiendo ser: text, number, date, numberDecimal, numberPassword, phone, textEmailAdress entre otros.

**hint:** representa a un placeHolder, el cual corresponde a un texto que da una pista de lo que se puede ingresar en el control.

## Button

Corresponde a un botón el cual el único propósito que tiene es dar funcionalidad a una aplicación.

```
<Button
    android:id="@+id/bt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="aceptar"
    android:onClick="scanner"
/>

public void scanner(View view){
    //evento
}
```

algunas de sus propiedades básicas son

**onClick:** hace referencia a un método llamado scan definido en una clase a la cual esta asociado el archivo xml (context),

## MainActivity

Corresponde a una clase la cual esta directamente asociada al archivo xml, en donde se diseña la interfaz, el objetivo de esta clase es poder capturar los controles necesarios para cumplir con la funcionalidad deseada.

Esta clase se puede definir por partes como indica la figura a continuación:

```
public class MainActivity extends AppCompatActivity {  
  
    //DECLARACION DE CONTROLES  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        //CAPTURA DE LOS CONTROLES DESDE EL ARCHIVO XML  
  
    }  
  
    //EVENTOS  
    public void evento1(View view){  
  
    }  
  
}
```

**Declaración de controles:** se pueden declara los controles que se desean utilizar para capturar sus datos o para cargar datos en ellos, pudiendo ser EditText o TextView. Ejemplo:

```
EditText txt;  
TextView label1;
```

**Captura de los controles:** android tiene un método que permite rescatar u obtener los controles definidos en el archivo xml, para ello es necesario que estén declarados como se explica en el concepto anterior, ejemplo:

```
txt = (EditText) findViewById(R.id.txt);
```

donde R corresponde a una clase dentro de android la cual en esta clase están definidos todos los controles que se han creado dentro del proyecto, principalmente incluyendo su ID.

Eventos: dentro del evento, se debe capturar o modificar los valores que poseen los controles obtenidos en el concepto definido anteriormente, por lo tanto todos los controles poseen dos propiedades importantes para editar su valor o modificar su valor. Ejemplo:

```
//MODIFICANDO UN CONTROL  
txt.setText("mensaje nuevo");
```

```
//OBTENIENDO EL VALOR DEL CONTROL  
String s = txt.getText().toString();
```

## Convenciones en Java String a Int - String a Double

En java podemos transformar los datos capturados a un valor numérico por medio de los métodos `parseInt` y `parseDouble` como muestra el siguiente ejemplo

```
int i;  
double j;  
try{  
    i = Integer.parseInt(txt.getText().toString());  
    //MAS CODIGO  
}catch(NumberFormatException e){  
    Toast.makeText(this, "error", Toast.LENGTH_SHORT).show();  
}
```

el bloque `try` analiza si la conversión esta correcta, es decir, el texto sólo tiene números, si el texto no tiene números se ejecuta el bloque `catch`, el cual lanza un mensaje de aviso al usuario.

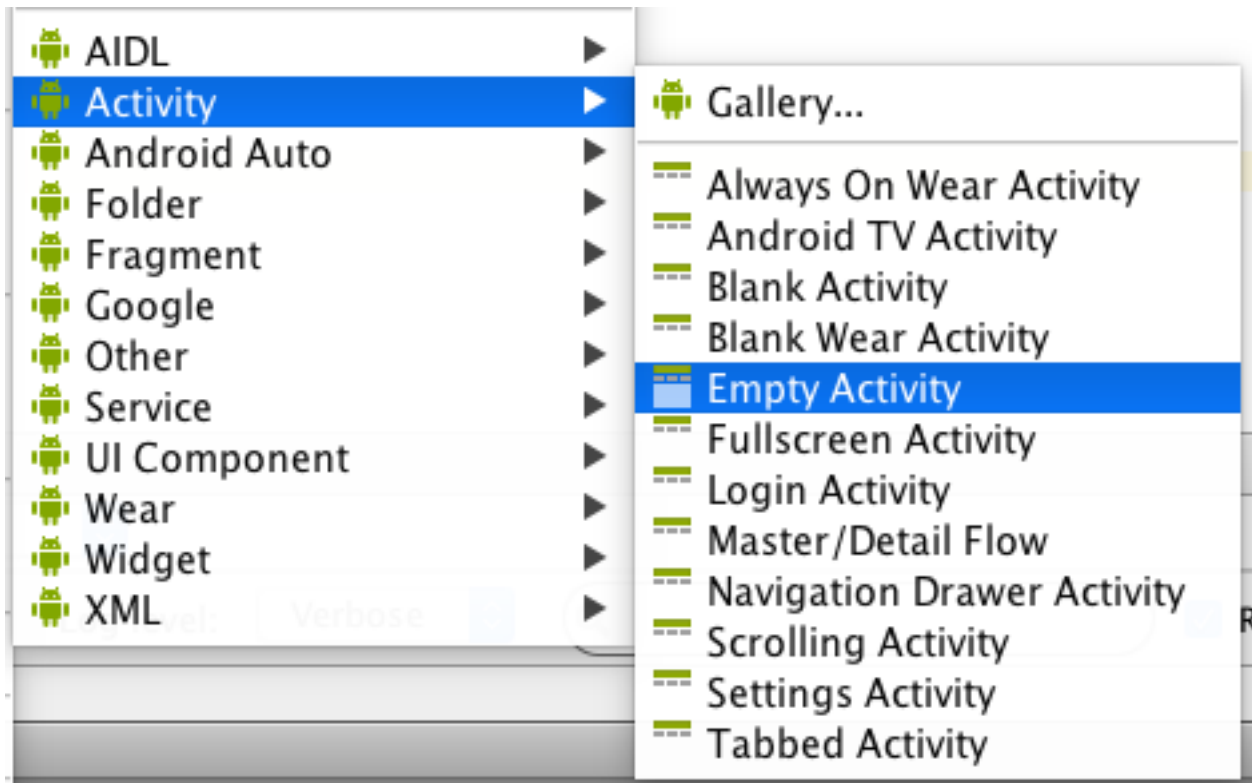
Para el caso de los decimales, es necesario tener el mismo bloque `try-catch` pero la conversión cambia, ejemplo:

```
int i;  
double j;  
try{  
    j = Double.parseDouble(txt.getText().toString());  
    //MAS CODIGO  
}catch(NumberFormatException e){  
    Toast.makeText(this, "error", Toast.LENGTH_SHORT).show();  
}
```

## Ir de un Activity a Otro

Cuando creamos un nuevo Activity dentro de la categoría java podemos enlazar a estos activities, es decir ir desde un activity principal a otro. Para crear este activity debe ser por medio de la siguiente instrucción

clic derecho en el dominio (package del activity)->new->Activity->EmptyActivity



Para ir a otro activity se puede hacer por medio de un evento de botón y la manera como se llama se representa por el siguiente ejemplo:

```
public void eventoIrOtroActivity(View view){  
    Intent i = new Intent(this, BuscarPersona.class);  
    startActivity(i);  
}
```

Un Intent sirve para invocar componentes (Activities), el cual posee un constructor, en donde se especifica cual es el contenedor de donde nace el nuevo activity, y la estructura del nuevo activity.