

Práctica 2

Sistemas Operativos Multimedia

Introducción a Android



1. ¿Qué es Android?

1.1. El sistema operativo Android

Android es un sistema operativo basado en Linux con una interfaz de programación (API) en Java. El sistema nos ofrece una serie de herramientas: compilador, depurador, emuladores de dispositivos así como su propia máquina virtual de Java, conocida como Dalvik Virtual Machine – DVM. Esta máquina virtual implementa el modelo de concurrencia de Java, utilizando objetos compartidos y monitores. Para la implementación de hilos, sigue el modelo definido en el estándar POSIX de pthreads, dónde los hilos se encuentran separados en grupos y las aplicaciones pueden lanzar múltiples hilos. Como veremos a continuación, la planificación y el manejo de interrupciones se delegan al Kernel de Linux.

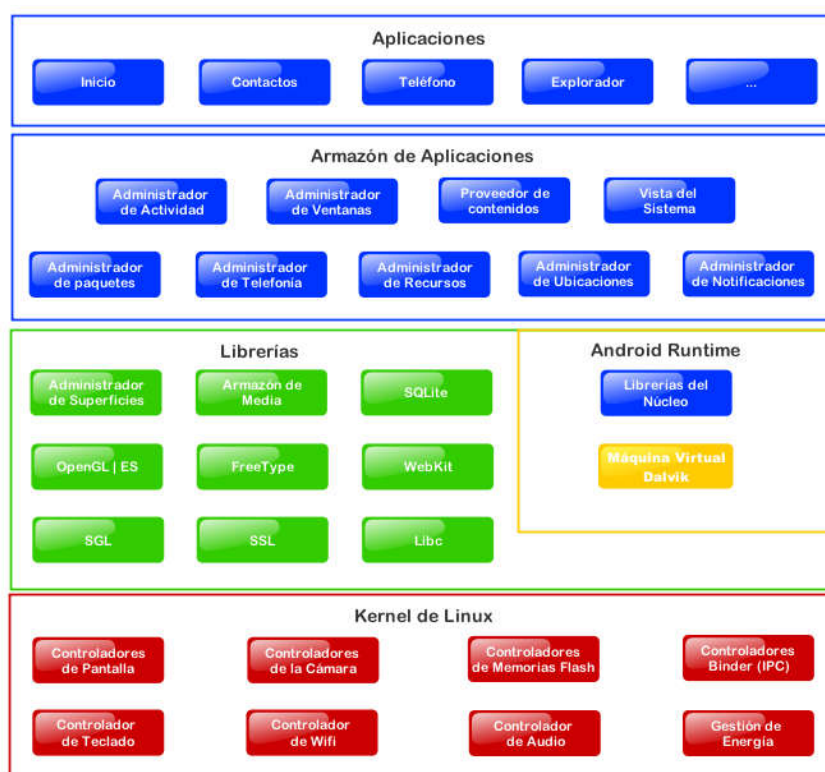


Ilustración 1. Componentes sistema operativo Android

Android ofrece soporte para gráficos 2D y 3D utilizando librerías OpenGL y además nos ofrece soporte de almacenamiento a través de una base de datos SQLite.

Cada aplicación desarrollada en Android se ejecuta sobre su propio proceso y sobre su propio usuario, el cual es generado por el sistema operativo durante el despliegue de la aplicación. Además la aplicación se encuentra aislada de otras aplicaciones que se encuentren en ejecución de forma que las aplicaciones no puedan fácilmente perjudicarse entre ellas.

1.2. Principales componentes de una aplicación Android

Una aplicación en Android está compuesta en parte por los siguientes elementos:

Activities: representan la capa de presentación de una aplicación en Android, por ejemplo, la pantalla que el usuario ve. Una aplicación Android puede tener varias actividades y se puede cambiar entre ellas durante la ejecución de la aplicación.

Views: Las interfaces de usuario de las actividades son construidas con clases de widgets, las cuales heredan de "android.view.View". El diseño de las vistas es gestionado por "android.view.ViewGroups". Las vistas a su vez poseen una serie de atributos, los cuales pueden ser usados para cambiar su apariencia y comportamiento.

Services: ejecutan tareas en segundo plano. Utilizan el framework de notificaciones de Android.

Content Providers: capa de datos para las aplicaciones. A través del proveedor de contenidos tu aplicación puede compartir información con otras aplicaciones. Los dispositivos Android incluyen de serie un conjunto de proveedores de contenidos nativos que permiten acceder a datos del terminal, como por ejemplo los contactos o el contenido multimedia.

Intents: mensajes asíncronos, los cuales permiten a la aplicación solicitar funcionalidades de otros servicios o aplicaciones. Una aplicación puede llamar directamente a un servicio o actividad, o incluso preguntar al sistema Android sobre los servicios registrados y aplicaciones. Las aplicaciones se registran a sí mismas a través del IntentFilter.

Receivers: recibe mensajes del sistema e Intents implícitos, puede ser utilizado para reaccionar ante ciertas condiciones del sistema. Una aplicación puede registrarse como receptor de broadcast para algunos eventos y puede comenzar ciertas tareas en caso de que algún evento predefinido ocurra.

Widgets: Componentes interactivos usados principalmente en la pantalla principal o "escritorio" Android para mostrar datos sin necesidad de lanzar la aplicación correspondiente

1.3. Seguridad y permisos

Una aplicación en Android utiliza el fichero “AndroidManifest.xml” para definirse a sí misma, especificando en este fichero atributos para su configuración y ejecución. Este fichero debe especificar también los permisos requeridos para la aplicación. Por ejemplo, si la aplicación requiere acceso a la red debe ser especificado en el fichero. El fichero “AndroidManifest.xml” puede ser visto como un descriptor de despliegue para una aplicación Android.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.eps.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

El atributo “**package**” del AndroidManifest.xml define el paquete base para los siguientes elementos. Este además debe ser único ya que el Market de Android solo permite definir una aplicación bajo el mismo paquete. Es un buen hábito usar el nombre de tu dominio al revés para evitar colisiones con otros desarrolladores, por ejemplo: es.ua.multimedia.android.nombreApp

“android:versionName” y “android:versionCode” especifican la versión de tu aplicación. “versionName” es lo que el usuario ve y puede ser cualquier tipo de String. “versionCode” debe ser un entero y el Market de Android lo utiliza para determinar si hay una nueva versión de la aplicación y en ese caso solicitar la actualización en los dispositivos donde la aplicación se encuentre instalada. Normalmente se empieza con un valor “1” y conforme se van sacando nuevas versiones se incrementa este valor.

“**activity**” define una actividad en nuestra aplicación, esta se encuentra asociada a una clase Java. En el caso de nuestro AndroidManifest.xml, apunta a la clase: es.ua.multimedia.so.android.calculadoraActivity

Un “filter intent” es registrado también para la clase anterior, de forma que especifica que la clase se ejecuta una vez que la aplicación ha sido ejecutada. (action android:name=”android.intent.action.MAIN”). La definición de “category” especifica que la aplicación se añade al directorio de aplicaciones del dispositivo (category android:name=”android.intent.category.LAUNCHER”).

El “uses-sdk” define la versión mínima del sdk para el cual tu aplicación es válida. Esto previene que la aplicación sea instalada en dispositivos con versiones antiguas del sdk.

1.4. R.Java, Resources y Assets

El directorio “gen” en un proyecto Android contiene valores generados. “R.java” es una clase generada la cual contiene referencias hacia recursos de la carpeta del proyecto

“res”. Estos recursos son definidos en el directorio “res” y pueden ser valores, iconos, imágenes, animaciones, diseños, etcétera. Por ejemplo un recurso puede ser una imagen o un fichero Xml donde se definen mensajes para la aplicación.

Si creamos un nuevo recurso, la correspondiente referencia es automáticamente creada en “R.java”. Estas referencias son valores int estáticos, el sistema operativo Android provee métodos para acceder al recurso necesario. Por ejemplo para acceder a un String con el id de referencia “R.string.tuString” utilizamos el método `getString(R.string.tuString)`; Es recomendable no modificar manualmente la clase R.java.

1.5. Activities y Layouts

Las interfaces de usuario para una “Activity” son definidas a través de layouts. En tiempo de ejecución los layout son instancias de “android.view.ViewGroups”. El layout define los elementos de la UI, sus propiedades y su disposición.

Los elementos de la UI están basados en la clase “android.view.View”. ViewGroup es una subclase de la clase “View” y un layout puede contener componentes UI (Views) u otros layouts (ViewGroups). No se deben anidar demasiados ViewGroups ya que esto tiene un impacto negativo sobre el rendimiento de la aplicación.

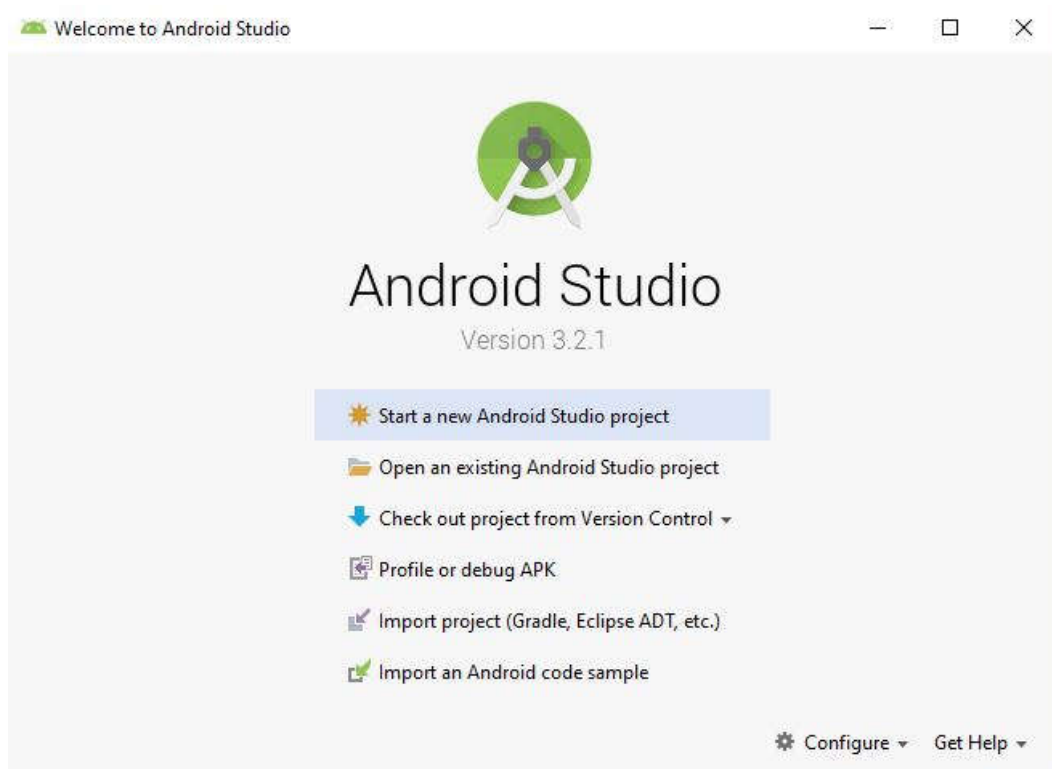
Un layout puede ser definido a través de código Java o vía XML. Los layout definidos usando XML son definidos a través de un fichero recurso en la carpeta “/res/layout”.

Este fichero especifica los ViewGroup, Views, sus relaciones y sus atributos. En el código Java estos elementos de la UI son accesibles a través del método `findViewById(R.id.tuvalor)`;

La definición de los layout en ficheros XML es la opción más recomendable debido a que nos permite desacoplar la lógica de programación de la definición del diseño de la aplicación.

2. Entorno de desarrollo.

El entorno de desarrollo utilizado en dicha práctica será Android Studio 3.2.1. En esta sección, describiremos paso a paso como realizar nuestra primera aplicación Android y para ello arrancamos Android Studio que nos mostrará la siguiente pantalla.



Seleccionaremos Star a new Android Studio project y pasamos a la siguiente pantalla en la que debemos indicar el nombre de la aplicación. En un principio dejaremos el que viene por defecto.

Create New Project

Create Android Project

Application name
My Application

Company domain
eps.example.com

Project location
C:\Users\EPS\AndroidStudioProjects\MyApplication2 ...

Package name
com.example.eps.myapplication Edit

☐ Include C++ support

☐ Include Kotlin support

Previous Next Cancel Finish

La siguiente pantalla nos va a permitir indicar el dispositivo Android para el que vamos a desarrollar la aplicación y la API mínima con la que será compatible. Por defecto se deja como esta (Tablets y Teléfonos) y pulsamos Next.

Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ **Phone and Tablet**

API 15: Android 4.0.3 (IceCreamSandwich)

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ **Wear**

API 21: Android 5.0 (Lollipop)

☐ **TV**

API 21: Android 5.0 (Lollipop)

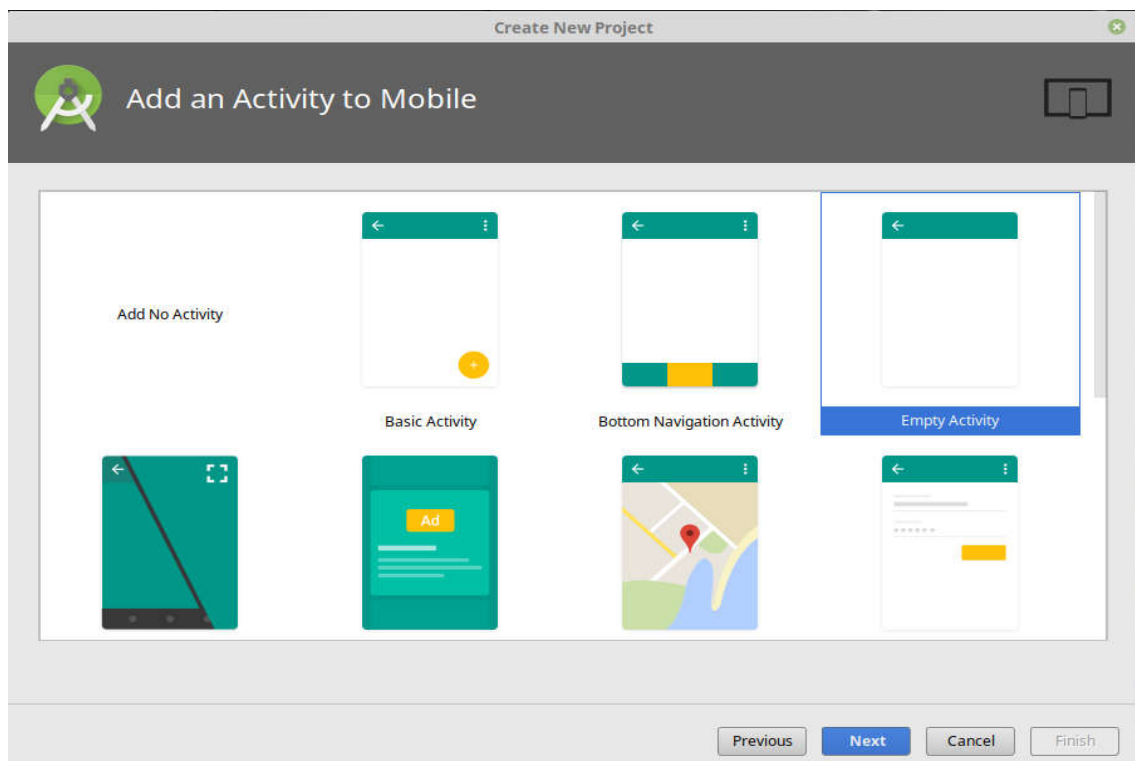
☐ **Android Auto**

☐ **Android Things**

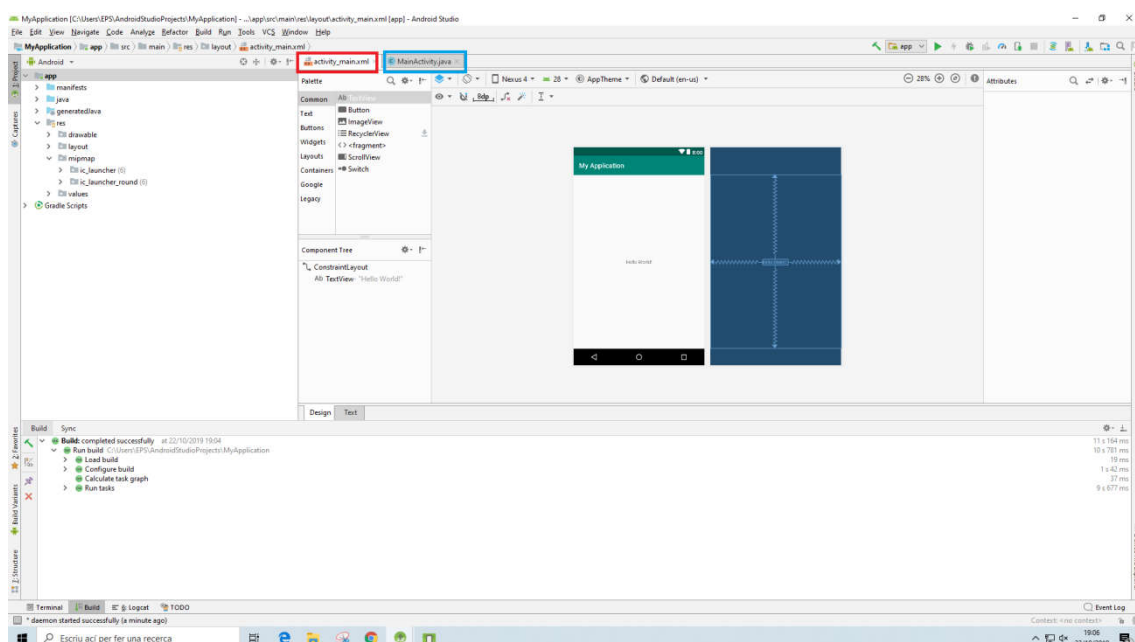
API 24: Android 7.0 (Nougat)

Previous Next Cancel Finish

A continuación debemos de escoger el tipo de actividad y de las diferentes opciones escogemos Empty Activity y pulsamos Next.



Llegados a este punto ya tenemos nuestro entorno de trabajo preparado para ir desarrollando nuestra aplicación. Trabajaremos con las dos pestañas remarcadas en la siguiente imagen de modo que en MainActivity.java (azul) escribiremos el código que determina el funcionamiento de nuestra aplicación y en activity_main.xml (rojo) escribiremos el código que determina el aspecto visual de la aplicación con la que vamos a trabajar.



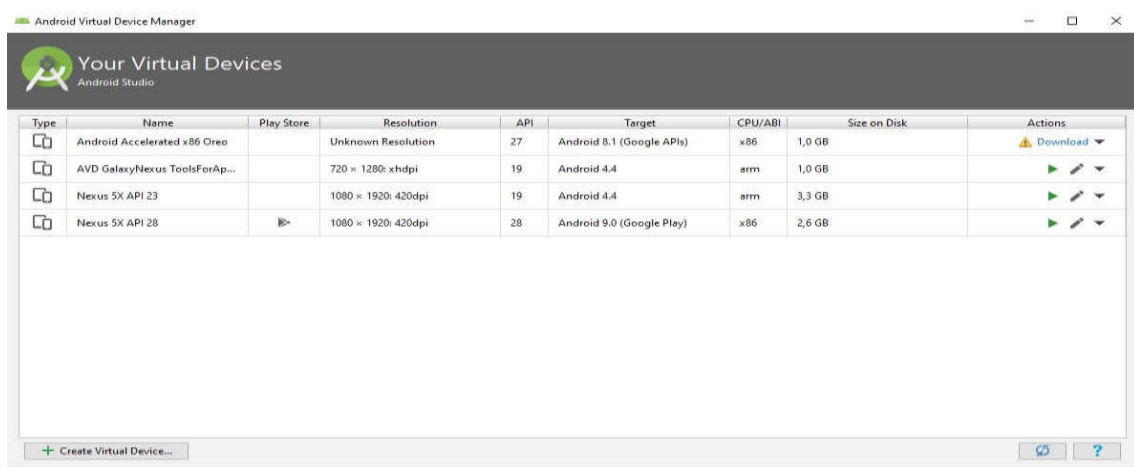
Llegados a este punto tenemos la posibilidad de comprobar el funcionamiento de nuestra aplicación y para ello podemos optar por utilizar el AVD Manager que nos gestiona un dispositivo virtual similar a un móvil o utilizar nuestro propio móvil android conectado al puerto USB del ordenador.

Para comprobar el correcto funcionamiento de la aplicación utilizaremos unos de los emuladores disponibles en la herramienta AVD Manager que nos solicitara que elijamos cuando vayamos a ejecutar nuestra aplicación. Dicha herramienta nos permite crear nuestros propios dispositivos virtuales.

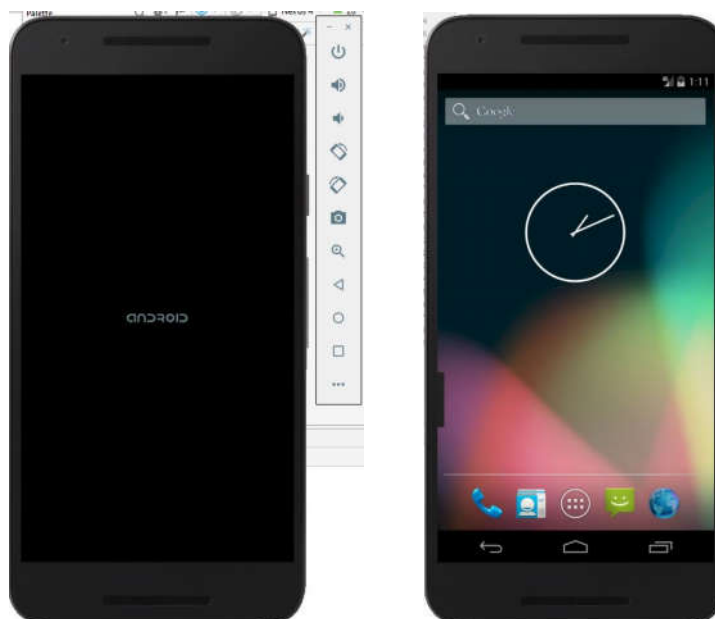


Icono del AVD Manager.

Nos aparecerá la siguiente pantalla:



Seleccionaremos uno de los dos móviles con CPUs del tipo arm. En la siguiente imagen se ve el proceso de inicio del móvil Nexus SX API 23



El siguiente paso sería vomprobar el funcionamiento de la aplización y para ello y desde android studio seleccionamos el icono que nos permite ejectuar la aplicación ► que llamaría al AVD Manager para indicarnos que dispositivo deseamos utilizar para realizar la simulación. En este caso indicaremos que el dispositivo a utilizar será el que ya hemos seleccionado y tenemos disponible (Nexus 5X API 23). A partir de este momento se procede a la instalación de la aplicación en el dispositivo móvil virtual y se lanza la actividad apareciendo en la pantalla del móvil.



Desarrollo de la práctica.

Para llevar a cabo la práctica se deja el código necesario para realizar una sencilla aplicación que lleva a cabo la suma de dos números y muestra su resultado. Este será el código de partida que nos permitirá ir avanzando en los diferentes puntos de la práctica

En MainActivity.java escribiremos el siguiente código:

```
package com.example.eps.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
    private EditText numA,numB,sumaNumeros;
    private Button botonSuma;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Obtenemos una referencia a los controles de la interfaz gráfica
        numA = (EditText) findViewById(R.id.numA);
        numB = (EditText) findViewById(R.id.numB);
        sumaNumeros = (EditText) findViewById(R.id.sumaNumeros);
        botonSuma = (Button) findViewById(R.id.botonSuma);
        //Implementamos el evento click del botón
        botonSuma.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Realizamos la suma
                String valor1=numA.getText().toString();
                String valor2=numB.getText().toString();
                int numero1=Integer.parseInt(valor1);
                int numero2=Integer.parseInt(valor2);
                int suma=numero1+numero2;
                String resultado=String.valueOf(suma);
                sumaNumeros.setText(resultado);
            }
        });
    }
}
```

Y en activity_main.xml escribiremos el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.eps.myapplication.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="NUM A"
        android:id="@+id/textView1" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/numA"
        android:textColor="#FF0"
        android:layout_below="@+id/textView1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="NUM B"
        android:id="@+id/textView2"
        android:layout_below="@+id/numA" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/numB"
        android:textColor="#FF0"
        android:layout_below="@+id/textView2" />

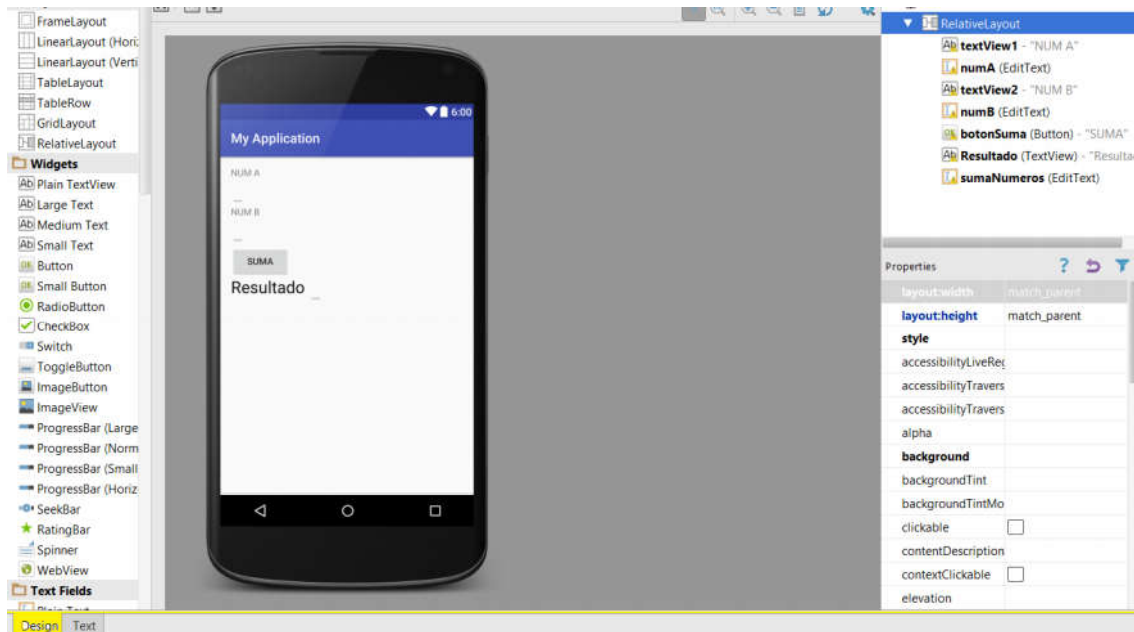
    <Button android:id="@+id/botonSuma"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SUMA"
        android:layout_below="@+id/numB" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Resultado "
        android:id="@+id/Resultado"
        android:singleLine="false"
        android:textColor="#000"
        android:textSize="25sp"
        android:layout_below="@+id/botonSuma"/>

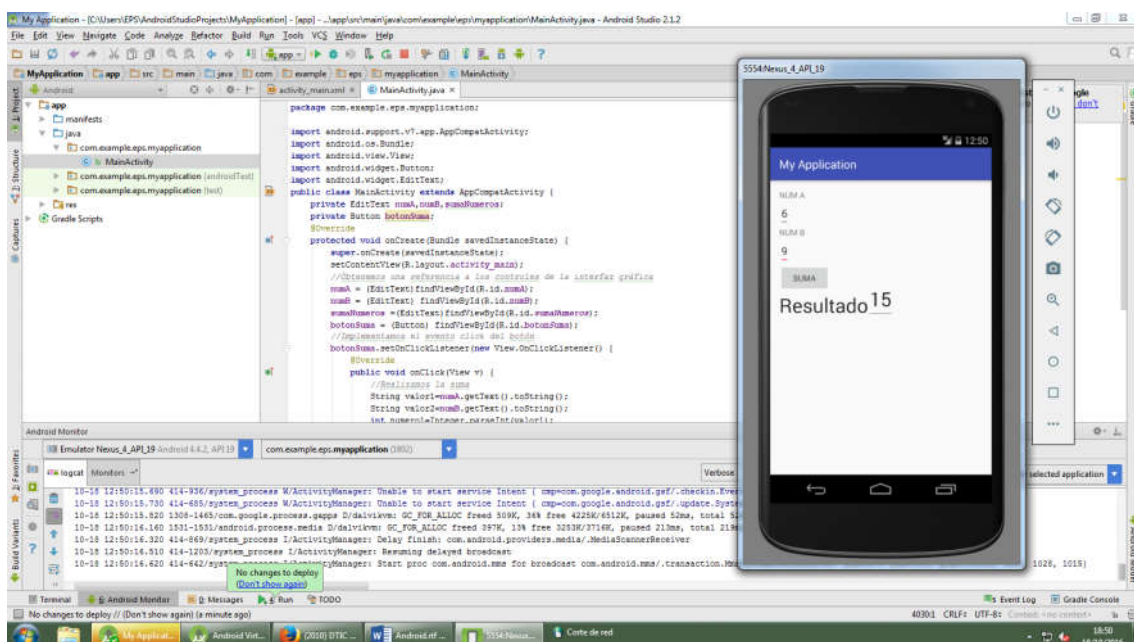
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/sumaNumeros"
        android:textSize="16sp"
        android:textColor="#000"
        android:layout_below="@+id/botonSuma"
        android:layout_toRightOf="@+id/Resultado"/>

</RelativeLayout>
```

Si seleccionamos la pestaña de Design veremos sobre la pantalla del móvil el resultado gráfico del código que hemos escrito



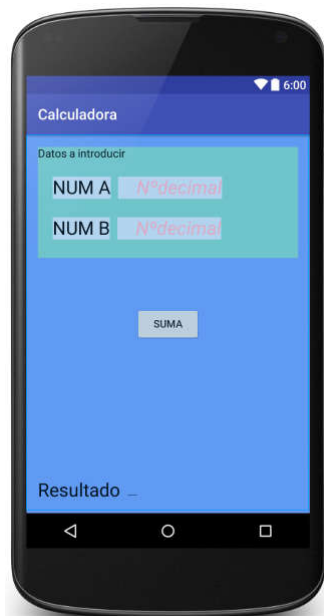
El último paso consiste en comprobar el correcto funcionamiento de la aplicación y para ello utilizaremos el procedimiento descrito anteriormente. El resultado que veríamos sobre este es el siguiente:



EVALUACIÓN PRÁCTICA 2

Ejercicio 1 (6 puntos). Aplicación básica guiada.

- Ejecuta el código proporcionado y comprueba el correcto funcionamiento.
- Realiza las modificaciones en la interfaz grafica para paso a paso llegar a obtener los siguientes avances en la interface. El objetivo es hacer una interface más vistosa



- Añade las operaciones aritméticas de RESTA, MULTIPLICACION Y DIVISION.



- Modifica el código para trabajar solo con números binarios naturales de 8 bits. En caso de que el resultado supere el rango máximo o mínimo representable mostraremos una notificación por pantalla indicando que la operación no puede realizarse.

Ejercicio 2 (2,5 puntos). Añadir nuevas funcionalidades

- Realiza las modificaciones oportunas para que el programa pueda realizar operaciones aritméticas en binario natural, Signo y Magnitud así como en complemento a 2.



Ejercicio 3 (1,5 puntos). Modificación

- Durante la corrección en clase se plantearán modificaciones sobre alguno de los ejercicios anteriores.

NORMAS DE ENTREGA

- Se establecerá un control de entrega de prácticas en el CV para la práctica P2. Dicha entrega será individual y la fecha límite de entrega será el 15 de Diciembre 2019. La corrección de la práctica en laboratorio se realizará el día 17 Diciembre de 2019 para los grupos del martes y del día 20 de Diciembre para los grupos del viernes.