



PREPÁRATE
PARA SER EL
MEJOR



+ **ENTREMIENTO
EXPERIENCIA**



BIENVENIDOS.





ANDROID DESDE CERO CON KOTLIN

Ing. Marco Estrella
Instructor Tecnologías Java y Android

mestrella@galaxy.edu.pe



AGENDA

SESIÓN

02 Android desde Cero con Kotlin

- ▶ Breve historia de Android.
- ▶ Plataforma Android.
- ▶ Plataforma Android componentes.
- ▶ Primer proyecto.
- ▶ Android Studio.
- ▶ Estructura de un proyecto
- ▶ ¿Qué es un Activity?
- ▶ Activity – El Layout
- ▶ Activity – La Clase
- ▶ Activity declaración en el manifest
- ▶ Ciclo de vida de un Activity



AGENDA

SESIÓN

02 Android desde Cero con Kotlin

- ▶ La clase View
- ▶ Jerarquía de la clase View.
- ▶ Componentes Hijos de la Clase View.
- ▶ La clase View Group.
- ▶ Jerarquía de la clase Viewgroup.
- ▶ Viewgroups.
- ▶ Linearlayout
- ▶ La clase R
- ▶ Eventos Onclick
- ▶ Intent
- ▶ Dundle

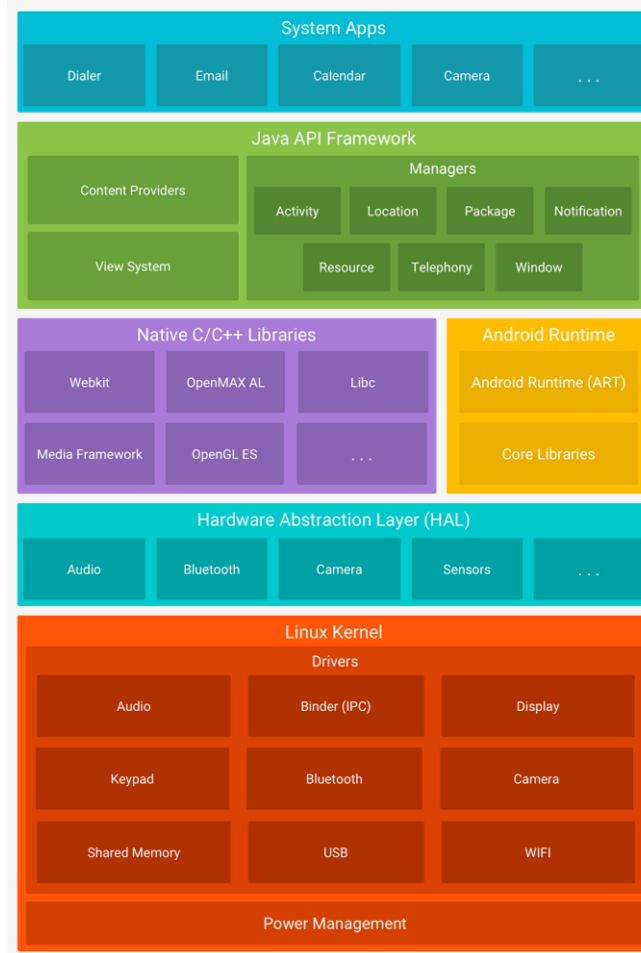


Android 7.0 Nougat

2016



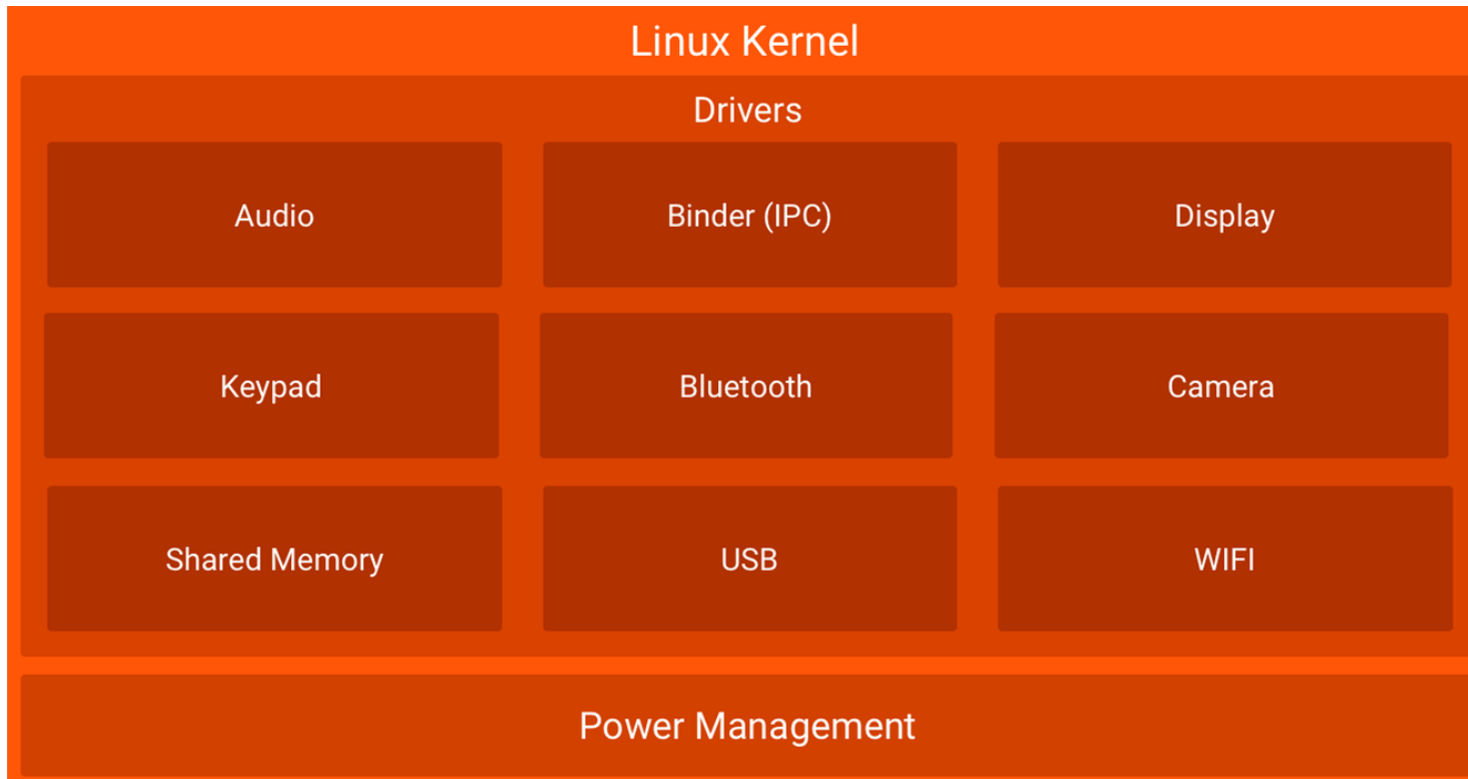
2017



Android es un conjunto de softwares open source basados en Linux.

Fue creado para una gama inmensa de dispositivos, entre ellos smartphones, tablets, watches, TV's y autos.

El diagrama muestra los componentes principales de la plataforma Android.



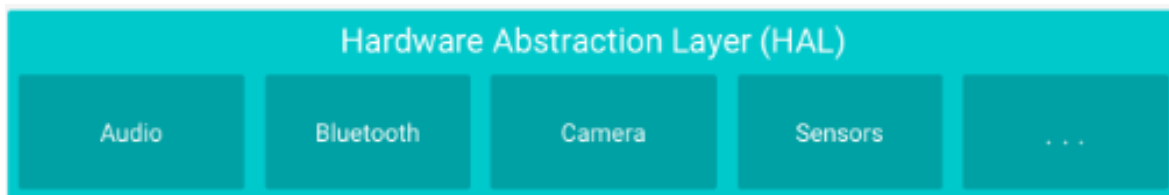
La base de la plataforma Android es el kernel de linux.

El kernel se encarga de la gestión de hilos y administración de memoria de bajo nivel entre sus tareas principales.

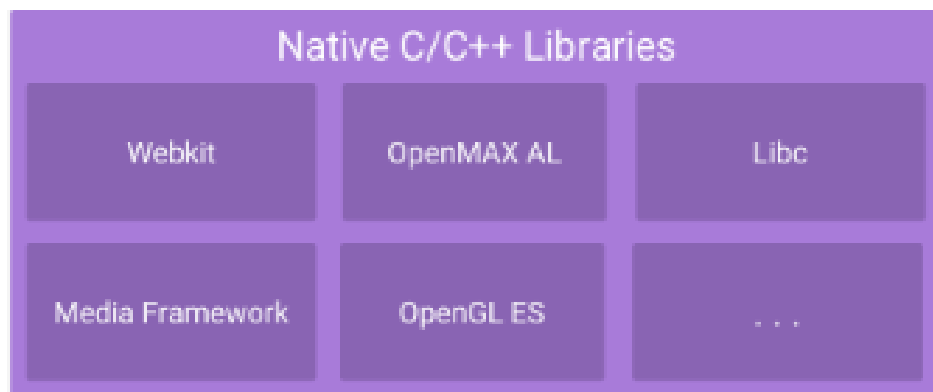
Ya que el kernel de linux es muy famoso entre los fabricantes de dispositivos, estos pueden tener mejor acceso y facilidad al crear drivers.



PLATAFORMA ANDROID – COMPONENTES (cont...)



HAL consiste en múltiples módulos de librerías, cada una implementa una interfaz para un tipo de componente de hardware, Tal como la cámara o el módulo de bluetooth.

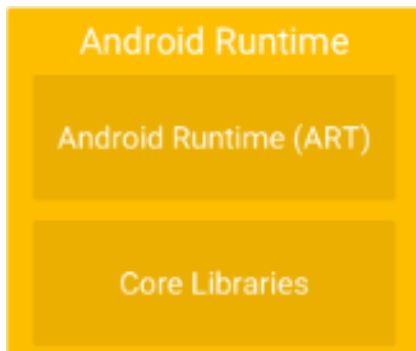


Conjunto de librerías nativas hechas en C/C++ con la intención de comunicarse a otros componentes nativos interno, tal como HAL o ART.

Por otro lado, los Java API Frameworks usan estas librerías para exponer funcionalidades a las aplicaciones.



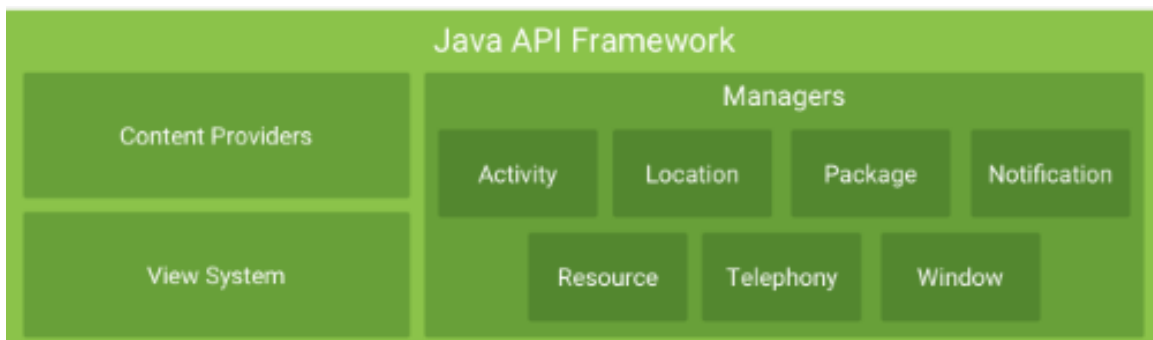
PLATAFORMA ANDROID - COMPONENTES



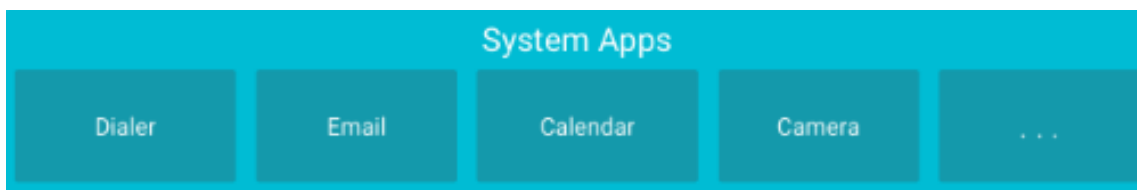
Es el motor que ejecuta las aplicaciones android.

Cada aplicación corre en un proceso y con su propia instancia de ART.

Ejecuta archivos DEX, un formato de bytecode diseñado especialmente para android.

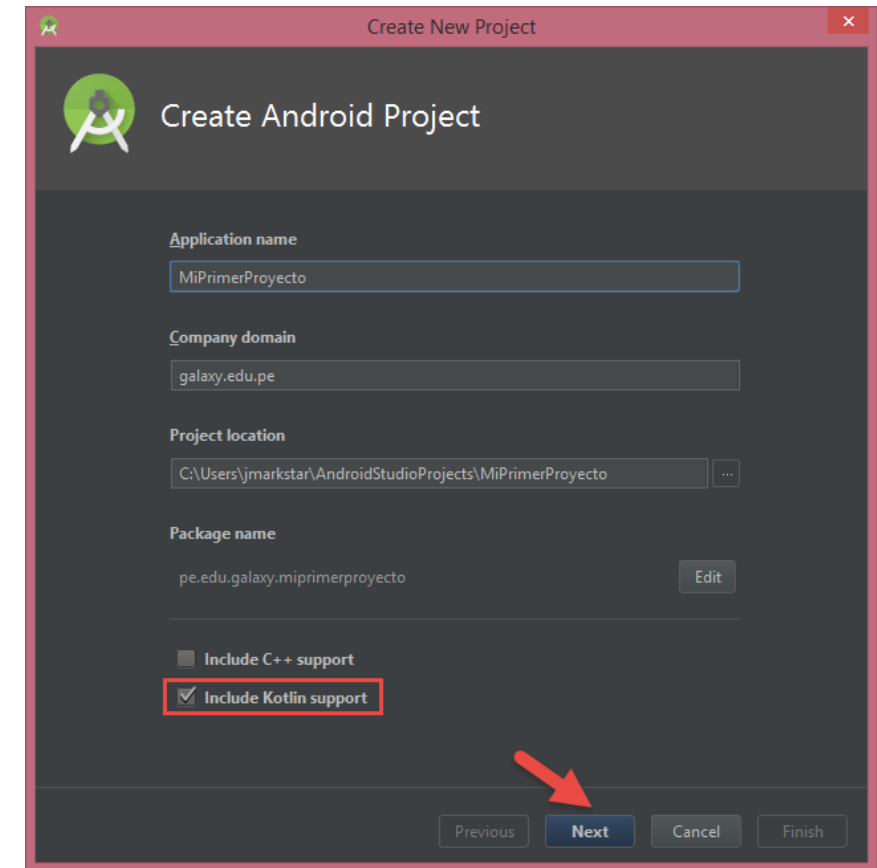
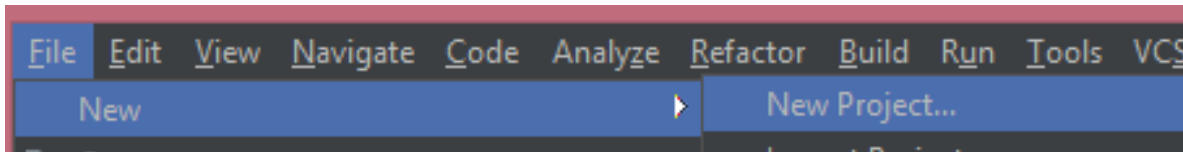


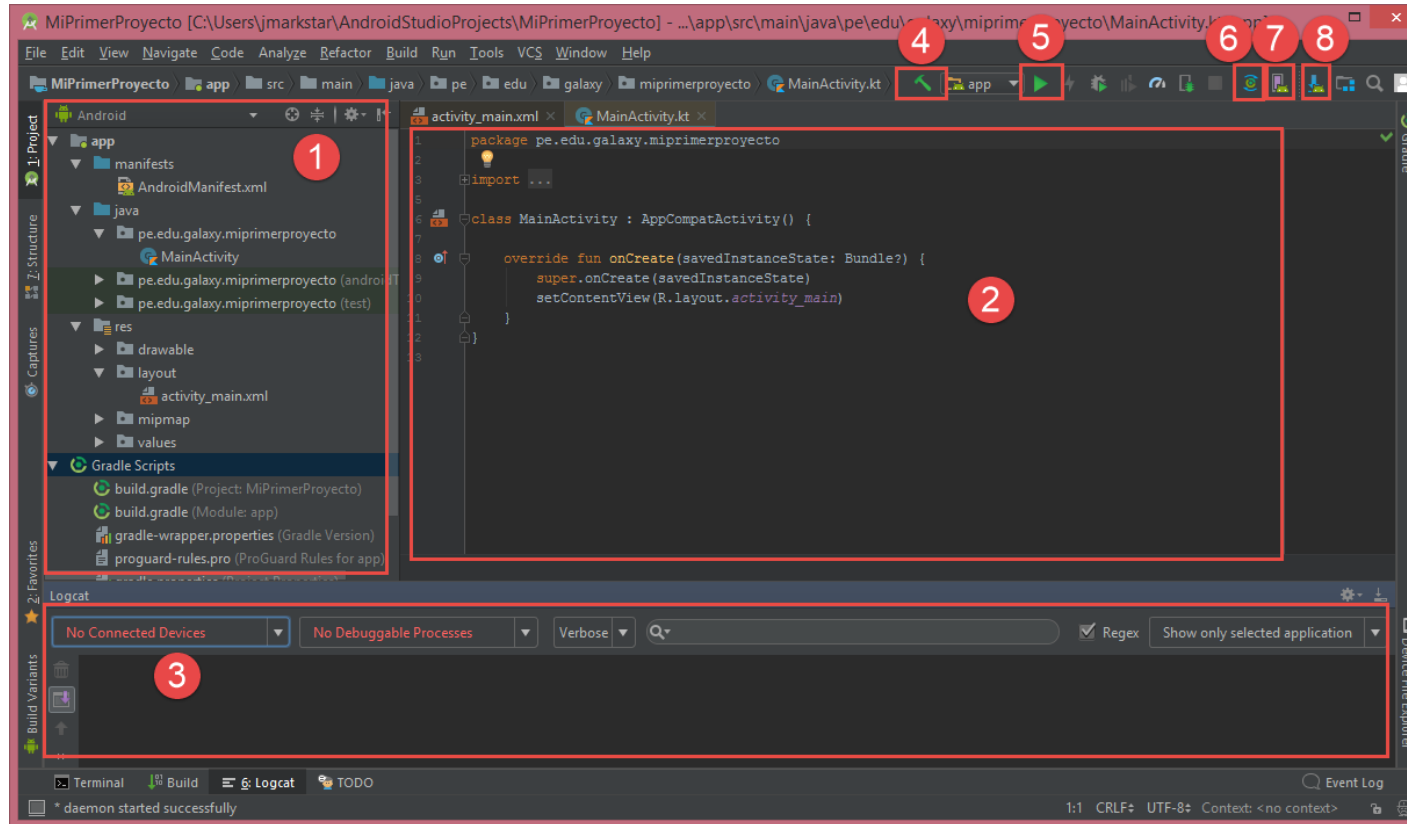
Son las librerías del SDK hechas en java para el desarrollo de aplicaciones.



Algunas apps preinstaladas que ya vienen con el sistema operativo y los apps que crearas en el futuro.

Nos dirigimos a File > New > New Project





1 Explorador del proyecto.

2 Editor de Texto.

3 Logcat, nos ayuda a ver logs y errores que podrían ocurrir en nuestro app.

4 Build, compila y crear un apk en modo desarrollo.

5 Run, compila y ejecuta el proyecto en un emulador o dispositivo real.

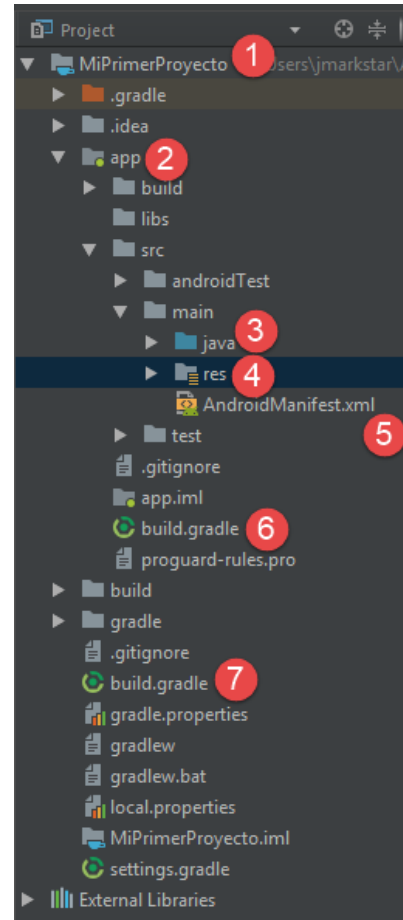
6 Sync, descarga las librerías que nuestro app necesita.

7 ADV Manager, nos ayuda a crear emuladores.

8 SDK Manager, nos ayuda a descargar versiones de plataformas android.



ESTRUCTURA DE UN PROYECTO



1 Nombre del proyecto(carpeta raíz).

2 Módulo app(nuestro app principal).

3 Carpeta donde se crean y guardan las clases en kotlin o java.

4 Carpeta donde se crean y guardan los recursos y layouts.

5 Archivo principal del app donde declaramos los activities, servicios, permisos, etc.

6 Archivo gradle del módulo app.

7 Archivo gradle del proyecto completo.



ESTRUCTURA DE UN PROYECTO



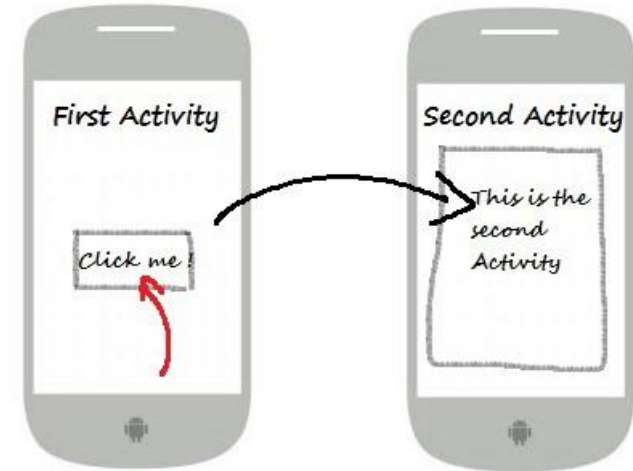
- 1 Cuando creamos un proyecto, el nombre asignado será el nombre de la carpeta raíz de todo el proyecto, este puede tener muchos módulos internos. Un módulo podría ser un **android app**, una **librería android** o una **librería java**, entre otros pero estos son los más usados.
- 2 Cuando se crea un proyecto en Android Studio, este crea un módulo por defecto llamado “**app**” que es un módulo de tipo aplicación móvil.
- 3 Internamente, en un módulo de tipo aplicación móvil se crea una carpeta **main**, dentro de ella existen 2 carpeta:: **java** y **res**; en la carpeta java se guardan todos los paquetes, clases e interfaces escritas en kotlin o java que podrías crear para tu proyecto.
- 4 En la carpeta **res** se guardan los archivos *.xml para el diseño de las pantallas para nuestra aplicación, además de imágenes, strings y otros tipos de recursos que podríamos usar en nuestro app.
- 5 El **AndroidManifest.xml** es el archivo que contiene las configuraciones más importantes del app, como por ejemplo los permisos que el app usará en el dispositivo: la cámara, acceso a los datos del GPS, acceso a los datos del usuario, etc. Además, Aquí se declaran los activities, services del app.
- 6 El archivo **build.gradle** que se crea dentro del módulo ‘app’ contiene las configuraciones para las versiones que soportará nuestro app, las librerías que usará, entre otros.
- 7 El archivo **build.gradle** que se crea en la raíz del proyecto contiene las configuraciones generales del app como por ejemplo, los plugins de gradle y los repositorios de donde se descargan las librerías terceras que necesitemos.



¿QUÉ ES UN ACTIVITY?

Una actividad es algo que el usuario puede hacer, puede interactuar con ellos.

Básicamente, es la representación de una pantalla.



Normalmente está dividido en 2 partes: Un archivo **XML** y una clase **Java**.



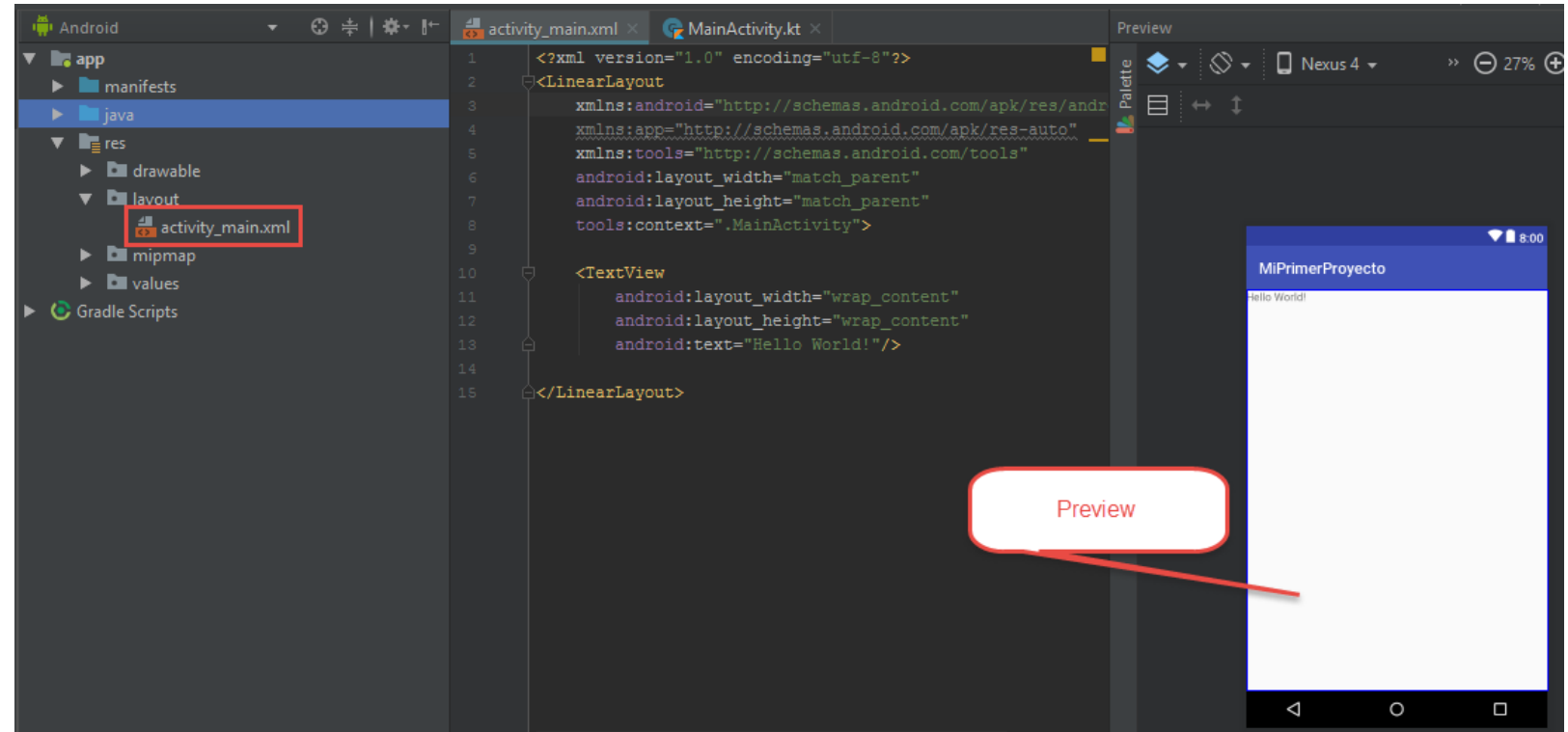
ACTIVITY - EL LAYOUT



Se encuentra en
la ruta:

res >
layout > activity_main.xml

En el este archivo
xml
diseñamos la
pantalla.





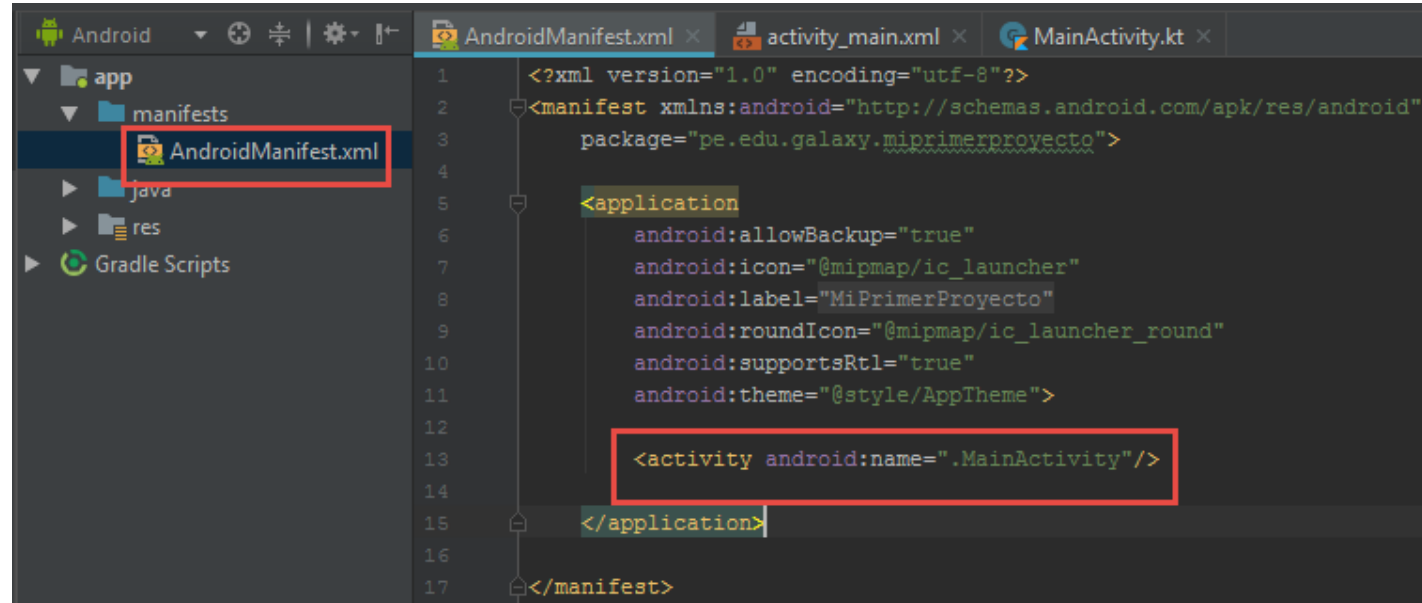
ACTIVITY – LA CLASE

```
1 package pe.edu.galaxy.miprimerproyecto
2
3 import android.support.v7.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
```

En este archivo java programamos el comportamiento de la pantalla.



ACTIVITY - DECLARACIÓN EN EL MANIFEST

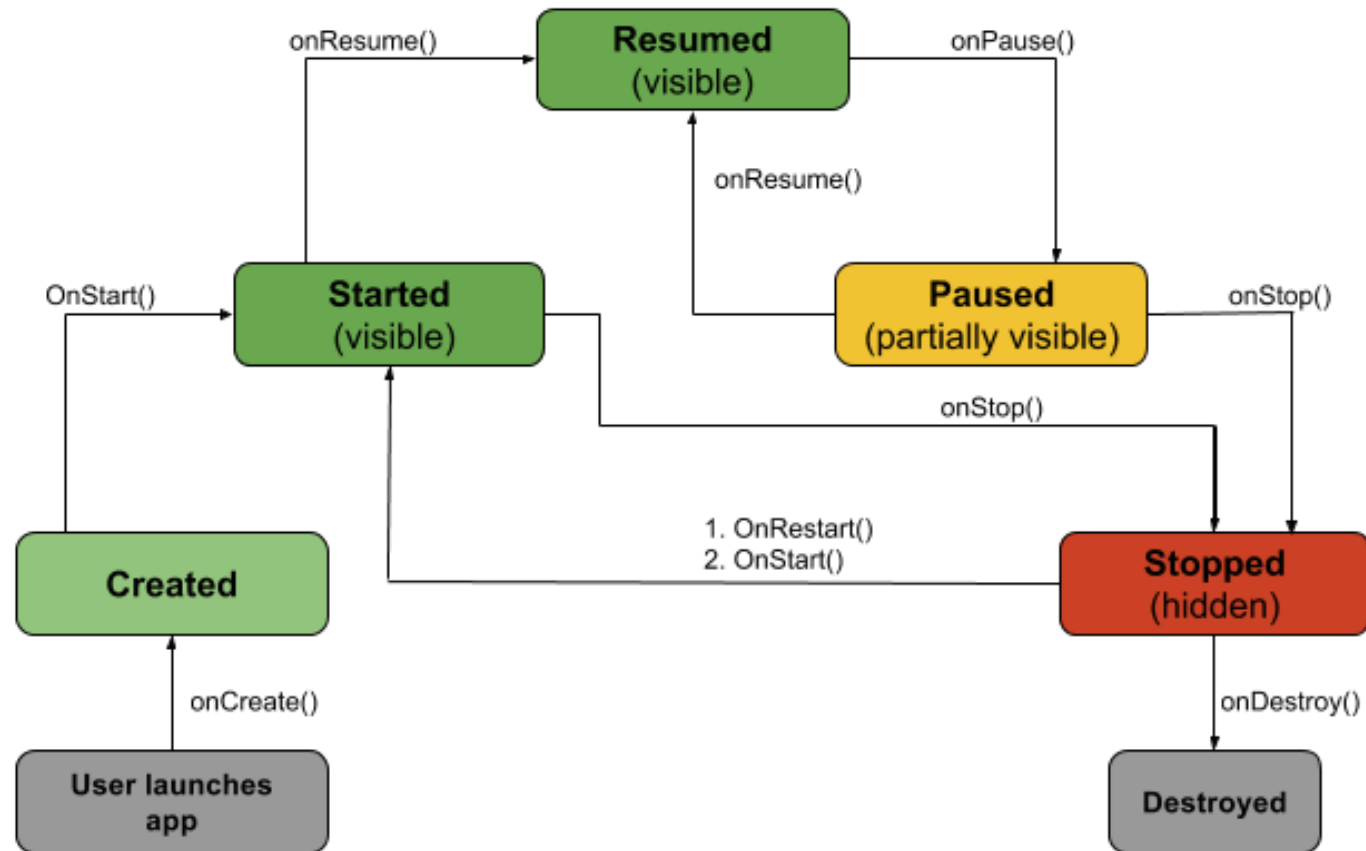


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="pe.edu.galaxy.miprimerproyecto">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="MiPrimerProyecto"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportRtl="true"
11        android:theme="@style/AppTheme">
12
13        <activity android:name=".MainActivity"/>
14
15    </application>
16
17 </manifest>
```

Es muy importante declarar el activity creado en el archivo **AndroidManifest.xml**, de lo contrario el Activity jamás podrá ser usado y ocurría un error al intentarlo.

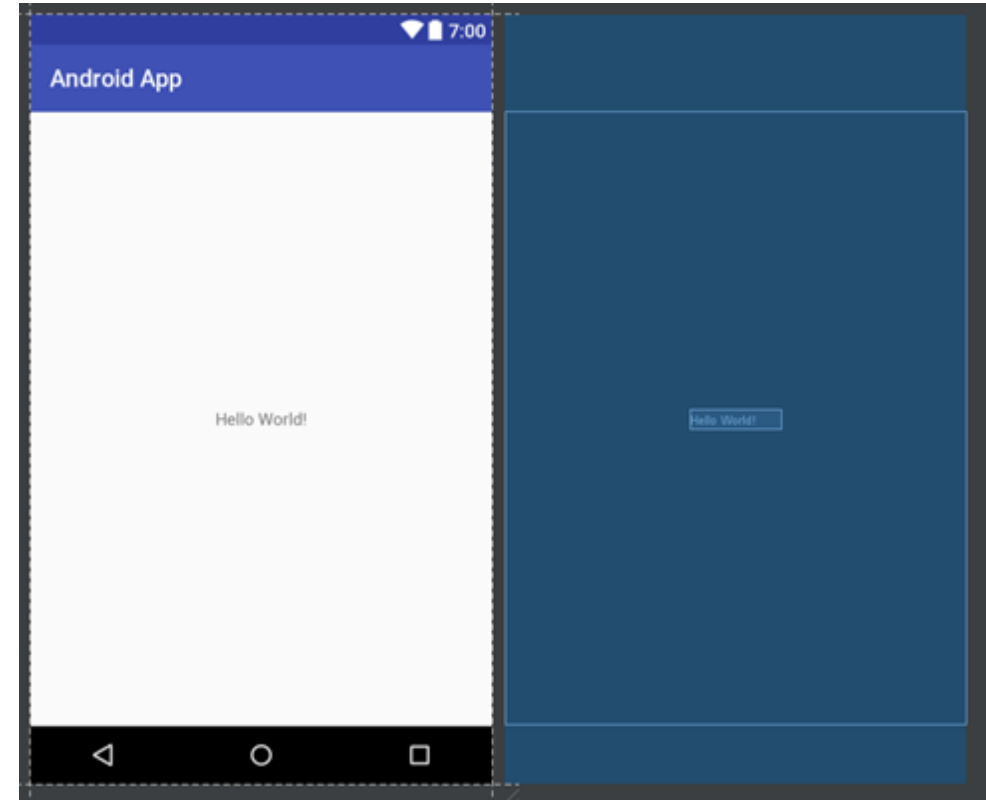


CICLO DE VIDA DE UN ACTIVITY



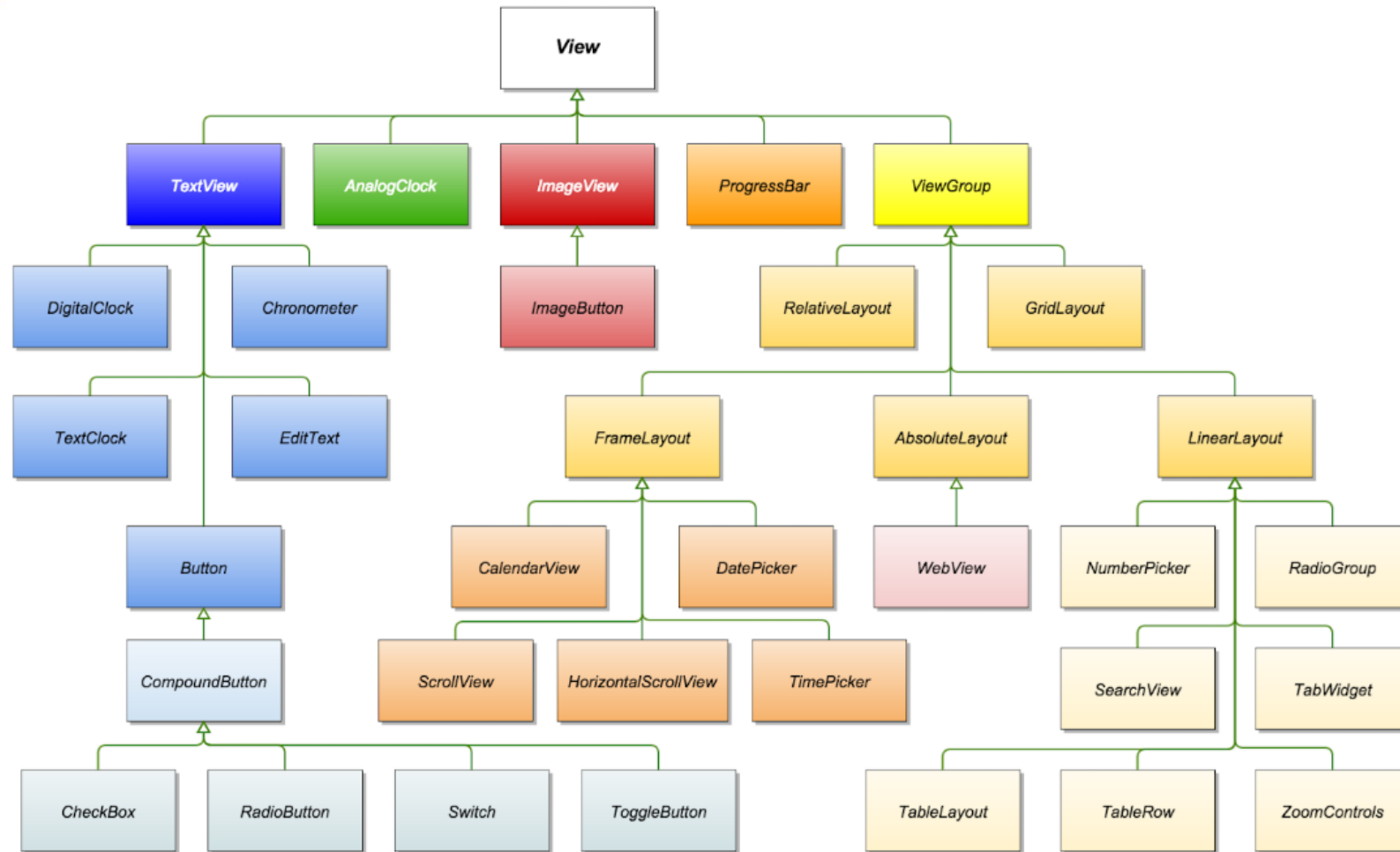
Representa el bloque de construcción para los componentes UI.

Ocupa un área rectangular en la pantalla y es responsable de dibujar y manejar los eventos.





JERARQUÍA DE LA CLASE VIEW





COMPONENTES HIJOS DE LA CLASE VIEW



TextView

Hello World!

```
<TextView android:id="@+id/tv_my_textview"  
    android:text="Hello World!"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

Button

MI BOTON

```
<Button android:id="@+id/btn_my_button"  
    android:text="MI BOTON"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```



EditText

Ingrese si nombre

```
<EditText android:id="@+id/et_name"  
    android:hint="Ingrese si nombre"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

RadioButton

☐ Opción 1

```
<RadioButton android:id="@+id/rn_option_1"  
    android:text="Opción 1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

CheckBox

☐ ¿Esta casado?

```
<CheckBox android:id="@+id/cb_mi_checkbox"  
    android:text="¿Esta casado?"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```



Es la clase base para los diseños(layouts) de cada pantalla.

Es un contenedor invisible que puede contener Views y otros ViewGroups y define sus propiedad de diseño

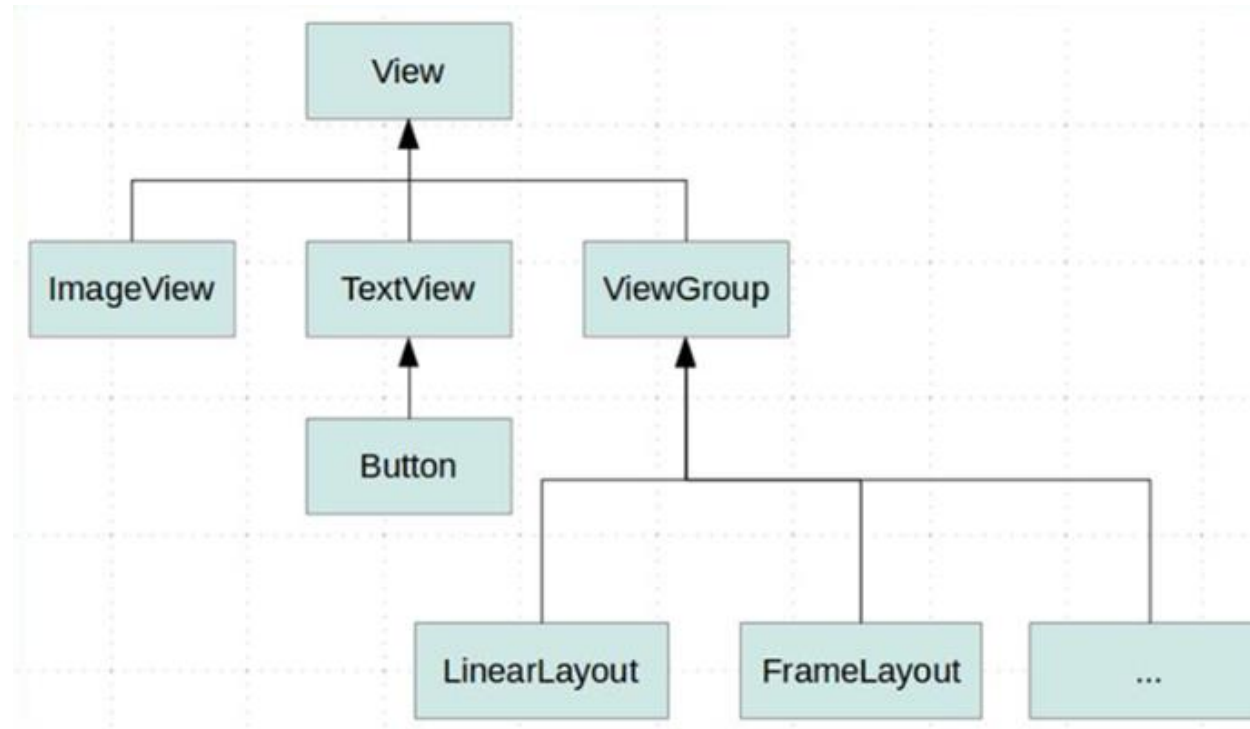
Layout?
Es la forma en que las partes de algo son organizadas.





JERARQUÍA DE LA CLASE VIEWGROUP

La clase ViewGroup también es hijo de View



LinearLayout

ScrollView

GridView

ViewPager

TableLayout

RelativeLayout

ListView

RecyclerView

ConstraintLayout

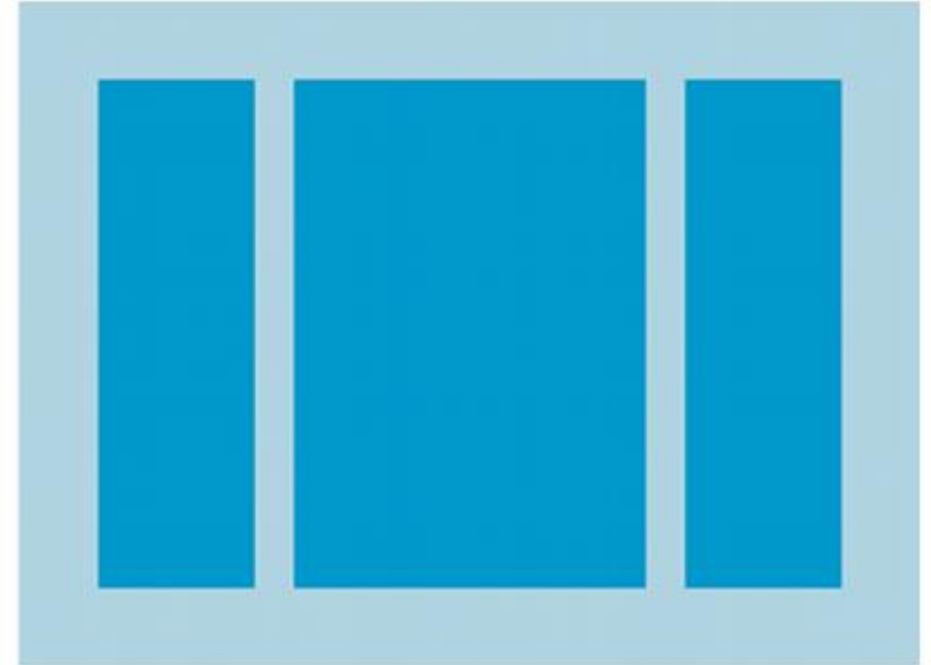
FrameLayout

CoordinatorLayout

Ordena a los hijos de manera vertical u horizontal.

Especifica el alineamiento de todos los hijos.

Su tamaño es especificación con
`LinearLayout.Params`



Es una clase que se autogenera en tu proyecto y te permite acceder a los recursos que has creado; como por ejemplo una imagen, un string, un entero, un array, un layout, etc.

Se usa desde la clase
java
de un activity.

Ejemplo: `R.string.nombre`

Se usa desde el layout
xml
de un activity.

Ejemplo: `@string/nombre`



Todos los Views pueden tener eventos OnClickListener

Permite ejecutar un bloque de código cuando el usuario hace clic en un componente de vista.



```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val myView: View = View(context, this)  
  
        myView.setOnClickListener { it: View!  
            //Escribe el código a ejecutar.  
        }  
    }  
}
```

Clase que nos permite movernos de una pantalla a otra.

Nos permite enviar información a la siguiente usando su método:

putExtra(clave, valor)

finalmente, tenemos que ejecutar el método **startActivity(intent)** para activar el Intent.

```
class HomeActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_segundo)  
  
        /// MI CODIGO  
    }  
  
    fun irAPantalla2(nombreUsuarioLogeado: String, token: String){  
        val intent = Intent( packageContext: this, HomeActivity::class.java).apply { this: Intent  
            putExtra( name: "usuario", nombreUsuarioLogeado)  
            putExtra( name: "token", token)  
        }  
  
        startActivity(intent)  
    }  
}
```


La clase Bundle no permite cargar los datos a enviar de forma separada.

para cargar el Bundle al Intent se tiene que usar el método:

intent.putExtras(mybundle)

y para obtenerlo en la otra pantalla se usaria el metodo:

intent.getExtras()

```
class HomeActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_segundo)  
  
        /// MI CODIGO  
    }  
  
    fun irAPantalla2(nombreUsuarioLogeado: String, token: String){  
  
        val myBundle = Bundle()  
        myBundle.putString("usuario", nombreUsuarioLogeado)  
        myBundle.putString("token", token)  
  
        val intent = Intent( packageContext: this, HomeActivity::class.java).apply { this: Intent  
            putExtras (myBundle)  
        }  
  
        startActivity(intent)  
    }  
}
```



Thank you!
Questions?





GALAXY
TRAINING