

Desarrollo de Aplicaciones Móviles para Android: Primer Acercamiento a la Programación

Dr. Marco Aurelio Nuño-Maganda

Universidad Politecnica de Victoria

Mayo 2023



Contenido

- 1 Introducción
- 2 Pasos para crear una primera aplicación
- 3 Configurar smartphone en modo de Depuracion
- 4 Modificacion de la Interfaz de usuario
- 5 Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)
- 6 Etapa 2: Implementar el comportamiento de la aplicacion
- 7 Conclusiones

1 Introducción

2 Pasos para crear una primera aplicación

3 Configurar smartphone en modo de Depuracion

4 Modificacion de la Interfaz de usuario

5 Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)

6 Etapa 2: Implementar el comportamiento de la aplicacion

7 Conclusiones

Computadora y Programabilidad

Computadora

Es una máquina (electrónica) programable* que recibe y procesa datos para convertirlos en información útil. Contiene periféricos de entrada (para introducir datos) y salida (para mostrar resultados)

Computadora y Programabilidad

Computadora

Es una máquina (electrónica) programable* que recibe y procesa datos para convertirlos en información útil. Contiene periféricos de entrada (para introducir datos) y salida (para mostrar resultados)

Algoritmo

Conjunto finito de instrucciones para resolver una tarea específica

Computadora y Programabilidad

Computadora

Es una máquina (electrónica) programable* que recibe y procesa datos para convertirlos en información útil. Contiene periféricos de entrada (para introducir datos) y salida (para mostrar resultados)

Algoritmo

Conjunto finito de instrucciones para resolver una tarea específica

Programación

El proceso de crear un software utilizando un lenguaje de programacion (C, C++, Java, Python, Kotlin, etc)

Computadora y Programabilidad

Computadora

Es una máquina (electrónica) programable* que recibe y procesa datos para convertirlos en información útil. Contiene periféricos de entrada (para introducir datos) y salida (para mostrar resultados)

Algoritmo

Conjunto finito de instrucciones para resolver una tarea específica

Programación

El proceso de crear un software utilizando un lenguaje de programacion (C, C++, Java, Python, Kotlin, etc)

Programa

Un conjunto de instrucciones que una computadora interpreta en una secuencia logica para llevar a cabo una tarea en particular

Computadora y Programabilidad

Computadora

Es una máquina (electrónica) programable* que recibe y procesa datos para convertirlos en información útil. Contiene periféricos de entrada (para introducir datos) y salida (para mostrar resultados)

Algoritmo

Conjunto finito de instrucciones para resolver una tarea específica

Programación

El proceso de crear un software utilizando un lenguaje de programacion (C, C++, Java, Python, Kotlin, etc)

Programa

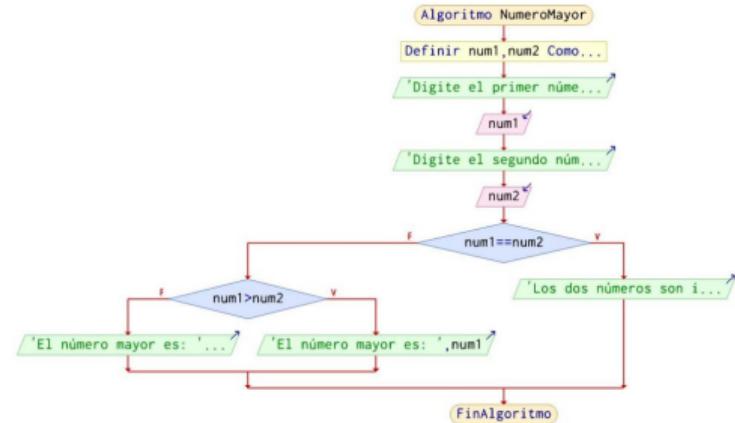
Un conjunto de instrucciones que una computadora interpreta en una secuencia logica para llevar a cabo una tarea en particular

Diagrama de Flujo

Un diagrama de flujo es una representación gráfica de un algoritmo, los pasos que los componen y la secuencia de ejecución de sus instrucciones

Algoritmo

Diseñar un algoritmo para comparar dos números



Codificación de un algoritmo en varios lenguajes de programación (Python - Kotlin)

Codificación del Algoritmo en Python

```
1 def CompararNumeros(A,B):
2     if (A==B):
3         print ("Son Iguales ")
4     else:
5         if (A>B):
6             print ("A es mayor que B ")
7         else:
8             print ("B es mayor que A ")
9
10 As = input ("Escribe el primer numero (A): ")
11 A = int(As)
12 Bs = input ("Escribe el segundo numero (B): ")
13 B = int(Bs)
14 CompararNumeros(A,B)
```

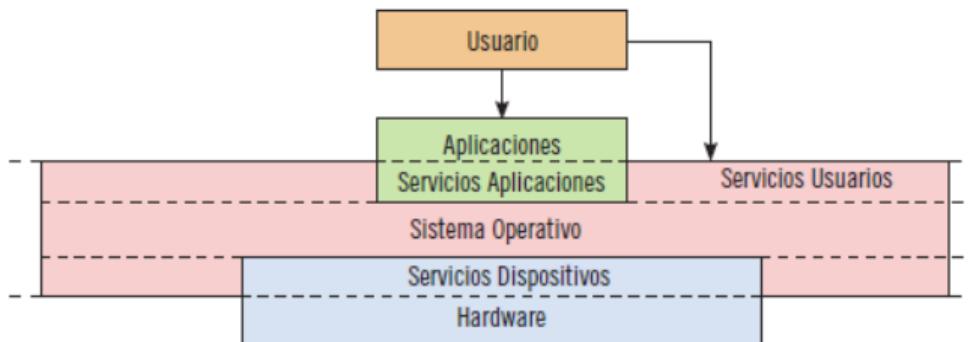
Codificación del Algoritmo en Kotlin

```
1 package com.example.myapplication
2 fun CompararNumeros (A : Int, B: Int) {
3     if (A==B) {
4         print ("Son Iguales")
5     } else {
6         if (A > B) {
7             print ("A es mayor que B")
8         } else {
9             print ("B es mayor que A")
10        }
11    }
12 }
13 fun main() {
14     print("Escribe el primer numero (A):")
15     val As = readLine()!!
16     val A= Integer.parseInt(As)
17     print("Escribe el segundo numero (B):")
18     val Bs = readLine()!!
19     val B= Integer.parseInt(Bs)
20     CompararNumeros(A,B)
21 }
```

Sistema Operativo

Un Sistema Operativo (SO) es un programa (software) que al arrancar la computadora** se encarga de gestionar todos los recursos del sistema informático permitiendo así la comunicación entre el usuario y la computadora.

Estructuración de los servicios del sistema operativo



<https://reader.digitalbooks.pro/content/preview/books/38230/book/0EBPS/Text/c1.html>

Bloques funcionales de un sistema operativo





Telefono Celular No-inteligente vs Telefono Celular Inteligente

Teléfono No-inteligente

- Su funcionalidad principal era la comunicación (llamadas o mensajes) a través de la red celular (GSM)

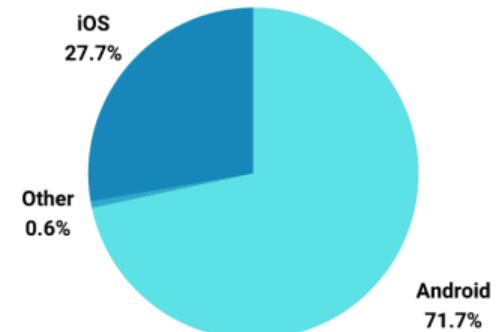
Teléfono inteligente

- Interfaz de entrada: Pantalla Touch (a color, de alta definición)
- Conexión a Internet: WiFi, GSM (4G o 5G)
- Comunicación con otros dispositivos: Bluetooth, NFC
- Cámaras (Frontal y Posterior)



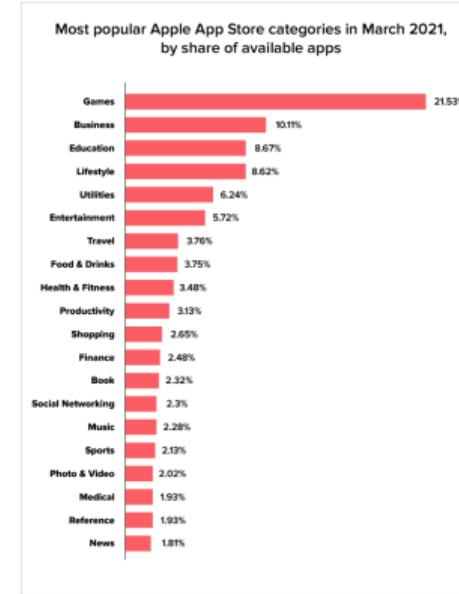


- Android es un sistema operativo móvil basado en Linux
- Principalmente orientado a dispositivos de pantalla táctil (Smartphone, tablets, smartwatches, etc)
- Fue desarrollado por Android Inc (Adquirida por Google en 2005)
- Vinculado con un grupo de empresas (HTC, Sony, Motorola, Samsung, LG, Lenovo, entre otras) para la creación de un SO común para sus dispositivos
- A la fecha (Q1 2023), los teléfonos con SO Android concentran mas del 70% del mercado global.



Aplicaciones Móviles

- Ejecutadas en el teléfono
- La entrada de datos es mediante un teclado “virtual”
- El apuntador del ratón es la pantalla
- Incluyen una interfaz de usuario gráfica (GUI)
- Es posible descargar miles de éstas en nuestros dispositivos



[https://www.netsolutions.com/insights/
top-10-most-popular-apps-2018/](https://www.netsolutions.com/insights/top-10-most-popular-apps-2018/)

- Android Studio es un entorno oficial de desarrollo integrado (IDE) para el sistema operativo Android de Google
- La primera versión se libera en el año 2013, siendo el lenguaje de programación Java
- En 2019, se reemplaza el lenguaje oficial de desarrollo por Kotlin, aunque Java todavía es soportado
- Es gratis, se puede descargar e instalar en cualquier computadora sin importar el sistema operativo (Windows, Linux y MacOS) <https://developer.android.com>

Contenido

1 Introducción

2 Pasos para crear una primera aplicación

3 Configurar smartphone en modo de Depuracion

4 Modificacion de la Interfaz de usuario

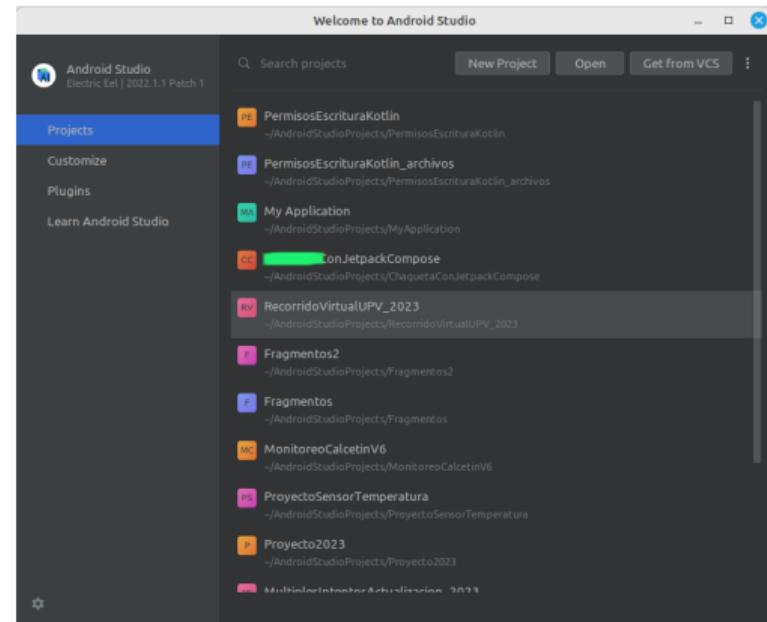
5 Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)

6 Etapa 2: Implementar el comportamiento de la aplicacion

7 Conclusiones

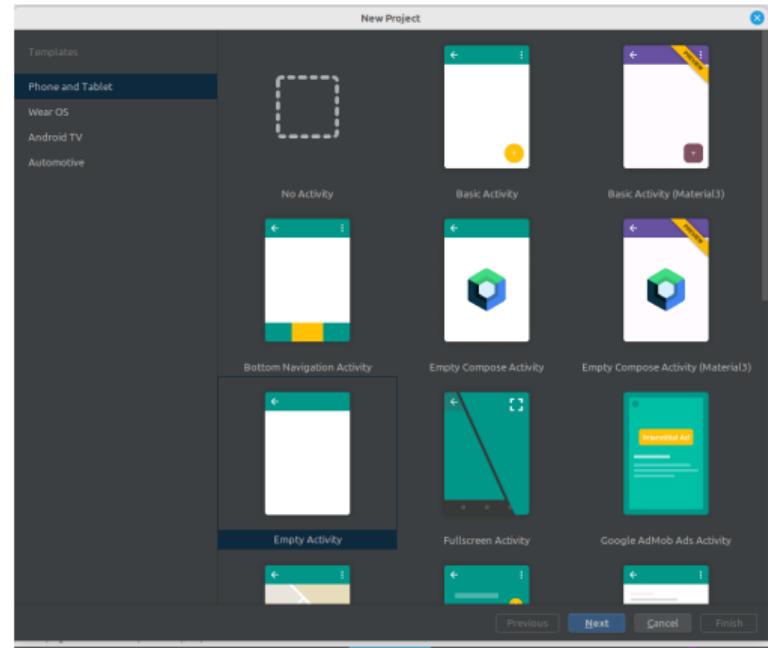
Crear un primer proyecto (1)

- Buscar el icono de la aplicación y dar doble click
- En caso de que requiera algunas actualizaciones, ser paciente, puede tardar.
- Aparecerá una ventana como la mostrada, seleccionar la opción “New project”



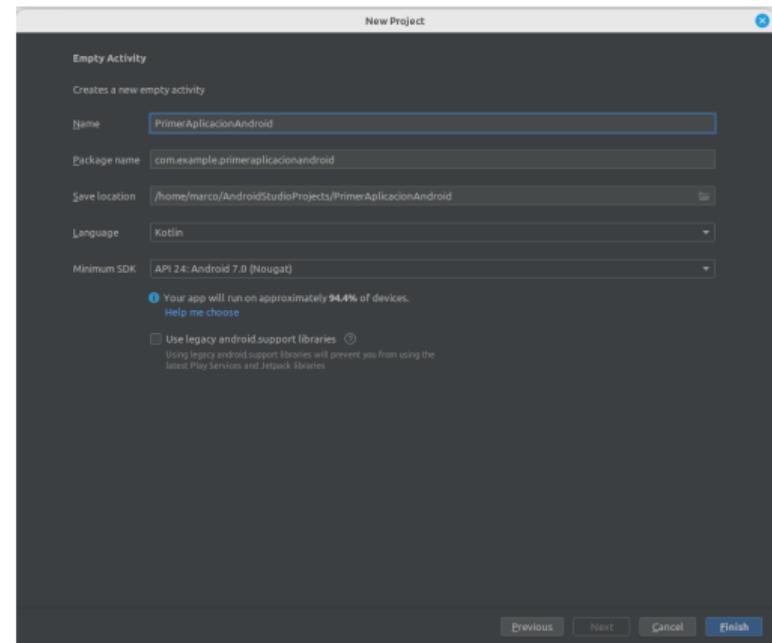
Crear un primer proyecto (2)

- Existen varios tipos de dispositivos para los que pueden crearse proyectos. Por defecto nos ubica en el primer grupo (Phone and Tablet)
- Seleccionar tipo de proyecto “Empty Activity” y dack click en Next



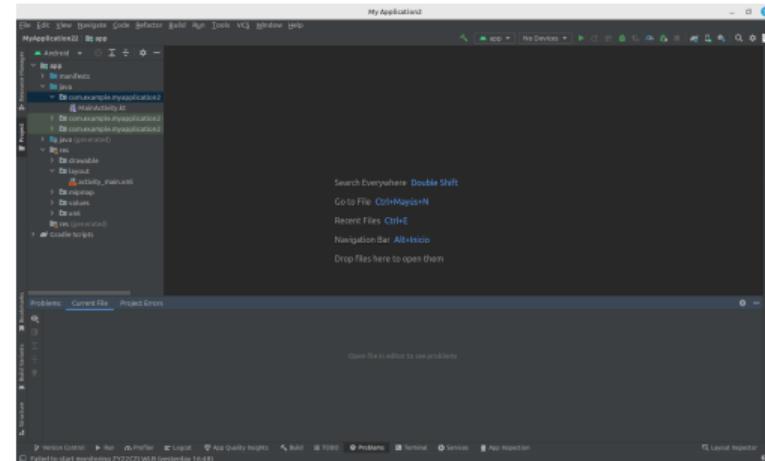
Crear un primer proyecto (3)

- Dejar los valores por defecto, solo se sugiere cambiar el nombre del proyecto
- **Debe estar seleccionado como lenguaje Kotlin.** De estar seleccionado uno diferente, cambiar
- Presionar el botón Finish



Crear un primer proyecto (4)

- La primera vez que se crea un proyecto, puede requerir la descarga de archivos necesarios. Le recomiendo ser paciente
- Ya una vez que el proyecto fue creado, aparece la ventana mostrada



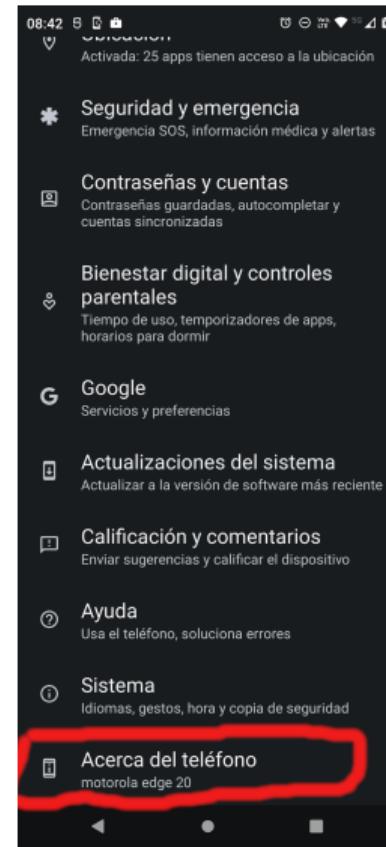
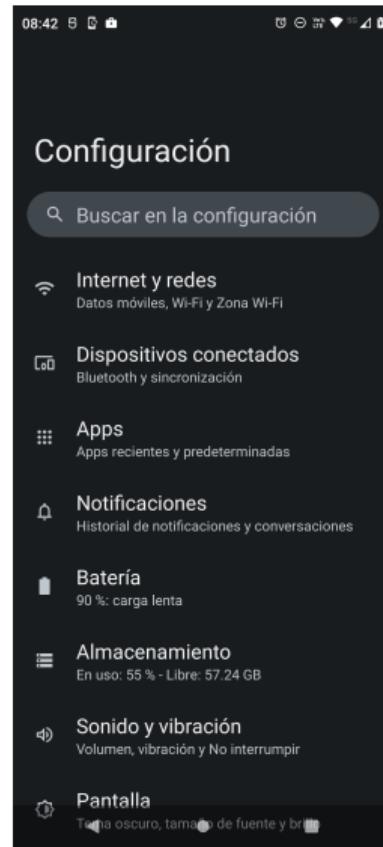
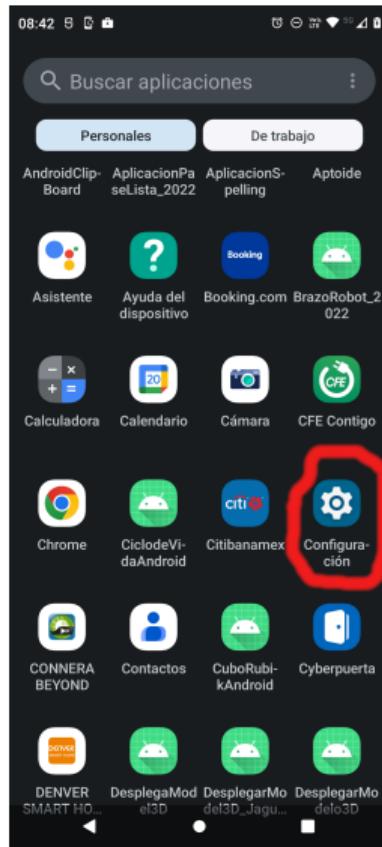
Contenido

- 1 Introducción
- 2 Pasos para crear una primera aplicación
- 3 Configurar smartphone en modo de Depuracion
- 4 Modificacion de la Interfaz de usuario
- 5 Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)
- 6 Etapa 2: Implementar el comportamiento de la aplicacion
- 7 Conclusiones

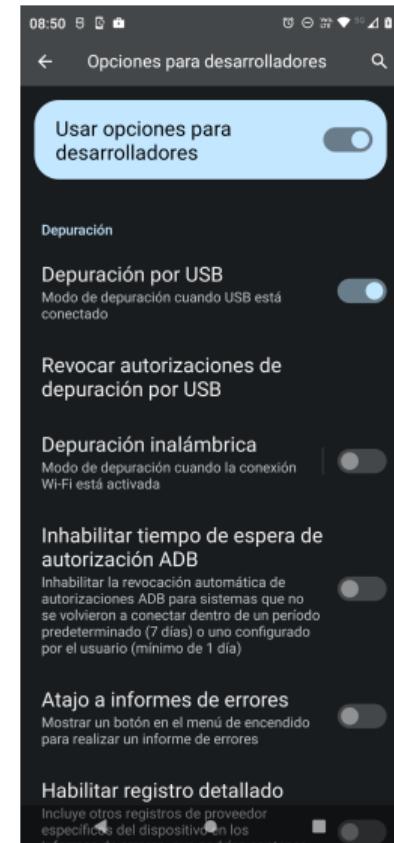
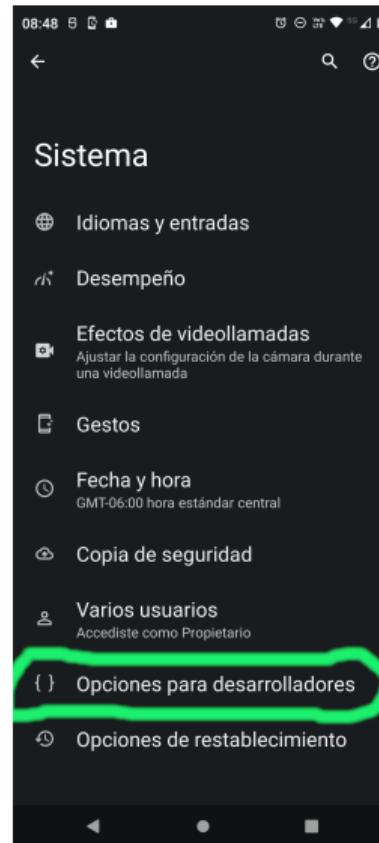
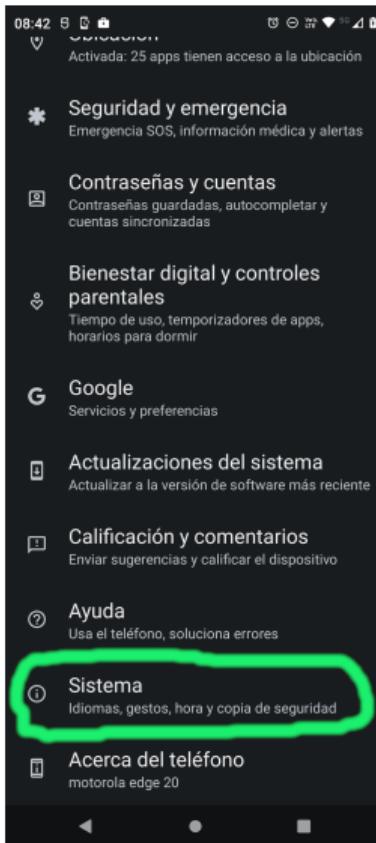
Configurar telefono inteligente en modo de desarrollador (0)

- Es necesario buscar en las opciones de configuración, en el apartado acerca del teléfono ubicar el número de compilación
- Dar 5 taps (toques) sobre el número compilación. Deberá aparecer un mensaje que diga que estás a X taps de ser un desarrollador
- Lo anterior habilita un nuevo menú en el apartado sistema dentro de opciones de configuración con título “opciones para desarrolladores”
- Debe habilitarse tanto “opciones para desarrolladores” como la opción “depuración USB”

Configurar teléfono inteligente en modo de desarrollador (1)



Configurar teléfono inteligente en modo de desarrollador (2)



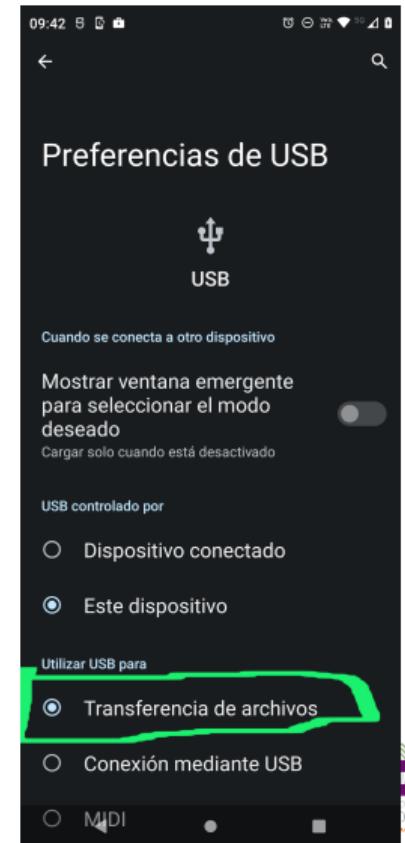
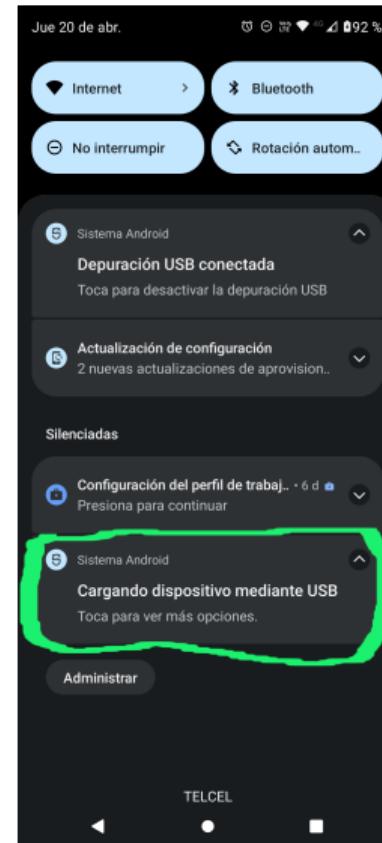
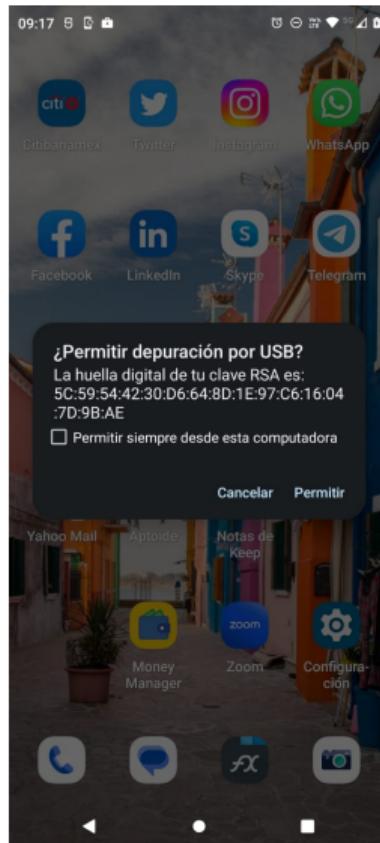
Configurar teléfono inteligente en modo de desarrollador (3)

- Al habilitar la depuración, aparece un mensaje de advertencia
- Una vez que hayas terminado tus pruebas, se recomienda deshabilitar este modo de depuración
- Para poder instalar una aplicación desarrollada con Android Studio, se debe conectar el teléfono inteligente a la computadora usando un cable USB



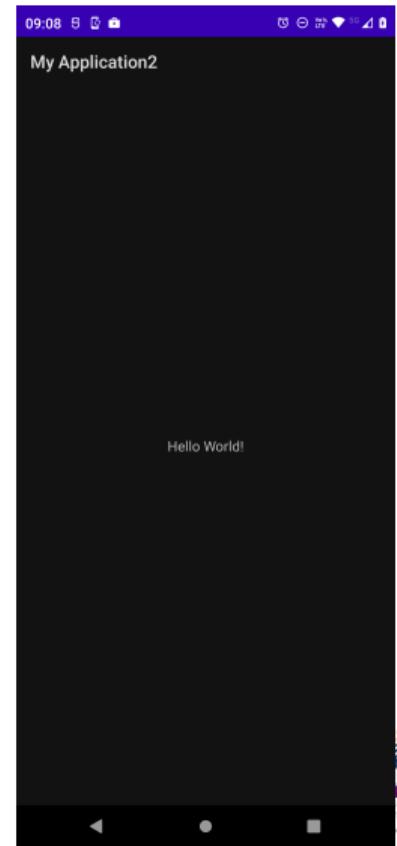
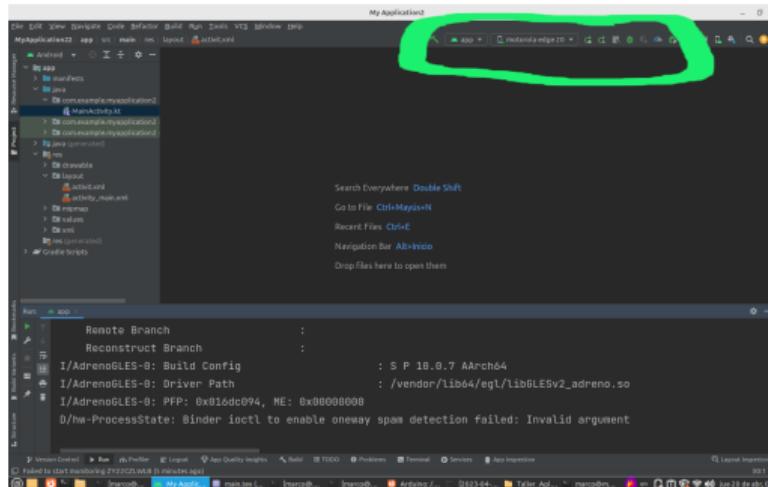
Conectar smartphone a la computadora (Cable USB)

- Al conectar el dispositivo por primera vez, aparece un mensaje de confirmación en el dispositivo
- Configurar el teléfono para que se conecte en modo de transferencia de archivos (por defecto, esta cargando)



Ejecutar proyecto en telefono inteligente

- Una vez conectado el telefono, debe aparecer el modelo en la parte superior (lado derecho) de tu proyecto de Android Studio
- Para instalar la aplicacion en el telefono, deber dar click en una flecha verde justo a un lado de nombre del telefono (sean pacientes, puede tardar un poco la primera vez)



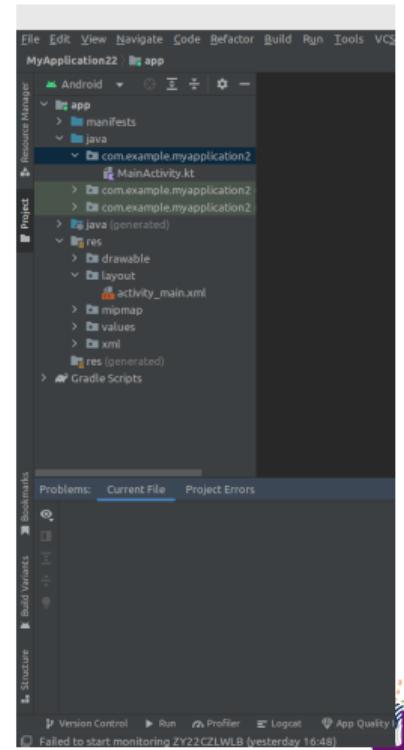
Contenido

- 1 Introducción
- 2 Pasos para crear una primera aplicación
- 3 Configurar smartphone en modo de Depuracion
- 4 Modificacion de la Interfaz de usuario
- 5 Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)
- 6 Etapa 2: Implementar el comportamiento de la aplicacion
- 7 Conclusiones

Carpetas del proyecto

Carpetas y archivos importantes

- *Manifest* - Por el momento es de interés
- *Java* - Código fuente - Dentro hay un archivo *MainActivity.kt*
- *Res* - Recursos (imágenes) y definición de interfaces. Dentro hay una carpeta llamada *layout*, con un archivo *activity_main.xml*



Agregar código a la interfaz

Apariencia

Archivo activity_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9     <TextView
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Hello World!"
13        app:layout_constraintBottom_toBottomOf="parent"
14        app:layout_constraintEnd_toEndOf="parent"
15        app:layout_constraintStart_toStartOf="parent"
16        app:layout_constraintTop_toTopOf="parent" />
17 </androidx.constraintlayout.widget.ConstraintLayout>
```

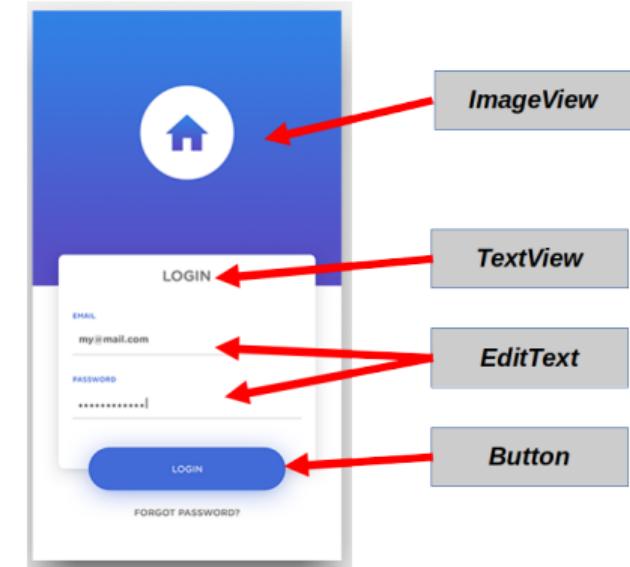
Comportamiento

Archivo MainActivity.kt

```
1 package com.example.primeraplicacionandroid
2 import androidx.appcompat.app.AppCompatActivity
3 import android.os.Bundle
4 class MainActivity : AppCompatActivity() {
5     override fun onCreate(savedInstanceState: Bundle?) {
6         super.onCreate(savedInstanceState)
7         setContentView(R.layout.activity_main)
8     }
9 }
```

Interfaz de Usuario

- Define los controles de interfaz que la aplicación muestra
- Estos controles pueden estar agrupados en contenedores
- El archivo **activity_main.xml** es la definición del layout (la organización de dichos componentes)



Primer modificación

Archivo Original **activity_main.xml**

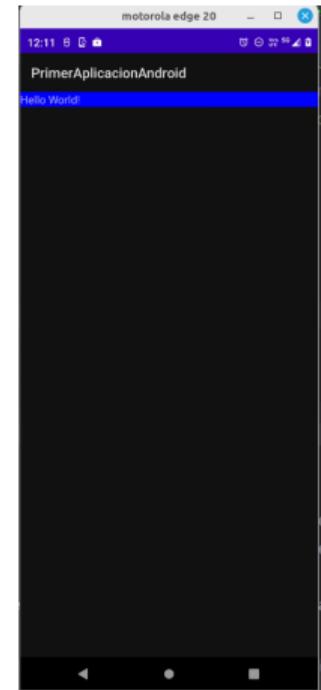
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9     <TextView
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Hello World!"
13        app:layout_constraintBottom_toBottomOf="parent"
14        app:layout_constraintEnd_toEndOf="parent"
15        app:layout_constraintStart_toStartOf="parent"
16        app:layout_constraintTop_toTopOf="parent" />
17 </androidx.constraintlayout.widget.ConstraintLayout>
```

Archivo Modificado **activity_main.xml**

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:orientation="vertical"
9     tools:context=".MainActivity">
10    <TextView
11        android:background="#00ffff"
12        android:layout_width="match_content"
13        android:layout_height="wrap_content"
14        android:text="Hello World!" />
15 </LinearLayout>
```

Primer modificación

- 1 Reemplazar *androidx.constraintlayout.widget.ConstraintLayout* por *LinearLayout*
 - Tipo de contenedor donde los componentes se agregan secuencialmente
- 2 Agregar propiedad *android:orientation=“vertical”* al *LinearLayout* principal
 - Los componentes serán ordenados verticalmente
- 3 Cambiar la propiedad *android:layout_width=“wrap_content”* por *android:layout_width=“match_parent”*
 - El ancho de *TextView* abarcar toda la pantalla
- 4 Agregar la propiedad *android:background=“#00ffff”* al *textview* -
 - La cadena “#00ffff” define el color azul. Se pueden probar con otros buscando en Internet un generador de colores hexadecimal y cambiando por otros de tu preferencia



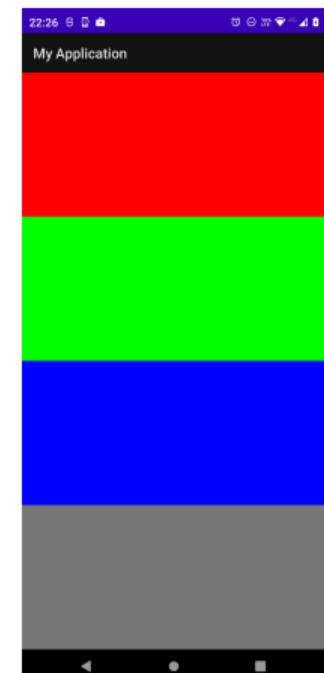
Contenido

- 1 Introducción
- 2 Pasos para crear una primera aplicación
- 3 Configurar smartphone en modo de Depuracion
- 4 Modificacion de la Interfaz de usuario
- 5 **Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)**
- 6 Etapa 2: Implementar el comportamiento de la aplicacion
- 7 Conclusiones

Fase 1: Dividir la pantalla en contenedores verticales

Archivo activity_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent" android:layout_height="match_parent"
6     android:orientation="vertical" tools:context=".MainActivity">
7     <LinearLayout android:background="#ff0000"
8         android:layout_width="match_parent" android:orientation="horizontal"
9         android:layout_height="0dp"
10        android:layout_weight="1">
11        </LinearLayout>
12        <LinearLayout android:background="#00ff00"
13            android:layout_width="match_parent" android:orientation="horizontal"
14            android:layout_height="0dp"
15            android:layout_weight="1">
16        </LinearLayout>
17        <LinearLayout android:background="#0000ff"
18            android:layout_width="match_parent" android:orientation="horizontal"
19            android:layout_height="0dp"
20            android:layout_weight="1">
21        </LinearLayout>
22        <LinearLayout android:background="#777777"
23            android:layout_width="match_parent" android:orientation="horizontal"
24            android:layout_height="0dp"
25            android:layout_weight="1">
26        </LinearLayout>
27    </LinearLayout>
```

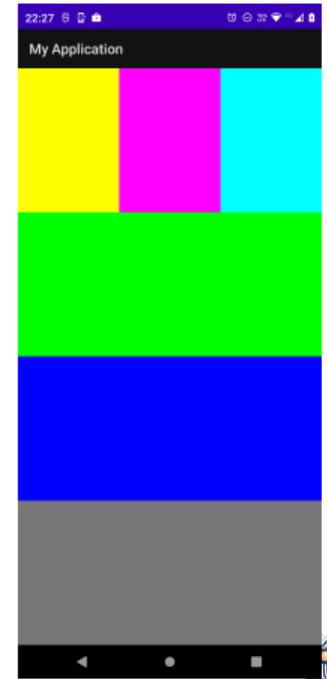


Fase 2: Agregar un contenedor horizontal dentro de uno vertical

Este fragmento de código sustituye a las líneas 7 a 10 del *activity_main.xml*

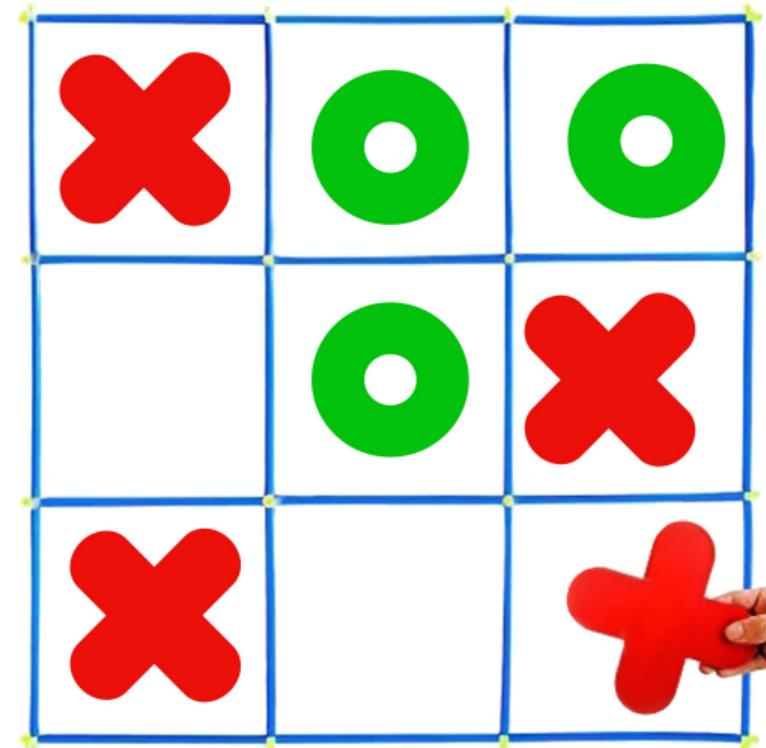
Porción del Layout que incluye tres contenedores verticales

```
1 <LinearLayout android:background="#ff0000"
2     android:layout_width="match_parent" android:orientation="horizontal"
3     android:layout_height="0dp"
4     android:layout_weight="1">
5     <LinearLayout android:background="#ffff00"
6         android:layout_width="0dp" android:orientation="vertical"
7         android:layout_height="match_parent"
8         android:layout_weight="1">
9     </LinearLayout>
10    <LinearLayout android:background="#ff00ff"
11        android:layout_width="0dp" android:orientation="vertical"
12        android:layout_height="match_parent"
13        android:layout_weight="1">
14    </LinearLayout>
15    <LinearLayout android:background="#00ffff"
16        android:layout_width="0dp" android:orientation="vertical"
17        android:layout_height="match_parent"
18        android:layout_weight="1">
19    </LinearLayout>
20 </LinearLayout>
```



Aplicación TIC-TAC-TOE

- Juego para dos jugadores
- Cada jugador juega una de dos posibles combinaciones (circulo o tacha)
- El control debe decidir cuando un jugador gana o cuando hay empate
- Llevar el manejo de los turnos
- Requerimos un componente de interfaz que se comporte como un botón y que despliegue una imagen. Solución: *ImageButton*

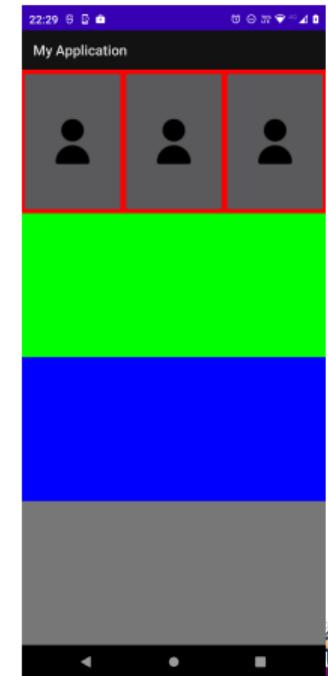


Fase 3 (Parte 1): Agregar 3 botones en orientación al contenedor horizontal al primer Contenedor

Este fragmento de código sustituye a las lineas 7 a 10 del *activity_main.xml*

Primer Layout Horizontal incluido tres *ImageButtons*

```
1 <LinearLayout android:background="#ff0000"
2     android:layout_width="match_parent"
3     android:layout_height="0dp"
4     android:layout_weight="1">
5     <ImageButton android:id="@+id/boton1_1"
6         android:src="@drawable/usuario"
7         android:layout_width="0dp"
8         android:layout_height="match_parent"
9         android:layout_weight="1"/>
10    <ImageButton android:id="@+id/boton1_2"
11        android:src="@drawable/usuario"
12        android:layout_width="0dp"
13        android:layout_height="match_parent"
14        android:layout_weight="1"/>
15    <ImageButton android:id="@+id/boton1_3"
16        android:src="@drawable/usuario"
17        android:layout_width="0dp"
18        android:layout_height="match_parent"
19        android:layout_weight="1"/>
20
21 </LinearLayout>
```



Fase 3 (Parte 2): Repetir para los otros contenedores

Este fragmento de código sustituye a las líneas 12 a 21 del *activity_main.xml*

Segundo y tercer Layouts horizontales, cada uno con tres *ImageButtons*

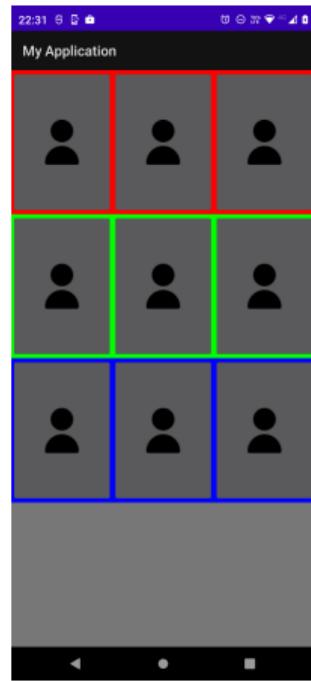
```
1 <LinearLayout android:background="#00ff00" android:layout_width="match_parent"
2   android:layout_height="0dp" android:layout_weight="1">
3     <ImageButton android:id="@+id/boton2_1"
4       android:src="@drawable/usuario" android:layout_width="0dp"
5       android:layout_height="match_parent" android:layout_weight="1"/>
6     <ImageButton android:id="@+id/boton2_2"
7       android:src="@drawable/usuario" android:layout_width="0dp"
8       android:layout_height="match_parent" android:layout_weight="1"/>
9     <ImageButton android:id="@+id/boton2_3"
10       android:src="@drawable/usuario" android:layout_width="0dp"
11       android:layout_height="match_parent" android:layout_weight="1"/>
12   </LinearLayout>
13 <LinearLayout android:background="#0000ff" android:layout_width="match_parent"
14   android:layout_height="0dp" android:layout_weight="1">
15   <ImageButton android:id="@+id/boton3_1"
16     android:src="@drawable/usuario" android:layout_width="0dp"
17     android:layout_height="match_parent" android:layout_weight="1"/>
18   <ImageButton android:id="@+id/boton3_2"
19     android:src="@drawable/usuario" android:layout_width="0dp"
20     android:layout_height="match_parent" android:layout_weight="1"/>
21   <ImageButton android:id="@+id/boton3_3"
22     android:src="@drawable/usuario" android:layout_width="0dp"
23     android:layout_height="match_parent" android:layout_weight="1"/>
24 </LinearLayout>
```

Fase 3 (Parte 3): Al ultimo contenedor agregar un TextView

Este fragmento de código sustituye a las líneas 22 a 26 del *activity_main.xml*

Último contenedor del Layout

```
1 <LinearLayout android:background="#777777"  
2     android:layout_width="match_parent"  
3     android:layout_height="0dp"  
4     android:layout_weight="1">  
5     <TextView android:id="@+id/textView_Pizarra_Estatus"  
6         android:layout_width="match_parent"  
7         android:layout_height="match_parent"  
8         android:textSize="18dp"  
9         android:gravity="center"/>  
10    </LinearLayout>
```



Código Completo

P1

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent" android:layout_height="match_parent"
6     android:orientation="vertical" tools:context=".MainActivity">
7     <LinearLayout android:background="#ff0000"
8         android:layout_width="match_parent"
9         android:layout_height="0dp"
10        android:layout_weight="1">
11            <ImageButton android:id="@+id/boton1_1"
12                android:src="@drawable/usuario"
13                android:layout_width="0dp"
14                android:layout_height="match_parent"
15                android:layout_weight="1"/>
16            <ImageButton android:id="@+id/boton1_2"
17                android:src="@drawable/usuario"
18                android:layout_width="0dp"
19                android:layout_height="match_parent"
20                android:layout_weight="1"/>
21            <ImageButton android:id="@+id/boton1_3"
22                android:src="@drawable/usuario"
23                android:layout_width="0dp"
24                android:layout_height="match_parent"
25                android:layout_weight="1"/>
26        </LinearLayout>
```

Código Completo (2)

P2

```
1 <LinearLayout android:background="#00ff00" android:layout_width="match_parent"
2     android:layout_height="0dp" android:layout_weight="1">
3     <ImageButton android:id="@+id/boton2_1"
4         android:src="@drawable/usuario" android:layout_width="0dp"
5         android:layout_height="match_parent" android:layout_weight="1"/>
6     <ImageButton android:id="@+id/boton2_2"
7         android:src="@drawable/usuario" android:layout_width="0dp"
8         android:layout_height="match_parent" android:layout_weight="1"/>
9     <ImageButton android:id="@+id/boton2_3"
10        android:src="@drawable/usuario" android:layout_width="0dp"
11        android:layout_height="match_parent" android:layout_weight="1"/>
12 </LinearLayout>
13 <LinearLayout android:background="#0000ff" android:layout_width="match_parent"
14     android:layout_height="0dp" android:layout_weight="1">
15     <ImageButton android:id="@+id/boton3_1"
16         android:src="@drawable/usuario" android:layout_width="0dp"
17         android:layout_height="match_parent" android:layout_weight="1"/>
18     <ImageButton android:id="@+id/boton3_2"
19         android:src="@drawable/usuario" android:layout_width="0dp"
20         android:layout_height="match_parent" android:layout_weight="1"/>
21     <ImageButton android:id="@+id/boton3_3"
22         android:src="@drawable/usuario" android:layout_width="0dp"
23         android:layout_height="match_parent" android:layout_weight="1"/>
24 </LinearLayout>
```

Código Completo

P3

```
1 <LinearLayout android:background="#777777"
2     android:layout_width="match_parent"
3     android:layout_height="0dp"
4     android:layout_weight="1">
5     <TextView android:id="@+id/textView_Pizarra_Estatus"
6         android:layout_width="match_parent"
7         android:layout_height="match_parent"
8         android:textSize="18dp"
9         android:gravity="center"/>
10    </LinearLayout>
11 </LinearLayout>
```

Contenido

- 1 Introducción
- 2 Pasos para crear una primera aplicación
- 3 Configurar smartphone en modo de Depuracion
- 4 Modificacion de la Interfaz de usuario
- 5 Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)
- 6 **Etapa 2: Implementar el comportamiento de la aplicacion**
- 7 Conclusiones

Fase 1: Crear variables para la interfaz

MainActivity.kt

```
1 package com.example.myapplication
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 // imports externos que se van a ir agregado (*1)
6
7 class MainActivity : AppCompatActivity() {
8
9     // En esta sección se declaran las variables de clase (*2)
10
11    override fun onCreate(savedInstanceState: Bundle?) {
12        super.onCreate(savedInstanceState)
13        setContentView(R.layout.activity_main)
14
15        // Se deben llamar los métodos que inicializar las variables
16        // o la interfaz de usuario (*3)
17
18    }
19
20    // En esta sección se declaran las funciones de la clase (*4)
21
22 }
23
24 // Declaración de otras clases (*5)
```

Fase 1: Crear variables para la interfaz

(*1)

```
1 import android.widget.ImageButton  
2 import android.widget.TextView  
3 import android.widget.Toast
```

(*2)

```
1 lateinit var B1_1 : ImageButton  
2 lateinit var B1_2 : ImageButton  
3 lateinit var B1_3 : ImageButton  
4 lateinit var B2_1 : ImageButton  
5 lateinit var B2_2 : ImageButton  
6 lateinit var B2_3 : ImageButton  
7 lateinit var B3_1 : ImageButton  
8 lateinit var B3_2 : ImageButton  
9 lateinit var B3_3 : ImageButton  
10 lateinit var TextViewPizarra : TextView  
11 var Turno : Int = 1 // 1-Turno Cruz, 2-Turno Corazon  
12 var numRows : Int = 3  
13 var numCols : Int = 3  
14 lateinit var matrix: Array<Array<CeldaGato?>>  
15  
16 var ConteoCruces : Int = 0 // # de Celdas Marcadas Cruz  
17 var ConteoCorazones : Int = 0 // de Celdas Marcadas Corazon
```

(*3)

```
1 InicializarBotonesConIds()  
2 AsignarListenerABotones()  
3 CrearMatrizEstatus ()  
4 ReiniciarControles()
```

(*4)

```
1 fun InicializarBotonesConIds () {  
2     TextViewPizarra = findViewById(R.id.textView_Pizarra_Estatus)  
3     B1_1 = findViewById (R.id.boton1_1)  
4     B1_2 = findViewById (R.id.boton1_2)  
5     B1_3 = findViewById (R.id.boton1_3)  
6     B2_1 = findViewById (R.id.boton2_1)  
7     B2_2 = findViewById (R.id.boton2_2)  
8     B2_3 = findViewById (R.id.boton2_3)  
9     B3_1 = findViewById (R.id.boton3_1)  
10    B3_2 = findViewById (R.id.boton3_2)  
11    B3_3 = findViewById (R.id.boton3_3)  
12 }
```

Fase 1: Agregar un Listener

(*4)

```
1 fun AsignarListenerABotones () {
2     B1_1.setOnClickListener (btnListener)
3     B1_2.setOnClickListener (btnListener)
4     B1_3.setOnClickListener (btnListener)
5     B2_1.setOnClickListener (btnListener)
6     B2_2.setOnClickListener (btnListener)
7     B2_3.setOnClickListener (btnListener)
8     B3_1.setOnClickListener (btnListener)
9     B3_2.setOnClickListener (btnListener)
10    B3_3.setOnClickListener (btnListener)
11 }
```

(*4)

```
1 val btnListener = View.OnClickListener {
2     val (fila, columna) = ObtieneFilaColumna (it)
3     Toast.makeText(applicationContext,
4         "Presionaste Boton ["+fila+","+columna+"]"+
5             matrix[fila][columna]!!.deviceStatus,
6             Toast.LENGTH_SHORT).show()
7     TextViewPizarra.text = "Alguna informacion util"
8 }
```

(*4)

```
1 fun CrearMatrizEstatus() {
2     matrix = Array(numRows) { row ->
3         Array(numCols) { col ->
4             CeldaGato(row, col)
5         }
6     }
7 }
```

(*4)

```
1 fun ObtieneFilaColumna (it : View): Pair<Int, Int> {
2     var fila = 0;
3     var columna = 0
4     when (it.id) {
5         R.id.boton1_1 -> { fila = 1; columna = 1 }
6         R.id.boton1_2 -> { fila = 1; columna = 2 }
7         R.id.boton1_3 -> { fila = 1; columna = 3 }
8         R.id.boton2_1 -> { fila = 2; columna = 1 }
9         R.id.boton2_2 -> { fila = 2; columna = 2 }
10        R.id.boton2_3 -> { fila = 2; columna = 3 }
11        R.id.boton3_1 -> { fila = 3; columna = 1 }
12        R.id.boton3_2 -> { fila = 3; columna = 2 }
13        R.id.boton3_3 -> { fila = 3; columna = 3 }
14        else -> { fila = -1; columna = -1 }
15    }
16    return Pair(fila-1, columna-1)
17 }
18 }
```

Fase 1: Reiniciar Controles a su Estado Inicial (1)

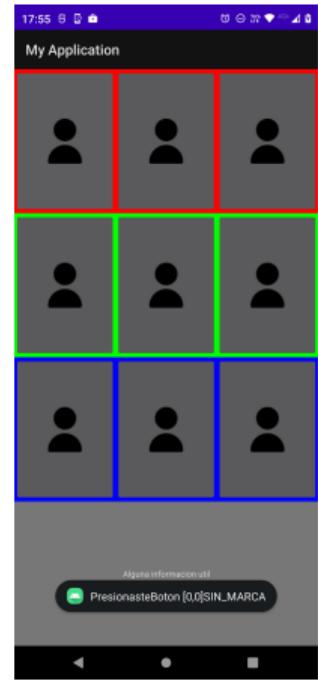
(*4)

```
1 fun ReiniciarControles() {
2     for (i in 0..numRows - 1) {
3         for (j in 0..numCols - 1) {
4             matrix[i][j]!!.deviceStatus = "SIN_MARCA"
5         }
6     }
7     B1_1.setImageResource(R.drawable.usuario)
8     B1_2.setImageResource(R.drawable.usuario)
9     B1_3.setImageResource(R.drawable.usuario)
10    B2_1.setImageResource(R.drawable.usuario)
11    B2_2.setImageResource(R.drawable.usuario)
12    B2_3.setImageResource(R.drawable.usuario)
13    B3_1.setImageResource(R.drawable.usuario)
14    B3_2.setImageResource(R.drawable.usuario)
15    B3_3.setImageResource(R.drawable.usuario)
16    Turno = 1 // 1 - Turno Cruz, 2 - Turno Corazon
17    ConteoCruces = 0 // Numero de Celdas MArcadas con Cruz
18    ConteoCorazones = 0 // Numero de Celdas Marcadas con Gato
19
20    TextViewPizarra.text = "Inicio del Juego: " +
21        "\n Conteo Cruces: " + ConteoCruces +
22        "\n Conteo Corazones: " + ConteoCorazones
23 }
```

Fase 1: Agregar un Listener

(*5)

```
1 class CeldaGato (val row: Int, val col: Int) {  
2     var deviceStatus = "SIN_MARCA"  
3     constructor(row: Int, col: Int, statusCode: Int)  
4         : this(row, col) {  
5         deviceStatus = when (statusCode) {  
6             0 -> "TACHA"  
7             1 -> "CORAZON"  
8             else -> "SIN_MARCA"  
9         }  
10    }  
11 }  
12 }
```



- Hasta este punto, la aplicación responde con un mensaje en cada celda seleccionada, con el numero de fila y columna en la cual se ubica

Fase 2: Cambiar Estado de la Casilla (1)

(*4)

```
1 fun CambiarEstadoCasilla(b: ImageButton, fila: Int, columna: Int) {
2     if (matrix[fila][columna]!!.deviceStatus.equals("SIN_MARCA")) {
3         if (Turno == 1) {
4             b.setImageResource(R.drawable.cruz)
5             matrix[fila][columna]!!.deviceStatus = "CRUZ"
6             ConteoCruces += 1
7         } else {
8             b.setImageResource(R.drawable.corazon)
9             matrix[fila][columna]!!.deviceStatus = "CORAZON"
10            ConteoCorazones += 1
11        }
12        if (Turno == 1) Turno = 2 else Turno = 1
13        ActualizarEstatusTablero(fila, columna)
14    }
15 }
16 }
```

(*4)

```
1 fun ActualizarEstatusTablero(fila: Int, columna: Int) {
2     TextViewPizarra.text = "Ultimo Turno: " +
3         matrix[fila][columna]!!.deviceStatus +
4         "\n Conteo Cruces: " + ConteoCruces +
5         "\n Conteo Corazones: " + ConteoCorazones
6 }
```

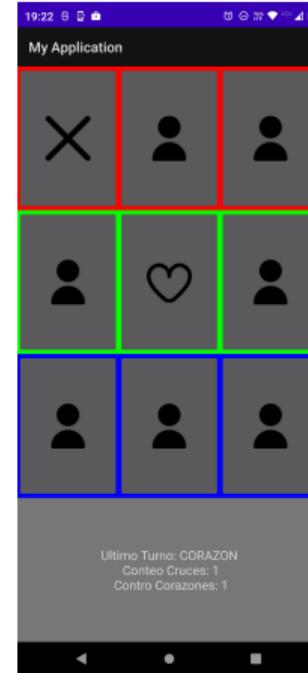
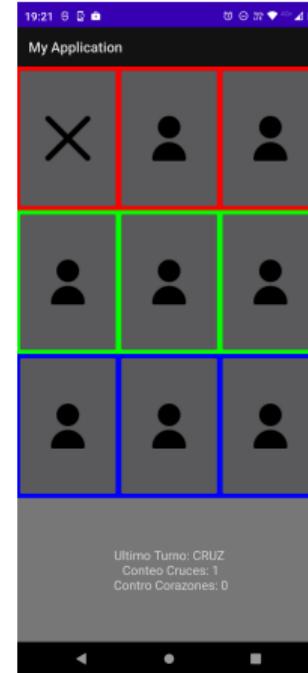
Fase 2: Cambiar Estado de la Casilla

(*1)

```
1 import android.view.View  
2 import android.widget.ImageButton  
3 import android.widget.TextView  
4 import android.widget.Toast  
5 import android.widget.Toast  
6 import androidx.appcompat.app.AlertDialog
```

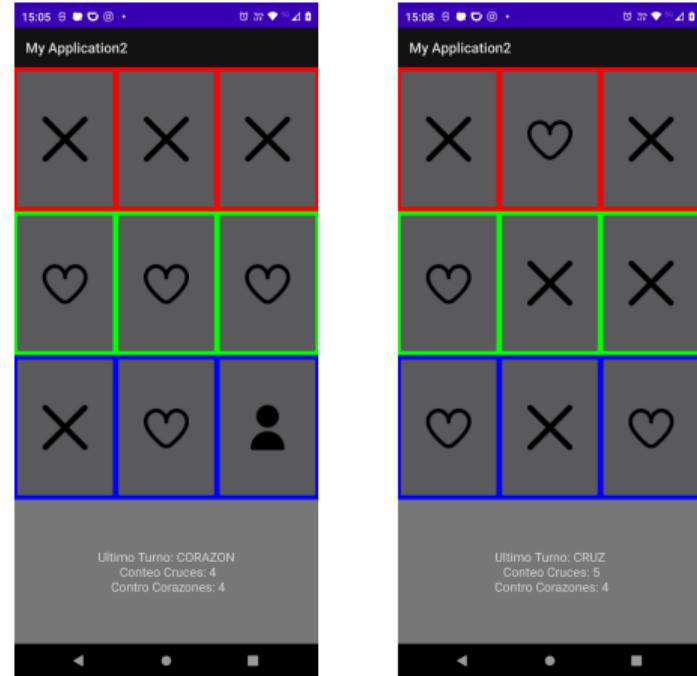
(*4) - Reemplazar

```
1 val btnListener = View.OnClickListener {  
2     val (fila, columna) = ObtieneFilaColumna (it)  
3     //Toast.makeText(applicationContext,  
4     //    "PresionasteBoton ["+fila+","+columna+"]"+  
5     //    matrix[fila][columna]!!.deviceStatus,  
6     //    Toast.LENGTH_SHORT).show()  
7     //TextViewPizarra.text = "Alguna informacion util"  
8  
9     val b: ImageButton = findViewById(it.id)  
10    CambiarEstadoCasilla (b, fila, columna)  
11 }  
12 }
```



Problemas pendientes de resolver

- 1 No detecta al ganador
- 2 Permite continuar el juego aun habiendo ganado uno (o dos)
- 3 No detecta el empate



Fase 3: Checar Ganador (1)

(*4)

```
1 fun MostrarAlertDialog(C: String) {
2     val builder = AlertDialog.Builder(this)
3     builder.setTitle("Juego Finalizado")
4     builder.setMessage(C)
5     builder.setCancelable(false)
6     builder.setPositiveButton("Reiniciar") { dialog, which ->
7         Toast.makeText(applicationContext, android.R.string.yes, Toast.LENGTH_SHORT).show()
8         ReiniciarControles()
9     }
10    builder.setNegativeButton("Salir") { dialog, which ->
11        Toast.makeText(applicationContext, android.R.string.no, Toast.LENGTH_SHORT).show()
12        finish()
13    }
14    builder.show()
15 }
```

Fase 3: Checar Ganador (2)

(*4)

```
1 fun ChecarGanadorPorFilas(): Triple<Boolean, Int, String> {
2     // Recorrido por filas
3     var Pivote: String = ""
4     for (i in 0..numRows - 1) {
5         Pivote = matrix[i][0]!!.deviceStatus
6         for (j in 1..numCols - 1) {
7             if (matrix[i][j]!!.deviceStatus.equals(Pivote))
8                 Pivote = matrix[i][j]!!.deviceStatus
9             else
10                Pivote = "NO"
11         }
12         if (Pivote.equals("CRUZ") || Pivote.equals("CORAZON")) {
13             return Triple(true, i, Pivote)
14         }
15     }
16     return Triple(false, -1, "No");
17 }
```

Fase 3: Checar Ganador (3)

(*4)

```
1 fun ChecarGanadorPorColumnas(): Triple<Boolean, Int, String> {
2     // Recorrido por columnas
3     var Pivote: String = ""
4     for (i in 0..numCols - 1) {
5         Pivote = matrix[0][i]!!.deviceStatus
6         for (j in 1..numRows - 1) {
7             if (matrix[j][i]!!.deviceStatus.equals(Pivote))
8                 Pivote = matrix[j][i]!!.deviceStatus
9             else
10                Pivote = "No"
11         }
12         if (Pivote.equals("CRUZ") || Pivote.equals("CORAZON")) {
13             return Triple(true, i, Pivote)
14         }
15     }
16     return Triple(false, -1, "NO");
17 }
```

Fase 3: Checar Ganador (4)

(*4)

```
1 fun ChecarGanadorPorDiagonales(): Triple<Boolean, Int, String> {
2     var Pivote: String
3     Pivote = matrix[0][0]!!.deviceStatus
4     for (i in 1..numRows - 1) {
5         if (matrix[i][i]!!.deviceStatus.equals(Pivote))
6             Pivote = matrix[i][i]!!.deviceStatus
7         else
8             Pivote = "NO"
9     }
10    if (Pivote.equals("CRUZ") || Pivote.equals("CORAZON")) {
11        return Triple(true, 0, Pivote)
12    }
13    var Col = 2
14    Pivote = matrix[0][Col]!!.deviceStatus
15    for (i in 1..numRows - 1) {
16        Col -= 1
17        if (matrix[i][Col]!!.deviceStatus.equals(Pivote))
18            Pivote = matrix[i][Col]!!.deviceStatus
19        else
20            Pivote = "NO"
21    }
22    if (Pivote.equals("CRUZ") || Pivote.equals("CORAZON")) {
23        return Triple(true, 1, Pivote)
24    }
25    return Triple(false, -1, "NO");
26 }
```

Fase 3: Checar Ganador (5)

(*4)

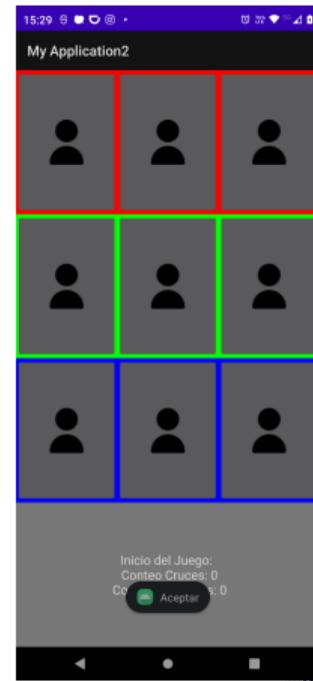
```
1 fun ChecarAlgunGanadorFin(fila: Int, columna: Int): Boolean {
2     if ((ConteoCorazones > 2) || (ConteoCruces > 2)) {
3         // Checa si hay fila ganadora
4         var (Ganador, Indice, Jugador) = ChecarGanadorPorFilas()
5         if (Ganador) { // Hay una fila ganadora
6             TextViewPizarra.text = "Gano: " + Jugador + " Fila: " + Indice
7             MostrarAlertDialog("Ganador: " + Jugador)
8             return (true)
9         } else { // Checa si hay columna ganadora
10            var (Ganador2, Indice2, Jugador2) = ChecarGanadorPorColumnas()
11            if (Ganador2) { // Hay una columna ganadora
12                MostrarAlertDialog("Ganador: " + Jugador2)
13                TextViewPizarra.text = "Gano: " + Jugador2 + " Columna: " + Indice2
14                return true
15            } else {
16                var (Ganador3, Indice3, Jugador3) = ChecarGanadorPorDiagonales()
17                if (Ganador3) { // Ganador por alguna columna
18                    MostrarAlertDialog("Ganador: " + Jugador3)
19                    TextViewPizarra.text = "Gano: " + Jugador3 + " Diagonal: " + Indice3
20                    return true
21                } else
22                    return (false)
23            }
24        }
25    } else {
26        ActualizarEstatusTablero(fila, columna)
27    }
28    return false
29 //return false
```

Fase 3: Checar Ganador (6)

(*4) - Actualizar

```
1  val btnListener = View.OnClickListener {  
2      val (fila, columna) = ObtieneFilaColumna (it)  
3      //Toast.makeText(applicationContext,  
4      //    "PresionasteBoton ["+fila+","+columna+"]"+  
5      //    matrix[fila][columna]!!.deviceStatus,  
6      //    Toast.LENGTH_SHORT).show()  
7      //TextViewPizarra.text = "Alguna informacion util"  
8  
9      val b: ImageButton = findViewById(it.id)  
10     CambiarEstadoCasilla (b, fila, columna)  
11     ChecarAlgunGanadorFin(fila, columna)  
12 }  
13 }
```

¿Qué problema falta por resolverse?



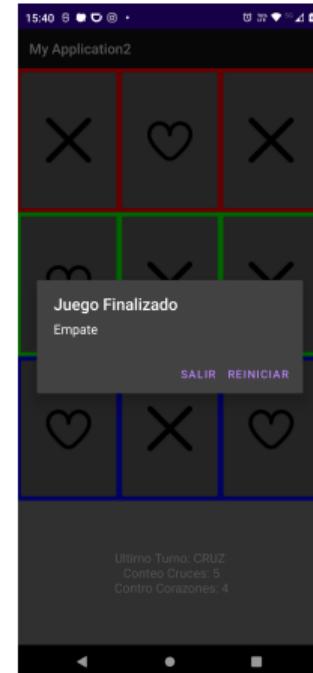
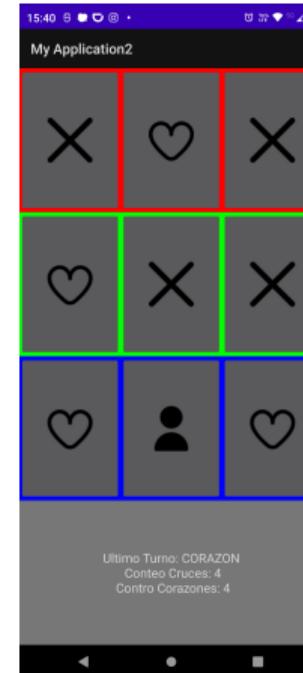
Fase 4: Checar Empate

(*4)

```
1 fun ChecarEmpate() {  
2     // Numero de Celdas Marcadas con Gato  
3     if (ConteoCruces + ConteoCorazones == 9) {  
4         MostrarAlertDialog("Empate")  
5     }  
6 }  
7
```

(*4) - Actualizar

```
1 val btnListener = View.OnClickListener {  
2     val (fila, columna) = ObtieneFilaColumna (it)  
3     //Toast.makeText(applicationContext,  
4     //    "Presionaste boton ["+fila+","+columna+"]"+  
5     //    matrix[fila][columna]!!.deviceStatus,  
6     //    Toast.LENGTH_SHORT).show()  
7     //TextViewPizarra.text = "Alguna informacion util"  
8  
9     val b: ImageButton = findViewById(it.id)  
10    CambiarEstadoCasilla (b, fila, columna)  
11    if (!ChecarAlgunGanadorFin(fila, columna))  
12        ChecarEmpate()  
13 }  
14
```



Contenido

- 1 Introducción
- 2 Pasos para crear una primera aplicación
- 3 Configurar smartphone en modo de Depuracion
- 4 Modificacion de la Interfaz de usuario
- 5 Etapa 1: Definir Apariencia de la Aplicacion (Interfaz de usuario)
- 6 Etapa 2: Implementar el comportamiento de la aplicacion
- 7 Conclusiones

- En este taller describimos conceptos fundamentales de programación, computadoras y sistemas operativos
- Configuramos un teléfono inteligente en modo de depuración
- Creamos una aplicación vacía en Android Studio, instalar y ejecutar tal aplicación en el teléfono inteligente
- Comprendemos los archivos principales de la aplicación
- Modificamos la interfaz para entender el concepto de contenedor
- Modificamos la interfaz para implementar el juego de TIC-TAC-TOE
- Modificamos el comportamiento de la aplicación agregando variables, métodos y clases

Agradecimiento

Presentación y código fuente disponible en el siguiente enlace:

https://github.com/mnunom-upv/Curso_Desarrollo_APLICACIONES_Moviles_2023

Comentarios o Dudas: mnunom@upv.edu.mx