Outline

Fase 1: Crear variables para la interfaz

MainActivity.kt

```kotlin
package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
// imports externos que se van air agregado (*1)

class MainActivity : AppCompatActivity() {

    // En esta seccion se declaran las variables de clase (*2)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Se deben llamar los metodos que inicializar las variables
        // o la interfaz de usuario (*3)

    }

    // En esta seccion se declaran las funciones de la clase (*4)

}

// Declaracion de otras clases (*5)
```

Concepto de Matriz

Matriz

- Colección ordenada de valores, conformada por filas y columnas
- Los elementos pueden ser identificados por la tupla (Fila, Columna), de manera análoga a un estante con fila (entepaño) o columna
- Por convención, la primera fila o columna tiene el indice cero

Fase 1: Crear variables para la interfaz

(*1)

```
1  import android.widget.ImageButton
2  import android.widget.TextView
3  import android.widget.Toast
4  import android.view.View
```

(*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
```

(*3)

```
1
2  InicializarBotonesConIds()
3  AsignarListenerABotones()
4  CrearMatrizEstatus ()
5  ReiniciarControles()
6
```

(*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              "PresionasteBoton ["+fila+","+columna+"]"+
5                  matrix[fila][columna]!!.deviceStatus,
6              Toast.LENGTH_SHORT).show()
```

(*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 }
```

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

(*4)

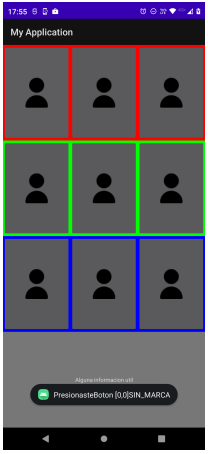
```
1   fun ReiniciarControles() {
2           for (i in 0..numRows - 1) {
3               for (j in 0..numCols - 1) {
4                   matrix[i][j]!!.deviceStatus = "SIN_MARCA"
5               }
6           }
7           B1_1.setImageResource(R.drawable.usuario)
8           B1_2.setImageResource(R.drawable.usuario)
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 Contro 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)

```
fun CambiarEstadoCasilla(b: ImageButton, fila: Int, columna: Int) {
    if (matrix[fila][columna]!!.deviceStatus.equals("SIN_MARCA")) {
        if (Turno == 1) {
            b.setImageResource(R.drawable.cruz)
            matrix[fila][columna]!!.deviceStatus = "CRUZ"
            ConteoCruces += 1
        } else {
            b.setImageResource(R.drawable.corazon)
            matrix[fila][columna]!!.deviceStatus = "CORAZON"
            ConteoCorazones += 1
        }
        if (Turno == 1) Turno = 2 else Turno = 1
        ActualizarEstatusTablero(fila, columna)
    }
}

```

(*4)

```
fun ActualizarEstatusTablero(fila: Int, columna: Int) {
    TextViewPizarra.text = "Ultimo Turno: " +
            matrix[fila][columna]!!.deviceStatus +
            "\n Conteo Cruces: " + ConteoCruces +
```

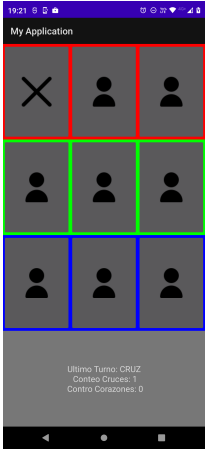
Fase 2: Cambiar Estado de la Casilla

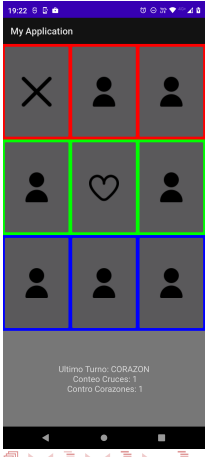
(*1)

```
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

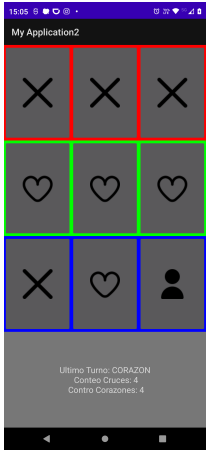
```
1  val btnListener = View.OnClickListener {
2      val (fila, columna) = ObtieneFilaColumna (it)
3      //Toast.makeText(applicationContext,
4      //     "PresionasteBoton ["+fila+","+columna+"]"+
5      //         matrix[fila][columna]!!.deviceStatus,
6      //     Toast.LENGTH_SHORT).show()
7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

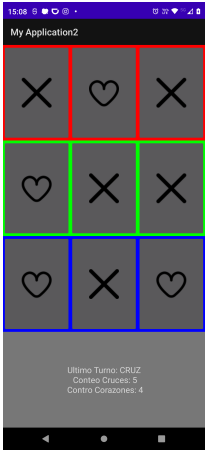




Problemas pendientes de resolver

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");
```

Fase 3: Checar Ganador (5)

(*4)

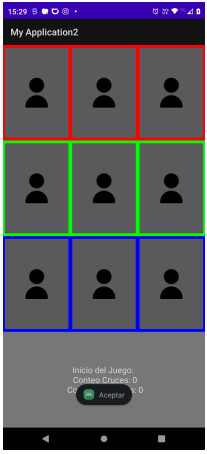
```
1  fun ChecarAlgunGanadorFin(fila: Int, columna: Int): Boolean {
2      if ((ConteoCorazones > 2) || (ConteoCruces > 2)) {          // Checa si hay fila ganadora
3          var (Ganador, Indice, Jugador) = ChecarGanadorPorFilas()
4          if (Ganador) { // Hay una fila ganadora
5              TextViewPizarra.text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

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)
0      CambiarEstadoCasilla (b, fila, columna)
1      ChecarAlgunGanadorFin(fila, columna)
2  }
3
```

¿Qué problema falta por resolverse?



Fase 4: Checar Empate

(*4)

```
fun ChecarEmpate() {
    // Numero de Celdas Marcadas con Gato
    if (ConteoCruces + ConteoCorazones == 9) {
        MostrarAlertDialog("Empate")
    }
}
```

(*4) - Actualizar

```
val btnListener = View.OnClickListener {
    val (fila, columna) = ObtieneFilaColumna (it)
    //Toast.makeText(applicationContext,
    //    "PresionasteBoton ["+fila+","+columna+"]"+
    //        matrix[fila][columna]!!.deviceStatus,
    //    Toast.LENGTH_SHORT).show()
    //TextViewPizarra.text = "Alguna informacion util"

    val b: ImageButton = findViewById(it.id)
    CambiarEstadoCasilla (b, fila, columna)
    if (!ChecarAlgunGanadorFin(fila, columna))
        ChecarEmpate()
}
```