UNIVERSIDAD POLITÉCNICA
DE TECÁMAC

APLICACIÓN MOVIL “JUEGO DEL GATO”

JULIA YAMILE GONZALEZ LUNA
DIEGO ARMANDO OCHOA MARTINEZ

PROGRAMACION PARA MOVILES II

MATRICULAS: 1319104687, 1319104600

GRUPO: 1922IS

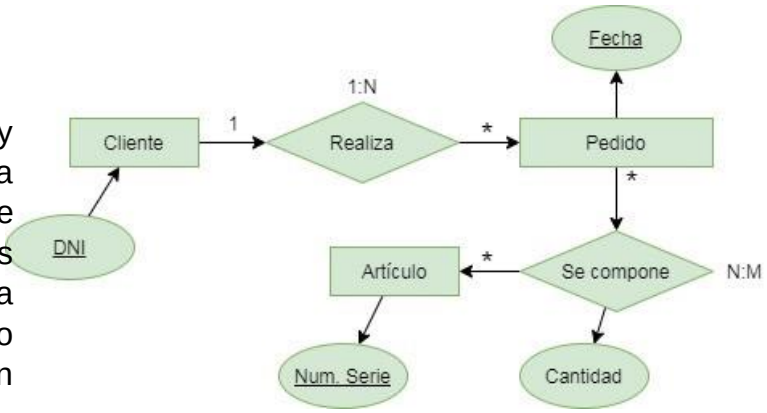
MATRO: EMMANUEL TORRES SERVIN

Ingenieria en software | 28/05/22

MODELADO DE DATOS

¿Qué es el modelado de datos?

El modelado de datos es el proceso de analizar y definir todos los diferentes datos que su empresa recopila y produce, así como las relaciones entre esos bits de datos. El proceso de modelado de sus datos crea una representación visual de datos a medida que se utilizan en su negocio, y el propio proceso es un ejercicio de conocimiento y aclaración de sus requisitos de datos.



Por qué es importante el modelado

Al modelar sus datos, documentará los datos que tiene, cómo los usa y cuáles son sus requisitos relacionados con el uso, la protección y la gobernanza. Mediante el modelado de datos, su organización:

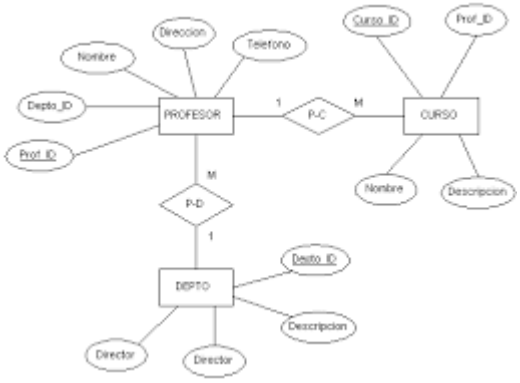
- Crea una estructura para la colaboración entre su equipo de TI y sus equipos comerciales.
- Expone oportunidades para mejorar los procesos comerciales, al definir las necesidades y los usos de los datos.
- Ahorra tiempo y dinero en TI y en inversiones en procesos, mediante una planificación adecuada por adelantado.
- Reduce los errores (y la entrada de datos redundantes propensa a errores), al tiempo que mejora la integridad de los datos.
- Aumenta la velocidad y el rendimiento de la recuperación y el análisis de datos, al planificar la capacidad y el crecimiento.

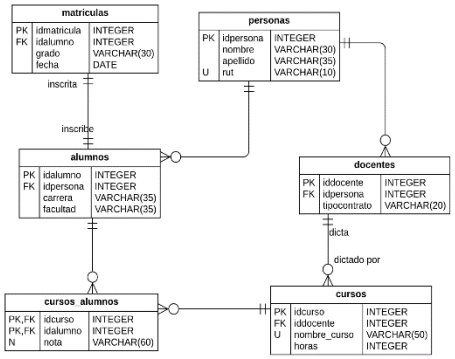
Por lo tanto, no se trata solo de lo que obtiene con el modelado de datos, sino también de cómo lo obtiene. El propio proceso ofrece importantes ventajas.

Ejemplos de modelado de datos

Conceptual

Un modelo de datos conceptual define la estructura general de su negocio y sus datos. Se utiliza para organizar conceptos de negocio, según la definición de las partes interesadas en el negocio y los arquitectos de datos. Por ejemplo, puede tener datos de clientes, empleados y productos, y cada uno de esos depósitos de datos, conocidos como entidades, tiene relaciones con otras entidades. Tanto las entidades como las relaciones entre entidades se definen en su modelo conceptual.



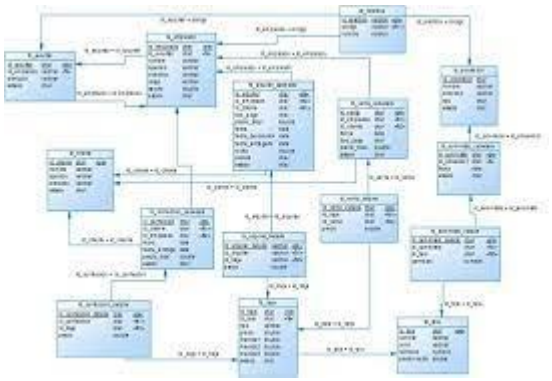


Físico

Un modelo de datos físicos es su implementación específica del modelo de datos lógicos, y lo crean los administradores de la base de datos y los desarrolladores. Está desarrollado para una herramienta de base de datos específica o tecnología de almacenamiento de datos, y con conectores de datos para entregar los datos en todos los sistemas de negocio a los usuarios según sea necesario. Esta es la “cosa” a la que han estado conduciendo el resto de modelos: la implementación real de su estado de datos.

Lógico

Un modelo de datos lógicos se basa en el modelo conceptual con atributos específicos de datos dentro de cada entidad y relaciones específicas entre esos atributos. Por ejemplo, el cliente A compra el producto B al asociado de ventas C. Este es su modelo técnico de las reglas y estructuras de datos definidas por los arquitectos de datos y analistas de negocios, y ayudará a tomar decisiones sobre qué modelo físico requieren sus datos y necesidades de negocio.



MANIPULACION DE DATOS EN PRM

La manipulación de datos es el proceso de cambiar o alterar datos para hacerlos más legibles y organizados. Por ejemplo, puede ordenar los datos alfabéticamente para acelerar el proceso de búsqueda de información útil

XML.

E xtensible Markup Language (XML) es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos en un formato que es a la vez legible y legible por máquina .

JSON.

JSON (/ dʒeɪsɒn / jah-soun, / dʒeɪsən / ja-hijo), o JavaScript Object Notation, es un formato estándar abierto que utiliza texto legible para transmitir objetos de datos que constan de pares atributo-valor. Se utiliza sobre todo para transmitir datos entre un servidor y aplicaciones web, como alternativa a XML.

Aunque en un principio derivado de la lengua scripting JavaScript, JSON es un formato de datos independiente del lenguaje, y el código para analizar y generar datos JSON está fácilmente disponible en una gran variedad de lenguajes de programación.

PERSISTENCIA DE DATOS EN MOVILES

La persistencia en Android consiste en tres tipos de almacenamientos con un propósito muy específico.

1. Persistencia en Android: Preferencias Compartidas o Shared Preferences

Con Shared Preferences podemos almacenar y recuperar en el formato clave-valor información como texto, booleanos y números; lo que lo convierte en potencial para almacenar configuraciones del usuario como: estilos, preferencias, etc.

Persistencia en Android: Almacenar archivos en memoria

- Este tipo de persistencia es uno de los más conocidos debido a que son soportados por la mayoría de los lenguajes de programación aparte de Java; consiste en guardar y recuperar la información en archivos; Android permite escribir y leer archivos que se encuentren ubicados en la propia Memoria Interna del dispositivo; al igual que con Shared Preferences.

Almacenar archivos en Memoria Interna

- Hay que tener en cuenta que estos archivos son guardados en la Memoria Interna del dispositivo la cual puede ser limitada en el dispositivo y puede ralentizar el mismo.

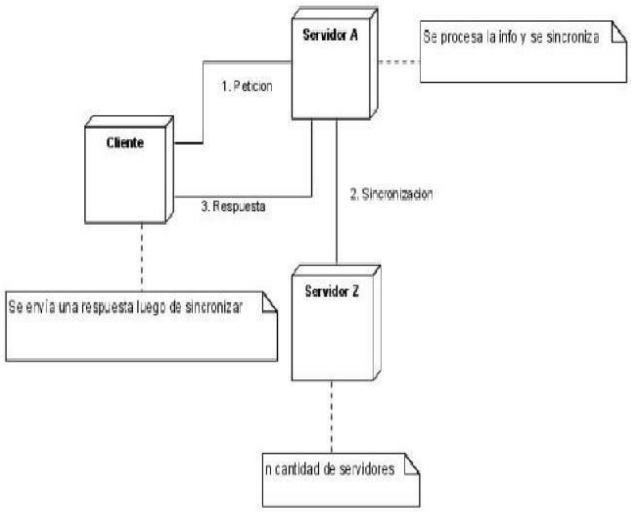
Consideraciones

- No es necesario pedir permisos especiales en el Manifest.

Sobre su uso: Apertura y Escritura de archivos en Android

- Android proporciona el método OutputStreamWriter para escribir archivos en Android a través de la apertura del mismo con el método openFileOutput; el mismo recibe como parámetro:
- El nombre del archivo.
- El modo de acceso:
- Es posible configurar los archivos para que solo puedan ser gestionados por la aplicación y por nadie ni nada más; ni siquiera el usuario; los modos de acceso son los siguientes:
    - MODE_PRIVATE: Solo es asesible por la aplicación y por nadie ni nada más (crea el archivo o lo sobrescribe si ya existe).
    - MODE_APPEND: Añade contenido a un archivo existente en el dispositivo.
    - MODE_WORLD_READABLE: Permite que otras aplicaciones puedan leer el archivo (deprecated desde el API 17).
    - MODE_WORLD_WRITEABLE: Permite que otras aplicaciones puedan escribir el archivo (deprecated desde el API 17).

TOLERANCIA A FALLOS



Tolerancias a fallos en informática se determina a la capacidad de un sistema de almacenamiento de acceder a información o al recurso aún en caso de producirse algún fallo. Esta falla puede deberse a daños físicos (mal funcionamiento) en uno o más componentes de hardware lo que produce la pérdida de información almacenada. La tolerancia a fallos requiere para su implementación que el sistema de almacenamiento guarde la misma información en más de un componente de hardware o en una máquina o dispositivo externos a modo de respaldo. De esta forma, si se produce alguna falla con una consecuente pérdida de datos, el sistema debe ser capaz de acceder a toda la información recuperando los datos faltantes desde algún respaldo disponible.

Nivel proceso

La latencia de tolerancia a fallas (FTL - Fault Tolerance Latency) es el tiempo requerido para completar todos los pasos secuenciales que se necesitan para recuperar un error. Esta definición puede volverse sumamente importante en sistemas de tiempo real, donde la latencia de cualquier política de manejo de errores (tales como detección, enmascaramiento y recuperación de errores, por ejemplo) no debe ser superior a la latencia requerida por la aplicación (ARL - Application Required Latency), que es un tiempo que depende del proceso.

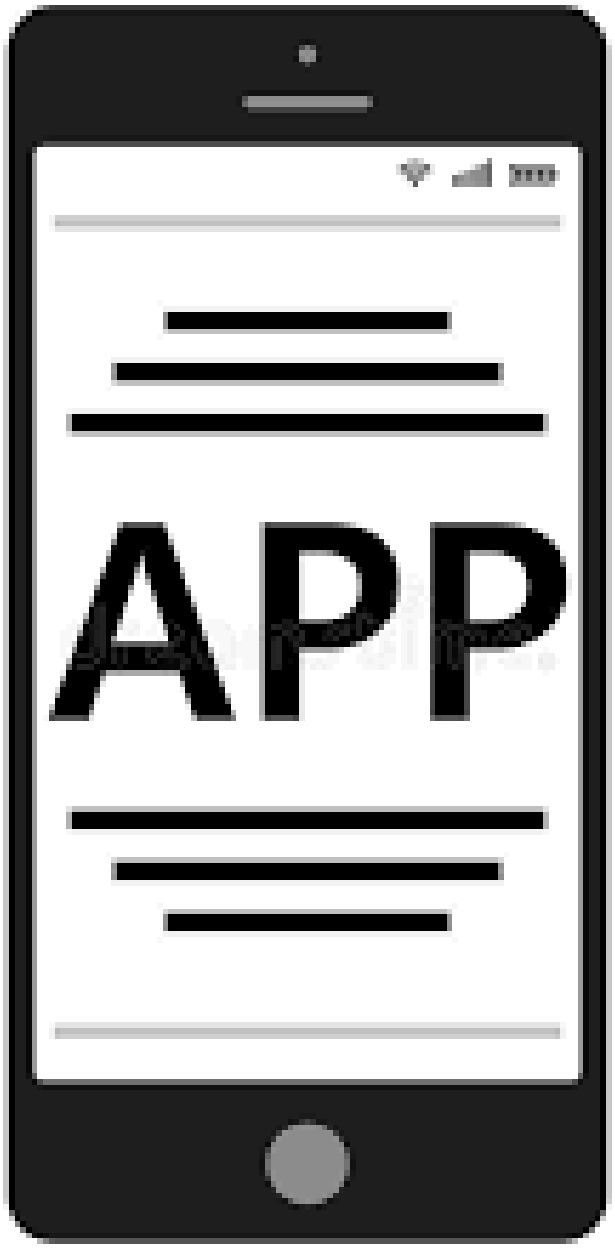
Al evaluar la FTL considerando distintos mecanismos de tolerancia a fallas, se puede utilizar dicho FTL para comprobar si una determinada estrategia de tolerancia a fallas elegida para una aplicación puede cumplir las metas (debe ser FTL ≤ ARL).

Nivel almacenamiento

Que el sistema de archivos sea tolerante a fallos implica que el Sistema de Archivos tenga un sistema de recuperación de transacciones, además de guardar varias copias de los archivos en distintas máquinas para garantizar la disponibilidad en caso de fallo del servidor principal. Además, se ha de aplicar un algoritmo que nos permita mantener todas las copias actualizadas de forma consistente, o un método alternativo que sólo nos permita acceder al archivo actualizado; como por ejemplo el invalidar el resto de copias cuando en cualquiera de ellas se vaya a realizar una operación de escritura. El uso de memorias caché para agilizar el acceso a los archivos también es recomendable, pero este caso requiere analizar con especial atención la consistencia del sistema.

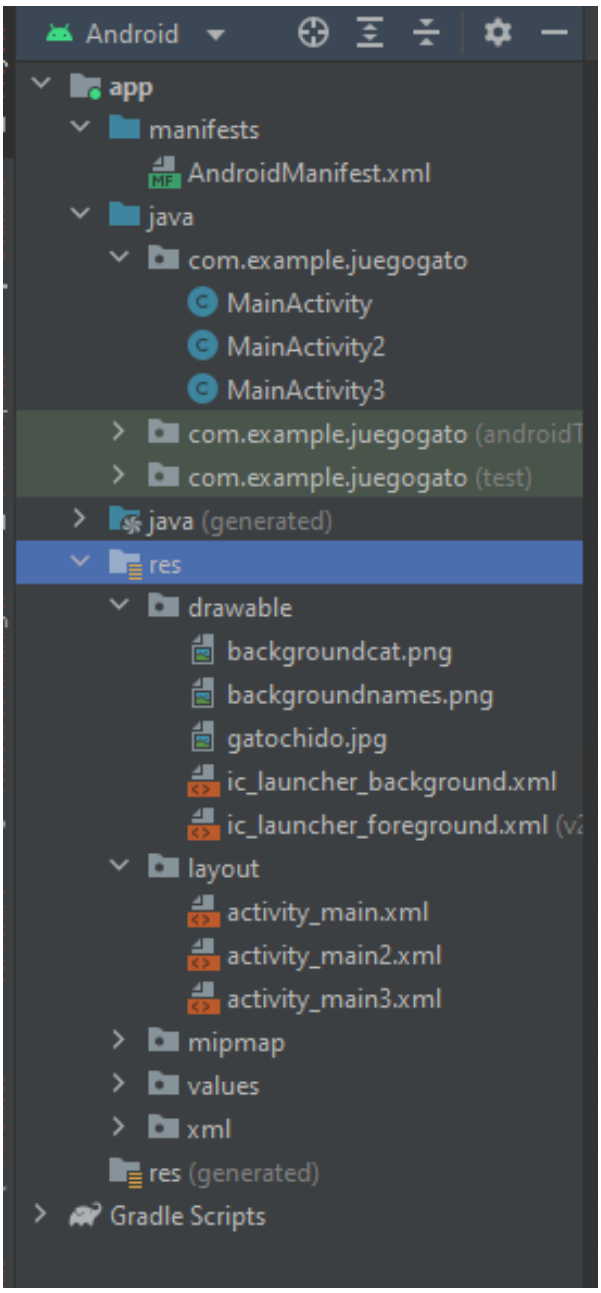
Las características fundamentales de diseño para sistemas de archivos distribuidos son:

- La utilización efectiva de la *memoria caché* en el cliente para conseguir iguales prestaciones o mejores que las de los sistemas de archivos locales.
- El mantenimiento de la consistencia entre múltiples copias de archivos en el *caché* de los clientes cuando son actualizadas.
- La recuperación después de un fallo en el servidor o en el cliente.
- El alto rendimiento en la lectura y escritura de archivos de todos los tamaños.
- La escalabilidad.



APLICACIÓN MOVIL “JUEGO DEL GATO”

Para la estructura del proyecto únicamente se tomaron en cuenta tres activitiy lo que quiere decir que esta constaría de tres pantallas para el usuario cada una con sus correspondientes funcionalidades y estilos



ACTIVITY 1

.JAVA

```java
package com.example.juegogato;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClickJugar (View view){
        Intent intent=new Intent( packageContext: MainActivity.this,MainActivity3.class);
        startActivity(intent);
        finish();
    }
}
```

.XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/gatochido"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn_ingresar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:backgroundTint="@color/pink"
        android:onClick="onClickJugar"
        android:text="Jugar Ahora"
        android:textColor="@color/black"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.472"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3"
        tools:ignore="TextContrastCheck" />
```

ACTIVITY 2

.JAVA

```java
public class MainActivity3 extends AppCompatActivity {
    public String playerOne="";
    public String playerTwo="";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
    }
    public void onClickAceptar (View view){
        EditText nameOne=(EditText) findViewById(R.id.txtName1);
        EditText nameTwo=(EditText) findViewById(R.id.txtName2);
        playerOne=nameOne.getText().toString();
        playerTwo=nameTwo.getText().toString();
        if(playerOne.isEmpty()){
            playerOne="Player 1";
        }
        if(playerTwo.isEmpty()){
            playerTwo="Player 2";
        }
        Intent intent=new Intent( packageContext: MainActivity3.this,MainActivity2.class);
        intent.putExtra( name: "PlayerOne",playerOne);
        intent.putExtra( name: "PlayerTwo",playerTwo);
        startActivity(intent);
        finish();
    }
}
```

.XML

```xml
<EditText
    android:id="@+id/txtName2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="34dp"
    android:layout_marginTop="36dp"
    android:background="@color/white"
    android:ems="10"
    android:inputType="textPersonName"
    app:layout_constraintStart_toStartOf="@+id/textView6"
    app:layout_constraintTop_toBottomOf="@+id/textView6"
    tools:ignore="TouchTargetSizeCheck,SpeakableTextPresentCheck" />

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="60dp"
    android:backgroundTint="@color/pink"
    android:onClick="onClickAceptar"
    android:text="Aceptar"
    android:textColor="@color/teal_200"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/txtName2" />
```

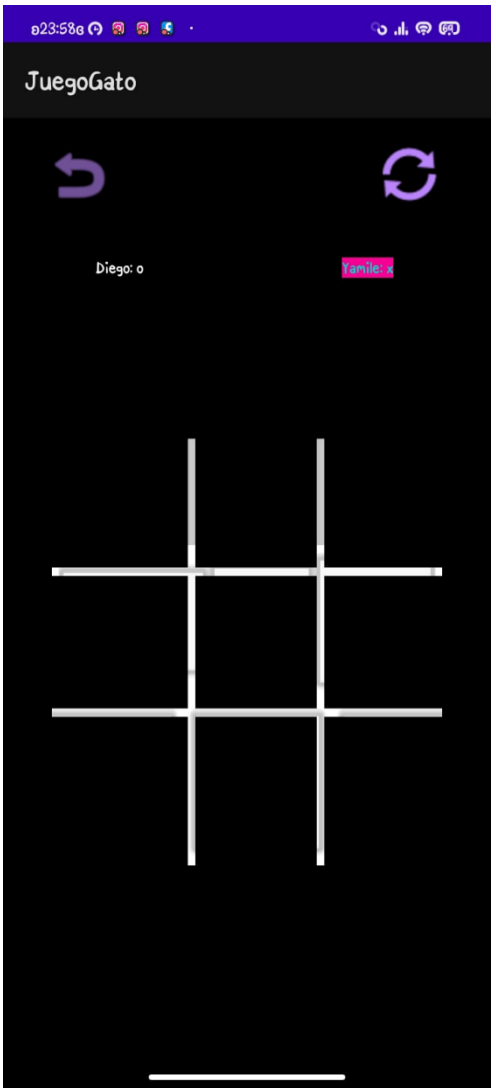
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/backgroundnames"
    tools:context=".MainActivity3">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="136dp"
        android:background="@color/black"
        android:text="Ingresa nombre jugador 1:"
        android:textColor="@color/white"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
```

ACTIVITY 3

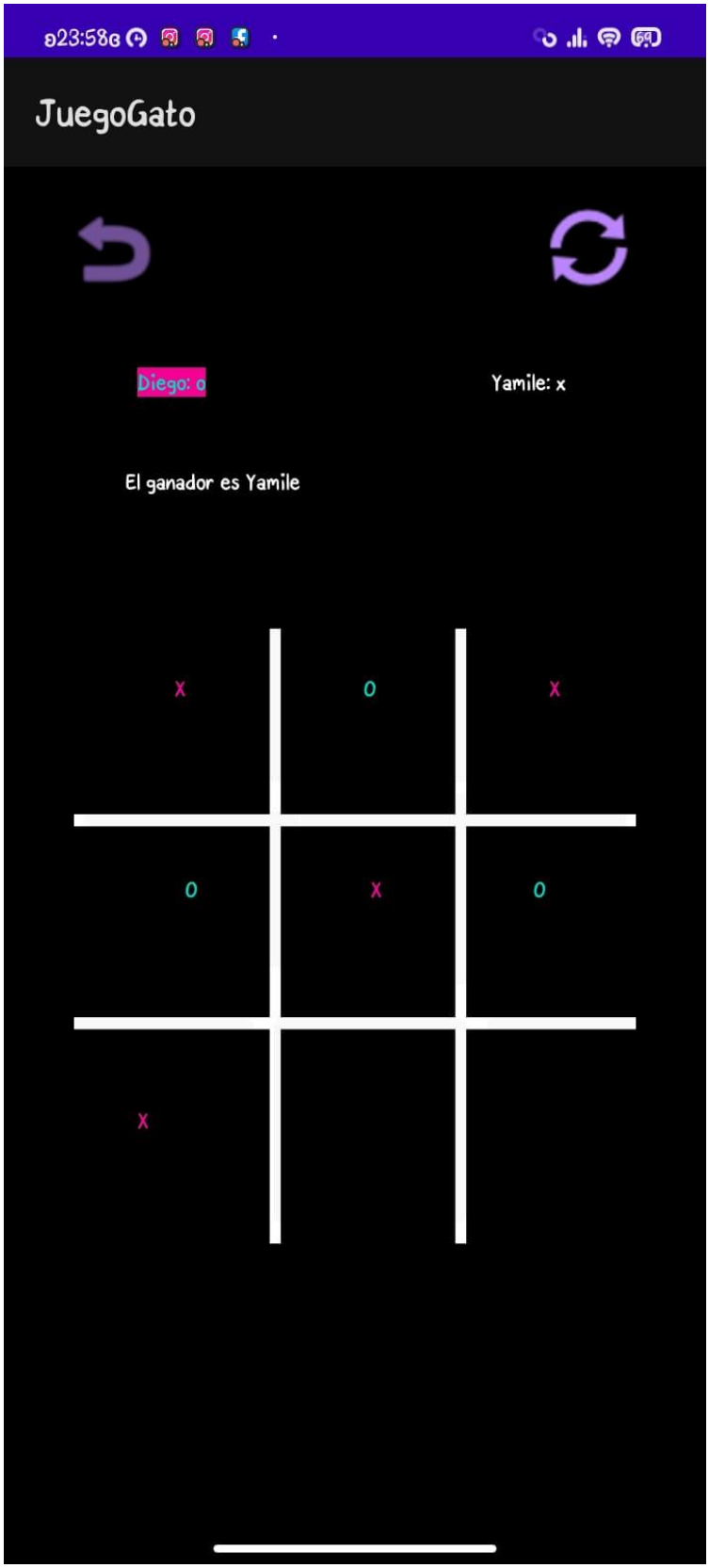
.XML

```
<Button
    android:id="@+id/btnA1"
    android:layout_width="103dp"
    android:layout_height="114dp"
    android:layout_weight="1"
    android:backgroundTint="@color/cardview_shadow_end_color"
    android:onClick="onclickA1"
    tools:ignore="SpeakableTextPresentCheck" />

<Space
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_weight="1" />

<Button
    android:id="@+id/btnA2"
    android:layout_width="101dp"
    android:layout_height="143dp"
    android:layout_weight="1"
    android:backgroundTint="@color/cardview_shadow_end_color"
    android:onClick="onclickA2"
    tools:ignore="SpeakableTextPresentCheck" />
```

```
<LinearLayout
    android:id="@+id/linearLayout3"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="41dp"
    android:layout_marginEnd="41dp"
    android:layout_marginBottom="275dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout2">
```



,JAVA


23:58
JuegoGato
Diego: o
Yamile: x
El ganador es Yamile
X
O
X
O
X
O
X