MEDAC
Instituto Oficial de Formación Profesional

Trivialist

Fernando Rodríguez-Millán Pérez

Juan Pablo Domínguez Fernández-Sambruno

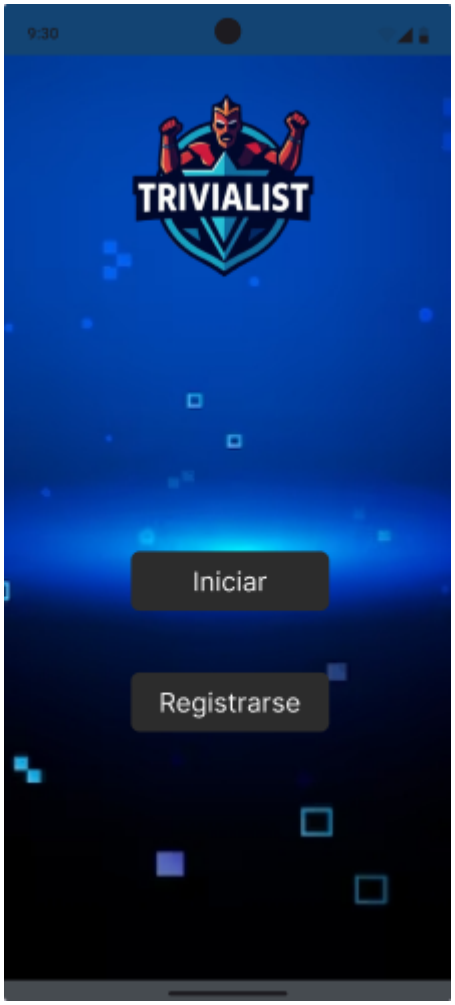
David Vázquez Vicario

Indice

Proyecto Móvil: Trivialist

1. Activity Inicio

1.1 Descripción de la Activity



Esta es la primera activity la cual encontramos el logo de nuestra app, llamada Trivialist. En esta Activity observamos dos botones, el botón Iniciar que se usaría para dirigirnos al inicio de sesión para empezar a jugar y el otro botón de Registrarse sirve para ir a la pantalla de registro. El fondo de pantalla está formado por un juego de colores azules futuristas para darle un rol más dinámico y activo.

1.2 Descripción del código

```java
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;


import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.databinding.DataBindingUtil;

import com.example.trivialist.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding= DataBindingUtil.setContentView(this,R.layout.activity_main);

        binding.registerButton.setOnClickListener(onClick(view) -> {


                Intent intent =new Intent(MainActivity.this ,activity_register.class);
                startActivity(intent);
        });
        binding.logInButton.setOnClickListener(onClick(view) -> {
                Intent intent =new Intent(MainActivity.this ,LoginActivity.class);
                startActivity(intent);
        });
```

Este código define la clase **`MainActivity`**, que representa la actividad principal de nuestra aplicación Android. Utiliza la técnica de **Data Binding** para conectar elementos de la interfaz gráfica (definidos en un archivo XML) directamente con el código, lo que simplifica la interacción con los componentes de la UI.

Clase `MainActivity`

- Extiende `AppCompatActivity`, una clase base para actividades que soportan características modernas de Android y compatibilidad con versiones antiguas.
- La actividad es responsable de manejar la lógica de navegación inicial de la aplicación.

Configuración del Data Binding

- La clase utiliza **Data Binding** para acceder a los elementos de la interfaz gráfica sin la necesidad de llamar a `findViewById`.
- El archivo de diseño asociado (`R.layout.activity_main`) genera una clase de enlace (`ActivityMainBinding`) que se utiliza para referenciar directamente los componentes definidos en el archivo XML.

Método `onCreate`

- Este método se ejecuta al iniciar la actividad y realiza las siguientes tareas:
    - Configura la vista principal de la actividad utilizando **`DataBindingUtil.setContentView`.**
    - Inicializa listeners para los botones de la interfaz gráfica.

Navegación entre Actividades

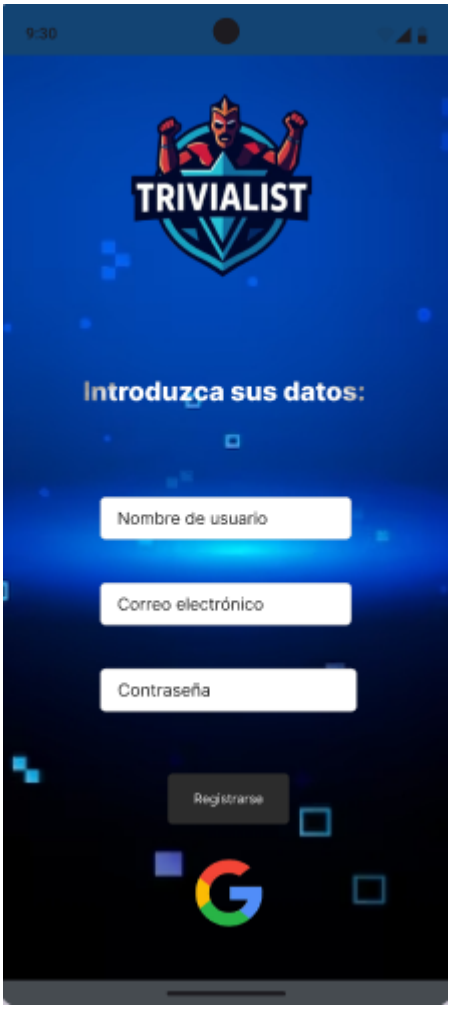
- La actividad implementa la navegación a otras pantallas utilizando `Intents`.
- Dos botones principales (**`registerButton`** y `logInButton`) permiten al usuario navegar a:
    - **`activity_register`**: Representa la pantalla de registro de usuarios.
    - **`LoginActivity`**: Representa la pantalla de inicio de sesión.

Eventos de Clic

- Se añaden listeners (**`setOnClickListener`**) para gestionar los clics en los botones:
    - Cuando el usuario hace clic en el botón de registro, se crea un `Intent` para iniciar la actividad **`activity_register`.**
    - Cuando hace clic en el botón de inicio de sesión, se inicia `LoginActivity`.
- Estos eventos definen el comportamiento interactivo de los botones y permiten una experiencia fluida para el usuario.

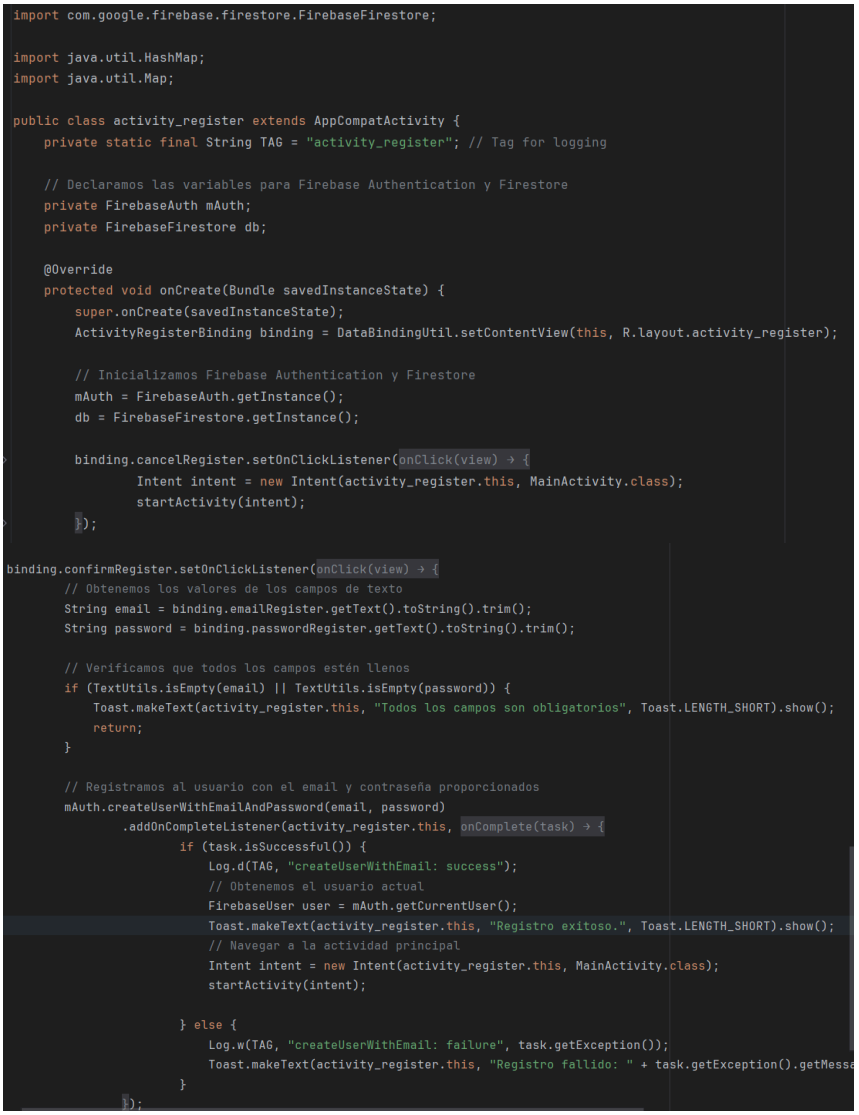
Activity Registro

2.1 Descripción de la Activity



En esta segunda Activity encontramos igual que en la anterior el logo de nuestro juego, pero lo importante es la funcionalidad de esta Activity la cual es la de el registro de nuestro nuevo usuario el cual tendrá que introducir sus respectivos datos como son; Nombre de usuario, Correo electrónico y una contraseña al rellenar todos los campos clicamos en el botón de registrarse y el nuevo usuario ya estará registrado en nuestra base de datos.

2.2 Descripción del código

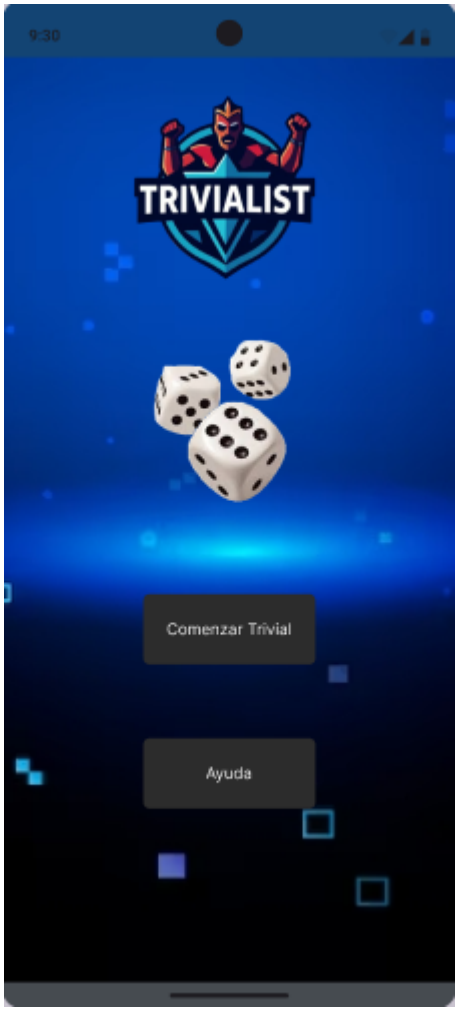


```
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;

public class activity_register extends AppCompatActivity {
    private static final String TAG = "activity_register"; // Tag for logging

    // Declaramos las variables para Firebase Authentication y Firestore
    private FirebaseAuth mAuth;
    private FirebaseFirestore db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityRegisterBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_register);

        // Inicializamos Firebase Authentication y Firestore
        mAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();

        binding.cancelRegister.setOnClickListener(onClick(view) -> {
                Intent intent = new Intent(activity_register.this, MainActivity.class);
                startActivity(intent);
        });

binding.confirmRegister.setOnClickListener(onClick(view) -> {
        // Obtenemos los valores de los campos de texto
        String email = binding.emailRegister.getText().toString().trim();
        String password = binding.passwordRegister.getText().toString().trim();

        // Verificamos que todos los campos estén llenos
        if (TextUtils.isEmpty(email) || TextUtils.isEmpty(password)) {
            Toast.makeText(activity_register.this, "Todos los campos son obligatorios", Toast.LENGTH_SHORT).show();
            return;
        }

        // Registramos al usuario con el email y contraseña proporcionados
        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(activity_register.this, onComplete(task) -> {
                        if (task.isSuccessful()) {
                            Log.d(TAG, "createUserWithEmail: success");
                            // Obtenemos el usuario actual
                            FirebaseUser user = mAuth.getCurrentUser();
                            Toast.makeText(activity_register.this, "Registro exitoso.", Toast.LENGTH_SHORT).show();
                            // Navegar a la actividad principal
                            Intent intent = new Intent(activity_register.this, MainActivity.class);
                            startActivity(intent);

                        } else {
                            Log.w(TAG, "createUserWithEmail: failure", task.getException());
                            Toast.makeText(activity_register.this, "Registro fallido: " + task.getException().getMessa
                        }
                });
```

Este código implementa una pantalla de registro de usuarios en una aplicación Android. Permite registrar un nuevo usuario utilizando su **correo electrónico** y **contraseña**, empleando **Firebase Authentication** para gestionar la autenticación. Adicionalmente, se utiliza **Data Binding** para vincular directamente los elementos del diseño con el código, lo que hace que la interacción con la interfaz gráfica sea más limpia y eficiente.

Activity Pantalla Principal

3.1 Descripción de la Activity



Esta podría decirse que es la pantalla principal o también el Lobby en este punto ya estaríamos dentro de Trivialist y nos encontramos con el logo de nuestro juego en la parte superior de nuestra pantalla, justo debajo un gif de unos dados haciendo una pequeña referencia a la esencia de este juego y por otro lado tenemos dos botones uno de Comenzar Trivial y otro de Ayuda, los cuales veremos a continuación a dónde nos llevan y para qué se usan.

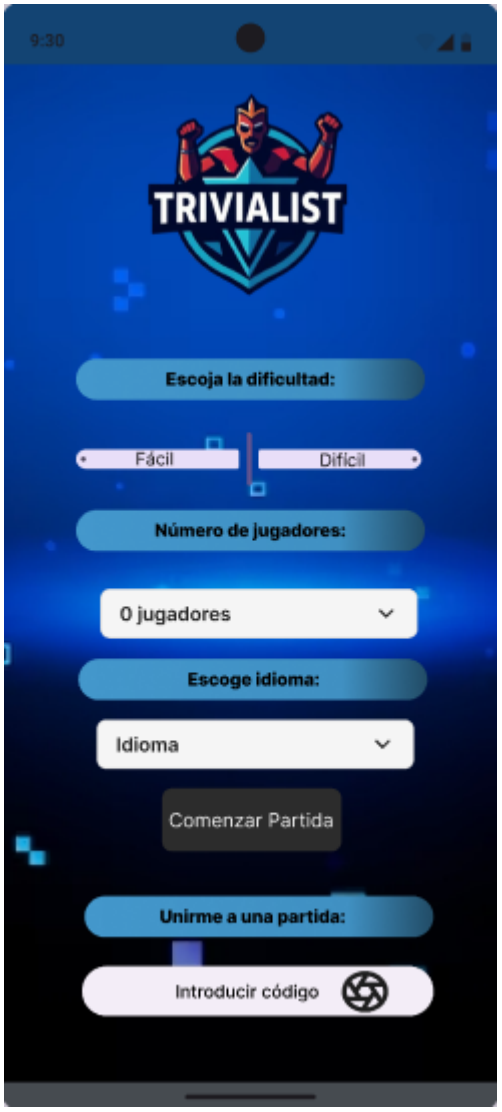
3.2 Descripción del código

```java
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.databinding.DataBindingUtil;

import com.example.trivialist.databinding.ActivityLobbyBinding;

public class LobbyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityLobbyBinding binding = DataBindingUtil.setContentView(this , R.layout.activity_lobby);

        Intent intent = getIntent();
        String nameuser=intent.getStringExtra("nameUser");

        binding.buttonPlay.setOnClickListener(onClick(view) -> {
                Intent intent = new Intent(LobbyActivity.this, ConfigLobbyActivity.class);
                intent.putExtra("nameUser",nameuser);
                startActivity(intent);
        });
        binding.buttonHelp.setOnClickListener(onClick(view) -> {
                Intent intent = new Intent(LobbyActivity.this, HelpActivity.class);
                startActivity(intent);
        });
    }
}
```

La clase **`LobbyActivity`** define una pantalla principal (lobby) de una aplicación Android. Actúa como un punto de entrada donde los usuarios pueden interactuar con dos botones principales: uno para jugar (**Play**) y otro para acceder a la ayuda (**Help**). Esta actividad utiliza **Firebase Data Binding** para simplificar el acceso a los componentes de la interfaz y **Intents** para navegar entre actividades. Además, recibe información de actividades previas para personalizar la experiencia del usuario.

Activity Comenzar Partida

4.1 Descripción de la Activity



En esta activity nos encontramos con la pantalla previa al inicio de la partida. En este punto tendremos que crear nosotros mismos la partida seleccionando la dificultad de las preguntas, el número de jugadores y el idioma. Por otro lado podemos unirnos a una partida que haya creado algún amigo y tendremos que introducir el código para unirnos a la misma.

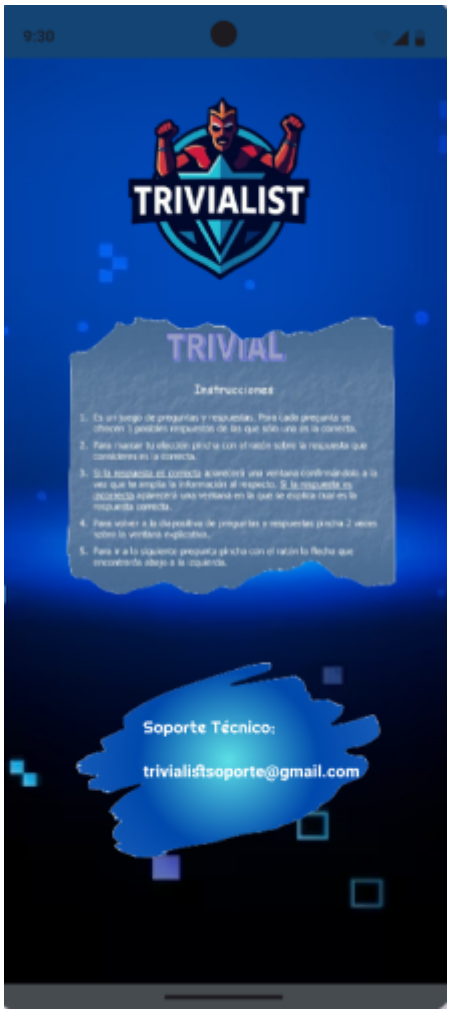
4.2 Descripción del código

```java
public class ConfigLobbyActivity extends AppCompatActivity {  3 usages

    String dificultad = "facil";  2 usages
    int jugadores = 2;  4 usages
    private ActivityConfigLobbyBinding binding;  11 usages
    private String nameUser;  3 usages
    private String codigoSala;  2 usages
    @Override  no usages
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this, R.layout.activity_config_lobby);
        Intent intent = getIntent();
        nameUser = intent.getStringExtra("nameUser");
        setupSeekBar();
        setupRadioGroup();
        setupButton();
        setupJoinButton();
    }

    private void setupSeekBar() {  1 usage
        binding.seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override  no usages
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                if (progress <= 50) {
                    dificultad = "facil";
                } else {
                    dificultad = "dificil";
                }
            }

            @Override  no usages
            public void onStartTrackingTouch(SeekBar seekBar) {
                int progress = binding.seekBar.getProgress();
                if (progress <= 50) {
                    binding.seekBar.setProgress(1);
                } else {
                    binding.seekBar.setProgress(100);
                }
            }

            @Override  no usages
            public void onStopTrackingTouch(SeekBar seekBar) {}
        });
    }

    private void setupRadioGroup() {  1 usage
        binding.radioGroupJugadores.setOnCheckedChangeListener((radioGroup, checkedId) -> {
            if (checkedId == binding.radioButton2Jugadores.getId()) {
                jugadores = 2;
            } else if (checkedId == binding.radioButton4Jugadores.getId()) {
                jugadores = 4;
            }
        });
    }

    private void setupButton() {  1 usage
        binding.buttonInit.setOnClickListener(view -> {
            codigoSala = generateRoomCode();
            startUDPServer();
            Intent intent = new Intent(ConfigLobbyActivity.this, GameActivity.class);
            intent.putExtra("nameUser", nameUser);
            intent.putExtra("codigoSala", codigoSala);
            intent.putExtra("numJugadores", jugadores);
            startActivity(intent);
        });
    }

    private void setupJoinButton() {  1 usage
        binding.buttonJoinRoom.setOnClickListener(view -> {
            EditText codeInput = binding.editTextRoomCode;
            String roomCode = codeInput.getText().toString();

            if (roomCode.isEmpty()) {
                Toast.makeText(ConfigLobbyActivity.this, "Por favor, introduce un código de sala válido.", Toast.LENG
                return;
            } else {
                joinRoom(roomCode);
            }
        });
    }

    private void startUDPServer() {  1 usage
        try {
            } catch (Exception e) {
                e.printStackTrace();
            }
    }

    private void joinRoom(String roomCode) {  1 usage
        Intent intent = new Intent(ConfigLobbyActivity.this, GameActivity.class);
        intent.putExtra("nameUser", nameUser);
        intent.putExtra("codigoSala", roomCode);
        intent.putExtra("numJugadores", jugadores);
        startActivity(intent);
    }

    // Método para generar un código de sala aleatorio de 5 números
    private String generateRoomCode() {  1 usage
        Random random = new Random();
        int code = 10000 + random.nextInt(90000);
        return String.valueOf(code);
    }
}
```

La clase **`ConfigLobbyActivity`** es una actividad que permite a los usuarios configurar parámetros importantes antes de participar en un juego. Esta configuración incluye elegir el nivel de dificultad, establecer el número de jugadores, crear una nueva sala de juego generando un código único o unirse a una sala existente mediante un código introducido por el usuario. Además, la actividad hace uso de **Firebase Data Binding** para interactuar eficientemente con los elementos de la interfaz.

Activity de Ayuda

5.1 Descripción de la Activity



En esta Activity de ayuda, podemos encontrar las instrucciones del juego con las reglas y formas de ganar y perder, además del correo de contacto de Trivialist para poder contarnos los errores o fallos que pudiera tener el juego para solucionarlo nuestro equipo de soporte técnico.

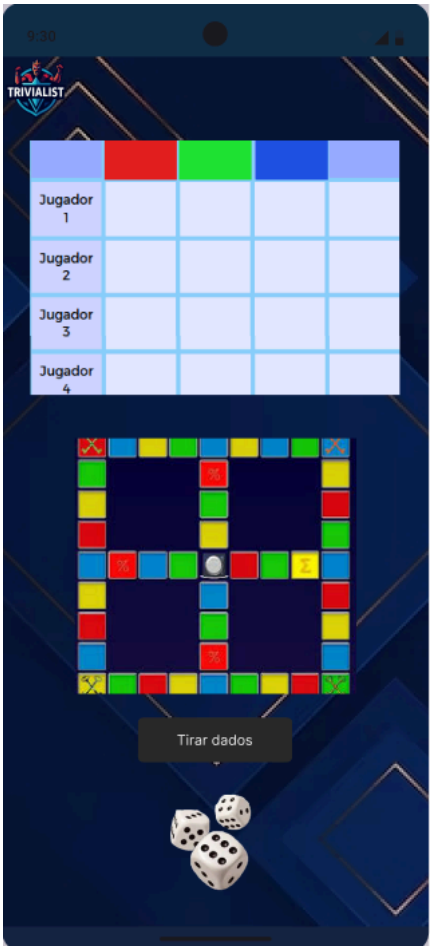
5.2 Descripción del código

```java
package com.example.trivialist;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.databinding.DataBindingUtil;

import com.example.trivialist.databinding.ActivityGameBinding;
import com.example.trivialist.databinding.ActivityHelpBinding;

public class HelpActivity extends AppCompatActivity {
    private static ActivityHelpBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this, R.layout.activity_help);
        binding.cancelHelp.setOnClickListener(onClick(view) -> {
                Intent intent = new Intent(HelpActivity.this ,LobbyActivity.class );
                startActivity(intent);
        });

    }
}
```

-Esta actividad se centra exclusivamente en la funcionalidad de un botón, el cual permitirá regresar a la actividad anterior. El propósito principal de este activity es proporcionar información sobre el funcionamiento del juego.

Activity Partida

6.1. Descripción de la Activity



En esta Activity podemos ver la pantalla de la partida en la que tenemos el tablero con la tabla de los aciertos y los distintos jugadores y abajo del tablero el botón para tirar los dados, además de unos dados para que se vean los resultados de los dados en la partida.

6.2. Descripción del código

```
    super.onCreate(savedInstanceState);

    binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_game);

    // Agrega un ImageView en coordenadas específicas x e y
    addImageAtPosition();

    Intent intent = getIntent();
    numjugadores = intent.getIntExtra( name: "numJugadores",  defaultValue: 2);
    nombreJugador = intent.getStringExtra( name: "nameUser");
    Toast.makeText( context: this, nombreJugador, Toast.LENGTH_SHORT).show();

    codigoSala = intent.getStringExtra( name: "codigoSala");
    Toast.makeText( context: this, codigoSala, Toast.LENGTH_SHORT).show();
    binding.codeRoom.setText(codigoSala);

    GridLayout gridLayout = binding.marcador;
    for (int i = 0; i < numjugadores; i++) {
        TextView textView = new TextView( context: this);
        textView.setText("Jugador " + (i + 1));

        int height = dpToPx( context: this,  dp: 40);
        GridLayout.LayoutParams params = new GridLayout.LayoutParams(
                GridLayout.spec( start: i + 1),
                GridLayout.spec( start: 0)
```



```
GridLayout gridLayout = binding.marcador;
for (int i = 0; i < numjugadores; i++) {
    TextView textView = new TextView( context: this);
    textView.setText("Jugador " + (i + 1));

    int height = dpToPx( context: this,  dp: 40);
    GridLayout.LayoutParams params = new GridLayout.LayoutParams(
            GridLayout.spec( start: i + 1),
            GridLayout.spec( start: 0)
    );
    params.width = GridLayout.LayoutParams.WRAP_CONTENT;
    params.height = height;

    textView.setLayoutParams(params);
    gridLayout.addView(textView);

    for (int j = 1; j < 6; j++) {
        ImageView imageView = new ImageView( context: this);
        imageView.setImageResource(R.drawable.ic_launcher_foreground);

        GridLayout.LayoutParams imageParams = new GridLayout.LayoutParams(
                GridLayout.spec( start: i + 1),
                GridLayout.spec(j)
        );
        imageParams.width = dpToPx( context: this,  dp: 40);
```

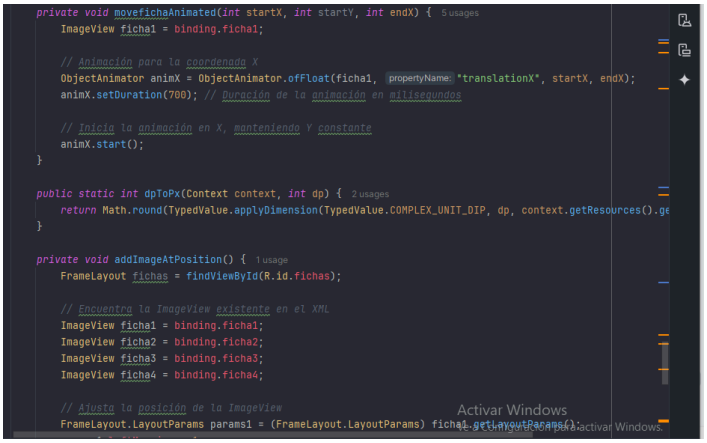
```
    }

    binding.imageDado.setImageResource(R.drawable.dice_1);
    binding.buttonDado.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            rollDice(binding.imageDado);
        }
    });

    conectarAlServidor();
}

private void conectarAlServidor() {  1 usage
    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                socket = new Socket( host: "your_server_ip",  port: 5555); // Cambia "your_server_ip" p
                out = new DataOutputStream(socket.getOutputStream());
                in = new DataInputStream(socket.getInputStream());

                // Enviar acción para unirse a la sala
                out.writeUTF( str: "unir_sala");
                out.writeUTF(nombreJugador);
                out.writeUTF(codigoSala);
```



```
    }

    private void rollDice(final ImageView imageDado) {  1 usage
        ObjectAnimator rotate = ObjectAnimator.ofFloat(imageDado,  propertyName: "rotation",  ...values: 0f, 360f);
        rotate.setDuration(700);
        rotate.start();
        rotate.addListener(new AnimatorListenerAdapter() {
            @Override
            public void onAnimationEnd(Animator animation) {
                super.onAnimationEnd(animation);
                setRandomDiceImage(imageDado);
            }
        });
    }

    private void setRandomDiceImage(final ImageView imageDado) {  1 usage
        Random random = new Random();
        int randomNumber = random.nextInt( bound: 5) + 1;

        int[] posicion = getPosition(binding.ficha1);
        int posicionx = posicion[0];
        int posiciony = posicion[1];

        // Ajusta la posición de la ficha en base al número del dado
        switch (randomNumber) {
            case 1:
```

```
    switch (randomNumber) {
        case 1:
            imageDado.setImageResource(R.drawable.dice_1);
            movefichaAnimated(posicionx, posiciony, endX: 169);
            break;
        case 2:
            imageDado.setImageResource(R.drawable.dice_2);
            movefichaAnimated(posicionx, posiciony, endX: 339);
            break;
        case 3:
            imageDado.setImageResource(R.drawable.dice_3);
            movefichaAnimated(posicionx, posiciony, endX: 509);
            break;
        case 4:
            imageDado.setImageResource(R.drawable.dice_4);
            movefichaAnimated(posicionx, posiciony, endX: 679);
            break;
        case 5:
            imageDado.setImageResource(R.drawable.dice_5);
            movefichaAnimated(posicionx, posiciony, endX: 849);
            break;
    }

    // Envía los puntos al servidor
    enviarPuntosAlServidor(randomNumber);
```

```
final Intent intent = new Intent( packageContext: GameActivity.this, QuestionActivi
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        switch (randomNumber) {
            case 1:
                intent.putExtra( name: "categoria", value: "Matematicas");
                break;
            case 2:
                intent.putExtra( name: "categoria", value: "Geografia");
                break;
            case 3:
                intent.putExtra( name: "categoria", value: "Arte");
                break;
            case 4:
                intent.putExtra( name: "categoria", value: "Deportes");
                break;
            case 5:
                intent.putExtra( name: "categoria", value: "Historia");
                break;
        }
        startActivity(intent);
    }
}, delayMillis: 1500); // 1500 milisegundos = 1.5 segundos
}
```



```
private void movefichaAnimated(int startX, int startY, int endX) {  5 usages
    ImageView ficha1 = binding.ficha1;

    // Animación para la coordenada X
    ObjectAnimator animX = ObjectAnimator.ofFloat(ficha1,  propertyName: "translationX", startX, endX);
    animX.setDuration(700); // Duración de la animación en milisegundos

    // Inicia la animación en X, manteniendo Y constante
    animX.start();
}

public static int dpToPx(Context context, int dp) {  2 usages
    return Math.round(TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, dp, context.getResources().ge
}

private void addImageAtPosition() {  1 usage
    FrameLayout fichas = findViewById(R.id.fichas);

    // Encuentra la ImageView existente en el XML
    ImageView ficha1 = binding.ficha1;
    ImageView ficha2 = binding.ficha2;
    ImageView ficha3 = binding.ficha3;
    ImageView ficha4 = binding.ficha4;

    // Ajusta la posición de la ImageView
    FrameLayout.LayoutParams params1 = (FrameLayout.LayoutParams) ficha1.getLayoutParams();
```


```
    ImageView ficha3 = binding.ficha3;
    ImageView ficha4 = binding.ficha4;

    // Ajusta la posición de la ImageView
    FrameLayout.LayoutParams params1 = (FrameLayout.LayoutParams) ficha1.getLayoutParams();
    params1.leftMargin = -1;
    params1.topMargin = -1;
    FrameLayout.LayoutParams params2 = (FrameLayout.LayoutParams) ficha2.getLayoutParams();
    params2.leftMargin = 0;
    params2.topMargin = 98;
    FrameLayout.LayoutParams params3 = (FrameLayout.LayoutParams) ficha3.getLayoutParams();
    params3.leftMargin = 96;
    params3.topMargin = 0;
    FrameLayout.LayoutParams params4 = (FrameLayout.LayoutParams) ficha4.getLayoutParams();
    params4.leftMargin = 96;
    params4.topMargin = 96;

    ficha1.setLayoutParams(params1);
    ficha2.setLayoutParams(params2);
    ficha3.setLayoutParams(params3);
    ficha4.setLayoutParams(params4);
}
```

El GameActivity es una actividad de Android que maneja la UI y la lógica de un juego. Incluye conexión a un servidor para recibir y enviar información del juego.

Configuración Inicial: Configura la UI y obtiene datos del intent.

Conexión al Servidor: Establece una conexión de socket y envía el nombre del jugador y el código de la sala al servidor.

Gestión de Dados: Anima y cambia la imagen del dado, ajustando la posición de la ficha según el resultado.

Actualización de la UI: Añade y posiciona vistas dinámicamente en la UI.

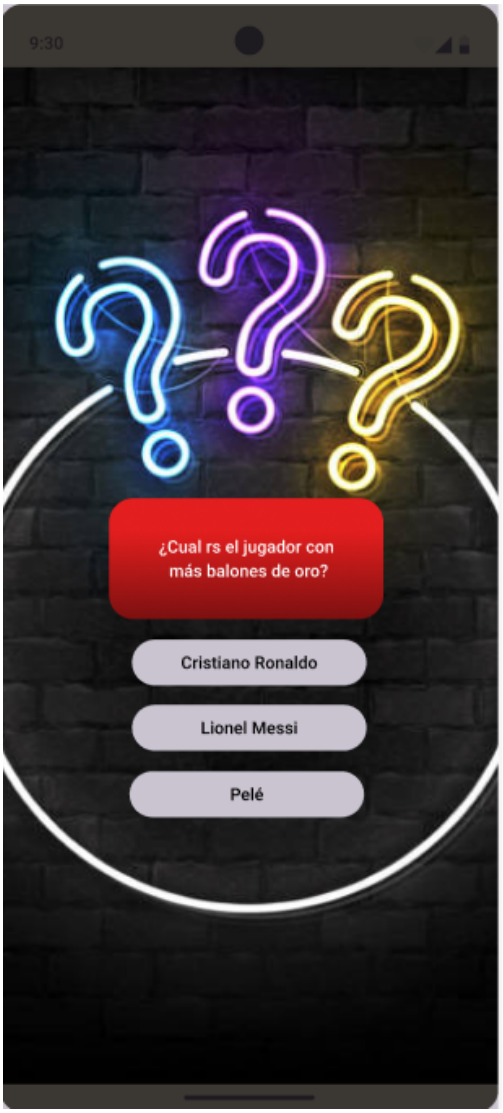
Métodos de Utilidad

dpToPx: Convierte dp a px.

getPosition: Obtiene la posición de una ImageView.

Activity Preguntas

7.1 Descripción Activity



En este Activity, podemos ver la pantalla de las preguntas, todas serían iguales, lo único que cambia de una a otra sería el color de fondo de la pregunta que indica la categoría de esta. En este caso, la pregunta que tenemos es de Deportes y abajo de la pregunta aparecen las tres opciones de respuesta para responder.

7.2 Descripción del código

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_question);

    Intent intent = getIntent();
    String categoria = intent.getStringExtra( name: "categoria");

    // Define las opciones de respuesta para cada categoría
    String[] opciones = new String[3];

    switch (categoria) {
        case "Matematicas":
            binding.Tittlequestion.setText(categoria);
            binding.questiondescript.setText("¿Cuál es el valor de π (pi) con dos decimales?");
            opciones = new String[]{"3.14", "2.71", "1.61"};
            respuestaCorrecta = "3.14";
            break;
        case "Geografia":
            binding.Tittlequestion.setText(categoria);
            binding.questiondescript.setText("¿Cuál es el río más largo del mundo?");
            opciones = new String[]{"El Amazonas", "El Nilo", "El Misisipi"};
            respuestaCorrecta = "El Amazonas";
            break;
```

```
        case "Arte":
            binding.Tittlequestion.setText(categoria);
            binding.questiondescript.setText("¿Quién pintó la famosa obra \"La Noche Estrellada\"?");
            opciones = new String[]{"Vincent van Gogh", "Pablo Picasso", "Leonardo da Vinci"};
            respuestaCorrecta = "Vincent van Gogh";
            break;
        case "Historia":
            binding.Tittlequestion.setText(categoria);
            binding.questiondescript.setText("¿Cuál fue el motivo principal que desencadenó la Primera Gue
            opciones = new String[]{"El asesinato del archiduque Francisco Fernando de Austria", "La firma
            respuestaCorrecta = "El asesinato del archiduque Francisco Fernando de Austria";
            break;
        case "Deportes":
            binding.Tittlequestion.setText(categoria);
            binding.questiondescript.setText("¿En qué deporte es conocida Serena Williams por haber ganado
            opciones = new String[]{"Tenis", "Baloncesto", "Natación"};
            respuestaCorrecta = "Tenis";
            break;
    }
```

```
// Añade las opciones al RadioGroup
for (String opcion : opciones) {
    RadioButton radioButton = new RadioButton( context: this);
    radioButton.setText(opcion);
    binding.radioGroupQuestions.addView(radioButton);
}

binding.enterQuestion.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        int selectedId = binding.radioGroupQuestions.getCheckedRadioButtonId();
        if (selectedId == -1) {
            // El usuario no ha marcado ninguna opción
            Toast.makeText( context: QuestionActivity.this,  text: "Por favor, marque una opción.", Toast.LI
        } else {
            // El usuario ha marcado una opción
            RadioButton selectedRadioButton = findViewById(selectedId);
            String selectedText = selectedRadioButton.getText().toString();

            if (selectedText.equals(respuestaCorrecta)) {
                // Respuesta correcta
                Toast.makeText( context: QuestionActivity.this,  text: "¡Correcto!", Toast.LENGTH_SHORT).sh
```

```
                Toast.makeText( context: QuestionActivity.this,  text: "¡Correcto!", Toast.LENGTH_SHORT).sh
            } else {
                // Respuesta incorrecta
                Toast.makeText( context: QuestionActivity.this,  text: "¡Incorrecto!", Toast.LENGTH_SHORT)
            }
            // Volver a GameActivity
            Intent intent = new Intent( packageContext: QuestionActivity.this, GameActivity.class);
            startActivity(intent);
        }
    }
});
}
}
```

QuestionActivity presenta una pregunta de trivia al jugador basada en la categoría seleccionada y valida la respuesta.

Configuración Inicial: Configura la UI y obtiene la categoría de la pregunta.

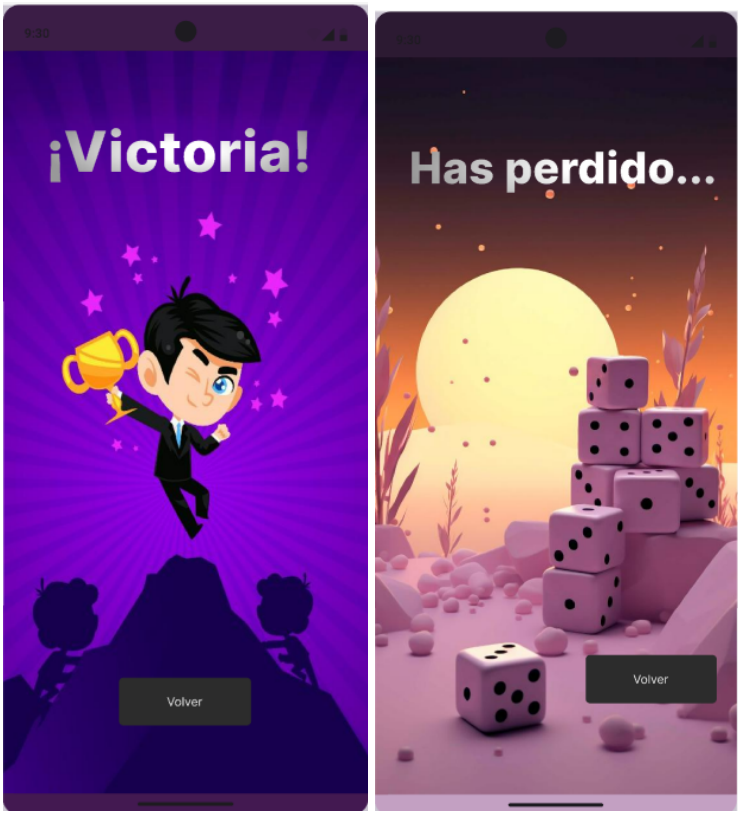
Presentación de Pregunta y Opciones: Muestra la pregunta y sus opciones basadas en la categoría.

Validación de Respuesta: Comprueba si la respuesta seleccionada es correcta o incorrecta y muestra un mensaje.

Este código se encarga de mostrar una pregunta de trivia y validar la respuesta del usuario de manera interactiva.

Activity Victoria y Derrota

8.1 Descripción Activity



Por último, estas son las Activity de Victoria y Derrota. La Activity de Victoria muestra un personaje ficticio levantando un trofeo en signo de felicidad y de haber ganado, mientras que la de derrota se pueden apreciar una montaña de dados solos en signo de desolación y derrota. Ambas Activity tienen un botón de Volver para ir al menú principal para poder empezar una nueva partida.

8.2 Descripción del código

En este activity tendremos un botón el cual nos mandará al activity el cual nos muestra ayuda o crear partida una vez que el juego finalice