TRABAJO FIN DE CICLO

SMASH TOPO

ANTONIO NICOLAS LOIS SANMARTÍN

DESARROLLO DE APLICACIONES MULTIPLATAOFRMA

ÍNDICE

1. DESCRIPCIÓN DE PROYECTO Y ÁMBITO DE IMPLANTACIÓN

El proyecto que se presenta a continuación se trata de un juego para dispositivos móviles llamado **SMASH TOPO**. La mecánica de funcionamiento del juego no es muy compleja, lo que tiene como resultado que pueda ser jugado por un gran número de personas de diferentes edades. La finalidad de esta aplicación es el entretenimiento, siendo un juego de fácil jugabilidad, de corta duración y entretenido. Por todas las razones mencionadas anteriormente es un juego que puede tener un gran acogimiento entre sus usuarios y una buena idea de crear una forma de entretenimiento.

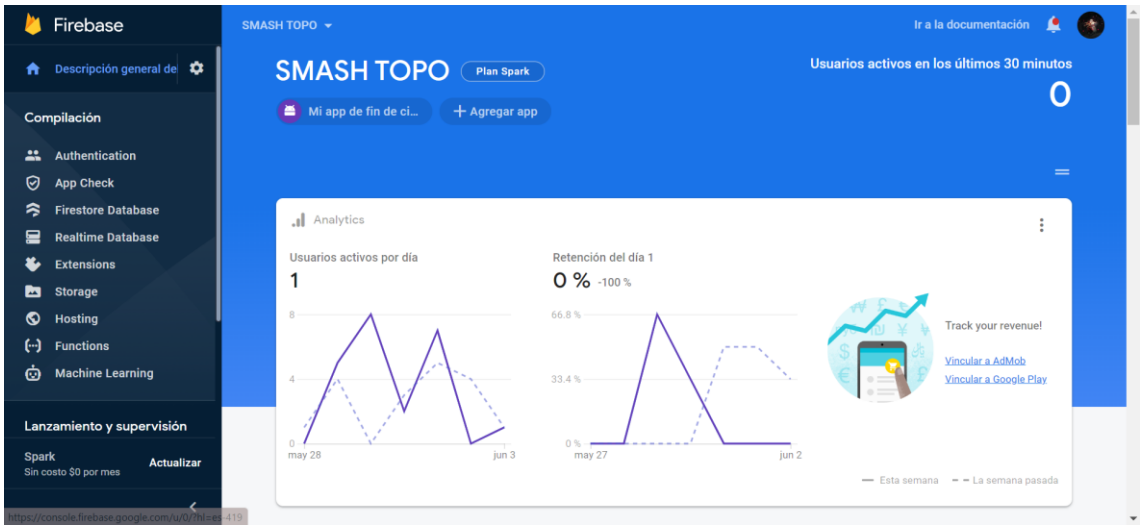
Primeramente el juego nos muestra una pantalla de carga con el título, una imagen representativa y una animación de carga. Una vez finalizada esta pantalla de carga , accederemos a una pantalla donde tendremos dos opciones, **registro** y **login** , la opción de registro creará un usuario nuevo, para ello solicita un nombre, un correo electrónico válido pero no necesariamente tiene que existir, y una contraseña para posteriormente poder acceder. Esta pantalla consta de diversos campos de textos, una imagen ,una animación, así como un botón para poder registrar al usuario, además los diferentes campos existentes tienen ciertas condiciones, como un tipo de formato de correo válido, una contraseña y un nombre con una longitud mínima aceptable. Todas estas condiciones se deberán cumplir para realizar el registro, si no aparecerá el error correspondiente por pantalla.

Una vez realizado el registro ya podremos iniciar sesión y jugar, para ello accedemos a la pantalla de login desde el menú, en esta pantalla tenemos un campo de correo electrónico y una contraseña, todos estos datos fueron introducidos anteriormente. Cuando estos datos estén completados se realizará click en el botón de login, y se buscara al correspondiente usuario dentro de la base de datos. Una vez iniciada la sesión, el usuario se encontrará con un menú, donde puede ver su última puntuación, su nombre, su correo, una foto de perfil que podrá modificar, además de los botones de jugar, clasificaciones(que más

adelante tendrá plena funcionalidad), cerrar sesión y editar perfil. Si el usuario realiza click en el botón jugar, accederá a la pantalla de juego donde existe una cuenta atrás con un tiempo inicial de 30 segundos, aunque se puede cambiar fácilmente en el código y un contador con el total de topos aplastados. Una vez que se acabe esa cuenta atrás se guardará la cantidad de topos que el usuario ha sido capaz de aplastar. Inicialmente un topo está colocado en un punto randmon de la pantalla y el usuario deberá clickar encima de él, si lo logra sumará un punto, y aparecerá otro topo en otro sitio randmon, si no pulsa encima del topo no sumará ningún punto ,en un futuro la idea es desarrollar un método que detecte si el usuario realizo click satisfactoriamente o no, y en caso de que no fuera así, le restara puntos o algún tipo de penalización. Como conclusión, el juego trata de aplastar los máximos topos posibles en el tiempo dado.

Algunos de los cambios que se prevén para un futuro, sería la implementación de una pantalla donde se mostrarán todas las puntuaciones de los jugadores, además de una mejora en la jugabilidad, mejoras tales como bonificaciones, mejor optimización del movimiento del topo, algún tipo de tienda para poder comprar apariencias, etc.

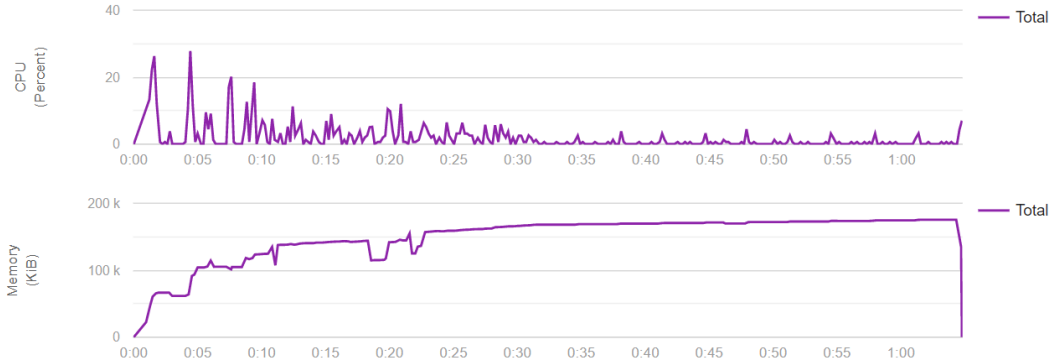
Existen diversas tecnologías implicadas en el proyecto, pero la más destacable sería Firebase, una tecnología creada por Google, que es una herramienta creada para ayudar de gran forma a la creación de apps, aportando soporte de base de datos entre otras funciones, como; realtime database, autenticación de usuarios, almacenamiento en la nube, crash reporting, test lab, remote config, cloud messaging, hosting, etc.



2. RECURSOS HARDWARE Y SOFTWARE

Para poder ejecutar esta aplicación tendremos que disponer de los siguientes componentes **hardware**:

- Un dispositivo android:
  - Con al menos 40mb de memoria libre.
  - Los componentes necesarios para que el dispositivo pueda tener acceso a internet.
  - Pantalla táctil.



Captura obtenida de Test Lab Firebase, donde se muestra el uso de la CPU y el uso de memoria, previo a la inicialización de la actividad MapaJuego.

En lo referente a los recursos de **software** sería necesario:

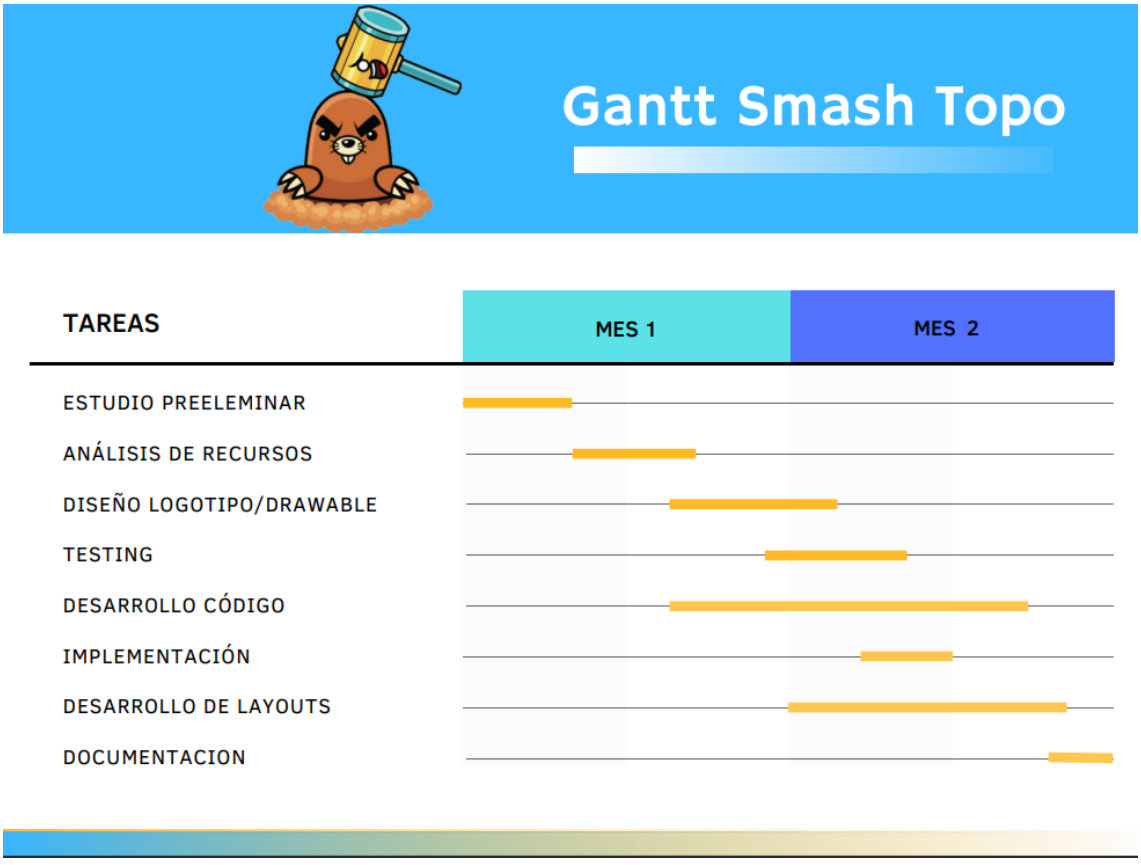
- La app instalada en el dispositivo correspondiente.
- minSdkVersion 23, targetSdkVersion 31

3. TEMPORALIZACIÓN DEL PROYECTO Y FASES DEL DESARROLLO.

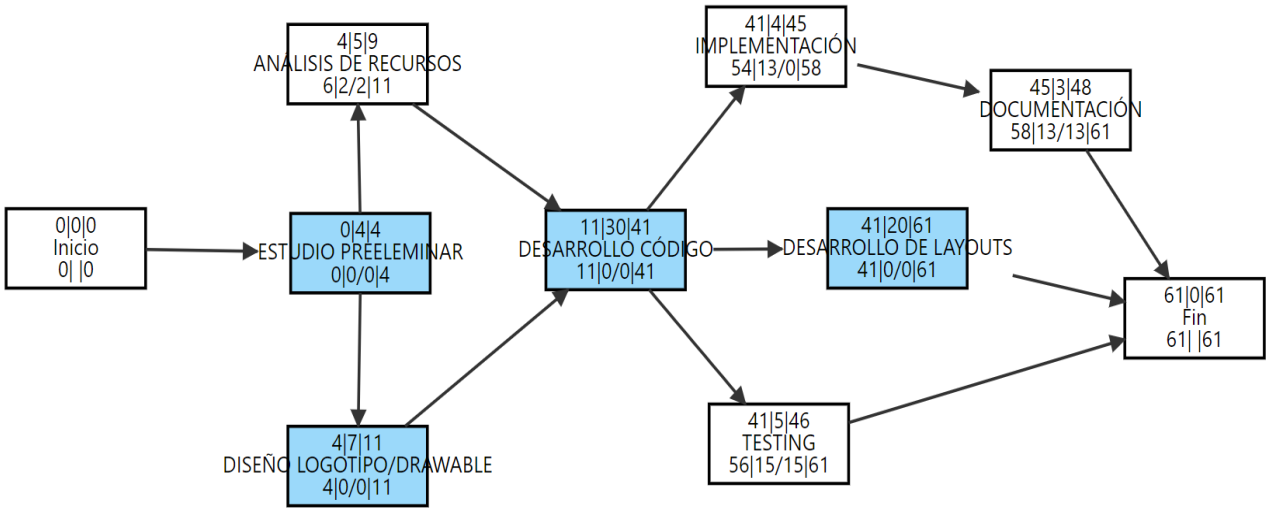
Existen diversas **dificultades** que me encontré durante el proceso asi como otras que aun sigo teniendo, como por ejemplo con las clasificaciones de los jugadores, donde obtuve diferentes errores, tales como error en los setters y getters porque la variable userID daba problemas, luego con un método de la propia clase de java, Activity.class, isChangingConfigurations(), y otros que tenía como errores cuando quería cargar los valores en la dataBase, o cuando hacía referencia a la base de datos y el campo de referencia estaba mal escrito o

directamente había problemas con las instancias, o con los permisos de lectura y escritura de los usuarios , cuando quería obtener las dimensiones de la pantalla, cuando tenía problemas a la hora de colocar de forma random el topo en la pantalla de juego, porque debido a un problema de dimensiones salía fuera de la pantalla y tardé en poder arreglar estos errores, la solución fue buscar ayuda en los diferentes foros, como en el conocido StackOverFlow, además de visualizar diferentes tutoriales de firebase. Todo esto ayudó a que entendiera mejor el funcionamiento de firebase, sus métodos y del funcionamiento de android studio, su personalización, diseño de layouts, etc..

➔ **Diagrama de Gan**



➔ **DIAGRAMA DE PERT:**



4. DESCRIPCIÓN DE DATOS.

A continuación se mostrarán las clases más importantes así como sus partes más relevantes.

CLASES:

➔ **MainActivity:**

- **Datos:** ImageButton boton_login,boton_registro
- **Funcionamiento:** Se declaran dos botones de tipo ImageButton, y se conectan con los botones existentes en el layout. En el método onCreate(), se programa para que cuando estos sean pulsados cada uno lleve al usuario a la pantalla que le corresponda, el de login a la pantalla de inicio de sesión, y el de registro, al de registro de usuario.

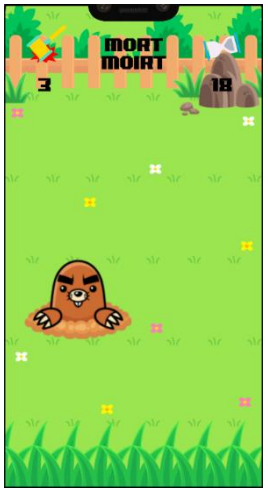
- ➔ **LoginClass:**
  - **Datos:**
    - **FirebaseAuth** auth;
    - **EditText** editTextTextEmailAddressLogin,editTextTextPasswordLogin;
    - **Button** boton_loginClass;
  - **Funcionamiento:** Con las variables iniciales mencionadas anteriormente, uno las respectivas variables , con sus localizadores en los layout, luego de los edittext de los layout, obtengo el texto que deseo, correo, y contraseña. Luego con un if, compruebo que las condiciones de los edittext sean correctas, en caso de que lo sean guardo correctamente las variables y las envío al método de inicio de sesión, donde con otro if compruebo que la tarea anterior estuviera completa, e inicio sesión con el nuevo usuario, accediendo así a la actividad de menú.
- ➔ **MapaJuego:**
  - **Datos:**
    - **//VARIABLES**
      - **String** UID, NOMBRE, Topos;
      - **TextView** contadorToposEscenario, nombreJugadorEscenario, tiempoEscenario;
      - **ImageView** topoescenario**;**
    - **//VARIABLES PARA MEDIDA PANTALLA**
      - **int** pantalla_ancho;
      - **int** pantalla_alto;
    - **//VARIABLES PARA GAME OVER**
      - **Dialog;**
      - **boolean** GameOverBoolean = false;
    - **//TOPOS**
      - **Random** posicionAleatoria**;**
      - **int** contadorTopos**;**
    - **//FIREBASE**
      - **FirebaseDatabase** firebaseDatabase**;**
      - **FirebaseAuth** firebaseAuth**;**

- **FirebaseUser** firebaseUser**;**
- **DatabaseReference** databaseJugadores**;**

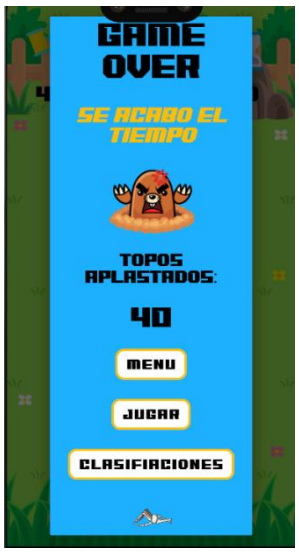
- **Funcionamiento:** Como ya mencioné en los anteriores funcionamientos de las otras clases, declaro las variables, y luego conecto aquellas que tengan los elementos en el layout, como los String o los TextView. A continuación inicializo las variables correspondientes de FireBase, para poder acceder a la base de datos, obtener el usuario actual, y asi poder obtener sus datos, que los recupero guardándolos en las variables correspondientes.

```
//INICIALIZACION
firebaseAuth = FirebaseAuth.getInstance();
firebaseUser = firebaseAuth.getCurrentUser(); //para obtener los datos del usuario que inicia sesion
firebaseDatabase = FirebaseDatabase.getInstance();
databaseJugadores= firebaseDatabase.getReference( path: "DATABASE JUGADORES REGISTRADOS");

//RECUPERAMOS DATOS
Bundle intent = getIntent().getExtras();
UID = intent.getString( key: "UID");
NOMBRE = intent.getString( key: "NOMBRE");
Topos = intent.getString( key: "Topos");
```



Luego llamo al método ScreenSizes, que obtiene las dimensiones del dispositivo actual para poder realizar el método MovimientoTopo adecuadamente, ya que en este método se generan puntos aleatorios entre un margen de valores para los ejes x e y. Existe un if que cree para que si el usuario realiza x topos aplastados, la imagen que aparezca cuando se le da a un topo sea distinta. Además también cree un método que crea una cuenta atrás donde le indiqué que tenga una duración de 30 segundos, y que se reduzca de 1 segundo en 1 segundo, y cuando esta cuenta atrás llega a cero, se declara una variable tipo boolean como true, y esto provoca que se acceda al dialog de game over, donde se muestra la puntuación y una serie de opciones en forma de botones, como volver a jugar, ir al menú o mirar las clasificaciones, método que aún no está completamente integrado.



➔ **MenuClass:**

- **Datos:**
  - **//DECLARACION DE VARIABLES**
    - **FirebaseAuth** auth**;**
    - **FirebaseUser** user**;**
    - **FirebaseDatabase** firebaseDatabase**;**
    - **DatabaseReference** databaseJugadoresRegistrados**;**
  - **//ELEMENTOS DE LA PANTALLA**
    - **Button** CerrarSesion,Jugar,Clasificaciones;
    - **TextView** cantidadTopos, userID, nombreJugadorMenu, correoJugadorMenu, botonEditarPerfil;
    - **CircleImageView** foto_perfil_player;
  - **//ELEMENTOS ALMACENAMIENTO CAMBIO DE IMAGEN**
    - **private StorageReference** storageReferenceUser;
    - **private String** pathAlmacenamiento = "ImagenPerfil/*";
    - **private String []** permisoAlmacenar;
    - **private Uri** imagen_uri;
    - **private String** perfilJugador;
  - **//CODIGOS DE PERMISOS Y SOLICTUDES**
    - **private static final int** codigoSolicitudAlmacenamiento = 8520;
    - **private static final int** codigoSeleccionImagen=7410;
- **Funcionamiento:** Como ya mencioné en los anteriores funcionamientos de las otras clases, declaro las variables, y luego conecto aquellas que tengan los elementos en el layout, como los String, los TextView, los Button o las imágenes. Se realizan las inicializaciones que corresponden; base de datos , usuario,.. se solicita permiso de almacenamiento para poder editar la foto de perfil que el usuario podrá elegir accediendo a su galería, haciendo referencia al manifest. Existen múltiples métodos en esta clase, como los que comprueban si el usuario ha cerrado o no sesión, y

en caso de que no, inicia la sesión automáticamente, para que así el usuario no tenga que iniciar sesión cada vez que accede al juego, además de los que cierran la sesión , comprueban los permisos del usuario , el que hace un upload de la imagen de perfil, accediendo a la base de datos, atreves del path , del id del usuario y la referencia. También existe un método donde el usuario puede cambiar su nombre, con condición de longitud máxima.

Para todo lo mencionado anteriormente, existen diversos botones para poder acceder a los métodos, como se muestra a continuación:

➔ **RegistroClass:**

- **Datos:**
  - **// DECLARACIÓN DE VARIABLES**
    - **EditText** editTextTextEmailAddress, editTextTextPassword, editTextTextPersonName;
    - **TextView fecha**_registro**;**
    - **CircleImageView** foto_perfil_player;
    - **Button** boton_registar;
  - **// AUTENTICACIÓN DE FIREBASE**
    - **FirebaseAuth** fireBaseAutenticacion;
- **Funcionamiento:** Como ya mencioné en los anteriores funcionamientos de las otras clases, declaro las variables, y luego conecto aquellas que tengan los elementos en el layout, como los String, los TextView y los Button. En esta clase existen diversos edittext, y su funcionan es recoger los datos que introduzca el usuario, pero además existen diversas condiciones a través de un

if, como que el correo electrónico tenga una forma valida, que la contraseña tenga una longitud mínima, que el nombre tenga una máxima, y si todo esto no se cumplen aparecen los correspondientes mensajes de error, pero en cambio sí se realizan correctamente, se accede a la base de datos de fire base y con un hasmap se regustra al usuario con sus correspondientes valores, y de igual forma si ocurre algo se recibe un error.

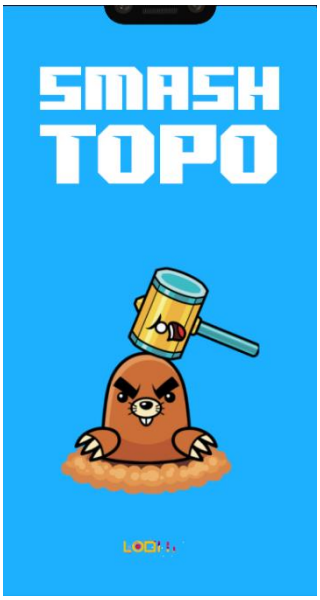
```
private void RegistrarUsuario(String correoElectronico, String contraseñaCorreo, String nombreUsuario) {
   fireBaseAutenticacion.createUserWithEmailAndPassword(correoElectronico,contraseñaCorreo)
         .addOnCompleteListener(task -> { // EN CASO DE QUE EL REGISTRO SE EFECTUE DE MANERA
CORRECTA SE MUESTRA UN MENSAJE EN PANTALLA
            if(task.isSuccessful()) {
               FirebaseUser user = fireBaseAutenticacion.getCurrentUser(); //registro del usuario actual

               int contadorTopos = 0;
               assert user != null; // COMPRUEBA QUE EL USUARIO NO SEA NULO
               String uidStringUser = user.getUid();
               String correoStringUser = editTextTextEmailAddress.getText().toString();
               String contraseñaStringUser = editTextTextPassword.getText().toString();
               String nombreStringUser = editTextTextPersonName.getText().toString();
               String fechaStringUser = fecha_registro.getText().toString();

               HashMap<Object, Object> DatosPLAYER = new HashMap<>(); // ESTO PERMITE ASIGNAR
CLAVES A LOS VALORES PARA PODER ENVIARLO A FIREBASE
               DatosPLAYER.put("UID", uidStringUser);
               DatosPLAYER.put("CORREO ELECTRONICO", correoStringUser);
               DatosPLAYER.put("CONTRASEÑA", contraseñaStringUser);
               DatosPLAYER.put("NOMBRE", nombreStringUser);
               DatosPLAYER.put("FECHA", fechaStringUser);
               DatosPLAYER.put("Topos", contadorTopos);
               DatosPLAYER.put("Imagen Jugador","foto_perfil_player");

               FirebaseDatabase database = FirebaseDatabase.getInstance();
               DatabaseReference reference = database.getReference("DATABASE JUGADORES
REGISTRADOS");
               reference.child(uidStringUser).setValue(DatosPLAYER); //almacenamiento de los datos del jugador
               startActivity(new Intent(RegistroClass.this, MenuClass.class));
               Toast.makeText(RegistroClass.this, "USUARIO REGISTRADO CORRECTAMENTE",
Toast.LENGTH_SHORT).show();
               finish();
            }else{
               Toast.makeText(RegistroClass.this, "Error", Toast.LENGTH_LONG).show();
            }

         }).addOnFailureListener(e -> { // EN CASO DE QUE EL REGISTRO SE EFECTUE DE MANERA
INCORRECTA SE MUESTRA UN MENSAJE EN PANTALLA
      Toast.makeText(RegistroClass.this, "Error: "+e.getMessage(), Toast.LENGTH_LONG).show();
         });
}
```



➔ **SplashClass:**

- **Datos:** int duracion_pantalla=3000;
- **Funcionamiento:** Esta clase únicamente está dedicada para que aparezca un layout de forma de carga de actividad, donde se muestra un textview, una animación y una imagen durante 3 segundos.

```
public class SplashClass extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        // EL OBJECTIO HANDLER NOS VA A PERMITIR EJECUTAR LAS LINEAS DE CODIGO (DENTRO DEL RUN)EN UN TIEMPO DETERMINADO
        int duracion_pantalla=3000;// 3 segundos

        new Handler().postDelayed(() -> {
            Intent intent = new Intent( packageContext: SplashClass.this, MenuClass.class);
            startActivity(intent);
        },duracion_pantalla);
    }
}
```

5. ARQUITECTURA *SOFTWARE* Y DE APLICACIÓN.

El funcionamiento de la aplicación necesitaría 3 cosas para poder funcionar, estas serían, una conexión a internet, únicamente si queremos jugar guardando los datos, la propia aplicación y un dispositivo que soporte la mencionada aplicación. La supervisión de la base de datos está bajo mi control, pudiendo acceder a los perfiles de los jugadores, test de firebase, etc, pero no es necesario realizar algún tipo de mantenimiento de la misma. A continuación adjunto un UML con los componentes software del proyecto y su interrelación.