



RODOFROLL

ROCIO ROMERO  
GAIADREAM  
2°DAM

## Índice

### Contenido

---

Índice.....	2
Título.....	5
Introducción.....	6
Objetivos.....	6
Justificación.....	6
Análisis de lo existente.....	7
Necesidades Empresariales.....	8
Recursos humanos.....	8
Prevención de riesgos laborales.....	9
Iniciativa emprendedora.....	10
Requisitos funcionales.....	11
¿Qué es el rol?.....	11
Qué entendemos por acciones de jugador.....	11
Qué entendemos por acciones de máster.....	11
Qué es un combate.....	12
Que es un personaje.....	12
El personaje en el combate.....	14
Que es un monstruo.....	14
Diseño y análisis.....	15
Arquitectura.....	15
Firebase Authentication.....	16
Firebase rules.....	16

Firebase Functions.....	18
Firebase RealtimeDatabase.....	18
Flujo de la aplicación.....	19
Estructura de datos.....	20
A nivel de aplicación.....	20
A nivel de base de datos.....	27
Codificación.....	29
Aspectos relevantes.....	31
Login Activity.....	31
MainActivity.....	32
MyFirebaseMessagingService.....	37
ListenUserService.....	38
Manual de usuario.....	39
Inicio de la aplicación.....	39
Despliegue del Menú.....	40
Bloque Jugador.....	40
Creación de un personaje.....	40
Atributos.....	41
Biografía.....	42
Inventario.....	43
Combate.....	44
Búsqueda de combates.....	45
Bloque Máster.....	46
Creación de un monstruo.....	46
Creación de un combate.....	46
Bloque General.....	52

Mi cuenta.....	52
Notificaciones.....	53
Datos.....	54
Instalación.....	56
Para la realización de este proyecto.....	56
Creación de un proyecto en firebase.....	56
Firebase Functions y como he trabajado con ellas.....	59
Para los profesores.....	61
Conclusiones.....	62
Bibliografía.....	62

## Título

Uno de los aspectos más importantes que se tiene en cuenta en este proyecto es el título, pues representará la marca del servicio que ofrecerá nuestra empresa.

La idea final fue llamarlo **RodOfRoll**, una combinación de palabras anglosajonas que significa la Vara del Rol en nuestra lengua. Por un lado, es una combinación de sonidos potente, y dado que nuestro producto va dirigido a un grupo joven la barrera del inglés no es un factor de riesgo, de hecho, es incluso más atrayente, además de que nos daría la posibilidad de expandirnos en un futuro.

Finalmente, esta marca no está registrada con anterioridad. Para comprobarlo hemos buscado en la OEPM.

The screenshot shows the official website of the Spanish Patent Office (OEPM) at [oepm.es/es/signos\\_distintivos/resultados.html?denominacion=Contenga&texto=RodOfRoll](http://oepm.es/es/signos_distintivos/resultados.html?denominacion=Contenga&texto=RodOfRoll). The search term 'RodOfRoll' is highlighted in yellow. The page displays search results for 'Signos distintivos' (Distinctive signs). On the left, there is a sidebar with links to 'Marcas nacionales' (National trademarks) including 'Manual del solicitante' (Applicant's manual), 'Tasas' (Fees), 'Formularios' (Forms), and 'Trámites en línea' (Online procedures). The main content area shows a message: 'No hay datos disponibles' (No data available).

## Introducción

### Objetivos

Nuestro proyecto pretende ser una herramienta para los jugadores de rol, un juego donde estos interpretan a un personaje, en un escenario imaginario creado por el director del juego o máster.

Los usuarios de esta aplicación podrán guardar los atributos de su personaje, vida, armadura..., sus detalles biográficos y su inventario. Esta app permitirá almacenar monstruos y crear combates donde los demás jugadores se podrán asociar, enviando uno de sus personajes. De esta manera el máster podrá controlar el turno, la vida y la armadura temporal de los monstruos y personajes que están en su combate.

Además, dispone de un tirador de dados una acción que se repite continuamente en este juego.

### Justificación

En el rol se suele hacer uso de distintos materiales, hojas de personajes, dados, mapas, etc. que en ocasiones pueden desbordar la mesa del juego y suponer una dificultad para el buen desarrollo de la partida , aparte para el máster el control de la vida de los monstruos que están en el combate a veces puede ser una tarea complicada.



## Análisis de lo existente

Existen diversas aplicaciones de rol: **Roll20, RPG Simple Dice, Dungeon20, 5e Companion App**. Pero la mayoría de ellas tienen una interfaz un poco compleja, no son de Android, o no incorporan el tirador de dados.

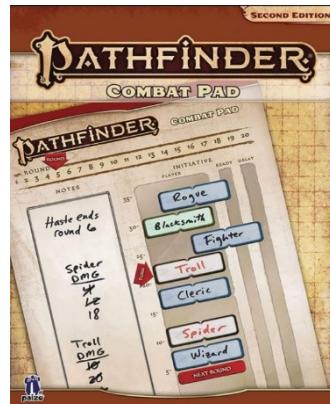
Esta aplicación tiene una interfaz muy sencilla y además los jugadores pueden buscar al máster y enviarle los datos de su personaje. Lo más innovador es que el máster podrá avisar al jugador de que va a ser su turno, a través de la herramienta de **Firebase Functions**.

Obviando las herramientas informáticas, la aplicación pretende ser una reinterpretación de los utensilios que se suelen usar en una partida de rol, aquí algunos ejemplos.



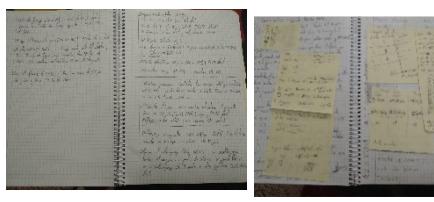
Ficha de personaje

Se recogen las características del personaje



Combat Pad

Es una herramienta para llevar el control de la batalla y los turnos



Libreta

La suelen usar tanto los jugadores como el máster, para llevar control de la vida que tienen sus combatientes en ese combate entre otras cosas

Pantalla del máster

Sobre todo, tiene la finalidad de ocultar las cosas del máster para que los jugadores no las puedan ver

## Necesidades Empresariales

Nuestra empresa se llamará Gaia Dream.SL. Una empresa dedicada a la elaboración de aplicaciones móviles, enfocadas al mundo de los Juegos de rol. La idea es crear una aplicación más ambiciosa que la actual que sea totalmente sostenible económicamente, con una versión web en la que se puedan elaborar tableros virtuales para los jugadores y sea una “red social” para todos los amantes de las partidas de rol.

Para realizar esa aplicación será necesario un equipo de personas más numeroso que en el actual, (una sola persona).

Para no ceñirnos a un solo proyecto la empresa seguiría desarrollando otro tipo de aplicaciones, quizás más enfocadas a los videojuegos.

A partir de aquí describiremos como estaría formada esa empresa.

## Recursos humanos

El equipo estaría formado en un principio por tres socios.

Socios promotores:

Rocio Romero principal promotora, encargada de la dirección de la empresa, y arquitecta de las funcionalidades que realizará RodOfRoll2.0. A parte sería una de las primeras en programar la aplicación RodOfRoll2.0.

Javier Pomares, socio encargado del diseño de la aplicación, será el que establezca qué apariencia debe tener la aplicación para ser más estética. También se encargará del marketing de esta.

Andreu Llinares, será el responsable de la creación de la base de datos, así como de su mantenimiento.

Una vez construida la aplicación se procederá a contratar trabajadores en función de la carga de trabajo que genere está. En principio serían contratos eventuales y se pagarían por trabajo realizado a 6€/hora.

La empresa debería inscribirse en la Seguridad Social para poder contratar trabajadores (afiliados también previamente), así mismo tendrá la obligación de dar de alta a estos en el Régimen General de Seguridad Social. Los socios también tendrán que estar inscritos en el Régimen General de la Seguridad. Para hacer este trámite tendremos que ir a la Tesorería General de la Seguridad Social.

Otro factor importante a tener en cuenta es la localización. Se ha optado por el teletrabajo, y el coworking. Esta última opción se basa en el alquiler por horas, de salas de reunión. La empresa que hemos decidido ,para realizar tareas asociadas a la contratación de trabajadores ,se llama Ulab y esta en [Plaza de San Cristobal, 14, 03002 Alicante.](#)

### **Prevención de riesgos laborales**

El trabajo con ordenadores supone algunos riesgos laborales que pueden afectar a la salud de los individuos que los utilizan. Es muy importante trazar un plan de actuación para evitar estos, además de que también es otro trámite para la puesta en marcha de una empresa.

Algunos de estos son: dolores de espalda, tensión en los hombros, dolores de cabeza, escozor en los ojos sequedad, calambres en las piernas etc.

A largo plazo pueden derivar en problemas mayores como hernias o pérdida prematura de la visión.

Para prevenir esto recogemos aquí algunas recomendaciones:

#### **La luminosidad**

Una buena iluminación en el lugar de trabajo, evitar deslumbramientos en la pantalla, bajar el brillo de la pantalla o subirlo.

#### **La ergonomía**

Acondicionamiento de los materiales de trabajo es decir, la utilización de sillas adaptadas a la postura de los trabajadores, mesas de trabajo a una buena altura etc.

#### **Estiramientos**

El trabajador también puede evitar por si mismo estos riesgos, por ejemplo hacer breves pausas visuales de 10 minutos por cada 90 minutos y/o realizar breves estiramientos cada hora.

## Iniciativa emprendedora

La forma jurídica escogida para esta empresa es la **sociedad limitada**. Se ha optado por esta porque nuestra empresa empezaría siendo una pequeña empresa “indie”. La gran ventaja es que el capital inicial mínimo para la constitución de la empresa es de 3000€. Además la responsabilidad es limitada al capital inicial (no recaería en el patrimonio de los individuos).

Obviando los trámites en materia de Seguridad Social nos faltaría por mencionar los siguientes.

En la Agencia Tributaria tendríamos que hacer:

Hacer la declaración censal, es decir solicitar el alta en el Censo de Empresario, Profesionales y Retenedores.

Pagar el impuesto de Actividades Económicas sólo si ganáramos un millón de euros anuales.

Pagar el impuesto sobre Sociedades. Los empleados tendrán que tributar sobre el impuesto de personas físicas.

Dotar a la sociedad de un nombre GaiaDream.SL y una personalidad jurídica. Así mismo tendremos que inscribirla en el Registro Mercantil Central.

## Requisitos funcionales

### ¿Qué es el rol?

Hablemos un poco del rol. El rol es un juego de mesa donde los jugadores interpretan personajes en un mundo fantástico. Normalmente estos asumen el papel de héroes, que para proseguir en su aventura tienen que enfrentarse a las fuerzas del mal. En este juego los jugadores pueden asumir dos papeles el de jugador (el que da vida a los héroes), o el de máster que es el narrador de la aventura.

Esta aplicación está en desarrollo, es decir es un proyecto que puede ampliarse, por tanto, se ha escogido lo esencial a fin de facilitar la comprensión y la finalidad del proyecto.

La idea principal es que un usuario pueda tener en una misma aplicación principalmente dos cosas: la posibilidad de hacer acciones de jugador o de máster.

### Qué entendemos por acciones de jugador

- Puede disponer de distintos personajes, en una misma partida no es lo usual, pero puede tener varias fichas de personaje.
- Debe pasarle los aspectos de su personaje al director del juego, si quiere que su personaje esté en la partida.
- Podrán controlar la vida y la armadura temporal de sus personajes en el combate.

### Qué entendemos por acciones de máster

- Un máster puede crear muchos monstruos.
- Debe crear combates para juntar a los monstruos con los jugadores.
- Podrá tener control sobre la vida y la armadura de sus monstruos, pero los jugadores no podrán ver estas.



## Qué es un combate

Un combate es un encuentro entre los héroes de la aventura y uno o diversos monstruos. El fin de este es que los jugadores acaben con la criatura maligna. Se puede dar el caso sin embargo de que los jugadores mueran y no maten al monstruo.

En el rol el orden de las acciones (ataques, hechizos, movimiento, etc.) de los distintos combatientes lo determina la iniciativa de cada uno de los entes.

La iniciativa es la suma del azar y la “destreza” del personaje, eso se transcribe a una tirada de un dado d20 + el modificador de iniciativa de este. De manera que el que saque más iniciativa será el primero en poder realizar sus movimientos.

## Que es un personaje

Un personaje es el héroe del jugador que usará durante toda la aventura hasta que esta finalice o este muera.

Este tiene distintas características que tienen que ser guardadas en nuestra aplicación, el proyecto se ha enfocado en las más importantes, los atributos, la biografía, y el inventario.

Los **atributos** son la vida base, la armadura base, el ataque base y la velocidad y por otro lado estarían las características : fuerza, destreza, constitución, inteligencia, sabiduría , carisma; y las salvaciones : fortaleza, reflejos, voluntad.

**Características:** Las características son ,las propiedades de un personaje, sirven para determinar la condición de dicho personaje. Si una habilidad requiere de fuerza harán una tirada con un d20 más lo que tenga en fuerza.

**Fuerza:** Mide el músculo y la potencia física.

**Destreza:** Mide la agilidad, los reflejos el equilibrio.

**Constitución:** Representa la salud y el aguante del personaje, la vida base se calcula a partir de esta.

**Inteligencia:** Determina el aprendizaje o el razonamiento.

**Sabiduría:** Describe la fuerza de voluntad, el sentido común la percepción y la intuición de un personaje.

**Carisma:** Mide la personalidad de un personaje, su magnetismo personal.

**Salvaciones:** Hace referencia a una tirada de dados para saber si el personaje se salva de una situación peligrosa.

**Fortaleza:** Salvaciones que dependen de la constitución, por ejemplo resistir un veneno.

**Reflejos:** Salvaciones que dependen de la destreza, por ejemplo esquivar hechizos o proyectiles.

**Voluntad:** Salvaciones que dependen de la sabiduría, por ejemplo eludir un miedo.

La **biografía** es la descripción del personaje , su imagen , su nombre, su raza, su historia, su alineamiento , su experiencia y su nivel.

**Alineamiento:** Es la perspectiva moral y ética del personaje, en el rol se distinguen las siguientes: Legal bueno, neutral bueno, caótico bueno, legal neutral, neutral, caótico neutral, legal malvado, neutral malvado, caótico malvado.

**Experiencia:** Es un número que determina la subida de nivel, a más combates realices más experiencia ganarás y al superar el requisito aumentaras de nivel.

**Nivel:** Determina tu grado, a más nivel más cosas podrás hacer en el combate, más hechizos, más ataques etc.

El **inventario** son las cosas que tiene el personaje, sus armas, sus paciones, las cosas que recolecta etc. Todas ellas tienen un peso, y la suma de ellas no debe superar el peso límite que puede cargar el personaje. También en el inventario debe quedar recogido el dinero del héroe.

### El personaje en el combate

En el combate el personaje puede morir. Eso quiere decir que su vida llega a 0 o dependiendo de las reglas del juego incluso a menos.

Desde el punto de vista de la aplicación, es un valor que puede cambiar durante todo el combate. Esta junto con la armadura del personaje se han considerado variables que se deben poder cambiar durante toda la partida.

### Que es un monstruo

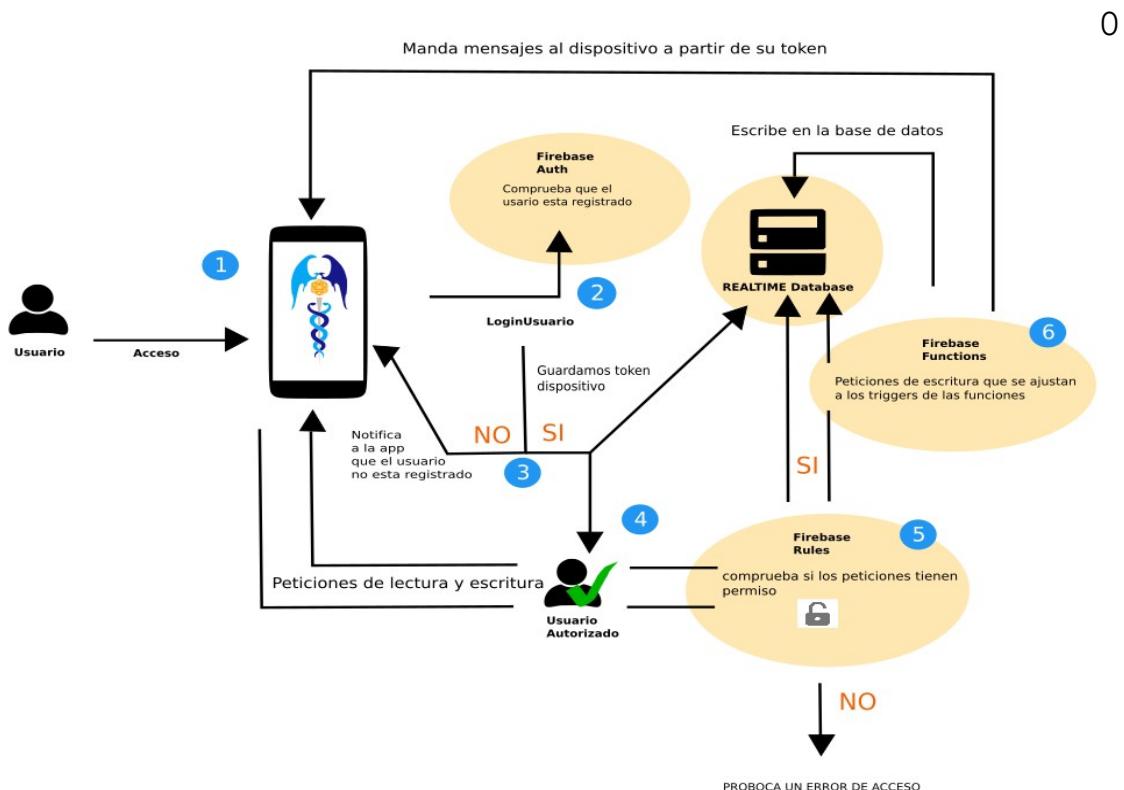
Los monstruos son los seres que se enfrentarán a los héroes, en esta aplicación también tendrán los atributos y la biografía, pero en esta última la experiencia será un reflejo de lo que da ese monstruo al matarlo.



## Diseño y análisis

### Arquitectura

El diagrama siguiente refleja la relación existente entre la aplicación y la plataforma de Firebase. Así pues los círculos anaranjados son herramientas propias de Firebase.



1. El usuario accede a la aplicación.
2. Se procede a loguear, al hacer eso la herramienta de **FirebaseAuth** procede a verificar que ese usuario existe y que la contraseña es correcta.
3. Si las credenciales son correctas se procede al siguiente paso, si no, se a notifica a la aplicación de que el usuario no es correcto.
4. Si ha dado como resultado un usuario correcto las peticiones serán propias de un usuario autorizado.
5. Las peticiones antes de pasar por la base de datos, pasan por la herramienta de **Firebase Rules** para determinar cuales se pueden realizar o cuáles no.
6. Las peticiones de escritura que se ajustan a los triggers de **Firebase Function** causan una posible reacción, estas pueden ser una escritura adicional en base de datos y/o una notificación a un dispositivo.

## Firebase Authentication

Es un servicio de Firebase, que permite identificar a los usuarios, en este proyecto a través de su email y su contraseña.

De aquí lo que hay que destacar, es que al crear un nuevo usuario, se genera una clave llamada UID que estará vinculada a esa cuenta, y que debe ser privada para el resto de usuarios.

Identificador	Proveedores	Creado	Accediste a tu cuenta	UID de usuario ↑
maria123@gmail.com	✉	8 abr. 2020		EI95FKYBwUVRZ5afBJI92v45bh1
ismael430@gmail.com	✉	8 abr. 2020		R6Jig0GhKFajgiSbWMagihLgUQZ2

## Firebase rules

En este punto tiene que quedar claro que es una petición de escritura y de lectura.

Una petición de Android a Firebase tiene este formato:

```
mDatabase.child("users").child(userId).child("username").setValue(name);
```

Esto indicaría que queremos escribir un dato en la siguiente ruta `/usuarios/{id}/username`. Lo que va entre corchetes lo podemos considerar un parámetro comodín.

Las reglas de Firebase son las que se interponen entre los datos y los hackers, son el medio de seguridad para que no cualquiera pueda escribir o leer nada.

Aquí le estaríamos diciendo que solo los usuarios autorizados que tengan la misma uid que el índice que está por debajo de usuarios (el mismo comodín del que hemos hablado antes) pueden escribir y leer.

```
"rules": {
  "usuarios": {
    ".indexOn": "id",
    "$uid": {
      ".read": "auth != null && auth.uid == $uid",
      ".write": "auth != null && auth.uid == $uid"
    }
  }
}
```

Estas son las reglas de mi proyecto:

```
{  
  "rules": {  
  
    "usuarios": {  
      ".indexOn":"id",  
      "$uid":{  
        ".read": "auth != null && auth.uid == $uid",  
        ".write": "auth != null && auth.uid == $uid",  
  
        //Sole se puede escribir si el dato que hay debajo de esta ruta tiene un hijo id  
        ".validate":"newData.hasChild('id')"  
      }  
    },  
    "publico": {  
      ".read": "auth != null",  
      "$id":{  
        //Para escribir el id tiene que ser el mismo que esta almacenado en la ruta usuarios  
        ".write": "$id === root.child('usuarios').child(auth.uid).child('id').val() "  
      }  
    },  
    "personajes":{  
      "$idp":{  
  
        //Sole se puede escribir si el dato que hay debajo de esta ruta tiene  
        //de hijos atributo/biografia e inventario|  
        ".validate":"newData.hasChildren(['atributos','biografia','inventario'])",  
        "combates":{  
          ".indexOn":"combateid"  
        }  
      }  
  
    },  
    "monstruos":{  
      "$idm":{  
        ".validate":"newData.hasChildren(['atributos','biografia'])",  
      }  
    }  
  },  
  "combates":{  
    "$uid":{  
      ".indexOn":"personajeid",  
      ".read": "auth.uid!=null",  
      ".write": "auth.uid != null",  
      "$id":{  
        "ordenturno":{  
          "$idpersonaje":{  
            ".validate":"newData.hasChild('ismonster')",  
            ".indexOn":"personajekey",  
          }  
        }  
      }  
    }  
  },  
  "notificaciones":{  
    "$id":{  
      ".read": "$id === root.child('usuarios').child(auth.uid).child('id').val() ",  
      ".write":"$id === root.child('usuarios').child(auth.uid).child('id').val()"  
    }  
  }  
}
```

## Firebase Functions

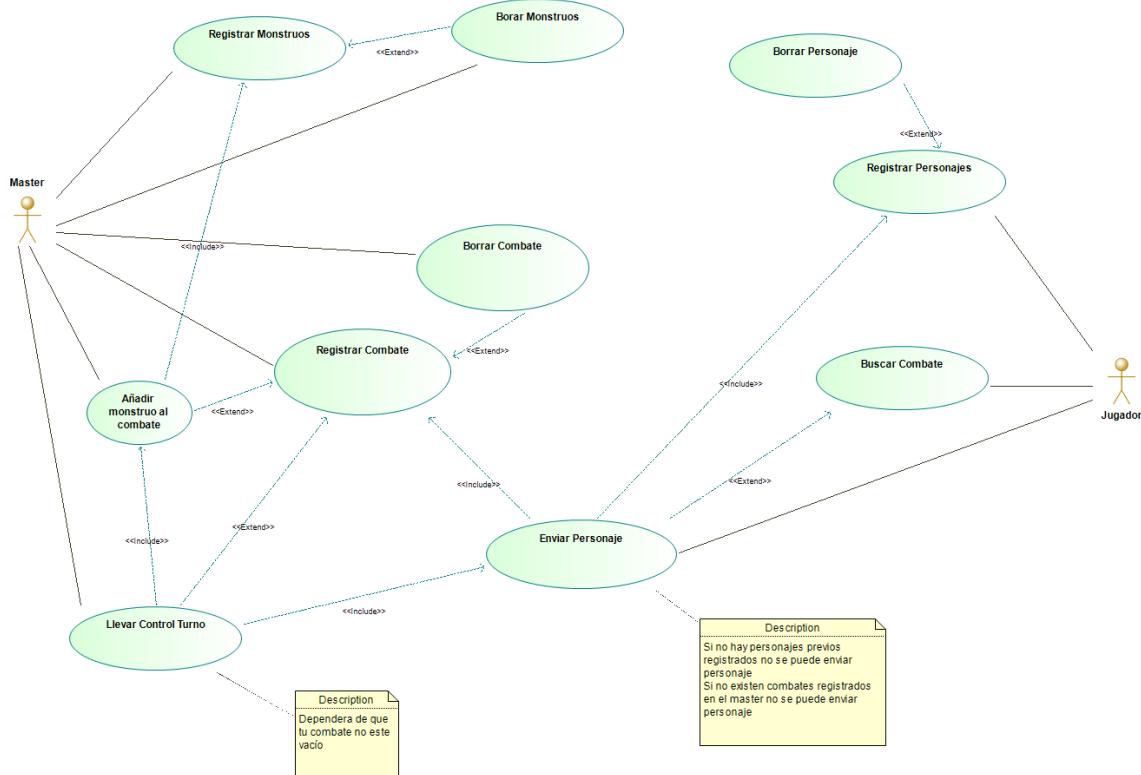
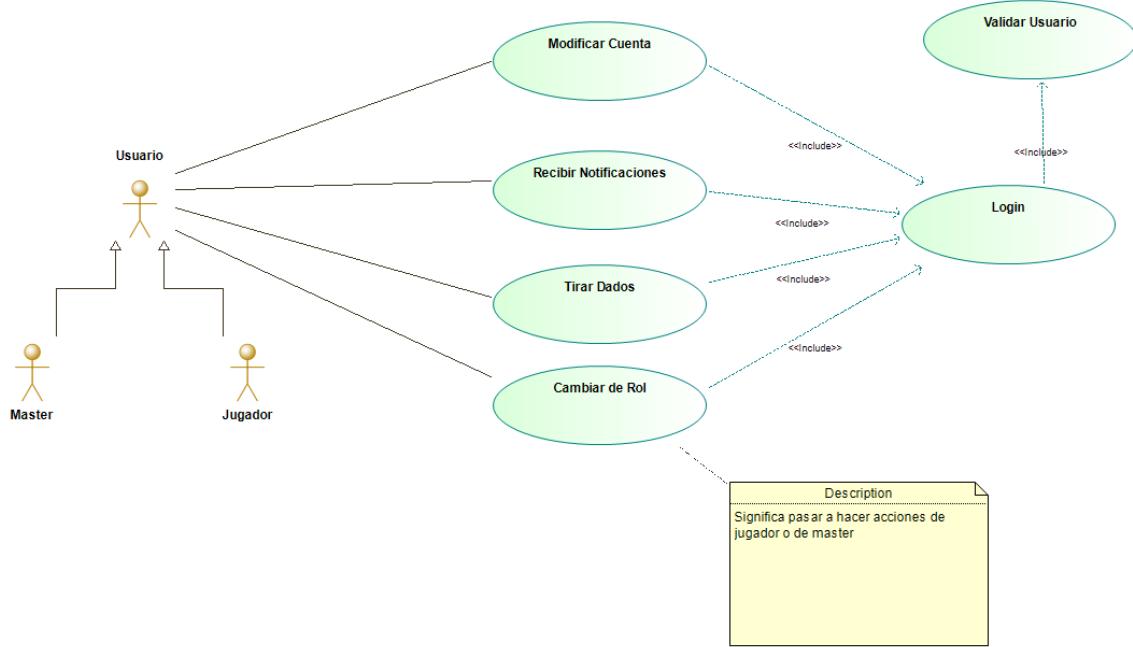
Firebase Functions, podríamos decir que funcionan como un servicio rest, en el sentido de que intercepta y responde a los eventos de creación, actualización y eliminación. Te permite ejecutar automáticamente código en respuesta a eventos que se producen en la base de datos.

## Firebase RealtimeDatabase

Es la base de datos de Firebase. Los datos se alojan en la nube en formato JSON y se sincronizan en tiempo real con cada cliente conectado.



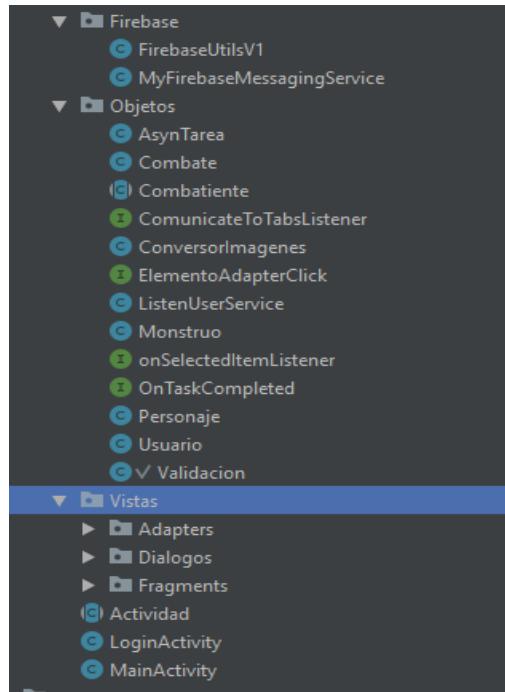
## Flujo de la aplicación



## Estructura de datos

### A nivel de aplicación

Esta ordenada por tres paquetes , Objetos,Vistas y Firebase y tres objetos de tipo actividad.



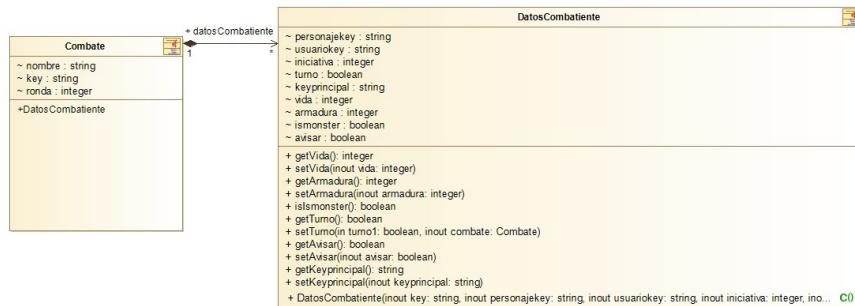
**Firebase:** Es un paquete enfocado a la gestión de los recursos de Firebase. Podemos encontrar dos objetos:

**FirebaseUtilsV1:** Es un objeto **singleton** que se encarga de almacenar datos del usuario que ha iniciado la aplicación. Además sirve como recurso para consultar cualquier dirección de la base de datos.

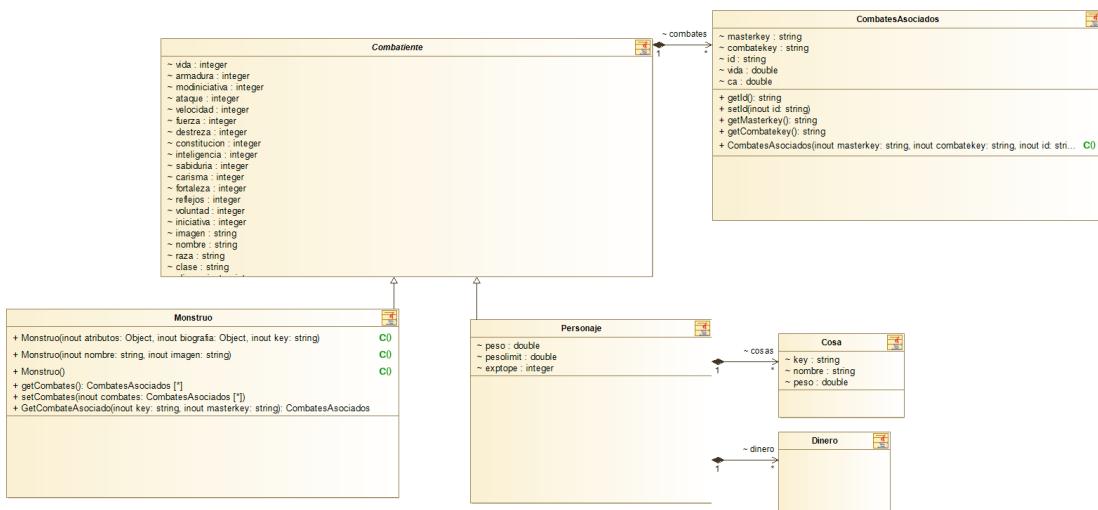
**MyFirebaseMessagingService:** Es un servicio que extiende de FirebaseMessagingService, se encarga de recibir datos que vienen desde Firebase y mostrarlos como notificaciones para el usuario.

**Objetos:** Es un paquete que almacena distintas clases creadas para el funcionamiento del programa. Aquí encontramos los objetos que darán la estructura a nuestra base de datos NoSQL.

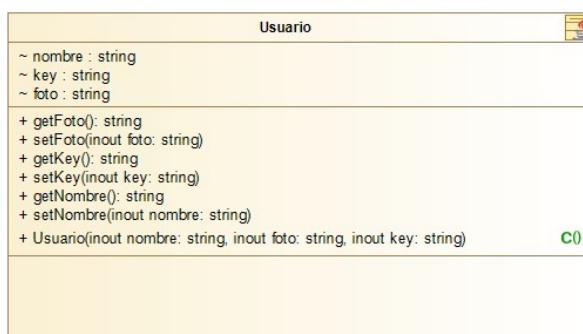
## Combate:



## **Combatiente, Personaje y Monstruo**



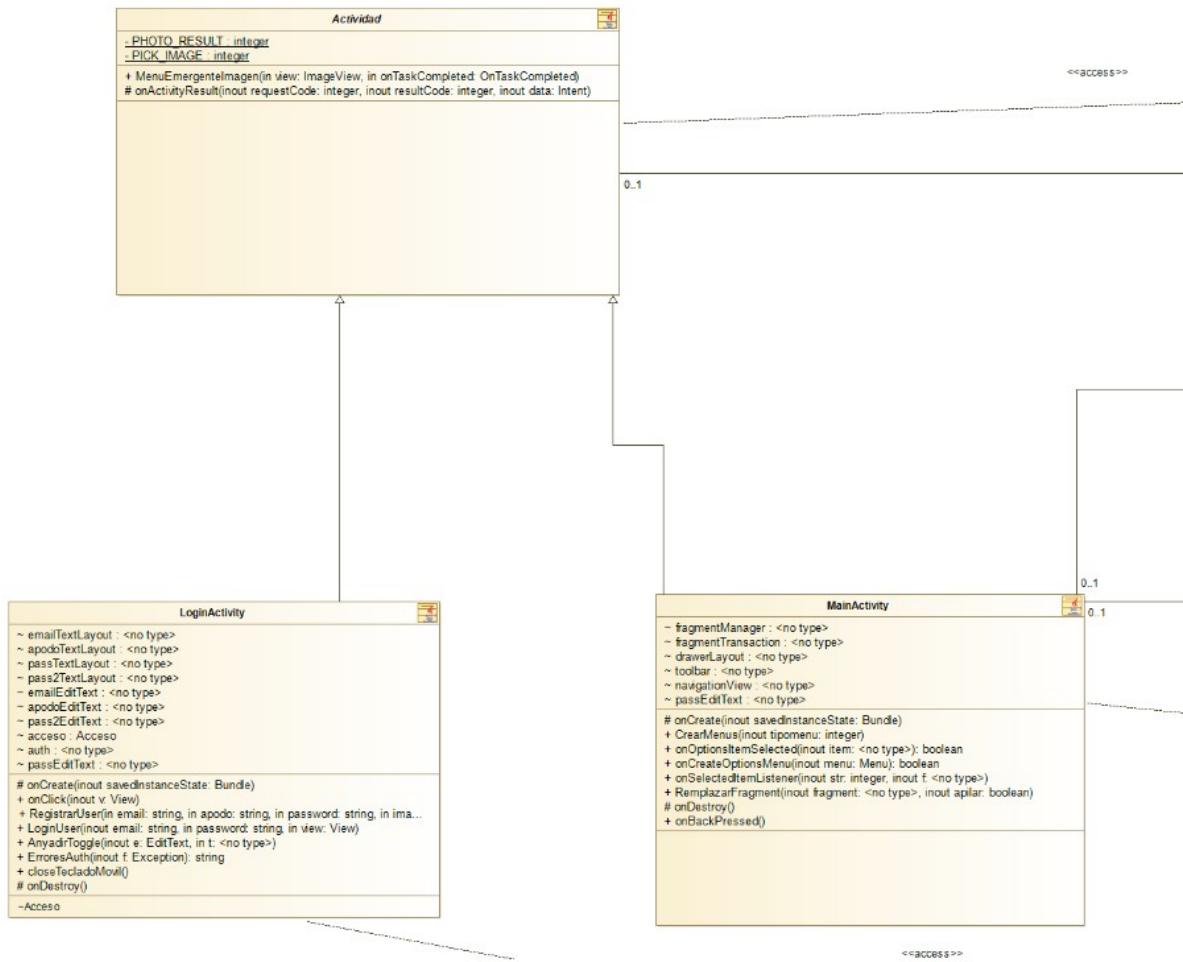
## **Usuario:**

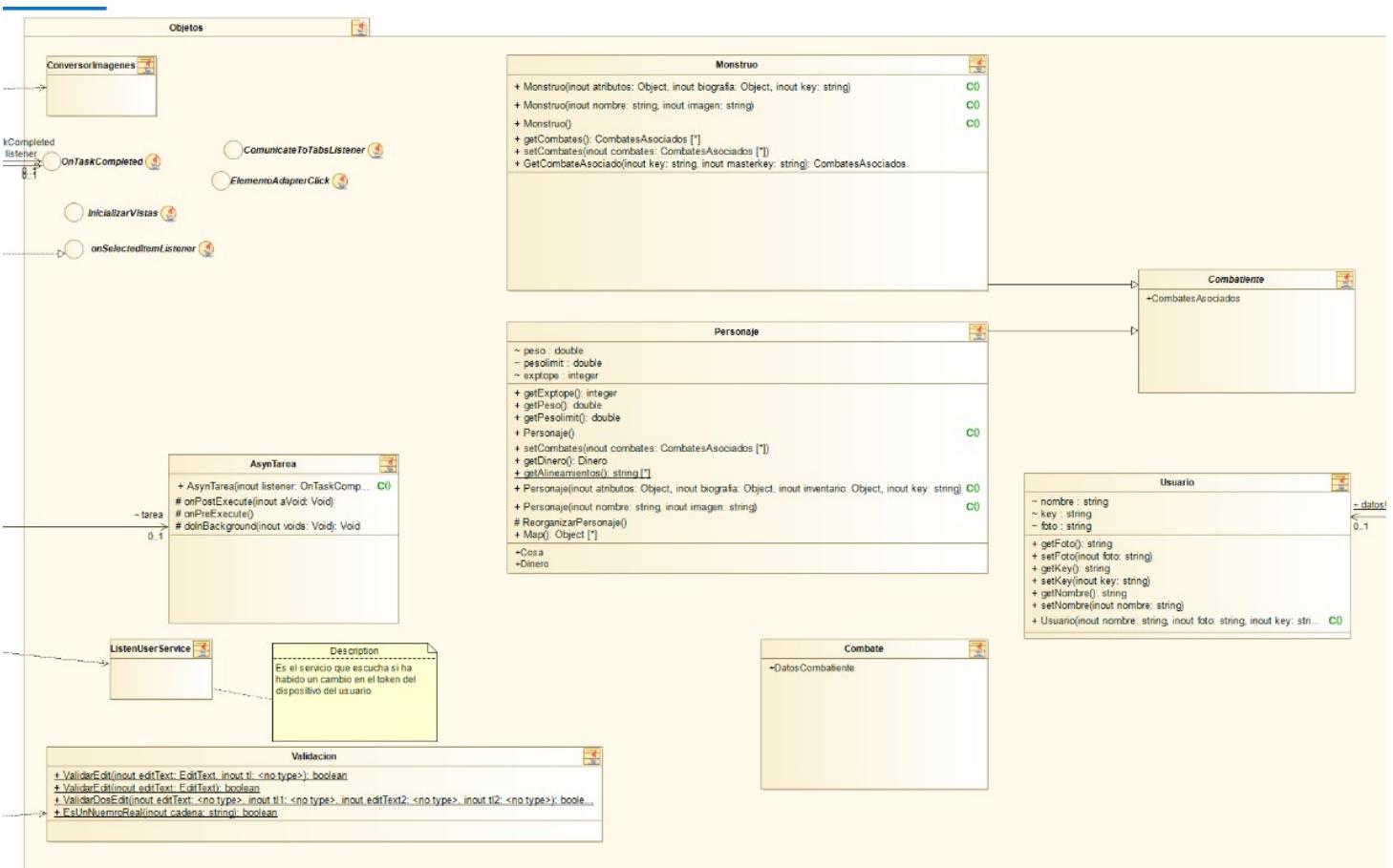


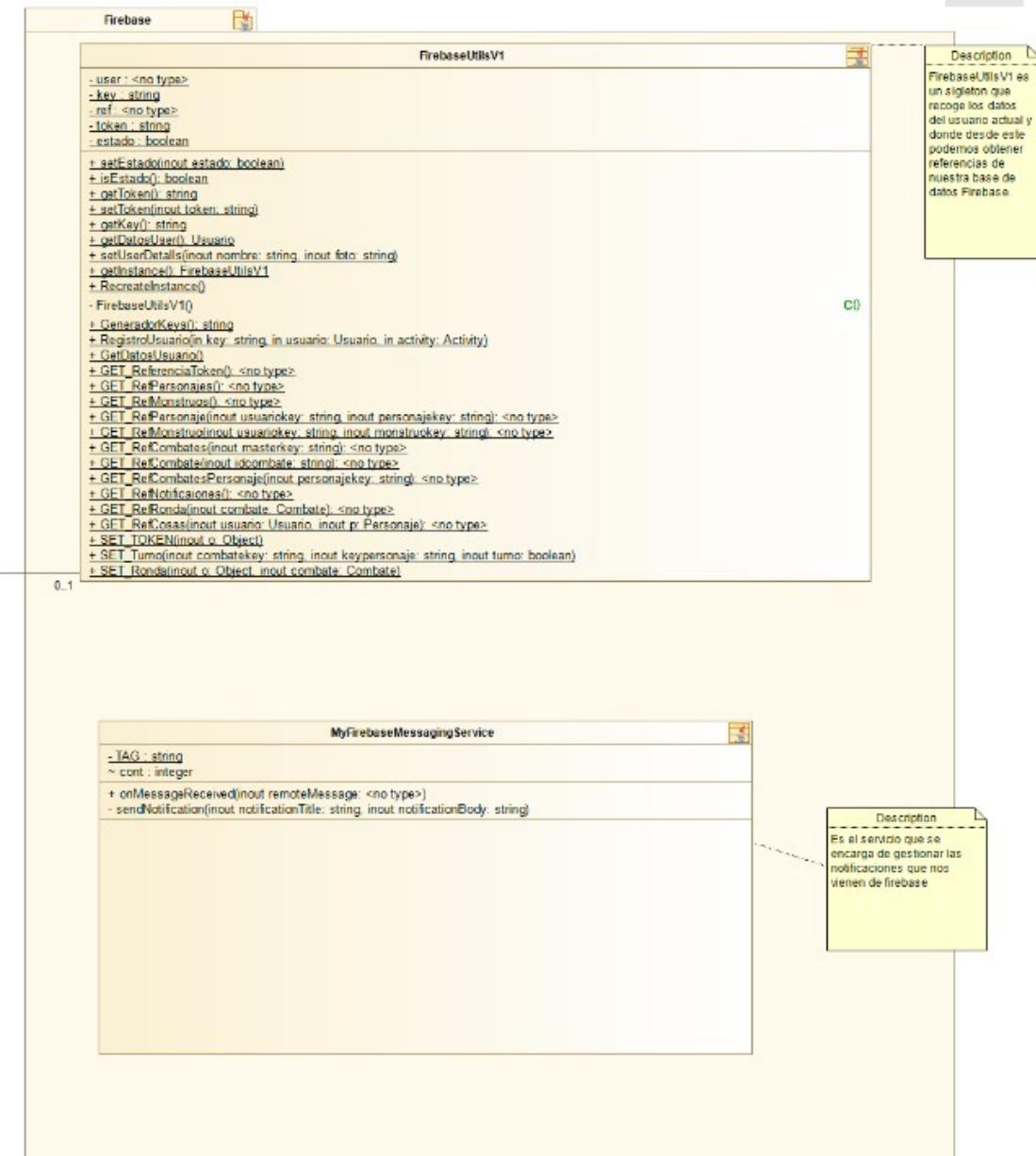
**Vistas:** Se encarga de agrupar los objetos que tienen que ver con la visualización de la aplicación. Encontraremos tres distinciones entre ellos: Adapters, Diálogos, y Fragments.

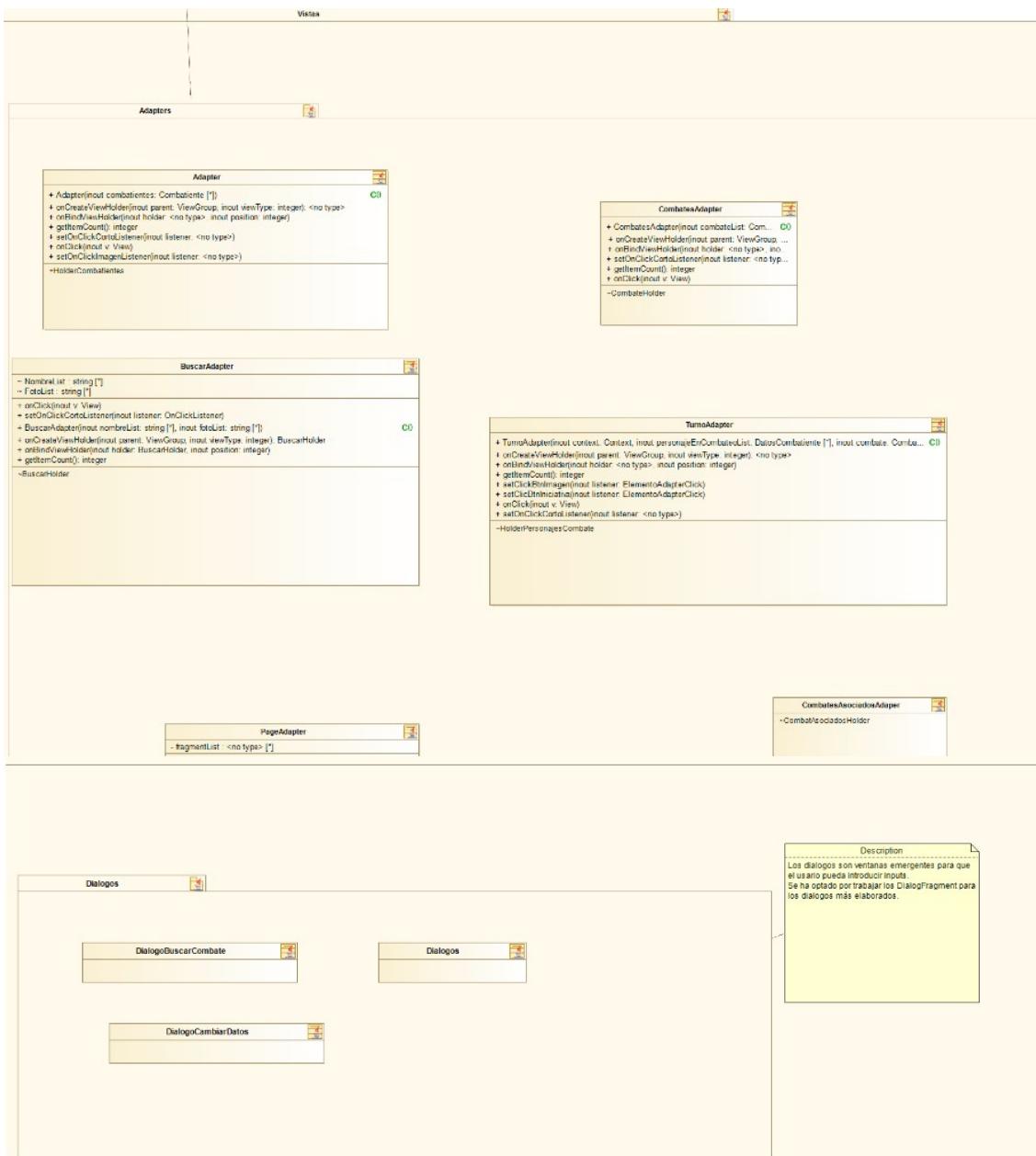
Para finalizar encontraremos las tres actividades,:Actividad, LoginActivity y MainActivity. Estas dos ultimas heredan de la primera que es de tipo abstracto.

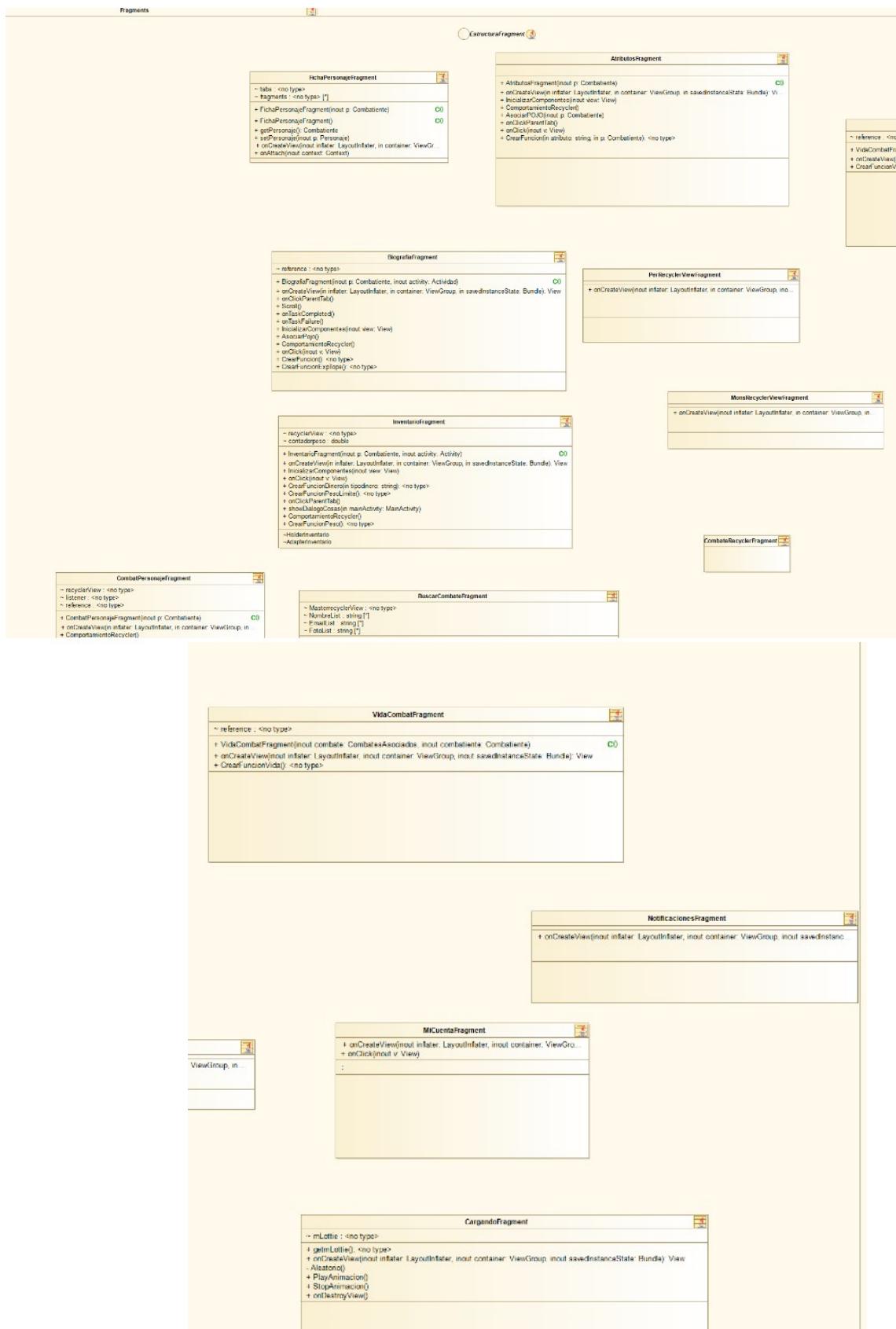
A continuación veremos el diagrama completo por trozos.







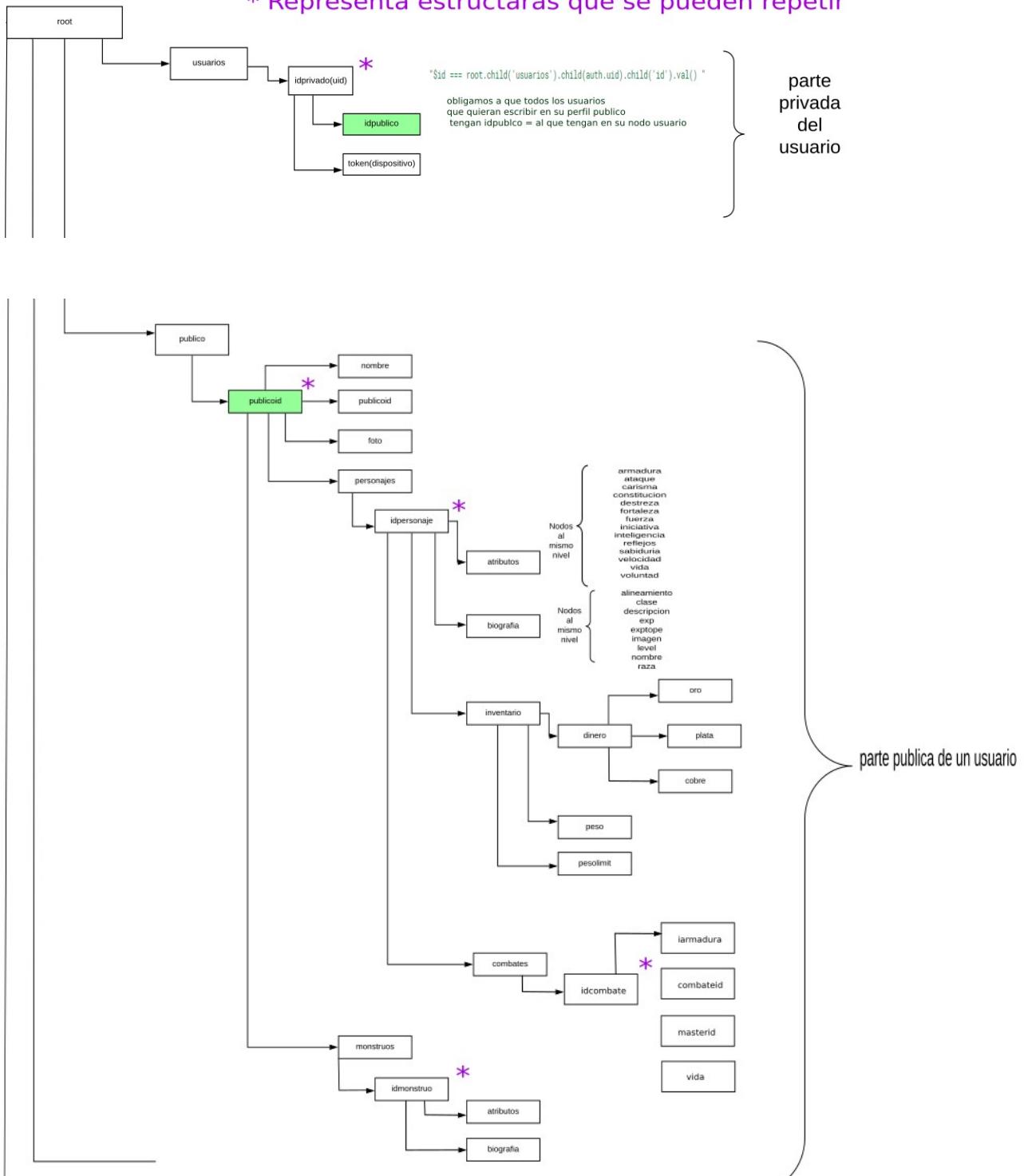


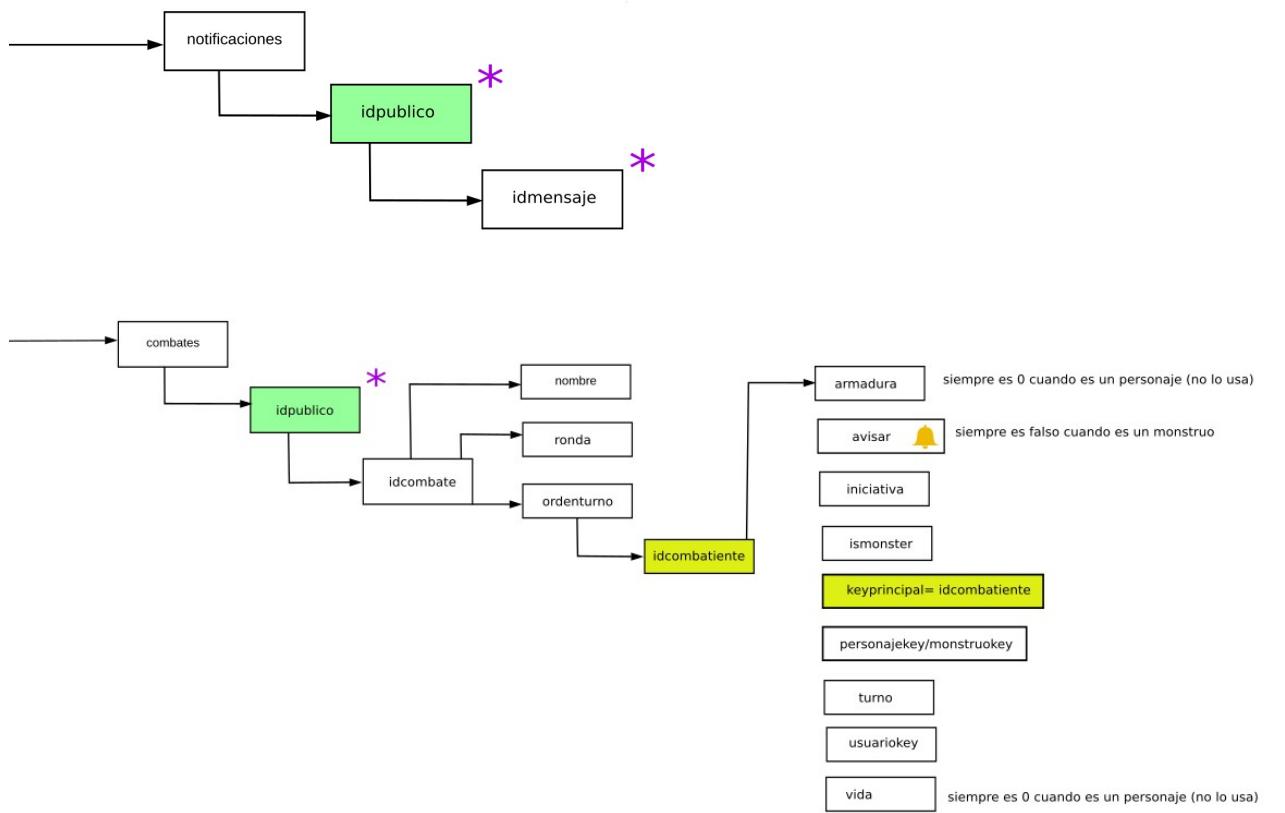


## A nivel de base de datos

La estructura de la base de datos seguirá el siguiente árbol. En el que podemos distinguir cuatro hijos directos de root/rodofroll (usuarios,publico,notificaciones,combates).

\* Representa estructuras que se pueden repetir



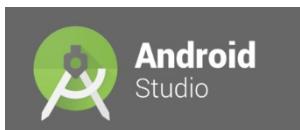


## Codificación

Para la realización de este proyecto hemos utilizado 2 lenguajes de programación: java en el entorno de Android Studio, y javascript para realizar las funciones de Firebase Functions.

Al ser un programa enfocado en móvil se ha requerido usar el entorno de Android Studio, y este dispone solo de dos lenguajes para codificar, Kotlin o Java. Se ha escogido este último por ser el visto en clase.

Las herramientas para el desarrollo del proyecto han sido:



**Android Studio** es un entorno de desarrollo para realizar programas que se van a ejecutar en el sistema operativo de Android.



**Firebase** es una plataforma para el desarrollo de aplicaciones móviles y web.



**GitHub** es un sistema para controlar versiones del proyecto.



**Node.js** es un entorno de ejecución multiplataforma , para la capa del servidor basado en JavaScript. Su uso se ha requerido para poder implementar las funciones de Firebase.

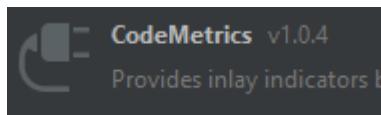


**Atom** es un editor de código fuente con soporte para múltiples plug-in escritos en Node.js.



**Postman** es una herramienta que permite el envío de peticiones HTTP REST.

Otros utensilios son:



**CodeMetrics** es un plugin de Android Studio que nos provee de indicaciones basadas en las customización y complejidad del código.



**Modelio** es una herramienta UML, que sirve para la creación de diagramas enfocados a la explicación de software.



**Inkscape** es un editor de gráficos vectoriales libre y de código abierto, para la creación de logotipos con formato vectorial.



**Asset Studio** es una herramienta que está integrada en Android Studio , y la he utilizado para pasar las imágenes de vector a formato xml.



**Lottie** es un banco de animaciones gratuitas para incluir en distintos proyectos.

## Aspectos relevantes

Podríamos empezar dando un primer vistazo al Manifest de la aplicación, aquí podemos encontrar dos activitys y dos servicios.

```
<manifest version="1.0" encoding="utf-8">
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.rodoroll">
    <uses-permission android:name="android.permission.INTERNET"/><uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application
        android:allowClearTaskTraffic="true"
        android:icon="@mipmap/ic_launcher"
        android:label="RodOfRoll"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".LoginActivity"
            android:screenOrientation="portrait"
            android:theme="@style/SplashTheme"
            >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity"
            android:screenOrientation="portrait"
            >
            </activity>
        <meta-data
            android:name="preload_fonts"
            android:resource="@array/preloaded_fonts" />
        <service
            android:name=".MyFirebaseMessagingService"
            android:exported="false"
            <intent-filter>
                <action android:name="com.google.firebase.MESSAGING_EVENT" />
            </intent-filter>
        </service>
        <service android:name=".Objetos.ListenUserService" android:stopWithTask="false" android:description="Es un servicio que borra datos propios de RodOfRoll cuando el usuario se loguea...">
        </service>
    </application>
</manifest>
```

} Login Activity es la actividad con la que se inicia el programa

} MainActivity es la actividad que define como tal la aplicación

} MyFirebaseMessagingService es el servicio que se encarga de recibir notificaciones de Firebase

} ListenUserService se encarga de comprobar que el usuario no ha cambiado de dispositivo

## Login Activity

En esta parte lo más curioso a destacar son la forma en cómo se comprueban las credenciales del usuario y los distintos errores de login que puede mostrar la aplicación.

```
//Metodo que devuelve el tipo de error
public String ErroresAuth(Exception f) { Complexity is 10 It's time to do something...
    if (FirebaseNetworkException.class.equals(f.getClass())) {
        return "Error de red";
    } else if ( FirebaseAuthInvalidCredentialsException.class.equals(f.getClass())) {
        return "Contraseña incorrecta";
    } else if ( FirebaseAuthUserCollisionException.class.equals(f.getClass())) {
        return "Usuario ya existente";
    } else if ( FirebaseAuthInvalidUserException.class.equals(f.getClass())) {
        return "Usuario inexistente";
    }
    return "Error";
}
```

## MainActivity

De esa actividad voy a destacar:

**El tirador de dados:** De aquí quería destacar la función de WhatsApp.



Al pulsar en este botón podremos enviar una tirada en formato texto a cualquier usuario que tengamos en WhatsApp. Es un código bastante sencillo pero para esta aplicación puede ser bastante útil.

```
whatsappimage.setOnClickListener(v) {
    //Creamos un Intent con la accion de enviar
    Intent whatsappIntent = new Intent(Intent.ACTION_SEND);
    //Ponemos el tipo de dato que vamos a enviar texto plano
    whatsappIntent.setType("text/plain");
    //Seteamos hacia que aplicacion tiene que ir
    whatsappIntent.setPackage("com.whatsapp");
    //Enviamos los datos
    whatsappIntent.putExtra(Intent.EXTRA_TEXT, value: "*" + nombretiradatextview.getText().toString() + "*" +
        "\n" + "Informacion:" + info.getText().toString() + "\n" +
        "Resultado:" + resultado.getText().toString() + "\n" + "Dados:" + tiradastextview.getText().toString());
    try {
        //Iniciamos
        activity.startActivity(whatsappIntent);
    } catch (android.content.ActivityNotFoundException ex) {
        //Si hay un error mostramos un Toast
        Toast.makeText(activity, text: "Whatsapp no instalado", Toast.LENGTH_SHORT).show();
    }
};
```



**El control de los turnos:** Me podría extender mucho en esta parte pero voy a destacar el método que controla el cambio de turno

```
//Metodo que ocurre al pulsar el boton de pasar turno
public void PasarTurno(List<Combate.DatosCombatiente> personajeEnCombateoList, Combate combate) { Complexity is 10 It's time to do something...
    boolean todosfalso= true;
    Combate.DatosCombatiente pos=null;
    int posi=0;

    //Se busca saber si alguno de los personajes en el combate tiene el campo turno a verdadero
    //Si es asi se guarda ese personaje y su posicion
    for(int i =0;i<personajeEnCombateoList.size();i++){
        if(personajeEnCombateoList.get(i).getTurno()){
            todosfalso=false;
            pos=personajeEnCombateoList.get(i);
            posi = i;
        }
    }
    //Si ninguno tiene el campo a true es decir si todos son falsos se asigna el turno al combatiente con la mayor iniciativa el cual
    //en nuestra lista seria el ultimo (recordemos que la lista esta ordenada ascendemente de mas a mas)
    //Tambien seteamos el valor de la ronda del combate y la inicializamos a 1
    if(todosfalso){
        //Los combatientes vienen de forma ascendente , asi que como nosotros necesitamos los datos de forma descendente consideramos al ultimo el primero
        personajeEnCombateoList.get(personajeEnCombateoList.size()-1).setTurno( turno: true, combate);

        FirebaseUtilsV1.SET_Ronda( o: 1,combate);

        }
    //Si alguno es verdadero como lo que queremos es pasar el turno lo inicializamos al false
    else{
        personajeEnCombateoList.get(posi).setTurno( turno: false, combate);
        //Como los turnos funcionan como un circulo al pasar el turno desde el ultimo se lo tendriamos que pasar al primero
        //Pero en este caso el primero de la lista es el ultimo asi que lo tenemos que hacer al revés
        //La cuestion es que al siguiente le ponemos el valor de turno a true
        if(posi==0){
            personajeEnCombateoList.get(personajeEnCombateoList.size()-1).setTurno( turno: true, combate);
            //Aque estamos recuperando el valor de la ronda y al pasar del ultimo al primero tenemos que aumentarlo en uno
            int rondaint = combate.getRonda();
            rondaint++;

            combate.setRonda(rondaint);
            ronda.setText(String.valueOf(combate.getRonda()));

        }
        else{
            personajeEnCombateoList.get(posi-1 ).setTurno( turno: true, combate);
        }
    }
}
```

y como funciona la opción de avisar a los usuarios de que es su turno.

```
//Evento que ocurre al clickear en la imagen del cardview con forma de campana crea un toast para informar al usuario que avisara
//al jugador del personaje de que es su turno
adapter.setOnClickListener((personEnCombate) > {
    Toast.makeText(getApplicationContext(), text: "Avisando...", Toast.LENGTH_SHORT).show();

    FirebaseUtilsV1.GET_RefCombate(combate.getKey()).child("ordenturno").child(personEnCombate.getKeyprincipal()).child("avisar").setValue(true);
});
```

Esto provoca que en [combates/publicoid/combatesid/ordenturno/combatienteid/avisar](#) un cambio que se intercepta por la siguiente función de firebase:

```
1 //Esta funcion se ejecuta cuando el hijo avisar de un combatiente ha cambiado
2 exports.AvisarPersonaje = functions.database.ref('combates/{publicoId}/{combateId}/ordenturno/{idpersonaje}/avisar')
3 .onWrite(async(snapshot,context)=>{
4 //Se comprueba que el cambio no se deba a una eliminacion de datos
5 if(!snapshot.after.exists()){
6   return null;
7 }
8 //Se comprueba que el nuevo valor sea true si es asi se creara una notificacion al propietario del personaje
9 if(snapshot.after.val()===true){
0
1   var retorno = null;
2     //Sacamos del personaje su publicoID y le notificamos
3   |
4     var db = admin.database();
5     var ref = db.ref("combates/"+context.params.publicoId+"/"+context.params.combateId+"/ordenturno/"+context.params.idpersonaje);
6     var personajevalor = (await ref.once("value")).val();
7
8     var refcombatee = db.ref("combates/"+context.params.publicoId+"/"+context.params.combateId+"/nombre");
9     var combate = (await refcombatee.once("value")).val();
0     console.log(combate);
1     var idpublico = personajevalor.usuariokey;
2
3 //obtenemos el token del jugador al que se tiene que avisar
4     var token = await buscarTokenDispositivo(idpublico);
5
6 //se crea la notificacion y se guarda esta en notificaciones
7     var payload =crearNotificacion("Es tu turno","En el combate",combate,idpublico);
8
9 //se vuelve a poner a false el valor avisar
0     admin.database().ref("combates/"+context.params.publicoId+"/"+context.params.combateId+"/ordenturno/"+context.params.idpersonaje+"/avisar").set(false);
1     //Ponemos a false el valor
2
3 //solo se envia notificacion a los usuarios que esten activos es decir que su token de dispositivo no sea nulo
4     if(token!==null){
5       retorno = admin.messaging().sendToDevice(token,payload);
```

## La Tarea Asincrona de carga

```
import ...
//Es la tarea que gestiona la recuperacion de los datos del usuario mientras muestra una pantalla de carga
public class AsynTarea extends AsyncTask<Void,Void(Void> {
    Complexity is 11 You must be kidding

    FragmentManager fm;
    CargandoFragment fragment;
    //Es una interfaz que nos servira para determinar si se ha completado la recuperacion de los datos del usuario
    private OnTaskCompleted listener;
    //Es el tiempo en el que se inicia la tarea
    long inicio;

    //Le pasaremos la interfaz y el fragment manager desde la MainActivity
    public AsynTarea(OnTaskCompleted listener,FragmentManager fm) throws InterruptedException {
        this.fm=fm;
        this.listener=listener;
    }

    //Después de terminar la tarea se llama al metodo de la interfaz
    @Override
    protected void onPostExecute(Void aVoid) { Complexity is 3 Everything is cool!
        super.onPostExecute(aVoid);
        if (listener != null) {
            listener.onTaskCompleted();
        }
    }

    //Antes de ejecutar la tarea creamos el fragment CargandoFragment y remplazamos el fragment anterior por este , guardamos el tiempo en el que se inicia la app
    @Override
    protected void onPreExecute() {
        super.onPreExecute();

        fragment= new CargandoFragment();

        FragmentTransaction fragmentTransaction = fm.beginTransaction();
        fragmentTransaction.replace(R.id.fragment_container,fragment);
        fragmentTransaction.commit();
        inicio=System.currentTimeMillis();
    }

    //La tarea espera a que los datos del usuario tengan un estado no nulo
    @Override
    protected Void doInBackground(Void... voids) { Complexity is 10 It's time to do something...
        double tiempo;
        //Sirve para visualizar el contador una sola vez
        int contador_info=0;
        do{
            //Se cuenta el tiempo duracion entre el inicial y el actual
            tiempo = (double) ((System.currentTimeMillis() - inicio)/1000);

            //Si por algun motivo se sobrepasa mas de 20 segundos se muestra un snackbar informando al usuario de que pruebe a reiniciar la aplicación
            if(tiempo>20&&contador_info==0){
                Snackbar.make(fragment.getView(), text: "Pruebe a reiniciar la aplicación", Snackbar.LENGTH_LONG).show();
                contador_info++;
            }
        }while (!FirebaseUtilsV1.isEstado());

        //Sirve para mostrar la animación de inicio por lo menos durante 3 segundos
        try {
            Thread.sleep( millis: 3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        return null;
    }
}
```

```
public class CargandoFragment extends Fragment { Complexity is 6 It's time to do something...

    //Es el objeto Lottie que sirve para contener la informacion la animacion
    LottieAnimationView mLottie;

    public LottieAnimationView getmLottie() { return mLottie; }

    @Nullable
    @Override
    //Al crear el fragment se carga la vista del fragment y se inicializa el objeto lottie
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

        View view = inflater.inflate(R.layout.cargando_layout,container, attachToRoot: false);
        mLottie= view.findViewById(R.id.animacion);

        //Se llama al metodo aleatorio
        Aleatorio();
        return view;
    }

    //Se procede a decidir aleatoriamente entre las tres animaciones aleatorias que hay
    private void Aleatorio() { Complexity is 5 Everything is cool!

        Random aleatorio = new Random();
        int n = aleatorio.nextInt( bound: 3);

        switch (n) {
            case 0:
                mLottie.setAnimation("brujo1.json");

                break;
            case 1:
                mLottie.setAnimation("brujo2.json");
                break;
            case 2:
                mLottie.setAnimation("brujo3.json");
                break;
        }

        //Se ejecuta la animacion
        PlayAnimacion();
    }
    public void PlayAnimacion() { mLottie.playAnimation(); }
    public void StopAnimacion() { mLottie.cancelAnimation(); }

    //Al destruir el fragment se para la animacion para prevenir errores
    @Override
    public void onDestroyView() {
        super.onDestroyView();
        StopAnimacion();
    }
}
```

## MyFirebaseMessagingService

```
public void onMessageReceived(RemoteMessage remoteMessage) { Complexity is 3 Everything is cool!
    String notificationTitle = null, notificationBody = null;
    //miramos que el mensaje no llegue vacio y creamos dos variables que corresponderan con el titulo de la notificacion y el cuerpo
    if(remoteMessage.getData().size()>0){
        //Guardamos el titulo
        notificationTitle = remoteMessage.getData().get("title");
        //Guardamos el cuerpo de los datos que nos llegan
        notificationBody = remoteMessage.getData().get("body");
    }
    /*if (remoteMessage.getNotification() != null) {
        Log.d(TAG, "Message Notification Body: " + remoteMessage.getNotification().getBody());
        notificationTitle = remoteMessage.getNotification().getTitle();
        notificationBody = remoteMessage.getNotification().getBody();
    }*/
    sendNotification(notificationTitle, notificationBody);
}

private void sendNotification(String notificationTitle, String notificationBody) { Complexity is 5 Everything is cool!
    NotificationManager nm;
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        //Creamos el nombre del canal
        CharSequence name = "Notificaciones";
        String description = "Nuevos datos ";
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel( id: "RodRoll", name, importance);
        channel.setDescription(description);

        nm= getSystemService(NotificationManager.class);
        nm.createNotificationChannel(channel);

        NotificationCompat.Builder builder = new NotificationCompat.Builder(getApplicationContext(), channel.id: "RodRoll")
            .setSmallIcon(R.drawable.carta)
            .setContentTitle(notificationTitle)
            .setContentText(notificationBody)
            .setVibrate(new long[]{500, 1000})
            .setPriority(NotificationCompat.PRIORITY_HIGH);

        nm.notify(cont, builder.build());
        if(cont>20){
            cont++;
        }
        else{
            cont=0;
        }
    }
}
```

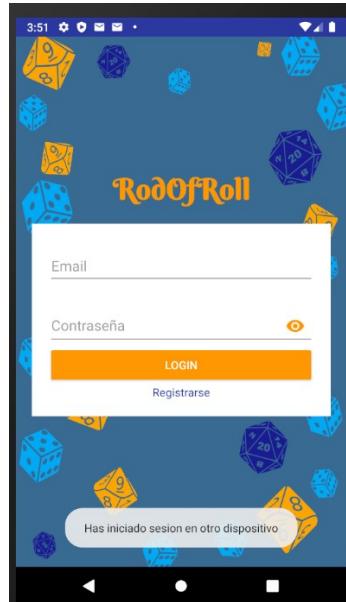
Este servicio escucha los datos que envía Firebase. El siguiente código de las funciones de Firebase es clave para enviar estos datos:

```
1 //Es una funcion que crea el formato que tendra la notificacion que enviaremos
2 function crearNotificacion(titulo,mensaje,valor,id){
3     var payload = {
4         data:{
5             title:titulo,
6             body: mensaje+" "+valor
7         }
8     };
9     //Escribimos en la base de datos la notificacion para que quede registrada para el usuario
10    admin.database().ref("notificaciones/"+id).push().set(title+""+mensaje+" "+valor);
11    return payload;
12 }
```

```
    retorno = admin.messaging().sendToDevice(token,payload);
```

## ListenUserService

Es el servicio que se encarga de escuchar si se ha iniciado el usuario en otro dispositivo. De esta manera se evita que un usuario pueda estar en dos dispositivos al mismo tiempo.



```
@Override
public int onStartCommand(Intent intent, int flags, int startId) { Complexity is 9 It's time to do something...
    //Al iniciar el servicio se crea un escuchado que apunta al token del usuario en base de datos
    FirebaseUtilsV1.GET_RefenciaToken().addValueEventListener(new ValueEventListener() { Complexity is 7 It's time to do something...
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) { Complexity is 6 It's time to do something...
            try {
                //Si el token que hay en base de datos no es el mismo que el de este dispositivo y este token en base de datos no esta vacio se entra a la condicion
                if (!((String) dataSnapshot.getValue()).equals(FirebaseUtilsV1.getToken())) && !((String) dataSnapshot.getValue()).isEmpty()) {
                    //Se muestra un toast de que se ha iniciado en otro dispositivo
                    Toast.makeText(getApplicationContext(), text: "Has iniciado sesion en otro dispositivo", Toast.LENGTH_LONG).show();
                    //Se pausa 3 segundos para poder mostrar el toast
                    try {
                        Thread.sleep( millis: 3000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    //Se sale de la aplicacion
                    FirebaseAuth.getInstance().signOut();
                    Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                    FirebaseUtilsV1.Borrar();
                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                    Toast.makeText(getApplicationContext(), text: "Has iniciado sesion en otro dispositivo", Toast.LENGTH_LONG).show();
                    startActivity(intent);
                }
            } catch (NullPointerException ex){
            }
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
    });
}
```

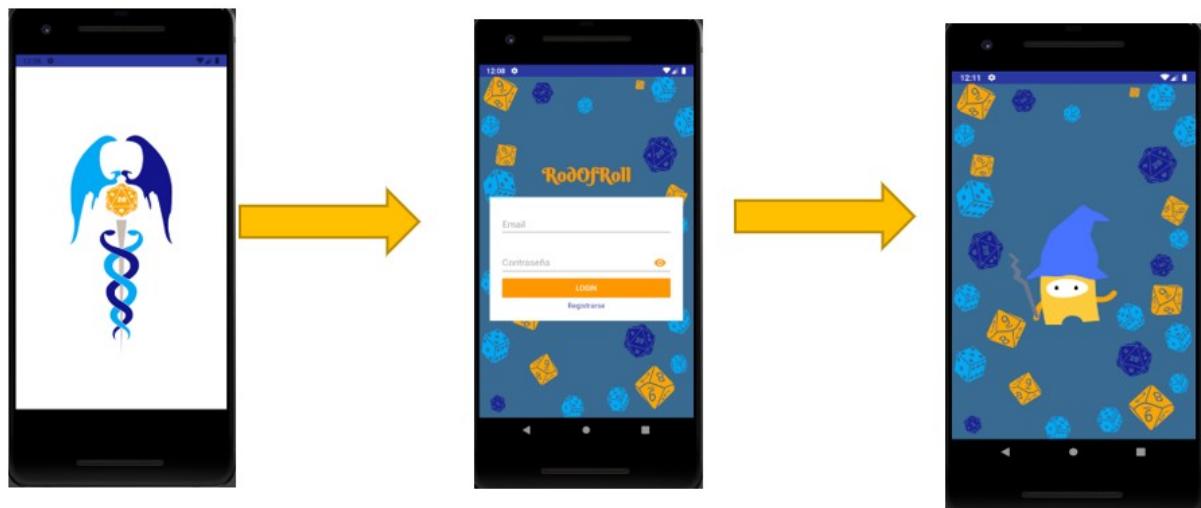
## Manual de usuario

### Inicio de la aplicación

Al iniciar la aplicación aparecerá un splashscreen y a continuación la aplicación nos pedirá un email y una contraseña.

Procederemos a loguearnos o registrarnos pulsando el botón amarillo. Para cambiar entre login o registro pulsaremos el texto donde pone Registrar/Login.

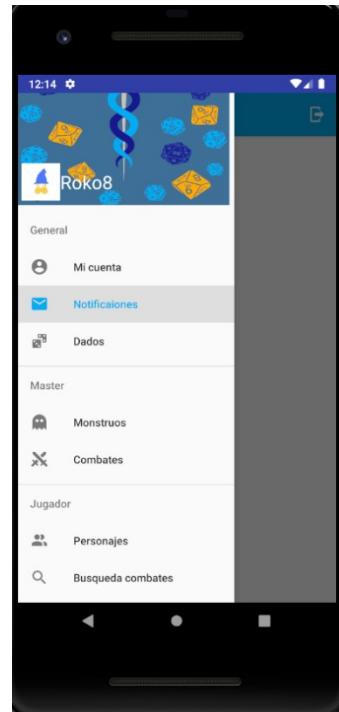
Una vez logueado esperaremos a que la aplicación enlace el usuario registrado con los datos esenciales de este.



## Despliegue del Menú

Haremos un swipe de izquierda a derecha en la pantalla o daremos al botón de las tres barritas sitiado en la esquina superior izquierda y se nos presentara el menú lateral. Distinguiremos entre tres tipos de bloques : general,máster,jugador.

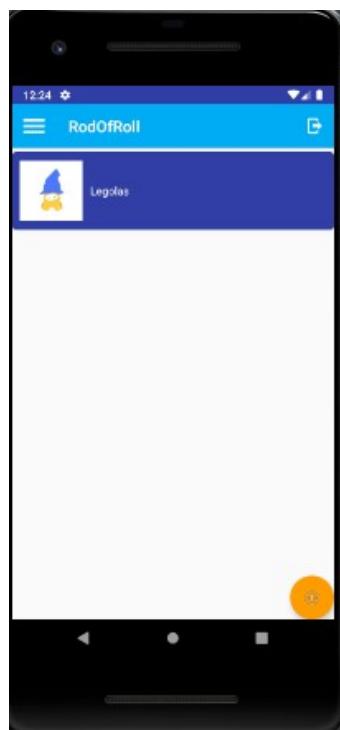
Empezaremos explicando desde el último hasta el primero.



## Bloque Jugador

### Creación de un personaje

Pulsaremos la opción de Personajes y nos saldrá una pantalla en blanco con un botón circular con el símbolo de agregar, pulsaremos en este y nos mostrara un dialogo pidiendo el nombre y la imagen de nuestro nuevo personaje.



La pantalla en blanco se actualizara y nos mostrara el personaje que hemos creado. Para ver detalles sobre este pincharemos sobre él y si queremos eliminarlo desplazaremos el cardview a un lateral.

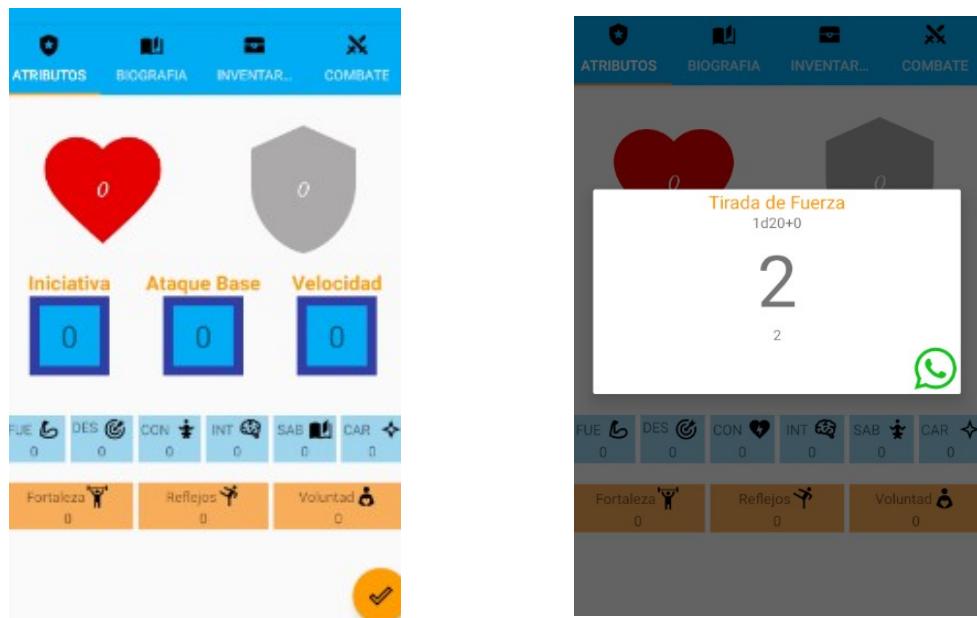
Al entrar podremos observar un TabLayout con las distintas características del personaje que podremos modificar o ver.

### Atributos

En Atributos podremos modificar la vida y la armadura del personaje pulsando a las imágenes del corazón y el escudo, así como los demás atributos que están en los rectángulos.

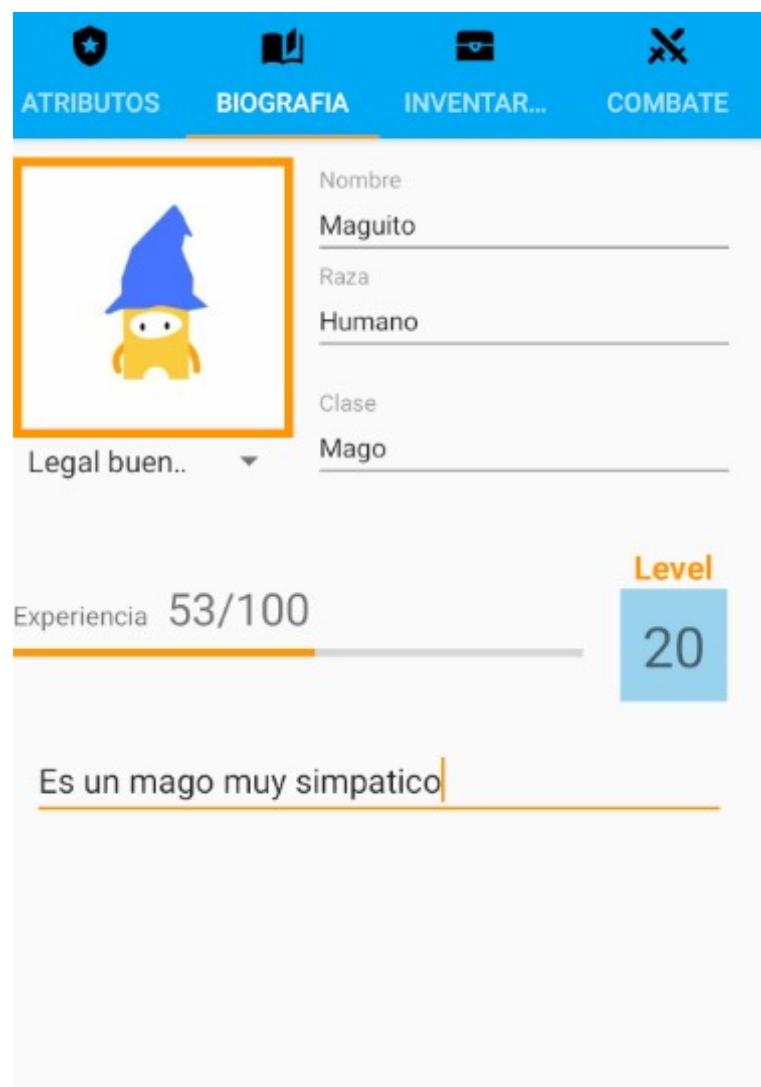
Si presionamos durante mas tiempo sobre los rectángulos azules claros y los naranjas podremos realizar una tirada de característica/salvación.

Pinchando sobre el icono de WhatsApp podremos enviar la tirada a uno de nuestros contactos.



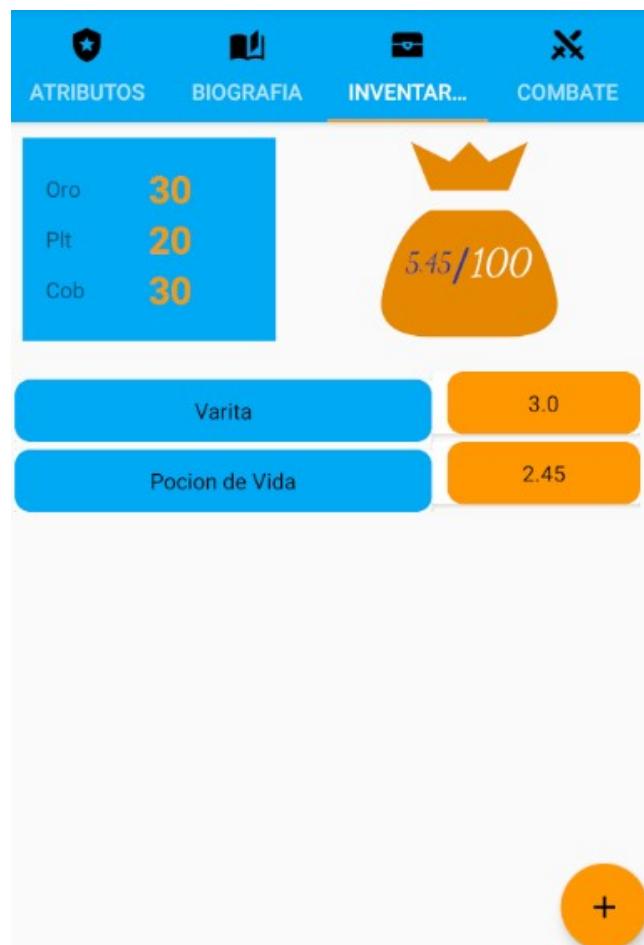
## Biografía

Podremos ver la imagen del personaje así como su nombre , su clase ,su raza , su descripción su nivel, su experiencia y su alineamiento.



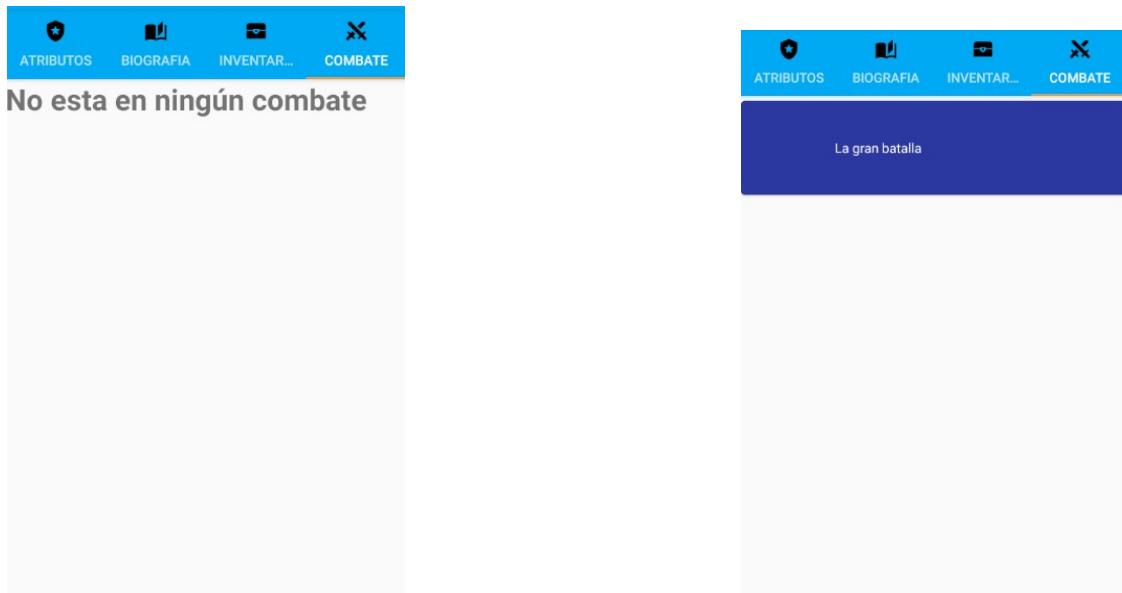
## Inventario

Pretende ser una manera de gestionar el dinero de los personajes ,así como de tener un recuento de los objetos que tiene y lo que pesan. Se podrán añadir objetos con el botón de suma. Al añadir nos pedirá el nombre del objeto y su peso y se comprobara el peso de este con la suma del peso de los demás objetos .La suma de ambos tendrá que ser menor o igual a la carga máxima que puede llevar el usuario



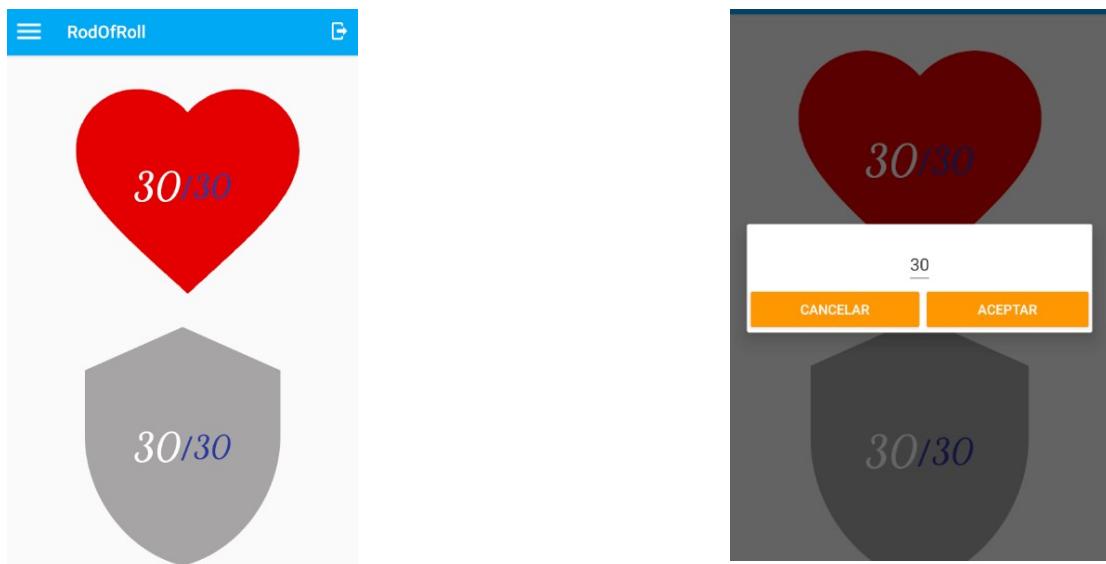
## Combate

En esta parte podremos ver en cuantos combates esta nuestro personaje y podremos modificar la vida y la armadura que tiene en cada uno de ellos. Si un personaje no esta en ningún combate nos aparecerá un texto informándonos.



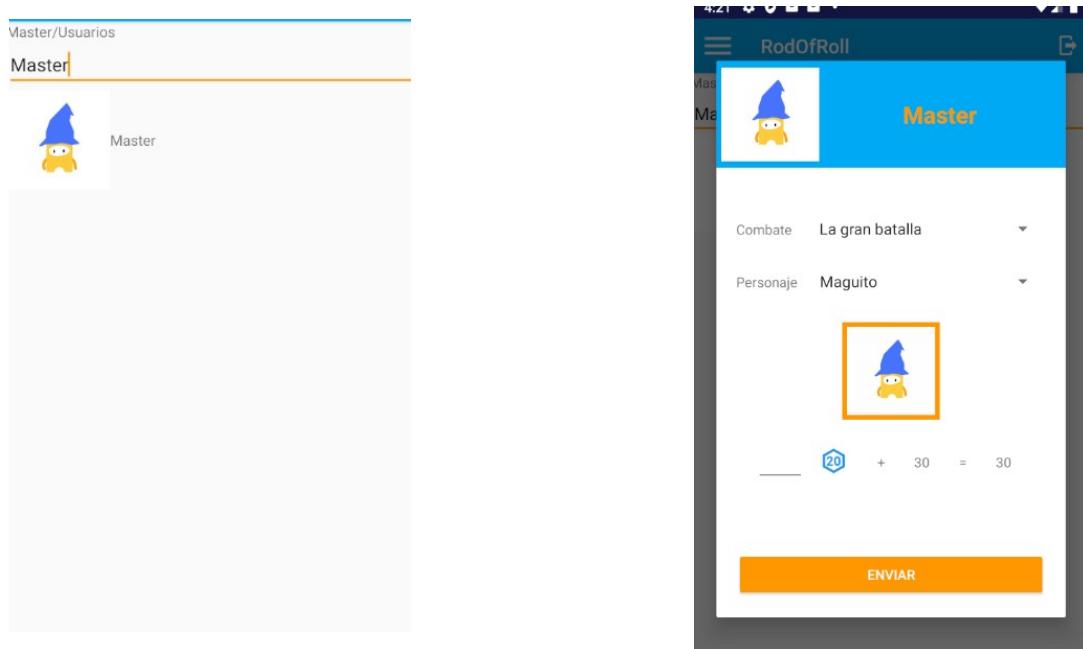
Al entrar en el combate aparecerá la siguiente pantalla.

Pulsaremos en el corazón y en la armadura y podremos cambiar la vida y la armadura temporal.

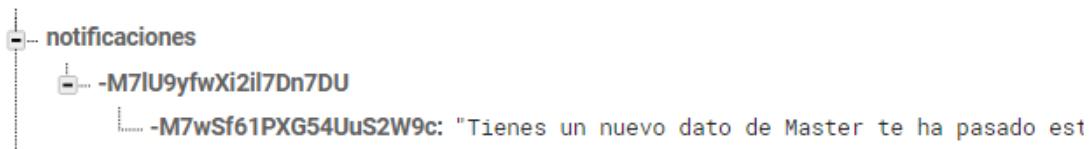


## Búsqueda de combates

Al pulsar en esta opción nos saldrá una barra de búsqueda para buscar al usuario que va a ser nuestro máster, lo buscaremos por su nombre. Lo seleccionaremos y nos aparecerá un dialogo donde podremos ver los combates que tiene. Elegiremos un combate y el personaje que queremos enviar a este. Una vez elegidos procederemos a darle al botón del dado para realizar una tirada de iniciativa



Después le pulsaremos al botón enviar. Esto desencadenara una notificación en el usuario elegido



✉ RodOfRoll • now

Tienes un nuevo dato de Roko  
te ha pasado este personaje Maguito

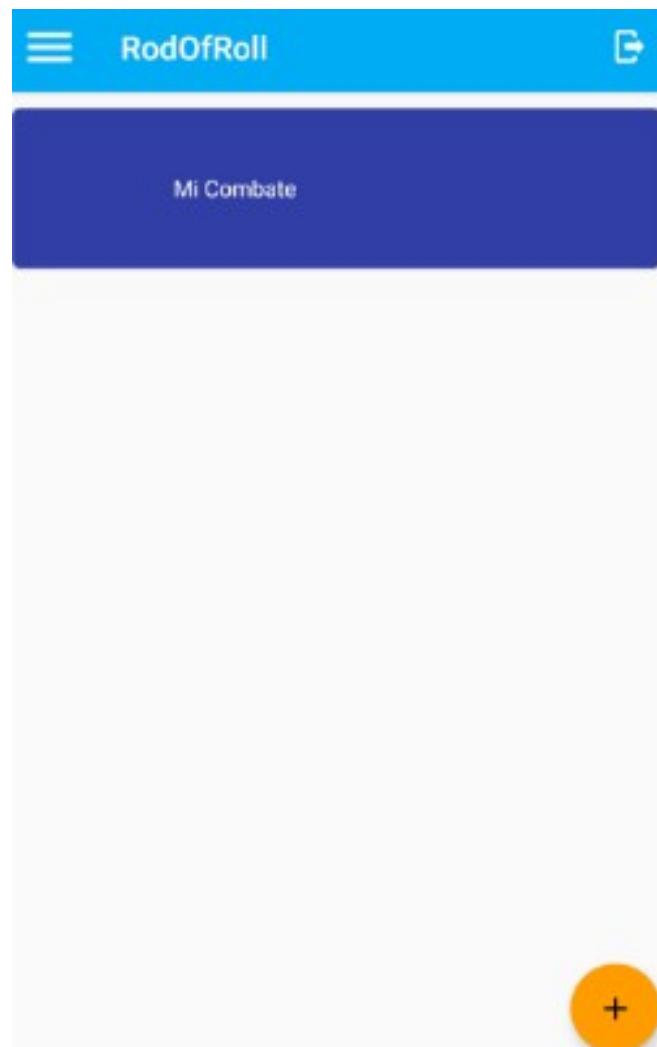
## Bloque Máster

### Creación de un monstruo

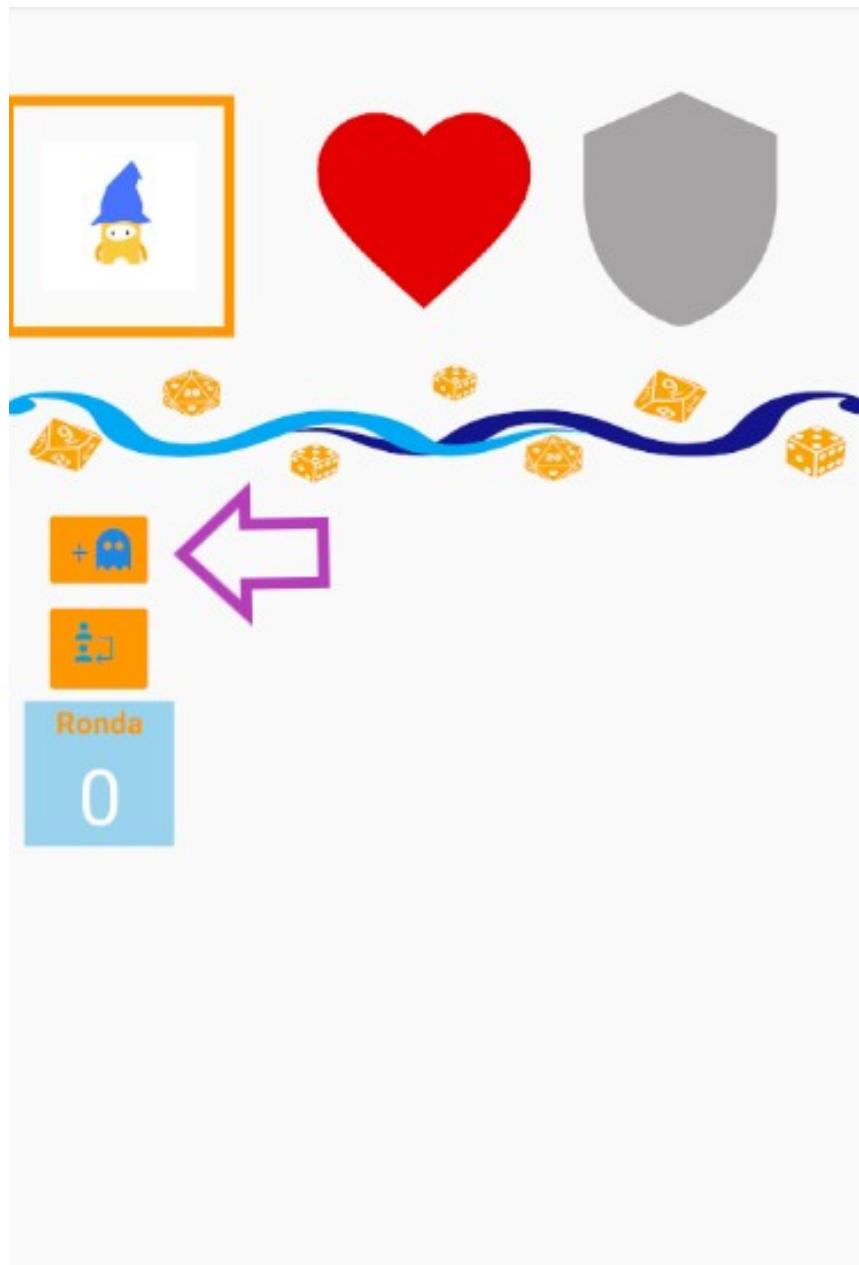
Es muy similar a la creación de un personaje.

### Creación de un combate

Tendremos una pantalla en blanco, pulsaremos el botón de agregar introduciremos un nombre para el combate y se nos actualizará nuestra lista de combates. Podremos borrar el combate haciendo swipe sobre este.



Pulsaremos en el combate creado y entraremos en nuestro combate. Esperaremos a que entren jugadores , e iremos agregando monstruos a través del botón con el icono del fantasma.



En la parte de arriba podremos ver el panel informativo. Cuando pulsemos en alguno de nuestros combatientes este panel se actualizara y nos mostrara el nombre, la foto, la vida y la armadura temporal y máxima de ese combatiente. Podremos cambiar las temporales solo si este es un monstruo.



Pinchamos sobre el corazón o el escudo para cambiar la armadura o la vida temporal. De esta manera nos aparecerá un dialogo para poder cambiar estos atributos.

Para pasar el turno pulsaremos al botón de turno.



Si es la primera vez que se ha pulsado en ese combate, podremos observar que el primer personaje de nuestra lista de iniciativas tiene un borde color amarillo, si volvemos a pulsar pasaremos el turno al siguiente en la lista.

The image shows two side-by-side initiative lists. On the left, the first card for 'Dragon' has a yellow border around its 'Iniciativa 18' box. On the right, after a turn is taken, the first card for 'Maguito' has a yellow border around its 'Iniciativa 5' box, and there is a yellow bell icon to the right of the list.

Personaje	Iniciativa
Dragon	18
Maguito	5
Dragon	1

Cuando el turno llega a un personaje podremos observar que aparece el símbolo de una campana al pulsar sobre esta avisaremos al jugador de que le ha llegado su turno.

The image shows a mobile application interface. On the left, a list of characters with their initiative values: 'Maguito' (Iniciativa 5) and 'Dragon' (Iniciativa 1). The 'Maguito' card has a yellow border. A yellow bell icon is positioned to the right of the list. On the right, there is a message from 'RodOfRoll' saying 'ahora'. Below it, a message says 'Es tu turno en el combate Mi Combate'. At the bottom left, a circular button contains the text 'Avisando...'.

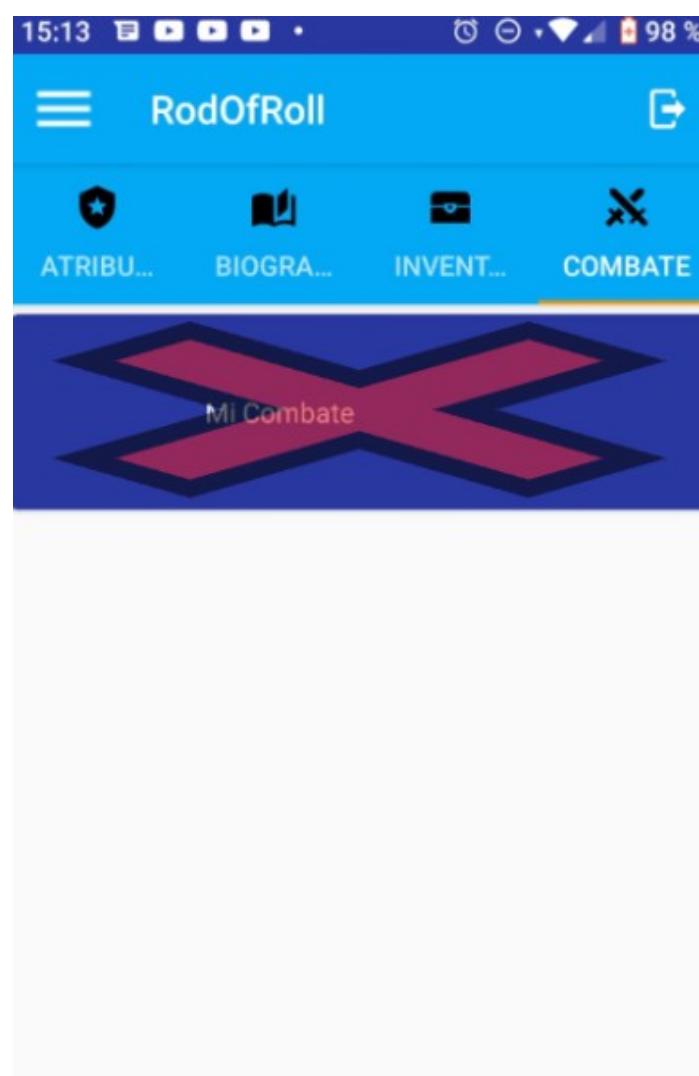
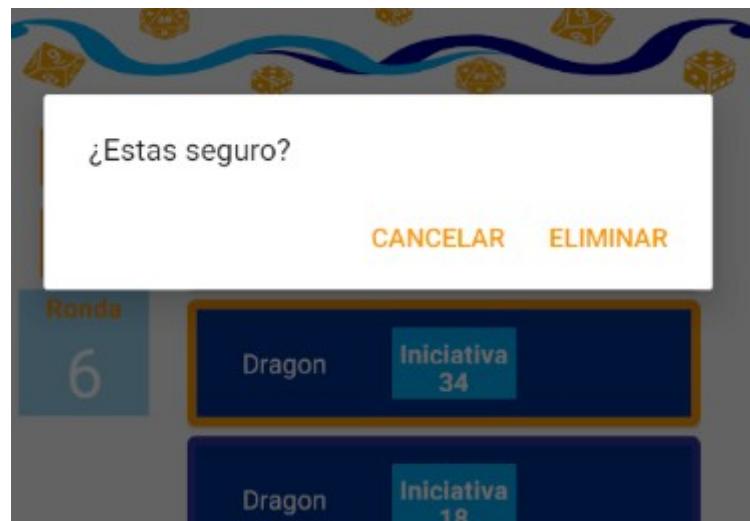
Personaje	Iniciativa
Maguito	5
Dragon	1

Cuando el turno esta sobre el último de los participantes y volvemos a pasar el turno el turno vuelve al primero, al hacer esto el elemento ronda aumenta en uno. Podremos modificar el valor de la ronda pinchando sobre ella.



Podemos observar que todos los elementos de nuestra lista de iniciativas tienen un recuadro azul de iniciativa al pulsar sobre este podremos cambiar la iniciativa. Al cambiar el valor de esta reordenaremos la lista de iniciativas pero el turno no cambiara.

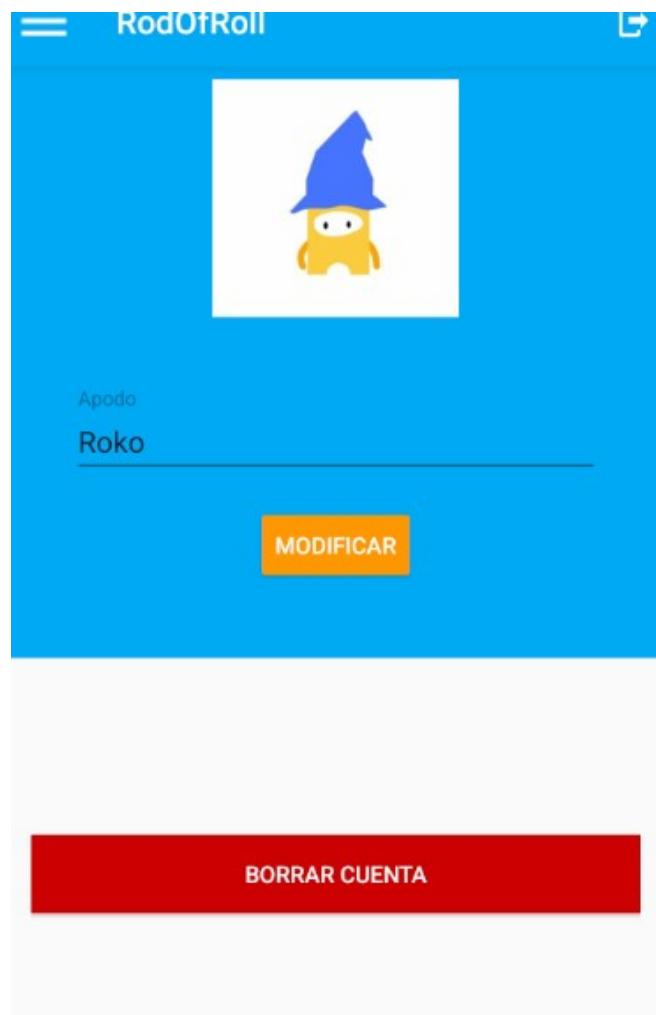
Para eliminar a cualquiera de nuestros combatientes solo tendremos que hacer un “swipe” sobre ellos. Al eliminar un personaje en la lista de combates de este se eliminara el combate donde estaba enlazado.



## Bloque General

### Mi cuenta

En este apartado podremos modificar la imagen de nuestra cuenta y nuestro nombre. También podremos borrar nuestro usuario con todo nuestro contenido.



## Notificaciones

Aquí podremos tener control de todas las notificaciones que nos han llegado. Podremos borrarlas todas pulsando el botón de la basura

Tienes un nuevo dato de Roko te ha pasado este personaje Maguito



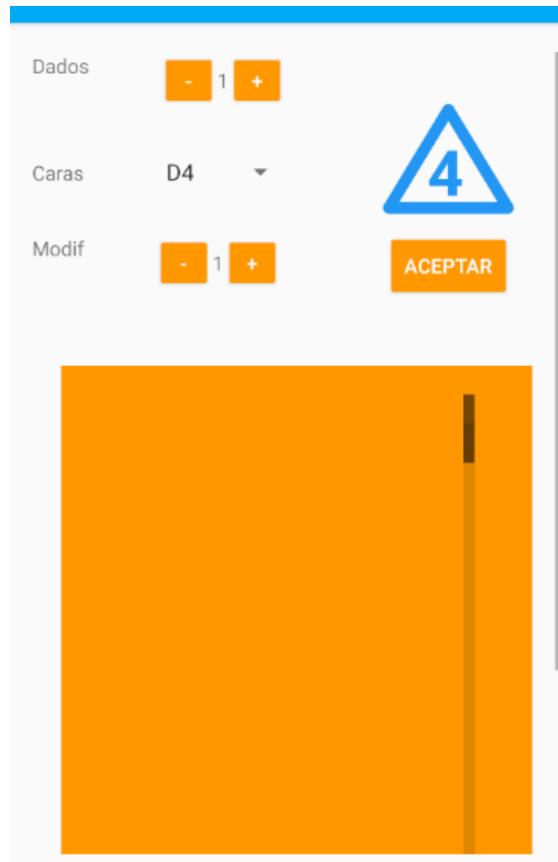
## Dados

Sirve para hacer tiradas de dados.

En **dados** podremos aumentar la cantidad de dados.

En **caras** indicaremos el tipo de dado que vamos a lanzar

En **modif** podremos agregar un modificador.



Al pulsar en aceptar nos aparecerá un dialogo mostrándonos el resultado de nuestra tirada.

En la parte de arriba **1d4+1** nos indica lo que hemos seleccionado. En este caso hemos tirado un dado d4 al que le hemos añadido un modificado +1.

Abajo nos muestra la tirada de ese dado, es decir el dado d4 ha sacado un 1.

Y para finalizar en el centro del dialogo aparece la suma de la tirada y el modificador es decir en este caso un 2.

Tirada normal

1d4+1

2

1



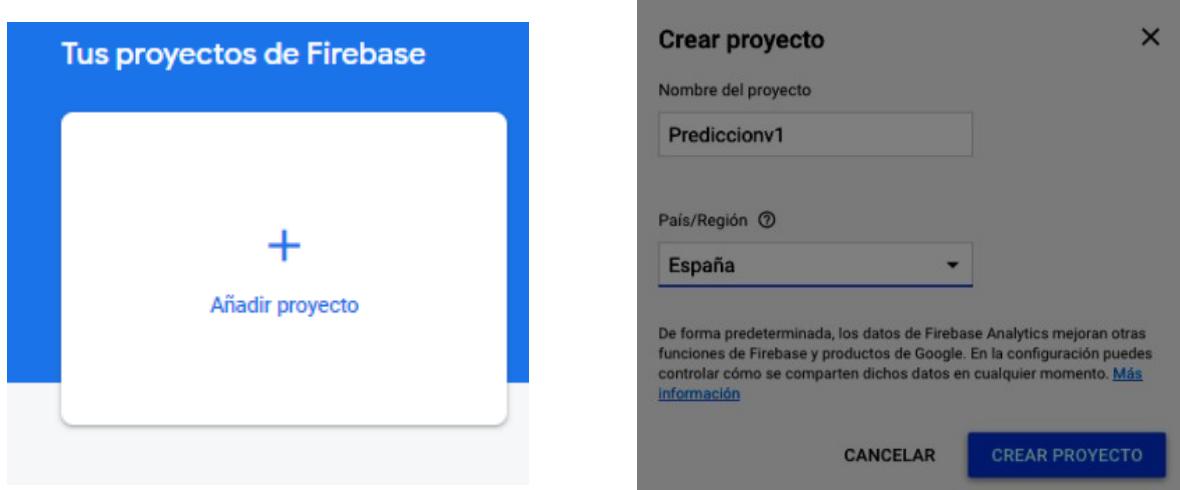
## Instalación

### Para la realización de este proyecto

Para realizar este proyecto hemos trabajado con Android Studio y Firebase. En primer lugar nos hemos creado un proyecto de Android y luego hemos hecho los siguientes pasos.

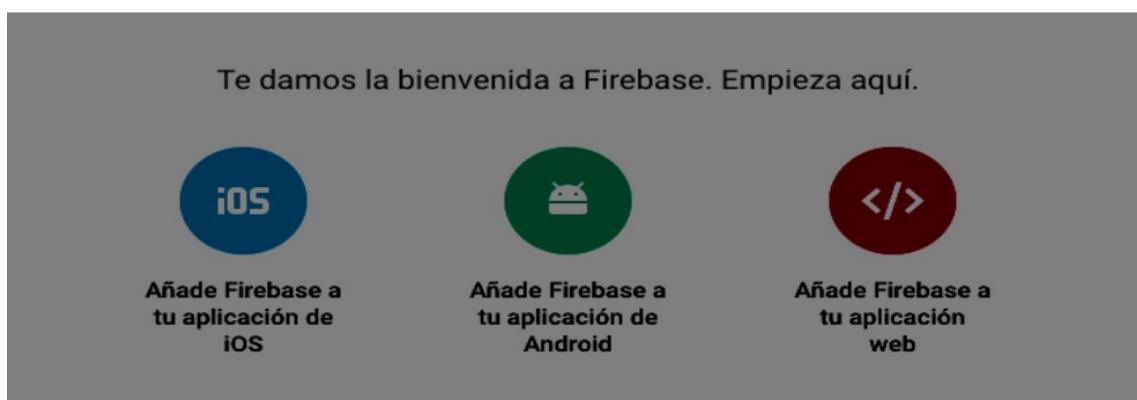
### Creación de un proyecto en firebase

Nos hemos registrado previamente en firebase y a continuación hemos agregado un proyecto. Le hemos proporcionado un nombre y un lugar de origen.

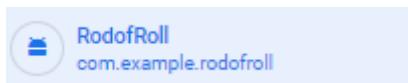


The image contains two screenshots. The left screenshot shows a blue header bar with the text 'Tus proyectos de Firebase' and a white card below it with a large blue plus sign and the text 'Añadir proyecto'. The right screenshot shows a modal window titled 'Crear proyecto' with fields for 'Nombre del proyecto' (containing 'Prediccionv1') and 'País/Región' (containing 'España'). Below the form is a note about Firebase Analytics. At the bottom are 'CANCELAR' and 'CREAR PROYECTO' buttons.

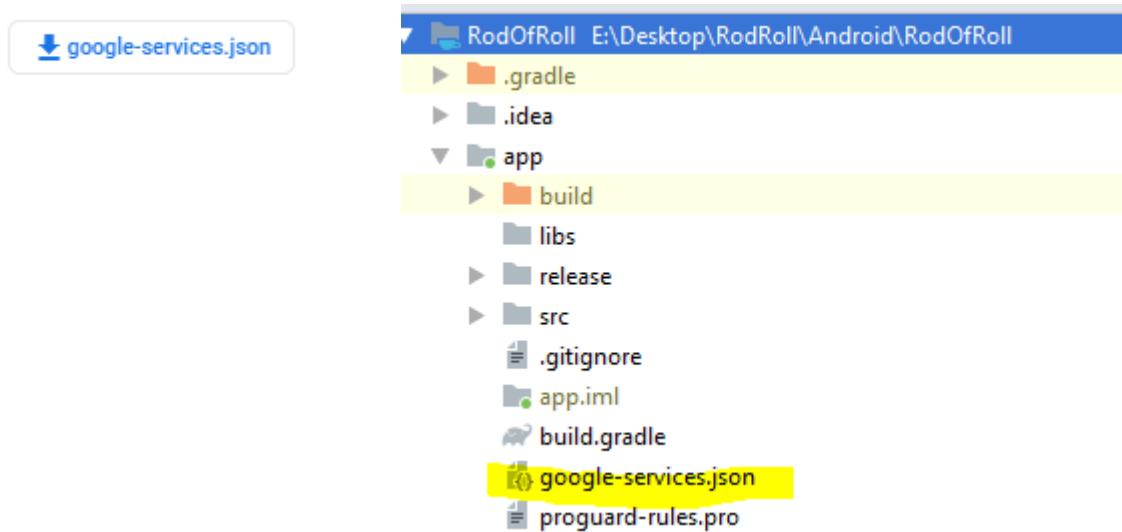
A continuación nos preguntará si queremos añadir Firebase a nuestra aplicación.



Introduciremos el nombre de dominio de nuestra aplicación.



Cuando se termina de crear la aplicación descargaremos un archivo JSON que tendremos que instalar en nuestro proyecto.



Seguiremos las instrucciones y añadiremos las dependencias en el build.gradle de la app y del proyecto.

En el proyecto

```
,  
dependencies {  
    classpath 'com.android.tools.build:gradle:3.5.3'  
    classpath 'com.google.gms:google-services:4.3.2'
```

## En la App

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-ct

    //noinspection GradleCompatible
    implementation 'com.android.support:design:28.0.0'

    implementation 'com.google.firebaseio:firebase-analytics:17.2.
    implementation 'com.google.firebaseio:firebase-auth:19.2.0'
    implementation 'com.google.firebaseio:firebase-database:19.2.0
    implementation 'com.firebaseioui:firebase-ui-database:1.1.1'
    implementation 'com.google.firebaseio:firebase-messaging:20.1.
    implementation 'androidx.gridlayout:gridlayout:1.0.0'
    implementation 'com.google.code.gson:gson:2.8.5'
    implementation 'com.airbnb.android:lottie:3.1.0'

    implementation 'com.firebaseioui:firebase-ui-auth:4.3.1'
    implementation 'com.google.firebaseio:firebase-storage:19.1.0'

}

apply plugin: 'com.google.gms.google-services'
```

Como podemos ver tenemos más recursos de Firebase. Estos se han ido instalando para la autorización de usuarios y para la gestión de las notificaciones de Firebase.

## Firebase Functions y como he trabajado con ellas

Para utilizar las funciones de Firebase nos hemos tenido que descargar Node.js

The screenshot shows the official Node.js download page. At the top, there's a navigation bar with links for INICIO, ACERCA, DESCARGAS, DOCUMENTACIÓN, PARTICIPE, SEGURIDAD, and NOTICIAS. Below the navigation is a section titled "Descargas" with a note that the current version is 12.16.3 (including npm 6.14.4). A sub-section titled "Actual" (Current) with "Últimas características" (Last features) is shown. Three download options are presented: "Instalador Windows" (Windows Installer), "Instalador macOS" (macOS Installer), and "Código Fuente" (Source Code). The source code option has a file size of "node-v12.16.3.tar.gz". Below this, a file manager interface shows a file named "node-v12.14.0-x64" with a size of "19.052 KB" and a modified date of "31/12/2019 6:52".

junto con el gestor de paquetes npm. Usando el terminal de comandos procederemos a instalar npm.

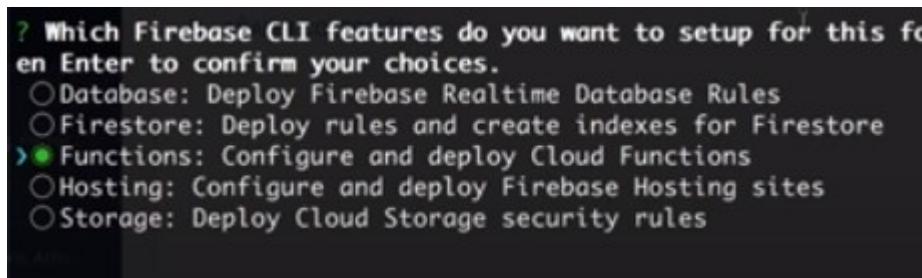
**npm install -g firebase-tools**

Seguidamente tendremos que inicializar nuestro proyecto en una carpeta vacía que hayamos creado. Una vez creada esta nos situaremos con el terminal sobre ella y realizaremos los siguientes comandos.

**firebase login** para loguearnos con nuestra cuenta de firebase.

Pincharemos en el enlace que nos mostrara el terminal y procedemos a acceder a nuestra cuenta de firebase.

**firebase init** para inicializar el proyecto. Elegiremos la configuración de Functions.



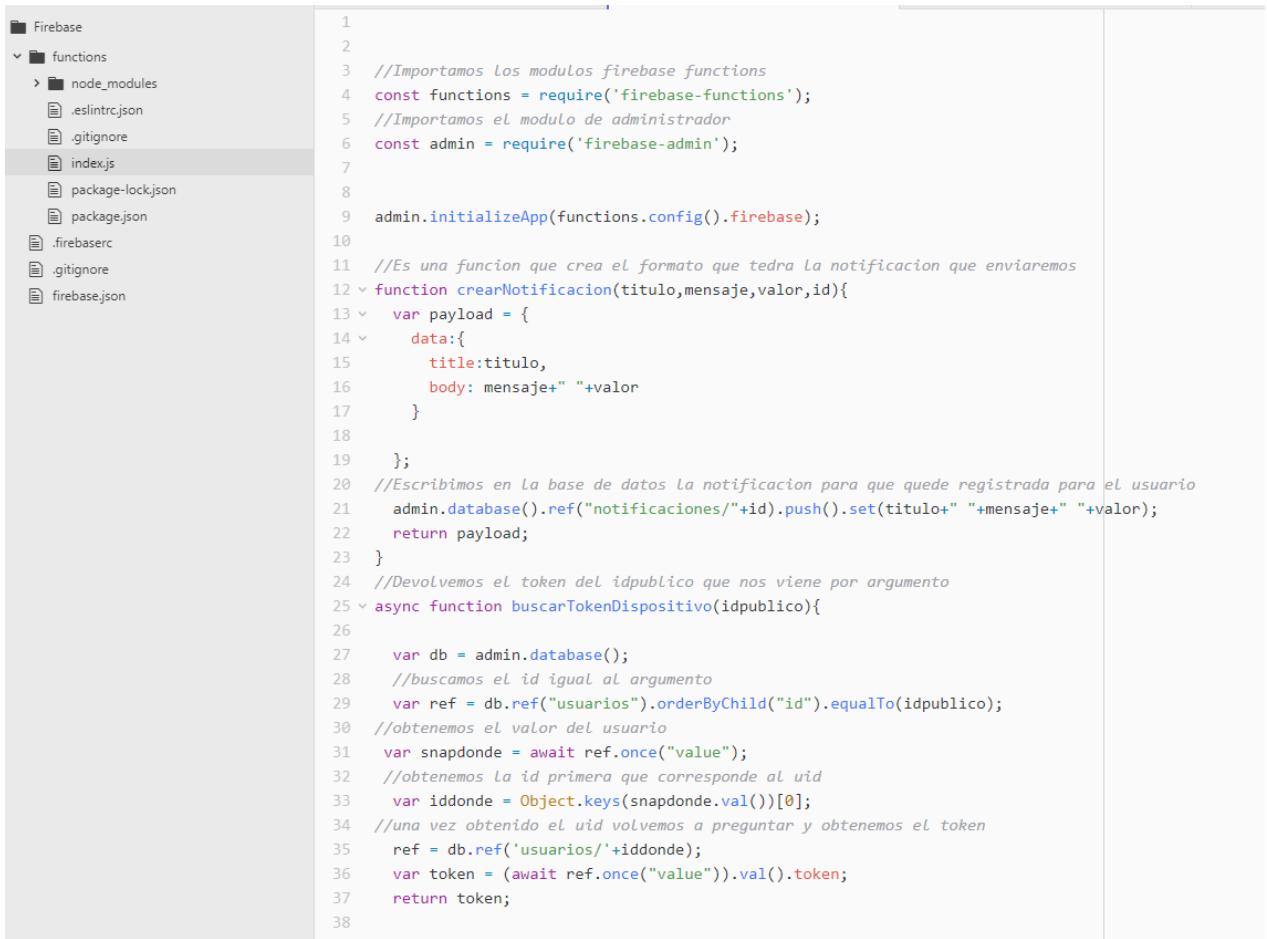
A continuación tendremos que elegir nuestro proyecto y seguidamente tendremos que elegir el tipo de lenguaje o TypeScript o Javascript. Elegiremos el último.

Finalmente nos preguntará si queremos usar TSLint, esto sirve para detectar posibles errores en nuestro código, pulsaremos que sí.

Al terminar tendremos la siguiente estructura de carpetas.

Este equipo > Disco local (C:) > Firebase				
Nombre	Fecha de modificación	Tipo	Tamaño	Buscar en Fireb
functions	26/02/2020 0:55	Carpeta de archivos		
.firebaserc	31/12/2019 7:47	Archivo FIREBASE...	1 KB	
.gitignore	31/12/2019 7:47	Documento de te...	2 KB	
firebase	31/12/2019 7:47	Archivo JSON	1 KB	

Yo he utilizado Atom para editar mis funciones y para abrir esta carpeta.



The screenshot shows the Atom code editor interface. On the left, the file tree displays the project structure: Firebase, functions, node\_modules, .eslintrc.json, .gitignore, index.js (which is selected), package-lock.json, package.json, .firebaserc, .gitignore, and firebase.json. The main pane shows the content of the index.js file:

```
1
2
3 //Importamos los modulos firebase functions
4 const functions = require('firebase-functions');
5 //Importamos el modulo de administrador
6 const admin = require('firebase-admin');
7
8
9 admin.initializeApp(functions.config().firebase);
10
11 //Es una funcion que crea el formato que tendrá la notificación que enviaremos
12 function crearNotificación(título,mensaje,valor,id){
13     var payload = {
14         data:{
15             title:título,
16             body: mensaje+" "+valor
17         }
18     };
19     //Escribimos en la base de datos la notificación para que quede registrada para el usuario
20     admin.database().ref("notificaciones/"+id).push().set(título+" "+mensaje+" "+valor);
21     return payload;
22 }
23
24 //Devolvemos el token del idpublico que nos viene por argumento
25 async function buscarTokenDispositivo(idpublico){
26
27     var db = admin.database();
28     //buscamos el id igual al argumento
29     var ref = db.ref("usuarios").orderByChild("id").equalTo(idpublico);
30     //obtenemos el valor del usuario
31     var snapdonde = await ref.once("value");
32     //obtenemos la id primera que corresponde al uid
33     var iddonde = Object.keys(snapdonde.val())[0];
34     //una vez obtenido el uid volvemos a preguntar y obtenemos el token
35     ref = db.ref('usuarios/'+iddonde);
36     var token = (await ref.once("value")).val().token;
37     return token;
38 }
```

## Para los profesores

A los profesores os he pasado el apk de la aplicación. Podéis probarlo con dispositivos con la versión de sdk 21 o superior.

**minSdkVersion 21**

Android10	10	API nivel 29
Pie	9	API nivel 28
Oreo	8.1.0	API nivel 27
Oreo	8.0.0	API nivel 26
Nougat	7.1	API nivel 25
Nougat	7.0	API nivel 24
Marshmallow	6.0	API nivel 23
Lollipop	5.1	API nivel 22
Lollipop	5.0	API nivel 21

También os pasare la carpeta de funciones de Firebase. Las podréis encontrar en la ruta [.\Firebase\functions\index.js](#) .

Os daré permisos de lectura a mi proyecto. Aquí encontrareis las funciones implementadas. Además veréis los datos de mi base de datos

The screenshot shows the Firebase Functions console interface. On the left is a sidebar with various icons for managing projects, regions, and logs. The main area has a header 'Functions' with tabs for 'Panel', 'Estado', 'Registros', and 'Uso'. Below this is a table listing eight functions:

Función	Activador	Región	Tiempo de ejecución	Memoria	Tiempo de espera
AvistarPersonaje	ref.write bases.firebaseio.com/usuarios/{uid}/combates/{ordenId}/personajes/{id}/avisar	us-central1	Node.js 8	256 MB	60s
DeleteCombate	ref.delete bases.firebaseio.com/usuarios/{uid}/combates/{ordenId}	us-central1	Node.js 8	256 MB	60s
DeleteCombateDesdePersonaje	ref.delete bases.firebaseio.com/usuarios/{uid}/personajes/{id}/combates/{ordenId}	us-central1	Node.js 8	256 MB	60s
DeletePersonaje	ref.delete bases.firebaseio.com/usuarios/{uid}/personajes/{id}	us-central1	Node.js 8	256 MB	60s
DeletePersonajeCombate	ref.delete bases.firebaseio.com/usuarios/{uid}/combates/{ordenId}/personajes/{id}	us-central1	Node.js 8	256 MB	60s
DeleteUsuarios	ref.delete bases.firebaseio.com/usuarios/{uid}	us-central1	Node.js 8	256 MB	60s
NuevoCombate	ref.create bases.firebaseio.com/usuarios/{uid}/combates/{ordenId}	us-central1	Node.js 8	256 MB	60s
NuevoCombatiante	ref.create bases.firebaseio.com/usuarios/{uid}/combates/{ordenId}/personajes/{id}	us-central1	Node.js 8	256 MB	60s

Finalmente os enviaré la carpeta con el código de mi aplicación. Deberéis abrirlo en AndroidStudio.

## Conclusiones

Cuando se me planteo trabajar con Firebase, pensé que sería todo un desafío. En clase habíamos trabajado más con otras bases de datos y no tenía nada claro como enfocar mi proyecto. A pesar de todo he de reconocer que Firebase es una herramienta muy potente que me ha permitido mandar notificaciones a distintos dispositivos sin problema, además de poder tener una base de datos a tiempo real.

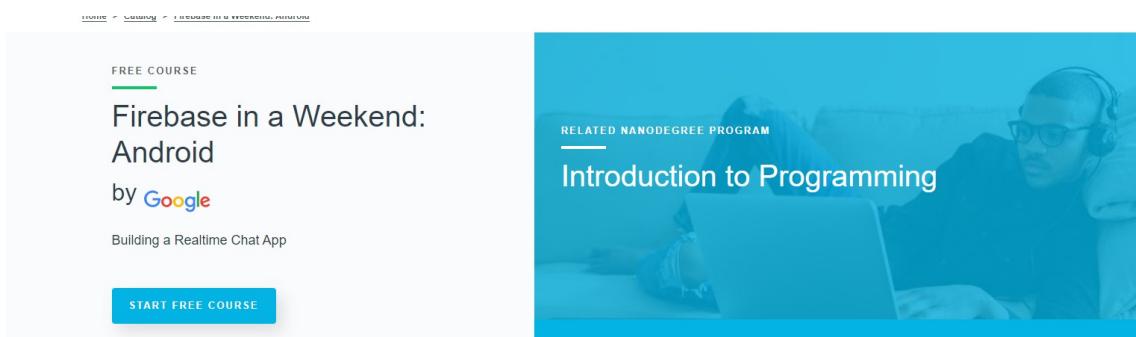
En cuanto a mi trabajo con AndroidStudio, tuve varios problemas al principio, y gracias a estos he aprendido mucho sobre el funcionamiento de Android. Por suerte nunca he tenido problemas con la conocida R de Android, así que muchos de los errores que he tenido han sido conceptuales.

Para finalizar quería concluir diciendo que mi proyecto tiene mucho por mejorar, pero de cierto modo me siento satisfecha con el trabajo realizado. Si tuviera que incluir algo más sería un chat para los jugadores, de esta manera me podría enfrentar al reto que suponen los sockets en Android.

## Bibliografía

Aconsejo a cualquiera que vaya a trabajar con Firebase que haga este curso gratuito de Udacity, recomendado además por el equipo de Firebase.

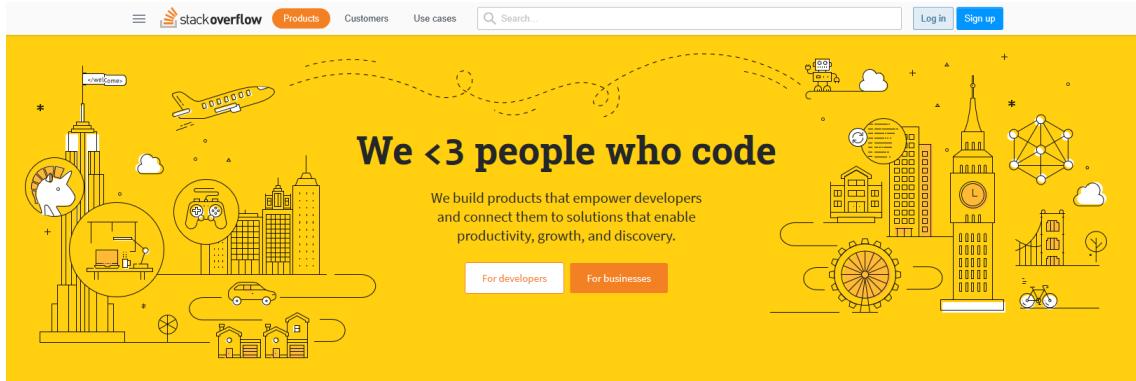
<https://www.udacity.com/course/firebase-in-a-weekend-by-google-android--ud0352>



Es un curso que me ha ayudado mucho a la comprensión del funcionamiento de esta herramienta, además después de cada video hay un pequeño test. Es bastante sencillo y ameno.

También he hecho uso de la conocida página StackOverflow

<https://stackoverflow.com/>



### For developers, by developers

Stack Overflow is an [open community](#) for anyone that codes. We help you get answers to your toughest coding questions, share knowledge with your coworkers in private, and find your next dream job.