




# MARVEJUEGOS



Rodrigo Martin Vegas 50566054W  
2º DAM IES LAS SALINAS Anahí Mula de la Banda

# ÍNDICE

## INDEX

RESUMEN / ABSTRACT .....	2
INTRODUCCIÓN / INTRODUCTION.....	3
EVOLUCIÓN DE LOS SMARTPHONES.....	3
SMARTPHONES' EVOLUTION .....	4
JUSTIFICACIÓN DEL PROYECTO / PROJECT JUSTIFICATION .....	6
OBJETIVOS / OBJECTIVES.....	7
ANÁLISIS DEL PROYECTO Y EMPRESA / PROYECT ANALYSIS .....	7
DIAGRAMAS / DIAGRAMS .....	9
DESARROLLO / DEVELOPMENT .....	11
DESARROLLO DE INTERFACES / INTERFACE DEVELOPMENT.....	30
MANUALES / MANUALS .....	31
MANUAL DE INSTALACIÓN / INSTALLATION MANUAL .....	31
MANUAL DE USUARIO / USER MANUAL .....	31
MANUAL DE ADMINISTRADOR / ADMIN MANUAL .....	31
CONCLUSIONES / CONCLUSIONS.....	32
BIBLIOGRAFÍA / BIBLIOGRAPHY.....	32

## **RESUMEN / ABSTRACT**

El mercado de los juegos en Android ha sufrido muchos cambios a lo largo de los últimos veinte años, desde míticas aplicaciones como el "Snake" popularizado por Nokia, hasta portabilidades de juegos de PC, como el famoso título de Epic Games, Fortnite, o el recién salido del horno Wild Rift, creado por la empresa Riot Games, autores de League of Legends.

Los grandes olvidados de este sector a lo largo de los años empiezan a ser los mismos que comenzaron el propio trabajo: los minijuegos. Estos minijuegos han sido abandonados por los jugadores casuales, que siempre buscan algo más comercial de cara al uso diario, o algo que para ellos sea más "adictivo".

El objetivo de este trabajo es dar un nuevo enfoque a las aplicaciones de minijuegos en Android, a través de una interfaz más sencilla y pulida, para conseguir atraer a los "gamers" del sector de los smartphones, los casuales, con juegos más simples y menos consumidores.

The Android gaming market has undergone many changes over the past twenty years, from mythical applications such as the "Snake" popularized by Nokia, to PC game portabilities, such as the famous Epic Games title, Fortnite, or the fresh out of oven, Wild Rift, created by the company Riot Games, authors of League of Legends.

The great forgotten on this sector over the years are the same ones that started the job itself: mini-games. These mini-games have been abandoned by casual gamers, who are always looking for something more commercial for everyday use, or something that is more "addictive" for them.

The objective of this work is to give a new approach to the mini-game applications on Android, through a simpler and more polished interface, to be able to catch more nowadays casual players with simpler and less consumer games.

# **INTRODUCCIÓN / INTRODUCTION**

## **EVOLUCIÓN DE LOS SMARTPHONES**

En España, a diferencia del resto del mundo, el auge de los teléfonos móviles vino de la mano de la primera oleada de operadoras durante la segunda mitad de los 90. Desde entonces y hasta hace muy poco, las principales compañías han ofertado los últimos modelos de telefonía móvil a cambio de seguir un contrato, y eso nos ha malacostumbrado a estar a la vanguardia en los terminales.

Sin embargo, con la llegada del 3G se abrió la veda para que pudiéramos descargarnos una infinidad de juegos desde los propios servicios móviles, e incluso con códigos publicitados en prensa y televisión. Ya era algo oficial, queríamos que nuestros móviles fueran máquinas potentes capaces de poder ejecutar los juegos más poderosos.

Los juegos móviles han presentado una gran evolución desde sus inicios y en cierto modo han contribuido a impulsar las nuevas tecnologías. Los smartphones han sido los más beneficiados al darse la posibilidad de llegar de una forma implacable al público.

Durante esta evolución, vimos como los videojuegos evolucionaban no sólo en su aspecto, sino en su jugabilidad y sistema de juego. Lo que en principio era un pequeño entretenimiento, ha resultado ser un gran negocio y muchas grandes compañías se han subido al carro de los juegos móviles.

Sin lugar a duda, la primera causa que posibilitó la evolución de los juegos para móviles, ha sido la mejora del rendimiento de los smartphones. Hemos pasado de sistemas con recursos limitados, a smartphones con especificaciones técnicas de alto rendimiento capaces de mover juegos en tres dimensiones. La mejora de la potencia de los móviles puso al alcance de desarrolladores y usuarios la posibilidad de aprovechar toda la potencia del móvil.

Otra de las causas fundamentales que provocó el boom de los juegos móviles se inició en las redes sociales, como Twitter e Instagram. Ya no hacía falta tener un PC cerca para jugar, desde nuestro propio móvil teníamos acceso a nuestros juegos favoritos. La conexión permanente a internet con la llegada de las tarifas de datos, se abrió la veda de los juegos online en los que se compite con gente conocida o a través de las redes sociales.

Los smartphones y las tablets se empiezan a comer al resto de dispositivos de entretenimiento, mientras la explosión de estudios indies ensanchan las fórmulas jugables en cuanto a nuestros dispositivos portables, las grandes desarrolladoras de juegos como Capcom, Electronic Arts, o Activision-Blizzard apuestan drásticamente por el formato que les es más rentable con sus ambiciosos recursos técnicos.

Para poder compatibilizar nuestros terminales con todas estas nuevas oleadas de juegos, necesitaremos procesadores capaces de responder a las pantallas de alta tecnología y con un bajo consumo energético. Estos procesadores ya están saliendo a la luz, como es el ejemplo de los modelos Helio X20 de MediaTek, que ofrece tres núcleos de procesamiento, lo cual permitiría el desarrollo de actividades como mandar un Whatsapp o leer Twitter tras echar una partida al Fortnite, apagando los núcleos innecesarios para prologar la duración de la batería, algo que al comprador siempre le preocupa.

Frente a las grandes empresas competidoras, a lo largo de los años se han mantenido en la industria del juego móvil mínimamente a flote los minijuegos, apps o webs más simples que otras y que pueden proporcionar la misma diversión a los usuarios. Y eso es lo que busca Marvejuegos, mantener en el mercado aquellos juegos que a nosotros nos parecen interesantes para el público, ya sean jóvenes o mayores.

### **SMARTPHONES' EVOLUTION**

In Spain, unlike the rest of the world, the boom in smartphones came from the first wave of operators during the second half of the 90s. Since then and until very recently, the main companies have offered the latest models of mobile telephony in exchange for following a contract, and that has made us unusually at the forefront of terminals.

However, with the arrival of 3G a whole new world was opened so that we could download an infinity of games from the mobile services themselves, and even with codes published in the press and television. It was already something official, we wanted our phones to be powerful machines capable of running the most powerful games.

Mobile games have undergone a great evolution since their inception and in a way have contributed to promoting new technologies. Smartphones have been the most benefited by giving themselves the chance to reach the public in a relentless way.

During this evolution, we saw how video games evolved not only in their appearance, but in their gameplay and game system. What was initially small entertainment has turned out to be big business, and many big companies have jumped on the mobile gaming bandwagon.

Undoubtedly, the first cause that enabled the evolution of mobile games has been the improvement in the performance of smartphones. We have gone from systems with limited resources, to smartphones with high-performance technical specifications capable of moving games in three dimensions. The improvement of the power of the mobiles made available to developers and users the possibility of taking advantage of all the power of the mobile.

Another of the fundamental causes that caused the boom in mobile games began on social networks, such as Twitter and Instagram. It was no longer necessary to have a PC nearby to play, from our own mobile we had access to our favorite games. The permanent connection to the internet with the arrival of data rates, opened the ban on online games in which you compete with well-known people or through social networks.

Smartphones and tablets are beginning to eat the rest of entertainment devices, while the explosion of indie studios widens the playable formulas in terms of our portable devices, major game developers such as Capcom, Electronic Arts, or Activision-Blizzard bet drastically for the format that is most profitable with its ambitious technical resources.

In order to make our terminals compatible with all these new waves of games, we will need processors capable of responding to high-tech screens and with low energy consumption. These processors are already coming to light, as is the example of MediaTek's Helio X20 models, which offers three processing cores, which would allow the development of activities such as sending a WhatsApp or reading Twitter after playing a game on Fortnite, turning off unnecessary cores to extend battery life, something the buyer is always concerned about.

Compared to the large competing companies, over the years the mini-games, apps or websites have remained minimally afloat in the mobile gaming industry, which are simpler than others and that can provide the same fun to users. And that's what Marvejuegos is all about, keeping an eye on the games we think are interesting for the audience, those being younglings or older people.

## **JUSTIFICACIÓN DEL PROYECTO / PROJECT JUSTIFICATION**

Como bien se ha dicho en el Abstract, el motivo de la realización de este proyecto es atraer a los jugadores casuales de móvil, los cuales están atrapados en juegos más mainstream de empresas mucho más grandes, lo cual no da cabida en el mercado a empresas más pequeñas que luchan por sacar adelante juegos más fáciles. Lo que se busca es darle al usuario una experiencia de juego y de uso mucho más fácil, con juegos simples y divertidos al alcance de unos toques en su pantalla.

Se busca poder almacenar sus datos más básicos, sin pedirles confirmaciones no necesarias, como las típicas de número de teléfono o la propia confirmación de correo. Todo aquel que quiera entrar en la app simplemente necesitara introducir un email (el cual no se verificará), un nombre de usuario y una contraseña, que se almacenaran de forma segura en nuestras bases de datos.

As stated in the Abstract, the reason for carrying out this project is to attract casual mobile gamers, who are caught up in more mainstream games from much larger companies, which gives smaller companies a struggle to make easier games work. We look after giving the user a much easier gaming and user experience, with simple and fun games within making a few taps on your screen.

The aim is to be able to store your most basic data, without asking for unnecessary confirmations, such as the typical telephone number or the email confirmation itself. Anyone who wants to enter the app simply needs to enter an email (which will not be verified), a username and password, which will be stored securely in our databases.

## **OBJETIVOS / OBJECTIVES**

Simplificar y cambiar la manera en la que la gente puede jugar a juegos fáciles.

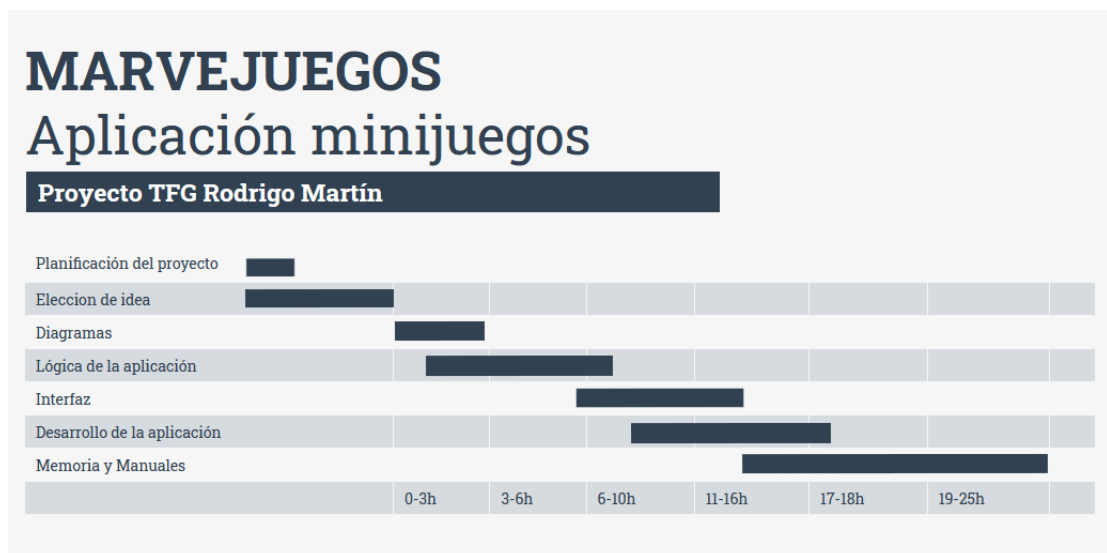
Presentar minijuegos que pueden ser divertidos a un público que los desconoce.

Simplify and change the way people can play easy games.

Present minigames that can be fun to an audience that doesn't know them.

## **ANÁLISIS DEL PROYECTO Y EMPRESA / PROJECT ANALYSIS**

Nuestro diagrama de Gantt quedaría de la siguiente forma:



*Imagen 1. Diagrama de Gantt 1*

Los requisitos mínimos que se necesitan para poder utilizar nuestra aplicación son los siguientes: el móvil debe tener sistema operativo Android 4.0.3 (Ice Cream Sandwich) o superior para el correcto funcionamiento de la aplicación, una pantalla de 5" o superior y conexión a Internet/Wi-Fi, ya que algunos de nuestros juegos se redirigen hacia una web.

El presupuesto de este proyecto estaría alrededor de unos 5000 euros, dividido en lo siguiente: dos móviles de prueba, de unos 600-800€ cada uno; tres ordenadores, dos de sobremesa y otro portátil, con un precio de 1200€ cada uno; la licencia de Google Play, la cual supone unos 25\$ (unos 22€) y un servidor HP ProLiant AMD Opteron, con un precio de 365€.

Todos los ordenadores llevarán Windows 10 instalado y Android Studio, al igual que los móviles llevarán Android 10 como versión predeterminada para probar cada aplicación que



se desarrolle en la actividad normal de la empresa. La seguridad de los ordenadores requerirá de un usuario y una contraseña, vinculados ambos a un email, al igual que los propios usuarios de la aplicación. La oficina en la cual se encuentra el servidor que mantiene activa la base de datos de la aplicación se mantendrá cerrada con llave siempre que no haya nadie trabajando en ese momento allí.

Para la gestión de todos estos gastos y conexiones necesarias con proveedores se ha utilizado Odoo, un ERP que nos permite una fácil comunicación y uso de herramientas necesarias para automatizar y mantener de forma sencilla toda la infraestructura de la empresa. Dentro de esta misma herramienta gestionamos todos los proyectos que se realizan en la empresa, gracias a su herramienta de análisis de trabajo.

The minimum requirements needed to be able to use our application are the following: the mobile must have an Android 4.0.3 (Ice Cream Sandwich) or higher operating system for the correct functioning of the application, a 5 " screen or higher and connection to Internet / Wi-Fi, since some of our games are redirected to a website.

The budget for this project would be around 5,000€, divided into the following: two test mobiles, of around 600-800 € each; three computers, two desktop and one laptop, priced at 1,200€ each; the Google Play license, which is about 25\$ (about 22€) and an HP ProLiant AMD Opteron server, with a price of 365€.

All computers will have Windows 10 installed and Android Studio, just as mobiles will have Android 10 as the default version to test each application that is developed in the everyday activity of the company. Computer security will require a username and password, both linked to an email, just like the users of the application. The office where the server that maintains the application database is located will be kept locked as long as no one is currently working there.

For the management of all these expenses and necessary connections with suppliers, Odoo has been used, an ERP that allows us to easily communicate and use the necessary tools to easily automate and maintain the entire infrastructure of the company. Dentro de esta misma herramienta gestionamos todos los proyectos que se realizan en la empresa, gracias a su herramienta de análisis de trabajo.

# DIAGRAMAS / DIAGRAMS

## Diagrama E-R

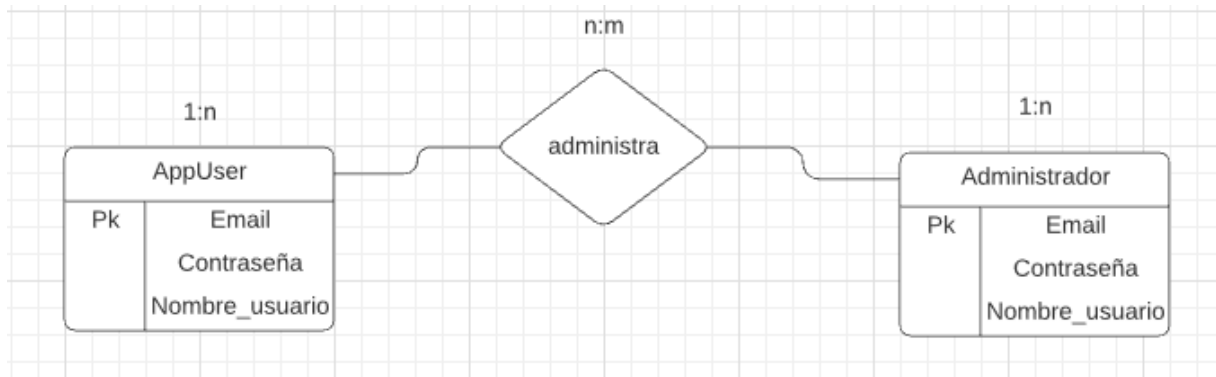


Imagen 2. Diagrama 1

## Diagrama de clases

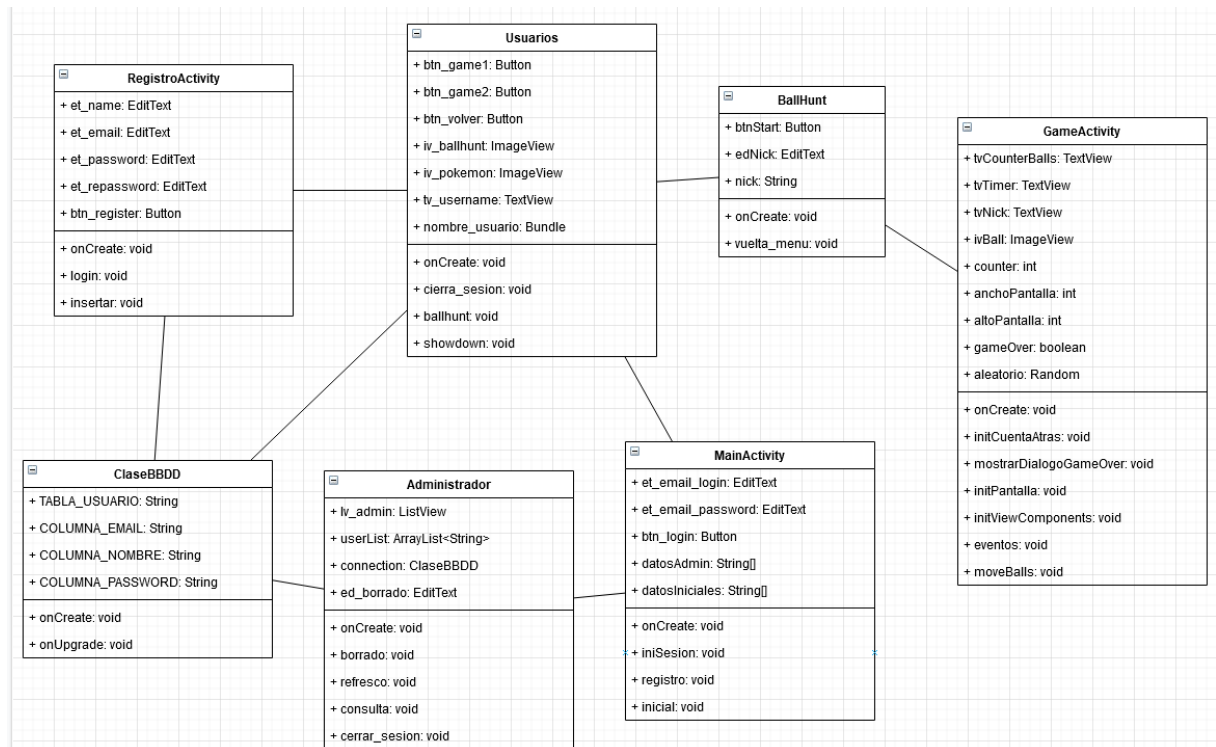


Imagen 2. Diagrama 2

## Diagramas de casos de uso

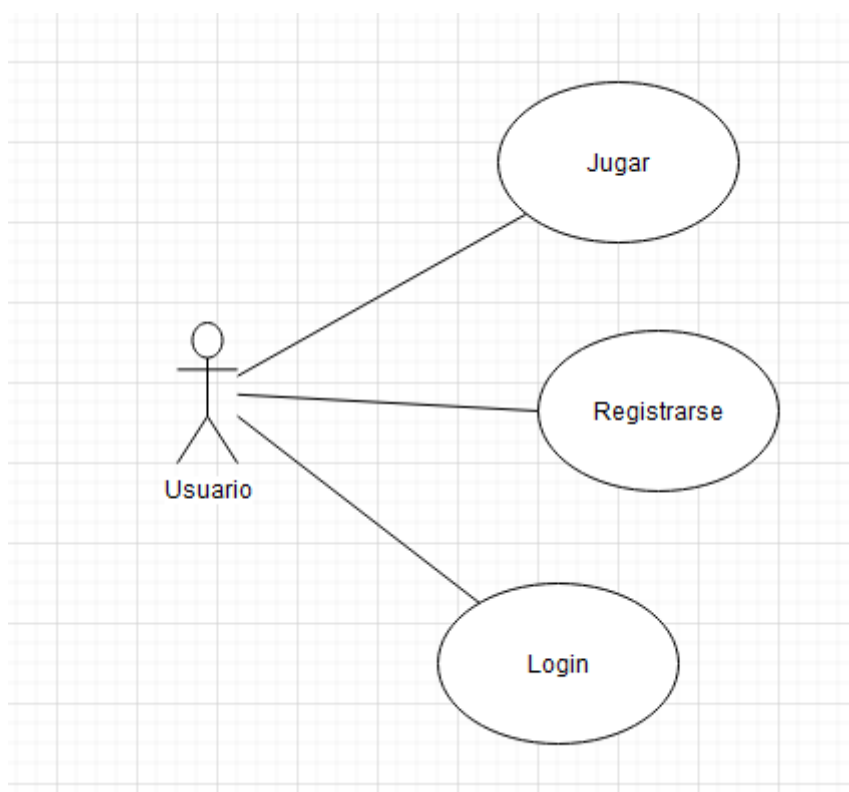


Imagen 2. Diagrama 3

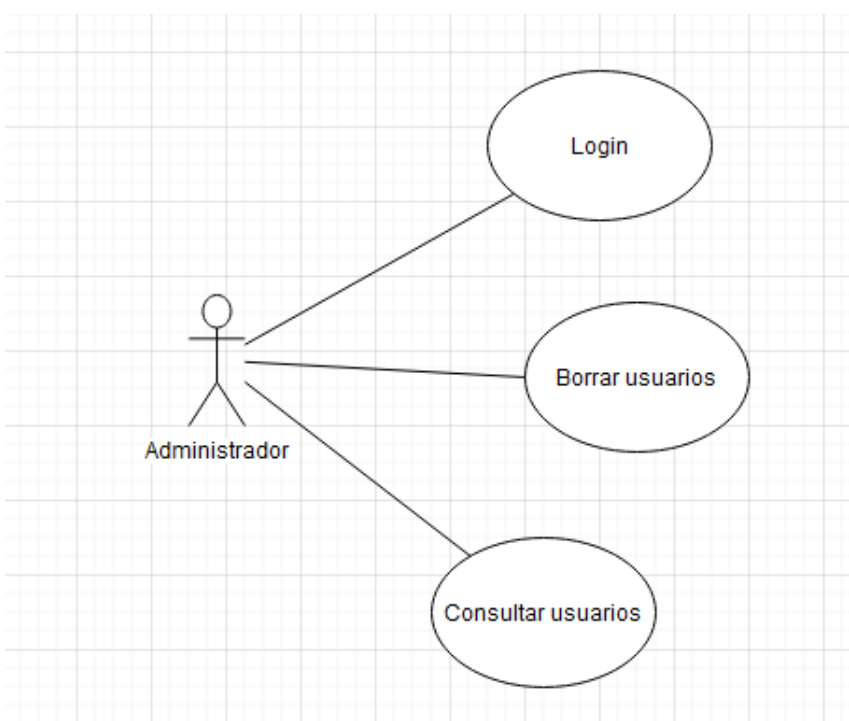


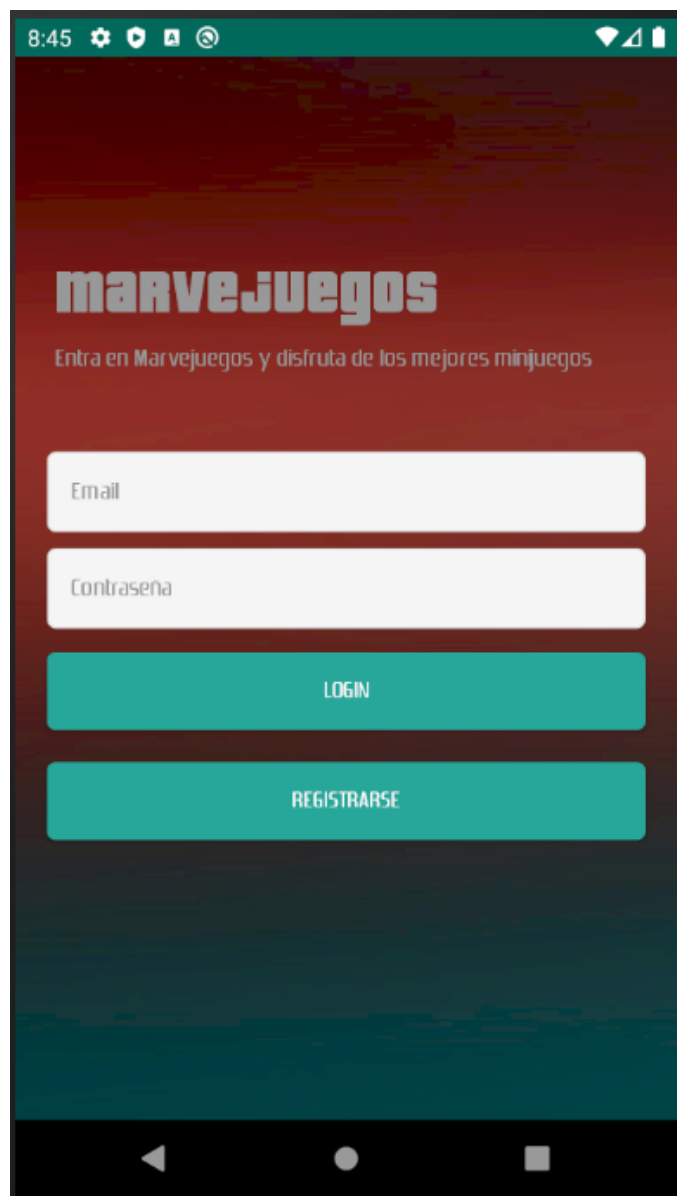
Imagen 2. Diagrama 4

## DESARROLLO / DEVELOPMENT

Hemos utilizado el IDE Android Studio como herramienta principal de desarrollo de la aplicación.

Lo primero que se ha realizado es la creación de la pantalla de Login, la cual está compuesta por 2 textViews, que recogen el email y la contraseña del usuario, y 2 botones que se utilizan para realizar la acción de Login y para pasar a la pantalla de Registro.

The first thing that has been done is the creation of the Login screen, which is composed of 2 textViews, which collect the user's email and password, and 2 buttons that are used to perform the Login action and to go to the Registration screen.



*Imagen 3. Desarrollo 1*

Clase que recoge los campos de texto, botones y la inicialización de una clase explicada más adelante.

Class that collects the text fields, buttons and the initialization of a class explained later.

```
public class MainActivity extends AppCompatActivity {
    EditText et_email_login, et_password_login;
    Button btn_login;
    String[] datosAdmin = {"romarvedam@gmail.com", "marvejuegos", "admin"};
    String[] datosIniciales = {"alvarobf98@gmail.com", "alvaropls", "1234"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        et_email_login = findViewById(R.id.et_email_login);
        et_password_login = findViewById(R.id.et_password_login);
        btn_login = findViewById(R.id.btn_login);

        inicial(datosAdmin);
        inicial(datosIniciales);
    }
}
```

Imagen 3. Desarrollo 2

Clase que recoge el Intent el cual nos permite cambiar a la ventana de usuario o, si eres administrador y cumples los requisitos del if (email.equals), a la ventana de administración. En la última condición se recoge en un putExtra () el nombre del usuario para después introducirla en un textView en la pantalla de Usuarios.

Class that includes the Intent which allows us to change to the user window or, if you are an administrator and meet the requirements of the condition if (email.equals), to the administration window. In the last condition, the user's name is collected in a putExtra () and then recovered in a textView on the Users screen.

```
public void iniSesion(View v){
    Intent inicio;
    ClaseBBDD admin = new ClaseBBDD( context: this, name: "users", factory: null, version: 1);
    SQLiteDatabase Database = admin.getWritableDatabase();
    String email = et_email_login.getText().toString();
    String contrasena = et_password_login.getText().toString();
    Cursor fila = Database.rawQuery( sql: "SELECT contrasena, nombre_usuario FROM users WHERE email = " + email + " ", selectionArgs: null);
    String nombreUsuario = "";
    String contraUsuario = "";
    if(fila.moveToFirst()) {
        contraUsuario = fila.getString( columnIndex: 0);
        nombreUsuario = fila.getString( columnIndex: 1);
    }
    if(email.equals("romarvedam@gmail.com") && contrasena.equals(contraUsuario)){
        inicio = new Intent( packageContext: this, Administrador.class);
        startActivity(inicio);
    }
    if(!email.equals("romarvedam@gmail.com") && !email.equals("") && contrasena.equals(contraUsuario)) {
        inicio = new Intent( packageContext: this, Usuarios.class);
        inicio.putExtra( name: "nombre_usuario", nombreUsuario); //pasarlo cada vez que se pida
        startActivity(inicio);
    }
}
```

Imagen 3. Desarrollo 3

Clase que nos permite cambiar a la pantalla del registro, en caso de que el usuario no tenga ninguna cuenta registrada en la aplicación.

Class that allows us to switch to the registration screen, in case the user does not have any account registered in the application.

```
public void registro(View v){  
    Intent intent_registro=new Intent( packageContext: this,RegistroActivity.class);  
    startActivity(intent_registro);  
}
```

*Imagen 3. Desarrollo 4*

Ésta clase es la que inicia los usuarios de administrador y un dummy para las pruebas, la cual escribe en la base de datos los datos que se han introducido en la clase de MainActivity ().

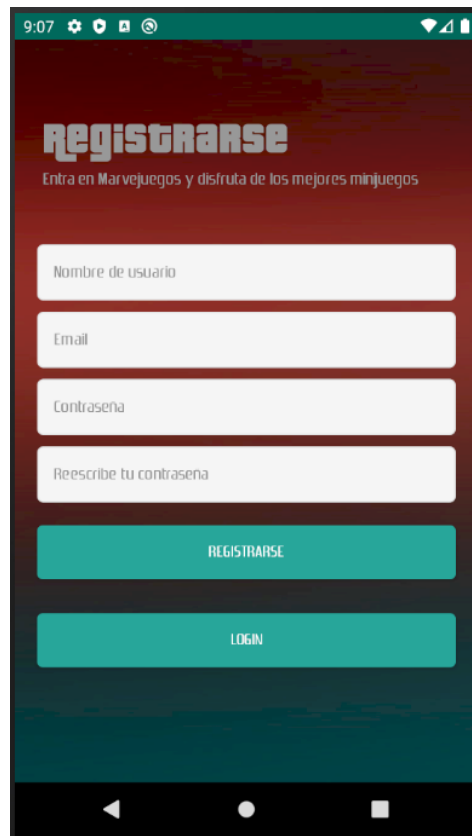
This class is the one that starts the administrator users and a dummy for the tests, which writes to the database the data that has been entered in the MainActivity () class.

```
public void inicial(String[]datos){  
    ClaseBBDD admin=new ClaseBBDD( context: this, name: "users", factory: null, version: 1);  
    SQLiteDatabase Database=admin.getWritableDatabase();  
    ContentValues contenido=new ContentValues();  
    contenido.put("email",datos[0]);  
    contenido.put("nombre_usuario",datos[1]);  
    contenido.put("contrasena",datos[2]);  
  
    Database.insert( table: "users", nullColumnHack: null,contenido);  
    Database.close();  
}
```

*Imagen 3. Desarrollo 5*

La pantalla de Registro contiene en este caso 4 textViews, que recogerán el email, nombre de usuario, contraseña, y confirmación de contraseña. Esta pantalla también contendrá 2 botones, uno que confirmará el registro y otro que te devolvería a la pantalla original de Login.

The Registration screen contains in this case 4 textViews, which will collect the email, username, password, and password confirmation. This screen will also contain 2 buttons, one that will confirm the registration and another that will return you to the original Login screen.



*Imagen 3. Desarrollo 6*

Clase que recoge los elementos de la pantalla de Registro.

Class that collects the elements of the Registration screen.

```
public class RegistroActivity extends AppCompatActivity {
    EditText et_name,et_email,et_password,et_repasword;
    Button btn_register;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro);

        et_name=findViewById(R.id.et_name);
        et_email=findViewById(R.id.et_email_login);
        et_password=findViewById(R.id.et_password_login);
        et_repasword=findViewById(R.id.et_repasword);
        btn_register=findViewById(R.id.btn_register);
    }
}
```

*Imagen 3. Desarrollo 7*

Clase que permite que el botón de Login de la pantalla de Registro () te lleve de vuelta al Login. Se ha introducido por si se da el caso de que el usuario ya tiene un correo registrado o que se equivoque al pulsar el botón en la pantalla original.

Class that allows the Login button on the Registration screen () to take you back to Login. It has been entered in case it happens that the user already has a registered email or that they make a mistake when pressing the button on the original screen.

```
public void login(View v){
    Intent intent=new Intent( packageContext: this,MainActivity.class);
    startActivity(intent);
}
```

*Imagen 3. Desarrollo 8*

Método que permite insertar los datos que introducen los usuarios en los textView dentro de la base de datos. Este método recoge los campos de texto de que se presentan en la pantalla de Registro, los almacena, y si la contraseña y la repetición de la misma coinciden, introduce los campos en la base de datos y manda al usuario a la pantalla de Usuarios.

Method that allows inserting the data entered by users in the textViews within the database. This method collects the text fields that are presented on the Registration screen, stores them, and if the password and the repetition of the same match, enter the fields in the database and send the user to the Users screen.

```
public void insertar(View view){
    ClaseBBDD admin=new ClaseBBDD( context: this, name: "users", factory: null, version: 1);
    SQLiteDatabase Database=admin.getWritableDatabase();
    String email=et_email.getText().toString();
    String nombre=et_name.getText().toString();
    String contraseña=et_password.getText().toString();
    String recontrasena=et_repasword.getText().toString();
    if (contrasena.equals(recontrasena)==true){
        ContentValues contenido=new ContentValues();
        contenido.put("email",email);
        contenido.put("nombre_usuario",nombre);
        contenido.put("contrasena",contrasena);

        Database.insert( table: "users", nullColumnHack: null,contenido);
        Database.close();
        Intent intent=new Intent( packageContext: this,Usuarios.class);
        startActivity(intent);
    }
}
```

*Imagen 3. Desarrollo 9*



Lo siguiente es la implementación de la base de datos de SQLite para que ésta nos registre cuándo se loguea un usuario o el registro de un usuario nuevo. Para ello hemos creado esta clase Java, la cual recoge los nombres de la tabla y los campos, y la consulta que crea la base de datos.

The following is the implementation of the SQLite database so that it records when a user logs in or the registration of a new user. For this we have created this Java class, which collects the names of the table and the fields, and the query that creates the database.

```
package com.example.marvejuegos;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class ClaseBBDD extends SQLiteOpenHelper {

    private static final String TABLA_USUARIOS = "users";

    // nombres de las columnas de la tabla
    private static final String COLUMNA_EMAIL = "email";
    private static final String COLUMNA_NOMBRE = "nombre_usuario";
    private static final String COLUMNA_PASSWORD = "contrasena";

    public ClaseBBDD(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // consulta de creacion de la tabla
        db.execSQL("CREATE TABLE " + TABLA_USUARIOS + "("
            + COLUMNA_EMAIL + " varchar(40) PRIMARY KEY ," + COLUMNA_NOMBRE + " varchar(15),"
            + COLUMNA_PASSWORD + " varchar(20)" + ")");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }

}
```

Imagen 3. Desarrollo 10

Desde la pantalla de Login tenemos dos opciones: el Registro, o el propio Login. Una vez hemos entrado con nuestro usuario, tenemos otras dos opciones: si el campo del email y la contraseña cumplen unas determinadas condiciones, te lleva a la pantalla de Administrador; mientras que si no las cumple, te lleva a la de Usuario.

Una vez dentro de la pantalla de Usuarios, se nos presentan 3 botones, con sus respectivas funcionalidades, y un textView que nos muestra el nombre del usuario que está logueado.

The Login screen gives us two options: the Registry, or the Login itself. Once we have entered with our user, we have another two options: if the email field and the password meet certain conditions, it takes you to the Administrator screen; while if it does not comply, it takes you to the User.

Once inside the Users screen, we can see 3 buttons, with their respective functionalities, and a textView that shows us the name of the user that is logged in.

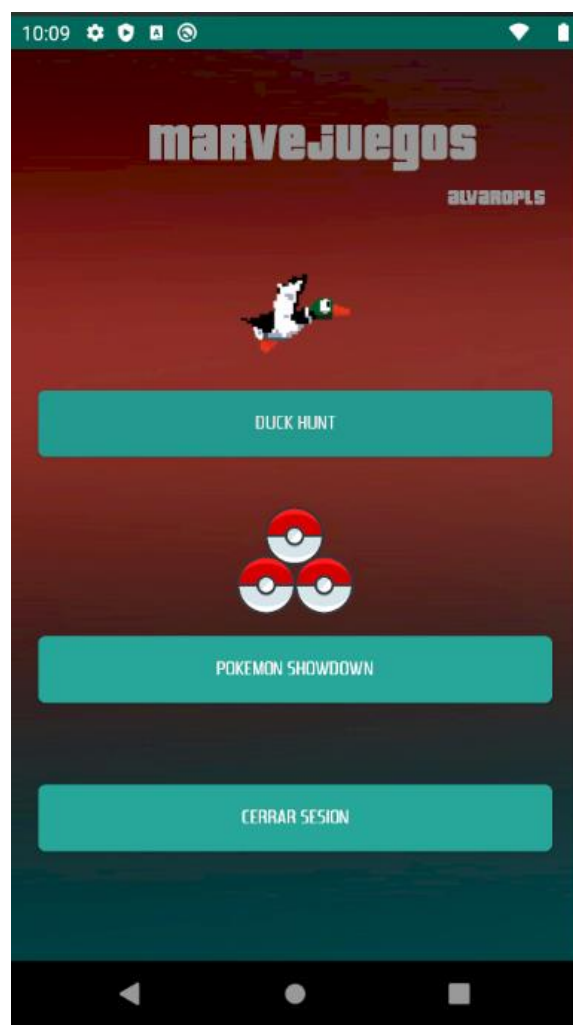


Imagen 3. Desarrollo 11

Como en el resto de las pantallas, el principio de la clase Java de los usuarios recoge todas las entidades que se encuentran en dicha pantalla.

Cabe destacar que en esta clase se añade la función de Bundle para recoger el nombre de usuario, almacenado en un putExtra (), y así poder mostrarlo en el textView del nombre.

As in the rest of the screens, the beginning of the users Java class includes all the entities that reside on that screen.

It should be noted that in this class the Bundle function is added to collect the username, stored in a putExtra (), and thus be able to show it in the textView of the name.

```
public class Usuarios extends AppCompatActivity {
    Button btn_game1, btn_game2, btn_volver;
    TextView tv_username;
    ImageView iv_duckhunt, iv_pokemon;
    Bundle nombre_usuario;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_usuarios);

        btn_game1=findViewById(R.id.btn_game1);
        btn_game2=findViewById(R.id.btn_game2);
        btn_volver=findViewById(R.id.btn_volver_usuario);
        iv_duckhunt=findViewById(R.id.iv_duckhunt);
        iv_pokemon=findViewById(R.id.iv_pokemon);
        tv_username=findViewById(R.id.tv_username);

        nombre_usuario=getIntent().getExtras();
        tv_username.setText(nombre_usuario.getString( key: "nombre_usuario"));
    }
}
```

*Imagen 3. Desarrollo 12*

El resto de la clase simplemente incluye tres métodos los cuales solo llevan Intents, bien hacia los distintos juegos que hay, o hacia la pantalla de Login.

En el caso del juego Pokemon Showdown, se ha utilizado un Intent implícito, que te lleva a una URL de Internet.

The rest of the class simply includes three methods which only take Intents, either to the different games out there, or to the login screen.

In the case of the Pokemon Showdown game, an implicit Intent has been used, which takes you to an Internet URL.

```

public void cierra_sesion(View v){
    Intent intent_cerrar=new Intent( packageContext: this,MainActivity.class);
    startActivity(intent_cerrar);
}

public void duck_hunt(View v){
    String username=tv_username.getText().toString();
    Intent intentduckhunt=new Intent( packageContext: this, DuckHunt.class);
    intentduckhunt.putExtra( name: "nombre_usuario",username);
    startActivity(intentduckhunt);
}

public void showdown(View v){
    Intent intent_showdown = new Intent(Intent.ACTION_VIEW, Uri.parse("https://pokemonshowdown.com/"));
    startActivity(intent_showdown);
}

```

Imagen 3. Desarrollo 13

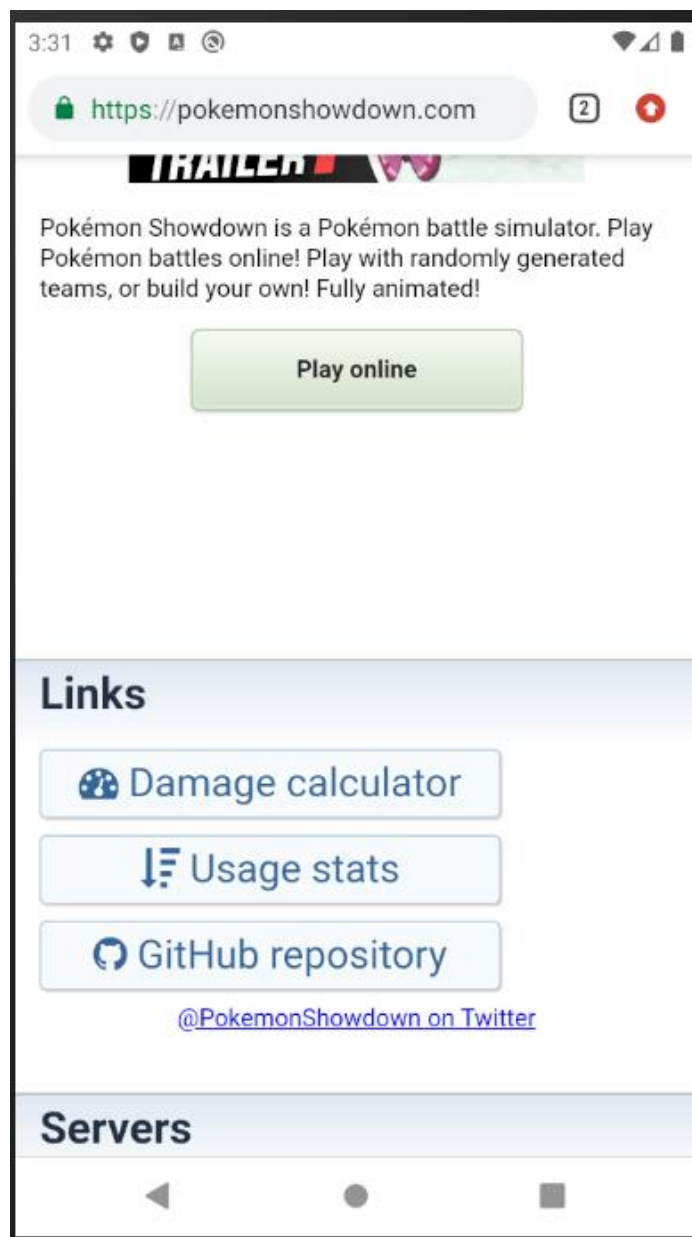
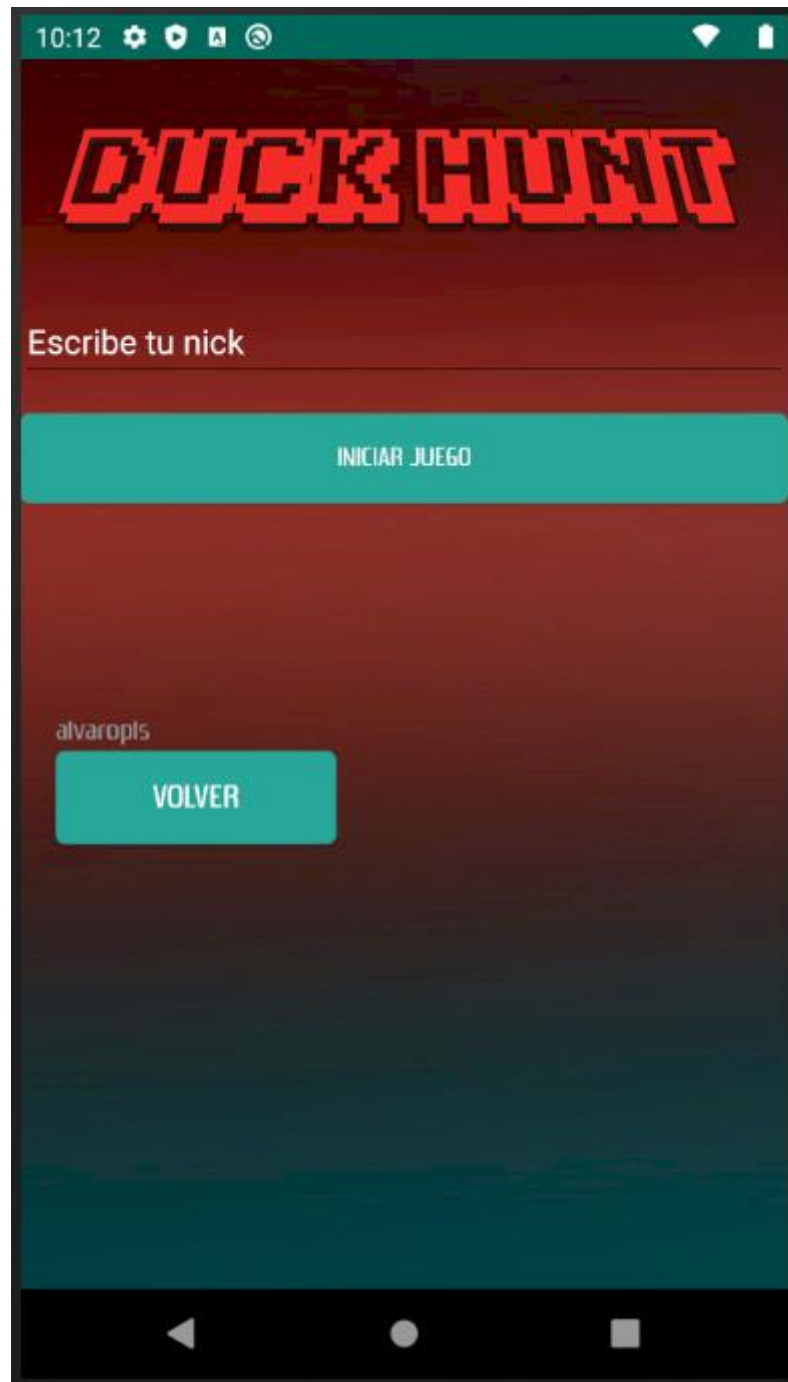


Imagen 3. Desarrollo 14

Si el usuario utiliza el botón de “Duck Hunt”, le llevará a una pantalla como ésta.



*Imagen 3. Desarrollo 15*

En el código de ésta pantalla nos encontraremos el `onCreate` y la condición que permite funcionar al botón de “Iniciar juego”, que te lleva con un `Intent` a la actividad en la que se desarrolla el juego.

In the code of this screen, we will find the `onCreate` method and the condition that allows us to “Start game”, which takes you with an `Intent` to the activity in which the game takes place.

```

public class DuckHunt extends AppCompatActivity {
    Button btnStart,btnVuelve;
    TextView tv_username;
    EditText edNick;
    String nick;
    Bundle nombre_usuario;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_duck_hunt);

        edNick=findViewById(R.id.editTextNick);
        btnStart=findViewById(R.id.buttonStart);
        btnVuelve=findViewById(R.id.btn_back);
        tv_username=findViewById(R.id.tv_nombre);

        nombre_usuario=getIntent().getExtras();
        tv_username.setText(nombre_usuario.getString( key: "nombre_usuario"));
    }

    public void botonStart (View view){
        nick=edNick.getText().toString();
        if(nick.isEmpty()){
            edNick.setError("El nombre es obligatorio");
        }else if(nick.length()<3){
            edNick.setError("El nombre debe tener al menos 3 caracteres");
        }else{
            Intent intent=new Intent( packageContext: this,GameActivity.class);
            intent.putExtra( name: "nick",nick);
            startActivity(intent);
        }
    }
}

```

Imagen 3. Desarrollo 16

También se ha introducido un botón de “Volver”, para regresar al menú principal del usuario.

A "Back" button has also been introduced, to return to the main user menu.

```

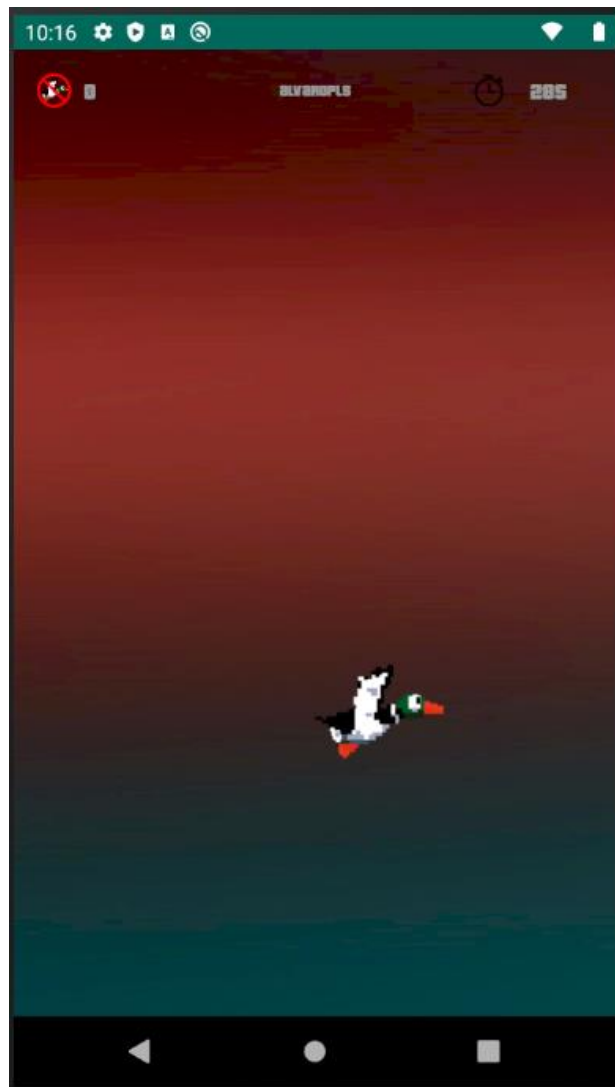
public void volver(View v){
    String username=tv_username.getText().toString();
    Intent intent_vuelta=new Intent( packageContext: this,Usuarios.class);
    intent_vuelta.putExtra( name: "nombre_usuario",username);
    startActivity(intent_vuelta);
}

```

Imagen 3. Desarrollo 17

Al introducir un nick para nuestra partida y habiéndole dado al botón de Iniciar, se nos presenta una ventana en la cual el jugador tiene que tocar el pato de la pantalla para sumar puntos en el marcador de la esquina superior izquierda. El juego consiste en recoger el mayor número de patos en treinta segundos. Una vez termina el tiempo, salta un mensaje en pantalla con el número final de patos y nos da la opción de salir o jugar de nuevo.

When entering a nickname for our game and having pressed the Start button, we get into a window where the player has to touch the duck on the screen to score points in the score of the upper left corner. The game is based in collecting the largest amount of ducks possible in thirty seconds. Once the time is up, a message appears on the screen with the final number of ducks collected and gives us the option to go out or play again.



*Imagen 3. Desarrollo 18*

El código de ésta ventana es algo más complejo, ya que también incluye funciones algo más complicadas como CountDownTimer o los diálogos en pantalla. El método onCreate nos iniciará todos los métodos que después se crean.

The code in this window is somewhat more complex, since it also includes somewhat more complicated functions such as CountDownTimer or the on-screen dialogs. The onCreate method will start all the methods that are later created.

```
public class GameActivity extends AppCompatActivity {

    TextView tvCounterDucks,tvTimer,tvNick;
    ImageView ivDuck;
    int counter=0;
    int anchoPantalla;
    int altoPantalla;
    Random aleatorio;
    boolean gameOver=false;

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

        initViewComponents();
        eventos();
        initPantalla();
        initCuentaAtras();
        moveDucks();
    }
}
```

Imagen 3. Desarrollo 19

Tras esto, creamos el método initCuentaAtras (), que recoge la cuenta regresiva de treinta segundos que se nos muestra en la esquina superior derecha.

After this, we create the initCuentaAtras () method, which collects the thirty-second countdown shown in the upper right corner.

```
private void initCuentaAtras() {
    new CountDownTimer( millisInFuture: 31000, countDownInterval: 1000) {

        public void onTick(long millisUntilFinished) {
            long segundosRestantes=millisUntilFinished/1000;
            tvTimer.setText(segundosRestantes+"s");
        }

        public void onFinish() {
            tvTimer.setText("0s");
            gameOver=true;
            mostrarDialogoGameOver();
        }
    }.start();
}
```

Imagen 3. Desarrollo 20



Lo siguiente que podemos encontrar en el código es el método que inicializa el dialogo que se nos presenta al terminar la cuenta regresiva. Esto inicia un AlertDialog el cual nos mostrará las opciones de volver o de reintentar.

The next thing we can find in the code is the method that initializes the dialog that appears to us when the countdown ends. This starts an AlertDialog which will show us the options to go back or retry.

```
private void mostrarDialogoGameOver() {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setMessage("Has cazado "+counter+" patos").setTitle("Game over");
    builder.setPositiveButton( text: "Reiniciar", (dialog, id) → {
        counter=0;
        tvCounterDucks.setText("0");
        gameOver=false;
        initCuentaAtras();
        moveDucks();
    });
    builder.setNegativeButton( text: "Salir", (dialog, id) → {
        dialog.dismiss();
        finish();
    });
    AlertDialog dialog = builder.create();
    dialog.show();
}
```

Imagen 3. Desarrollo 21

El método initPantalla nos iniciará la pantalla con el ancho y el alto que nosotros hayamos definido en las variables globales del principio.

The initPantalla method will start the screen with the width and height that we have defined in the global variables at the beginning.

```
private void initPantalla() {
    Display display =getWindowManager().getDefaultDisplay();
    Point size =new Point();
    display.getSize(size);
    anchoPantalla=size.x;
    altoPantalla=size.y;

    aleatorio=new Random();
}
```

Imagen 3. Desarrollo 22

El método `initViewComponents` inicia todos los componentes de la pantalla, y recoge en un `Bundle` el Nick que hayamos introducido en la pantalla de inicio del juego.

The `initViewComponents` method starts all the components of the screen, and collects in a `Bundle` the Nick that we have entered in the game's start screen.

```
private void initViewComponents() {
    tvCounterDucks=findViewById(R.id.textViewCounter);
    tvTimer=findViewById(R.id.textViewTimer);
    tvNick=findViewById(R.id.textViewNick);
    ivDuck=findViewById(R.id.imageViewDuck);

    Bundle extras=getIntent().getExtras();
    String nick=extras.getString( key: "nick");
    tvNick.setText(nick);
}
```

*Imagen 3. Desarrollo 23*

Eventos nos creará un `onClick`Listener que comprobará que mientras que no se cumpla la condición `gameOver`, se sumarán puntos al marcador y hará el cambio de imagen en el caso de que el pato sea tocado.

Events will create an `onClick`Listener that will verify that while the `gameOver` condition is not met, points will be added to the marker and will make the image change in the event that the duck is touched.

```
private void eventos() {
    ivDuck.setOnClickListener((v) -> {
        counter++;
        tvCounterDucks.setText(String.valueOf(counter));
        ivDuck.setEnabled(false);
        ivDuck.setImageResource(R.drawable.duck_clicked);

        new Handler().postDelayed(() -> {
            ivDuck.setImageResource(R.drawable.duck);
            moveDucks();
        }, delayMillis: 500);
    });
}
```

*Imagen 3. Desarrollo 24*

MoveDucks es el método que nos moverá de forma aleatoria dentro de la pantalla el objeto ImageView que debemos tocar para sumar puntos a nuestro marcador.

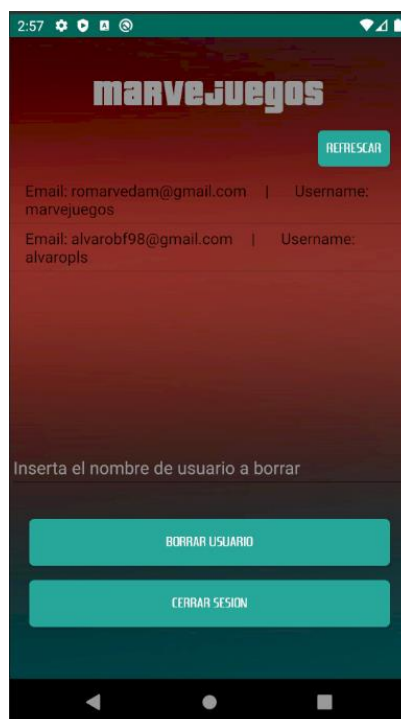
MoveDucks is the method that will move the ImageView object within the screen that we must touch to add points to our marker.

```
private void moveDucks() {  
    int minimo=0;  
  
    int maxX=anchoPantalla - ivDuck.getWidth();  
    int maxY=altoPantalla - ivDuck.getHeight();  
  
    int randomX=aleatorio.nextInt( bound: maxX-minimo+1);  
    int randomY=aleatorio.nextInt( bound: maxY-minimo+1);  
  
    ivDuck.setX(randomX);  
    ivDuck.setY(randomY);  
    ivDuck.setEnabled(true);  
}
```

*Imagen 3. Desarrollo 25*

Por la parte del administrador, solo se podría entrar si cumples el requisito de usar el correo y la contraseña que están determinadas en la clase Java principal para el admin. Una vez introducidas nos aparece una pantalla como ésta

On the administrator side, it could only be entered if you meet the requirement to use the email and password that are determined in the main Java class for admin. Once entered, a screen like this appears



*Imagen 3. Desarrollo 26*

En esta pantalla se pueden identificar tres botones, uno que refrescará la misma para renovar los datos reflejados en el ListView de usuarios; otro para el propio borrado de usuarios, y un último para cerrar sesión. También nos encontraremos un editText el cual nos permite escribir un nombre de los usuarios que tenemos en nuestra base de datos para proceder a borrarlo.

El código de ésta pantalla incluirá cuatro métodos aparte del onCreate, uno para borrado, uno para el refresco de la pantalla, otro para la inclusión de la consulta de los usuarios en el ListView, y otro para el cierre de sesión.

El método onCreate incluirá el inicio del método de consulta, el cual se especifica más adelante.

Three buttons can be identified on this screen, one that will refresh it to refresh the data reflected in the ListView of users; one for the deletion of users, and one for closing session. We will also find an editText that will allows us to write a username from those we have in our database and give us the opportunity to delete it.

The code for this screen will include four methods apart from the onCreate, one for deleting, one for refreshing the screen, one for including the user query in the ListView, and one for logging out.

The onCreate method will include the start of the query method, which is specified later.

```
public class Administrador extends AppCompatActivity {
    ListView lv_admin;
    ArrayList<String> userList=new ArrayList<>();
    ClaseBBDD connection;
    EditText ed_borrado;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_administrador);

        connection=new ClaseBBDD( context: this, name: "users", factory: null, version: 1);
        lv_admin=findViewById(R.id.lv_usuarios);
        ed_borrado=findViewById(R.id.ed_borrado);

        consulta();
    }
}
```

Imagen 3. Desarrollo 27

En el método de borrado nos encontramos la llamada a la base de datos de nuestros usuarios y comprueba a través de una condición que el editText de borrado de usuarios esté lleno o vacío. Si está vacío, simplemente manda un mensaje por pantalla pidiendo que se introduzca un nombre, mientras que si está lleno ejecuta la sentencia de borrado del usuario y confirma la operación con otro mensaje por pantalla.

In the deletion method we find the call to the database of our users and it checks through a condition that the editText of deletion of users is full or empty. If it is empty, it simply sends a message on the screen asking for a name to be entered, while if it is full it executes the user deletion statement and confirms the operation with another message on the screen.

```
public void borrado(View v) {
    ClaseBBDD admin=new ClaseBBDD( context: this, name: "users", factory: null, version: 1);
    SQLiteDatabase borrado=admin.getReadableDatabase();
    String username=ed_borrado.getText().toString();
    if(!username.isEmpty()){
        borrado.delete( table: "users", whereClause: "nombre_usuario='"+username+"'", whereArgs: null);
        Toast.makeText( context: this, text: "Se ha borrado el usuario", Toast.LENGTH_SHORT).show();
        borrado.close();
    }else{
        Toast.makeText( context: this, text: "Introduce un username", Toast.LENGTH_SHORT).show();
    }
}
```

*Imagen 3. Desarrollo 28*

El método de refresco simplemente recarga la pantalla de administrador para mostrar cualquier cambio que se haya podido realizar en la base de datos de usuarios.

The refresh method simply reloads the administrator screen to show any changes that may have been made to the user database.

```
public void refresco(View v){
    Intent refrescar = new Intent( packageContext: this, Administrador.class);
    startActivity(refrescar);
}
```

*Imagen 3. Desarrollo 29*

En el método de consulta hacemos lo mismo que en el método de borrado, pero esta vez instanciamos las columnas email y nombre\_usuario de la base de datos para mostrarlos en el ListView de forma que aparezcan ordenados.

In the query method we do the same as in the delete method, but this time we instantiate the email and username columns from the database to display them in the ListView so that they appear ordered.

```
private void consulta() {
    ClaseBBDD admin=new ClaseBBDD( context: this, name: "users", factory: null, version: 1);
    SQLiteDatabase consulta=admin.getReadableDatabase();
    Cursor fila=consulta.rawQuery( sql: "SELECT email,nombre_usuario FROM users", selectionArgs: null);
    if(fila.moveToFirst()){
        do{
            userList.add("Email: "+fila.getString( columnIndex: 0)+ " | Username: "+ fila.getString( columnIndex: 1));
        }while(fila.moveToNext());
    }
    consulta.close();
    ArrayAdapter<String> adaptador= new ArrayAdapter<>( context: this,android.R.layout.simple_list_item_1,userList);
    lv_admin.setAdapter(adaptador);
}
```

*Imagen 3. Desarrollo 30*

Por último, el método cerrar sesión simplemente te devuelve a la pantalla de login, bien para entrar como un usuario normal, o solo para dejar la sesión de administrador cerrada.

Finally, the logout method simply returns you to the login screen, either to log in as a normal user, or just to leave the administrator session closed.

```
public void cerrar_sesion(View v){
    Intent cerrar=new Intent( packageContext: this,MainActivity.class);
    startActivity(cerrar);
}
```

*Imagen 3. Desarrollo 31*

## **DESARROLLO DE INTERFACES / INTERFACE DEVELOPMENT**

La interfaz está basada en la contraposición de colores cálidos, como es el rojo de la parte superior del fondo, con colores fríos, como son el verde de la parte de abajo del propio fondo y el verde claro de los botones de la aplicación. Hemos optado por un diseño fácil, el cual permite tanto al usuario normal como al administrador reconocer de forma intuitiva todo lo que puede o no puede realizar dentro de nuestra aplicación.

Al ser una aplicación basada en juegos y minijuegos, hemos decidido utilizar una fuente del estilo del reconocido videojuego Grand Theft Auto, de la compañía Rockstar Games, ya que da ese aspecto de interfaz clásica de videojuegos. También se ha optado por el uso del color blanco y gris para todos los letreros y cuadros de texto o botones para resaltar de cara a la vista del usuario dónde puede pulsar y qué debe leer.

The interface is based on the contrast of warm colors, such as red at the top of the background, with cool colors, such as green at the bottom of the background itself and light green on the buttons of the application. We have opted for an easy design, which allows both the normal user and the administrator to intuitively recognize everything that can or cannot be done within our application.

Being an application based on games and minigames, we have decided to use a font style like the renowned video game Grand Theft Auto, from the company Rockstar Games, since it gives that aspect of a classic video game interface. The use of white and gray for all signs and text boxes or buttons has also been chosen to highlight where the user can press and what he should read.

## **MANUALES / MANUALS**

### **MANUAL DE INSTALACIÓN / INSTALLATION MANUAL**

Para la instalación de la aplicación, tenemos varias opciones: descargar la aplicación desde Google Play buscándola por su nombre; o descargar el apk desde el siguiente enlace de MediaFire: <http://www.mediafire.com/file/5b0sassjpoo9d7t/marvejuegos.apk/file>

For the installation of the application, we have several options: download the application from Google Play searching for it by name; or download the apk from the following MediaFire link:

<http://www.mediafire.com/file/5b0sassjpoo9d7t/marvejuegos.apk/file>

### **MANUAL DE USUARIO / USER MANUAL**

El usuario básico de esta aplicación tendrá la opción de registrarse en el caso de que sea nuevo y la opción de loguearse en el caso de que no sea nuevo. Una vez dentro de la aplicación, podrá jugar a los juegos propuestos en la pantalla de Usuarios (ver Imagen 3 Desarrollo 11).

The basic user of this application will have the option to register if it is new and the option to log in if it is not new. Once inside the application, you can play the games proposed on the Users screen (see Image 3 Development 11).

### **MANUAL DE ADMINISTRADOR / ADMIN MANUAL**

El administrador tendrá la posibilidad de loguearse con su email y su contraseña, las cuales vienen definidas ya dentro del onCreate de la pantalla principal (ver Imagen 3 Desarrollo 2). Una vez logueado, podrá consultar usuarios, borrar usuarios y refrescar dentro de la pantalla Administrador (ver Imagen 3 Desarrollo 26).

The administrator will have the possibility to log in with his email and password, which are already defined within the onCreate of the main screen (see Image 3 Development 2). Once logged in, you can consult users, delete users and refresh within the Administrator screen (see Image 3 Development 26).



## **CONCLUSIONES / CONCLUSIONS**

Este proyecto, al ser más corto de lo normal, es algo más limitado de lo que a mí me hubiera gustado realizar, pero creo que cumple con la idea que yo generé en un principio de tener una aplicación sencilla en la que pudiese tener juegos creados por mí o juegos que yo considere buenos y pueda redirigir a través de la misma. De todas formas ha sido útil de cara al futuro, ya que me ha ayudado a resolver situaciones en las que nunca me había visto comprometido antes.

This project, being shorter than normal, is somewhat more limited than I would have liked to do, but I think it complies with the idea that I originally generated of having a simple application in which I could have games created by me or games that I consider good and can redirect through it. In any case, it has been useful for the future, as it has helped me to resolve situations in which I had never been involved before.

## **BIBLIOGRAFÍA / BIBLIOGRAPHY**

Para la realización de los diagramas:

- <https://app.diagrams.net/>
- <https://www.lucidchart.com/pages/es>

Información e introducción:

- <https://www.xatakamovil.com/n/los-juegos-que-marcaron-un-hito-en-la-evolucion-de-los-moviles>
- <https://developer.android.com/studio/intro>