

# Game Source

Proyecto Desarrollo de Aplicaciones  
Multiplataforma

Miguel Parra Peñas  
Xavier García Robles  
Jorge Ballesteros Alonso

2º DAM

# Introducción

## Resumen

El propósito de esta aplicación es la de hacer más sencilla y eficaz la obtención de los videojuegos que están disponibles de forma gratuita con el fin de incrementar la comodidad de los usuarios y *Gamers* que los disfrutamos a diario. Haciendo que la demás gente se incluya dentro de esta comunidad dado que serán más accesibles los videojuegos gratuitos.

El medio que usaremos es Android Studio como IDE para poder desarrollar dicha aplicación para Android, dado que es el SO más extendido del mercado, además de ser el más común dentro del sector de bajo coste dentro de los teléfonos inteligentes.

Para la obtención y disposición de datos hemos usado Selenium con Python para hacer Web Scraping a las principales webs de distribución de videojuegos, tales como Steam, PlayStation, EpicGames.

## Abstract

The purpose of this app is to make it easier and efficient the way people look up for free games. At the end we want to increase the commodities of the gamer community and avoid barriers for new people, also help the most unfortunate people to this medie by helping them to play for free and making new people get involved in our community.

The medium that we are going to use is Android Studio as the IDE to develop our app for Android. Due to its the most extended OS in the world and usually is used in low cost smartphones.

For the obtaining and provision of the information we used Python as program language with Selenium and beautifulsoup libraries to scrape in different webs and obtain the information we need. We worked with different pages like epic games, steam, or ps Store for the moment.

# Agradecimientos

Este proyecto no habría sido llevado a cabo de la misma manera de no ser por la gente que nos rodea, haciendo especial incapié en nuestros profesores, quienes nos transmitieron todo el conocimiento y las ganas de crecer necesarios para seguir adelante a pesar de las dificultades



Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las siguientes condiciones:

- Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su enlace.
- La copia será literal y completa
- No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia

original completa(jurídicamente válida) que puede encontrarse en:

<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>

# Índice

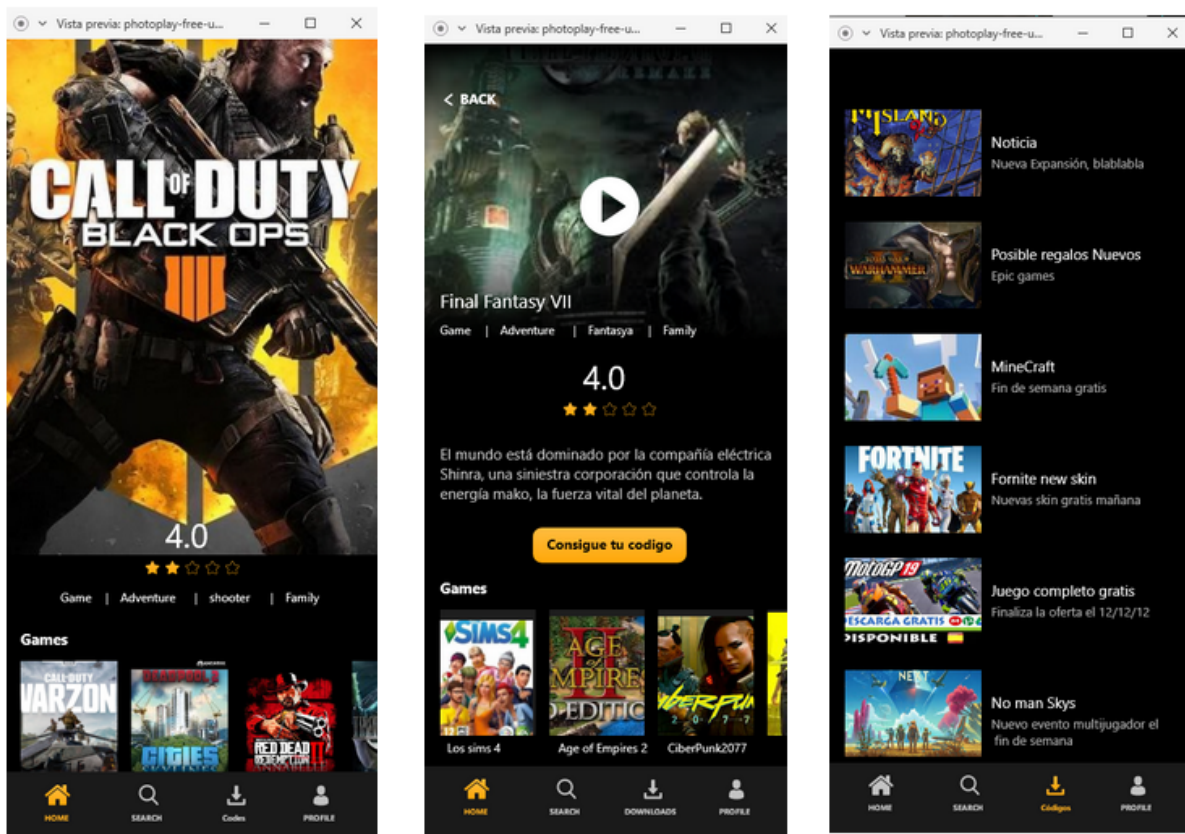
<b>Introducción</b>	<b>2</b>
Resumen	2
Abstract	2
<b>Agradecimientos</b>	<b>3</b>
<b>Índice</b>	<b>4</b>
<b>Desarrollo del proyecto</b>	<b>5</b>
Diseño en Android Studio	6
Dependencias gradle	8
Colores xml	8
Themes	8
String xml	9
Splash Screen	9
Login Activity	10
Register Activity	12
Barra de navegación	14
Main Activity	18
Calendar Activity	18
Search Activity:	19
Favoritos Activity	19
Option Activity	20
SwipeRefreshLayout:	22
<b>Obtención de datos de Firebase</b>	<b>23</b>
Clase AccesoFirebase.java	23
Filtrado por búsqueda de usuario	24
<b>Back End</b>	<b>25</b>
Objetivo y automatización	25
Diseño y herramientas: Programación	25
Diseño y herramientas: Base de Datos	26
IMPORTS	27
EJEMPLO CLASE JUEGO	27
Proceso de creación: Documentación y Formación	28
EJEMPLO DE SCRAPING	28
EJEMPLO SUBIDA DE DATOS A FIREBASE	29
EJEMPLO BAJADA DE IMAGEN Y SUBIDA AL STORAGE	30
EJEMPLO DE CRONTAB EN RASPBERRY	31
<b>Conclusiones</b>	<b>33</b>
<b>Webgrafía</b>	<b>34</b>

## Desarrollo del proyecto

Para desarrollar convenientemente el proyecto, inicialmente hicimos un mockup para poder partir de una base y así tener un objetivo: llegar a tener una estética determinada.

Para comenzar, elaboramos un mockup con el diseño que queríamos que tuviera la aplicación ya terminada.

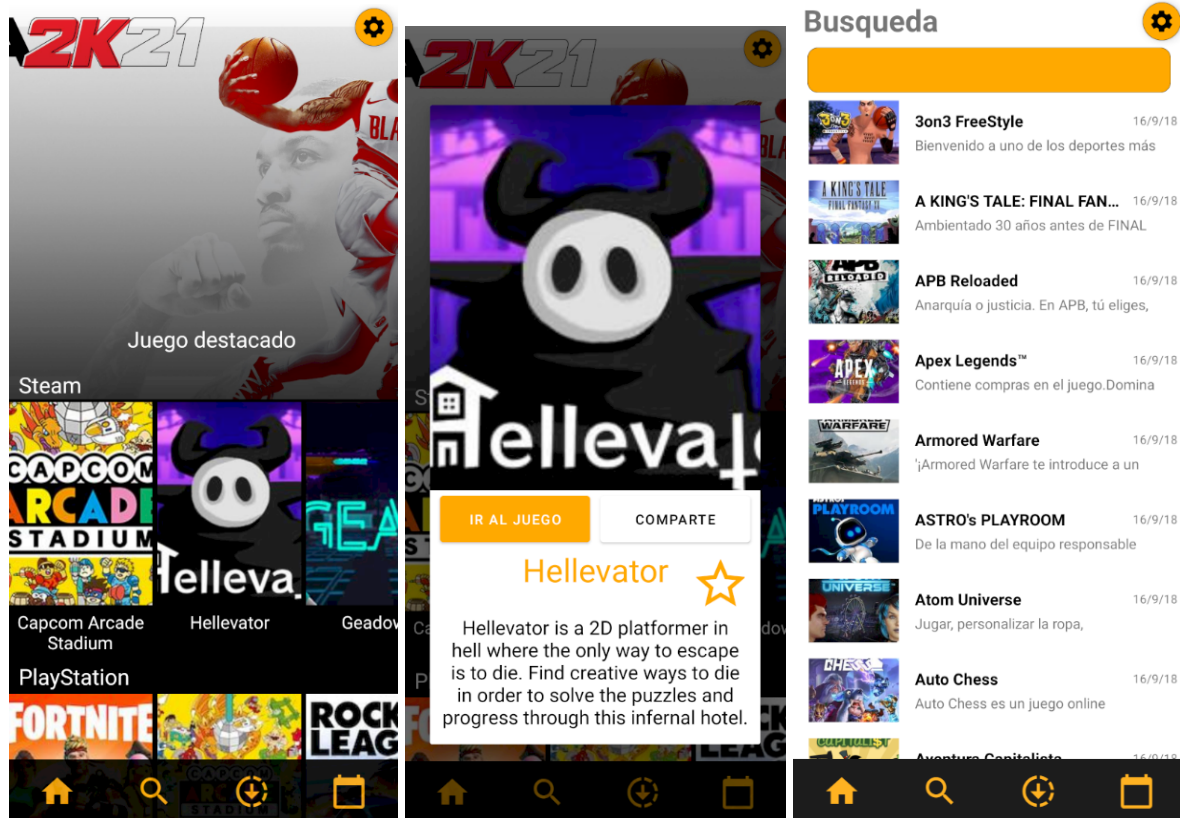
Nos basamos en una plantilla dado que nos gustaba la combinación de colores y la disposición de los elementos. Pero la remodelamos a nuestro gusto para adaptarla a nuestras necesidades.



Así era como queríamos que se viese la aplicación, ahora tocaba diseñarla en Android Studio, el IDE escogido para ello.

## Diseño en Android Studio

Teniendo el diseño elegido, nos pusimos a plasmarlo en la plataforma de desarrollo ciñéndonos a él. El resultado fue este tras afinar todos los detalles que queríamos.



La forma de implantar una interfaz gráfica en Android es mediante Layouts en formato XML. Por ejemplo, el archivo del Main es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout>

    <ScrollView>

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <ImageView/>

            <View />

            <androidx.appcompat.widget.AppCompatTextView/>

            <androidx.appcompat.widget.AppCompatTextView />

            <androidx.appcompat.widget.AppCompatTextView/>

            <androidx.appcompat.widget.AppCompatTextView />

            <androidx.recyclerview.widget.RecyclerView/>

            <androidx.recyclerview.widget.RecyclerView />

            <androidx.recyclerview.widget.RecyclerView />

            <androidx.constraintlayout.widget.Guideline/>

            <TextView/>
        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>

    <androidx.swiperefreshlayout.widget.SwipeRefreshLayout>

        <ListView />

    </androidx.swiperefreshlayout.widget.SwipeRefreshLayout>

    <androidx.constraintlayout.widget.ConstraintLayout>

        <ImageView/>

        <ImageView />

        <ImageView />

        <ImageView
            app:srcCompat="@drawable/calendar_logo_foreground" />

        <androidx.constraintlayout.widget.Guideline />

        <androidx.constraintlayout.widget.Guideline/>

        <androidx.constraintlayout.widget.Guideline />

    </androidx.constraintlayout.widget.ConstraintLayout>
</com.google.android.material.floatingactionbutton.FloatingActionButton/>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Dentro de cada una de las etiquetas se definen los atributos de los elementos que tendrá la pantalla correspondiente.

## Dependencias gradle

Implementamos varias dependencias para la funcionalidad y el diseño de la aplicación:

- implementation 'com.google.android.material:material:1.2.1'
- implementation "androidx.swiperefreshlayout:swiperefreshlayout:1.1.0"
- implementation 'com.firebaseui:firebase-ui-database:7.1.1'

## Colores xml

Colores principales establecidos para la app:

- Black\_GameSource: **#FF000000**
- White\_GameSource: **#FFFFFFFF**
- Orange\_GameSource: **#FFA90A**
- Gray\_GameSource: **#2B2A2A**
- Grayligh\_GameSource: **#3A3939**

## Themes

Se han implementado dos themes, modo normal y modo oscuro. En cada theme está el modo normal y el modo sin barra de acción por defecto de android.

Los colores de los temas son los implementados en colores



## String xml

Todos los textos de la aplicación están implementados en el archivo de idiomas string, con el fin de poder implementar varios idiomas con facilidad.

Strings.xml es un archivo que permite la traducción de la aplicación a otros idiomas de forma muy sencilla y automática, ya que es el propio Android el que determina el idioma del dispositivo y cambia el archivo que se usa en cada situación para adaptarse a las necesidades del usuario en cuestión.

## Splash Screen

El diseño del Splash Screen está siguiendo el patrón de diseño de MockUp, con los colores establecidos.

se ha añadido un sonido de inicio al abrir la aplicación.

Aparece de fondo en movimiento el nombre de la app, de forma sutil y dándole más profundidad, mientras aparece otra animación entrelazando las iniciales de la app y un rebote entre las palabras de Game source, dejando al final las letras iniciales y el nombre de la app entrelazadas.

Todo esto ocurre en cinco segundo de forma coordinada con el sonido inicial y entonces pasa a la pantalla de login.

Animaciones usadas: 8.

Tipos de animaciones: Alpha, Translate, Scale.

Sonidos inicializados: 1.



## Login Activity

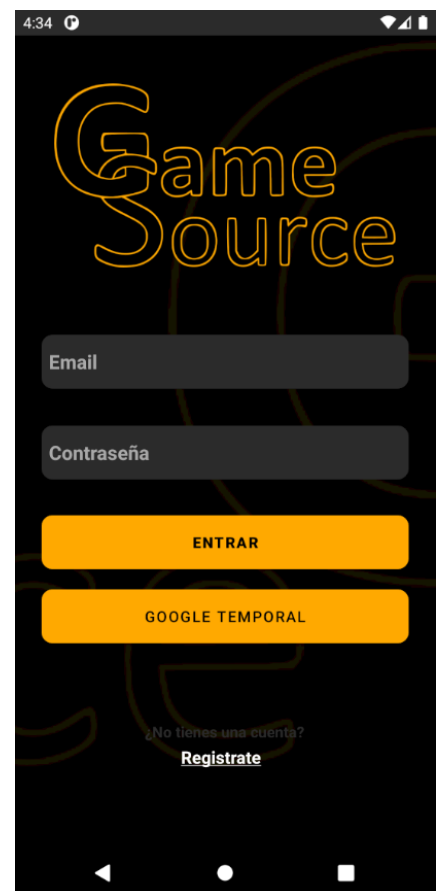
Login activity sigue la misma estética que el Splash Screen, contiene una animación sutil de fondo, con el nombre de la aplicación en movimiento. Tiene un estilo sencillo donde el usuario tiene 3 opciones:

- Introducir su email y password si ya tiene cuenta para acceder a la aplicación.
- Entrar con su cuenta de google si dispone de una o crearse una en el mismo apartado.
- Acceder al Activity register para poder registrarse sin google account.

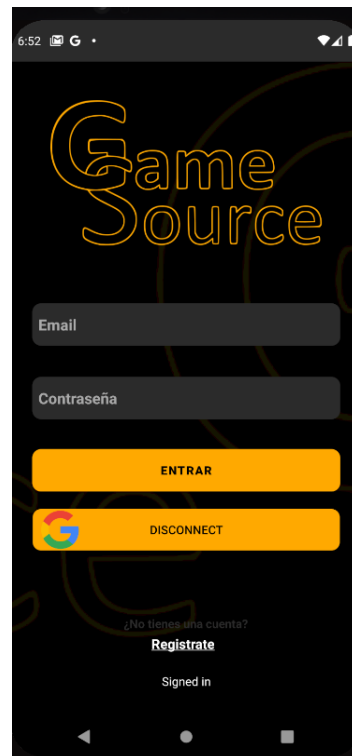
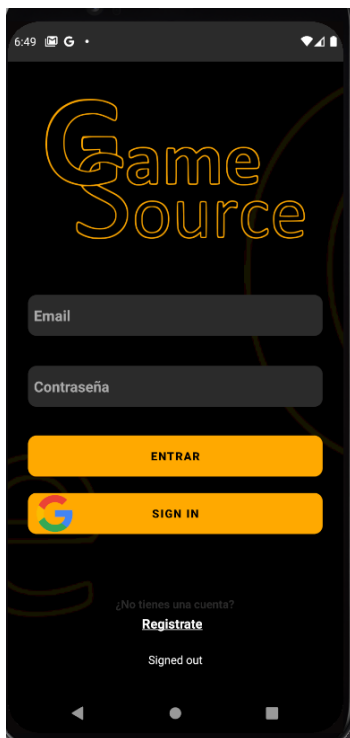
Dispone de dos text input con un estilo gris, un hint sutil donde hace referencia a lo que tiene que introducir el usuario, en el momento de escribir el hint desaparece y se escribe los caracteres del usuario con un color de letra naranja, siguiendo el estilo de la app.

Tanto los botones como los text input disponen de un redondeado, definido en round.xml, con el fin de hacerlos más atractivos a la vista y están implementados con google.android.material.

El Activity implementa de OnClickListener y todos los botones están en un Switch case, para que la navegación sea más ordenada y el código esté más limpio.



Ya se han implementado los métodos necesarios para las conexiones con las cuentas de Google y poder acceder a la aplicación usando la autenticación de Google.



También se añadirá recuperación de cuenta en caso de pérdida de contraseña.

Animaciones usadas: 2.

Tipos de animaciones: Translate.

String: todo.

## Register Activity

El Activity de registro tiene el mismo estilo y la misma dinámica, comparten los mismos redondeo, colores, implementación y mantiene las mismas animaciones que en login.

La diferencia principal es, que en este caso los datos que el usuario introduce y le da al botón de registrar, estos se guardaran en la base de datos de Firebase y a partir de ahora podrá entrar con estas credenciales desde el login.



Buscar por dirección de correo electrónico, número de teléfono o UID de usuario					Agregar usuario		
Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario			
casa@gmail.com		26 may. 2021	26 may. 2021	S8xdCqy4vGdlXkGn1EYVB2PnoJ3			
cuentadeprueba@gmail.com		3 jun. 2021	3 jun. 2021	pED0ia8xY3NcA1qJPZAcNdro4v33			
pepe@gmail.com		26 may. 2021	31 may. 2021	sXagS8WkW7YL5S6xQRGxEaQXO...			
Filas por página: 50					1 – 3 of 3	<	>

Cada campo de esta ventana funciona acorde con el contenido que requiere, es decir, que el campo del email te abre el teclado con la “@”, además de sugerir los correos usados frecuentemente.


De igual manera lo hacen los campos de las contraseñas, a medida que se escribe, los caracteres se convierten en puntos para ocultar la contraseña que estamos introduciendo.

Se han establecido unos parámetros de complejidad para que la contraseña tenga un mínimo de caracteres con el fin de mantener las cuentas de los usuario más seguras.

En la base de datos sólo almacenamos los hash de las contraseña para mayor seguridad.

#### Parámetros de hash de contraseña



 Estos parámetros incluyen información sensible de la cuenta del usuario. Asegúrate de mantenerlos en privado.


La siguiente información se puede usar para migrar usuarios con contraseña.


```
hash_config {  
  algorithm: [REDACTED]  
  base64_signer_key: [REDACTED]  
  base64_salt_separator: [REDACTED]  
  rounds: [REDACTED]  
  mem_cost: [REDACTED]  
}
```



Nuestros proveedores de servicio de momento son Firebase y Google

#### Proveedores de acceso

Proveedor	Estado
 Correo electrónico/contraseña	Habilitada

 Google	Habilitada
--	------------

Pero dentro de poco implementaremos más proveedores como facebook, Twitter etc.

 Facebook	Inhabilitado
 Twitter	Inhabilitado

## Barra de navegación

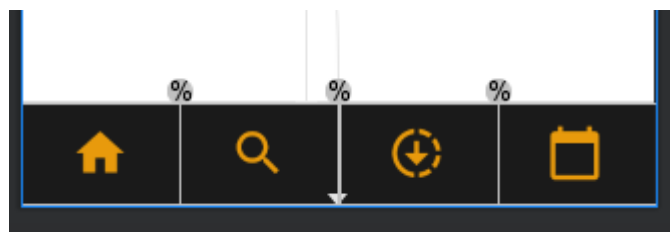
Una vez que se accede a la aplicación todos los activity compartes una barra de navegación, donde se podrá mover por todos los escenarios de la aplicación fácilmente.



El diseño de la barra de navegación está compuesto por:

Un Constraint Layout que almacena todos los componentes de la barra de navegación

Se han establecido 3 Guideline que establecen el ancho de cada botón dejándolos simétricos a un 25% por botón y ajustándose al tamaño de la pantalla del dispositivo en el que se esté ejecutando la app.



Los botones en si son las imágenes que se ven dentro del constraint y anclados a los Guideline.

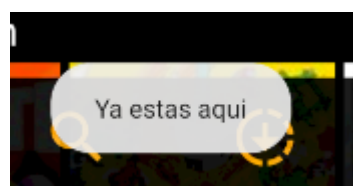
```
<ImageView
    android:id="@+id/img_Home_Logo"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    app:layout_constraintEnd_toStartOf="@id/guideline_25"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/home_logo_foreground" />
```

Se les ha implementado un OnClickListener y declarado en las clases java

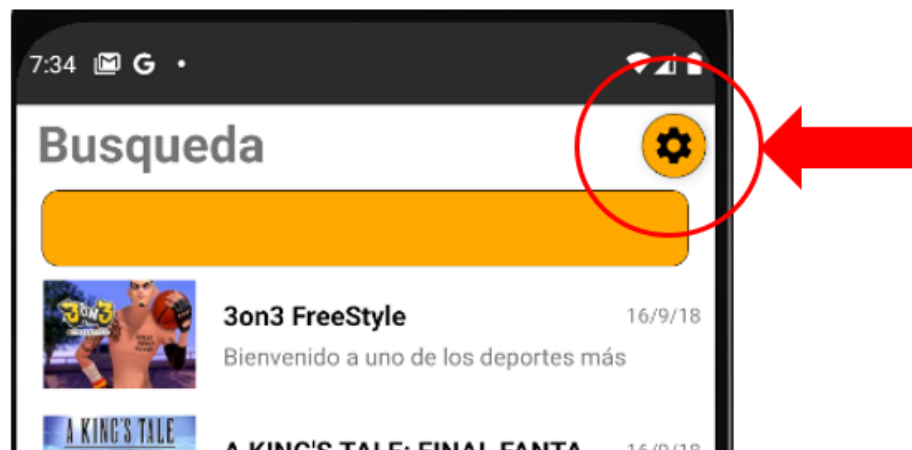
```
//Inicialización de los botones del bottom app bar  
findViewById(R.id.img_Home_Logo).setOnClickListener(this);  
findViewById(R.id.img_Search_Logo).setOnClickListener(this);  
findViewById(R.id.img_Historial_Logo).setOnClickListener(this);  
findViewById(R.id.img_Calendar_Logo).setOnClickListener(this);
```

```
@Override  
public void onClick(View v) {  
    switch (v.getId()) {  
        case R.id.img_Home_Logo:  
            Toast.makeText(context, this, "Ya estas aqui",  
                Toast.LENGTH_SHORT).show();  
            break;  
        case R.id.img_Search_Logo:  
            Intent intent9 = new Intent( packageContext, MainActivity.this, SearchRecycler.class);  
            startActivity(intent9);  
            break;  
        case R.id.img_Historial_Logo:  
            Intent intent10 = new Intent( packageContext, MainActivity.this, FavoritosActivity.class);  
            startActivity(intent10);  
            break;  
        case R.id.img_Calendar_Logo:  
            Intent intent11 = new Intent( packageContext, MainActivity.this, CalendarActivity.class);  
            startActivity(intent11);  
    }  
}
```

Cada botón dirige a su activity correspondiente excepto si pulsas el botón del activity en el que ya estás situado, en cuyo caso saltará un mensaje avisando que ya te encuentras en ese activity



También se ha implementado un botón alternativo, que te permite ir a un activity de opciones donde dispondrás de diversas opciones



Este botón va a parte de la barra de navegación, ya que es un botón que no se usa mucho.

El diseño del botón es un botón flotante o Fab button

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/btn_fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/orange_GameSource"
    android:src="@drawable/img_setting_foreground"
    app:backgroundTint="@color/black_GameSource"
    app:fabCustomSize="38dp"
    app:fabSize="normal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:ignore="ContentDescription" />
```



El diseño interior está definido en un XML

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:width="158dp"  
    android:height="158dp"  
    android:viewportWidth="108"  
    android:viewportHeight="108"  
    android:tint="@color/orange_GameSource">  
    <group android:scaleX="5.0"  
        android:scaleY="5.0"  
        android:translateX="-6.0"  
        android:translateY="-6.0">  
        <path  
            android:fillColor="@color/white_GameSource"  
            android:pathData="M19.14,12.94c0.04,-0.3 0.06,-0.61 0.06,-0.94c0,-0.32 -0.02,-0.64 -0
```

con el que lo adaptamos al Fab button.

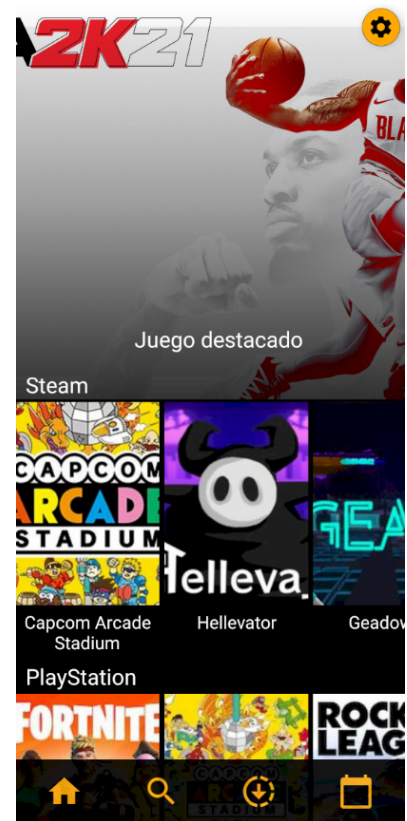
Este botón estará disponible en las cuatro pantallas principales de navegación.

## Main Activity

Lo que caracteriza a la pantalla principal de nuestra aplicación es el uso de tres RecyclerView que nos muestran todo el catálogo de videojuegos disponibles en nuestra base de datos

Todo el Layout de esta pantalla está encapsulado dentro de un ScrollView dado que no cabe toda la información de un vistazo, así el usuario puede deslizar la pantalla hacia abajo y puede ver el resto de la información.

Al igual que el resto de nuestra aplicación, todos los Strings de la misma, están en el archivo Strings.xml el cual facilita inmensamente la traducción de la misma a otros idiomas.

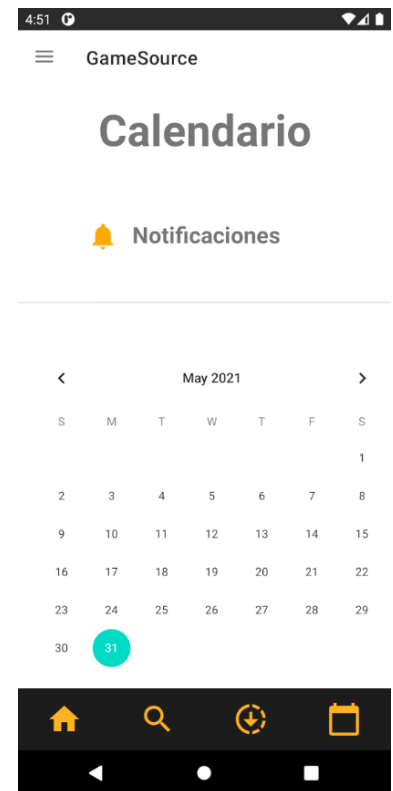


## Calendar Activity

Este activity implementa un botón de notificaciones, donde se podrán configurar las notificaciones que desea el usuario (Por implementar). También se muestra un calendario de Google, donde se puede seleccionar la fecha que se desee y se mostrarán los lanzamientos de los videojuegos correspondientes a esa fecha.

(Falta por implementar que se queden registradas las fechas de los próximos lanzamientos de juegos o eventos futuros.)

El objeto calendario no es el que viene predefinido de Google, para poder añadir los eventos dentro del mismo calendario, tuvimos que investigar alternativas y encontramos una librería (ver webgrafía) que implementa un calendario altamente personalizable y más sencillo de usar.

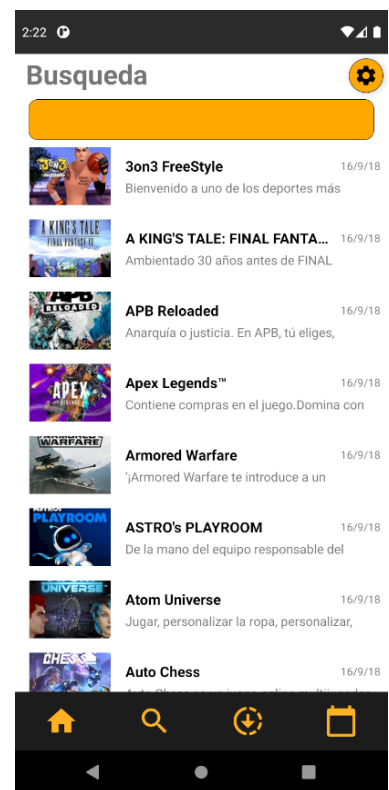


## Search Activity:

A través de este activity el usuario será capaz de buscar el videojuego que quiera en segundos dado que solo con escribir la primera letra del videojuego, se van actualizando los resultados sin importar las mayúsculas y minúsculas que contenga el título del videojuego.

Los videojuegos que van apareciendo lo hacen con una transición que los inserta en un RecyclerView vertical que muestra la foto, el título y el inicio de la descripción del mismo.

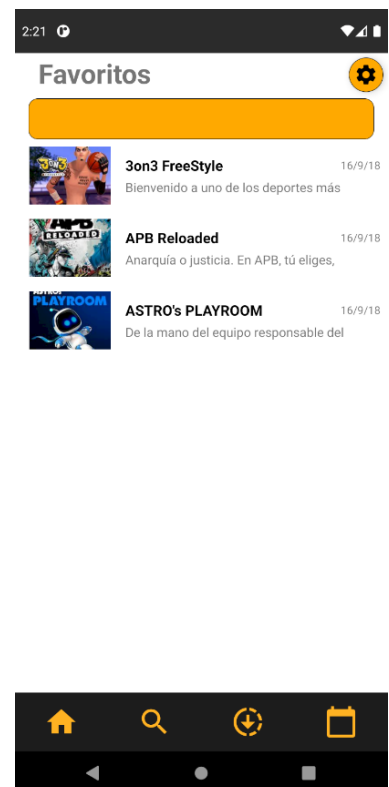
Al pulsar en cada uno de ellos, nos abre la misma ventana de información que aparece al pulsarlos en el Main Activity.



## Favoritos Activity

Este Activity contiene los videojuegos que se han marcado como favorito en cualquiera de las otras vistas.

El aspecto es el mismo que el de la pantalla de búsqueda con la diferencia de que en el título de la ventana cambia el texto de “búsqueda” por el de “favoritos”



## Option Activity

Este activity tiene un diseño minimalista y contará con 4 opciones en principio



El activity dispone de los colores del theme de la aplicación y el logo. Tiene implementado la barra de navegación para facilitar cuando quiera salir el usuario del activity.

Dispone de la opciones: Notificaciones, activa y desactiva con pulsar el botón o el texto.

Opciones

Game Source



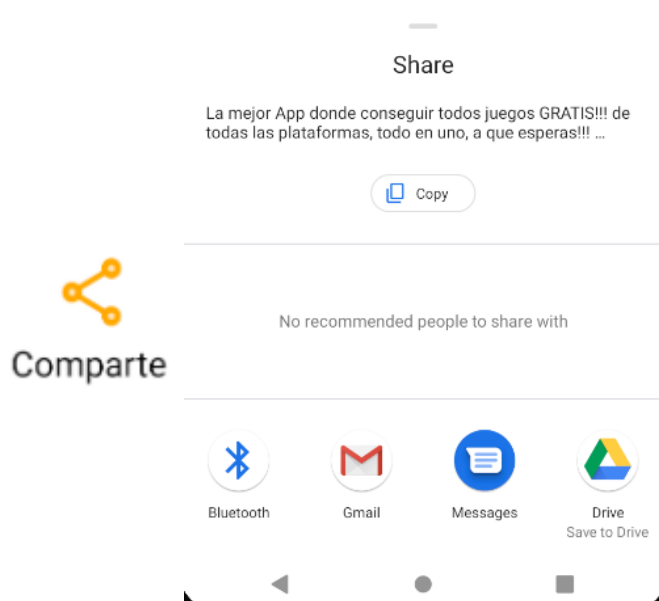
Notificaciones

Opciones

Game Source



Notificaciones



Comparte, el usuario pulsado este botón, se le abrirá una ventana donde podrá elegir distintas formas de enviar un mensaje prefijado con una breve descripción de la aplicación así como el link de PlayStore para que puedan descargarla.



Puntúa, con este botón le llevará directamente a la PlayStore a la sección de reseñas, con el fin de que puedan puntuar y añadir reseñas sobre la aplicación, para que otros usuarios puedan beneficiarse de la experiencia del usuario que pone la reseña.

Por último y de momento está el botón de Ayuda, este botón aún no tiene la implementación, pero la idea es o que le dirija a una guía de usuario o que pueda contactar directamente con el personal de la aplicación.

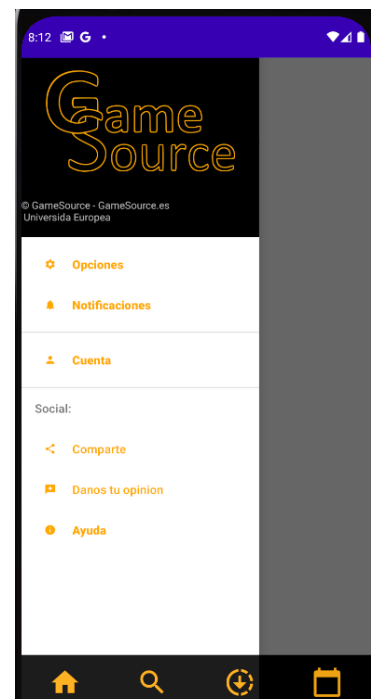


**Ayuda**

Próximamente se añadirán nuevas opciones como, selección de notificaciones por plataforma, por eventos o por juegos.

Selección de temas desde la aplicación.

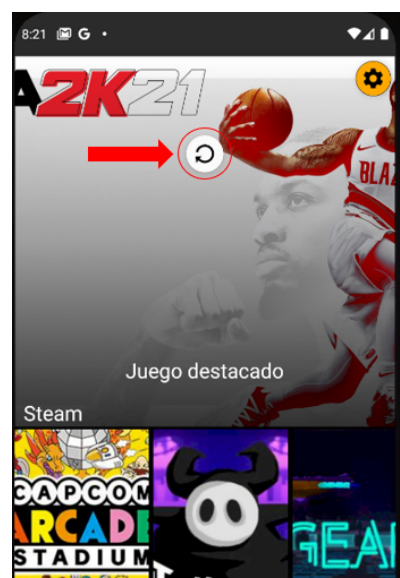
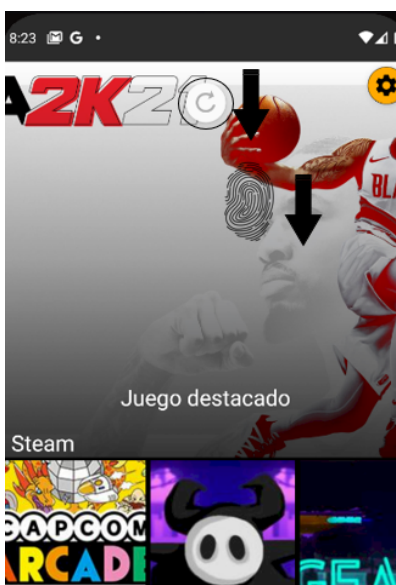
También se añadirá un menú lateral para hacer más accesible la navegación en la app.



## SwipeRefreshLayout:

Se ha implementando un refresher en el main, pero se ma incorporar en todos los activity, gracias a esta función el usuario podrá actualiza el activity con el simple gesto de pasar con el dedo por la pantalla hacia abajo y soltar, con esto el activity entero se actualizará por si hay algún dato nuevo que llegará después de la inicialización de la aplicación.

```
protected SwipeRefreshLayout.OnRefreshListener mOnRefreshListener = new SwipeRefreshLayout.OnRefreshListener() {  
  
    @Override  
    public void onRefresh() {  
        AccesoFirebase.obtenerVideojuegosGratis(context);  
        AccesoFirebase.obtenerVideojuegosPS(context);  
        AccesoFirebase.obtenerVideojuegosSteam(context);  
        Toast.makeText(context: MainActivity.this, text: "Actualizado", Toast.LENGTH_SHORT).show();  
  
        swipeLayout.setRefreshing(false);  
    }  
};
```



# Obtención de datos de Firebase

## Clase AccesoFirebase.java

La forma en la que accedemos a los datos de la base de datos es a través de una clase Java intermedia que se encarga de gestionar todas las peticiones del usuario a la misma, tanto la de obtención como la de inserción.

En total hay 7 métodos que obtienen los datos de diferentes maneras, tanto los favoritos del usuario logueado como por cada plataforma para los RecyclerView del MainActivity hasta la obtención filtrada de los videojuegos de la pantalla de búsqueda.

Para manejar las Callbacks se han creado varias interfaces que son implementadas en las activity que lo requieran para evitar fallos en la creación de los adaptadores de los RecyclerView, puesto que se crean una vez han llegado los datos a través de la implementación de la interfaz correspondiente.

```
/**
 * Método que devuelve todos los videojuegos gratuitos de la base de datos
 * Firebase
 *
 * @param a Interfaz de actualización para poder recuperar los datos en el main
 * o en la pantalla que se le requiera
 */
public static void obtenerVideojuegosGratis(ActualizarVideojuegosGratis a) {
    Log.d("MENSAJE", "Obteniendo datos de Firebase...");
    ArrayList<Videojuego> videojuegosGratis = new ArrayList<Videojuego>();
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull @NotNull DataSnapshot snapshot) {
            for (DataSnapshot esnapshot : snapshot.getChildren()) {
                for (DataSnapshot eesnapshot : esnapshot.getChildren()) {

                    videojuegosGratis.add(eesnapshot.getValue(Videojuego.class));
                }
            }
        }
    });
}
```

```

    }
    //Log.d("MENSAJE", snapshot.getValue(Videojuego.class).toString());
    a.recuperarVideojuegos(videojuegosGratis);
}

@Override
public void onCancelled(@NonNull @NotNull DatabaseError error) {

}
});}

```

Todos los métodos han sido debidamente documentados para una correcta interpretación de los mismos así como para su uso de cara a una expansión de la aplicación. También se han seguido todas las convenciones de código y usado nombres de variables descriptivos.

## Filtrado por búsqueda de usuario

Dado que nuestra aplicación tiene una pantalla de búsqueda, todos los datos deben ser filtrados por el parámetro de búsqueda del usuario. Esto se hace a través de la obtención de los datos de Firebase, con el método `orderByChild("nombre").startAt(parámetroDelUsuario)` se ordenan los videojuegos por el nombre y únicamente los que comienzan por el texto que está introduciendo el usuario.

El método que se encarga de realizar dicha tarea es el siguiente:

```

public static void obtenerVideojuegosFiltrado(ActualizarVideojuegosGratis a, String nombre)
{
    HashSet<String> nombres = new HashSet<>();
    FirebaseDatabase databaseF = FirebaseDatabase.getInstance();
    DatabaseReference myRefFiltrar =
    databaseF.getReference().child("gratis").child("ps_store_free");

    Log.d("MENSAJE", "Obteniendo datos de Firebase...");
    ArrayList<Videojuego> videojuegosGratis = new ArrayList<Videojuego>();

    myRefFiltrar.orderByChild("nombreMin").startAt(nombre.toLowerCase()).endAt(nombre.toLowerCase() + "\uffff").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull @NotNull DataSnapshot snapshot) {
            for (DataSnapshot esnapshot : snapshot.getChildren()) {
                if (nombres.add(esnapshot.getValue(Videojuego.class).getNombre()))
                    videojuegosGratis.add(esnapshot.getValue(Videojuego.class));
            }
        }

        DatabaseReference myRefFiltrarSteam =
        databaseF.getReference().child("gratis").child("steam_free");

        myRefFiltrarSteam.orderByChild("nombreMin").startAt(nombre.toLowerCase()).endAt(nombre.toLowerCase() + "\uffff").addValueEventListener(new ValueEventListener() {

```



```

        @Override
        public void onDataChange(@NonNull @NotNull DataSnapshot snapshot) {
            for (DataSnapshot esnapshot : snapshot.getChildren()) {
                if
(nombres.add(esnapshot.getValue(Videojuego.class).getNombre()))
                    videojuegosGratis.add(esnapshot.getValue(Videojuego.class));

            }

            Log.d("MENSAJE", videojuegosGratis.toString());
            a.recuperarVideojuegos(videojuegosGratis);
        }

        @Override
        public void onCancelled(@NonNull @NotNull DatabaseError error) {

        }

    });
}

@Override
public void onCancelled(@NonNull @NotNull DatabaseError error) {

}

});
}

```

## Back End

### Objetivo y automatización

En este proyecto hemos decidido automatizar la obtención de datos. Para ello vamos a realizar Web Scraping sobre distintas páginas, principalmente tiendas de las que obtendremos los juegos que se encuentran disponibles de forma gratuita, obtendremos los datos que necesitamos y los subiremos a nuestra base de datos formateados como nos interese.

Esto lo realizaremos a través de varios proyectos pequeños que se automatizan sobre un sistema linux con la herramienta cron.

### Diseño y herramientas: Programación

Primero pensamos en qué contenedor íbamos a colocar nuestro programas que automaticen la recogida y subida de datos. decidimos utilizar una raspberry 3b+ debido a su bajo consumo eléctrico y alta disponibilidad.



Como el lenguaje nativo para desarrollar en raspberry es python decidimos seguir con él, utilizando las librerías de selenium, para la automatización de acciones y esperas en las páginas, y beautifulsoup para el scraping, de esta manera nos también podriamos realizar el proyecto en 2 lenguajes distintos.

Para la navegación de selenium utilizamos el navegador Google Chrome

Finalmente incluimos las librería pyrebase para conectar con nuestra base de datos, también incluimos las librerías request, random, os, time, datetime, PIL e io, para las distintas funcionalidades necesarias durante el proceso tales como bajar imagenes, parsear fechas o seleccionar tiempos aleatorios de espera para evitar antibots.

## Diseño y herramientas: Base de Datos

Decidimos utilizar Firebase por su sencillez a la hora de trabajar con ella, además de ser gratuita y tener experiencia previa con ella, también nos permite utilizar más funcionalidades aparte de Realtime Database como el Storage que usamos para almacenar imágenes.

Los scripts de python conectaran con ambos servicios a través de la librería pyrebase.

## IMPORTS

```
import urllib
from bs4 import BeautifulSoup
import os
import time
from random import randint
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
import pyrebase
import json
from PIL import Image
from io import BytesIO
import requests
```

## EJEMPLO CLASE JUEGO

```
class Juego():
    def __init__(self, nombre, fecha, descripcion, my_db_image, generos, url_origen):
        self.nombre = nombre
        self.nombre_min = nombre.lower()
        self.fecha = fecha
        self.descripcion = descripcion
        self.my_db_image = my_db_image
        self.generos = generos
        self.plataforma = 'pc'
        self.url_origen = url_origen

    def __str__(self):
        return (f"el juego {self.nombre} con fechas {self.fecha} estado {self.estado} ")

    def __json__(self):
        return {'nombre': self.nombre, 'nombreMin': self.nombre_min, 'fechas':
        [self.fecha[0],self.fecha[1]],
        'descripcion': self.descripcion, 'image_url': self.my_db_image, 'generos': self.generos,
        'url_origen': self.url_origen, 'plataforma': self.plataforma}
```

## Proceso de creación: Documentación y Formación

Primero estuvimos buscando información de cómo podríamos realizar todo esto ya que carecíamos de conocimientos suficientes para realizarlos.

Empezamos informándonos sobre selenium y beautifulsoup ya que los problemas de recolección de datos fueron los primeros que encontramos, primero con un breve curso en la plataforma openwebinar y luego a través de internet, buscando tanto en la documentación oficial como en páginas de consultas.

## EJEMPLO DE SCRAPING

```
url = 'https://www.epicgames.com/store/es-ES/'

try:
    chromeOptions = webdriver.ChromeOptions()
    prefs = {"download.default_directory" : os.getcwd()+"/Resultados
Selenium"}
    chromeOptions.add_experimental_option("prefs",prefs)
    driver = webdriver.Chrome("chromedriver",options=chromeOptions)
    driver.get(url)
except Exception as e:
    print(e)

WebDriverWait(driver,5).until(EC.presence_of_element_located((By.CSS_S
ELECTOR,'div[class="css-shu771"]')))
html =
driver.find_element_by_xpath("//body").get_attribute('outerHTML')
soup = BeautifulSoup(html, "lxml")
driver.quit()
section = soup.find_all('section',
class_='css-1nzk0w-CardGrid-styles__groupWrapper')
divs = soup.find_all("div", class_="css-shu771")
imgs =
soup.find_all('img',class_='css-1s4ypbt-Picture-styles__image-OfferCardImageA
rt_picture-OfferCardImageLandscape__picture-Picture-styles__visible')
juegosLista = []
cont = 0
urls_juegos = getUrlJuegos(section)
for div in divs:
    datos_juego = []
```

```

        spans = div.find_all('span')
        image_url_page = imgs[cont]['data-image']
        for span in spans:
            datos_juego.append(span.text)
        juegosLista.append(sacar_datos(datos_juego, urls_juegos[cont],
image_url_page))
        cont += 1

```

Tras conseguir nuestros primeros pasos y conseguir nuestra primera extracción de datos investigamos sobre la conexión con firebase en lenguaje python, donde encontramos 2 posibles librerías que usar ya que solo 2 eran compatibles con python 3. Decidimos usar pyrebase por su sencillez.

## EJEMPLO SUBIDA DE DATOS A FIREBASE

```

db = firebase.database()
db.child('gratis').child('ps_store_free').remove()
for j in juegos_lista:
    db.child("gratis").child("ps_store_free").push(j.__json__())

```

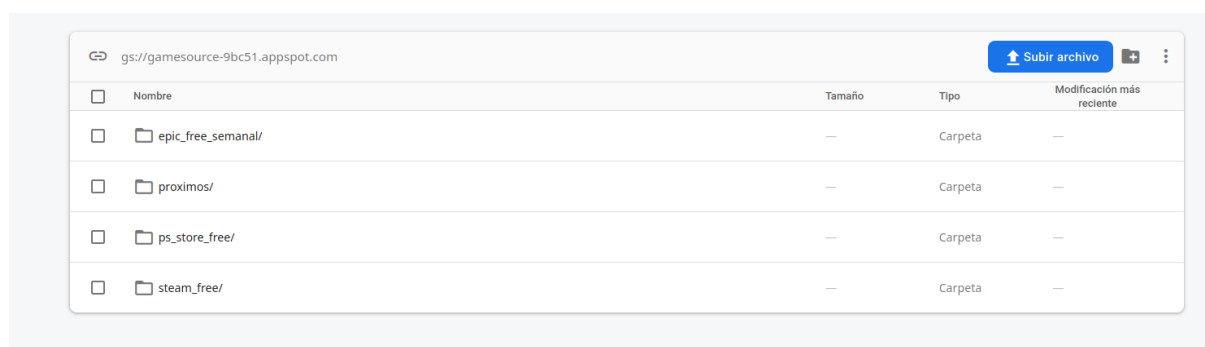












Tras esta primera parte empezamos a ampliar páginas encontrándonos con problemas como la necesidad de crear patrones de Web Crawling para poder extraer datos en su totalidad sobre los juegos, debido a la cantidad de información a procesar. También decidimos sumar en esta parte la automatización de descarga y subida de imágenes al Firebase

## EJEMPLO BAJADA DE IMAGEN Y SUBIDA AL STORAGE

```
def guardarImagen(image_url, nombreJ):
    nombreJ = nombreJ.replace('/', '*').replace(' ', '')
    image_object = requests.get(image_url)
    image = Image.open(BytesIO(image_object.content))
    image.save("img/ps_store_free/" + nombreJ + "." + image.format,
image.format)
    nombre_juego = nombreJ + "." + image.format

storage.child('ps_store_free/'+nombre_juego).put("img/ps_store_free/" +
nombre_juego)
url_my_db = storage.child('ps_store_free/'+
nombre_juego).get_url(None)
return url_my_db
```



gs://gamesource-9bc51.appspot.com > proximos				Subir archivo	
<input type="checkbox"/>	Nombre	Tamaño	Tipo	Modificación más reciente	
<input type="checkbox"/>	 Akiba'sTrip:Hellbound&Debriefed.JPEG	32.06 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 AlexKiddinMiracleWorldDX.JPEG	28.66 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 BlasterMasterZero3.JPEG	30.87 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 Chicory:AColorfulTale.JPEG	24.67 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 Chivalry2.JPEG	19.42 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 CrisTales.JPEG	35.63 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 DCSuperHeroGirls:TeenPower.JPEG	33.31 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 DariusBurstAnotherChronicleEX+.JPEG	30.86 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 DestroyAllHumans!.JPEG	28.37 KB	image/jpeg	2 jun. 2021	
<input type="checkbox"/>	 Don'tForgetMe.JPEG	27.9 KB	image/jpeg	2 jun. 2021	

Tras esto vino el último problema, automatizar en una raspberry los scripts, los cuales no funcionaban al principio por un problema con el selenium ya que necesitaba permisos específicos para abrir el navegador chromiun.

## EJEMPLO DE CRONTAB EN RASPBERRY

```
00 11 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority
/usr/bin/python2.7 /home/pi/sourceScript/script.py ; rm
/home/pi/sourceScript/img/scriptFolder/*
```

```
pi@pi: ~  
Tabs Help  
3.2 /tmp/crontab.1ZE9h8/crontab  
is file to introduce tasks to be run by cron.  
task to run has to be defined through a single line  
ing with different fields when the task will be run  
at command to run for the task  
line the time you can provide concrete values for  
e (m), hour (h), day of month (dom), month (mon),  
ay of week (dow) or use '*' in these fields (for 'any').  
e that tasks will be started based on the cron's system  
on's notion of time and timezones.  
ut of the crontab jobs (including errors) is sent through  
l to the user the crontab file belongs to (unless redirected).  
example, you can run a backup of all your user accounts  
5 a.m every week with:  
* * * 1 tar -zcf /var/backups/home.tgz /home/  
r more information see the manual pages of crontab(5) and cron(8)  
h dom mon dow command  
00 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/python3 /home/pi/sourceScript/epic_free_semanal.py ; rm /home/pi/sourceScript/img  
12 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/python3 /home/pi/sourceScript/epic_free_semanal.py ; rm /home/pi/sourceScript/img  
02 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/python3 /home/pi/sourceScript/humble.py  
13 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/python3 /home/pi/sourceScript/humble.py  
04 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/python3 /home/pi/sourceScript/steam_free.py ; rm /home/pi/sourceScript/img/steam_f  
16 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/python3 /home/pi/sourceScript/steam_free.py ; rm /home/pi/sourceScript/img/steam_fr  
05 * * * env -i DISPLAY=:0.0 XAUTHORITY=/home/pi/.Xauthority /usr/bin/python3 /home/pi/sourceScript/ps_Store_free.py ; rm /home/pi/sourceScript/img/ps_St  
Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo M-A Mark Text To Bracket Previous  
Exit Read File Replace Uncut Text To Spell Go To Line M-E Redo M-G Copy Text Where Was Next
```

Como se puede observar el comando de crontab tiene dos partes separadas por “;”. La primera lanza el script y la segunda borra las fotos una vez que esta se ha completado para evitar la acumulacion masiva de imagenes en la raspberry.



# Conclusiones

Resumiendo todo lo aprendido, hemos extraído información de las distribuidoras principales de videojuegos gratuitos, hemos usado nuestro sistema de gestión de la información en la nube (Firebase) tanto para subir la información recolectada como para usarla y disponerla al usuario.

Esta disposición de la información la hemos hecho siguiendo un diseño inicial, además de filtrarse por la plataforma de la que proviene dicha información. También se filtran según la búsqueda por nombre que introduzca el usuario y se actualizan los resultados a cada carácter que introduzca el mismo.

Así pues, hemos logrado desarrollar una aplicación completa, con su proyecto a futuro, dado que queremos añadir más funcionalidades y fuentes de videojuegos. También queremos añadir mejores opciones de búsqueda, además de filtros.

# Webgrafía

Aquí se encuentran las diferentes páginas web que hemos consultado para poder realizar todas las características que queríamos que tuviera el proyecto:

Para poder editar el texto del botón de iniciar sesión de Google:

<https://stackoverflow.com/questions/18040815/can-i-edit-the-text-of-sign-in-button-on-google>

Para eliminar datos de Firebase:

<https://firebase.google.com/docs/firestore/manage-data/delete-data?hl=es>

Para hacer los iconos adaptables de la app:

[https://developer.android.com/guide/practices/ui\\_guidelines/icon\\_design\\_adaptive?hl=es-419](https://developer.android.com/guide/practices/ui_guidelines/icon_design_adaptive?hl=es-419)

La librería del calendario que hemos usado:

<https://github.com/SundeepK/CompactCalendarView>

Para acceder a Firebase desde Python:

<https://github.com/thisbejim/Pyrebase>