CFGM DESARROLLO DE APLICACIOENES MULTIPLATAFORMA

Proyecto Final de Ciclo

TITULO

Autor: Mario Zapata Martínez

Tutor: Alvaro Ortega Marmol

Fecha de entrega: 4/12/2019

Convocatoria: S1 2019-2020

Índice

1.- INTRODUCCIÓN

1.1.- MOTIVACIÓN

Es uno de mis pasatiempos consumir el tipo de contenido del que trata el proyecto, series y películas, por lo que había pensado en realizar algo que estuviese relacionado con este tema y cumpliese los requisitos para ser un proyecto.

1.2.- ABSTRACT

This project tries to create an Android application that gets information from a database hosted in Firebase. The type of information stored in the database will be about series, movies and producers.

The application will show in its main activity a list built with a RecyclerView, which will contain series and movies sorted by release date in descending order. In this list, for each element, an image, a name and the genres to which it belongs will be displayed. You can click on each element of this list, which will lead to another activity with more detailed information about the chosen element. There will be three types of different elements: series, movie and producer. Each of these elements will have its own activity since they will store different information.

In addition, the application will have a search function by name, which will search series, movies and producers.

I will also implement a system to remember and authenticate users, so that a user can register and later login with their account. Logged in users will be able to bookmark content, which will be displayed in an activity other than the main activity that can be accessed from a lower navigation bar.

1.3.- OBJETIVOS PROPUESTOS (GENERALES Y ESPECÍFICOS)

Los objetivos generales son los siguientes:

1. Realizar un proyecto que me valga para aprobar el proyecto de fin de grado superior.
2. Crear una aplicación Android utilizando Android Studio como entorno de desarrollo y almacenando la información en una base de datos Firebase.

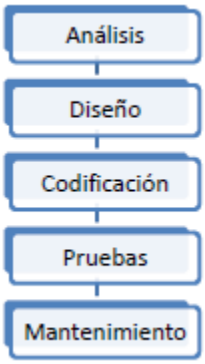
Los objetivos específicos son los siguientes:

1. Crear la base de datos en Firebase.
2. Crear la actividad principal de la aplicación con un RecyclerView.
3. Crear una segunda actividad que muestre información sobre un elemento concreto del RecyclerView.
4. Conseguir que la aplicación se comunique con la base de datos.

5. Implementar la función de búsqueda por nombre.
6. Implementar un sistema de autenticación de usuarios que permita logearse con una cuenta Google.
7. Implementar la función de marcado de contenido.
8. Crear la barra de navegación inferior.
9. Crear la actividad en la que los usuarios ven su contenido marcado.

2.- METODOLOGÍA UTILIZADA

Para la realización de este proyecto utilizaré el modelo de desarrollo en cascada. En este modelo, las etapas del desarrollo software tienen un orden, de forma que para empezar una etapa es necesario finalizar la etapa anterior.



Fases del desarrollo software en cascada

Este modelo permite hacer iteraciones. Por ejemplo, durante la etapa de mantenimiento del producto, el cliente requiere una mejora, esto implica que hay que modificar algo en el diseño, lo cual significa que habrá que hacer cambios en la codificación y se tendrán que realizar de nuevo las pruebas. Es decir, si se tiene que volver a una de las etapas anteriores, hay que recorrer de nuevo el resto de las etapas.

3.- TECNOLOGÍA Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO

Las tecnologías que voy a utilizar para realizar el proyecto son las siguientes: Android Studio, Java, XML, Firebase.

- Android Studio: es el entorno de desarrollo oficial para el desarrollo de aplicaciones para Android. Elijo esta herramienta porque proporciona más especificación a la hora de programar, un editor de diseño que permite arrastrar y soltar componentes de la interfaz de usuario, herramientas Lint para detectar problemas de rendimiento, plantillas para crear diseños comunes de Android y un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones.

- Java: es un lenguaje de programación utilizado por Android Studio para escribir el código ejecutable de la aplicación Android. Java es un lenguaje orientado a objetos, lo que quiere decir que el software se diseña de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones, formando entidades llamadas objetos.

- XML: es un lenguaje de marcas utilizado por Android Studio para diseñar la interfaz de usuario de la aplicación.

- Firebase: es una plataforma de desarrollo de aplicaciones web y aplicaciones móviles ubicada en la nube. Elijo esta herramienta para crear la base de datos de mi aplicación puesto que proporciona una base de datos en tiempo real y organizada en forma de árbol JSON, la información se almacena en la nube de Firebase. La sincronización en tiempo real de esta base de datos permite que los usuarios accedan a la información de sus datos desde cualquier dispositivo, y cada vez que se realice una modificación en los datos, se almacena dicha información en la nube y se notifica simultáneamente al resto de dispositivos.

4.- ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN

Para la planificación y seguimiento del proyecto he realizado un diagrama de Gantt, en el que se ve reflejado la estimación del tiempo necesario para realizar cada tarea y el tiempo real dedicado. Las columnas intermedias del diagrama representan las horas dedicadas a cada tarea en la semana que especifican.

ACTIVIDAD	16/12/2019 a 22/12/2019	2/01/2020 a 05/01/2020	06/01/2020 a 12/01/2020	13/01/2020 a 19/01/2020	20/01/2020 a 26/01/2020	27/01/2020 a 02/02/2020	03/02/2020 a 06/02/2020	ESTIMACIÓN	TOTAL
DOCUMENTACIÓN	5 horas	1 hora	2 horas	1 hora 30 minutos	30 minutos	1 hora	16 horas	30 horas	27 horas
ANÁLISIS	25 horas							15 horas	25 horas
DISEÑO		8 horas						10 horas	8 horas
IMPLEMENTACIÓN		4 horas	9 horas	14 horas	3 horas	21 horas		50 horas	51 horas
PRUEBAS		30 minutos	1 hora 30 minutos	4 horas	1 hora	6 horas		10 horas	12 horas

Diagrama de Gantt

5.- DESARROLLO DEL PROYECTO

5.1.- ANÁLISIS

Para la fase de análisis primero he realizado un diagrama de Entidad-Relación con la herramienta web www.lucidchart.com, en la que he creado las siguientes entidades:

- Serie: esta entidad tiene como clave primaria un identificador, también tiene una clave foránea que contiene el identificador de la productora de la serie. Además contiene información específica de la serie.

- Película: esta entidad tiene como clave primaria un identificador, también tiene una clave foránea que contiene el identificador de la productora de la película. Además contiene información específica de la película.

- Productora: esta entidad tiene como clave primaria un identificador, además contiene información específica de la productora.

- Categoría_Serie: la clave primaria de esta entidad está compuesta por el nombre de la categoría y por el identificador de la serie a la que corresponde, que es también una clave foránea.

- Categoría_Película: la clave primaria de esta entidad está compuesta por el nombre de la categoría y por el identificador de la serie a la que corresponde, que es también una clave foránea.

- Usuario: esta entidad tiene como clave primaria un identificador, además contiene información específica del usuario.

- Marcador: la clave primaria de esta entidad está compuesta por un identificador de usuario y por un identificador de contenido. Además, contiene el tipo de contenido para poder saber a qué entidad se refiere el identificador de contenido (serie o película).

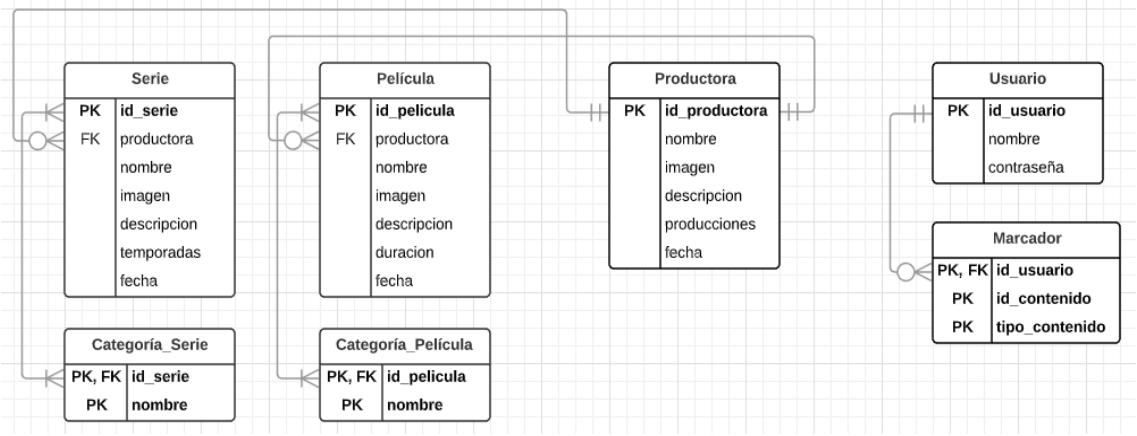


Diagrama de Entidad-Relación

Aunque como voy a usar la base de datos de Firebase, la cual es no relacional, este diagrama no me vale para mucho. Este diagrama sólo lo usaré para hacerme una idea de como estructurar la base de datos en objetos JSON.

Después he analizado los requisitos funcionales y no funcionales de la aplicación:

Requisitos funcionales:

- La actividad principal contendrá el nombre de la aplicación, un RecyclerView, un botón de búsqueda, un menú que contenga la opción de cerrar sesión y una barra de navegación inferior.

- El RecyclerView de la actividad principal contendrá una lista de series y películas ordenadas por fecha de salida en orden descendiente.

- El botón de búsqueda debe realizar una búsqueda por nombre de series, películas y productoras en la base de datos.

- Al clicar en un elemento del RecyclerView se abrirá una segunda actividad que contendrá información más específica del elemento y un botón que permita marcar el elemento.

- La barra de navegación inferior debe poder llevar a la actividad que muestra los contenidos marcados por el usuario.

- El botón de marcado debe estar presionado si el elemento ha sido marcado anteriormente, de forma que no se pueda marcar el mismo elemento más de una vez.

- Cuando se aprieta el botón de marcado, si ya está presionado, se debe eliminar en la base de datos el elemento, de forma que ya no conste como contenido marcado del usuario.

Requisitos no funcionales:

- No se podrá realizar ninguna operación sobre la base de datos si el usuario no ha iniciado sesión.

- El inicio de sesión debe ser seguro, para ello se utilizarán los métodos de la biblioteca FirebaseAuth.

- La interfaz de la aplicación debe ser simple, se deben poder visualizar todas las funciones que es capaz de realizar la aplicación. También deben ser intuitivas, utilizando iconos relacionados con la función que se va a usar, por ejemplo: para la función de búsqueda utilizar el icono de una lupa.

- La aplicación debe cargar los contenidos del RecyclerView con cierta rapidez y sin usar barras de progreso.

5.2.- DISEÑO

Para la fase de diseño primero he realizado unos diagramas de casos de uso con la herramienta web www.lucidchart.com:

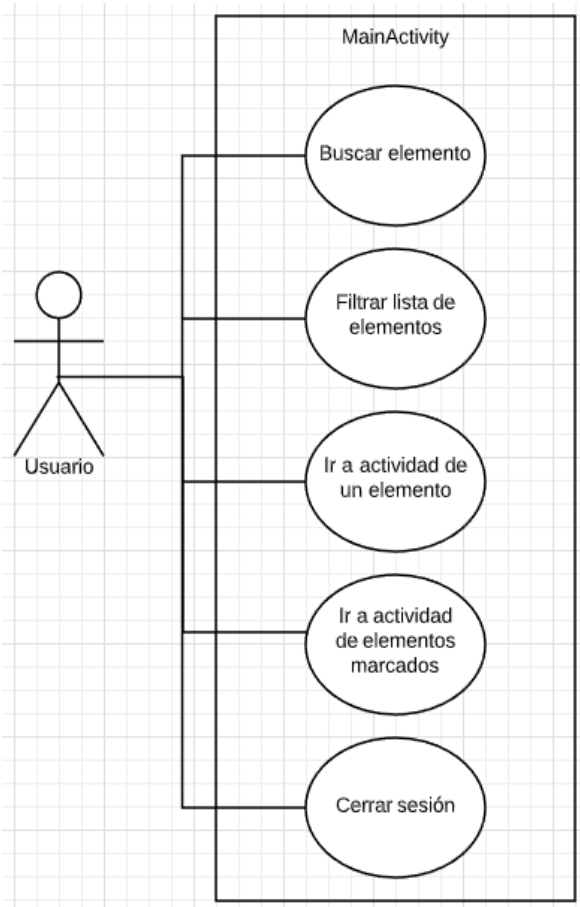


Diagrama de casos de uso 1

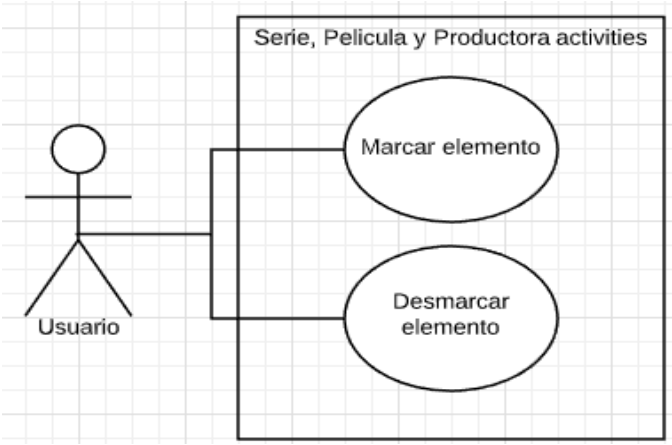


Diagrama de casos de uso 2

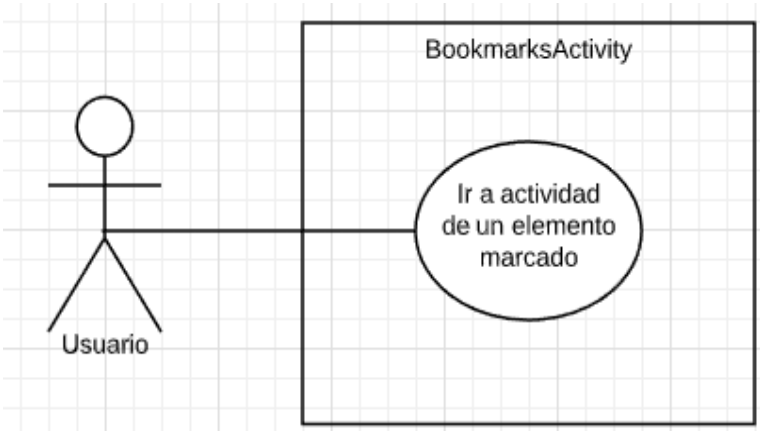


Diagrama de casos de uso 3

5.2.1. Crear la base de datos de Firebase

Primero he leído la documentación y una guía de inicio en https://firebase.google.com/ para crear un proyecto en Firebase y agregarlo al proyecto Android.

Para crear el proyecto en Firebase y conectarlo con Android he seguido los siguientes pasos:

- En console.firebase.google.com he creado un nuevo proyecto al que he llamado infoseries-firebase.

- Con Android Studio he creado el proyecto Android con una actividad principal vacía al que he asignado un nivel de API mínimo de 16 (requisito de Firebase).

- En la consola de Firebase he registrado la aplicación Android que acabo de crear y he descargado el archivo de configuración google-services.json.

- En el proyecto Android, después de añadir el archivo de configuración he añadido el SDK de Firebase al gradle.

- Por último he ejecutado la aplicación para verificar que se ha instalado y todo funciona correctamente. Para poder ejecutar la aplicación he tenido que crear un dispositivo virtual con el AVD Manager de Android Studio y habilitar la virtualización en la BIOS.

Antes de empezar a crear una base de datos en Firebase he decidido hacer un curso gratuito en www.udacity.com llamado Firebase in a weekend en el que he aprendido:

- Firebase almacena los datos en objetos JSON.

- Los ID de los objetos son únicos y creados por Firebase, se llama pushed IDs. De esta forma si dos dispositivos envían datos al mismo tiempo, como el ID lo crea Firebase, se asegura que el ID va a ser diferente para estos datos.

- Para trabajar con imágenes hay que añadir la dependencia: 'com.github.bumptech.glide:glide'.

- En la actividad principal hay que crear los siguientes atributos para trabajar con Firebase: FirebaseDatabase fbDb (referencia a toda la base de datos de Firebase) y DatabaseReference dbR (referencia a una parte específica de la base de datos).

- Para crear la instancia de la base de datos: FirebaseDatabase.getInstance().

- Para crear la instancia de una parte específica: fbDb.getReference() .child(String nombre)

- Para enviar datos a la base de datos: dbR.push().setValue(Object objeto)

- Mientras que la autenticación de usuario no está implementado hay que cambiar las reglas de seguridad de Firebase a "true" para poder hacer pruebas.

- Normalmente habría que estar comprobando continuamente si hay algún cambio en la base de datos, pero con Firebase, es Firebase quien se encarga de informar de forma proactiva a todos los dispositivos que ha habido un cambio.

- Para que la aplicación sepa cuando Firebase le dice que se ha realizado un cambio se debe implementar un ChildEventListener. Después hay que sobreescribir los métodos del ChildEventListener para indicar lo que la aplicación debe hacer, dependiendo del tipo de cambio que haya ocurrido en la base de datos (add, change o remove) la aplicación debe hacer cosas distintas. Por último hay que añadir a la referencia de la base de datos el ChildEventListener dbR.addChildEventListener(cEL).

- He aprendido sobre cómo configurar las reglas de seguridad en Firebase.

- Con FirebaseUI puedes habilitar muchas formas de autenticación de usuarios como Google, Facebook, Email, etc.

- Para utilizar FirebaseUI debes habilitar las formas de autenticación que quieres usar en la consola de Firebase, añadir las dependencias necesarias e implementar en el código de la aplicación la autenticación de usuarios.

- En este punto he dejado el curso de Firebase de lado puesto que los siguientes pasos en el curso son la implementación de la autenticación de usuarios, y el almacenamiento de imágenes, ambas son funcionalidades que planeo implementar más adelante en el proyecto. Pero, de momento quiero simplemente crear los datos en la base de datos y crear la actividad principal de la aplicación.

En el directorio java del proyecto he creado el paquete activities para almacenar las actividades de la aplicación y el paquete models para almacenar los Plain Java Objects (POJOs) de los que va a hacer uso la aplicación para almacenar datos en la base de datos.

He creado el POJO para la clase Productora con los atributos: nombre, imagen, descripcion, producciones y fecha.

Para crear una pequeña cantidad de datos para hacer pruebas y empezar a construir la aplicación, he creado primero una interfaz para subir datos a Firebase. Con esta interfaz he subido 2 productoras.

A parte de aprender como pushear datos a Firebase (utilizo push para que los IDs se generen automáticamente), he tenido varios problemas para poder almacenar el objeto Date de una forma satisfactoria y que cumpla con el formato que quiero (dd/MM/yyyy). Al final, después de bastante tiempo buscando formas de realizar esta tarea, he decidido guardar la fecha usando una variable long en la que almaceno el resultado del método getTime() de la clase Date, ya que es lo que se recomienda. De esta forma, puedo simplificar el almacenamiento de la fecha y puedo transformar en

cualquier momento este long a un objeto Date, ya que la clase Date contiene un constructor Date(long).

He creado el POJO para la clase Serie con los atributos: nombre, productora, imagen, descripcion, temporadas y fecha. Y también el POJO para la clase Película con los atributos: productora, nombre, imagen, descripcion, duracion, fecha.

Respecto a las categorías de las series y las películas, he decidido almacenarlas en las clases Serie y Película, ya que como no tengo necesidad de consultar solamente las categorías de las series para ninguna operación, las puedo almacenar como un String, separando cada categoría con un delimitador para luego poder usar la función split() y obtener un array de String con las categorías. De esta forma las clases Serie y Película siguen siendo POJOs.

He almacenado 2 películas y 8 series con la interfaz que he creado en la aplicación para mandar datos a Firebase. Con estos datos ya puedo empezar a desarrollar otros aspectos de la aplicación.

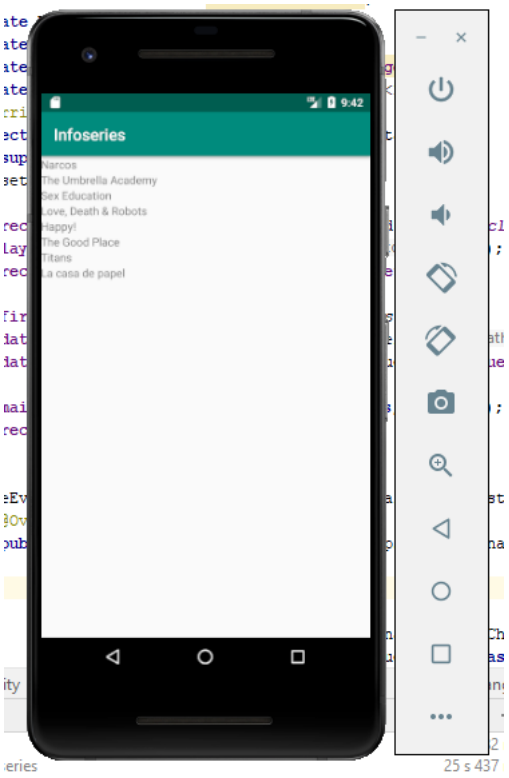
5.2.2 Creación del RecyclerView de la actividad principal

Para crear el RecyclerView primero tengo que seguir estos pasos:

- Añadir la dependencia 'androidx.recyclerview:recyclerview'.
- Crear el elemento RecyclerView en el recurso xml de la actividad principal.
- Crear el elemento del que ve a estar compuesto el RecyclerView. Para empezar he creado un recurso xml con un simple TextView dentro de un LinearLayout.
- Crear un adaptador que es el que se va a encargar de recibir los datos y enviárselos al RecyclerView.

Para personalizar el RecyclerView lo primero que voy a hacer es coger datos de la base de datos y mostrar al menos el nombre del contenido en los items del RecycleView. Para ello he cogido la referencia de la base de datos del objeto "series", he creado un ValueEventListener que es el que se va a encargar de leer los datos y tratarlos.

Para probar que tanto la conexión con Firebase como el RecyclerView funcionan correctamente, he pedido los datos de todas las series a Firebase, los he almacenado en una lista, he modificado el adaptador para que su constructor pida una lista y también para que por cada item de la lista muestre el nombre de la serie, he creado un adaptador pasandole la lista de series y he ejecutado la aplicación para comprobar que funciona. **(Captura 1)**



Captura 1: RecyclerView Simple

Como se puede observar en la captura, funciona como esperaba.

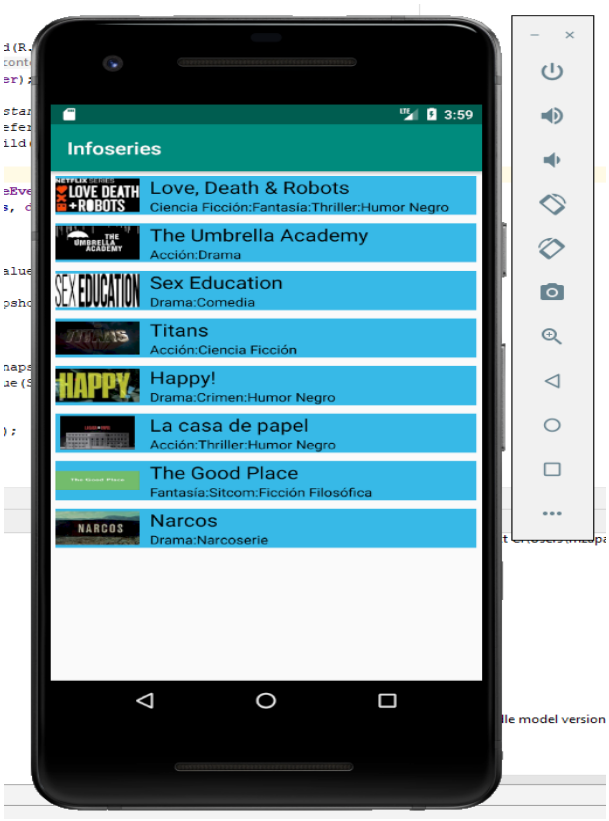
Después he pasado a modificar la vista de los items del RecyclerView para que muestre más información, concretamente, para que muestre las categorías y una imagen.

El siguiente problema que me he encontrado es ordenar los datos en orden descendiente por fecha. Firebase sólo permite ordenar por orden ascendiente, por lo que he creado una consulta que me devuelva todas las series en orden ascendiente usando la fecha y he guardado los datos en una lista, luego he utilizado el método Collections.reverse() para que la lista que estaba en orden ascendiente pasase a estar en orden descendiente.

El siguiente paso es almacenar las imágenes en Firebase y mostrarlas en la aplicación. Para ello voy a usar el storage de Firebase.

He conseguido hacer que la aplicación se comunique con el storage de Firebase y muestre imágenes usando Glide en un ImageView dentro de un item del RecyclerView. Aunque visualmente no es lo que esperaba puesto que las imágenes son de distintos tamaños y el resultado no es del todo estético, me voy a centrar más en la

funcionalidad. No me puedo permitir perder mucho tiempo en la estética del proyecto. **(Captura 2)**



Captura 2: RecyclerView personalizado

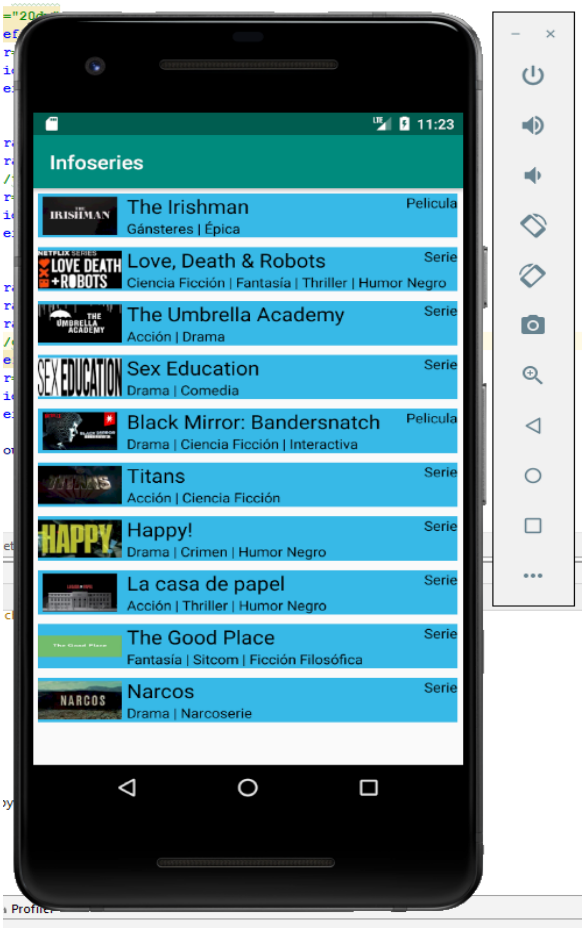
El siguiente problema que he abordado es el hecho de que de momento el RecyclerView sólo muestra las series y no las películas. Para solucionar esto he de juntar la consulta a Firebase que me devuelve las series y la consulta que me devuelve las películas en una sola lista y pasar esa lista al adaptador del RecyclerView. Esto me va a presentar dos problemas, el primero es que quiero que haya alguna diferencia visual en el RecyclerView que indique qué item es una serie y qué item es una película, el segundo problema es que voy a tener que cambiar la forma en la que ordenaba las series en orden descendiente ya que no me a valer en este caso usar el método reverse() de Collections.

He tenido que cambiar el API level mínimo para poder usar el método sort de Collections.

En el layout del item del RecyclerView he creado un TextView adicional que indique si el item que se está mostrando es una serie o una película. En el código, tanto en el ValueEventListener de la clase MainActivity como en el onBindViewHolder del MainAdapter, he tenido que hacer una comprobación del tipo de dato que recibo: si tiene el dato que recibo contiene el campo temporadas entonces es una serie, en caso contrario, es una película.

Para ordenar la lista de datos que se le pasa al MainAdapter he creado un CustomComparator que compare las fechas de los objetos de la lista, ya sean películas o series, y ordene la lista de forma decreciente en base a este campo.

He rediseñado el layout del item del RecyclerView para que los elementos estén un poco más ordenados. **(Captura 3)**



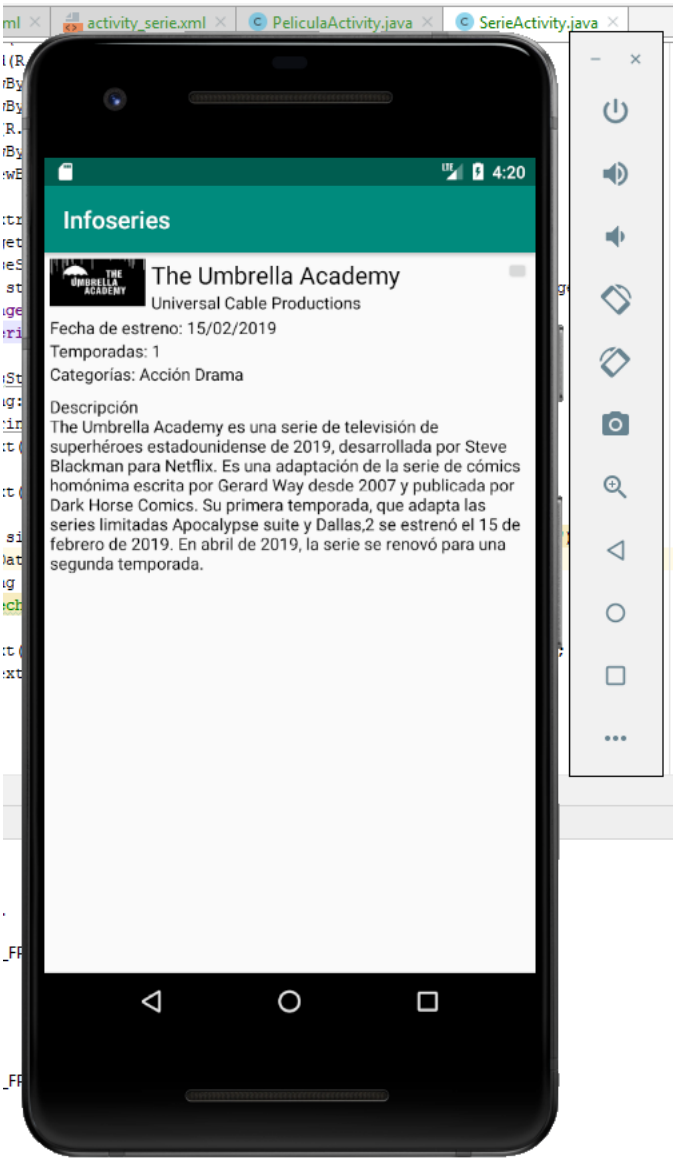
Captura 3: Ordenación de elementos en el RecyclerView

Para que cada elemento del RecyclerView cree una actividad en la que se muestre información del elemento clicado, primero he añadido un OnClickListener a los elementos del RecyclerView, de forma que se puedan clicar. En el método onClick del onClickListener se crea un intent al que se le añase un extra de nombre "Serie" o "Pelicula" (depende del dato que toque añadir) e inicie la actividad SerieActivity o PeliculaActivity, según corresponda.

He creado dos actividades, y por tanto, dos layouts distintos, porque Serie y Pelicula tienen datos distintos, pocos, pero lo suficiente como para que no pueda usar el mismo layout para representar los datos de los dos objetos.

Las actividades SerieActivity y PeliculaActivity tiene una estructura similar, primero cogemos la referencia a las vistas del layout, luego cogemos el objeto Serie o Pelicula del intent con getIntent().getSerializableExtra("MyClass") (para poder usar este método, tanto Serie como Pelicula deben implementar la interfaz Serializable), por último cogemos los datos del objeto Serie o Pelicula y se los añadimos a las vistas del layout, formateándolos primero para que sean más legibles.

Tras seguir estos pasos, el resultado de clicar en un elemento del RecyclerView es el siguiente. **(Captura 4)**



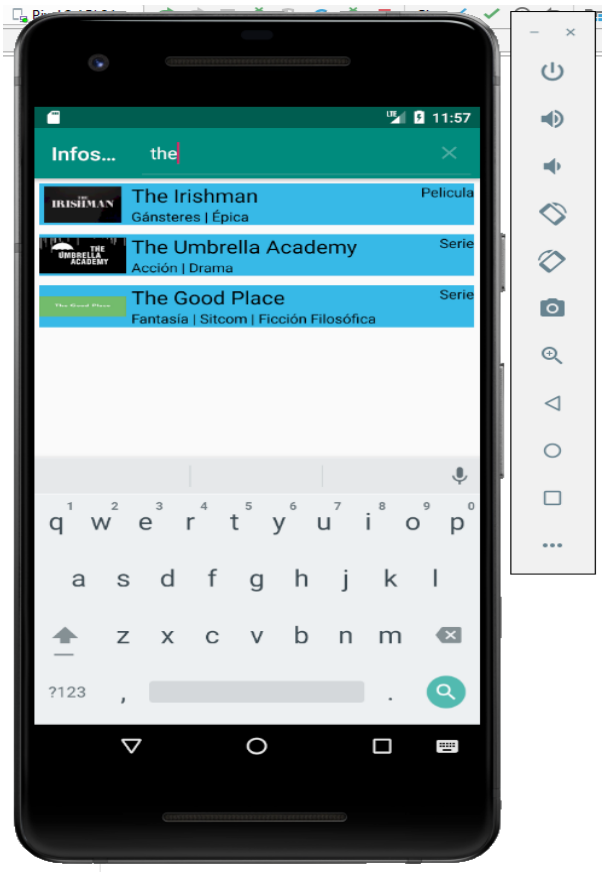
Captura 4: Actividad de un elemento del RecyclerView

5.2.3. Implementación de la función de búsqueda

Primero he creado una carpeta de recursos de tipo menu y en ella he creado el recurso menu_search.xml. En este recurso creo un item android con el icono de una lupa (icono representativo de la función de búsqueda), y que cuando es clicado crea un android.widget.SearchView. Este menu hay que inflarlo en la interfaz de la actividad principal dentro del método onCreateOptionsMenu.

La primera funcionalidad que he implementado es la de filtrar el contenido que ya está en el RecyclerView, sin necesidad de enviar la cadena de búsqueda. Para ello hay que añadir al SearchView un onQueryTextListener que recoja lo que se está escribiendo en el SearchView y se lo pase al método filtrador de contenido del adaptador del RecyclerView.

Para que el adaptador pueda filtrar contenido tiene que implementar la interfaz Filterable e implementar el método getFilter(). Par implementar el método getFilter(), que devuelve un filtro, hay que crear un Filter e implementar los métodos performFiltering (coge la cadena de búsqueda y filtra la lista de contenido) y publishResults (devuelve el resultado de filtrar). Así queda una simple búsqueda, sin enviar la cadena, sólo escribiéndola. **(Captura 5)**



Captura 5: Filtrado de elementos del RecyclerView

Después me he puesto a implementar la búsqueda. Ya que filtrar no es suficiente si quieres buscar productoras, que no están añadidas en el RecyclerView de la actividad principal. Para ello cuando se envía la cadena de búsqueda en el RecyclerView hago una query a la base de datos de Firebase para coger todas las productoras, las añado a la lista del adaptador del RecyclerView y por último filtro la lista de nuevo.

También he creado una actividad con su respectivo layout para mostrar información sobre una productora.

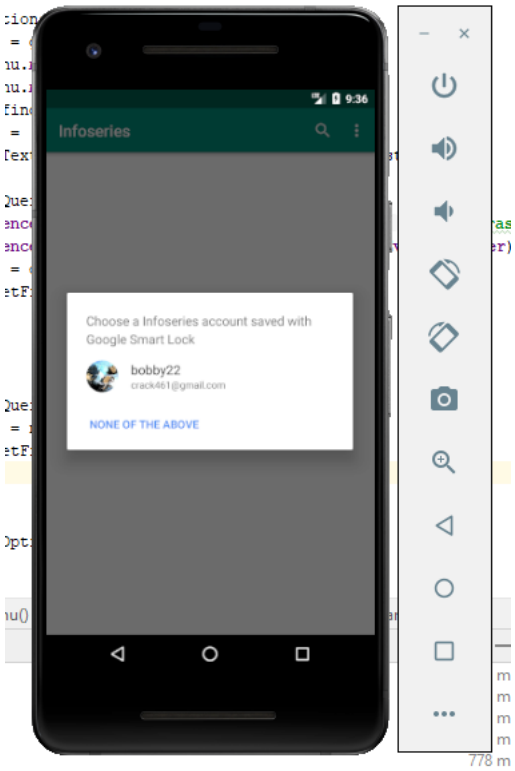
5.2.4. Implementación del sistema de autenticación de usuarios

Continuo con la implementación del sistema de autenticación de usuarios.

Las dependencias ya están añadidas, he creado las variables firebaseAuth y firebaseAuthListener. Después he creado la instancia de firebaseAuth y he creado el firebaseAuthListener en la actividad principal, rellenando el método onAuthStateChanged. Este método se ejecuta cuando el usuario se conecta o se desconecta, en él primero compruebo si el usuario está conectado, y luego, si está conectado lo desconecto, si no está conectado le muestro una actividad para que se conecte.

Al intentar ejecutar la aplicación para probar la autenticación me he encontrado con un problema, la aplicación me pide que actualice los servicios de google play para poder ejecutarse. Resolver este problema me ha llevado bastante tiempo, al final resulta que tenia que descargar unas cosas con el SDK Manager de Android Studio.

Al ejecutar la aplicación, efectivamente me pide una cuenta de google para poder usar la aplicación. **(Captura 6)**



Captura 6: Autenticación de usuario

Después he implementado una opción en el menu que permita cerrar sesión.

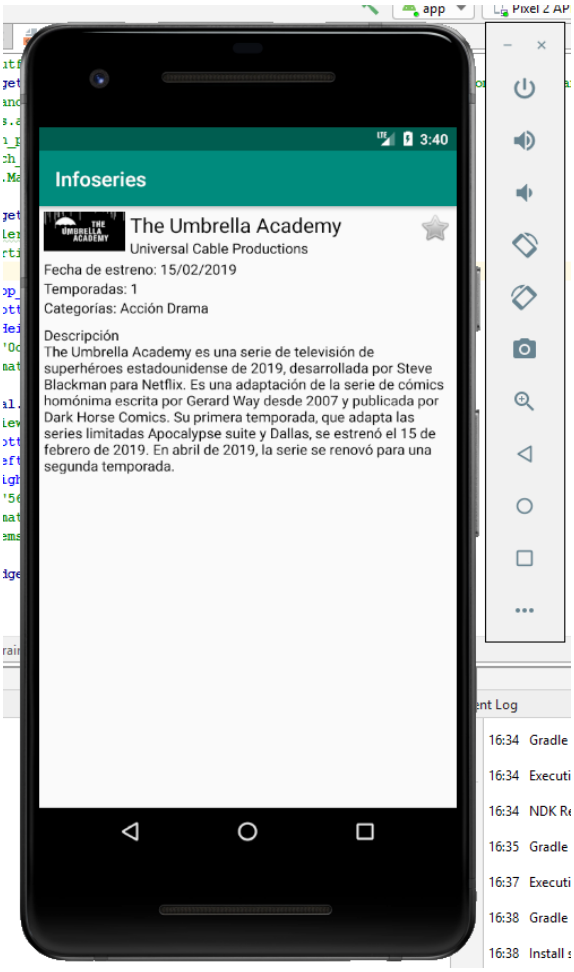
Cuando un usuario inicia sesión, me guardo el nombre de usuario y creo el recyclerview, ya que he actualizado las reglas de firebase para que no se puedan ni leer ni escribir datos si el usuario no inicia sesión primero. Cuando un usuario cierra sesión, borro los datos del recyclerView y borro el nombre de usuario.

5.2.5. Implementación de la función de marcado y la barra de navegación inferior

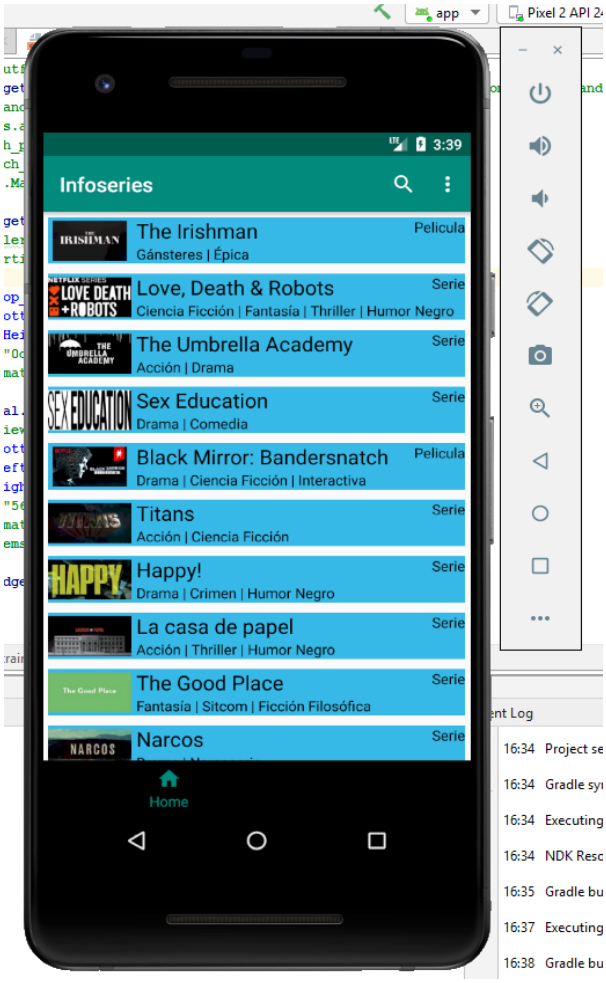
Lo siguiente que voy a implementar es un botón que permita marcar un contenido, de forma que se envia a la base de datos el usuario y el nombre del contenido marcado. Esta información es la que pretendo mostrar en un recyclerview para que el usuario pueda ver en cualquier momento el contenido que tiene marcado.

He creado un botón en las actividades de contenido (serie, película y productora) con la imagen de una estrella, aunque todavía no he implementado la funcionalidad.

También he creado un barra de navegación inferior sin funcionalidad, sólo se ve visualmente. **(Capturas 7 y 8)**

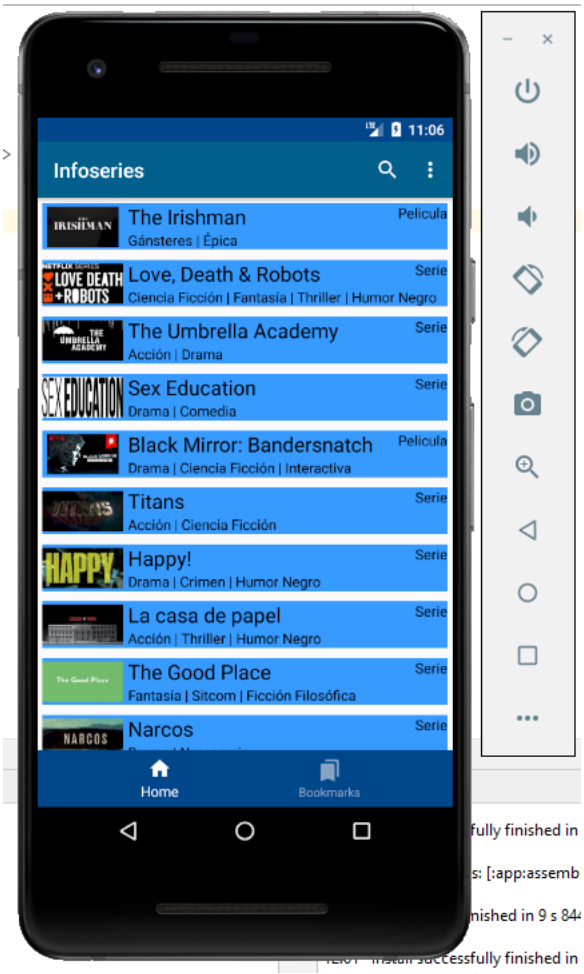


Captura 7: Botón de marcado



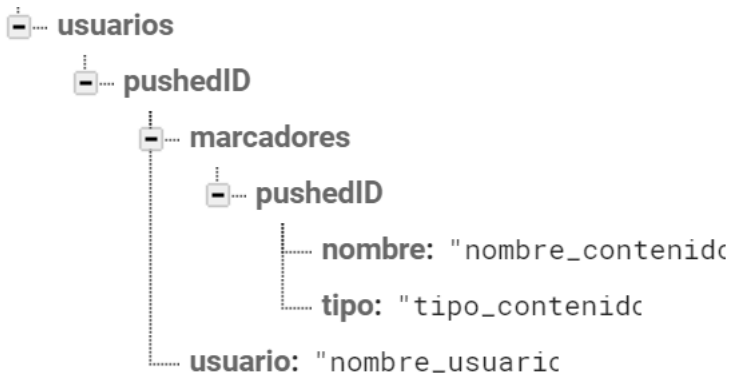
Captura 8: Barra de navegación inferior

Después de implementarle un poco de funcionalidad a la barra de navegación inferior, como que se puedan clicar los iconos y cambien de color según si están clicados o no, he estado un tiempo modificando los colores de todos los elementos de la aplicación para adaptarlos un poco más a mi gusto personal. **(Captura 9)**



Captura 9: Rediseño del layout

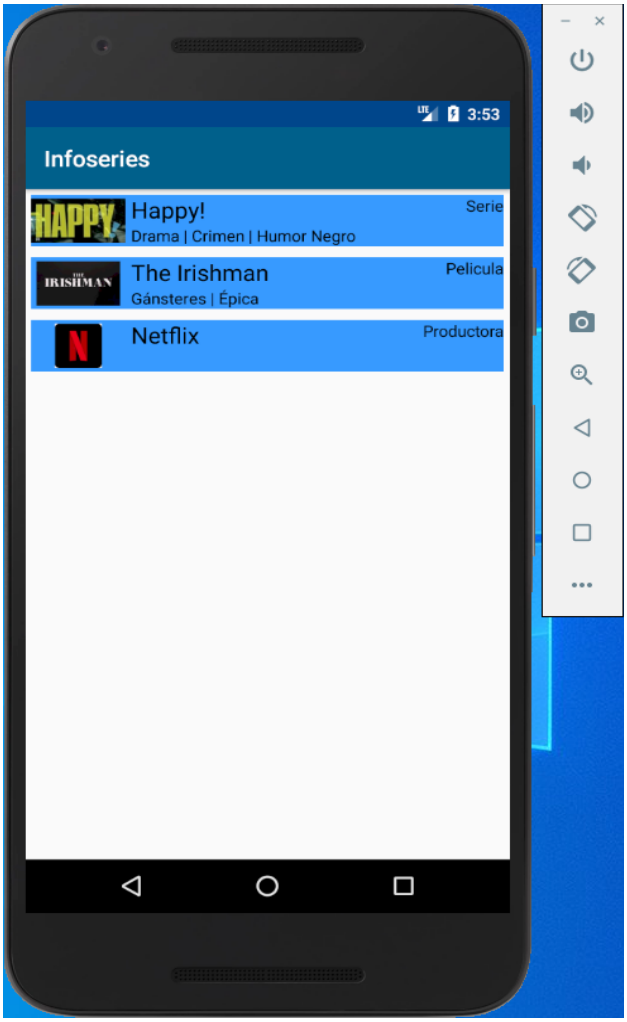
Lo siguiente que he decidido hacer es seguir con la funcionalidad de marcado, voy a hacer que cuando se clique en el botón con forma de estrella, se inserte información en la base de datos. Para ello voy a crear un nuevo nodo en la base de datos llamado "usuarios", cuyos nodos hijos serán los "pushedIDs" que genera firebase automáticamente, estos últimos tendrán como nodos hijos los campos "marcadores" y "usuario", a su vez, el campo "marcadores" estará compuesto de tantos nodos "marcador" (compuestos por "nombre" y "tipo") como contenido marque el usuario en la aplicación. La estructura quedaría así. **(Captura 10)**



Captura 10: Estructura de datos de usuarios

He creado una clase auxiliar llamada "DatabaseOperations" en un paquete Java llamado "tools", esta clase contiene métodos necesarios para implementar la funcionalidad de marcado de contenido, como estos métodos van a ser utilizados en distintas actividades, he querido implementarlos en una clase a parte para no tener mucho código duplicado. Estos métodos son: userExists, comprueba si el usuario ha marcado contenido anteriormente o es su primera vez; addUser, añade el usuario a la base de datos; isBookmarked, comprueba si el contenido ya ha sido marcado por el usuario anteriormente; addBookmark, añade el marcador en la base de datos; deleteBookmark, borra el marcador de la base de datos.

Por último, he creado una actividad en la que se listen los contenidos marcados por el usuario en un RecyclerView, y he hecho que la barra de navegación inferior lleve a esa actividad. **(Captura 11)**



Captura 11: Actividad de elementos marcados

6.- DESPLIEGUE Y PRUEBAS

Nº	ESPECIFICACIÓN DE PRUEBAS
1	Objetivo probado: función de base de datos en tiempo real. Entrada de la prueba: añadir manualmente un elemento a la base de datos. Salida de la prueba: el elemento se visualiza inmediatamente en el RecyclerView sin necesidad de reiniciar la aplicación
2	Objetivo probado: función de búsqueda. Entrada de la prueba: escribir una cadena y enviar la cadena en el SearchView de la actividad principal. Salida de la prueba: el RecyclerView se actualiza con los elementos cuyo nombre contienen la cadena escrita.
3	Objetivo probado: cerrar sesión. Entrada de la prueba: clicar la opción del menú que permite cerrar sesión. Salida de la prueba: el usuario ya no está conectado y cómo no tiene permisos para realizar consultas sobre la base de datos, el RecyclerView se vacía. La aplicación no permite realizar ninguna acción hasta que no inicie sesión.
4	Objetivo probado: actividad de elemento. Entrada de la prueba: clicar varios elementos del RecyclerView para probar los elementos de tipo Serie, Pelicula y Productora. Salida de la prueba: para todos los tipos las actividades se muestran con la información que corresponde.
5	Objetivo probado: función de marcado. Entrada de la prueba: crear dos usuarios de prueba para comprobar que los marcadores se añaden sólo al usuario que marca el elemento. Después iniciar sesión como uno de los dos usuarios y marcar varios elementos. Salida de la prueba: en la base de datos se puede observar cómo se han añadido los marcadores sólo para el usuario que marca los elementos.
6	Objetivo de la prueba: actividad de elementos marcados. Entrada de la prueba: clicar el botón de la barra de navegación inferior que lleva a la actividad de elementos marcados. Salida de la prueba: la lista de elementos marcados se muestra correctamente y concuerda con los datos de la base de datos.

7.-CONCLUSIONES

7.1.- OBJETIVOS ALCANZADOS

He alcanzado todos los objetivos propuestos:

- He creado una base de datos en Firebase.

- He creado una aplicación Android con un RecyclerView en la actividad principal.

- He creado una actividad que muestra información adicional sobre un elemento.

- He creado las operaciones GET, PUT y DELETE sobre la base de datos de Firebase. No he creado la operación UPDATE porque no hay ninguna acción que puedar hacer un usuario en la aplicación que cambie la información de un objeto en la base de datos.

- He creado un botón de búsqueda que a parte de realizar una búsqueda por nombre en la base de datos, también va filtrando la lista del RecyclerView conforme el usuario va escribiendo.

- He utilizado los métodos de FirebaseAuth para implementar un sistema de autenticación de usuarios, aunque los usuarios sólo pueden iniciar sesión mediante una cuenta Google.

- He creado un botón en las actividades de los elementos Serie, Pelicula y Productora, de forma que cuando el botón es clicado se guarda en la base de datos la información necesaria para saber qué usuario ha marcado qué contenido.

- He creado una simple barra de navegación inferior para que la aplicación sea más vistosa, esta barra tiene un botón denominado Bookmarks que lleva a la actividad que muestra el contenido marcado por el usuario.

- He creado una simple actividad en la que el usuario puede ver y acceder a su contenido marcado, reutilizando el RecyclerView y el MainAdapter creados para la actividad principal.

7.2.- CONCLUSIONES DEL TRABAJO

Aunque veo posibles mejoras y la interfaz de la aplicación no resulta del todo atractiva, creo que he realizado un buen proyecto en relación al tiempo dedicado, ya que al tener que compaginar las prácticas en empresa y la realización de este proyecto, no he podido dedicarle tanto como me habría gustado. En cualquier caso he conseguido realizar de una forma u otra todos los objetivos que me propuse antes de empezar el proyecto.

En cuanto a las dificultades que he encontrado a lo largo de la realización del proyecto, destacaría las siguientes:

- Ha sido difícil decidir cómo estructurar y presentar la información en la aplicación, ya que no se me da muy bien el diseño de interfaces.

- También he tardado en aprender a cómo trabajar con una base de datos no relacional y con datos almacenados en objetos JSON, es la primera vez que trabajo con este tipo de estructuración de los datos.

- Una de las mayores dificultades que ha presentado trabajar con Firebase en Android es que, en mayor parte, los métodos utilizados para trabajar con Firebase son asíncronos, aportando complejidad a la implementación de la mayoría de las tareas.

- Por lo general, ya sabía varias cosas sobre programación en Android Studio antes de empezar con este proyecto, pero al utilizar Firebase he tenido que aprender otras muchas interacciónes que no conocía.

7.3.- VÍAS FUTURAS

Se me ocurren las siguientes mejoras para la aplicación:

- Los items del RecyclerView podrían ser CardViews en vez de simples LinearLayouts, eso le daría un toque más profesional a la interfaz.

- Debería de haber más opciones a la hora de iniciar sesión en la aplicación, incluso la posibilidad de poder usarse como usuario anónimo mientras no sea necesario el almacenamiento de ningún dato de usuario.

- La barra de navegación inferior podría trabajar con Fragments en vez de enviar al usuario a otra actividad, eso haría la transición mucho más intuitiva.

8.- GLOSARIO

9.- BIBLIOGRAFÍA

1. Course Firebase in a Weekend (Android): https://classroom.udacity.com/courses/ud0352

2. Documentación Firebase: https://firebase.google.com/docs/

3. Documentación Android (para dudas varias): https://developer.android.com/docs

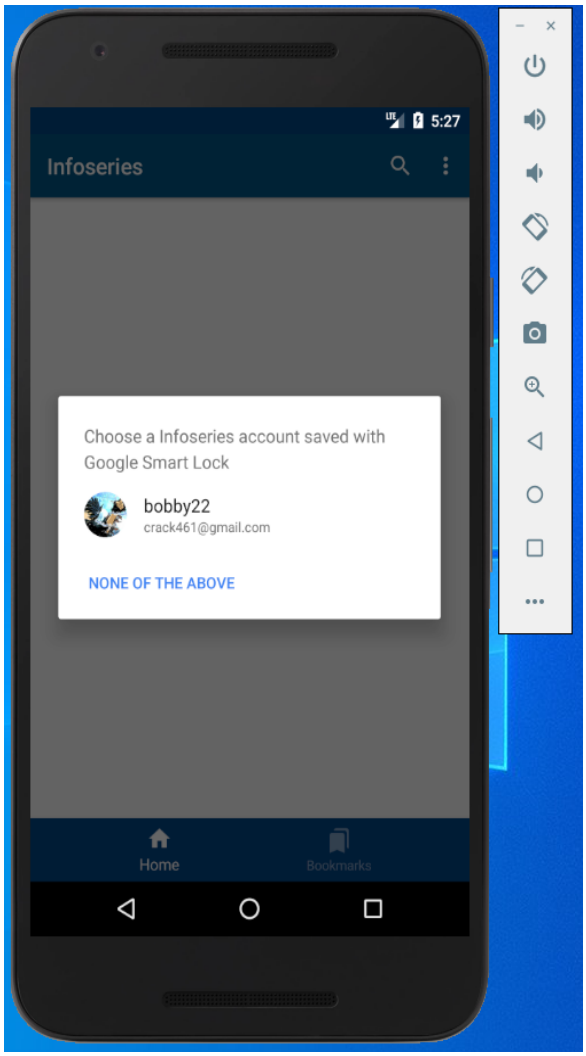
10.- ANEXOS

10.1.- MANUAL DE INSTALACIÓN.

Para instalar la aplicación debes descargar el archivo app-debug.apk del repositorio github: https://github.com/22bobby22/proyecto_fp. Después sólo debes buscar el archivo en el dispositivo, clicarlo y darle a instalar.

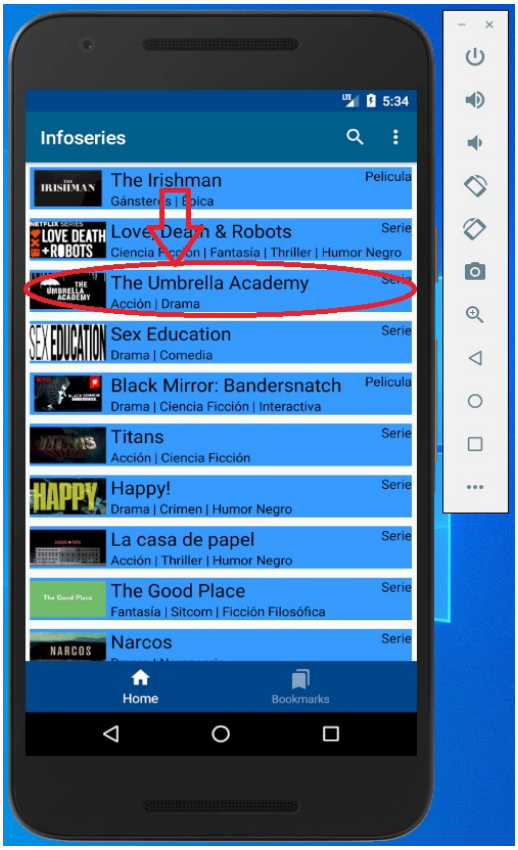
10.2.- MANUAL DE USUARIO

Cuando inicias la aplicación, lo primero que te pedirá es que inicies sesión con una cuenta Google:

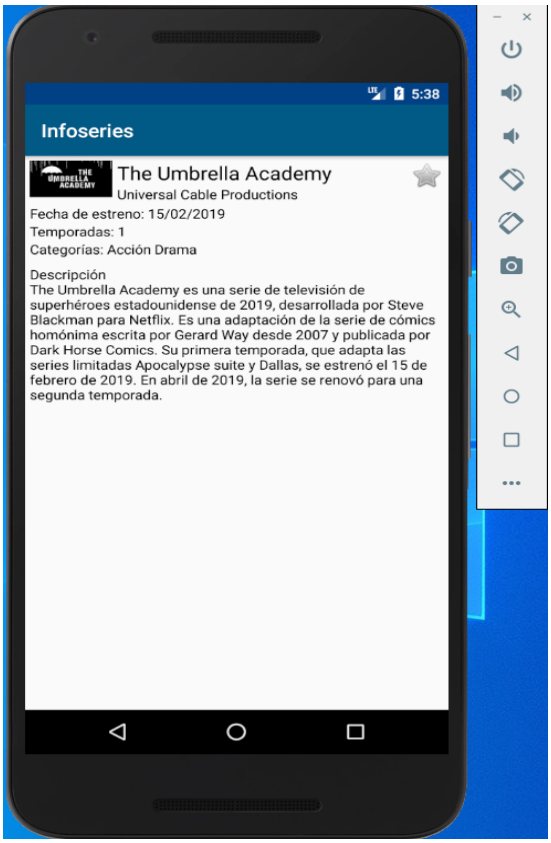


Manual de usuario 1: Inicio de sesión

Una vez has iniciado sesión puedes clicar en cualquier elemento de la lista que aparece para obtener más información sobre él:

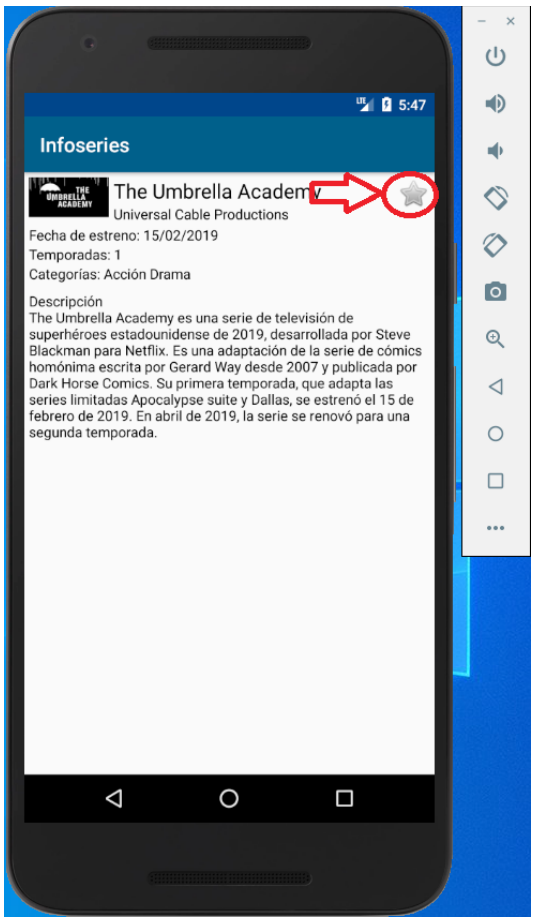


Manual de usuario 2: Resaltar un elemento

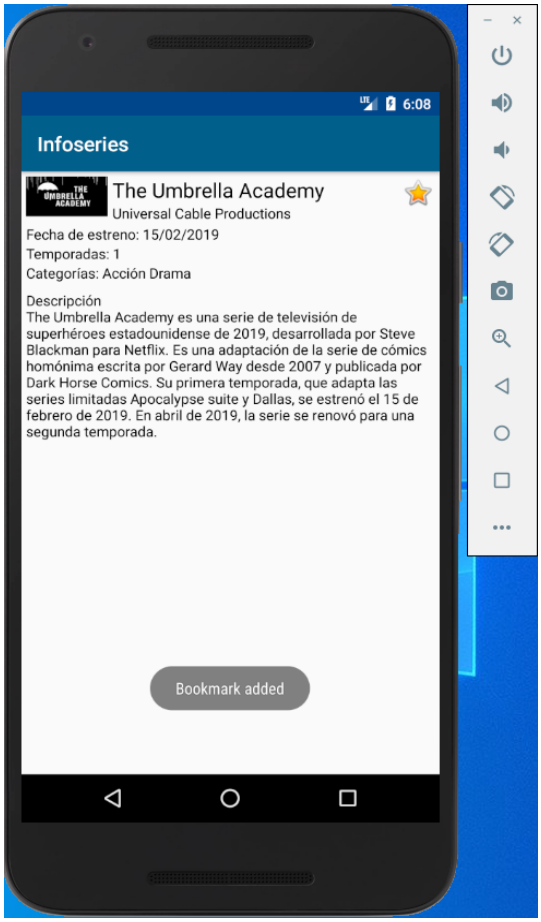


Manual de usuario 3: Actividad de un elemento

En las actividades de los elementos puedes clicar el botón con forma de estrella para marcar el elemento:

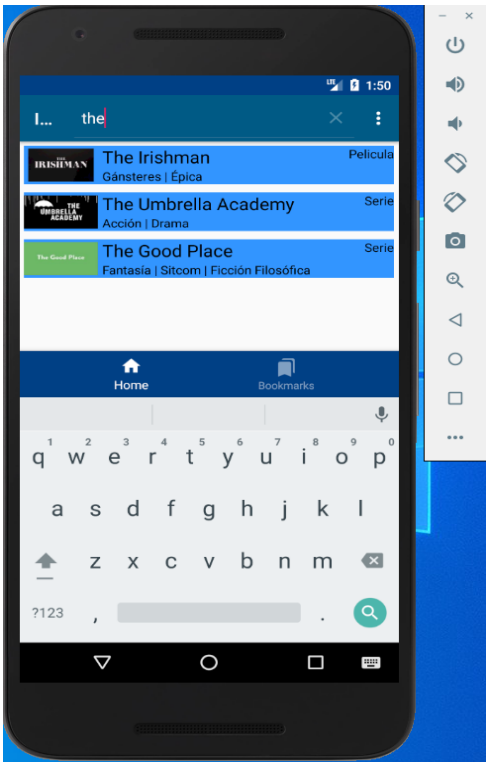


Manual de usuario 4: Resaltar botón de marcado



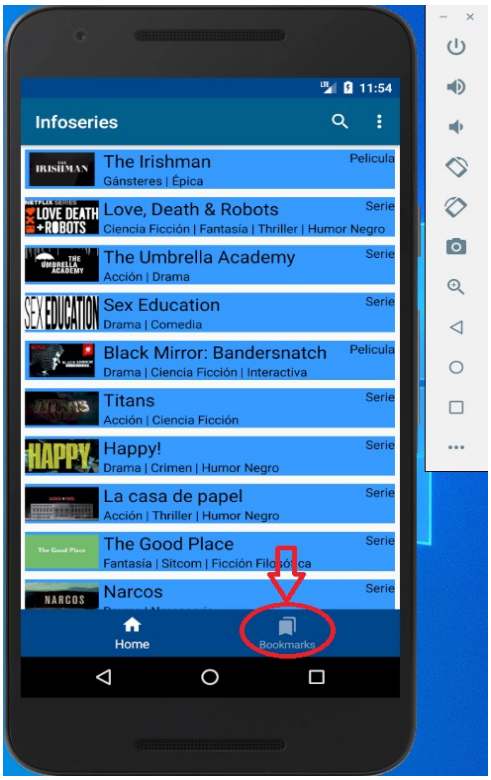
Manual de usuario 5: Botón de marcado presionado

En la actividad principal también se puede realizar una búsqueda por nombre:

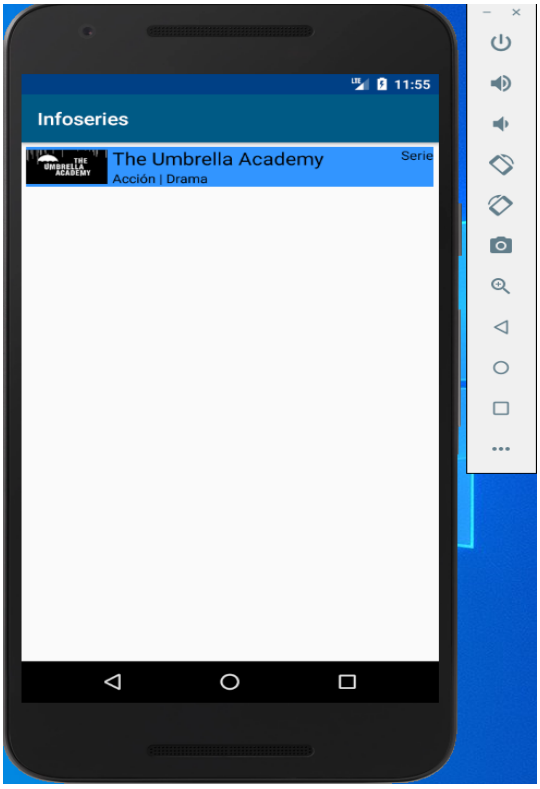


Manual de usuario 6: Búsqueda por nombre

En la actividad principal también se puede ir a una actividad en la que se muestran los elementos marcados:

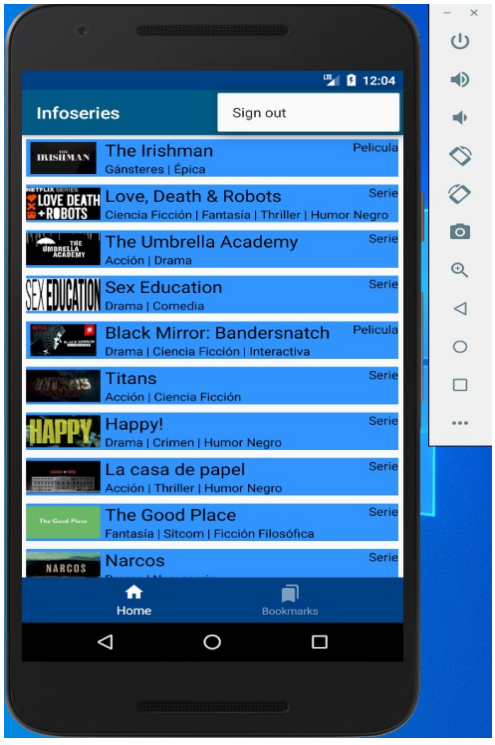


Manual de usuario 7: Resaltar botón de actividad de elementos marcados



Manual de usuario 8: Actividad de elementos marcados

Por último el usuario también puede cerrar sesión usando la opción del menú de la actividad principal:



Manual de usuario 9: Opción del menu para cerrar sesión