# MyFunkoApp Una aplicación móvil para coleccionistas de funko pop

Por
Alejandro Casado Benito


UNIVERSIDAD COMPLUTENSE
MADRID

Trabajo de fin de grado del Grado en Ingeniería del Software
Facultad de informática

*Dirigido por*
José Ignacio Hidalgo Pérez

**MyFunkoApp A mobile app for funko pop collectors**

Madrid, 2021–2022

# MyFunkoApp

## Una aplicación móvil para coleccionistas de funko pop

*Memoria que se presenta para el Trabajo de Fin de Grado*

**Alejandro Casado Benito**

*Dirigido por:*

**José Ignacio Hidalgo Pérez**

**Departamento de arquitectura de computadores y automática**
**Facultad de Informática**
**Universidad Complutense de Madrid**

**Madrid, 2021**

# Agradecimientos

Antes de comenzar con la explicación del desarrollo de la aplicación quiero añadir algunos agradecimientos.

A José Ignacio Hidalgo Pérez, por haber sido el director de este trabajo, haberme aconsejado y guiado en todo el proceso de desarrollo, dándome total libertad para tomar decisiones, y dedicándome su tiempo para llevar acabo las reuniones de seguimiento. Ha sido todo un placer y un honor haber podido trabajar junto a él en este proceso, no solo porque es un buen tutor sino que también es una excelente persona y le tengo mucho cariño.

A mi pareja Silvia Zolle Jurado por haberme apoyado, ayudado y aguantado durante todo el camino de la carrera y de este proyecto, con ella a mi lado todo ha sido más fácil.

A mi familia y amigos por apoyarme durante este camino largo y costoso pero muy gratificante como han sido los años de la carrera y la elaboración del TFG.

A los profesores e integrantes de la Facultad de Informática por formarme y entusiasmarme por este mundo tan bonito como es el desarrollar software, además de ayudarme a evolucionar como persona de una manera académica, personal y en futuro, profesional.

*Dedicado a mi Padre que, aunque ya no esté,*
*siempre estará conmigo. A mi familia, pareja y amigos.*

# Resumen

Los Funko Pops son muñecos cabezudos de vinilo que imitan en forma a una persona o personaje. Desde el año 2013 en adelante en España ha habido un aumento considerable en el interés por hacerse con estos Funko Pop. Tener una colección completa es un reto complicado pero divertido incluso a veces muy caro. En este trabajo de fin de grado se ha desarrollado una aplicación móvil para gestionar las colecciones de estos muñecos que hemos bautizado como *MyFunkoAppp*. Se trata de una aplicación móvil que ayuda al usuario a tener una buena organización de su colección personal. Además cuenta con diversas funcionalidades que la hacen más atractiva para el usuario. Cuenta con un gran catálogo de FunkoPop, permite la posibilidad de hacer una lista de deseos, incorpora una comunidad en la que poder añadir a amigos, conocer gente nueva y ver sus colecciones, ver la lista de deseos e incluso la información personal de otros usuarios amigos, permite la inclusión en la colección y la búsqueda mediante escaneo de código de barras y búsqueda por voz. Está concebida como una aplicación completamente comercial, que permite el acceso a plataformas de venta y que podría eventualmente utilizarse como aplicación de venta on-line en un entorno real.

**Palabras clave**: Funko, MVVM, Android, aplicación, actividad, fragmento, IU

# Abstract

Funko Pops are vinyl figures that imitate the shape of a person or character. Since 2013, in Spain there has been a considerable increase in the interest in getting hold of these Funko Pops. Having a complete collection is a complicated but fun challenge and sometimes even very expensive. In this final degree project we have developed a mobile application to manage the collections of these dolls that we named MyFunkoAppp. It is a mobile application that helps the user to have a good organisation of his personal collection. It also has several functionalities that make it more attractive for the user. It has a large FunkoPop catalogue, allows the possibility of making a wish list, incorporates a community in which you can add friends, meet new people and see their collections, see the wish list and even the personal information of other user friends, allows the inclusion in the collection and the search by barcode scanning and voice search. It is conceived as a fully commercial application, allowing access to sales platforms and could eventually be used as an on-line sales application in a real environment.

**Keywords**: Funko, MVVM, Android, application, activity, fragment, UI

# Índice general

**Página**

# Capítulo 1

# Introducción

El coleccionismo es una afición que consiste en invertir tu tiempo de ocio en recolectar objetos, ya sea por su valor monetario o simbólico. Este ha sido muy cambiante a lo largo de los años. El primer registro que se tiene de una colección son unas tablillas de barro del *rey asirio Asurbanipal*, unos siglos más adelante los filósofos reunieron colecciones de libros para mejorar sus enseñanzas, sin embargo los *reyes helenísticos* fueron quienes crearon las bibliotecas más grandes. Finalmente, con el paso de los siglos ha ido evolucionando de tal manera que se puede coleccionar cualquier cosa. [1]

Todos haciendo memoria podemos recordar nuestras primeras colecciones, ya fuera con tazos, pegatinas, sellos, chapas, dedales... Queriendo preservar recuerdos de viajes, situaciones o momentos únicos de manera tangible.

En mi caso personal, mi primera colección surgió en el patio del colegio jugando con mis compañeros. Los tazos de Pokémon, que en aquel entonces estaban de moda, todos queríamos conseguir los 151 que existían, peleabas por ellos o intercambiabas los repetidos para poder agrandar y conseguir los huecos que faltaban, en esa época no era consciente que hoy en día esa colección tendría un valor incalculable para mí.

El valor económico de las colecciones se puede regir por diferentes variables: antigüedad, exclusividad y auge (entre otras). Mi proyecto de TFG se centra en esta última y es con la llegada de los muñecos cabezudos conocidos como Funko Pop.

Para poneros en contexto el origen de Funko se remonta en 1998, cuando un diseñador gráfico de camisetas y coleccionista *Mike Becker*, quería hacerse con una figura de un

famoso restaurante, al ver el precio elevado pensó que por el mismo precio podría reproducir su copia. Al poco tiempo junto a *Rob Schwartz* y *Sean Wilkinson*, acordaron fundar Funko, aunque el concepto inicial era diferente al actual se mantenía cierta similitud y la primera figura que fue diseñada era un hombre con un ordenador en la cabeza. Años más tarde, Mike vendió Funko a *Brian Mariotti*, que dio con la tecla perfecta para el boom de Funko y su crecimiento actual.[2]

Hoy en día hay mucha gente interesada en coleccionar estos muñecos, no solo porque son bonitos y quedan bien en las estanterías, sino porque pueden alcanzar cifras muy elevadas en el mercado, lo que conlleva la apertura de un negocio muy poderoso y atractivo. Por eso como amante y coleccionista de Funko Pop he tenido la idea de recopilar en una sola aplicación el poder tener y compartir tu colección o lista de deseos a tus amigos para la posibilidad de intercambiar estas figuras o incluso tener la opción de poder comprarlos.

Hace ya tres años *Funko* lanzó al mercado una aplicación gratuita para móviles *Andoid* e *iOS*. En ella puedes organizar todos tus funkos, ya que cuenta con la licencia oficial de distribución de estos muñecos con más de 17.000 productos diferentes, además de poder tener tu lista de deseos para futuras compras. También puedes realizar un seguimiento de lo que cuesta tu colección, lista de deseos o cualquier funko ya que la compañía se ha asociado con *Pop Price Guide* [3] para tener la información al dedillo.

Las ventajas que posee la aplicación son:

- Poder incorporar a tu colección o lista de deseos los funkos a través del escaneado de código de barras.
- Recientemente han añadido el poder compartir listas creadas con otros usuarios.

Pero para el público español o latino tiene una desventaja muy grande y es que la aplicación está orientada para los estadounidenses, no solo porque esté en inglés, sino que las noticias también son de allí.

Como amante de estos cabezones y futuro desarrollador de software no podía dejar de intentar pasar esta oportunidad y crear una app que involucra todas las ventajas que tiene esta aplicación, sino que, además añadirle muchas de las peticiones que he observado y

se pueden seguir viendo en los comentarios de los usuarios que descargan esta aplicación. Por ejemplo, que esté español, que puedas tener amigos para poder comunicarte y poder conocer gente nueva con la misma afición, poder comprar en la propia app o incluso poner información sobre cómo detectar si el funko es falso, etc. ( Podéis observar en la figura 1.1 una parte de mi colección de funkos )


Figura 1.1: Una parte de mi colección

## 1.1. Objetivos

El objetivo principal de este proyecto es diseñar e implementar una aplicación para coleccionistas de Funkos, *MyFunkoApp*, que permita intercambiar información con otros

usuarios, crear una comunidad virtual, comprar, vender e intercambiar muñecos de la colección. Deseo aportar una aplicación útil para todo coleccionista de Funko Pop, en el que perciba resguardada su colección y al mismo tiempo la posibilidad de que crezca.

Los objetivos secundarios del proyecto, derivados del objetivo principal son:

- Crear una aplicación escalable, con el fin de que en el futuro se pueda seguir desarrollando y actualizando.
- Diseñar e integrar una arquitectura bajo el paradigma de modelo-vista modelo-devista (MVVM, en inglés Model View View Model) de aplicaciones móviles.
- Diseñar e integrar una interfaz de usuario cómoda y llamativa para una agradable experiencia.
- Diseñar e integrar funcionalidades atractivas para el usuario.

## 1.2. Plan de Trabajo

El plan de trabajo para el desarrollo de la aplicación ha sido de la siguiente manera:

Primero se ha diseñado un mockup de la aplicación, principalmente para poder ver de manera global cuál iba ser su funcionamiento. Con este se han alcanzado unos requisitos mínimos para poder empezar el desarrollo. Después se ha continuado con el diseño de las ventanas principales de aplicación, home (figura 4.19), lista de deseos (figura 4.22), colección (figura 4.24) y amigos (figura 4.26). Una vez finalizado el diseño de las vistas anteriores se comenzó con las funcionalidades de estas. Cuando se realizaron se consiguió una base sólida de la aplicación y así ir haciéndola crecer poco a poco.

Se generó una lista con nuevas funcionalidades para integrar en la aplicación y se llevaron a cabo solo las que dábamos el visto bueno mi tutor y yo en las reuniones que teníamos. Estas funcionalidades se desarrollaron de manera independiente, hasta que no se terminara una, no se empezaba otra. Algunas de estas actividades son: la búsqueda por escáner (figura 4.39) o búsqueda por voz (figura 4.40), catálogo de funkos (figura 4.36), detalle funko (figura 4.42), entre muchas otras que podréis ver más adelante.

Finalmente se probó todas las funciones de manera precisa para ver el correcto compor-

tamiento de ellas.

## 1.3. Organización de la memoria

A continuación, podemos encontrar la organización de la memoria:

En el Capítulo uno, encontramos una introducción sobre el mundo del coleccionismo y concretamente del mundo de los Funko Pop!, también veréis la aplicación oficial de funkos con sus ventajas e inconvenientes y por qué he decidido realizar esta app. Finalmente veréis los objetivos y el plan de trabajo.

Capítulo dos, es lo mismo que el capítulo uno, pero en inglés para seguir con la normativa del TFG.

Capítulo tres, se comentan las herramientas que se han utilizado para llevar a cabo el proyecto y la arquitectura utilizada.

Capítulo cuatro, se describe detalladamente la manera de obtener los datos y las funcionalidades que tiene la aplicación.

Capítulo cinco, se comentan las conclusiones del TFG

Capítulo seis, es lo mismo que el capítulo cinco, pero en inglés para seguir con la normativa del TFG.

Capítulo siete, se comentan las ideas para llevar a cabo en un futuro si el proyecto saliese adelante.

# Capítulo 2

# Introduction

Collecting is a hobby that involves spending your leisure time collecting objects, either for their monetary or symbolic value. It has changed a lot over the years. The first record of a collection are some clay tablets of the Assyrian king Assurbanipal, a few centuries later philosophers gathered collections of books to improve their teachings, however the Hellenistic kings were the ones who created the biggest libraries. Finally, over the centuries it has evolved in such a way that anything can be collected. [1]

All of us can remember our first collections, whether it was with cards, stickers, stamps, badges, thimbles... Wanting to preserve memories of trips, situations or unique moments in a tangible way.

In my personal case, my first collection came about in the school playground playing with my classmates. The Pokémon decks, which at that time were in fashion, we all wanted to get the 151 that existed, you fought for them or exchanged the repeated ones to enlarge and get the missing holes, at that time I was not aware that today that collection would have an incalculable value for me.

The economic value of collections can be governed by different variables: age, exclusivity and boom (among others). My dissertation project focuses on the latter and it is with the advent of the Funko Pop figurines.

To put you in context, the origin of Funko dates back to 1998, when a graphic designer of T-shirts and collector, *Mike Becker*, wanted to get hold of a figure of a famous restaurant, and seeing the high price he thought that for the same price he could reproduce his copy.

Soon after, together with Rob Schwartz and Sean Wilkinson, they agreed to found Funko, although the initial concept was different from the current one, they kept some similarity and the first figure that was designed was a man with a computer on his head. Years later, Mike sold Funko to Brian Mariotti, who hit the perfect key for Funko's boom and its current growth.

Nowadays there are many people interested in collecting these dolls, not only because they are beautiful and look good on the shelves, but also because they can reach very high figures in the market, which leads to the opening of a very powerful and attractive business. That's why as a lover and collector of Funko Pop I had the idea of compiling in a single application to have and share your collection or wish list to your friends for the possibility of exchanging these figures or even have the option to be able to buy them.

Three years ago, *Funko* launched a free application for mobile phones *Andoid* and *iOS*. In which you can organise all your funkos, as it has the official distribution license of these dolls with more than 17,000 different products, as well as being able to have your wish list for future purchases. You can also keep track of how much your collection, wish list or any funko costs as the company has partnered with *Pop Price Guide* to have the information up to the minute.

The benefits of the app are:

- The ability to add funkos to your collection or wish list via barcode scanning.
- They have recently added the ability to share lists created with other users.

But for the Spanish or Latin audience it has a very big disadvantage and that is that the application is oriented for Americans, not only because it is in English, but also because the news are also from there.

As a lover of these big heads and future software developer I could not miss this opportunity and try to create an app that involves all the advantages that this application has, but also add many of the requests that I have observed and can still be seen in the comments of users who download this application. As this in Spanish, you can have friends to communicate and to meet new people with the same hobby, to buy in the app

itself or even put information on how to detect if the funko is fake, etc.. (You can see in figure 2.1 a part of my collection of funkos.


Figura 2.1: Part of my collection

## 2.1. Objectives

The main objective of this project is to design and implement an application for Funko collectors, *MyFunkoApp*, which allows them to exchange information with other users, create a virtual community, buy, sell and exchange dolls from their collection. We want to provide a useful application for all Funko Pop collectors, in which they can feel that their collection is protected and at the same time have the possibility of growing it.

The secondary objectives of the project, derived from the main objective are:

- To create a scalable application, so that it can be further developed and upgraded in the future.
- Design and integrate an architecture under the Model-View-View Model (MVVM) paradigm of mobile applications.
- Design and integrate a comfortable and eye-catching user interface for a pleasant experience.
- Design and integrate attractive functionalities for the user.

## 2.2. Workplan

The work plan for the development of the application has been as follows:

First, a mockup of the application has been designed, mainly to be able to see in a global way how it would work. With this, the minimum requirements have been reached to be able to start the development. Then we continued with the design of the main windows of the application, home figure 4.19, wish list figure 4.22, collection figure 4.24 and friends figure 4.26. Once the design of the previous views was finished, we started with their functionalities. Once they were finished, a solid base of the application was achieved and so we could make it grow little by little.

A list of new functionalities to integrate into the application was generated and only those that were approved by my tutor and myself in the meetings we had were implemented. These functionalities were developed independently, until one was finished, another was not started. Some of these activities are: search by scanner figure 4.39 or search by voice

figure 4.40, funkos catalogue figure 4.36, funko detail figure 4.42, among many others that you can see later on.

Finally, all the functions were tested in a precise way to see the correct behaviour of them.

## 2.3. Memory organisation

Below you can find the organization of the memory:

Chapter one, you will find an introduction about the world of collecting and specifically the world of Funko Pop!, you will also see the official funkos app with its advantages and disadvantages and why I decided to make this app. Finally you will see the objectives and the work plan.

Chapter two, is the same as chapter one but in English to follow the TFG regulations.

Chapter three, discusses the tools that have been used to carry out the project and the architecture used.

Chapter four describes in detail the way in which the data is obtained and the functionalities of the application.

Chapter five, the conclusions about the TFG are discussed.

Chapter six, the same as chapter five but in English to follow the TFG regulations.

Chapter seven, discusses the ideas to be carried out in the future if the project goes ahead.

# Capítulo 3

# Herramientas y arquitectura

## 3.1. Herramientas

En este capítulo se habla de las herramientas que se han usado a lo largo del desarrollo del proyecto y han sido fundamentales para llevar acabo el TFG.

### 3.1.1. Trello

Trello es una herramienta para gestionar proyectos a través de un tablero virtual. Puede ser usada de forma colaborativa para el reparto de tareas, pero para el desarrollo del TFG se ha utilizado para organizar las tareas y el tutor viera el progreso.

### 3.1.2. Android Studio

Es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android. Además del potente editor de código ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android.
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla.

Ha sido la herramienta principal de mi proyecto. Con él se ha podido desarrollar y probar la aplicación. Como emulador se ha usado los que te vienen por defecto y el teléfono móvil

personal.

### 3.1.3. Firebase

Es una plataforma creada por Google que nos brinda diferentes herramientas para facilitar el desarrollo de aplicaciones tanto en móviles Android e iOS. Nos proporciona diversos servicios, como pueden ser:

- Autenticación, mediante correo electrónico, Google, etc.
- Servicios de almacenamiento en la nube
- Servicios de hosting
- Servicios de BBDD
- Detección de errores

y muchos otros más. En el proyecto se han utilizado varios servicios como la autenticación, la BBDD y no se descarta utilizar en un futuro más.

Se ha elegido Firebase porque es muy potente y cuenta con una documentación muy buena que hace que la integración en el proyecto sea sencilla y la corrección de errores sea fácil de manejar.

### 3.1.4. Notepad++

Es un editor de texto y código totalmente gratuito que soporta varios lenguajes de programación.

Se ha utilizado porque llevo años usándolo y me parece muy sencillo de utilizar y cumple con la función perfectamente.

### 3.1.5. Github

Es una plataforma de alojamiento que te da la posibilidad de crear repositorios de código y mantenerlos en la nube de manera segura, poder tenerlos de manera privada o pública.

A lo largo de la carrera lo he utilizado en varias ocasiones y por eso se ha elegido para subir el proyecto terminado a esta plataforma.

### 3.1.6. Google AdMob

Es una herramienta que ayuda a los desarrolladores a ganar dinero mediante anuncios de gran calidad en las aplicaciones móviles. Ya que contiene la mayor demanda global de los anunciantes, tiene varios formatos innovadores de anuncios y una integración rápida y sencilla. Se ha utilizado para monetizar la app en la que puedes ver más detalle en la sección 4.3.

## 3.2. Arquitectura

A lo largo de la carrera he desarrollado varios proyectos usando metodologías tradicionales como cascada y en otros, metodologías ágiles, ambas son correctas y creo que se podrían adaptar al proyecto, pero creo que la que más se adapta sería una combinación de ambas, conocida como metodología híbrida.

En el proyecto se necesita tener una planificación sólida para poder buscar soluciones o adaptaciones rápidas a cualquier problema que surja en mitad de desarrollo por estas razones la metodología híbrida es la que se va a utilizar.

Para el desarrollo de la aplicación móvil se ha utilizado el patrón modelo-vista-modelo de vista (model-view-viewModel) de aquí en adelante será mencionado como MVVM. Se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuarios de la lógica de negocio. Podéis observar su diagrama en la figura 3.1

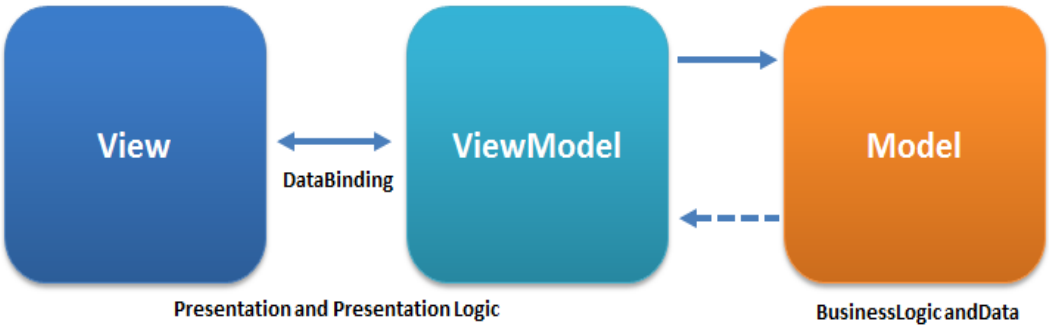



Figura 3.1: Patrón MVVM

Es actualmente uno de los patrones más utilizado para la creación de aplicaciones debido a su fácil implementación y su gran funcionamiento. Las partes de este patrón son las siguientes: modelo, vista, vista-modelo.

### 3.2.1. Modelo

El modelo es responsable de la abstracción de las fuentes de datos. Model y ViewModel trabajan juntos para obtener y guardar los datos.

### 3.2.2. Vista

La vista es la estructura, el diseño y la apariencia de lo que el usuario ve en la pantalla. Muestra una representación del modelo y recibe la interacción del usuario con la vista (clicks en la pantalla, entrada de texto, etc) y reenvía el manejo de estos al modelo de vista a través de la vinculación de datos.

### 3.2.3. Vista-Modelo

Es una abstracción de la vista que expone propiedades y comandos públicos. En lugar de controlador como en el patrón MVC (modelo-vista-controlador) tiene atributos denominados *Live-data* que sirven de comunicadores entre la vista y el modelo.

Por tanto, el funcionamiento de este patrón sería el usuario hace una acción sobre el teléfono y la vista notifica a la vista-modelo que a su vez se comunica con el modelo. La vista mientras, tiene los denominados observadores, que están escuchando si ha ocurrido algún cambio para mostrarlo, con lo mencionado anteriormente variables *Live-data*.

# Capítulo 4

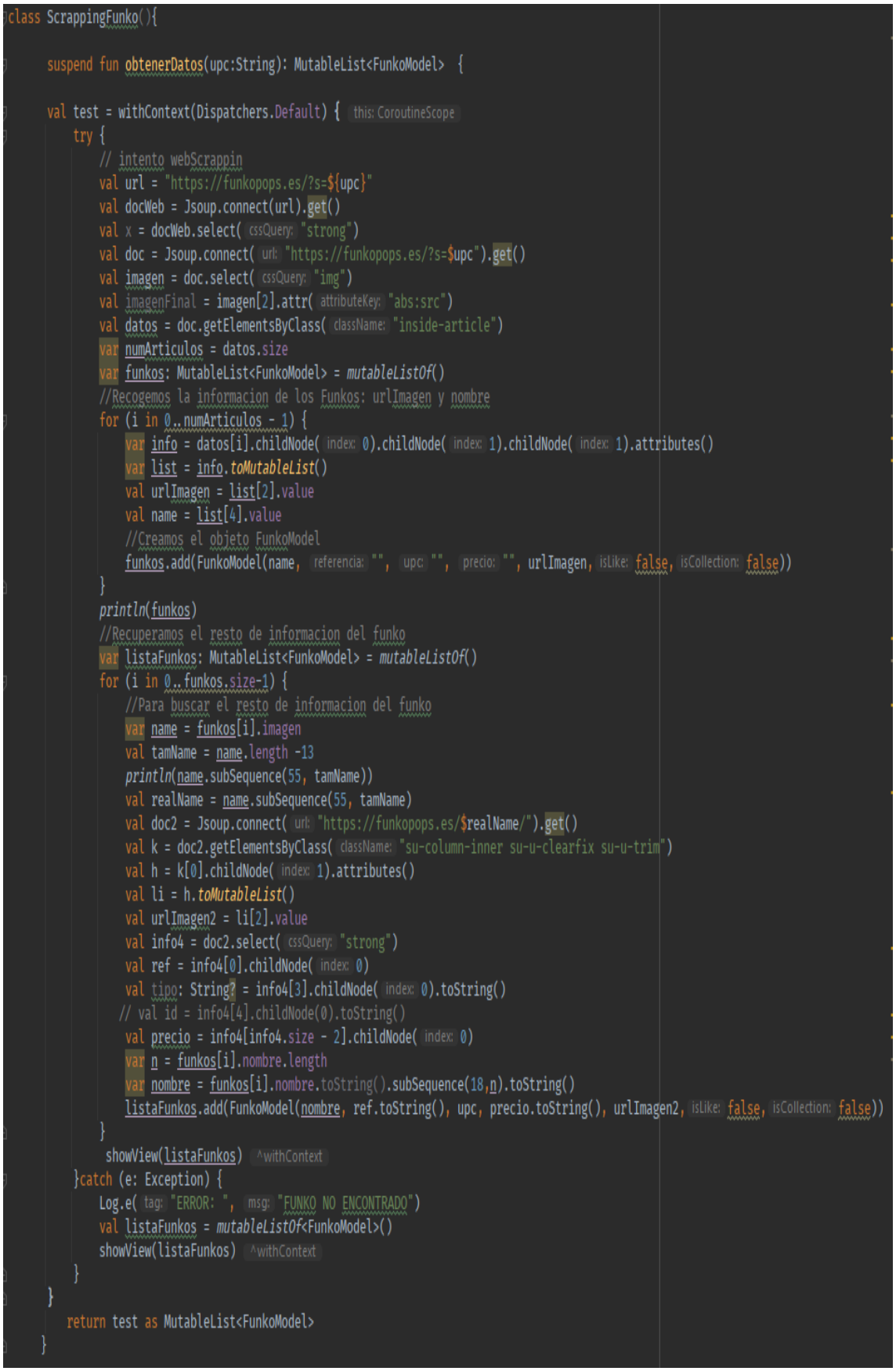
# MyFunkoApp

En este apartado encontraréis todas las funcionalidades de la aplicación explicadas detalladamente. Primero estarán las librerías externas que se han utilizado y a continuación veréis detalladamente las funcionalidades y su interfaz de usuario.

Hay que destacar las dos formas de obtener los datos de los funkos para llenar la BBDD, es decir, toda la información de la app. En primer lugar, se implementó una técnica conocida como web scrapping que sirve para extraer de manera automatizada la información de una página o páginas web. Hoy en día esta técnica es muy utilizada para conseguir grandes cantidades de información (Big data) prácticamente sin esfuerzo. ya que se implementa un algoritmo que rastrea la página web y obtiene toda la información deseada [4]. La página referente es *hobbydb* [5] ya que la aplicación oficial recoge los datos de ella, pero tienen restringida cualquier técnica de web scrapping así que se buscó una alternativa en la que poder realizar esta técnica y fue con *funkopops* [6] en la que podía sacar la información necesaria de cada funko. Para realizarlo se utilizó una librería externa y así poder extraer el código HTML de la página y recopilar la información. Esta técnica se utiliza para buscar el funko que el usuario ha escaneado con el código de barras o mediante la voz, en el que recojo el valor escaneado o la palabra dicha. Con ese valor recogido se hace la petición a la Web y mediante el algoritmo se recoge la información.

Inspeccionando sobre la página *hobbydb* encontré que hace una petición al servidor para que le devolviera un catálogo de funkos, justo lo que se quería conseguir, así que consideré cambiar la forma de obtener los datos ya que era una página oficial que tendría mejor información y mucha más cantidad de muñecos. Para llevar a cabo esta técnica utilicé otra

librería Volley [7] y de la misma forma que el web scrapping se implementó un algoritmo para sacar toda la información necesaria. Además con esta técnica se pudo llenar la BBDD de forma eficiente y de esta manera tener mayor cantidad de Funkos, logrando poco a poco tener miles de ellos.Además, es oficial de Android y sirve para facilitar y agilizar el uso de redes en las aplicaciones, es decir le mandamos una petición (una url) y te devuelve el valor de la página, en este caso devuelve un Json con la información de los funkos que de la misma manera creo mi objeto de tipo Funko con los datos necesarios. A continuación en la figuras 4.1 y 4.2, veréis los algoritmos implementados para cada una de las técnicas.



```
class ScrappingFunko(){

    suspend fun obtenerDatos(upc:String): MutableList<FunkoModel>  {

    val test = withContext(Dispatchers.Default) {  this: CoroutineScope
        try {
            // intento webScrappin
            val url = "https://funkopops.es/?s=${upc}"
            val docWeb = Jsoup.connect(url).get()
            val x = docWeb.select( cssQuery: "strong")
            val doc = Jsoup.connect( url: "https://funkopops.es/?s=$upc").get()
            val imagen = doc.select( cssQuery: "img")
            val imagenFinal = imagen[2].attr( attributeKey: "abs:src")
            val datos = doc.getElementsByClass( className: "inside-article")
            var numArticulos = datos.size
            var funkos: MutableList<FunkoModel> = mutableListOf()
            //Recogemos la informacion de los Funkos: urlImagen y nombre
            for (i in 0..numArticulos - 1) {
                var info = datos[i].childNode( index: 0).childNode( index: 1).childNode( index: 1).attributes()
                var list = info.toMutableList()
                val urlImagen = list[2].value
                val name = list[4].value
                //Creamos el objeto FunkoModel
                funkos.add(FunkoModel(name,  referencia: "",  upc: "",  precio: "", urlImagen, isLike: false, isCollection: false))
            }
            println(funkos)
            //Recuperamos el resto de informacion del funko
            var listaFunkos: MutableList<FunkoModel> = mutableListOf()
            for (i in 0..funkos.size-1) {
                //Para buscar el resto de informacion del funko
                var name = funkos[i].imagen
                val tamName = name.length -13
                println(name.subSequence(55, tamName))
                val realName = name.subSequence(55, tamName)
                val doc2 = Jsoup.connect( url: "https://funkopops.es/$realName/").get()
                val k = doc2.getElementsByClass( className: "su-column-inner su-u-clearfix su-u-trim")
                val h = k[0].childNode( index: 1).attributes()
                val li = h.toMutableList()
                val urlImagen2 = li[2].value
                val info4 = doc2.select( cssQuery: "strong")
                val ref = info4[0].childNode( index: 0)
                val tipo: String? = info4[3].childNode( index: 0).toString()
               // val id = info4[4].childNode(0).toString()
                val precio = info4[info4.size - 2].childNode( index: 0)
                var n = funkos[i].nombre.length
                var nombre = funkos[i].nombre.toString().subSequence(18,n).toString()
                listaFunkos.add(FunkoModel(nombre, ref.toString(), upc, precio.toString(), urlImagen2, isLike: false, isCollection: false))
            }
             showView(listaFunkos)  ^withContext
        }catch (e: Exception) {
            Log.e( tag: "ERROR: ",  msg: "FUNKO NO ENCONTRADO")
            val listaFunkos = mutableListOf<FunkoModel>()
            showView(listaFunkos)  ^withContext
        }
    }
       return test as MutableList<FunkoModel>
    }
```

Figura 4.1: Algoritmo de web scrapping

```
fun buscarFunkoApi(upc: String, funkoActivity: FunkoView) {
    viewModelScope.launch { this: CoroutineScope
        val listado: java.util.ArrayList<FunkoResponse> = arrayListOf()
        val requestQueue = Volley.newRequestQueue(funkoActivity)
        val URL =
            "https://www.hobbydb.com/api/catalog_items?filters=%7B%22barcodes%22:%22$upc%22%7D&include_cit=true&include_count=false
        val jsonObjectRequest = JsonObjectRequest(
            Request.Method.GET, URL, jsonRequest: null,
            { response →
                //aqui ya tengo el json
                try {
                    println(response)
                    //guardamos en la BBDD local
                    val datosArray = response.getJSONArray( name: "data")

                    val tam = datosArray.length()
                    var i = 0
                    while (i < tam) {
                        val info = datosArray.getJSONObject(i)
                        val atributos = info.getJSONObject( name: "attributes")
                        println(atributos)
                        val id = atributos.get("id").toString()
                        println(id)
                        var valor_estimado = atributos.get("estimated_value").toString()
                        if (valor_estimado == "null") {
                            valor_estimado = "0.0"
                        }
                        val nombre = atributos.get("name").toString()
                        val referencia = atributos.get("ref_number").toString()
                        val exclusivo = atributos.get("variant_details_summary").toString()
                        val fecha = atributos.get("date_from").toString()
                        val recuperarImagenes: JSONObject = atributos.getJSONObject( name: "images")
                        val im1 = recuperarImagenes.getString( name: "detail_url").toString()
                        val im2 = recuperarImagenes.getString( name: "main_photo_url").toString()
                        val im3 = recuperarImagenes.getString( name: "search_url").toString()
                        val imagenes: ArrayList<String> = arrayListOf()
                        imagenes.add(im1)
                        imagenes.add(im2)
                        imagenes.add(im3)
                        val recuperarSeries: JSONArray = atributos.getJSONArray( name: "series")
                        var series: ArrayList<String> = arrayListOf()
                        var j = 0
                        while (j < recuperarSeries.length()) {
                            val serie = recuperarSeries.getJSONObject(j)
                            val nombreSerie = serie.getString( name: "name")
                            series.add(nombreSerie)
                            j++
                        }
                        val funko = FunkoResponse(id, valor_estimado, nombre, referencia, exclusivo, fecha, upc, imagenes, series)
                        listado.add(funko)
                        i++
                    }
                    listadoFunkos.postValue(listado)
                } catch (e: JSONException) {
                    e.printStackTrace()
                }
            }
        ) { }
        requestQueue.add<JSONObject>(jsonObjectRequest)
    }
```

Figura 4.2: Algoritmo API REST

## 4.1. Librerías externas

### 4.1.1. Librerías nativas

La implementación de estas librerías es casi indispensable para un desarrollo y funcionamiento adecuado. Básicamente son para:

- El uso de actividades y fragmentos.
- Integrar el patrón MVVM.
- Realizar acciones en segundo plano, denominadas en Kotlin corrutinas [8]
- Componentes gráficos como recyclerview [9], navigation [10] entre otros.

Se puede observar en la figura 4.3 la implementación de todas librerías que se han necesitado para el desarrollo de la app.

```
implementation 'com.google.android.material:material:1.4.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
implementation "androidx.fragment:fragment-ktx:1.3.6"
implementation "androidx.activity:activity-ktx:1.3.1"
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"
implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.3.1"
implementation 'androidx.legacy:legacy-support-v4:1.0.0'
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.5.1'
implementation'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.4.1'
implementation 'androidx.recyclerview:recyclerview:1.2.1'
implementation 'androidx.navigation:navigation-fragment-ktx:2.3.5'
implementation 'androidx.navigation:navigation-ui-ktx:2.3.5'
```

Figura 4.3: Librerías nativas

### 4.1.2. Librerías no nativas

A diferencia de la otras, estas no son obligatorias, simplemente puedes usarlas para beneficiarte a la hora de desarrollar. Se han usado para:

- Lector de código de barras. [11]
- Cargar imágenes desde una url. [12]
- Ampliar imágenes dando doble clic sobre ellas. [13]
- Animar las imágenes. [14]

- Realizar webScrapping. [15]
- Integrar el servicio de Firebase. [16]

Se puede observar en la figura 4.4 la implementación de todas librerías que se han utilizado para el desarrollo de la app.

```
implementation 'com.google.zxing:core:3.4.0'
implementation 'com.journeyapps:zxing-android-embedded:4.1.0'
implementation 'org.jsoup:jsoup:1.11.3'
implementation  'com.squareup.picasso:picasso:2.71828'
implementation 'com.github.MikeOrtiz:TouchImageView:1.4.1'
implementation 'com.airbnb.android:lottie:4.0.0'
implementation platform('com.google.firebase:firebase-bom:28.4.1')
implementation 'com.google.firebase:firebase-analytics'
implementation 'com.google.firebase:firebase-auth-ktx'
implementation 'com.google.firebase:firebase-config-ktx'
implementation 'com.google.firebase:firebase-database-ktx'
implementation 'com.google.firebase:firebase-firestore-ktx'
implementation 'com.google.android.gms:play-services-auth:19.2.0'
implementation 'com.google.firebase:firebase-storage-ktx'
```

Figura 4.4: Librerías no nativas

## 4.2. Funcionalidades

### 4.2.1. Launch

Esta es la actividad main, su IU la podéis observar en la figura 4.5. Se encarga de arrancar la aplicación comprobando primero si tenemos conexión, en caso de que sí, determina si estamos logeados, nos lleva a Home (figura 4.2.5) o Account (figura 4.2.4) dependiendo de la respuesta. Si es la primera vez que accedemos a la app nos lleva Onboarding (figura 4.2.2).

El método para comprobar la conexión es genérico y me pareció perfecto usar el código de *Daniel Alvarez* para mi aplicación. Por tanto la manera de comprobar en todo momento si hay conexión es mediante su código.

Determinar si estamos logeados es sencillo si usamos *SharedPreferences* [17] para guardar el estado del usuario. De la misma manera lo utilizamos para redirigir a la vista Onboarding.

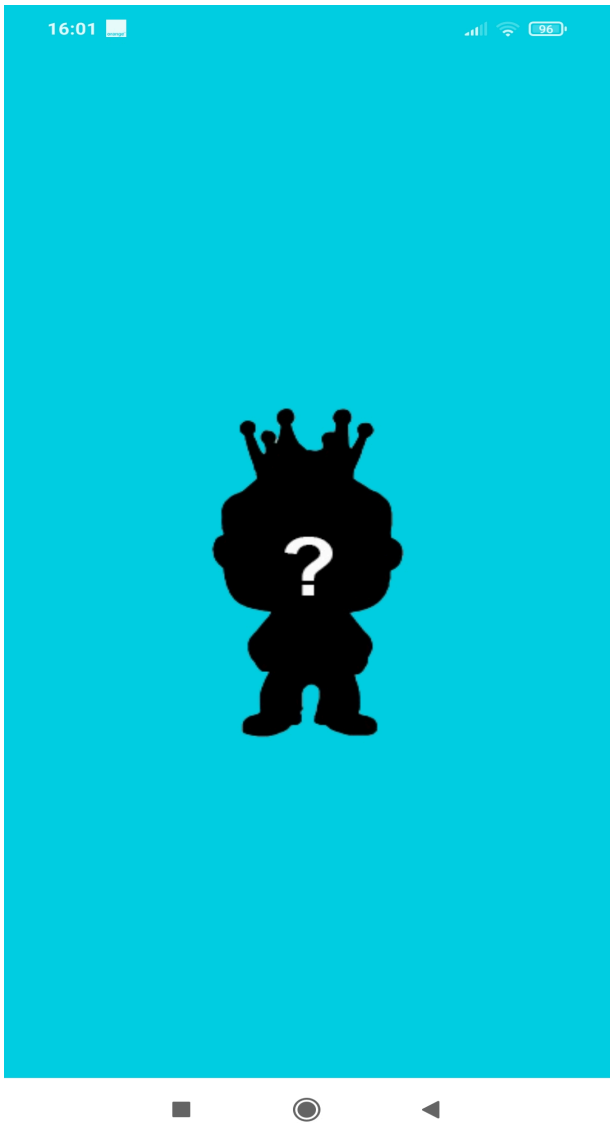



Figura 4.5: IU launch

### 4.2.2. Onboarding

Esta actividad sirve de información para el usuario, es un viewPager [18], es decir, son diferentes pantallas en las que va saliendo información acerca del funcionamiento de la aplicación. Es mostrada la primera vez que usas la app, al finalizar podrás registrarte para comenzar a disfrutar. Su IU la podéis observar en las figuras 4.6, 4.7, 4.8, 4.9, 4.10, 4.11 y 4.12


Figura 4.6: IU onBoarding

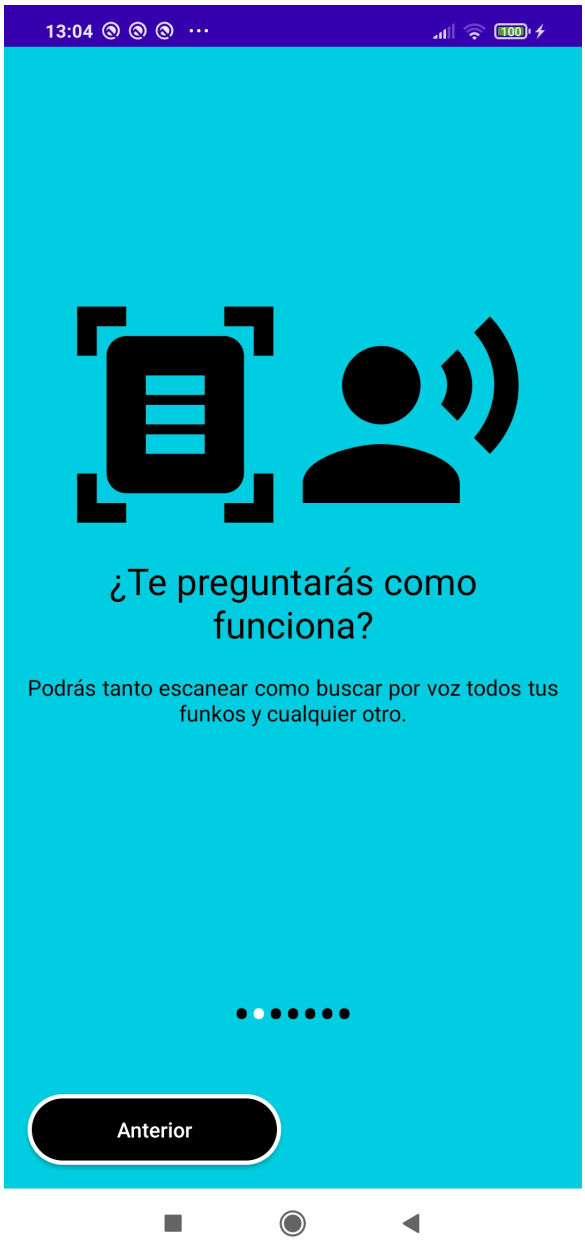



Figura 4.7: IU onBoarding


Figura 4.8: IU onBoarding


Figura 4.9: IU onBoarding

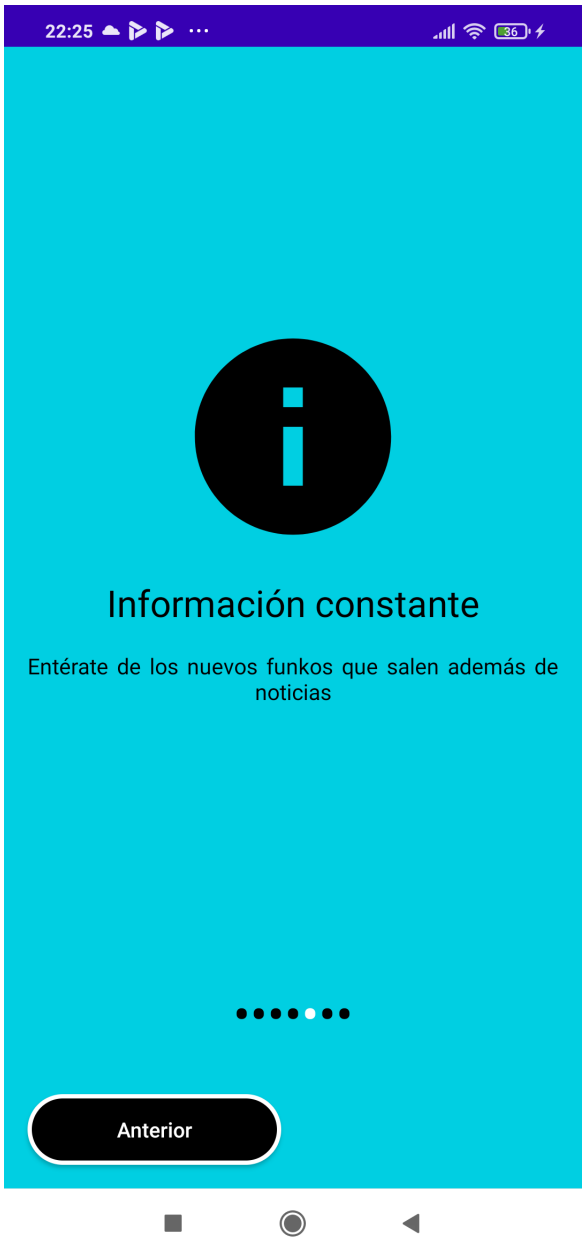



Figura 4.10: IU onBoarding


Figura 4.11: IU onBoarding


Figura 4.12: IU onBoarding

### 4.2.3. Registro

Esta actividad sirve para el registro de un usuario en la aplicación. Deberá rellenar unos campos obligatorios: nick (único), nombre, apellidos y fecha de nacimiento, la foto de perfil si no selecciona una se le asigna una por defecto. Se le da la oportunidad de usar una cuenta propia de email o con la cuenta oficial de Google. Una vez que te registras con cualquiera de las dos cuentas te redirige a Home 4.2.5. su IU la podéis observar en las figuras 4.13, 4.14, 4.15 y 4.16.

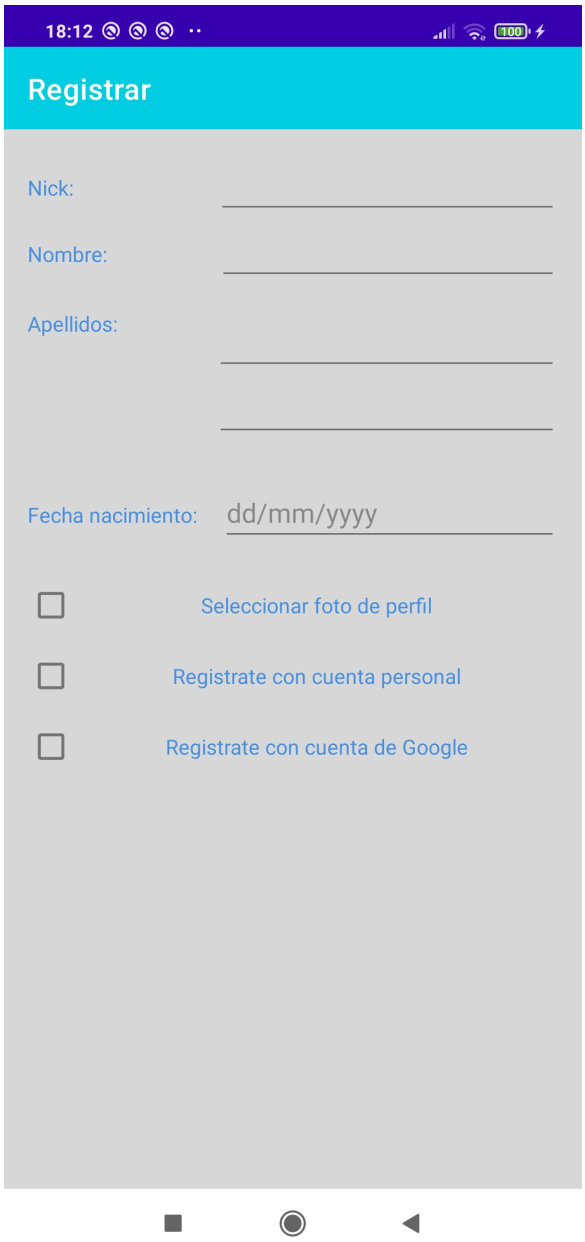



Figura 4.13: IU registrar


Figura 4.14: IU registrar seleccionar foto

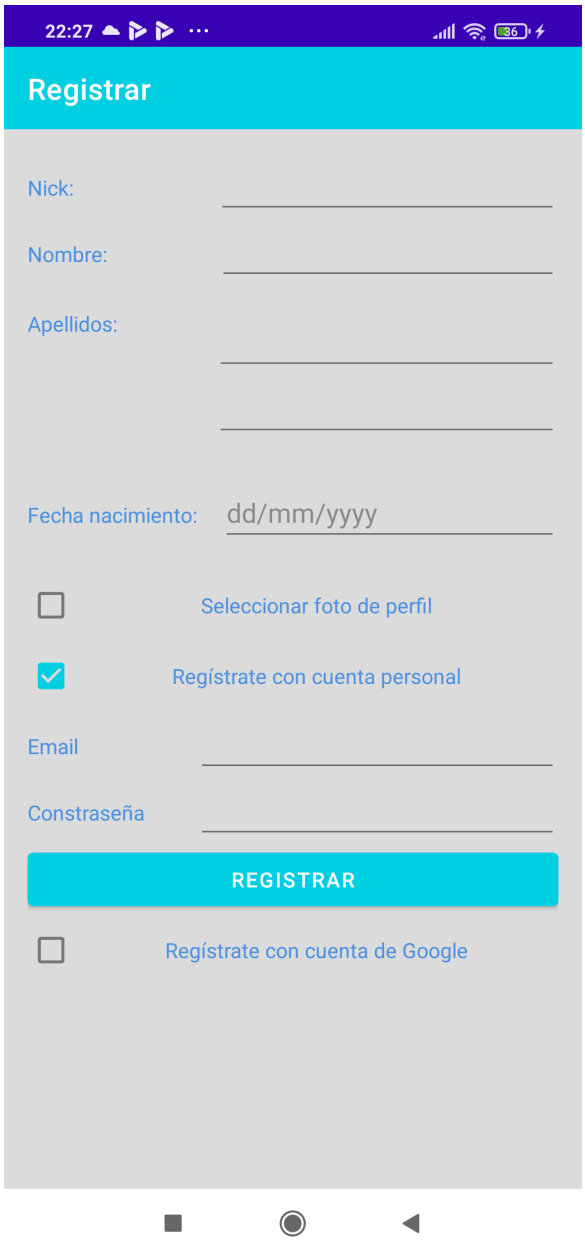



Figura 4.15: IU registrar cuenta propia

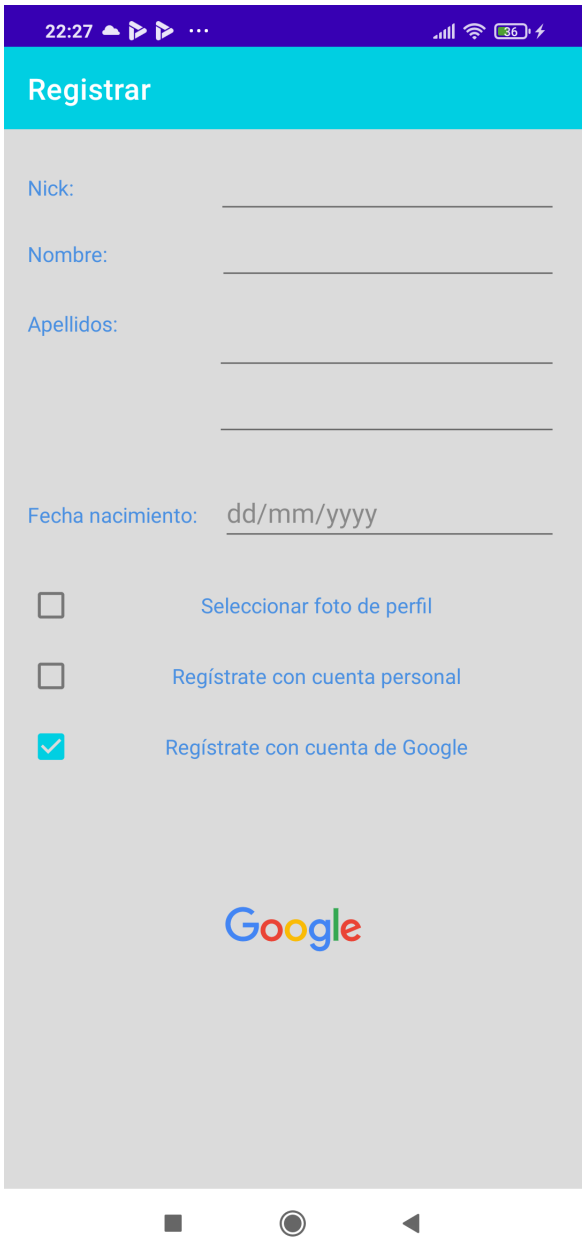



Figura 4.16: IU registrar cuenta Google

### 4.2.4. Accout

Esta actividad sirve para loguearse en la aplicación. Tiene las dos opciones, con cuenta propia o con cuenta Google. Se controla en todo momento que si un usuario nuevo quiere autenticarse se le avisa para que se registre. También tiene la opción de registrarse si no lo ha hecho todavía. Su IU la podéis observar en las figuras 4.17 y 4.18.

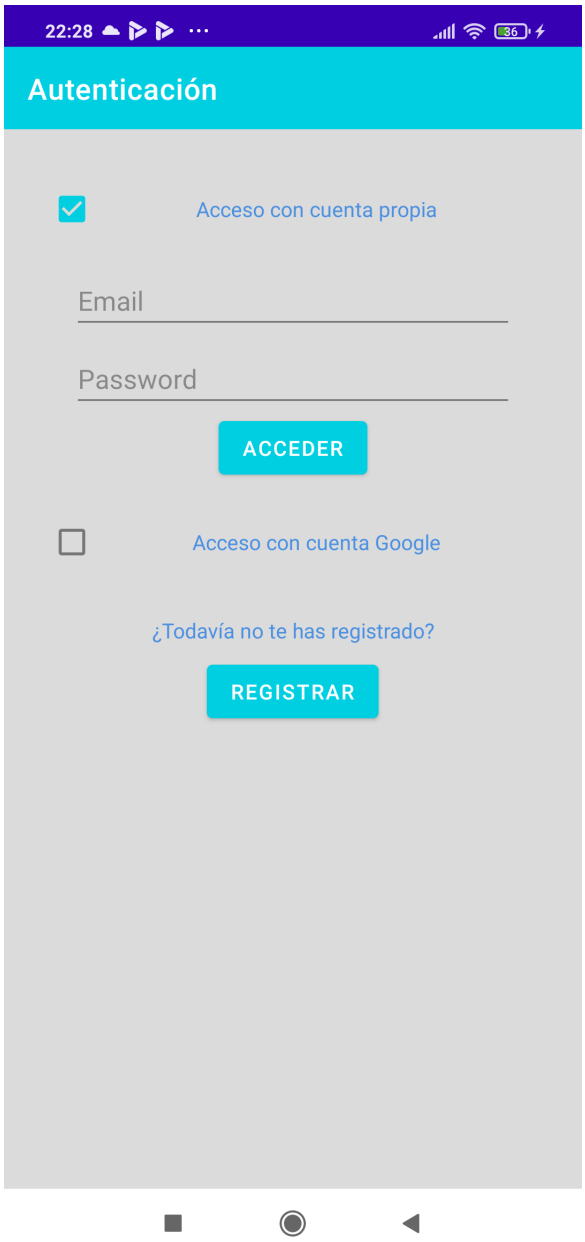



Figura 4.17: IU autenticar cuenta propia

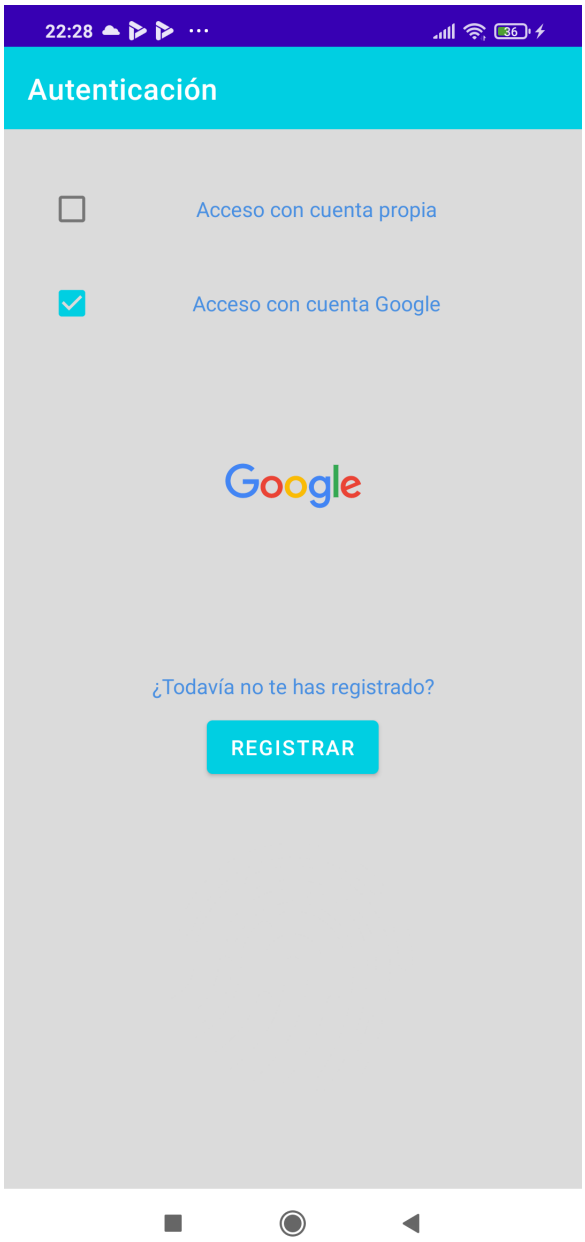



Figura 4.18: IU autenticar cuenta Google

### 4.2.5. Home

Esta actividad es la principal y controla las diferentes acciones, debido a que contiene el navigation bar para redirigir a inicio,lista de deseos, colección y amigos. Además de tener en el toolbar [19] para buscar funkos, escaneando el código de barras 4.2.12 o mediante la voz 4.2.12 y las notificaciones 4.2.8. También tiene un navigation drawer [20] para controlar las acciones de ver perfil 4.2.9, editar perfil 4.2.10, catálogo de funkos 4.2.11 y salir 4.2.15.

Contiene un FragmentContainerView [21] para mostrar las diferentes vistas mencionadas arriba. Inicialmente la vista que se muestra es la de inicio.

**Inicio**

Este fragmento es la vista inicial de la aplicación, contiene dos anuncios en formato banner arriba y abajo del todo, los podéis ver en las figuras 4.19 y 4.20 además muestra dos noticias y un listado de los funkos nuevos que van saliendo al mercado. Las noticias se cargan de la BBDD y tienen un formato en el cual si agrego una nueva noticia se carga sin ningún problema, esta se compone de una foto principal y un título. Si pulsas sobre la noticia se abre un nuevo fragmento con el detalle de la noticia (ver figura 4.21). La lista de los nuevos funkos de igual manera que las noticias las modifico a mi gusto, intentando tener los funkos actuales de cada mes.

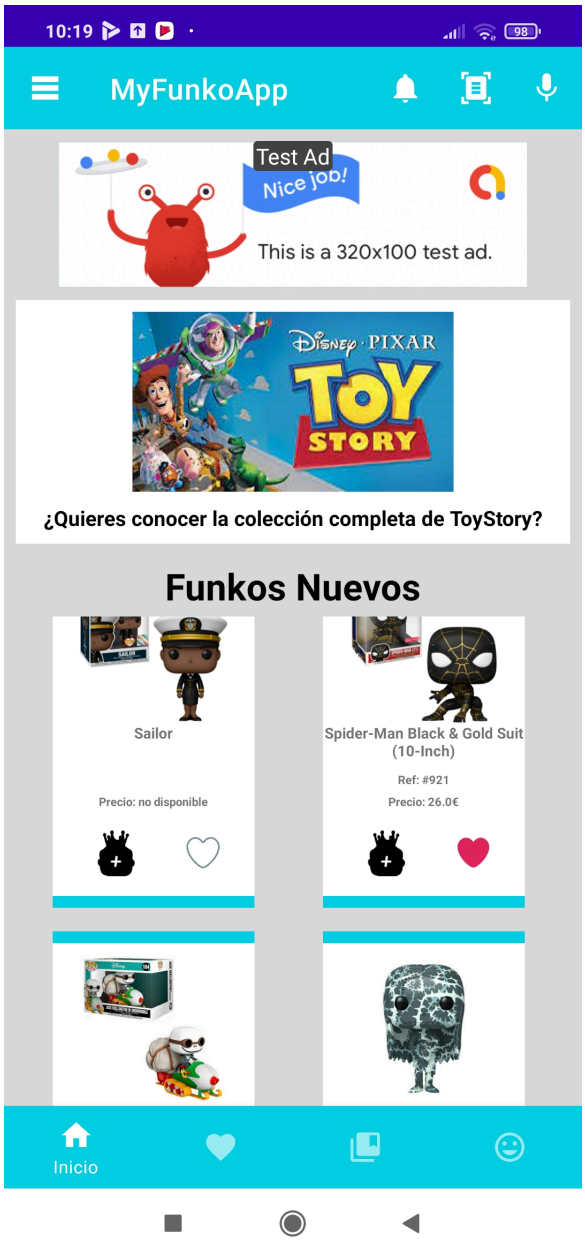



Figura 4.19: IU Inicio


Figura 4.20: IU Inicio


Figura 4.21: IU Noticia

### 4.2.6. Lista de deseos

Este fragmento muestra la lista de deseos del usuario. Se muestran en un recyclerview con un gridlayout cada funko. Se puede interactuar con cada uno pudiendo eliminarlo de la lista de deseos y agregarlo o eliminarlo de la colección. Se refresca automáticamente mostrando los cambios al instante. Además si tienes una lista grande de elementos se puede buscar por el nombre del funko y verás la búsqueda al instante. Puedes ver de un

vistazo el número de funkos que tienes en la lista y el valor total de los funkos. Su IU la podéis observar en las figuras 4.22 y 4.23.

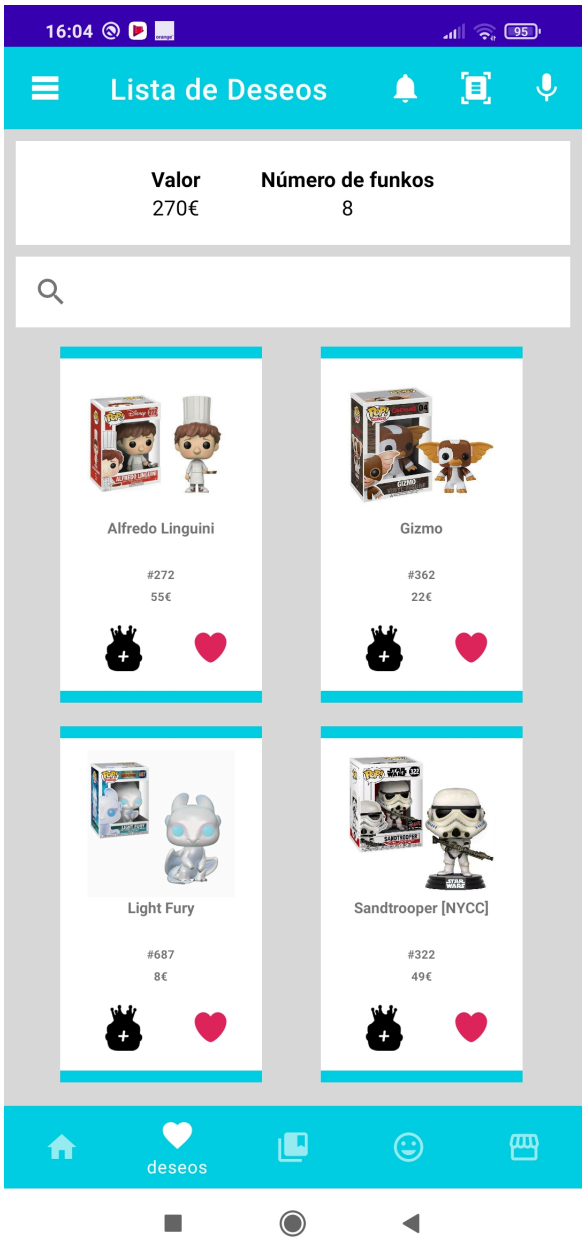



Figura 4.22: IU lista deseos

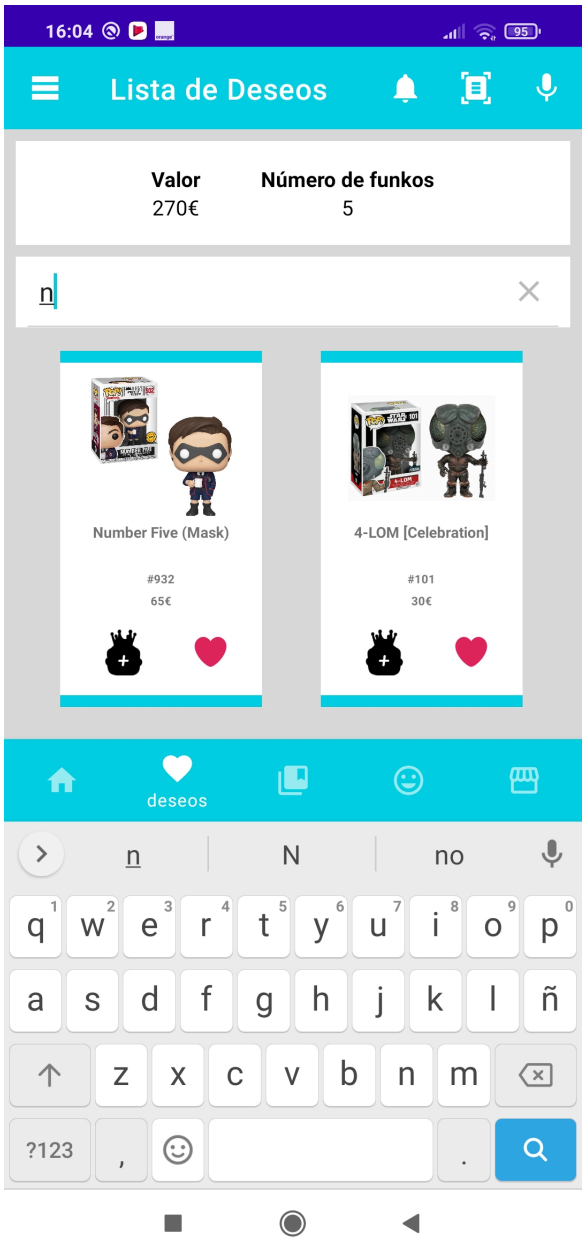



Figura 4.23: IU buscar lista deseos

### 4.2.7. Colección

Este fragmento muestra la colección del usuario. Se muestran en un recyclerview con un gridlayout cada funko. Se puede interactuar con cada uno pudiendo eliminarlo de la colección y agregarlo o eliminarlo de la lista de deseos. Se refresca automáticamente mostrando los cambios al instante. Además si tienes una lista grande de elementos se

puede buscar por el nombre del funko y verás la búsqueda al instante. Puedes ver de un vistazo el número de funkos que tienes en la colección y el valor total de los funkos. Su IU la podéis observar en las figuras 4.24 y 4.25.

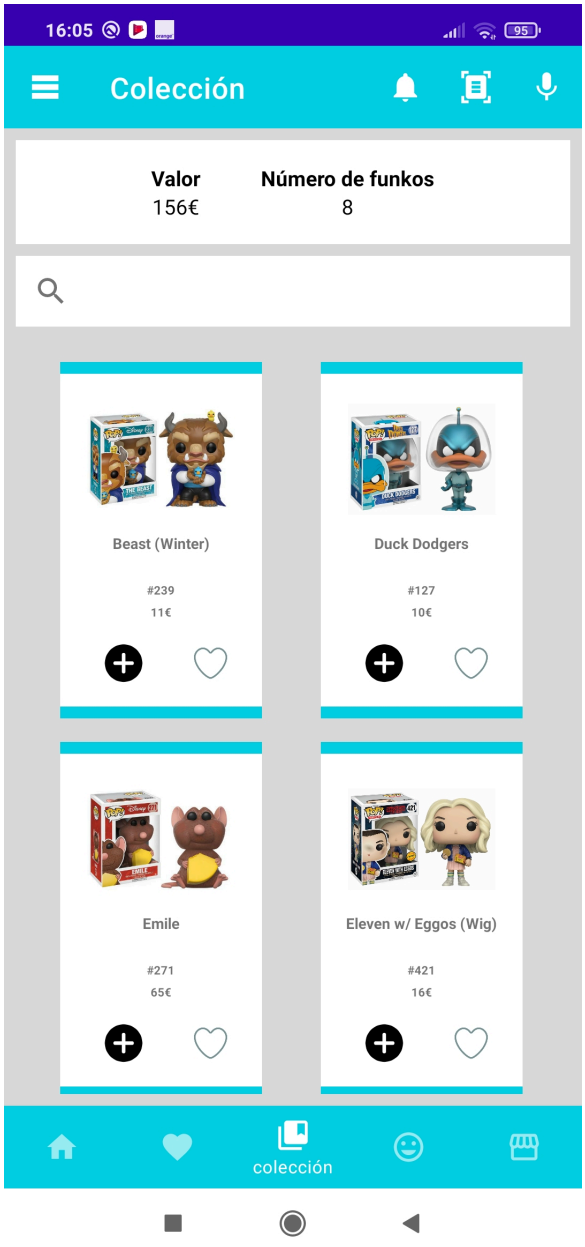



Figura 4.24: IU colección

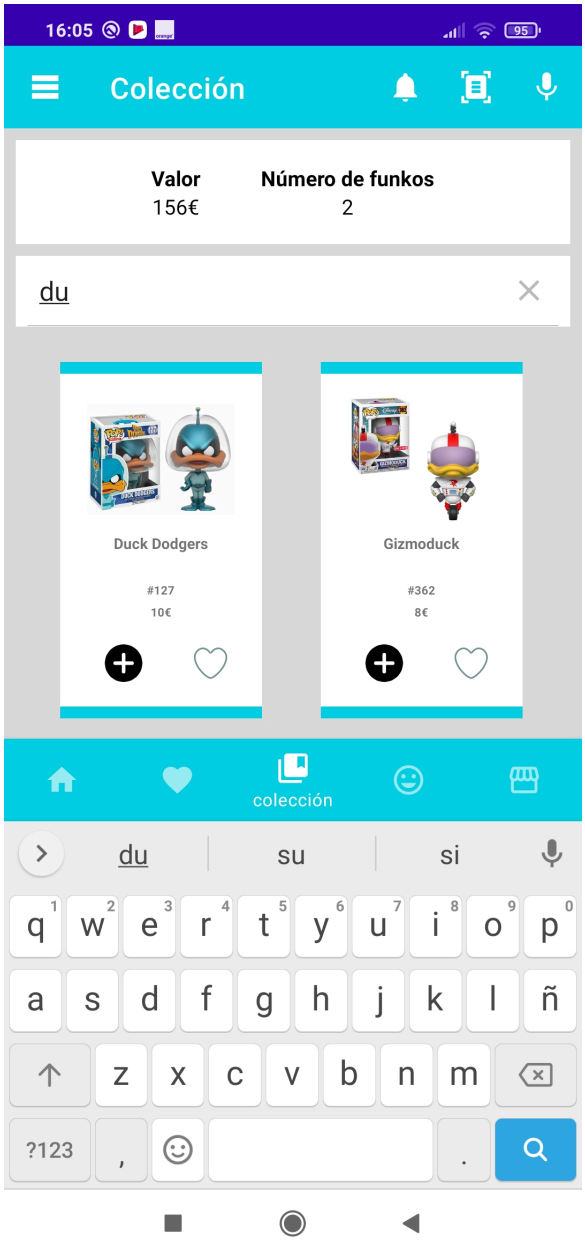



Figura 4.25: IU buscar colección

### 4.2.8. Amigos

Esta actividad podrás ver una lista de todos tus amigos (ver figura 4.26). Tienes la opción de eliminarlo que como consecuencia ni el usuario ni el amigo podrán ver el perfil. Además si pulsas sobre un amigo te redirige a la actividad ver perfil amigo 4.2.8. Por último tienes

el botón para buscar amigo que te lleva a la actividad buscar amigo 4.2.8.

**Buscar amigo**

Este fragmento te muestra una lista de todos los usuarios que hay en la aplicación y se puede enviar una solicitud de amistad. Puedes buscar a un amigo mediante su nombre o su nick (ver figura 4.27).

**Añadir amigo**

Cuando un usuario recibe una solicitud de amistad, puede aceptarla o denegarla. En el caso de que lo acepte ambos se registran como que son amigos y ya tienen permiso para ver el perfil. Si el caso es rechazarla, se elimina la solicitud.

**Eliminar amigo**

Cuando un usuario quiere eliminar un amigo, solo tiene que seleccionar a quien quiere eliminar y pulsar el botón. Salta un alerta para asegurar de que quiere eliminarlo realmente.

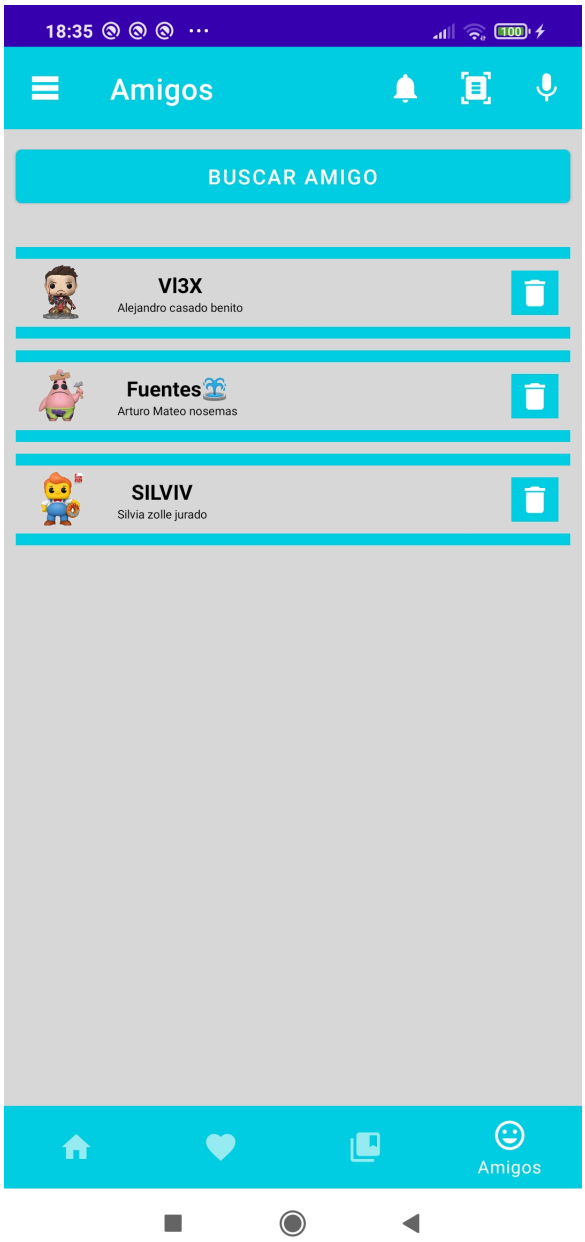



Figura 4.26: IU amigos

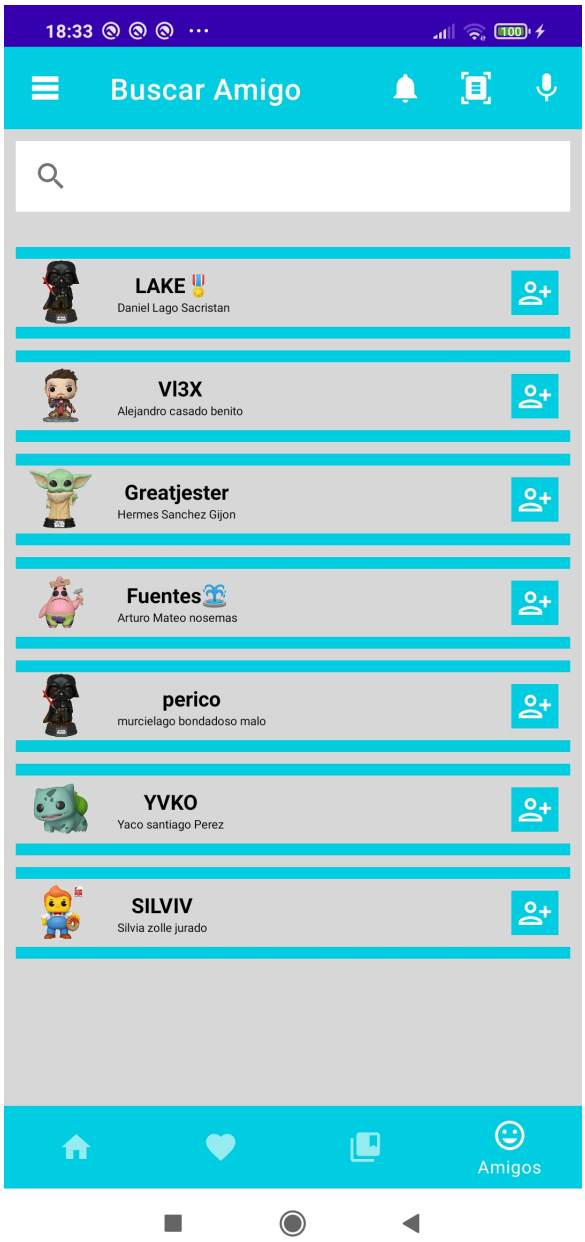



Figura 4.27: IU buscar amigo

### Ver perfil amigo

Cuando un usuario pulsa sobre un amigo se le abre el perfil. En el cual puede ver la información que haya escrito el amigo y puede ver la colección que tiene y la lista de deseos, estas se muestran como explicado anteriormente en las secciones lista de deseos 4.2.6 y colección 4.2.7. Su IU la podéis observar en las figuras 4.28, 4.29 y 4.30.


Figura 4.28: IU perfil amigo

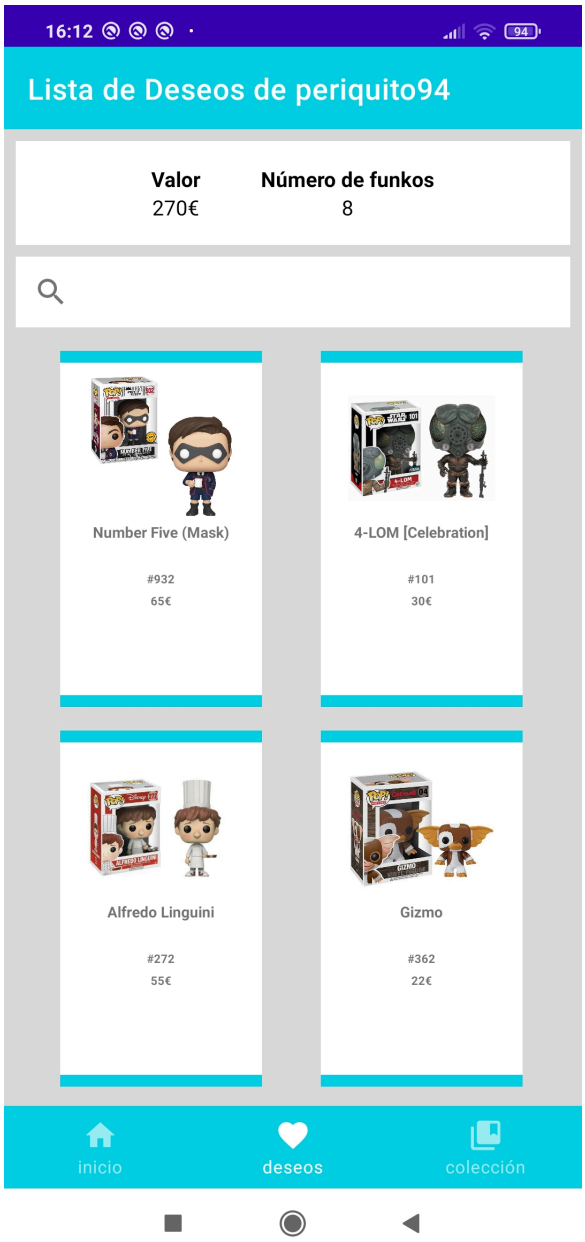



Figura 4.29: IU lista deseos amigo

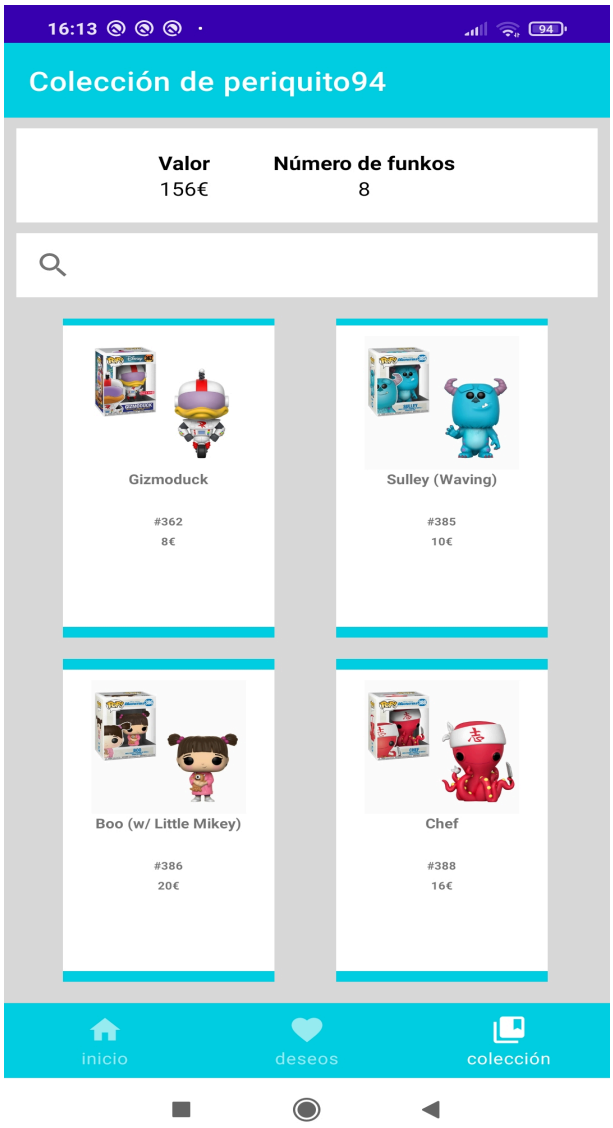



Figura 4.30: IU colección amigo

### Notificaciones

Este fragmento muestra las solicitudes de amistad que un usuario ha recibido. Puedes aceptarlas o rechazarlas. Si un usuario recibe una solicitud, se le marcará un círculo rojo encima del icono de notificaciones para avisarle de que tiene notificaciones pendientes. Su IU la podéis observar en las figuras 4.31 y 4.32.

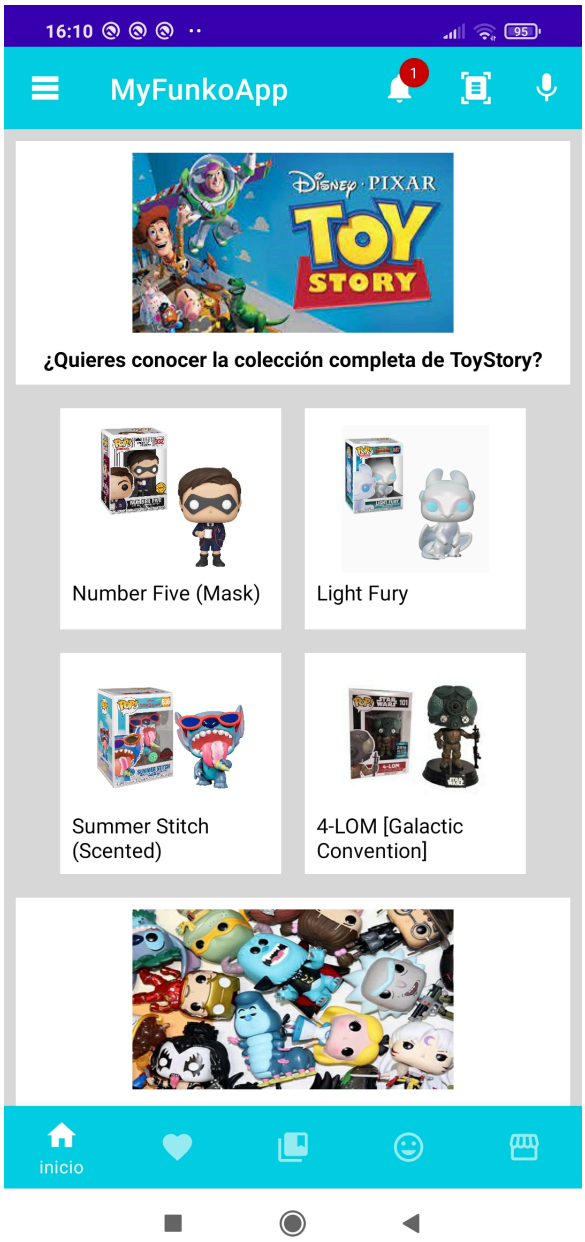



Figura 4.31: IU aviso notificación

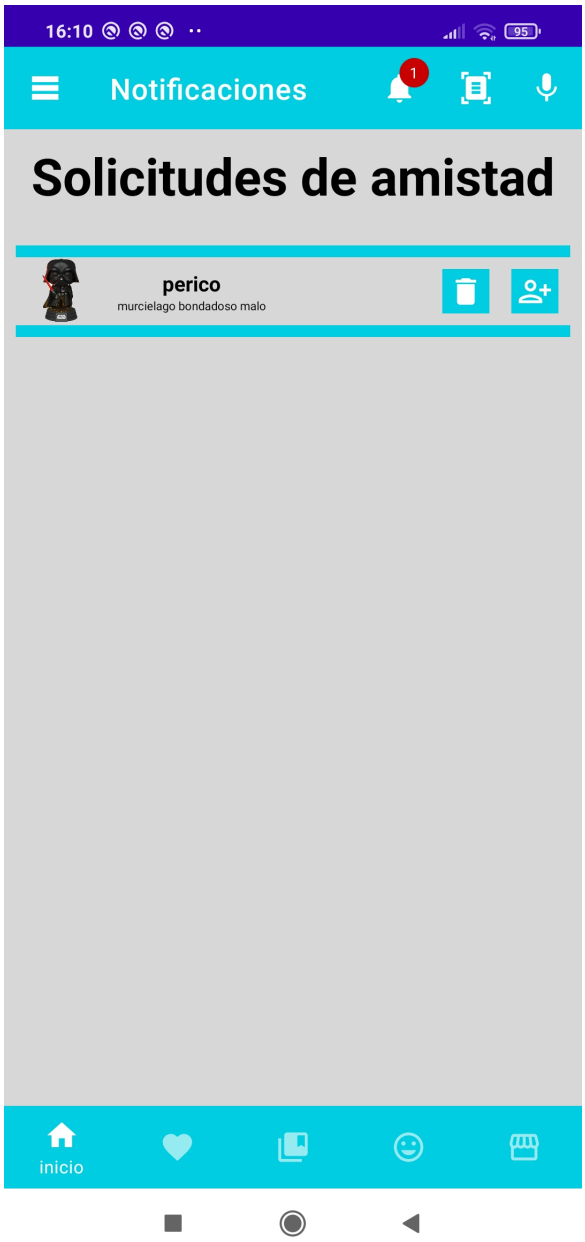



Figura 4.32: IU solicitudes de amistad

### 4.2.9. Ver Perfil

Este fragmento sirve para que el usuario puede ver cómo queda su perfil, con el que le ven sus amigos. Se muestra su foto, su nombre, cuántos funkos tiene en su colección, una descripción sobre él y los métodos de contacto que tiene agregados. Su IU la podéis observar en las figuras 4.33 y 4.34.

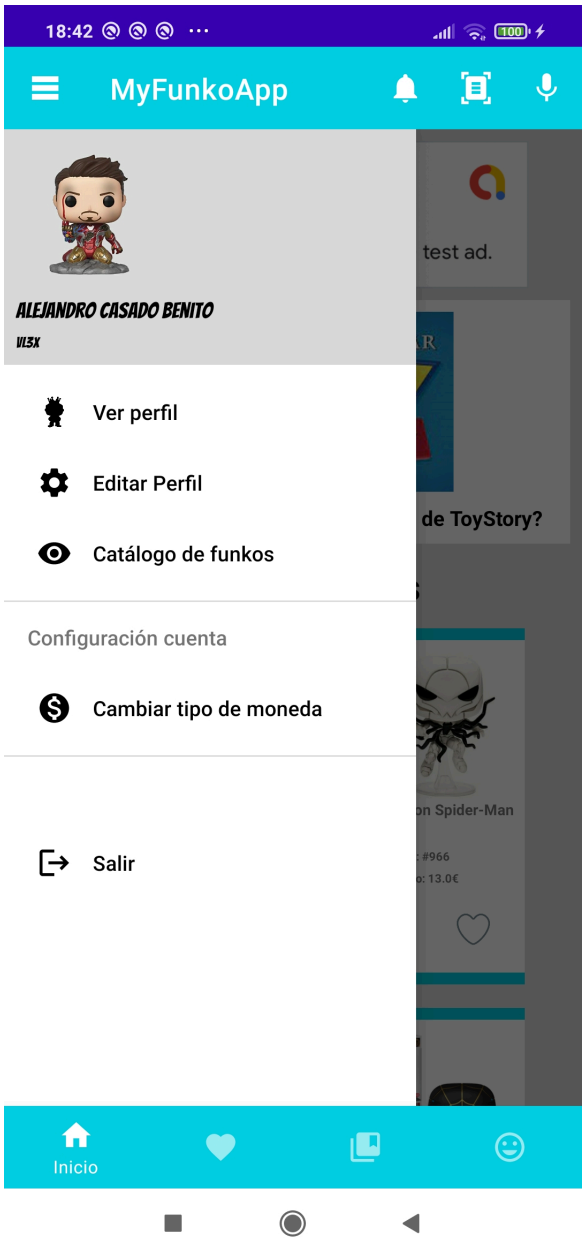



Figura 4.33: IU navigation drawer


Figura 4.34: IU ver perfil

### 4.2.10. Editar perfil

Este fragmento el usuario puede modificar todo acerca de su perfil salvo su email y su nick (ver figura 4.35).

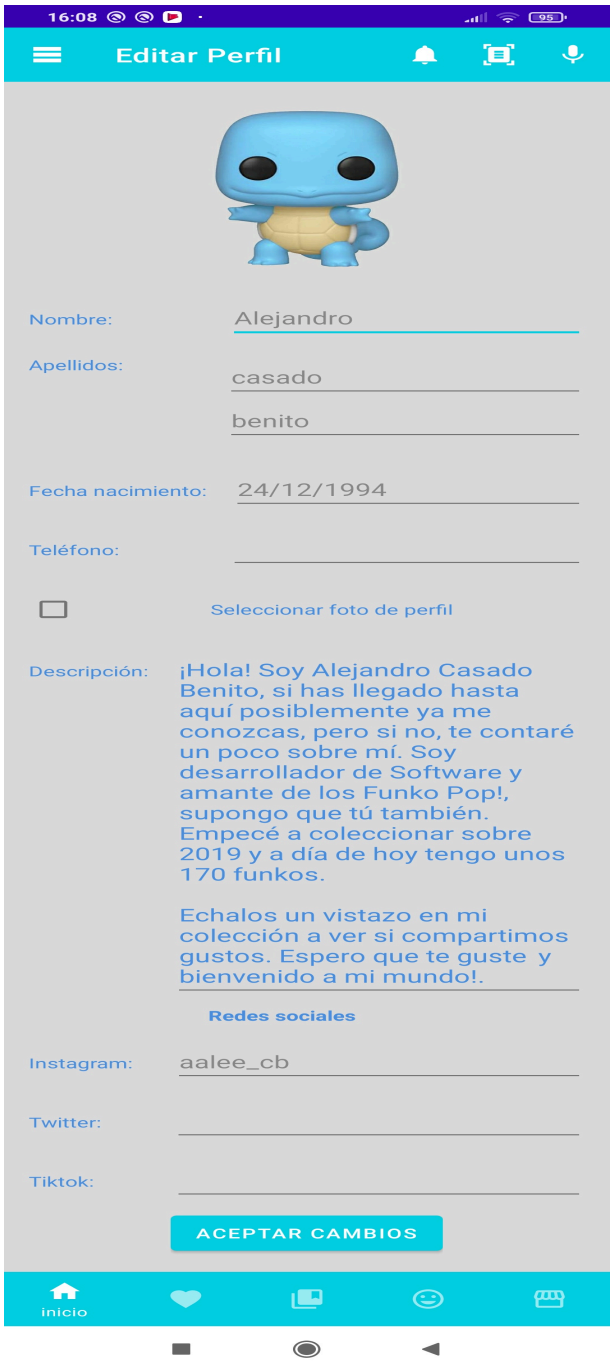



Figura 4.35: IU editar perfil

### 4.2.11. Catálogo de funkos

Este fragmento muestra todos los funkos que hay en la BBDD, sigue la misma mecánica que la lista deseos o la colección. Puedes agregarlos o eliminarlos tanto de la colección como de la lista de deseos. También te permite buscar por nombre o número de referencia del funko. Su IU la podéis observar en las figuras 4.36, 4.37 y 4.38.

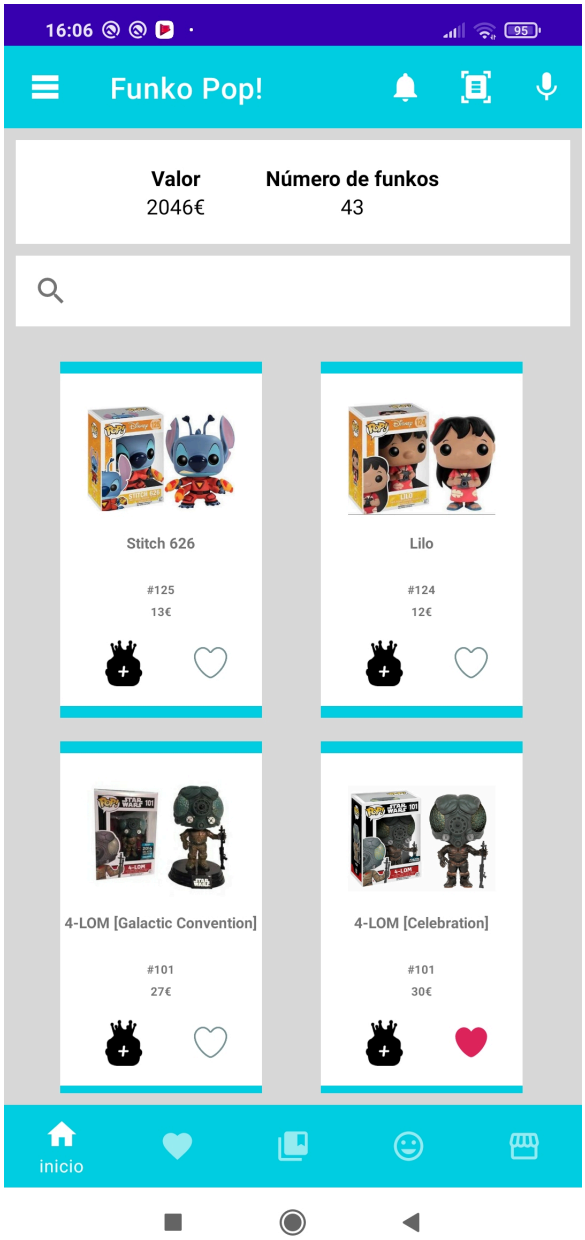



Figura 4.36: IU catálogo de funkos

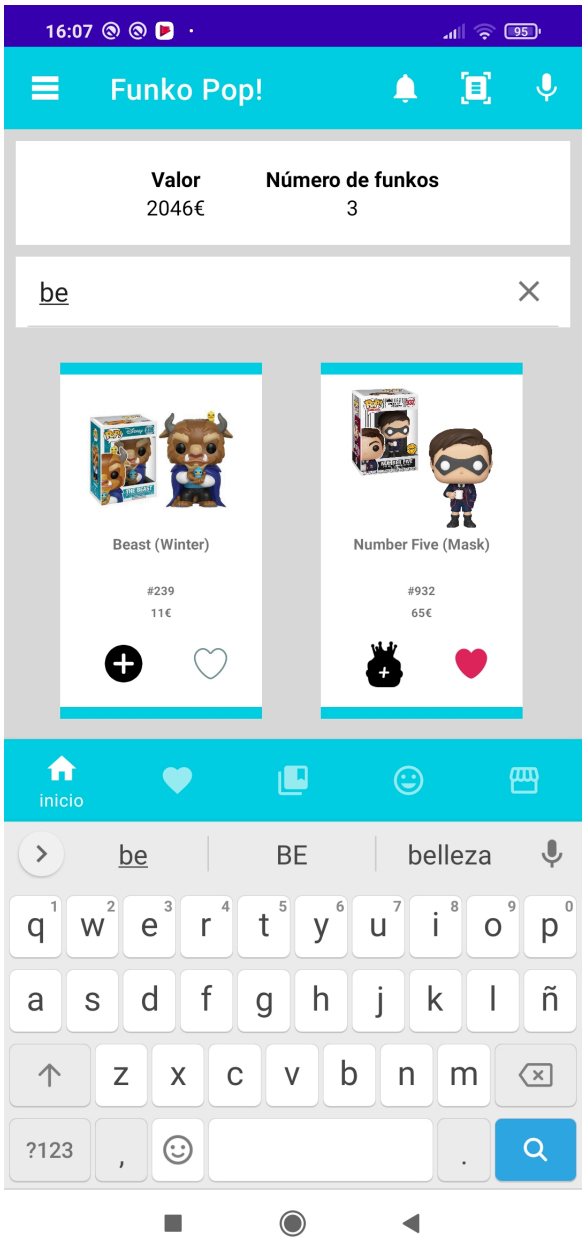



Figura 4.37: IU buscar en catálogo de funkos por nombre

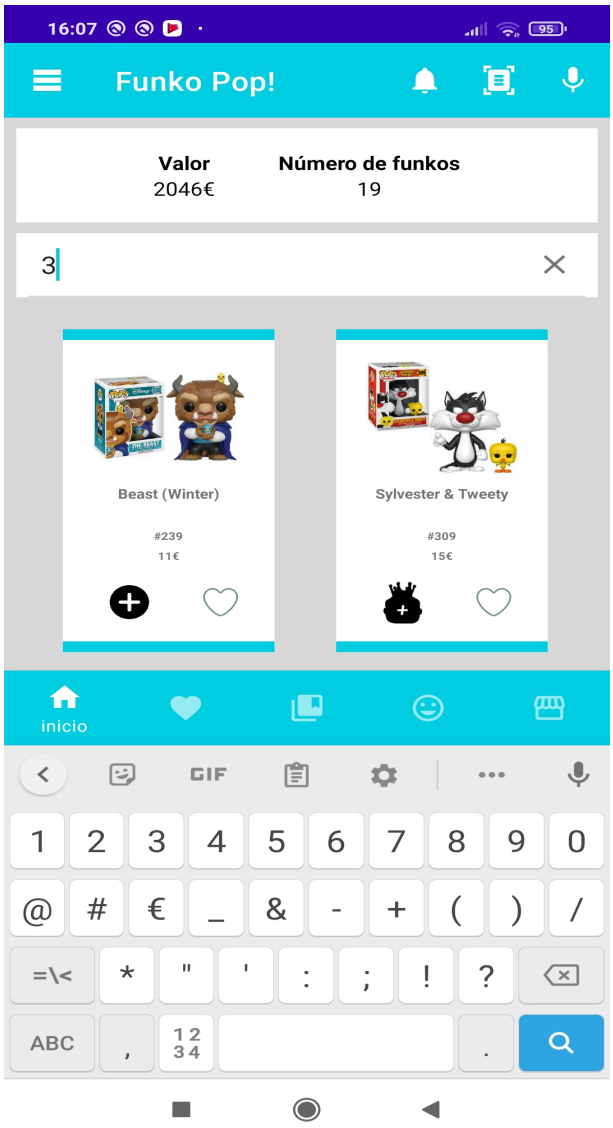



Figura 4.38: IU buscar en catálogo de funkos por referencia

### 4.2.12. Buscar funko

Se hará la búsqueda de dos maneras. La primera se realizará una petición a nuestra BBDD es la forma más rápida. Si en esta petición nos devuelve que no lo ha encontrado, entonces se hace una petición sobre la página *pop price guide* buscando el funko pedido. Si ambas búsquedas devuelven que no lo ha encontrado se redirige al fragmento Error de búsqueda ( ver figura 4.41 ), en caso positivo se le muestra el resultado al usuario dándole la posibilidad de agregarlo o eliminarlo de su lista de deseos o colección. Además de manera amigable si el usuario quiere buscar el funko en páginas como *Wallapop* o *Amazon* para comprarlo puede ir directamente con los enlaces agregados. Si pulsa sobre

la imagen del funko se copia en portapapeles el nombre, para que la búsqueda en las otras páginas sea rápida. Su IU la podéis observar en las figuras 4.39 y 4.40.

### Escanear código de barras

Mediante el escaneo de código de barras, una forma sencilla y rápida de que el usuario busque el funko.

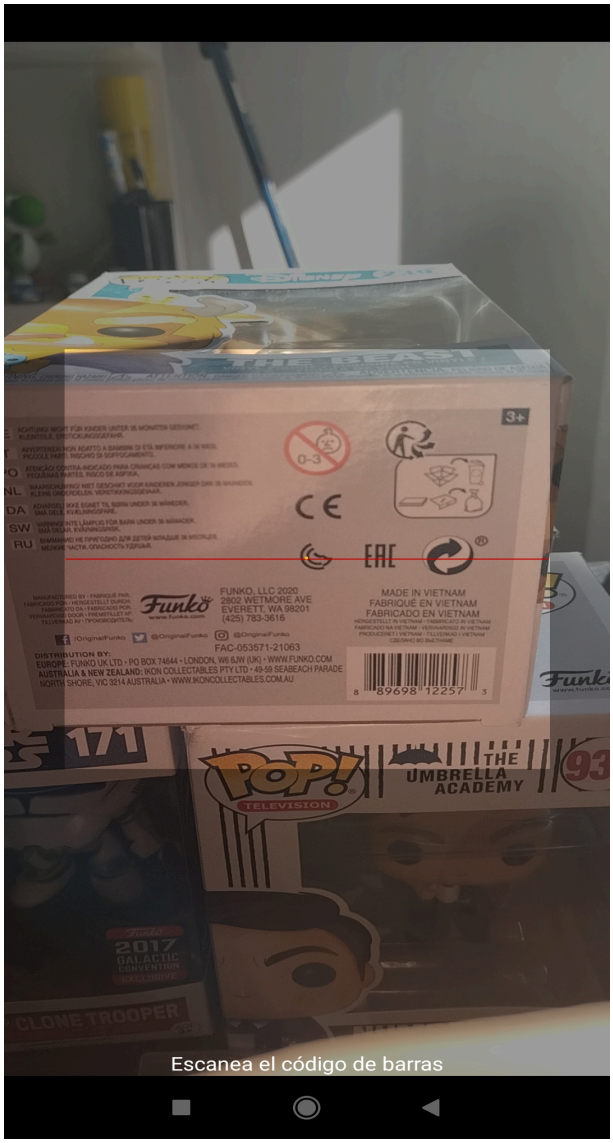



Figura 4.39: IU escaner código de barras

### Voz

Mediante el reconocimiento por voz, el usuario dice el nombre del funko, y este se encarga de buscarlo.

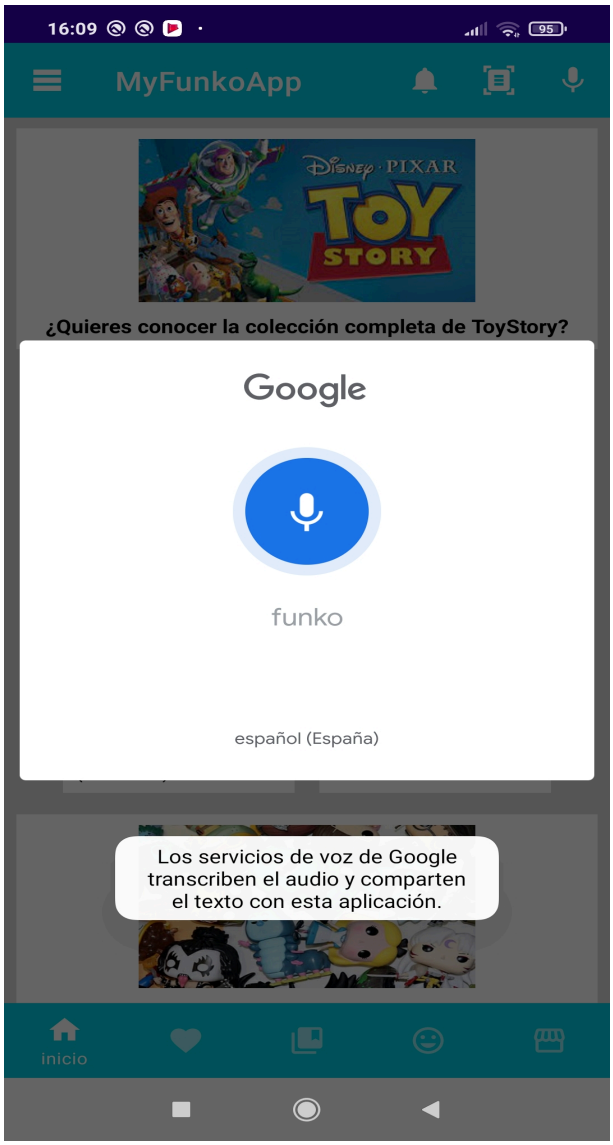



Figura 4.40: IU búsqueda por voz

### Error de búsqueda

Este fragmento muestra de manera amigable que el funko buscado no se encuentra


Figura 4.41: IU Error de búsqueda

### 4.2.13. Detalle Funko

Este fragmento muestra toda la información del funko seleccionado, además de los datos que puedes ver antes de entrar al detalle podrás ver más fotos, la fecha de lanzamiento, si es exclusivo y de dónde es exclusivo y la/s serie/s a las que pertenece. También muestra todos los funkos relacionados con él en función de su serie/s. Podrás interactuar con él agregándolo o eliminándolo de la colección o lista de deseos. Su IU la podéis observar en las figuras 4.42 y 4.43.


Figura 4.42: IU detalle funko pop


Figura 4.43: IU detalle funko pop

### 4.2.14. Sin conexión

Este fragmento se muestra cuando el teléfono con el que se abre la aplicación no tiene disponible una conexión a Internet. Avisa al usuario de que tiene que reiniciar la app y tener conexión para que funcione ( ver figura 4.44 ).

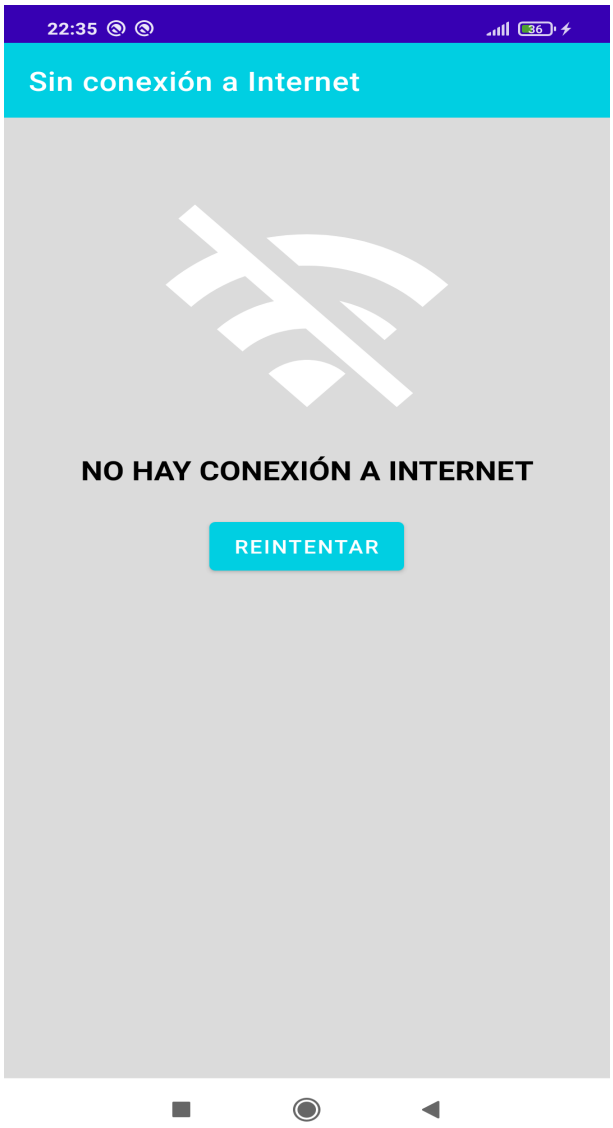



Figura 4.44: IU sin conexión

### 4.2.15. Salir

Este método sales totalmente de la aplicación. Cuando pulsas te abre una alerta por si estás seguro de que realmente quieres salir de la app. Si aceptas la próxima vez que abras la aplicación te llevará a la actividad de account 4.2.4 ( ver figura 4.45 ).

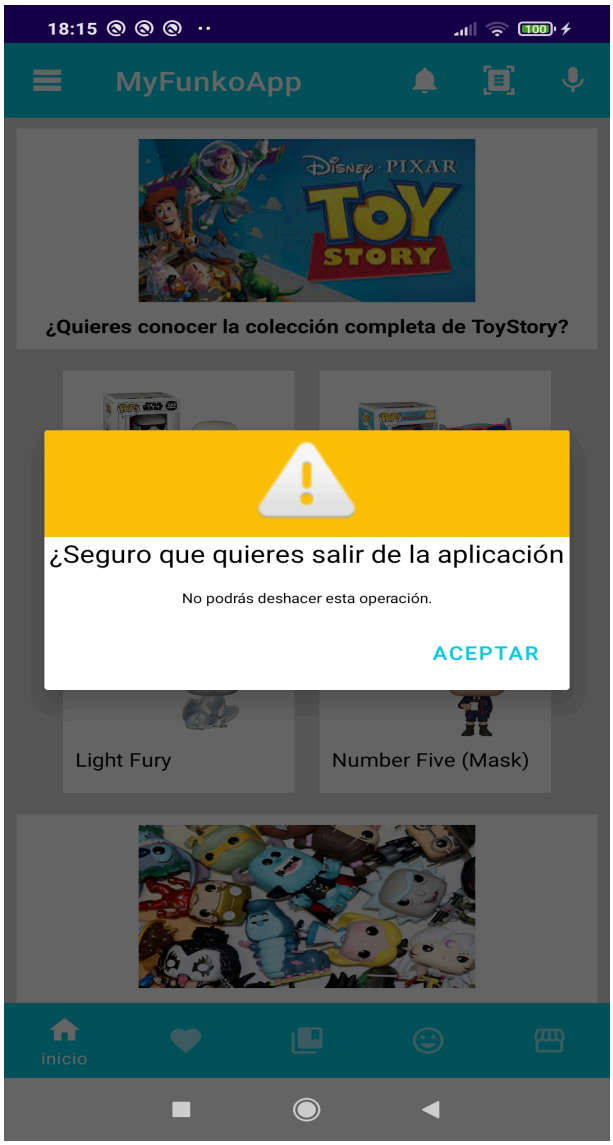



Figura 4.45: IU salir aplicación

## 4.3. Cómo monetizar la app

Una de las preguntas más frecuentes que surgen al crear una app es ¿Cómo gano dinero?, básicamente hay dos tipos de aplicación, las que son de pago y necesitas efectuar un pago para poder descargarla y utilizarla y las que son gratuitas. ¿Cuál es mejor?, no hay una respuesta concreta, si no que, hay elegir bien el modelo que quieres seguir y el que más se ajuste es el que utilizarás. Por ejemplo, Netflix que es mundialmente conocida, es una app de descarga gratuita, pero requiere una suscripción mensual para ver el contenido exclusivo. Otro de los ejemplos es una app que requiere un único pago por descargarla y puedes acceder a todo su contenido.

Actualmente existen varios modelos populares para monetizar tu app y son los siguientes:

**Freemium**

Este es el modelo que más se usa, se trata de aplicaciones de descarga gratuita pero hay pagos por contenidos adicionales. Dentro de este existen diferentes formas como

- Por uso, tiene un uso limitado y hay que pagar por más, como en el caso de Dropbox.
- Free trial, es gratuita durante un tiempo y una vez finalizado ese tiempo hay que pagar para utilizarla, como en el caso de Netflix o HBO.
- Funcionalidad, esto lo utilizan los juegos, normalmente es gratuito jugar pero para conseguir ciertas capacidades o avanzar más rápido tienes que pagar, como en el caso de Clash Royale.
- User experiencie, las apps contienen anuncios y pagas para eliminarlos, como en el caso de Spotify

**Pago por descarga**

Este modelo no es muy habitual, pagas por descargarte la app y accedes a todas las funcionalidades sin tener otras maneras de monetización. Es poco habitual porque llama más la atención algo gratuito que de pago. Pero si tu app aporta un valor muy exclusivo puede ser un modelo bueno.

**Paidmium**

Este modelo lo utilizan apps que además de pagar por descargarla incluye pagos dentro de ella. Una aplicación muy conocida es el videojuego Minecraft.

**Compras In-App**

Este modelo es el más utilizado en aplicaciones que ofrecen productos, son de descarga gratuita pero tienes que pagar para adquirir un bien, como es el caso de Amazon.

### Publicidad In-App

Actualmente está ganando mucho terreno este modelo, son apps totalmente gratuitas pero que contienen anuncios.

### Otros modelos

En algunas ocasiones puedes promocionar productos o servicios de otras empresas dentro de la app, esto es difícil porque solo las grandes aplicaciones tienen este alcance.

### Modelo seleccionado

Pues bien el modelo que voy a utilizar y que más se ajusta a mi aplicación es Publicidad In-App. Es un modelo sencillo pero eficaz y os voy a contar cómo lo voy a llevar a cabo. Principalmente en la ventana de inicio voy a mostrar dos banner que son anuncios que no molestan y aportan beneficio y también cada vez que se ejecute cinco veces el escaneo o la búsqueda por voz (de manera independiente) te mostraré un anuncio de pantalla completa conocido como interstitial, estos también aportan gran beneficio. Voy a utilizar la herramienta de Google AdMob que es de las mejores y además de que se integra de manera sencilla, te proporciona mucha información sobre las acciones de los clientes con tu app. En las figuras 4.46,4.47 y 4.48 podéis ver gráficamente cómo se utilizan en la app.

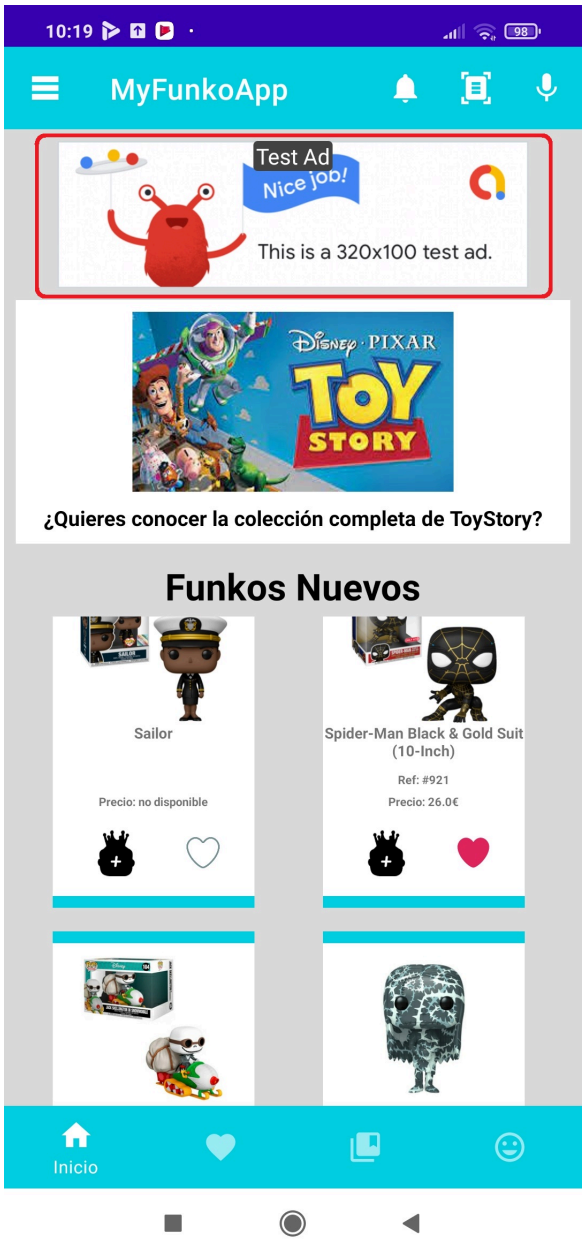



Figura 4.46: IU anuncio tipo banner


Figura 4.47: IU anuncio tipo banner

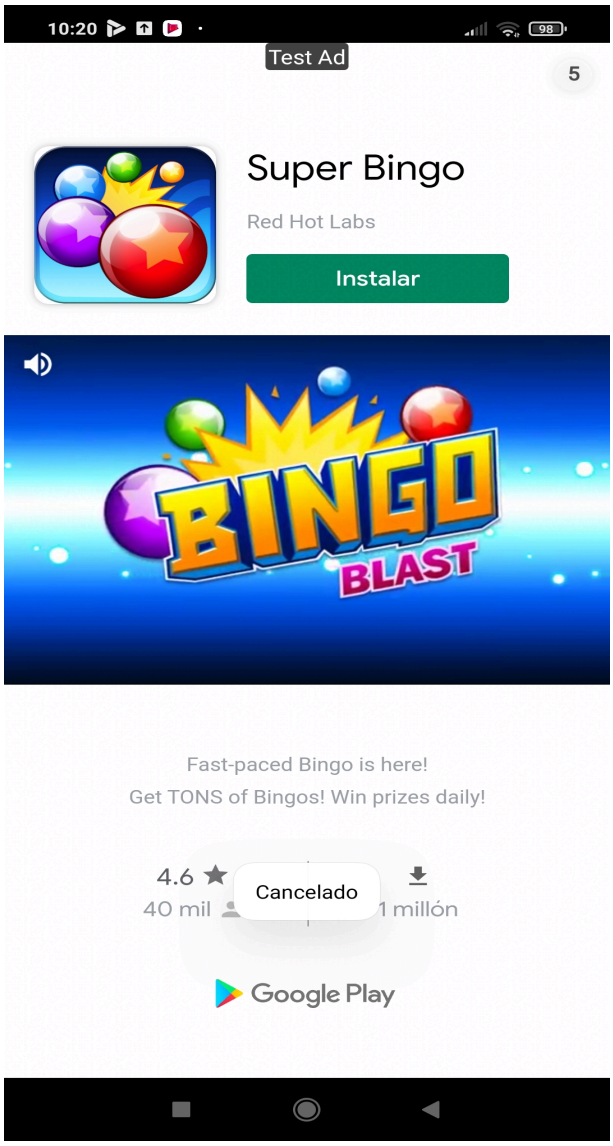



Figura 4.48: IU anuncio tipo interstitial

# Capítulo 5

# Conclusiones

El mundo de los Funko Pop! está en auge, cada vez más y más gente los conoce y quiere hacerse con ellos. En EEUU ya está muy consolidado, pero en España ha ido creciendo el interés en los últimos años, por eso, nace la necesidad de una aplicación que pueda controlar este mundillo.

Actualmente existe una app oficial de Funko que abarca principalmente el público estadounidense. Viendo todas sus valoraciones he llegado a la conclusión de que en España hace falta una aplicación con lo mejor de la oficial y además todas las necesidades que existen, como que esté disponible en español, que puedas tener amigos, ver sus colecciones o lista de deseos, que puedas ver noticias, etc.

Por todo esto y como desarrollador de software no podía dejar la oportunidad de crear MyFunkoApp, una aplicación muy visual, de uso sencillo y que tenga los objetivos estas necesidades mencionadas anteriormente. Como parte fundamental destaca el poder controlar tu colección y lista de deseos además de tener a todos tus amigos o incluso conocer gente con la que compartir este hobby. Se han añadido aspectos y funcionalidades llamativas y novedosas para que tenga una mayor satisfacción cuando el usuario la use, como la búsqueda mediante escáner o voz, animaciones al guardar y eliminar de la colección o lista de deseos con mensajes explicativos para el usuario, facilidad de autenticación mediante correo personal o cuenta de Google y una explicación de todas las funcionalidades que tiene la app cuando la abres por primera vez. En los entresijos, la herramienta fundamental ha sido Firebase [16] que me ha permitido tener una autenticación muy segura y una BBDD no relacional que se ajusta a mi proyecto perfectamente, además de otros

productos que ofrece como storage o crashlytics.

El desarrollo se ha llevado a cabo en Android nativo ya que es el sistema operativo más utilizado en España. Según la página statista [22] en 2020 el 87,1 % de la población utiliza Android frente a un 12,07 % de usuarios que utilizan iOS. Como siempre hay que crecer he seguido una arquitectura MVVM para que sea sencillo integrar nuevas funcionalidades de cara al futuro.

Personalmente, soy consciente de que el trabajo realizado está muy bien, le he dedicado mucho tiempo y amor para que el desarrollo sea profesional y se ajuste a todas las necesidades, con un diseño agradable, un uso sencillo y funcionalidades que aporten valor. Por ello esta app la lanzaré a Play Store para que la disfruten todas las personas que quieran. ¡Que comience la caza de Funko Pop!

# Capítulo 6

# Conclusions

The world of Funko Pop! is booming, more and more people know about them and want to get their hands on them. In the USA it is already well established but in Spain the interest has been growing in recent years, so the need for an application that can control this little world has arisen.

Currently there is an official Funko app that mainly covers the US public. Seeing all their ratings I have come to the conclusion that in Spain we need an application with the best of the official one and also all the needs that exist, such as being available in Spanish, that you can have friends, see their collections or wishlist, that you can see news, etc.

For all this and as a software developer I could not miss the opportunity to create MyFunkoApp, a very visual application, easy to use and that has the objectives these needs mentioned above. As a fundamental part of the application, it is important to be able to control your collection and wish list, as well as having all your friends or even acquaintances with whom to share this hobby. New and eye-catching features and functionalities have been added to make it more satisfying for the user to use, such as search by scanner or voice, animations when saving and deleting from the collection or wish list with explanatory messages for the user, easy authentication via personal email or google account and an explanation of all the functionalities the app has when you open it for the first time. In the intricacies, the fundamental tool has been Firebase [16] that has allowed me to have a very secure authentication and a non-relational database that fits my project perfectly, in addition to other products that it offers such as storage or crashlytics.

The development has been carried out in native Android as it is the most used operating system in Spain. According to statista [22] in 2020, 87.1 % of the population uses Android compared to 12.07 % of users who use iOS. As always, we have to grow, we have followed an MVVM architecture so that it is easy to integrate new functionalities for the future.

Personally, I am aware that the work done is very good, I have dedicated a lot of time and love so that the development is professional and fits all the needs, with a nice design, easy to use and features that add value, so I will release this app to the Play Store for everyone to enjoy it. Let the hunt for Funko Pop begin!

# Capítulo 7

# Trabajo futuro

En este capítulo encontraréis algunas ideas que podrían aportar más valor a la aplicación que por falta de recursos no se han podido llevar a cabo.

**Tienda**

Consiste en vender los propios funkos a través de la aplicación. Se han pensado dos formas de realizarla. La primera mediante una tienda propia en la que tener un stock de funkos y ofrecerlos en la app, en una nueva sección, incluso pudiendo enseñar de manera independiente a cada usuario los funkos que tengan en su lista de deseos y estén en stock para incrementar estas ventas y la segunda manera es asociarse a una o varias tiendas para que ofrezcan sus productos.

Esta idea se intento llevar a cabo, pero no pudo ser, no por temas de implementación si no, por temas legales, ya que a la hora de subir la app a la play store, se necesita ser autónomo para ello. Pero en un futuro si crece MyFunkoApp se puede llevar a cabo.

Aquí podéis ver como era la vista de inicio mostrando cuatro funkos que el usuario tenía en su lista de deseos y estaban disponibles para comprarlos, y se puede ver también el icono de la sección nueva de tienda ( ver figura 7.1.

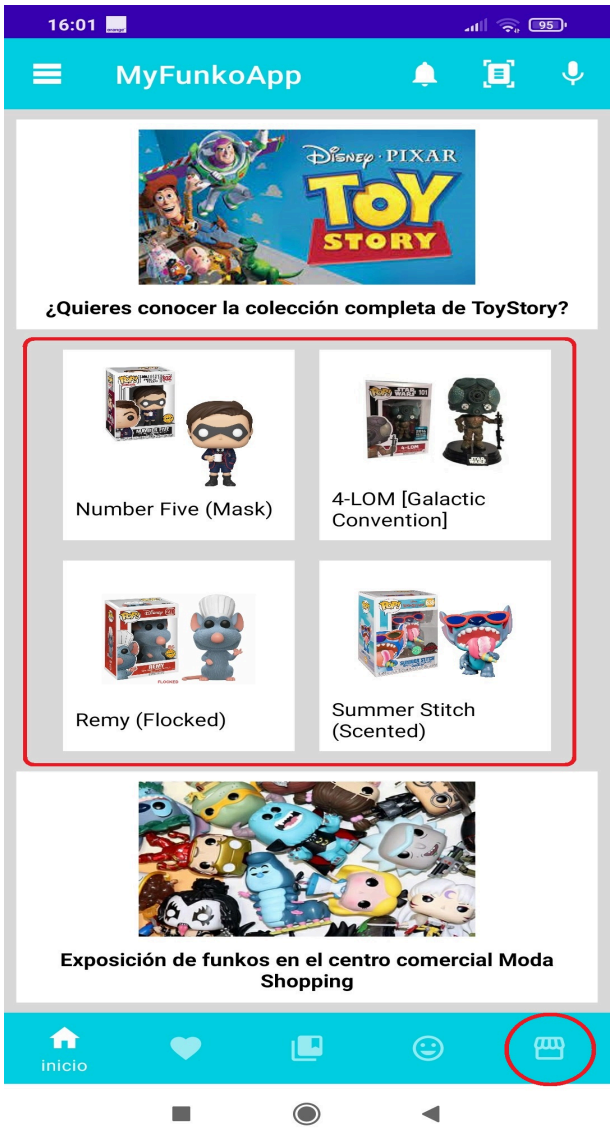



Figura 7.1: IU Muestra cuatro funkos para comprar

### Chat

Esta idea consiste en implantar un chat de intercambio de mensajes entre dos amigos. Sería en el perfil del amigo añadir una sección para entablar conversación y tener una conversación privada.

### Subir imágenes

Esta idea consiste en que un usuario pueda subir imágenes de su colección a su perfil y así poder enseñárselas a todos sus amigos

### Crear MyFunkoApp en iOS

Esta idea consta de recrear la misma aplicación pero para sistema operativo iOS. Sería la más lejana pero con ella conseguiríamos albergar casi el 100 % de la población.

# Índice de figuras

# Bibliografía


[1] "Historia del coleccionismo." https://es.wikipedia.org/wiki/Coleccionismo. Accedido: 21-06-2021.

[2] "Origen de Funko." https://es.wikipedia.org/wiki/Funko. Accedido: 21-06-2021.

[3] "Pop price guide." https://www.poppriceguide.com/, note =Accedido: 21-06-2021.

[4] "web scrapping página." https://sitelabs.es/web-scraping-introduccion-y-herramientas/. Accedido: 01-12-2021.

[5] "página referente de funko pop." https://www.hobbydb.com/. Accedido: 01-12-2021.

[6] "página en la que hice web scrapping." https://funkopops.es/. Accedido: 01-12-2021.

[7] "Android developer Volley." https://developer.android.com/training/volley?hl=es. Accedido: 1-12-2021.

[8] "Android developer corrutinas." https://developer.android.com/kotlin/coroutines?hl=es-419. Accedido: 21-06-2021.

[9] "Android developer recyclerview." https://developer.android.com/jetpack/androidx/releases/recyclerview. Accedido: 21-06-2021.

[10] "Android developer navigation." https://developer.android.com/guide/navigation. Accedido: 21-06-2021.

[11] "Librería para leer código de barras." https://github.com/zxing/zxing. Accedido: 22-06-2021.

[12] "Librería para cargar imágenes desde una url." https://square.github.io/picasso/. Accedido: 22-06-2021.

[13] “Librería para hacer zoom a una imágen.” https://github.com/MikeOrtiz/TouchImageView. Accedido: 22-06-2021.

[14] “Librería para animar las imágenes.” https://github.com/airbnb/lottie-android. Accedido: 22-06-2021.

[15] “Librería para webScrapping.” https://jsoup.org/. Accedido: 22-06-2021.

[16] “documentación oficial del producto firebase.” https://firebase.google.com/. Accedido: 22-06-2021.

[17] “Android developer SharedPreferences.” https://developer.android.com/reference/android/content/SharedPreferences. Accedido: 22-06-2021.

[18] “Android developer ViewPager.” https://developer.android.com/guide/navigation/navigation-swipe-view-2?hl=es-419. Accedido: 22-06-2021.

[19] “Android developer toolbar.” https://developer.android.com/training/appbar/setting-up?hl=es-419. Accedido: 22-06-2021.

[20] “Material designr navigation drawer.” https://material.io/components/navigation-drawer. Accedido: 22-06-2021.

[21] “Android developer fragmentContainerView.” https://developer.android.com/reference/androidx/fragment/app/FragmentContainerView. Accedido: 22-06-2021.

[22] “Cuota de mercado Android vs iOS en España.” https://es.statista.com/estadisticas/473759/tasa-penetracion-sistema-operativo-smartphone-espana/. Accedido: 09-11-2021.

[23] “Página para verificar el estado de la conexión,Daniel Alvarez.” https://medium.com/@alvareztech/verificar-estado-de-conexi%C3%B3n-a-internet-en-tu-aplicaci%C3%B3n-android-d55e2b501302. Accedido: 22-06-2021.