Desarrollo de una aplicación móvil Android para la gestión de una colección de videojuegos (GameItUp)

Trabajo fin de grado (Memoria final)

Autor: Antonio Gonzalez Sanz*1
Tutor: Fernando Pérez Costoya*2

Universidad Politécnica de Madrid (ETSII)
07 diciembre, 2018

Abstract: Memoria final del TFG para una aplicación móvil que permita al usuario gestionar su colección de videojuegos, así como su tiempo de ocio.
Keywords: Memoria final, TFG, Android, Gestión videojuegos, GameItUp

*1 Antonio González Sanz, Email: antonio.gonzalez.sanz@alumnos.upm.es | antoniogonzalezsanz@gmail.com
*2 Fernando Pérez Costoya, Email: fperez@fi.upm.es | fernando.perez@upm.es
Dedicatoria:

A mis padres, por no haberme dejado tirar la toalla y toda su ayuda durante esta larga etapa. A mis amigos por haberme prestado una mano todos estos años y haberme ayudado probando la aplicación, en especial a Carlos Gonzalez por haberme ayudado con el diseño, colores y el logotipo. A mis compañeros de carrera por tantas horas de risa y estudio, mención especial para Itxiar Gonzalez, Aroa López, Saul Quirós, Lucia Mondéjar. A Fernando por haber aceptado ser mi tutor del TFG y haberme apoyado para realizar el TFG que quería. A todos aquellos profesores de la carrera que han resuelto mis dudas y me han enseñado a soñar.

Debo agradecer especialmente a Cristina Choy, mi Daenerys, el haber estado apoyándome durante todos estos años con su cariño y amor incondicional.

Se cierra un capítulo de mi vida, pero empieza otro, y espero contar con todos vosotros en lo que depare el futuro.

Muchas gracias a todos vosotros. No hubiera sido capaz de llegar hasta aquí sin vuestro inestimable apoyo.

“Never tell me the odds!”
“Winter is coming”
# Tabla de contenidos:

<table>
<thead>
<tr>
<th>Capítulo</th>
<th>Título</th>
<th>Página</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Resumen</td>
<td>4</td>
</tr>
<tr>
<td>1.1</td>
<td>Resumen en español</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Resumen en inglés</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Introducción y objetivos</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Introducción</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Objetivos</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Trabajos previos (Estado del arte)</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Grouvee</td>
<td>10</td>
</tr>
<tr>
<td>3.2</td>
<td>The backloggery</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>CLZ Games: Games Database</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Diseño de la aplicación</td>
<td>14</td>
</tr>
<tr>
<td>4.1</td>
<td>Requisitos</td>
<td>14</td>
</tr>
<tr>
<td>4.2</td>
<td>Flujo y diseño de pantallas</td>
<td>15</td>
</tr>
<tr>
<td>4.3</td>
<td>Steam</td>
<td>19</td>
</tr>
<tr>
<td>4.4</td>
<td>HowLongToBeat</td>
<td>20</td>
</tr>
<tr>
<td>4.5</td>
<td>IsThereAnyDeal</td>
<td>22</td>
</tr>
<tr>
<td>4.6</td>
<td>Noticias RSS</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>Desarrollo de la aplicación</td>
<td>25</td>
</tr>
<tr>
<td>5.1</td>
<td>Herramientas</td>
<td>25</td>
</tr>
<tr>
<td>5.2</td>
<td>Metodología</td>
<td>27</td>
</tr>
<tr>
<td>5.3</td>
<td>Desarrollo frontend</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Desarrollo backend</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>Líneas futuras</td>
<td>60</td>
</tr>
<tr>
<td>6.1</td>
<td>Viabilidad económica</td>
<td>63</td>
</tr>
<tr>
<td>7</td>
<td>Resultados</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>Conclusiones</td>
<td>66</td>
</tr>
<tr>
<td>9</td>
<td>Bibliografía</td>
<td>68</td>
</tr>
</tbody>
</table>
1 Resumen

1.1 Resumen en español

Los videojuegos son una de las formas de ocio más importantes en la actualidad. Un consumidor de este modo de entretenimiento maneja un número muy elevado de juegos, tanto digitales como físicos, distribuidos en diversas plataformas, dedicando un tiempo considerable a su consumo.

El principal objetivo de este TFG es el desarrollo de una aplicación para permitir a los usuarios gestionar su colección de videojuegos, manteniendo la información relevante de cada juego. Asimismo, esta aplicación le permitirá encontrar ofertas de videojuegos y obtener noticias sobre los mismos. Facilitará al usuario el poder encontrar tiendas físicas basándose en su geolocalización. En el caso de videojuegos físicos, se le dotará a la aplicación de la funcionalidad para añadirlos a la colección de forma automática, valorándose alternativas como los códigos de barras, etiquetas NFC o sistemas OCR. Por otro lado, la aplicación ayudará al usuario a gestionar su tiempo pudiéndole ofrecer recomendaciones de a qué jugar dependiendo del tiempo del que disponga.
1.2 Resumen en inglés

Videogames are becoming one of the most popular entertainment options. Consumers of this type of entertainment must deal with large quantities of games in both digital and physical media. These are usually also spread amongst several different platforms making it very time-consuming sorting through the entire collection and finding something to play.

The main objective of this end-of-degree thesis is to make an application that will help users manage their videogame library. This application will also help users find videogame offers and discounts and show them relevant videogame news. The application will also find videogame stores near the user using geolocation. In the case of physical games, there will be a research finding a quicker way to add videogames using NFC or OCR techniques. On the other hand, this application will also help the user manage his gaming time offering him recommendations based on the time he has to play.
2 Introducción y objetivos

2.1 Introducción

En la actualidad, la industria del videojuego es una de las más potentes e importantes a nivel mundial. La industria del videojuego se puede dividir en 3 plataformas: PC, Videoconsolas y Móvil. Desde 2015, la industria del videojuego reporta anualmente más beneficios que la industria musical y cinematográfica juntas. En el año 2017 por ejemplo, solamente el mercado móvil superó a toda la industria cinematográfica1.


En 2017 el sector de los videojuegos móviles alcanzo unos beneficios globales de 46.1 mil millones de dólares (Figura 1), mientras que la industria cinematográfica apenas alcanzó los 41.2 mil millones de dólares (Figura 2).

---

¹ Artículo en inglés de Geekadelphia.co “How games overtook the movie industry”
² Figura extraída del informe de las ventas globales de la industria del videojuego en 2017 en Newzoo.com
Hoy día, los videojuegos tienen una mejor aceptación que hace años y además la oferta de acceso se ha multiplicado exponencialmente. La oferta de videojuegos actual es tan amplia que cualquier persona independientemente de su edad y situación socioeconómica podrá encontrar un juego al que jugar. Ya no hace falta tener una consola o un ordenador potente para poder jugar, existe una inmensa variedad de juegos disponibles para el móvil o incluso que se juegan online a través de un navegador cualquiera. Casi todo el mundo dispone de como mínimo una conexión a internet y un teléfono móvil.

Las consolas actuales ofrecen la posibilidad de jugar remotamente permitiendo conectarse a ellas por internet desde otro lugar\(^4\). Es decir, podrías jugar a la consola de tu casa desde la otra punta del mundo. Hay compañías como Google que están experimentando con ofrecer un servicio de streaming de videojuegos\(^5\) de tal forma que los juegos se ejecuten en sus servidores en la nube y los usuarios puedan jugar desde cualquier lugar, con un simple navegador web. Si todo sale bien, el plan es ofrecer un servicio de pago tipo Netflix con una cuota fija y acceso a un gran catálogo de videojuegos. Hay empresas como Electronic Arts mediante su plataforma Origin\(^6\), o PlayStation mediante PlayStation Now\(^7\) que ya han empezado a implementar un servicio similar.

---


\(^7\) PlayStation Now, servicio de suscripción de videojuegos ([https://www.vidaextra.com/ps4/ps-now-el-verdadero-netflix-de-los-videojuegos-recibe-los-primeros-titulos-de-ps4](https://www.vidaextra.com/ps4/ps-now-el-verdadero-netflix-de-los-videojuegos-recibe-los-primeros-titulos-de-ps4))
a Netflix. Los usuarios pagan una cuota de acceso y a cambio tienen acceso a un gran catálogo de videojuegos. Todos estos avances no hacen más que aumentar la oferta y disponibilidad. Permite que cualquier persona pueda jugar en cualquier momento y lugar a su juego favorito por poco dinero.

Además, hoy en día es más fácil que nunca el comprar un videojuego. Ya no hace falta tener que ir a una tienda física, si no que se pueden comprar online en diferentes portales online como Steam, Origin, o incluso mediante páginas web de intermediarios dedicadas como HumbleBundle, Voidu, etc. Es decir, podemos comprar un videojuego a cualquier hora y desde cualquier lugar.

Con tanta tienda online e intermediarios, se hace muy fácil el comparar precios. El comprador además puede comprar en tiendas de otras regiones propiciando que, la diferencia de cambio de la moneda puede resultar muy favorable al comprador permitiéndole ahorrar mucho dinero. Y si a eso le sumamos que estos portales y páginas web suelen hacer ofertas con asiduidad, el coste de adquirir videojuegos ha caído en picado. Estas tiendas también suelen ofertar juegos de manera gratuita durante unos pocos días para atraer a potenciales compradores nuevos.

Por ponerlo en perspectiva, si antiguamente con 60€ podíamos acceder a comprar 1 videojuego triple AAA, hoy es factible el poder conseguir 10 juegos triple AAA por el mismo precio esperando a las rebajas y mirando diferentes tiendas online. Incluso podríamos obtener 20 juegos, o más, si comprásemos juegos en bundles\textsuperscript{8} y juegos de compañías indie.

Esta situación ha provocado que sea muy fácil llenar la biblioteca virtual de juegos por muy poco dinero o incluso gratis. Esto hace que sea muy difícil manejar la enorme cantidad de videojuegos de los que dispone un jugador, puesto que, aunque la oferta de videojuegos tiende al infinito, el tiempo disponible para jugar es muy limitado. A este fenómeno se le conoce como backlog\textsuperscript{9}. Hay muchos usuarios que disponen de un gran backlog\textsuperscript{10}, y no saben cómo gestionar todos los juegos que tienen pendientes. A raíz de esto, han surgido páginas como Howlongtobeat.com que recopilan datos de los usuarios acerca de cuánto tiempo han tardado en pasarse un videojuego. Muchas personas utilizan estas páginas para calcular cuánto tiempo deberían invertir para terminar un videojuego e intentar organizar su backlog.

---

\textsuperscript{8} Bundles, cuando una página web agrupa una colección de varios videojuegos en un pack.

\textsuperscript{9} Backlog, se llama backlog a la acumulación de videojuegos pendientes por terminar.

\textsuperscript{10} Una simple búsqueda en Google permite observar la gran cantidad de gente que sufre de este problema: [https://www.google.com/search?q=too+much+gaming+backlog](https://www.google.com/search?q=too+much+gaming+backlog)
2.2 Objetivos

Este problema es el motivo que me ha llevado a plantear mi TFG. Este TFG pretende crear una aplicación que permita al usuario poder organizar su colección de videojuegos que además suele estar esparcida entre juegos físicos y digitales, y poder llevar un control sobre su colección y su tiempo de ocio.

Esta aplicación centralizará la colección de videojuegos (tanto físicos como digitales) del usuario, y le ofrecerá estadísticas de cuantos videojuegos dispone, los que ya tiene completados, y los que están en medias y sin empezar. También permitirá al usuario organizar su tiempo de ocio. Este podrá ver cuánto tiempo hay que invertir en un videojuego para terminarlo, y será capaz de saber cuánto tiempo le haría falta para terminar un videojuego ya empezado. Estos datos, le permitirán filtrar, organizar y encontrar nuevos juegos que no ha completado, a los que jugar según las horas de las que disponga para ello. Esto hará que su tiempo de ocio sea óptimo y pueda disfrutar de él en vez de tener que estar preocupándose de a que jugar.

También se va a aprovechar la aplicación para mostrárselo al usuario noticias relevantes relacionadas sobre el mundo de los videojuegos, así como permitirle encontrar tiendas físicas cercanas a su posición donde poder comprar más videojuegos en formato físico, o mostrársle algunas de las mejores ofertas digitales que están disponibles para que no se le escape ninguna oferta.
3 Trabajos previos (Estado del arte)

Existen bastantes productos que ofrecen servicios similares. En este apartado voy a realizar un estudio de algunos de los productos que he encontrado:

1- Grouvee
2- The backloggery
3- CLZ Games: Game Database

3.1 Grouvee

“Grouvee” es una página web disponible en: [https://www.grouvee.com/](https://www.grouvee.com/) Esta página web tras registrarte te ofrece un servicio para controlar tu colección de videojuegos y backlog. En la pantalla principal es donde se permite al usuario buscar juegos y añadirlos a su colección de manera manual (Figura 3). Estos juegos se pueden añadir como: completados, jugando, wishlist (juegos a los que deseas jugar), o backlog.

![Figura 3 – Página principal donde añadir juegos. (Fuente: Captura de pantalla propia de Grouvee)](image)

Incluye una opción para visualizar la librería de videojuegos (Figura 4). Se pueden filtrar los videojuegos por tipo, ordenar por categorías, plataformas, etc. Además, tiene una opción “Tell me what to play” en la que se te recomienda un videojuego de tu lista de manera aleatoria que no has marcado como completado.
Por último, la página web dispone de un foro, un componente social, donde los usuarios pueden charlar y hablar sobre videojuegos.

3.2 The backloggery

“The backloggery” sería otro ejemplo de página web que permite llevar un control del backlog. Está disponible en [https://backloggery.com](https://backloggery.com) Esta página web permite agregar videojuegos de manera manual. Además, ofrece muchas más opciones detalladas de progreso y versionado que la página web anterior (Figura 5).
Como Grouvee, te permite ver el listado de los videojuegos que has añadido (Figura 6), y te permite filtrarlos por sistema, y por estado (terminado, en progreso, etc.). Como novedad, esta página web permite que otros usuarios puedan enviarte comentarios sobre tu backlog. También ofrece como Grouvee una opción que te recomienda un juego al que jugar. De nuevo vuelve a ser aleatorio, solamente que te permite seleccionar una serie de filtros como, por ejemplo, juegos que no has jugado, que estás jugando o que están en tu wishlist (lista de deseos), o permitirte elegir la plataforma a la que jugar. A esta opción se le denomina “Fortune Cookie” (Figura 6).
3.3 CLZ Games: Games Database

Por último, vamos a ver una aplicación para Android que también se encarga de gestionar la colección de videojuegos del usuario. Esta aplicación móvil está disponible en la Play Store en el siguiente enlace:


Esta aplicación es freemium\(^\text{11}\). La versión “free” está limitada a una colección de 100 videojuegos. La versión “premium” cuesta 14.99€ y no tiene restricciones de ningún tipo en la colección. Esta aplicación como todas las demás, permite agregar juegos de manera manual, y ofrece la posibilidad de añadir juegos mediante búsqueda online o escaneando el código de barras (Figura 7). El usuario podrá guardar su base de datos en la nube en los servidores de CLZ. También añade un método nuevo para agregar videojuegos en formato físico, mediante el escaneo del código de barras (Figura 7). Nuevamente la aplicación ofrece la visualización de la librería de videojuegos y tiene diferentes filtros permitiendo por ejemplo buscar videojuegos por plataforma.

Podemos observar que existen numerosas aplicaciones que se encargan de gestionar la colección de videojuegos del usuario y su backlog. Cada una de ellas lo hace de una manera diferente y ofrece servicios diferentes. La aplicación propuesta en este TFG, expande estas opciones, añadiendo más versatilidad y funcionalidades extras.

\(^\text{11}\) Freemium, viene de la fusión de las palabras “free” y “premium”. Una aplicación “Freemium” es a la vez gratuita y de pago. La parte gratuita o “free” suele estar limitada, y ofrece funcionalidades básicas, mientras que la parte “premium” es de pago, y ofrece la versión completa sin limitaciones.
4 Diseño de la aplicación

4.1 Requisitos

Para diseñar la aplicación primero he definido la lista de requisitos para la aplicación, partiendo de la lista de tareas de la primera entrega. Los requisitos funcionales obtenidos para la aplicación son los siguientes:

1- Versión mínima de Android: Api 23 (6.0)
2- Añadir cuenta de Steam\(^{12}\)
3- Importar la lista de juegos de Steam del usuario
4- Recuperar información de la lista de juegos de Steam (Tiempo jugado, logotipo)
5- Añadir juegos manualmente
6- Visualizar la lista de videojuegos
7- Editar un juego de la lista
8- Borrar un juego de la lista
9- Encontrar información del tiempo promedio\(^{13}\) para un videojuego
10- Importar la información del tiempo promedio y asociarlo a cada videojuego de la librería
11- Método para encontrar un juego aleatorio basándose en el tiempo de juego del que dispone el usuario
12- Visualizar toda la lista de juegos disponibles para el usuario en base al tiempo de juego del que dispone
13- Encontrar ofertas de videojuegos digitales \(^{14}\)
14- Mostrar ofertas digitales al usuario
15- Encontrar la posición mediante geolocalización del usuario
16- Mostrar tiendas físicas cercanas al usuario
17- Recuperar noticias de videojuegos utilizando RSS
18- Mostrar noticias de videojuegos al usuario
19- Añadir un videojuego físico mediante el uso de NFC/OCR/Código de Barras

\(^{12}\) Steam, (https://store.steampowered.com/) es una plataforma desarrollada por "Valve Corporation". Esta plataforma es actualmente la mayor distribuidora de videojuegos en formato digital, así como de servicios multijugador y comunicaciones.

\(^{13}\) Para controlar el tiempo promedio de finalización de un videojuego se utilizará el servicio "HowLongToBeat" (https://howlongtobeat.com/) donde se recopilan estas estadísticas.

\(^{14}\) Para encontrar las ofertas digitales se usará el servicio de "IsThereAnyDeal" (https://isthereanydeal.com/) donde se recopilan ofertas de videojuegos.
4.2 Flujo y diseño de pantallas

Con los requisitos definidos, el siguiente paso fue diseñar la vista preliminar de las pantallas y el flujo de la aplicación. Para ello, se definió primero el flujo básico de la aplicación teniendo en cuenta los requisitos. El flujo básico de la aplicación (Figura 8) consta de 13 pantallas.

La primera pantalla es de bienvenida, desde la cual el usuario puede acceder a un menú principal. En este menú principal, el usuario va a disponer acceso a las funcionalidades básicas de la aplicación: Ver su librería de videojuegos; Organizar su tiempo; Encontrar tiendas cercanas; Ver ofertas digitales; Ver Noticias.

Cuando el usuario entre en la librería de videojuegos dispondrá a su vez de la opción de: ver su librería de videojuegos; Añadir su cuenta de Steam; Añadir un videojuego; Obtener el tiempo promedio de su colección de videojuegos. El usuario podrá ver su librería entera de videojuegos. Desde esta vista, el usuario dispondrá de una forma de editar el juego que seleccione para, por ejemplo, añadir tiempo de juego o marcarlo como completado. También podrá borrar un juego de su colección.

Si el usuario entra en organizar su tiempo, podrá buscar juegos al azar que cumplan sus criterios en base al tiempo del que disponga o un listado con todos los juegos que se han encontrado.
A continuación, se encuentran los diseños preliminares de las 13 pantallas.

Figura 9 – Pantalla de bienvenida
Figura 10 – Menú principal
Figura 11 – Menú “Tu librería”

Figura 11 – Ver librería de juegos
Figura 12 – Editar juego
Figura 13 – Añadir cuenta Steam
Figura 14 – Añadir videojuego
Figura 15 – Búsqueda tiempo
Figura 16 – Organizar tu tiempo

Figura 17 – Resultados encontrados
Figura 18 – Tiendas cercanas
Figura 19 – Ofertas digitales
Con el flujo determinado y el diseño de las pantallas preliminares, se procedió a determinar que servicios se iban a usar para la aplicación. En los siguientes apartados se examinarán los servicios que se escogieron.
4.3 Steam

Steam es una plataforma de videojuegos digital del grupo “Valve Corporation”. Esta plataforma es actualmente la mayor distribuidora de videojuegos en formato digital, así como de servicios multijugador y comunicaciones.

![Figura 21 – Pagina web principal de Steam](https://store.steampowered.com/)

Esta plataforma cuenta con unos 55 millones de usuarios activos, de los cuales, 31.5 millones de usuarios han estado activos en las últimas 2 semanas según las estadísticas de SteamSpy. Además, la media de juegos por usuario es de unos 75.5 juegos.

Con estos datos y sabiendo que Steam es la mayor plataforma de videojuegos, es lógico que los usuarios que juegan a videojuegos de PC quieran importar su librería de Steam. Steam cuenta con una API con la que realizar consultas, por lo tanto, es lógico ofrecer al usuario la posibilidad de importar su colección de videojuegos de Steam.

15 Captura de pantalla principal sacada de [https://store.steampowered.com/](https://store.steampowered.com/).
16 SteamSpy, [https://steamspy.com/](https://steamspy.com/) es una página web que ofrece estadísticas de Steam.
4.4   HowLongToBeat

Uno de los pilares fundamentales es poder ayudar a los usuarios de nuestra aplicación a gestionar su tiempo de ocio. Para ello necesitamos saber cuando tiempo en promedio es necesario para acabar un videojuego.

Existen diferentes páginas web que ofrecen este servicio, destacando:

- HowLongToBeat (HLTB) (https://howlongtobeat.com/)
- IGDB (https://www.igdb.com)

![Figura 22 – Página web principal de HowLongToBeat [Fuente: Captura de pantalla]](image)

Tras examinar ambas páginas web y que opciones ofrecen, en un primer momento decidí usar IGDB puesto que a pesar de que su base de datos es menor que HLTB cuenta con una API\(^\text{18}\) con opción de uso gratuita. La opción gratuita está limitada a 3k consultas al mes\(^\text{19}\). Tras un par de pruebas preliminares, determine que no sería suficiente, puesto que, manejando librerías

\(^{18}\) API IGDB, (https://api.igdb.com/) La página web de la API de IGDB.

\(^{19}\) Actualmente han aumentado el número de peticiones para la oferta gratuita de 3k a 10k peticiones al mes.
grandes de unos 800 juegos, agotaría las 3000 consultas al mes con relativa facilidad. La opción de pago cuesta 99$ y ofrece 50k consultas al mes. Debido a los altos costes de explotación se me hace inviable usar IGDB.

Tras esto decidí volver a examinar HowLongToBeat. HowLongToBeat no cuenta con ninguna API que permita consultar su página web de forma fácil. Por lo que para usar su página web y poder obtener la información que necesito voy a tener que *scrapear* la página web.

Para ayudarme en las tareas de scraping, estuve buscando alguna librería auxiliar en Java que me ayudase. Encontré la librería *JSOUP*\(^\text{21}\), que es una librería JAVA que permite el parseado HTML. Tras un par de pruebas positivas, se estableció el uso de la página web HowLongToBeat para obtener los tiempos promedios para completar un videojuego.

---

20 *Scrapear* proviene del término “Web scraping” y es una técnica para extraer información de sitios web. Utilizando la ingeniería inversa y simulando la navegación se pueden obtener los resultados deseados analizando el código HTML resultante que devuelve la página web.

21 *JSOUP*, [https://jsoup.org/](https://jsoup.org/) librería JAVA para el parseado de webs HTML.
4.5 IsThereAnyDeal

Otro de los requisitos de la aplicación es poder mostrarle al usuario ofertas digitales. Tras buscar diferentes páginas web que ofrezcan este servicio, la cosa quedó dividida entre 2 posibles candidatos:

- IsThereAnyDeal (https://isthereanydeal.com/)
- CheapShark (https://www.cheapshark.com/)

![Figura 23 – Arriba: página web principal de CheapShark; Abajo: página web principal de IsThereAnyDeal][Fuente: Captura de pantallas]
Ambas paginas son buenas candidatas, ya que ofrecen exactamente lo que busco, y ademas disponen de API\textsuperscript{22}. Tras realizar varias pruebas con ambas API, la web de CheapShark se queda un poco corta en comparación.

La web de IsThereAnyDeal es más completa, y su API esta mejor documentada. Ofrece funcionalidades avanzadas como graficas de precios, y comparativa entre diferentes tiendas, así como un historico de precios. Ademas, dispone de la posibilidad de registro y permite que los usuarios metan los juegos que desean y obtener notificaciones cuando estos bajen de precio. Esta funcionalidad extra no es algo que actualmente necesite, pero podría venir bien para versiones futuras de la aplicacion.

\textsuperscript{22} CheapShark API, más información de la API disponible en (https://www.cheapshark.com/api/); IsThereAnyDeal API, más información de la API disponible en (https://www.cheapshark.com/api/)
4.6 Noticias RSS

Por último, necesito encontrar páginas web que ofrezcan noticias mediante RSS de videojuegos. Tras la búsqueda de feeds RSS, obtuve los siguientes candidatos:

- Gamel informer, https://www.gameinformer.com/rss
- GameSpot, https://www.gamespot.com/feeds/
- Game Reactor, https://www.gamereactor.es/rss/

Tras examinar todos los feeds, decidí usar los de GameReactor y Game Informer. El resto de feeds fueron descartados, pero se podrían incluir en una futura versión, donde el usuario pudiera elegir que feeds quiere recibir e incluso permitir al usuario meter feeds propios.
5 Desarrollo de la aplicación

5.1 Herramientas

Para el IDE se va a usar Android Studio 3.2\(^{23}\), ya que tiene varias funcionalidades interesantes. Cuenta con la tecnología de “Instant Run” que permite hacer cambios de código instantáneos sin tener que volver a lanzar el programa en el emulador. Esto resulta muy útil, por ejemplo, cuando se están haciendo cambios mínimos al diseño de una pagina en concreto puesto que permite visualizar el efecto en tiempo real casi sin demora. Incluye una serie de modelos (templates) de código que otorgan al usuario un esqueleto de código desde el que empezar a desarrollar. Dispone de un editor bastante intuitivo para poder diseñar las vistas, y permite analizar en tiempo real el impacto y uso de la CPU / MEMORIA / RED.

Para el control de versiones, se ha elegido Bitbucket$^{24}$ junto con la aplicación SourceTree$^{25}$. Bitbucket ofrece un servicio similar al ofrecido por GitHub, con la salvedad de que permite tener repositorios privados gratuitos. Bitbucket ofrece todas las ventajas de un sistema basado en GIT permitiendo que se pueda ir salvaguardando el estado de la aplicación con los cambios y poder volver atrás si fuera necesario. SourceTree es una aplicación que dota de un interfaz gráfico a GIT, permitiendo un funcionamiento mucho más sencillo e intuitivo que usando la consola de comandos.

![Figura 25 – Vista de la aplicación de SourceTree [Fuente: Captura propia]](image)

Para la copia de seguridad, se va a utilizar el propio Bitbucket. Al subir el repositorio a Bitbucket, dispondremos de una copia en la nube, aparte de nuestra copia en local. Además, se va a realizar una copia semanal a un disco duro externo de la carpeta de desarrollo local con los últimos cambios.

Para la edición de imágenes se va a recurrir al programa de Paint.net$^{26}$, programa de edición de imágenes gratuito bastante potente. Por último, se va a utilizar Postman$^{27}$. Postman permite probar los parámetros, cabeceras, etc. de las peticiones a servicios API de una manera rápida.

$^{24}$ Bitbucket, [https://bitbucket.org/product](https://bitbucket.org/product) es un gestor de proyectos basado en GIT. Similar en funcionamiento a GitHub.

$^{25}$ SourceTree, [https://www.sourcetreeapp.com/](https://www.sourcetreeapp.com/) es un programa con una interfaz gráfica que permite gestionar proyectos basados en GIT, como por ejemplo aquellos alojados en Bitbucket de manera sencilla e intuitiva.

$^{26}$ Paint.net, [https://www.getpaint.net/](https://www.getpaint.net/) es una aplicación de edición de imágenes gratuitas.

$^{27}$ Postman, [https://www.getpostman.com/](https://www.getpostman.com/) es una aplicación que permite probar las conexiones a APIs o a páginas web HTML con total control sobre los parámetros, cabeceras de las peticiones.
5.2 Metodología

Para la realización de la aplicación se ha seguido una metodología ágil SCRUM\(^{28}\). La metodología establece la división del proyecto en pequeños ciclos llamados sprints. Esto requiere analizar el proyecto previamente, definiendo las tareas y requisitos del proyecto. Con esto se crea un backlog de tareas que se deben hacer. Para cada sprint, se determina el paquete de tareas, normalmente asociado a una determinada funcionalidad, y un plazo de tiempo. En ese plazo de tiempo se desarrolla la funcionalidad, se prueba, y se integra aumentando la funcionalidad del software (Figura 26).

\[\text{Figura 26 – Vista general de un proceso SCRUM [Fuente: JeronimoPalacios}^{29}\]}

Cuando hice la planificación del proyecto para la primera entrega, tuve esto en cuenta. Deje la planificación dividida en funcionalidades, de tal forma que me fuera más fácil luego planificar los sprints, sabiendo el scope y duración para cada sprint mirando el diagrama de Gantt.

El diagrama de Gantt se encuentra en la primera entrega, aunque se puede consultar en la siguiente dirección:

https://drive.google.com/file/d/1eZQU75kRHEGVjHrouxqZv9Zx4rBq91RY/view

\(^{28}\) SCRUM, (https://proyectosagiles.org/que-es-scrum/) es un tipo de metodología para la realización de proyectos. Se basa en un proceso iterativo donde se va completando funcionalidades poco a poco.

\(^{29}\) Figura sacada de la página web https://jeronimopalacios.com/srum/
5.3 Desarrollo frontend

En este apartado analizaremos en profundidad cómo desarrollé el frontend de la aplicación. Lo primero que hice fue definir la paleta de colores. Para la paleta de colores utilicé la página web Coolors para generar una paleta de colores. Tras probar varios tipos de paletas, me decidí por usar la siguiente:

![Imagen de la paleta de colores elegida](image)

**Figura 27 – Paleta de colores elegida [Fuente: Paleta de colores extraída de Coolors]**

De la paleta acabe usando los 3 primeros: #C61909, #CA200C, #E65920, persiguiendo una aplicación llamativa donde predominaran los colores rojizos. Al ser una aplicación que va destinada al mundo de los videojuegos, he intentado mantener una estética “gamer” llamativa.

Para la codificación de las pantallas se ha usado el formato XML. En este formato cada pantalla o vista, corresponde con un archivo en XML donde se detalla qué elementos hay en la vista, así como su posición y propiedades. Gracias al potente editor integrado en Android Studio, se puede ver una vista previa de cómo quedaría la vista (**Figura 28**). Además, permite editar el XML tanto a mano (**Figura 29**), como utilizando una GUI donde se pueden ver todas las opciones (**Figura 28**).

---

30 *Frontend*, es la parte de software que interactúa con el usuario. Es lo que ve el usuario en pantalla.
31 Coolors, ([https://coolors.co/](https://coolors.co/)) es una página web que permite al usuario generar una paleta de colores que se complemente.
Si se observa la figura 28, se puede ver como a la izquierda, en el “Component Tree” encontramos todos aquellos elementos que conforman la vista. En el centro aparece la previsualización de la vista, y se puede ver como esta organizado con respecto a otros elementos, así como los márgenes que hay de una manera rápida. A la derecha tenemos los “Attributes” que son las propiedades de ese elemento. En este caso esta seleccionado un “TextView” que se corresponde con una etiqueta de texto.
Si se quiere se puede editar la vista en modo XML (**Figura 29**), donde se nos muestra a la izquierda, el código XML que conforma la vista y a la derecha una vista previa del código XML. Esta dualidad en el editor es muy potente puesto que permite complementar la facilidad de uso del editor en formato GUI con la potencia de poder escribir el código directamente en XML para ajustar más a fondo ciertos aspectos.

Por ejemplo, para esta vista, tuve que agregar el siguiente fragmento de código XML (**Figura 30**) para que al entrar no tuviera el foco el primer `EditText` 32.

```
<EditText
    android:descendantFocusability="beforeDescendants"
    android:focusableInTouchMode="true"

Figura 30 – Código agregado al XML. [Fuente: captura propia]
```

A continuación, se muestran las diferentes pantallas que se han confeccionado para esta aplicación:

---

32 *EditText*, se refiere a un elemento que permite al usuario introducir datos.
1 - Pantalla de bienvenida:

Esta pantalla es la pantalla de bienvenida que se le presenta al usuario cuando inicia la aplicación. En ella se hace una breve presentación con el logo de la aplicación, se muestra la versión actual y dispone de un botón para que el usuario entre en la aplicación.

![Figura 31 – Pantalla principal de bienvenida [Fuente: Captura app]](image)

2 – Menú principal:

Este es el menú principal donde el usuario tendrá acceso a las distintas opciones principales. Los botones se han diseñado con imágenes evocativas de la temática de videojuegos.

Desde esta pantalla el usuario puede acceder a las 4 grandes opciones de la aplicación:
- Su librería
- Organizar su tiempo
- Noticias
- Tiendas Cercanas
- Ofertas Digitales

Como se puede ver, además se ha agregado una quinta opción que no estaba prevista originalmente llamada “Configuración”.

![Figura 32 – Menú principal [Fuente: Captura app]](image)
3 – Tu librería:

Esta es la pantalla donde se le muestra al usuario un breve resumen de su librería. Puede ver de cuántos juegos dispone su librería, así como el número de juegos completados y los juegos pendientes. Además se le muestra el número de juegos de los que se han podido recuperar estadísticas HLTB. Además, se le muestra un recordatorio de la cuenta de Steam que tiene vinculada.

Tiene acceso a 4 opciones:

- Ver su librería de videojuegos
- Añadir una cuenta de steam
- Añadir juego manualmente
- Obtener estadísticas HLTB

3.1 – Ver librería:

En esta pantalla se le muestra al usuario su colección de videojuegos completa por orden alfabético. Además, se dispone de 2 pestañas adicionales, donde el usuario puede consultar rápidamente los juegos que tiene pendientes y los que ha completado.

Para cada juego se le muestra un breve resumen con su título, el tiempo que lleva jugado, la plataforma a la que pertenece, si lo ha completado o no, y el tiempo promedio de HLTB si se ha encontrado para ese videojuego.

Además puede editar o borrar el videojuego si así lo desea.
3.1.1 – Editar juego:

En esta pantalla el usuario puede editar el videojuego para incluir, por ejemplo el numero de horas que ha jugado, o si lo ha completado. Tambien puede cambiar el nombre del juego y la plataforma.

Figura 35 – Editar datos de un juego [Fuente: Captura app]

3.1.2 – Borrar juego:

Cuando un usuario quiere borrar un juego se le presenta este popup donde el usuario puede decidir si quiere borrar un videojuego de su libreria o no.

Figura 36 – Borrar juego [Fuente: Captura app]
3.2 – Añadir cuenta de Steam:

Esta pantalla permite al usuario ingresar su cuenta de Steam. Para ello, se le da una serie de instrucciones para que encuentre su SteamID64 en caso de que no lo tenga.

![Añadir cuenta de Steam](Figura 37 – Añadir cuenta de Steam [Fuente: Captura app])

3.3 – Añadir juego:

Esta es la pantalla que permite al usuario agregar un videojuego de forma manual en caso de que lo tuviera en formato físico.

Se le pide rellenar el nombre, plataforma como opciones obligatorias, y luego podría meter las horas jugadas así como si lo ha terminado.

![Añadir juego](Figura 38 – Añadir juego [Fuente: Captura app])
3.4 – Encontrar estadísticas HLTB:

Cuando un usuario quiere encontrar nuevas estadísticas HLTB para su librería le aparece este popup. En el se muestra el progreso, con un resumen de los juegos que ya están procesados y disponen de estadísticas HLTB y el total. Además, se le permite al usuario realizar esta tarea en segundo plano, dejando el proceso que corra en background.

Figura 39 – Buscar información HLTB [Fuente: Captura app]

4 – Organizador de tiempo:

Cuando un usuario quiere encontrar un juego, se le muestra esta pantalla. Para ello selecciona cuanto tiempo dispone y la plataforma en la que quiere jugar y la aplicación le buscará juegos que cumplan dicho criterio.

Figura 40 – Organizador de tiempo [Fuente: Captura app]
4.1 – Juegos encontrados:

Al usuario se le muestra un juego al azar que se ha encontrado que cumple que se puede terminar en el tiempo del que dispone. Se le muestra un breve resumen del juego, con su título, una carátula, tiempo que lleva jugado, y cuanto tiempo lleva de media terminar el videojuego.

En caso de que el usuario quiera otro juego, puede pulsar el botón mostrar otro juego donde se le mostrara otro juego al azar, o ver una lista completa con todos los juegos que se han encontrado.

4.1.1 – Listado de juegos encontrados:

Si el usuario quiere ver todos los resultados encontrados, se le muestra la siguiente pantalla, donde tiene una lista de todos los juegos que cumplen ese criterio, ordenada por orden alfabético.

Para cada juego se le muestra un breve resumen, con su título, carátula, plataforma, tiempo que lleva jugado, y tiempo medio que le llevaría terminar el juego.
5 – Noticias:

Cuando el usuario quiera buscar nuevas noticias, se le muestra una lista donde se le muestran las noticias que se han recuperado de los diferentes RSS intercaladas.

Se muestra una breve imagen, una fecha y un título. Si el usuario quiere saber más acerca de una noticia, basta con que pulse en ella.

5.1 – Noticias navegador interno:

Si un usuario quiere más detalles sobre una noticia, al pulsar se le abre esta pantalla. Es un navegador interno, donde el usuario puede leer la noticia que le interese sin salir de la aplicación.
6 – Tiendas cercanas:

Esta es la pantalla que se le muestra al usuario cuando quiere encontrar tiendas cercanas. Por defecto, intenta tomar la posición GPS del usuario, aunque a veces se toma la de la dirección IP.

El usuario puede centrarse en su posición GPS usando el botón circular que se encuentra en el margen inferior derecho de la imagen. También puede navegar en una lista de tiendas que ha encontrado desplegando el desplegable de abajo, o pinchando si pincha sobre una tienda podrá encontrar más información.

Figura 45 – Arriba: Tiendas cercanas; Abajo: Detalle de información de una tienda
[Fuente: Captura app]
7 – Ofertas digitales:

Esta pantalla es la que le muestra al usuario las ofertas de videojuegos en formato digital que se han encontrado. Muestra un breve resumen con el nombre del juego, el precio actual y el precio normal entre parentesis. También se muestra el % de descuento que tiene el juego, y la plataforma donde se encuentra la oferta. Si un usuario quiere comprar el juego puede pulsar sobre el botón del carrito y se le lleva a la página para que lo pueda comprar.

Figura 46 – Ofertas digitales
[Fuente: Captura app]

7.1 – Comprar videojuego digital:

En el caso de que el usuario quiera comprar un videojuego se le abre su navegador favorito y se le redirige a la página web de la oferta para que pueda comprarlo. El usuario siempre puede volver hacia atrás y volver a GameItUp en cualquier momento.

Se realiza en el navegador externo por motivos de seguridad.

Figura 47 – Noticia ampliada
[Fuente: Captura app]
8 – Configuración:

Esta última pantalla muestra las opciones de las que dispone el usuario para la configuración de la aplicación. En esta iteración de la aplicación dispone de las opciones de borrar toda su librería en caso de que quiera empezar de nuevo, o borrar la cache HLTB.
5.4 Desarrollo backend

Tras haber examinado el desarrollo del frontend, vamos a pasar a ver el desarrollo del backend en este apartado. La aplicación ha seguido el patrón de diseño: Modelo – Vista – Controlador (MVC).

Vistas:

En la figura 49 se puede apreciar el listado de vistas completo mostrando todos los layouts correspondientes a esta aplicación. Las vistas corresponden a la interfaz que son las pantallas que ve el usuario tal y como se ha podido ver en el apartado anterior 5.3.

Figura 49 – Listado de archivos XML que forman las vistas [Fuente: Captura propia]

34 Layouts, se refiere al diseño de una vista en la aplicación. En el caso de Android son archivos codificados en XML.
Aquellos layouts que empiezan por “activity” corresponden a las vistas de las actividades. Los que empiezan por “fragment” se corresponden con los fragmentos y por lo último los que empiezan por “layout” se corresponden con los que utilizan los adaptadores para mostrar lista de resultados. Por último, hay que destacar el que empieza por “news”, que es el que muestra el listado de noticias, que es el diseño en fila de una noticia.

**Modelos:**

La aplicación cuenta con 7 modelos como se puede apreciar en la figura 50. Estos modelos se corresponden con diferentes tipos de datos que se tratan y usan en la aplicación:

- **Deal:** Es el modelo de datos para las ofertas de videojuegos. Contiene todos los campos que recuperamos del servicio IsThereAnyDeal para cada oferta de videojuegos que encontramos.

```java
public class Deal {
    String plainName;
    String name;
    String price_new;
    String price_old;
    String discount;
    String shopName;
    String url_buy;
    String url_moreinfo;
}
```

- **Game:** Es el modelo de datos para un videojuego. Contiene todos los campos que nos hacen falta de cara a mostrarte la información de un juego a tu usuario.

```java
public class Game {
    private String id;
    private String name;
    private Platform platform;
    private String year;
}
```
- **HLTBGameStats**: Es el modelo de datos que utilizamos para almacenar los datos que recuperamos de HowLongToBeat una vez escrapeados.

```java
public class HLTBGameStats implements Serializable {
    String gameName;
    String hltb_id;
    ArrayList<HLTBGameTime> gameStats;
    String hltb_image_url;
    String gameTime;
}
```

- **HLTBGameTime**: Es el modelo de datos que contiene los datos acerca del tiempo medio que hemos recuperado de HLTB.

```java
public class HLTBGameTime implements Serializable {
    String playstyle;
    String time;
}
```

- **ScrapperData**: Este es el modelo de datos que le pasamos al Scraper desarrollado con los datos del juego que queremos buscar.

```java
public class ScrapperData {
    ScrapperDelegate delegate;
    HLTBQueryType queryType;
    String searchName;
    String hltb_GameId;
    int position;
}
```

- **Platform**: Es un enumerador que contiene las diferentes plataformas de las que el usuario puede agregar videojuegos.

```java
public enum Platform { PC, PS4, PS3, XBOX360, XBOXONE, SWITCH }
```

- **HLTBQueryType**: Es un enumerador que sirve para definir el tipo de búsqueda que se quiere hacer para un videojuego.

```java
public enum HLTBQueryType { GET_GAME_DATA, FIND_GAME_DATA }
```
Controladores datos:

La aplicación cuenta con varios tipos de controladores. Por un lado, tiene una serie de controladores para los datos internos y conexiones y por otro lado otros para la interacción del usuario con las vistas.

Hay 5 controladores internos como se puede ver en la figura 51 que se encargan de la parte de los datos internos y conexiones que necesita la aplicación.

![Figura 51 – Listado de controladores internos [Fuente: Captura propia]](image)

- **DataManager**: 
  Se encarga de controlar los datos internos de la aplicación. Es el núcleo de la aplicación puesto que cuando otro controlador externo requiera acceder a los datos o realizar una conexión para recuperar información, llamara al DataManager que se encargara de ocultar el proceso y devolver al controlador que lo ha llamado los datos que necesita de manera transparente.

  El DataManager por lo tanto controla tanto la base de datos interna con la librería del usuario, la cache de estadísticas HLTB, así como las conexiones necesarias al exterior.

- **ConnectionManager**: 
  Cuando se requiere recuperar o establecer una conexión con el exterior, el DataManager llamara al ConnectionManager, que se es el encargado de realizar la conexión. Este controlador usara la librería Volley\(^{35}\) para realizar dicha petición, tratando los posibles errores y comunicando al

\(^{35}\)Volley, [https://github.com/google/volley](https://github.com/google/volley) es una librería auxiliar desarrollada por Google que permite al programador Android realizar conexiones HTTP más rápidas y fácilmente.
DataManager el resultado de la operación, así como los datos recuperados. Estos datos normalmente vendrán en un formato JSON, que tendrá que ser parseado después.

- **VolleyManager:**

  Es un controlador auxiliar para el ConnectionManager y se utiliza para gestionar las peticiones de tipo Volley que realiza dicho controlador.

- **DatabaseManager:**

  Este controlador será el responsable de manejar la base de datos interna. Para ello se usará la clase `SharedPreferences` interna de Android. DatabaseManager será llamado por parte del DataManager cuando haya que recuperar o guardar datos en la base de datos interna.

- **Scrapper:**

  Es el responsable de scrapear y encontrar la información requerida de la página web HowLongToBeat y pasársela al DataManager. Para su diseño tuve que analizar el funcionamiento interno de la página web HowLongToBeat. Empecé examinando el código HTML de la página web y cómo funcionaba una búsqueda.

![Image](https://developer.android.com/guide/topics/data/data-storage)

**Figura 52 – Peticiones realizadas al buscar un videojuego en HLTB [Fuente: Captura propia]**

---

36 `SharedPreferences`, es un tipo de forma de almacenar datos internamente como se puede consultar en la siguiente página web: [https://developer.android.com/guide/topics/data/data-storage](https://developer.android.com/guide/topics/data/data-storage)
Para ello utilice el depurador de Firefox y realice una búsqueda de un videojuego (Figura 52). Pude comprobar que la página web realiza una petición de tipo POST sobre la URL: “howlongotbeat.com/search_main.php?page=1”. También me permitió ver las cabeceras y parámetros necesarios para realizar dicha conexión (Figura 53).

Figura 53 – Cabeceras y parámetros pasados capturados [Fuente: Captura propia]
Con esta información pude hacer pruebas usando la herramienta de Postman para determinar los parámetros, cuerpo y cabeceras necesarias para poder buscar un videojuego y obtener la página web de resultados para poder “escraperla” (Figura 54).

Una vez fui capaz de llegar a buscar los juegos de manera correcta, el siguiente paso era diseñar el Scrapper correctamente para que pudiera extraer la información necesaria. Para ello, el Scrapper utilizando JSOUP (Como explique en el apartado 4.4) realiza una petición POST con los parámetros requeridos. El Scrapper entonces examina y analiza el código HTML devuelto por HLTB para extraer las partes que quiero.
De esta manera soy capaz de recuperar, la imagen del juego, las horas promedio requeridas para acabar dicho juego, así como el gameID interno que tiene el juego asignado en HowLongToBeat. Este gameID me permitirá en el futuro acceder a más información del videojuego si me fuera necesaria.

**Controladores de vistas:**

El resto de los controladores de la aplicación controlan la interacción del usuario con las vistas y los modelos de datos. Estos controladores se encuentran agrupados en carpetas por funcionalidad. En total existen 21 controladores distintos como se puede apreciar en la figura 55.

![Figura 55 – Controladores de vistas](image)

*Figura 55 – Controladores de vistas [Fuente: Captura propia]*
Adaptadores:

La aplicación además cuenta con varios adaptadores que son los encargados de mostrar los datos que se encuentran en formato de una lista al usuario. Existen 5 adaptadores, que se encargan de mostrar los diferentes listados de datos de la aplicación (*Figura 56*). Estos son:

- *GameDealRecyclerAdapter*: Muestra el listado de ofertas recuperadas.
- *GameLibraryCompletedRecyclerAdapter*: Muestra el listado de juegos de la librería que se han completado.
- *GameLibraryPendingRecyclerAdapter*: Muestra el listado de juegos de la librería que están pendientes de terminar.
- *GameLibraryRecyclerAdapter*: Muestra el listado de juegos de la librería entero.
- *TimeOrganizerRecyclerAdapter*: Muestra el listado de todos los videojuegos encontrados que cumplen con las restricciones de tiempo.

![Figura 56 – Adaptadores de datos [Fuente: Captura propia]](image)

Librerías y funciones auxiliares:

Existe una carpeta llamada “*Utils*” que contiene una serie de clases que contienen funciones auxiliares que se usan a lo largo de todo el proyecto. En total consta de 4 clases (*Figura 57*). Estas son:

![Figura 57 – Librería auxiliar [Fuente: Captura propia]](image)
Constants: Contiene todas las constantes que son utilizadas a lo largo de la aplicación (por ejemplo, la longitud del SteamID64).

NonScrollableViewPager: Un tipo vista ViewPager modificado que no permite el scroll.

Tags: Contiene la definición de todas las etiquetas que se usan para mostrar datos por la consola para comprobar el correcto funcionamiento.

Utils: Contiene pequeñas funciones auxiliares que son comunes entre varias clases. (Por ejemplo, para convertir los minutos en horas o viceversa)

Delegados (Callbacks):

La aplicación hace uso de distintos delegados. Estos delegados se utilizan para poder realizar la funcionalidad de Callback\textsuperscript{37}. Existen un total de 6 interfaces de delegado en la aplicación (Figura 58).

Es necesario usar los Callbacks, puesto que a veces una actividad requiere de datos que vamos a tardar en recibir. Por ejemplo, cuando buscamos las noticias, entre que se conecta, se descargan y se parsean los RSS pueden transcurrir varios segundos. Si no usáramos un Callback, el thread principal de la actividad se quedaría parado, haciendo que la aplicación se encontrase congelada. Mediante el uso de Callbacks, el thread principal de la actividad no se queda colgado, y cuando estén los datos listos, los recibirá en una función que actuara en consecuencia mostrando los datos recuperados o el error que se hubiera producido.

\textsuperscript{37} Callback, es una función (A) que se usa como argumento de entrada en otra función (B). Cuando la función (B) termina su trabajo, llama a la función (A) para que esta ejecute lo que necesite, normalmente en función del resultado de la función (B).
Parsers:

Se necesitan adaptar los datos que se reciben de las consultas externas tanto de Steam y IsThereAnyDeal a datos que pueda utilizar la aplicación en su modelo correspondiente. Para ello se hace uso de los parsers. En total la aplicación cuenta con 2 parsers ([Figura 59]) uno para transformar los datos recuperados de la librería de Steam del usuario y otro para las ofertas de IsThereAnyDeal.

Estos parsers toman el JSON devuelto por el servidor, y lo recorren extrayendo la información pertinente a su modelo correspondiente.

API de Steam:

Para recuperar la librería de juegos de Steam, se ha recurrido a usar su API pública [ROA.3]. La API de Steam pone a disposición de los desarrolladores 4 API38[ROA.4] con distinta funcionalidad:

- ISteamNews: Para recuperar información de noticias de juegos.
- ISteamUserStats: Para recuperar estadísticas globales de usuarios y juegos.
- ISteamUser: Para recuperar información del perfil de los usuarios de Steam.
- ITFItems_440: Para recuperar información acerca del inventario items de un usuario para el juego Team Fortress 2.

Nosotros vamos a usar la API de “ISteamUser” para recuperar la información de la librería de videojuegos del usuario, así como sus estadísticas en tiempo de juego y la caratula de sus juegos. Se ha de obtener previamente un una API Key de Steam, rellenando un formulario. Tras lo cual, Steam nos otorgara una clave para poder explotar su API ([Figura 60]).

---

38 Información de las API disponible en ([https://steamcommunity.com/dev](https://steamcommunity.com/dev))
Con esta clave ya podemos acceder y hacer peticiones a la API. El siguiente paso es encontrar la llamada a la API en la guía [ROA.3] y probar que funciona correctamente. La llamada utiliza el siguiente endpoint:

```
http://api.steampowered.com/IPlayerService/GetOwnedGames/v0001/?
```

Con los siguientes parámetros:

- **key** → La clave de API que nos ha proporcionado Steam
- **steamid** → El ID de Steam asociado a esa cuenta en formato SteamID64
- **format** → Indicamos en qué formato queremos los resultados (JSON/XML/VDF)
- **include_appinfo** → Si lo activamos con 1, quiere decir, que queremos que nos de más información del videojuego.

Por lo que la llamada resultante final es:

```
http://api.steampowered.com/IPlayerService/GetOwnedGames/v0001/?key=XXX&steamid=YYY&format=json&include_appinfo=1
```

El siguiente paso fue comprobar en Postman que la llamada funciona y devuelve los datos correctamente (Figura 61). Como se puede comprobar, la llamada devuelve un JSON que cuenta con 2 campos: **game_count**, que indica el número de juegos totales de la librería, y **games** que es un JSON Array que contiene varios campos, entre ellos el nombre del juego (name), el hash para la imagen del juego (img_logo_url), y el tiempo de juego que le ha dedicado el usuario (playtime_forever). La imagen del juego de Steam tiene como URL base:

```
http://media.steampowered.com/steamcommunity/public/images/apps/{hash}
```

---
39 Para encontrar el ID de Steam asociado a una cuenta se puede usar SteamID disponible en (https://steamid.io/)
El último paso es comprobar que se ha implementado correctamente comprobando que conseguimos el mismo resultado. Para ello se puede observar la salida en el Logcat\textsuperscript{40} (Figura 62). El numero total de videojuegos coincide, así como el JSON devuelto.

\textbf{Figura 61 – JSON devuelto [Fuente: Captura propia] *Por seguridad se ha tapado la API KEY*}

Gracias al uso de esta API, no hace falta que el usuario tenga que usar su usuario y contraseña de Steam, haciendo que el proceso de recuperación de su librería sea muy sencillo y seguro.

\textbf{Figura 62 – Logcat en Android Studio [Fuente: Captura propia] *Por seguridad se ha tapado la API KEY*}

\textsuperscript{40}Logcat, es parte del sistema de monitorización de Android y forma parte del ADB (Android Debug Bridge). En el logcat se pueden ver los logs y los mensajes enviados por el terminal.
API de IsThereAnyDeal:

Para recuperar las ofertas digitales, vamos a usar ITAD\textsuperscript{41} como se explicó en el apartado 4.5. ITAD cuenta con una API propia [ROA.6] que nos va a servir para nuestros propósitos de encontrar ofertas de videojuegos digitales.

Esta API proporciona bastante funcionalidad, pero en esta iteración del programa vamos a usar únicamente un endpoint para obtener las ofertas digitales. Este endpoint es el siguiente

https://api.isthereanydeal.com/v01/deals/list/?

Y cuenta con los siguientes parámetros opcionales según la documentación\textsuperscript{42}:

- **key** → Nuestra clave de API, proporcionada por ITAD.
- **offset** → Offset por el que empezar la lista.
- **limit** → Cuantas ofertas nos devuelve
- **region** → La región para la que vamos a buscar juegos.
- **country** → El país en el que vamos a encontrar ofertas
- **shops** → La tienda digital en la que queremos encontrar ofertas

En nuestro caso, en esta iteración nos vamos a centrar en encontrar ofertas para la región EU2, por lo que usaremos la siguiente URL:

https://api.isthereanydeal.com/v01/deals/list/?key=XXX&region=eu2

Comprobando con Postman (Figura 63) podemos comprobar que recuperamos correctamente las ofertas digitales en formato JSON

\textsuperscript{41} ITAD, abreviación de IsThereAnyDeal (https://isthereanydeal.com/)

\textsuperscript{42} Se puede encontrar más información en el siguiente link (https://itad.docs.apiary.io/#reference/deals/recent-deals/get-deals) desplegando la pestaña de “Get Deals”.

Página 54
Nos está devolviendo un JSON que contiene un campo “data” con un JSON array llamado “list” que contiene la información de las ofertas encontradas. Para cada oferta de videojuego encontramos mucho detalle, y en nuestro caso nos quedamos con el nombre, precio antiguo, precio nuevo, porcentaje de descuento, nombre de la tienda digital en la que está disponible y la URL para comprar.

El último paso es comprobar que se ha implementado correctamente en nuestra aplicación mirando nuevamente la salida del Logcat (Figura 64)
RSS Feed:

Para nuestra aplicación necesitamos un lector de feeds en formato RSS. Para el desarrollo de esta funcionalidad, se creo otro proyecto aparte para atajar este problema y una vez resuelto se incorpore en el proyecto principal. Este RSS Feed cuenta con 3 clases como se puede ver en la Figura 65.

![Figura 65 – Clases RSS Parser.](Fuente: Captura propia)

- **Article**: Es el modelo de datos para un artículo RSS. Contiene los siguientes campos:

  ```java
  public class Article {
      private String title;
      private String author;
      private String link;
      private Date pubDate;
      private String description;
      private String content;
      private String image;
      private List<String> categories;
  }
  ```

- **Parser**: Es la clase principal. Se encarga de manera asíncrona (similar al Scraper) de conectarse al servidor que contiene el RSS y descargárselo. Si se ha conseguido realizar la conexión, llamara a XMLParser que se encarga de parsear el XML y adaptarlo al modelo de datos de Article.

- **XMLParser**: Transforma el RSS descargado en formato XML a un objeto de datos de tipo Article. Para ello utiliza la API XMLPullParser\(^{43}\) [ROA.7], [ROA.8].

\(^{43}\) XMLPullParser, es una interfaz que ofrece funcionalidad de parser para documentos XML. Para mas información consultar en la bibliografía [ROA.7], y [ROA.8].
Google Maps:

Para la funcionalidad de encontrar tiendas físicas se eligió la solución de Google, ya que es una de las opciones más completas que existen hoy día. El primer paso era conseguir acceso a las API de Google. Una vez registrado, encontré las 2 APIs que necesitaba (Figura 66). Estas son “Maps SDK for Android” y “Places SDK for Android”. La primera te permite la utilización de Google Maps en tu aplicación Android. La segunda permite realizar búsquedas de lugares (en nuestro caso tiendas de videojuegos) y devolverte la información.

Con estas APIs ya podía empezar a trastear con Google Maps. Para ello utilicé tanto la documentación oficial de Google [ROA.9], como diferentes páginas web para poder hacer pequeñas pruebas [ROA.10], [ROA.11].
No tuve problemas para poder usar correctamente Google Maps y mostrar el mapa de Google, así como navegar por él. Con esto el siguiente paso consistía en familiarizarme con el “Places SDK for Android” ya que mediante esta API iba a conseguir las informaciones de las tiendas de videojuegos cercanas a mi posición. Para ello empecé a leer tanto la documentación oficial de Google [ROA.12] como algún tutorial que encontré [ROA.13].

Durante estas primeras investigaciones, me tope con un problema. Google había limitado a 1 búsqueda al día mi API del SDK de Places salvo que habilitara la facturación en el proyecto, metiendo un método de pago (Figura 67).

Con esta limitación me resulta imposible el poder utilizar la API de Places SDK para encontrar las tiendas como tenía planeado originalmente. Por lo que tuve que cambiar el diseño y basarme en una implementación de Google Maps dentro de mi aplicación.

El último paso era averiguar la posición GPS del móvil. Para ello utilicé el LocationRequest44 y conseguí recuperar la posición GPS del dispositivo. Cuando realicé pruebas con el emulador aproveché las herramientas de las que dispone (Figura 68), para meter unas coordenadas de

---

44 LocationRequest, es objeto interno de Android que permite recuperar con la mejor calidad las posiciones del dispositivo mediante el uso del FusedLocationProviderApi. Para más información consultar en la bibliografía [ROA.14]
forma manual. En los casos de prueba he estado usando las coordenadas de la FI UPM (40.4043, -3.83955)

![Figura 68 – Limitaciones para el Places SDK. [Fuente: Captura propia]](image)

Con esto consigo que, pulsando el botón de abajo a la izquierda, me centre en la posición GPS en y me encuentre las tiendas cercanas (Figura 69).

![Figura 69 – Derecha: Vista centrada en las coordenadas GPS; Izquierda: Tiendas de videojuegos encontradas cercas. [Fuente: Captura propia]](image)
6 Líneas futuras

Esta primera versión de la aplicación se puede tratar como un prototipo. En ella se ha conseguido implementar toda la funcionalidad requerida salvo el poder agregar videojuegos manualmente utilizando alguna técnica rápida como NFC/OCR.

Aunque este prototipo cumpla con toda la funcionalidad requerida, tiene muchos puntos de mejora para futuras mejoras.

Durante la fase de pruebas, mis amigos y la gente que la probó me dieron varias observaciones en cuanto a la usabilidad. Así mismo también hay muchas pequeñas mejoras que se pueden aplicar a la aplicación para un desarrollo en el futuro:

- **Filtros en librería**: En un primer momento el ver la librería de videojuegos iba a ser la librería tal cual. Gracias a los comentarios de la gente que probó me di cuenta de que tenía que haber algún filtro o forma de filtrar la librería. Por ello añadí otras 2 pestañas que permite ver la librería de videojuegos filtrada mostrando solamente los juegos que están completados o pendiente.
  A pesar de estos filtros, la gente seguía mostrando cierto descontento en el no poder buscar un videojuego en concreto o poder filtrar la librería por plataforma. Esto quedaría como una línea futura el rediseñar el sistema de filtros permitiendo mas versatilidad al usuario.

- **Introducción de horas jugadas**: Otro punto de mejora sería encontrar un método más rápido para introducir las horas jugadas en vez de tener que editar el juego en la librería e introducir las horas con dígitos. Para ello se ha pensado que se podría introducir las horas con algún tipo de picker de manera que fuera más fácil.

- **Gestos**: También se ha pensado en una forma más dinámica y actual para poder editar y borrar un juego en la vista de la librería. Para ello se podría hacer el uso de los swipes hacia los lados para mostrar las opciones de editar y borrar, y así poder aprovechar a rediseñar la vista de la lista de videojuegos.


46 Swipe, Cuando el usuario hace un movimiento deslizante en la pantalla.

47 Por ejemplo, utilizando algo parecido a: [https://github.com/daimajia/AndroidSwipeLayout](https://github.com/daimajia/AndroidSwipeLayout)
- **Rediseño de la búsqueda de tiendas:** Si se consiguiera financiación de algún modo, se podría rediseñar la búsqueda de tiendas, utilizando el Places SDK como se había pensado en un primer momento.

- **Varias interfaces:** Se podrían diseñar varios tipos de interfaces con diferentes combinaciones de colores para que el usuario pudiera elegir entre ellas.

- **Traducciones:** Se podría traducir la APP a diferentes idiomas. Se ha desarrollado la aplicación teniendo en cuenta esta posible línea futura, y todos los *Strings*\(^ {48}\) están metidos en el fichero *strings.xml* por lo que sería muy fácil traducir la aplicación a diferentes idiomas.

- **RSS Noticias:** Ofrecer al usuario la elección de escoger los feeds que quiere ver y ofrecerle diferentes alternativas teniendo en cuenta el idioma de la APP.

- **Importar librerías de otras fuentes:** Algo muy útil para una línea futura sería la posibilidad de poder importar la biblioteca de juegos de otras plataformas digitales como GOG\(^ {49}\), Ubisoft\(^ {50}\), Origin\(^ {51}\), etc. Por ahora no hay ninguna forma publica de hacer esto, pero quizás se podría llegar a algún acuerdo con estas compañías.

- **Utilización de un servidor:** Durante el desarrollo de la aplicación me he dado cuenta de que sería muy conveniente que la aplicación contara con un servidor centralizado. Este servidor contendría la librería de videojuegos del usuario, y se encargaría de encontrar las diferentes estadísticas que necesitamos de manera automática según se añaden juegos que nunca se han visto al servidor. En el servidor podríamos tener algún tipo de cache con la información de los juegos que ya conocemos evitando que se tardara tanto en encontrar cierta información. Además, esto permitiría expandir la funcionalidad de la aplicación hacia nuevas áreas.

- **Aplicación en PC / IOS / Web:** Se podría desarrollar una aplicación que ofrezca la misma funcionalidad para el PC o IOS. Esta aplicación podría recoger en tiempo real las horas que dedica el usuario a jugar a un videojuego y actualizar estos datos de manera automática y transparente para el usuario en vez de que este lo tenga que hacer manualmente.

\(^ {48}\) *Strings*, las cadenas de texto que se muestran al usuario.

\(^ {49}\) GOG (Good Old Games), tienda virtual de videojuegos sin DRM (https://www.gog.com/)

\(^ {50}\) Ubisoft, plataforma y tienda virtual de la compañía Ubisoft (https://www.ubisoft.com/es-es/)

Alternativamente, también se le podría dar acceso al usuario a una versión similar a la del móvil con una pagina web para cuando no tenga el móvil cerca.

- *Componente social*: Otro punto a desarrollar en alguna futura línea sería algún tipo de componente social (Chat / Foro), donde los usuarios pudieran intercambiar opiniones de los videojuegos, así como hacer recomendaciones. Esto le daría mas aliciente a la gente para usar la aplicación.

- *Usar datos como Metacritic*: Otro gran punto de mejora sería la incorporación de datos de valoraciones externos como Metacritic\(^{52}\). Esto permitirá que el usuario a la hora de buscar juegos a los que jugar ordenarlos en función de su calificación.

- *Añadir datos extra para los videojuegos*: Se podría añadir una pestaña con mas información para sus videojuegos, donde pudieran encontrar de manera rápida links a guías, videos de YouTube, información adicional como el año en el que salió, etc.

- *Hablar con HowLongToBeat*: Un gran punto a desarrollar en el futuro seria hablar con HowLongToBeat y llegar a un acuerdo por el cual tener acceso a una API privada para explotar, sin tener que recurrir al Scrapping. A cambio ellos recibirían estadísticas de uso de videojuegos cuando un usuario marcara un videojuego como completado añadiendo más estadísticas a su pagina web.

Como se puede ver, todavía se pueden extender mucho mas las funcionalidades de la aplicación, así como mejorar su calidad. Para emprender esta tarea en un futuro habría que estudiar la viabilidad económica de llevar a cabo este esfuerzo e inversión.

---

\(^{52}\) **Metacritic** ([https://www.metacritic.com/](https://www.metacritic.com/)) es una página web que se dedica a recopilar valoraciones de películas, series, videojuegos, etc... Recopila tanto las opiniones de los críticos como la de los usuarios.
6.1 Viabilidad económica

Para estudiar la viabilidad económica de la aplicación vamos a examinar por un lado los gastos que requiere la aplicación y por otro las posibles vías de monetización.

- **Gastos:**
  Tal cual esta la aplicación el único gasto sería el pago único que impone Google de 25$\(^{53}\) (unos 22€ al cambio). Este desembolso es bastante asumible, ya que es un único pago. Si a esto le sumamos una pagina web simple para mostrar el producto, podemos estar hablando desde unos 30€ a unos 50€ por año de un hosting barato con nombre de dominio incluido.

  Si añadimos un servidor de datos como Firebase, esta cobra en base a su utilización por lo que al principio nos saldría gratis y luego tendríamos que actualizar a la versión de 25$ al mes (otros 22€ al cambio) o si hubiera mucho tráfico una tarifa personalizada.

- **Monetización:**
  Monetizar esta aplicación sería relativamente fácil. El primer paso sería ofrecer la aplicación en 2 modalidades. Una gratis, que tuviera la inclusión de anuncios y otra con un pago único que quitara los anuncios. Este método de monetización es muy común en la PlayStore, y permitirá ganar dinero de la gente que usa la aplicación de manera gratuita con la inclusión de los anuncios, y de la gente que decidiera pagar por quitar los anuncios que sería una minoría en comparación.

  Otro punto de posible monetización es llegando a un acuerdo con IsThereAnyDeal.com. Ellos ganan dinero cuando un usuario compra un videojuego desde su página web. Si un usuario llegase a comprar un videojuego utilizando un enlace desde mi aplicación, ellos me pagarían un % de esa venta que ellos han ganado.

A falta de un estudio mas en profundidad, tal cual está la aplicación podría ser viable su explotación económica en la PlayStore en 2 versiones, de pago y gratuita con anuncios. El coste de inversión seria 22€ de la licencia de Google, que es un gasto asumible en caso de que no se recupere la inversión.

En el caso de que la aplicación ganara usuarios y por lo tanto una remuneración económica que permitiera ampliar su funcionalidad, sería entonces el momento de realizar una mayor inversión. Ese sería el punto para ampliar la aplicación con las líneas futuras.

\(^{53}\) Información extraída de [https://cincodias.elpais.com/cincodias/2015/02/01/lifestyle/1422792260_243066.html] y [https://play.google.com/apps/publish/signup/]
7  Resultados

A pesar de que el scope del proyecto era muy extenso, he podido llevar a cabo una implementación de todas las funcionalidades salvo una. De todos los requisitos y funcionalidades, he podido llevar a cabo todos, salvo la de encontrar una forma más rápida de añadir juegos manualmente usando NFC o OCR.

Esta funcionalidad se ha quedado en el tintero por falta de tiempo. Los problemas con IGDB y la API de Places SDK de Google han tenido un fuerte impacto en el tiempo, ocasionando que dicha funcionalidad no pudiera ser desarrollada. A cambio he aprendido a realizar Scrapping de una página web algo que es bastante útil e interesante de cara al futuro.

Estoy muy contento con la aplicación resultante. Todo el mundo que la ha probado ha quedado contento con ella y me han felicitado. Incluso un par de amigos me han pedido la aplicación para usarla para manejar su propia colección de videojuegos, que hasta ahora estaban usando una plantilla Excel. Incluso me han animado para que la suba a la PlayStore.

Este TFG me ha servido para aprender a programar en Android con soltura y a manejar una aplicación muy potente como es Android Studio. Ha sido una primera toma de contacto con un servicio de gestión de versiones como Bitbucket y a cómo usarlo de manera eficiente.

He abordado el TFG como si de un desarrollo real se tratase, y sin duda ha sido un trabajo bastante complicado, que me ha llevado a veces a cuestionar si no debería haber recortado funcionalidades. Me ha llevado una innumerible cantidad de horas, y de líneas de código terminar este proyecto (Figura 69).

![Figura 69 – Conteo de líneas del código finales utilizando Cloc](http://cloc.sourceforge.net/)

54 *Cloc*, [http://cloc.sourceforge.net/](http://cloc.sourceforge.net/) es una herramienta que permite contar las líneas de código de un proyecto. Te devuelve las estadísticas con el numero total de ficheros, líneas en blanco, comentarios y líneas de código finales, separadas por lenguaje de programación.
Al final este TFG me ha servido como un punto de partida para una futura aplicación que voy a acabar publicando en la PlayStore. Este verano tengo pensado darle un repaso a la aplicación, probarla mas a fondo y cambiar algún pequeño detalle de usabilidad para hacerla mas accesible y dinámica.
8 Conclusiones

Una de las mayores lecciones que he aprendido, en comparación con otras prácticas es la necesidad de analizar y planificar correctamente el problema antes de empezar a desarrollar. En otras prácticas de la carrera, siempre empezaba con el desarrollo e iba improvisando sobre la marcha. Sin esta planificación y estudio previo exhaustivo, no creo que hubiera sido capaz de abarcar con todo.

Otra lección muy importante ha sido la importancia de la adaptabilidad ante los problemas. He tenido varios escollos importantes, como tener que pasar de usar IGDB a usar HowLongToBeat. Esto me supuso un importante desafío y pérdida de tiempo puesto que todo el trabajo que tenía avanzado e investigado tuvo que ser descartado. Tuve que aprender a realizar Scrapping, y luego tuve muchos problemas en conseguir realizarlo correctamente ya que la página web de HowLongToBeat no sigue un patrón uniforme en su HTML, cambiando con diferentes juegos. Tuve que dedicarle muchas horas de investigación hasta que encontré la forma de extraer de forma fiable la información que necesitaba. Si me hubiera dejado llevar por el pánico y no hubiera sabido adaptarme no hubiera podido, a lo mejor, llegar a completar esta funcionalidad para el proyecto. Esto también se puede ver en el caso de la limitación del API de Places SDK que me llevo a tener que rediseñar por completo la forma de encontrar tiendas físicas. Gracias a que supe buscar una alternativa y adaptarme pude completar la funcionalidad dentro de un margen de tiempo razonable.

También me he dado cuenta de que mi plan de no usar ningún servidor es a la vez una ventaja y una desventaja importante. El almacenar todo en el teléfono móvil permite un desarrollo más rápido, pero a su vez hace que recaiga mucho procesamiento adicional en el teléfono, que, si bien en las pruebas los teléfonos móviles son capaces de llevarlo a cabo, un servidor sería la opción más lógica. Para solventar este hándicap, tuve que crear un sistema de cache con los tiempos de HLTB, ya que estaba tardando de media 1-2 segundos por videojuego en encontrar los datos que necesitaba. Con una librería de casi 800 juegos estamos hablando de casi 26 minutos para recorrer toda la librería de videojuegos. Cada vez que borraba la aplicación para volver a hacer pruebas tenía que esperar otros 26 minutos hasta que volvía a recuperar los datos. Habiendo utilizado un servidor, podría tener una base de datos donde ir guardando la información de los videojuegos de tal forma que las siguientes veces fuese instantáneo. Además, el servidor podría ir buscando el por su cuenta la información de nuevos juegos en vez de tener que ser el usuario manualmente cuando le diera al botón y tener que dejar la aplicación en background durante tanto tiempo.
Las pruebas con mis amigos me han desvelado la importancia de realizar un estudio de usabilidad previo. Debido al poco tiempo disponible, no pude dedicar el tiempo necesario para hacer un estudio de usabilidad a fondo. Esto provocó que surgieran ciertos problemas de usabilidad dentro del proyecto. Estos se podrían haber solventado antes, o incluso haber encontrado una solución más adecuada que ahora quedaría relegada a líneas futuras.

En líneas generales, la realización de este TFG me ha ayudado a aprender a programar aplicaciones en Android. Gracias a él, he aprendido como funciona Android, Google Maps, a realizar un Scraping de una página web y cómo realizar conexiones y peticiones a APIs externas. Se puede decir que en el estado actual de la aplicación me sirve como un punto de partida para una futura aplicación real.

Todo el conocimiento práctico y teórico adquiridos durante la realización del TFG me van a venir muy bien de cara a mi futuro laboral. El proceso de elaboración ha sido similar al que se utilizaría en un proyecto en el entorno laboral, con la salvedad de que lo he tenido que realizar yo solo.
9 Bibliografía

Se ha usado los siguientes recursos bibliográficos durante el estudio de Android:

**Libros de referencia:**

**Recursos online:**

Para la realización y desarrollo de la aplicación de la aplicación se han usado los siguientes recursos:

**Recursos online:**
<table>
<thead>
<tr>
<th><strong>Firmante</strong></th>
<th>CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fecha/Hora</strong></td>
<td>Sun Jan 13 19:35:42 CET 2019</td>
</tr>
<tr>
<td><strong>Emisor del Certificado</strong></td>
<td>EMAILADDRESS=<a href="mailto:camanager@fi.upm.es">camanager@fi.upm.es</a>, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES</td>
</tr>
<tr>
<td><strong>Numero de Serie</strong></td>
<td>630</td>
</tr>
<tr>
<td><strong>Metodo</strong></td>
<td>urn:adobe:com:Adobe.PPKLite:adbe.pkcs7.shal (Adobe Signature)</td>
</tr>
</tbody>
</table>