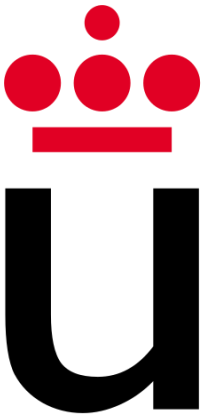

Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN DISEÑO Y DESARROLLO DE VIDEOJUEGOS

Curso Académico 2023/2024

Trabajo de Fin de Grado

GAMESHARE: Portal web para compartir experiencias de videojuegos basado en *Angular* y *Firebase*

Autor: Daniel Ayllón Peinado

Tutora: Lucía Serrano Luján

Esta obra está bajo licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/4.0/.

Agradecimientos

A mi tutora Lucía por ayudarme incluso cuando desaparecía durante meses y a mis padres y amigos por insistirme tanto en terminarlo de una vez.

RESUMEN

Los videojuegos se han convertido en una de las fuentes de entretenimiento más importantes de la historia de la humanidad, llegando a competir de tú a tú con medios bien establecidos como películas o series de televisión en cuanto a seguidores, calidad e ingresos. Por otro lado, también hemos sido testigos de la evolución de las redes sociales, donde encontramos una gran variedad, como *Facebook* o *Twitter,* que nos mantienen en contacto tanto con nuestro mejor amigo como con una persona que vive en la otra parte del planeta.

Gracias a este ascenso de las redes sociales y de los videojuegos, mucha gente puede encontrar otras personas a las que le apasione un videojuego que nadie de su círculo cercano conocía, pudiendo compartir experiencias, secretos, guías, etc. que han tenido jugando a ese título en concreto.

Por desgracia, las redes sociales no solo nos han aportado cosas buenas, también tienen sus lados negativos. Un videojuego puede ser tendencia en cualquier red social durante un tiempo determinado hasta que salga el siguiente producto popular, cayendo este en el ostracismo.

Uniendo las ideas de compartición de experiencias y el intento de no olvidar un videojuego cuando deja de ser popular nace el software que presento en este proyecto, *GameShare*.

GameShare es una plataforma web creada mediante el uso del *framework Angular* y la plataforma de desarrollo de *Google*, *Firebase* cuyo principal atractivo es la creación de reseñas de videojuegos entre aficionados para recomendar (o no) un videojuego. No solo eso, sino que además también permite la creación de listas personalizadas para ayudar a clasificar el estado de los juegos, tanto los que estamos jugando como los que hemos terminado pasando por los que hemos abandonado.

Palabras clave:

Plataforma web, experiencias, reseñas, listas, *Firebase*, *Angular*.

ABSTRACT

As we all know, video games have become one of the most essential sources of entertainment in human history, competing head-to-head with well-established media such as movies or TV series in terms of followers, quality and revenue.

On the other hand, we have also witnessed the evolution of a new type of socializing through various applications like Facebook or Twitter that keep us in touch with our best friend and a person living on the other side of the planet.

Thanks to the rise of social media and video games, a vibrant community has emerged. People can now connect with others who share their passion for a game, even if no one in their immediate circle is aware of it. This allows for the sharing of experiences, secrets, and guides, creating a sense of camaraderie among players.

Unfortunately, social media has not only brought us good things; it also has its negative sides. A video game can be a tren on any social network for a certain period of time until the next popular product comes out, falling into obscurity.

GameShare was born by combining the ideas of sharing experiences and the attempt not to forget a video game when it ceases to be popular.

GameShare, a web platform built using the Angular framework and Google's development platform, Firebase, puts the power in the hands of gamers. Its main feature is the creation of game reviews by fans, allowing them to recommend or advise against a game. Additionally, it offers the convenience of creating custom lists to help manage the status of games, from those currently being played to those that have been completed or abandoned.

Keywords

Web platform, experiences, reviews, lists, *Firebase*, *Angular*.

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. Introducción

La industria del videojuego está en alza y no podemos vislumbrar cuando alcanzará su máximo exponente. Este crecimiento viene en gran parte por la pandemia de COVID-19 y el cambio al entretenimiento en casa al que tuvimos que acostumbrarnos. Según la Asociación Española del Videojuego (AEVI), en 2019 la industria del videojuego facturó 1.479 millones de euros y el número de usuarios superó los 15 millones (Asociación Española de Videojuegos (AEVI), 2020). Si miramos esta misma información dos tres años después, en 2022, podremos ver como hubo un crecimiento del 36,05%, superando los 2.012 millones de euros y los 18,5 millones de video-jugadores en España (Asociación Española de Videojuegos (AEVI), 2023).

Es por eso por lo que, en un mundo en constante crecimiento y con la cantidad de lanzamientos que pueden llegar a salir en un solo día, puede llegar a abrumar a ciertos usuarios que a lo mejor quieran jugar a algún éxito reciente, pero es tapado repentinamente por el siguiente juego de éxito.

GameShare cubre una necesidad cada vez más incipiente de usuarios que olvidan ciertos videojuegos que han salido al mercado por los nuevos, quedando abandonados en su cada vez más larga lista de juegos pendientes. Por ello, este trabajo propone una aplicación de creación de listas, como por ejemplo 'Jugando', 'Terminados' o 'Abandonados', en la que el usuario pueda registrar todos los videojuegos que está jugando y así no olvidarse de ellos cuando empiece una nueva aventura en el último *The Legend of Zelda*.

Pero esta aplicación no solo contiene listas de videojuegos. Los usuarios también pueden compartir reseñas de los videojuegos que hayan terminado con la nota personal de cada uno. Además, cada semana se muestran los videojuegos más populares, por lo que, si algún usuario no sabe a qué jugar a continuación, siempre puede echar un vistazo a esa sección, que seguro que puede aportar ideas.

Esta aplicación se ha realizado utilizando *Angular* (Angular Team, 2024), un '*framework*' para la creación de aplicaciones web, y *Firebase* (Google LLC, 2024), una plataforma de desarrollo de *Google*, siguiendo una metodología *SCRUM* utilizando *Trello* (Atlassian, 2024) para el seguimiento de las tareas.

2. Estudio de aplicaciones similares

Para poder tener en cuenta qué funcionalidades debía tener la aplicación de este documento se investigaron varios tipos de aplicaciones que funcionaran de forma similar a *GameShare*. Una vez finalizada esta búsqueda se redujeron a tres candidatos, cada uno con distintas funcionalidades que serían interesantes implementar en *GameShare*.

2.1 *TV Time*

TV Time (TV TIME, 2024) es una aplicación de *Android*, *iOS* y escritorio cuyo principal objetivo es poder registrar qué series de televisión y películas está viendo cada usuario. El mayor atractivo de este software es su aplicación para dispositivos móviles. Nada más entrar el usuario se encuentra con todas las series que está viendo actualmente, capítulo por capítulo, además de una pequeña sección indicando qué día se estrenan los siguientes capítulos.

Si el usuario pulsa en los botones situados en la parte inferior de la aplicación, se desplazará a distintas secciones de la aplicación: una similar a la mencionada anteriormente, pero en este caso con películas, tanto las que el usuario quiera ver en el futuro y las que se estrenarán próximamente.

A continuación, podremos ver una sección en la que se encuentra una barra de búsqueda que permite al usuario encontrar series y películas. Debajo de esta se puede observar una sección dedicada a las series y películas más populares del momento.

Por último, existe una sección que muestra el perfil propio de cada usuario, indicando las últimas series/películas y sus favoritas, además de una pequeña sección de estadísticas como, por ejemplo, el número total de películas vistas.

2.2 *Backloggd*

Backloggd (BACKLOGGD, 2024), a diferencia de la aplicación previamente mencionada, es una plataforma web diseñada para permitir a los usuarios llevar un registro meticuloso de los videojuegos que están jugando actualmente. Esta plataforma ofrece a los jugadores la posibilidad de crear listas personalizadas, lo que les permite no solo llevar un registro de los juegos que están jugando en el presente, sino también de aquellos que han completado y de los que tienen la intención de jugar en el futuro.

Una característica especial de esta página web es que permite a los usuarios calificar los videojuegos y dejar sus propias reseñas. Esta funcionalidad es fundamental en cualquier aplicación de esta naturaleza, ya que proporciona a los consumidores la información necesaria para tomar decisiones informadas sobre su próximo videojuego.

Al igual que *TV Time, Backloggd* cuenta con una sección de estadísticas que ofrece información detallada sobre diversos aspectos de cada usuario, como el tiempo total jugado o los géneros más jugados. Esta funcionalidad permite a los usuarios obtener una visión general de sus hábitos de juego y seguir su progreso a lo largo del tiempo.

Por último, pero no menos importante, *Backloggd* permite la integración de otras bibliotecas de juegos, como *Steam*. Esta funcionalidad hace que la experiencia del usuario al transferir todos sus datos a esta plataforma sea mucho más cómoda y eficiente que si tuviera que hacerlo de forma manual.

En resumen, *Backloggd* es una herramienta integral que ofrece a los jugadores una plataforma eficiente y fácil de usar para llevar un registro de sus videojuegos, proporcionando al mismo tiempo una serie de funcionalidades adicionales que mejoran la experiencia de usuario y facilitan la toma de decisiones informadas sobre sus futuros videojuegos.

2.3 *Stash*

Stash (STASH, 2024), al igual que *Backloggd*, es otra aplicación de seguimiento de videojuegos disponible en web y aplicación móvil que permite a los jugadores gestionar los videojuegos que actualmente tienen o desean tener en un futuro. Esta permite realizar listas con los juegos que cada usuario ha jugado, está jugando o ha abandonado, un aspecto muy interesante de esta aplicación.

Al igual que *Backloggd*, *Stash* también permite importar los videojuegos que cada usuario tiene en su biblioteca de *Steam* para evitar que se haga de forma manual, algo que puede resultar tedioso en muchas ocasiones.

Otra característica de *Stash* es su capacidad de crear alertas de lanzamientos de videojuegos. Al igual que *TV Time*, la aplicación lleva una cuenta atrás de los videojuegos que todavía no hayan salido al mercado, pero el usuario ya lo tenga incluido en su lista de futuras aventuras.

Por último, también permite la creación de reseñas de juegos para que todos los usuarios puedan ver qué opina el público general de cada videojuego. Luego con esta información cada consumidor puede valorar si merece la pena ese videojuego.

Se pueden comparar las características de las aplicaciones en la siguiente tabla:

Tabla 1. Cuadro resumen con las principales características de las plataformas de seguimiento de contenido. (Cont.: contenido, VJ: Videojuego, S.C.: Seguimiento Contenido, Extras: Funciones extras, Interac. Comunidad: Interacción con la comunidad, Reseñas: publicación de reseñas).

Nombre Ap.	Cont.	S.C.	Descub. nuevo contenido	Interac. comunidad	Extras	Reseñas
TV Time	TV y películas	Sí	Sí	Sí	Listas, estadísticas	Sí
Backloggd	VJ	Sí	Sí	Sí	Tiempo de juego, listas	Sí
Stash	VJ	Sí	Sí	Sí	Nuevos lanzamientos.	Sí
GameShare	VJ	Sí	Sí	Sí	Estadísticas, tendencias	Sí

Como se puede observar con la tabla comparativa, las tres aplicaciones estudiadas recogen una serie de características comunes cuya implementación en *GameShare* es fundamental, las cuales se describen a continuación en el siguiente apartado.

2.4 Características esenciales de la aplicación

Una vez finalizada la investigación de las distintas alternativas que existen se tomó la decisión de incorporar estas características en la primera versión del producto final:

- **Reseñas de usuarios.** Tal y como se ha indicado en las secciones anteriores, las reseñas de los usuarios son una característica esencial en esta clase de aplicaciones ya que permiten a los usuarios tomar decisiones informadas del siguiente videojuego a empezar.
- **Listas personalizadas:** La aplicación debe dar la posibilidad al usuario de crear listas personalizadas para que pueda organizar correctamente los videojuegos que está jugando, que ha terminado, etc., y permita clasificar cada videojuego según su género, fecha de salida, desarrollador…
- **Popularidad.** La aplicación recoge en una lista en la pantalla de inicio los juegos más jugados y los más abandonados de la semana.

3. Objetivos

El objetivo principal de este proyecto consiste en el desarrollo de la aplicación Web *Gameshare*, que cree un entorno compartido de experiencia en el mundo de los videojuegos. A continuación, se definen los subobjetivos a alcanzar:

1. **Dominio de *Firebase* y consolidación de habilidades en *Angular*.** *Firebase* se destaca como una herramienta robusta que puede simplificar el desarrollo de numerosos aspectos de la aplicación, que de otra manera podrían considerarse laboriosos. Un ejemplo notable es la implementación de un sistema de seguridad para el inicio de sesión. La familiarización con *Firebase* se convierte en un componente esencial de este desarrollo, dado que la aplicación en su totalidad opera a través de esta herramienta. En paralelo, este proyecto también se centra en la consolidación de habilidades en *Angular*. A pesar de contar con conocimientos previos en *Angular*, el objetivo es aplicar estos conocimientos de manera efectiva para lograr un desarrollo ágil y limpio, al tiempo que se adquieren nuevos aprendizajes con las versiones más recientes de este *framework*.
2. **Finalización de un ciclo de desarrollo completo.** Rara vez se presenta la oportunidad de llevar a cabo un desarrollo integral de una aplicación, que abarca desde la recopilación de requisitos hasta las etapas posteriores a la finalización del desarrollo. Este proceso proporcionará un aprendizaje invaluable para futuros proyectos, tanto personales como profesionales. En esencia, este proyecto no solo representa la creación de una aplicación, sino también una experiencia de aprendizaje completa y enriquecedora que preparará para futuros desafíos en el campo del desarrollo del software.
3. **Implementación de habilidades profesionales en un proyecto personal.** La experiencia adquirida a través de la participación en proyectos auténticos en un entorno laboral ha proporcionado una base sólida de conocimientos y habilidades. Estos incluyen la gestión de sprint y la conducción de reuniones, que son prácticas comunes en el desarrollo de software. Este proyecto representa una oportunidad única para transferir y aplicar las habilidades profesionales en un contexto personal..
4. **Innovación y desarrollo de funcionalidades.** La creación de la aplicación, en sí misma, representa un desafío considerable. Sin embargo, la incorporación de

características observadas en el estudio de aplicaciones existentes puede resultar enriquecedora. Estas podrían incluir una sección dedicada a las tendencias actuales entre los usuarios, lista de amigos, creación de reseñas personalizadas y un carrusel de imágenes que exhiba las novedades. Adicionalmente, sería prudente implementar pruebas rigurosas para asegurar el correcto funcionamiento.

4. Metodología

En esta sección del documento se tratará la metodología empleada en el desarrollo del proyecto, además de los programas utilizados para el desarrollo de este y diferentes alternativas de entornos web similares.

4.1 Elección y descripción de la metodología

Como se ha mencionado al final del apartado anterior, la aplicación se ha construido utilizando *Angular* y *Firebase*. Se decidió el uso de estas tecnologías por su conocimiento previo al desarrollo del trabajo, además de la facilidad con la que se integran ambas tecnologías y la velocidad de desarrollo en comparación con tecnologías más clásicas como *Spring* y *SQLServer*.

Angular es un '*framework'* de desarrollo de software web que sirve para generar *Single Page Applications* (SPA) tanto en escritorio como en dispositivos móviles. Una SPA es una aplicación web que muestra todas las pantallas de una aplicación web en una misma pantalla, sin necesidad de recargar el navegador, agilizando la navegación y la experiencia de usuario general. Para el desarrollo de estas aplicaciones se usa *TypeScript* (Microsoft Corporation, 2024), un lenguaje de programación mantenido por *Microsoft*. La principal diferencia entre *JavaScript* y *TypeScript* es el tipado estático de variables, lo cual facilita el desarrollo y la detección de errores.

Firebase es una plataforma de desarrollo de aplicaciones móviles y web de *Google* que proporciona varios servicios como autenticación, almacenamiento en la nube y bases de datos que facilitan la elaboración del software a los desarrolladores. *Firebase* maneja las operaciones del servidor, haciendo que el programador solo se tenga que preocupar de la construcción de la parte frontal de su aplicación.

Normalmente *Firebase* está orientado al desarrollo en dispositivos *Android*, pero también permite su uso en aplicaciones de Angular utilizando librerías externas como *AngularFire*, (AngularFire, 2024) la cual contiene una documentación muy extensa explicando todas las partes esenciales que tienen las aplicaciones que usan esta tecnología.

Otro de los principales motivos para elegir esta tecnología es su escalabilidad. Ya que *Firebase* maneja automáticamente todas las operaciones que normalmente se manejarían entre *Spring* y el sistema de BBDD escogido permite aliviar el trabajo de los desarrolladores ya que pueden poner todos sus esfuerzos en la parte frontal de la aplicación y no en tres entornos distintos, que es lo que suele ocurrir en el desarrollo de aplicaciones.

Además, *Firebase* permite el manejo, el registro y el inicio de sesión de los usuarios de la aplicación de forma muy sencilla con su apartado *Auth* (Firebase, 2024). Asimismo, *Firebase* se encarga de la seguridad del usuario mientras que utilizando otras alternativas como *Spring Security* puede llegar a ser mucho más complicado de manejar.

Firebase contiene una base de datos que se actualiza en tiempo real llamada *Firestore* (Firebase, 2024) que permite sincronizar datos entre diferentes usuarios de la aplicación en el instante, lo cual agiliza esta tarea.

Para el desarrollo de esta aplicación se eligió trabajar bajo la metodología *SCRUM*. Esto se debe a su uso anterior en otros proyectos, con lo que ya se estaba familiarizado con el funcionamiento de esta metodología ágil, además de la facilidad de adaptación a los cambios de esta. En la Figura 1 se muestra un ejemplo de esquema *SCRUM*:

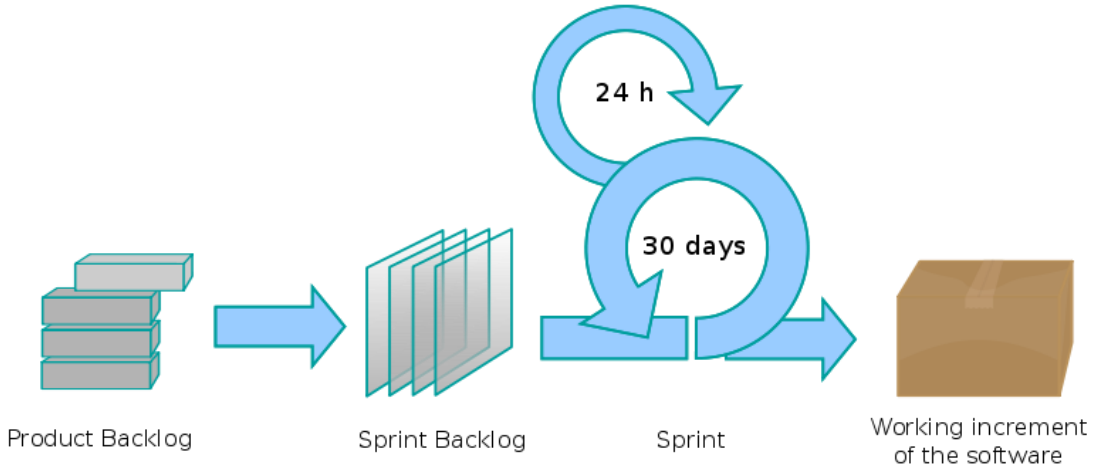

Figura 1. Esquema SCRUM (Fuente: (Orellana, 2017))

Esta imagen explica perfectamente en qué consiste la metodología *SCRUM*:

- ***Product Backlog*:** Todas las tareas del producto, normalmente ordenadas por prioridad.
- ***Sprint Backlog*:** Esto es una lista de tareas elegidas por el equipo de desarrollo para ser realizadas en el siguiente Sprint.
- ***Sprint*:** Un espacio de tiempo en el que se realizan las tareas seleccionadas en el *Sprint Backlog*. Los *Sprints* suelen durar aproximadamente entre una semana y un mes. Además, se hacen reuniones diarias para explicar el progreso de las tareas y si existe algún problema que impida avanzar en estas.
- Por último, al finalizar cada *Sprint* se realiza un seguimiento del incremento de del desarrollo del producto, en forma de demo o documento de evidencias, para demostrar el trabajo realizado. Además, se suele hacer una retrospectiva para analizar los puntos positivos y negativos del *Sprint* y proponer soluciones para mejorar de cara al siguiente.

Para llevar el seguimiento de forma eficiente de esta aplicación se ha utilizado la herramienta *Trello* por su facilidad de uso y su conocimiento previo. *Trello* es un programa muy útil usado para la gestión de proyectos en equipo o individuales. Permite la creación de 'columnas' (que en este caso representan a cada uno de los pasos de la metodología ágil *SCRUM* explicada en el apartado anterior) y 'tarjetas', que representan las tareas a realizar.

Se crearon varias columnas con cada una de las secciones indicadas anteriormente y tarjetas con las distintas tareas. La Figura 2 muestra un ejemplo de un tablero de esta aplicación:

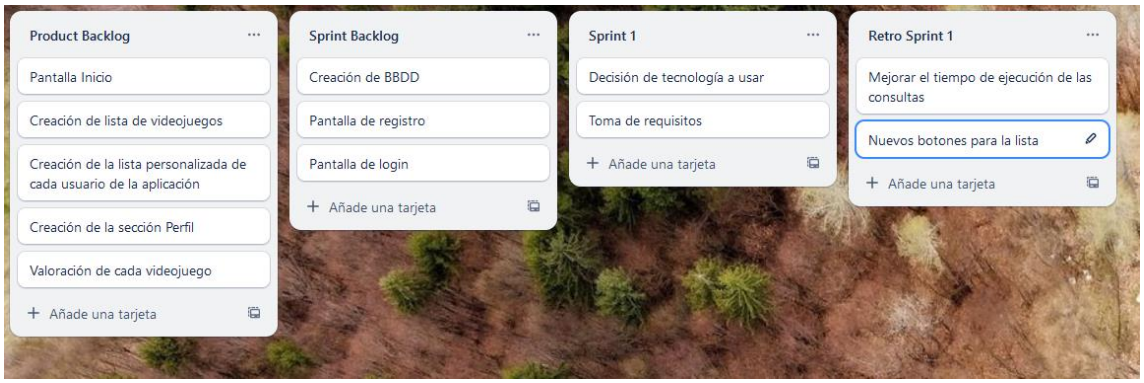

Figura 2. Tablero Trello. (Fuente: Elaboración propia).

Por último, pero no menos importante, se ha utilizado *GitHub* como una herramienta esencial para mantener un control de versiones y garantizar una copia de seguridad del proyecto en caso de pérdida inesperada de datos. *GitHub*, un servicio en la nube de renombre mundial, es una plataforma que permite a los desarrolladores subir y gestionar el código de sus aplicaciones de manera eficiente.

Este servicio es realmente útil para los desarrolladores, ya que proporciona un sistema de control de versiones que permite realizar un seguimiento detallado de todas las modificaciones realizadas en el código. Esto significa que cualquier cambio realizado en el código se registra, permitiendo a los desarrolladores volver a versiones anteriores si algo sale mal.

Además, *GitHub* facilita la colaboración en proyectos, permitiendo a varios desarrolladores trabajar juntos en el mismo código y fusionar sus cambios de manera eficiente. Esto es especialmente útil en proyectos grandes donde varios desarrolladores pueden estar trabajando en diferentes características al mismo tiempo.

Para este proyecto, se ha subido todo el código a mi repositorio personal de *GitHub*. Esto no solo permite mantener un registro de todos los cambios que he realizado, sino que también permite compartir el trabajo con otros. Al final de este documento se encuentra un enlace a dicho *GitHub* (Peinado, 2024).

4.2 Aplicación de la metodología

A la hora de aplicar la metodología *SCRUM* al proyecto se decidió una separación en cuatro *Sprints* de un mes de duración. Cada *Sprint* cubría un aspecto determinado de la aplicación:

1. Febrero 2024 - Investigación, elección de tecnologías y comienzo del desarrollo: el principal objetivo del primer *Sprint* era elegir qué tecnología se iba a utilizar en el proyecto, comparando entre varias opciones y la primera toma de requisitos para poder determinar qué necesitaba esta aplicación. Además, también comenzó el desarrollo con la creación de la pantalla de registro e inicio de sesión de usuarios, aunque todavía sin funcionalidad.
2. Marzo 2024 – Creación de BBDD e integración con *Firebase*: en este Sprint se fusionó el desarrollo del primer Sprint con la tecnología de *Firebase*. Se implementó la funcionalidad de registro de usuarios, una BBDD de una pequeña muestra de videojuegos que se usarían como ejemplo para la demo y nuevas pantallas como la lista de todos los videojuegos.
3. Abril 2024 – Funcionalidad completa de la aplicación: una vez unidas ambas tecnologías se desarrolló el resto de la aplicación (pantalla de inicio, lista de videojuegos, lista propia, etc.).
4. Mayo 2024 – Mejoras en la aplicación e integración con *GitHub Copilot* (GitHub, 2024): por último, en este *Sprint* se aplicaron ciertas mejoras de usabilidad de la aplicación y se experimentó con las capacidades de la IA generativa de *GitHub Copilot*.

A final de cada *Sprint* se realizaba una reunión con la tutora del Trabajo de Fin de Grado en forma de retrospectiva para valorar cambios a introducir y enseñar el progreso conseguido en el mes.

4.3 Retos y soluciones

Durante el desarrollo de este proyecto se han encontrado varios obstáculos significativos. El principal de ellos fue el desconocimiento general del entorno de *Firebase*. Para poder aprender a usar esta tecnología se tuvo que dedicar un mes aproximadamente solamente en leer documentación y casos de prueba para enfocar el desarrollo de la aplicación únicamente en las partes que harían falta una vez comenzada la elaboración del software.

Afortunadamente, *Firebase* cuenta con una amplia documentación que especifica cualquier operación que se quiera realizar, tanto si se trabaja en versión móvil como en escritorio, siendo este el caso a tratar.

Otro desafío en el desarrollo de la aplicación fue la elección de la versión de *Angular*. Puede parecer baladí, pero no lo fue. Esto se debe a que, en *Angular* 17 se introdujeron los '*Standalone Components'*. Esto implicaba que las nuevas aplicaciones hechas a partir de esa versión no tendrían módulos, que son ficheros que almacenan todos los componentes, servicios, directivas, etc. de la aplicación para que puedan ser utilizados por el resto de la aplicación. He aquí una pequeña tabla comparativa de los dos modelos de aplicación:

Tabla 2. Cuadro comparativo entre una aplicación de Angular creada mediante el método Standalone Components y otra mediante el uso de módulos.

Standalone Components	**Módulos**
Son independientes	Los componentes en un módulo dependen de este para su funcionamiento.
Son más flexibles a la hora de reutilizarlos dentro del mismo proyecto.	No son tan fácilmente reutilizables.
Reducen la complejidad ya que cada componente se encarga de importar lo que necesita.	Aumentan la complejidad ya que agrupan componentes y sus dependencias.

Una vez expuestas las diferencias entre ambos modelos puede parecer obvio que el camino a seguir es mediante *Standalone Components*, pero debido a la familiaridad anterior con módulos se eligió finalmente realizar la aplicación mediante el uso de estos. Aunque las versiones 17 y adelante de *Angular* crean las aplicaciones por defecto sin módulos, se puede elegir utilizar una aplicación con módulos añadiendo cierta configuración a la hora de la creación de esta, con lo que el problema fue resuelto fácilmente.

5. Proceso de desarrollo del software

5.1 Toma de requisitos

Una vez terminada la investigación de las alternativas existentes en aplicaciones de seguimiento, se crearon una serie de requisitos funcionales y no funcionales que debía tener la aplicación final.

5.1.1 Requisitos funcionales

- **RF1 – Registro e inicio de sesión:** los usuarios deben poder registrarse proporcionando un nombre de usuario, una dirección de correo electrónico válida y una contraseña segura. Una vez registrados, deben poder iniciar sesión utilizando sus credenciales.
- **RF2 – Navegación en la aplicación:** la aplicación debe proporcionar una interfaz de usuario intuitiva con un sistema de navegación claro que permita a los usuarios moverse fácilmente entre las diferentes secciones de la aplicación.
- **RF3 – Pantalla de inicio:** en la pantalla de inicio, se deben mostrar dos listas: una con los videojuegos más jugados de la semana y otra con los videojuegos más abandonados de la semana. Estas listas deben actualizarse automáticamente cada semana.
- **RF4 – Visualización de la lista de videojuegos:** la aplicación debe proporcionar una lista completa de todos los videojuegos disponibles en su base de datos.
- **RF5** – **Filtrado por género:** los usuarios deben poder filtrar esta lista por género.
- **RF6 – Filtrado por desarrollador:** los usuarios deben poder filtrar esta lista por desarrollador.
- **RF7 – Filtrado por editor:** los usuarios deben poder filtrar esta lista por editor.
- **RF8 – Filtrado por clasificación PEGI:** los usuarios deben poder filtrar esta lista por clasificación PEGI.
- **RF9 – Filtrado por fecha de lanzamiento:** los usuarios deben poder filtrar esta lista por fecha de lanzamiento.
- **RF10 – Búsqueda por título:** debe haber una barra de búsqueda que permita a los usuarios buscar videojuegos por título.
- **RF11 – Adición de videojuegos a la lista personal:** los usuarios deben poder añadir videojuegos a una lista personal única.
- **RF12 – Asignación de estado a los videojuegos:** cada videojuego añadido a la lista debe tener un estado asociado.
- **RF13 – Cambio de estado de los videojuegos:** los usuarios deben poder cambiar el estado de un videojuego en su lista.
- **RF14 – Acceso a la página de detalles del videojuego:** al seleccionar un videojuego de la lista, los usuarios deben poder ver una página de detalles.

- **RF15 – Visualización de la información del videojuego:** la página de detalles debe mostrar información completa sobre el videojuego, incluyendo la carátula del juego, el desarrollador, el editor, el género y la fecha de lanzamiento.
- **RF16 – Descripción del videojuego:** la página de detalles debe incluir una descripción del juego.
- **RF17 – Calificaciones y reseñas de los videojuegos:** los usuarios deben poder calificar los videojuegos en una escala de una a diez estrellas y dejar reseñas escritas. Esta funcionalidad solo debe estar disponible para los videojuegos que el usuario ha marcado como Terminados.
- **RF18 – Visualización de reseñas:** los usuarios deben poder ver las reseñas escritas por otros usuarios para cualquier videojuego en la base de datos de la aplicación, independientemente de si han terminado el juego o no.
- **RF19 – Visualización de la lista personal de videojuegos:** los usuarios deben poder ver su lista personal de videojuegos, que debe estar dividida en secciones según el estado del juego (Jugando, Terminado, Abandonado). Si el estado de un juego cambia, debe moverse automáticamente a la sección correspondiente.
- **RF20 – Consulta de datos personales:** los usuarios deben poder ver sus datos personales, incluyendo su foto de perfil, su nombre de usuario y su dirección de correo electrónico.
- **RF21 – Visualización de estadísticas:** los usuarios deben poder ver una serie de estadísticas relacionadas con su actividad en la aplicación, como el total de juegos añadidos a su lista, el número de juegos Terminados, Jugando y Abandonados, y su calificación media.
- **RF22 – Cambio de foto de perfil:** los usuarios deben poder cambiar su foto de perfil en cualquier momento.
- **RF23 – Cambio de nombre de usuario:** los usuarios deben poder cambiar su nombre de usuario en cualquier momento.
- **RF24 – Eliminación de cuenta:** los usuarios deben poder eliminar permanentemente su cuenta y todos los datos asociados.
- **RF25 – Cierre de sesión:** los usuarios deben poder cerrar sesión en la aplicación y volver a iniciarla sin problemas.

5.1.2 Requisitos no funcionales

- **RNF1 – Seguridad:** la aplicación debe garantizar la seguridad y la privacidad de los datos de los usuarios. Esto incluye la protección de las contraseñas y otros datos sensibles.
- **RNF2 – Rendimiento:** la aplicación debe ser capaz de manejar un gran número de usuarios y de solicitudes simultáneamente sin degradar el rendimiento.
- **RNF3 – Usabilidad:** la interfaz de usuario debe ser intuitiva y fácil de usar. Los usuarios deben poder navegar por la aplicación y utilizar sus funciones sin dificultad.
- **RNF4 – Disponibilidad:** la aplicación debe estar disponible para los usuarios en todo momento. Cualquier tiempo de inactividad debe ser mínimo.
- **RNF5 – Escalabilidad:** la aplicación debe ser capaz de manejar un aumento en el número de usuarios o en la cantidad de datos sin afectar al rendimiento.
- **RNF6 – Compatibilidad:** la aplicación debe ser compatible con los navegadores web más comunes y con diferentes tamaños de pantalla.
- **RNF7 – Mantenibilidad:** la aplicación debe ser fácil de mantener y actualizar. Esto incluye la capacidad de añadir nuevas funciones y de corregir errores de manera eficiente.
- **RNF8 – Fiabilidad:** La aplicación debe ser fiable y consistente en su funcionamiento. Los usuarios deben poder confiar en que la aplicación funcionará como se espera.
- **RNF9 – Accesibilidad:** la aplicación debe ser accesible para todos los usuarios, incluyendo a aquellos con discapacidades.
- **RN10 – Tolerancia a fallos:** la aplicación debe ser capaz de manejar los errores de manera elegante sin interrumpir la experiencia del usuario.
- **RN11 – Capacidad de respuesta:** la aplicación debe responder rápidamente a las solicitudes de los usuarios.
- **RN12 – Cumplimiento de las normativas:** la aplicación debe cumplir con todas las normativas y leyes pertinentes.

Una vez terminado el proceso de toma de requisitos se ideó un diagrama de flujo que representase la estructura de la interfaz de la aplicación tal y como se muestra en la Figura 3:

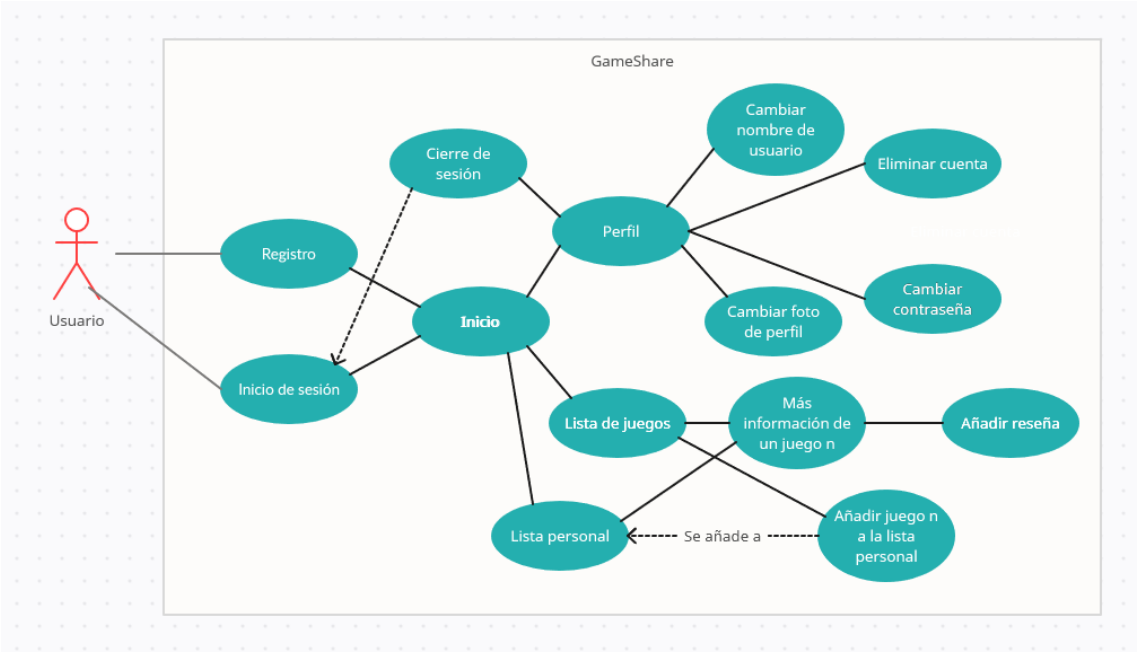

Figura 3. Diagrama de flujo de la navegación del usuario por la aplicación. (Fuente: creación propia mediante la herramienta Creately (Creately, 2024)).

5.2 *Sprint* 1

Una vez finalizada la toma de requisitos, empezó el desarrollo como tal de la aplicación. Se definieron las siguientes tareas para el primer *Sprint*:

Tabla 3. Cuadro que muestra las tareas a completar del primer Sprint. Incluyendo un tiempo de desarrollo estimado y los requisitos funcionales y no funcionales relacionados.

TAREA	TIEMPO ESTIMADO	REQUISITOS FUNCIONALES	REQUISITOS NO FUNCIONALES
Investigación de tecnologías	4 días		RNF1, RNF2, RNF5, RNF6, RNF7
Definición de requisitos	3 días		
Diseño de la interfaz de usuario para el registro e inicio de sesión	1 día	RF1	RNF3, RNF9
Desarrollo del frontend para el registro e inicio de sesión.	4 días	RF1	RNF1, RNF3, RNF6, RNF11

5.2.1 Investigación de tecnologías

Como ya se ha mencionado en un apartado anterior, el primer paso para el desarrollo de la aplicación fue la investigación de las tecnologías escogidas para el desarrollo de este. El principal escollo fue *Firebase*, ya que se contaba con un conocimiento previo bastante extenso de *Angular*.

De tal forma que se explicó anteriormente, *Firebase* es una plataforma de desarrollo de aplicaciones que cuenta con varios servicios para ayudar a los desarrolladores a construir sus aplicaciones. Una de las formas en las que *Firebase* facilita el desarrollo es mediante algunos módulos específicos de funcionalidades. Estos son algunos ejemplos:

- ***Firestore Database*:** una base de datos alojada en la nube.
- ***Firebase Authentication*:** proporciona servicios de autenticación a usuarios por correo electrónico, teléfono, Google, etc.
- ***Firebase Hosting*:** proporciona servicios de alojamiento de páginas web.
- ***Firebase Cloud Storage*** (Firebase, 2024)**:** servicio de almacenamiento de objetos que permite a los desarrolladores subir y descargar archivos.
- ***Firebase Analytics*:** proporciona informes detallados sobre la actividad y el uso de la aplicación.

Una vez exploradas todas las posibilidades que ofrecía esta potente herramienta se decidió por el uso de tres módulos: *Firestore*, *Authentication* y *Storage*.

Firestore servirá como base de datos de la aplicación, donde se cargarán todos los juegos, los usuarios y las listas de cada usuario. *Authentication* se utilizará para autenticar fácilmente las identidades de los usuarios, tanto en el inicio de sesión como en el registro de estos, facilitando el desarrollo de este apartado. Por último, *Storage* guardará las fotos de perfil que los usuarios quieran ponerse en la aplicación.

Firebase cuenta con una 'consola' que permite a los desarrolladores de las aplicaciones mantener el control absoluto de la aplicación, pudiendo acceder fácilmente a cada uno de los aspectos de *Firebase* además de la capacidad de añadir nuevos módulos con tutoriales muy fáciles de seguir. En la Figura 4 podemos ver un ejemplo de la consola de la aplicación:

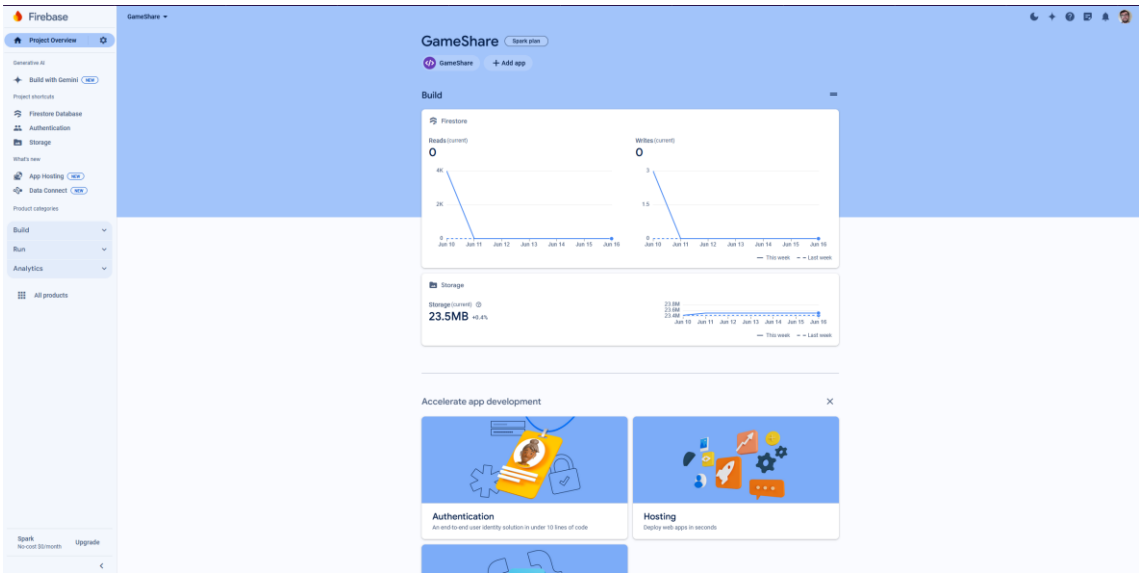

Figura 4. Consola de mandos de Firebase. (Fuente: Elaboración propia.)

En la barra lateral izquierda se encuentran todos los módulos activos en nuestra aplicación actualmente, mientras que en el panel central se nos muestra información relacionada con dichos módulos. En este caso podemos observar la cantidad de lecturas y escrituras en la base de datos y el espacio de almacenamiento que utiliza nuestra aplicación.

5.2.2 Definición de requisitos

Una vez terminada la investigación de las distintas tecnologías, se procedió a la toma de requisitos de la aplicación. Teniendo en cuenta la información obtenida en la exploración de aplicaciones alternativas y la indagación en *Firebase*, se formularon en total veinticinco requisitos funcionales y doce requisitos no funcionales, los cuales se expusieron en el apartado anterior.

5.2.3 Diseño de la interfaz de usuario para el registro e inicio de sesión

Antes del comienzo propio del desarrollo se bocetó la pantalla de inicio de sesión y registro de usuario. Para esta aplicación se requería un diseño simple, sin elementos innecesarios e intentando mantener un estilo elegante. Es por eso por lo que se decidió que la aplicación solo tendría tres colores: blanco, negro y un color azul claro con toques de cian (`#29ABE2`, `RGB (41, 171, 226)`), que actuaría como color de énfasis.

Una vez tomada esta decisión, finalmente comenzó el desarrollo de la aplicación con un boceto de las pantallas, empezando con el inicio de sesión, tal y como se muestra en la Figura 5:

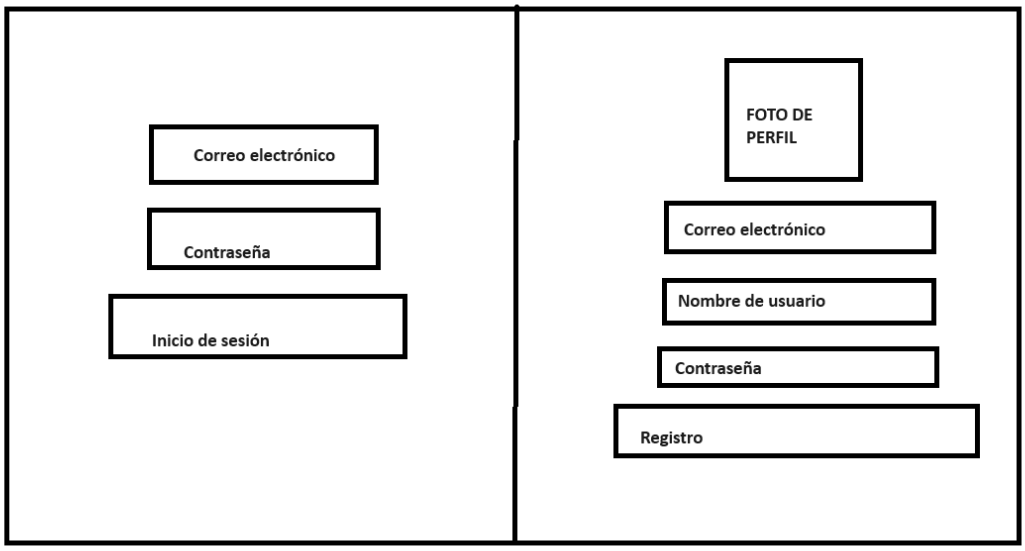

Figura 5. Boceto página registro e inicio de sesión. (Fuente: creación propia mediante la herramienta Paint (Microsoft Corporation, 2024)).

5.2.4 Desarrollo del *frontend* para el registro e inicio de sesión

El desarrollo con *Angular* empezó con la creación de la página de registro de usuarios. Al contrario que lo mostrado en el boceto, finalmente se decidió por dividir el inicio de sesión y el registro en dos pantallas distintas, como se muestra en la Figura 6:

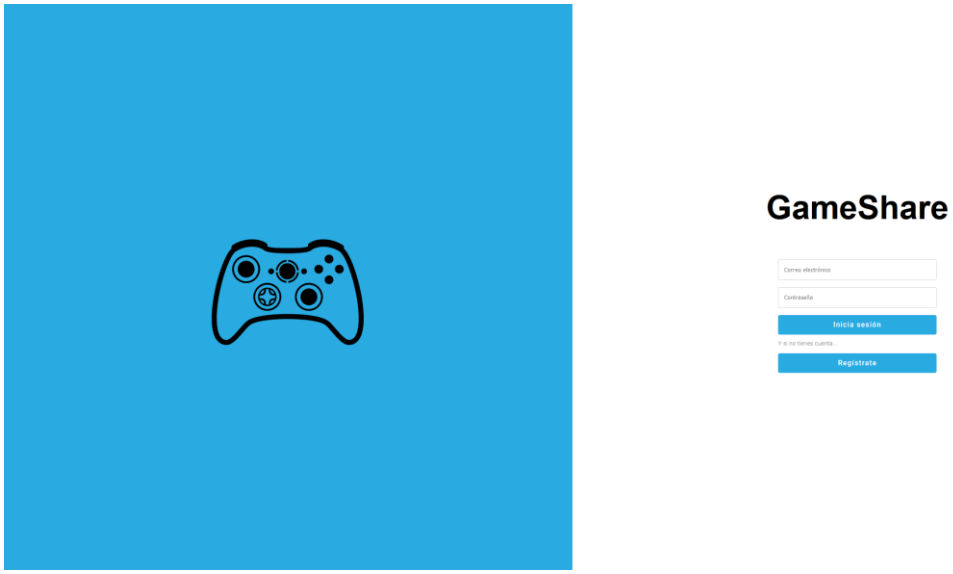

Figura 6. Página de inicio de sesión de la aplicación. (Fuente: Elaboración propia).

Esta es la primera pantalla que se encontrarán los usuarios nada más abrir un enlace a la aplicación. Se les pedirá un correo electrónico y una contraseña para acceder a *GameShare*. Si coinciden con los guardados en *Firebase Authentication*, podrán entrar en

el resto de la aplicación, en caso contrario se mostrará un aviso en la parte inferior de la pantalla.

Por otro lado, si el usuario todavía no está registrado en la aplicación podrá hacerlo fácilmente pulsando el botón de registro, que mostrará la siguiente pantalla. En la Figura 7 podemos apreciar cómo se verá esta pantalla:

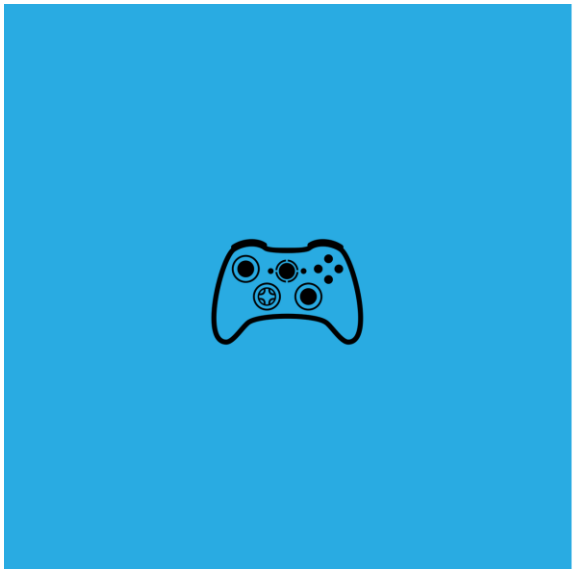

Figura 7. Página de registro de usuarios de la aplicación. (Fuente: Elaboración propia).

En esta pantalla el usuario escogerá su foto de perfil, su correo electrónico, su nombre de usuario y su contraseña. El botón de registro se bloqueará automáticamente hasta que el cliente rellene todos los campos.

5.3 *Sprint* 2

Tabla 4. Cuadro que muestra las tareas a completar del segundo Sprint. Incluyendo un tiempo de desarrollo estimado y los requisitos funcionales y no funcionales relacionados.

TAREAS	ESTIMACIÓN	REQUISITOS FUNCIONALES	REQUISITOS NO FUNCIONALES
Creación de la base de datos	6 días	RF4	RNF1, RNF2, RNF5
Integración con *Firebase*	8 días		RNF1, RNF2, RNF4,RNF5, RNF10
Implementación del registro de usuarios	3 días	RF1	RNF1, RNF3, RNF11
Implementación del inicio de sesión	1 día	RF1	RNF1, RNF3, RNF11
Creación de la pantalla de lista de videojuegos	2 días	RF2, RF4	RNF3, RNF9, RNF11

5.3.1 Creación de la base de datos

Como se ha mencionado anteriormente, *Firestore Database* es un módulo de *Firebase* que permite la creación de una base de datos NoSQL en la nube. *Firestore* funciona mediante un sistema de documentos, que se organizan en colecciones. Cada documento tiene un conjunto de pares clave-valor. Además, permite actualizaciones en tiempo real y funciona sin conexión ya que los datos se almacenan en el caché de la aplicación.

Para la aplicación se crearon tres colecciones generales: juegos, usuarios y *reviews*.

En ‘juegos’ se encuentran una serie de documentos cada uno con un ID único. En cada documento se almacena la información relativa a ese juego en concreto como, por ejemplo, su título, PEGI, fecha de salida, etc. además de dos enlaces que servirán para cargar las imágenes de las caratulas y las portadas de los videojuegos. Toda la información fue sacada directamente de la página de venta de videojuegos *Steam*. A continuación, en la Figura 8, podemos ver un ejemplo de la base de datos:

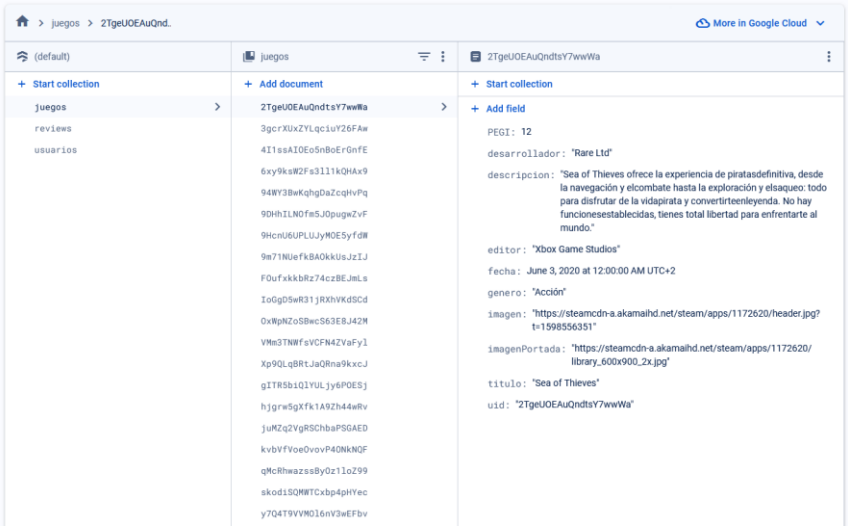

Figura 8. Captura de la colección de videojuegos en Firestore. (Fuente: Elaboración propia).

En 'usuarios' almacenamos los datos de los usuarios registrados en la aplicación. No se guardan las contraseñas ya que de esa parte se encarga *Authentication*, pero sí que almacena otros datos relevantes como su nombre o su foto de perfil en forma de enlace para poder recuperarlos de una manera sencilla. En la Figura 9 se muestra un ejemplo de la colección Usuarios:

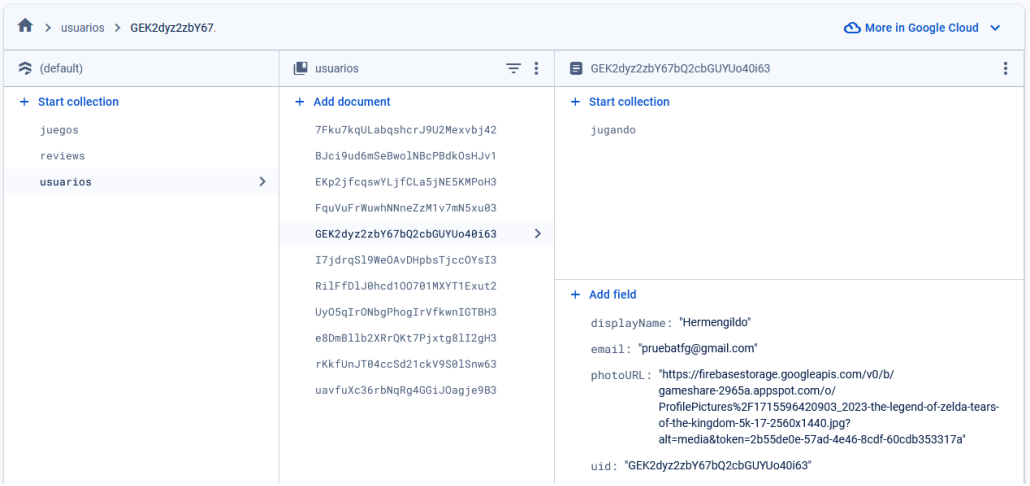

Figura 9. Captura de la colección de usuarios en Firestore. (Fuente: Elaboración propia).

Cuando los usuarios añaden un juego de la lista general a su propia lista personal, automáticamente se crea un subconjunto llamado 'jugando' en el documento del usuario que haya decidido agregar ese videojuego. Este subconjunto está formado por un pequeño extracto del videojuego (su nombre y su ID) y el estado en el que se encuentra para el jugador (Terminado, Jugando o Abandonado) para poder clasificarlo correctamente en cada lista. Podemos ver un ejemplo de un subconjunto en la Figura 10:

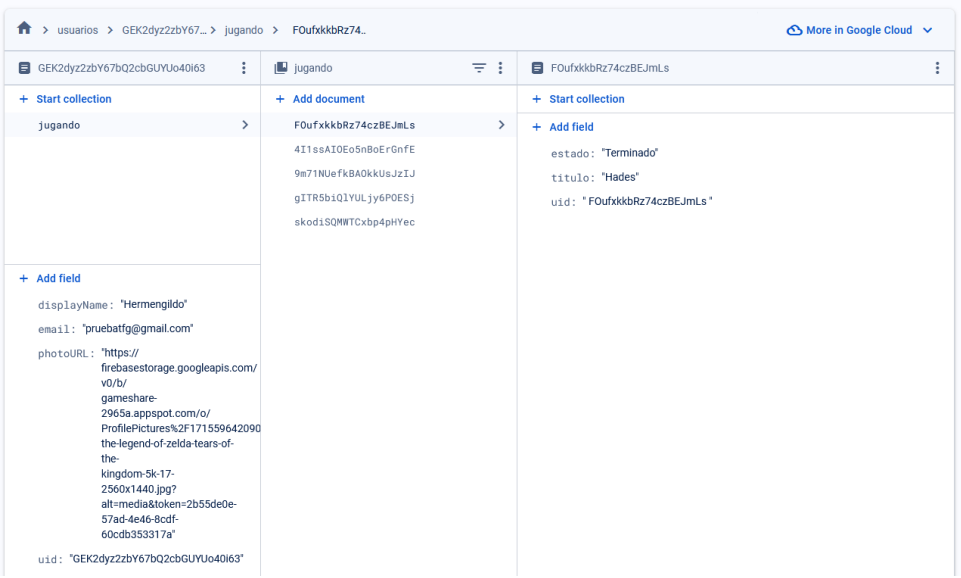

Figura 10. Captura del subconjunto 'Jugando' de un usuario en Firestore. (Fuente: Elaboración propia).

Por último, la colección *'reviews'* permite almacenar las reseñas que los usuarios pueden dejar en los videojuegos que ya han finalizado. Esta se compone de distintos documentos formados por el ID del juego y del usuario, el nombre del usuario, la fecha de la reseña, la nota que le ha dado y la reseña en texto que ha decidido dejar. Un usuario es capaz de dejar varias reseñas en el mismo videojuego. En la Figura 11 se observa un ejemplo de la colección '*reviews'*:

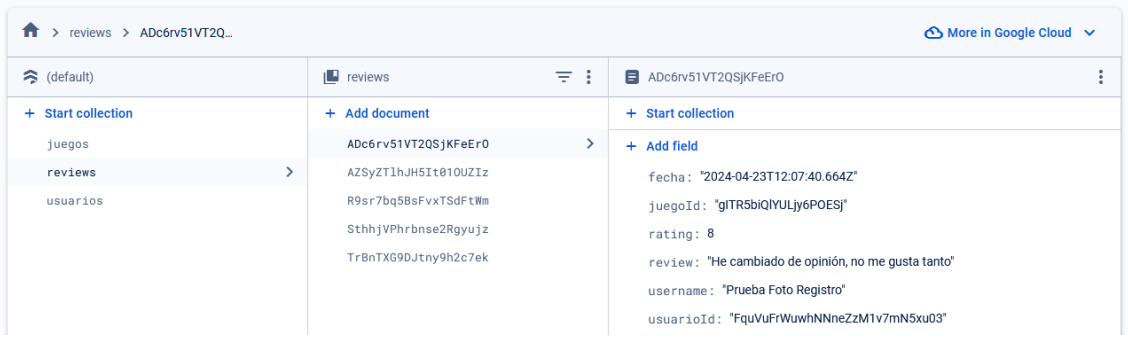

Figura 11. Captura de la colección de reseñas de Firestore. (Fuente: Elaboración propia).

Firestore permite la creación de reglas para controlar el acceso a la base de datos. Es una forma para determinar quién tiene permiso para leer y/o escribir en esta. Para el funcionamiento normal de la aplicación se crearon varias normas.

Por último, *Firestore* cuenta con índices para mejorar el rendimiento de las consultas a base de datos. Por ejemplo, para poder crear un filtro que permitiese filtrar a los usuarios por distintos campos como el PEGI o el género del videojuego, se crearon varios índices.

De esta descripción podríamos obtener el siguiente esquema UML, mostrado en la Figura 12:

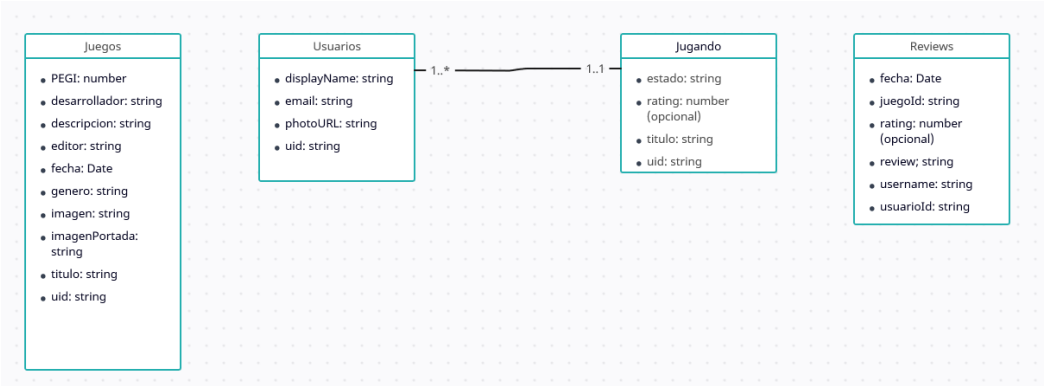

Figura 12. Diagrama UML de la base de datos de Firestore. (Fuente: Creación propia mediante la herramienta Creately).

Al tratarse de una base de datos NoSQL basada en colecciones y documentos no existen relaciones entre sí. La única relación existente sería entre cada usuario y su propio subconjunto de videojuegos, la cual se representaría mediante una relación uno a muchos (1: n).

5.3.2 Integración con *Firebase*

Afortunadamente la integración de la aplicación existente de *Angular* con *Firebase* fue muy sencilla gracias a la gran documentación y explicaciones paso por paso que se encuentran en la página web oficial de *Firebase*. Una vez descargadas las librerías necesarias, incluyendo la mencionada anteriormente *AngularFire*, fue muy sencillo conectar con éxito las dos aplicaciones.

5.3.3 Implementación del registro de usuarios

Para poder registrar a los usuarios en la aplicación de *Firebase* se necesitaba el módulo *Authentication*. Como ya se ha explicado anteriormente, *Authentication* permite la autenticación de usuarios en la aplicación mediante distintos medios como correo y contraseña, *Google*, *Twitter*, etc. Mientras que *Authentication* permite muchos tipos distintos de formas de registro, en la aplicación solo se permite mediante correo electrónico y contraseña.

Partiendo de lo desarrollado en el Sprint anterior, se implementó una función de registro de usuarios. Gracias a la librería *AngularFire* (AngularFire, 2024) esta tarea se puede

realizar llamando a la función `createUserWithEmailAndPassword` pasando como parámetros el email y la contraseña que ha introducido el usuario.

Por otro lado, el usuario puede seleccionar una foto de su propio PC para usarla como imagen de su perfil. Para poder cargar las fotos y luego recuperarlas se hizo uso del módulo de *Firebase Storage*.

Storage permite el almacenamiento de distintos tipos de datos para poder subirlos y descargarlos en cualquier momento y situación. En el caso de la aplicación solo se utilizó para guardar las imágenes de perfil de los usuarios. Cuando un usuario decide registrarse usando una foto de perfil esta se almacena en una carpeta dentro de *Storage* llamada '*ProfilePictures'*. Para evitar problemas de conflictos por el nombre de las imágenes, este se guarda con la fecha actual en la que se ha subido la imagen. He aquí un ejemplo de *Storage* mediante la Figura 13:

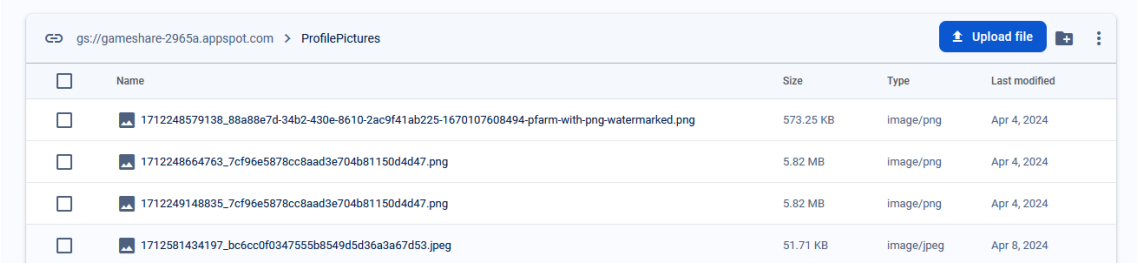

Figura 13. Muestra de las imágenes guardadas en la nube de Firebase Storage. (Fuente: Elaboración propia).

Volviendo al registro, una vez el usuario ha sido dado de alta en *Authentication* se guarda un nuevo documento en la colección 'usuarios' de *Firestore* indicando el nombre, el correo electrónico y el ID, además de un enlace a la imagen de perfil guardada en Storage para poder recuperarla cómodamente.

Si el usuario se ha podido registrar correctamente, se accederá a la página de inicio de la aplicación. En caso de errores, como error de *Firebase* o correo electrónico repetido, se mostrará un mensaje de error al usuario.

5.3.4 Implementación del inicio de sesión

Al igual que existe una función para registrar usuarios en *Authentication*, también existe una función para iniciar sesión a los usuarios si introducen el correo electrónico y la contraseña correctas. Esta función, que también se encuentra en la librería *AngularFire*

se llama `signInWithEmailAndPassword`. Si la comprobación es correcta, permitirá al usuario acceder a la aplicación con el usuario previamente creado, si es incorrecta mostrará un mensaje de error.

5.3.5 Creación de la lista de videojuegos

La última tarea del segundo *Sprint* consistía en la creación de una lista general de videojuegos, que mostrase todos los que se encuentran en la colección 'juegos' para que el usuario pudiera decidir cuáles escoger para su propia lista personal.

En primer lugar, se creó una barra lateral superior horizontal para poder navegar de la forma más cómoda posible. Cada elemento de esta barra permitiría el acceso a una parte distinta de la aplicación, como se puede ver a continuación en la Figura 14:

Figura 14. Barra lateral para desplazarse por la aplicación. (Fuente: Elaboración propia).

Como de costumbre se empezó el desarrollo de la pantalla bocetando, tal y como se puede ver en la Figura 15:

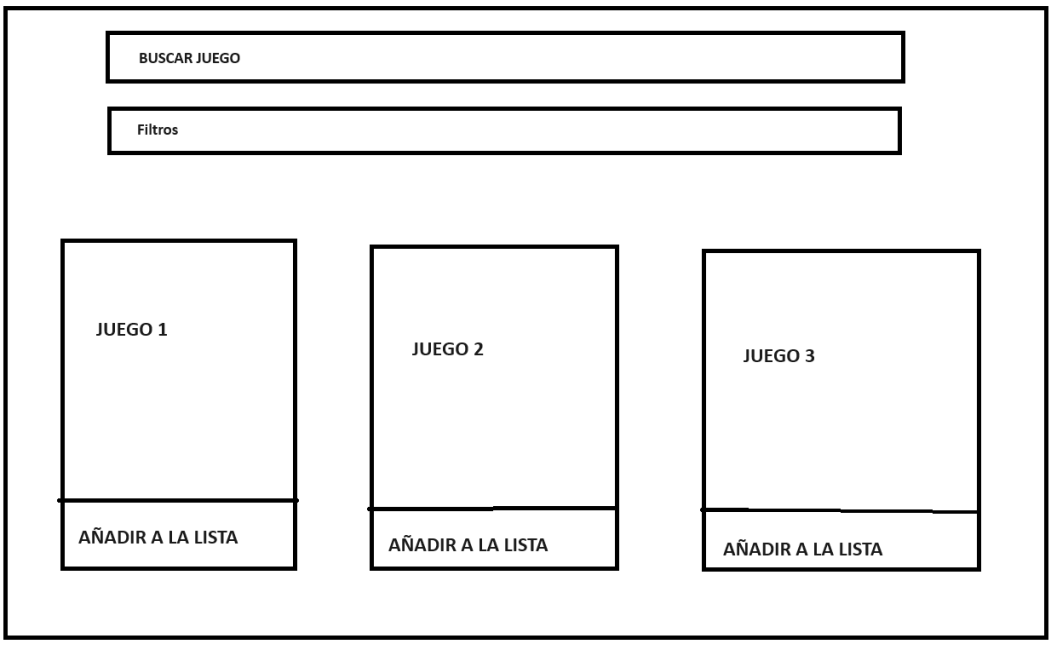

Figura 15. Boceto de la pantalla que muestra todos los videojuegos de la aplicación. (Fuente: creación propia con la herramienta Paint).

Cada juego debería estar dentro de una 'tarjeta' mostrando la carátula de este, para dar información visual al usuario, además de la opción de añadirlo a su lista personalizada.

También tendría que haber un botón en cada una de las tarjetas que, una vez pulsado mostrase la información guardada en la colección ‘juegos’ de la base de datos del juego en concreto.

Por último, tendría que haber una opción para poder buscar por título del juego y poder filtrar por distintos datos de este como el género o el desarrollador. Para esta primera fase de esta pantalla únicamente se implementó con funcionalidad la lista completa de videojuegos tal y como podemos ver en la Figura 16:

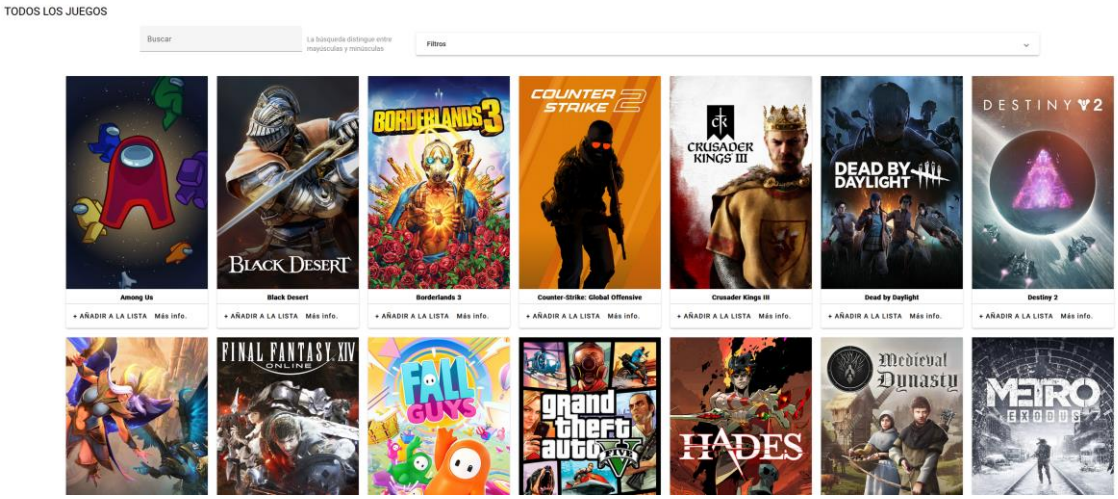

Figura 16. Muestra de videojuegos recuperados de Firestore en la pantalla de la aplicación. (Fuente: Elaboración propia).

Además, se añadió una opción para paginar los juegos. Al contar solo con veinte elementos dentro de la base de datos de ‘juegos’ se muestran todos los videojuegos uno detrás de otro en orden alfabético. Sin embargo, también podemos hacer que se muestren de cinco en cinco o de diez en diez, como se muestra en la Figura 17:

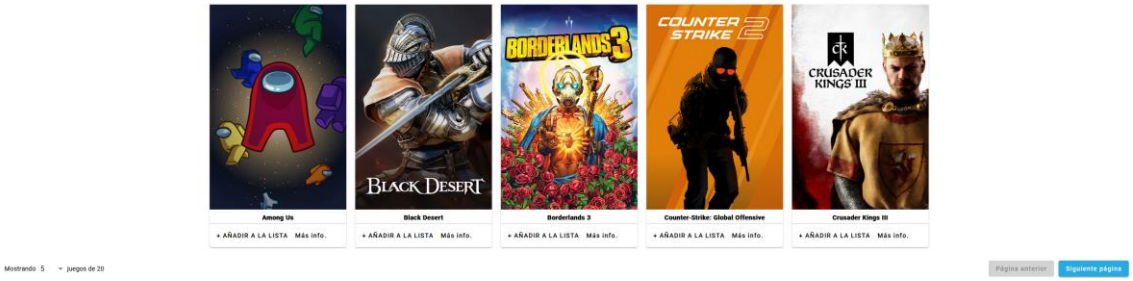

Figura 17. Muestra de videojuegos recuperados de Firestore en la pantalla de la aplicación paginados. (Fuente: Elaboracion propia).

Para poder acceder al resto de videojuegos se tendrá que pulsar el botón de ‘Siguiente Página’, que cargará los cinco siguientes en la lista y los mostrará por pantalla.

Como se ha comentado, el principal objetivo de esta tarea era poder recuperar los datos completos de la colección 'juegos' y mostrarlos en pantalla, sin ninguna funcionalidad más por el momento.

Para poder recuperar todos los videojuegos se ha creado una consulta que recuperase todos los videojuegos de la colección 'juegos' y los ordenase por el campo título, tal y como se haría en una consulta de base de datos normales. Para poder hacer esto se ha utilizado nuevamente la librería *AngularFire* (AngularFire, 2024), la cual permite acceder a las colecciones de nuestra aplicación. Veamos la Figura 18 para aprender cómo funciona:

```
this.afs.collection('juegos', ref => ref.orderBy('titulo')
```

Figura 18. Ejemplo de código para recuperar todos los videojuegos de la colección 'juegos' ordenados por 'titulo'. (Fuente: Elaboración propia).

En primer lugar, necesitamos una referencia a *Angular Firestore* con `this.afs`. Utilizando `collection` y especificando el nombre de la colección podemos acceder a los datos de esta. Además, en este caso para recibir todos los juegos ordenados por su campo `titulo` podemos añadir una condición a través de `orderBy`, como si se tratase de una consulta SQL. También podemos poner otro tipo de condicionantes que existen en las consultas SQL por lo que tenemos prácticamente la misma funcionalidad en este aspecto.

Una vez finalizada la página de juegos se dio por finalizado el segundo sprint.

5.4 *Sprint* 3

Tabla 5. Cuadro que muestra las tareas a completar del tercer Sprint. Incluyendo un tiempo de desarrollo estimado y los requisitos funcionales y no funcionales relacionados.

TAREAS	ESTIMACIÓN	REQUISITOS FUNCIONALES	REQUISITOS NO FUNCIONALES
Desarrollo de la pantalla de inicio	2 días	RF3	RNF3, RNF9
Implementación del filtrado de la lista de videojuegos	3 días	RF5, RF6, RF7, RF8, RF9	RNF3, RNF11
Desarrollo de la funcionalidad de búsqueda por título	1 día	RF10	RNF3, RNF11
Implementación de la lista personal de videojuegos	3 días	RF11, RF12, RF13, RF19	RNF1, RNF3, RNF11
Desarrollo de la página de detalles del videojuego	2 días	RF14, RF15, RF16, RF18	RNF3, RNF9,
Implementación de calificaciones y reseñas de videojuegos	2 días	RF17	RNF1, RNF3, RNF11
Desarrollo de la funcionalidad de consulta de datos personales y visualización de estadísticas	2 días	RF20, RF21	RNF1, RNF3,
Implementación del cambio de foto de perfil, contraseña y nombre de usuario	1 día	RF22, RF23	RNF1,
Desarrollo de la funcionalidad de eliminación de cuenta	1 día	RF24	RNF1

Implementación del cierre de sesión	1 día	RF25	RNF1

5.4.1 Desarrollo de la pantalla de inicio

Uno de los aspectos más importantes de cualquier aplicación es su pantalla de inicio. Al ser, normalmente, lo primero que ve el usuario nada más iniciar sesión debe contener la información necesaria para explicar el funcionamiento de la aplicación, pero sin llegar a abrumar con opciones y textos.

Siguiendo con la estética minimalista de la aplicación se bocetó la pantalla de inicio tal y como se puede apreciar en la Figura 19:

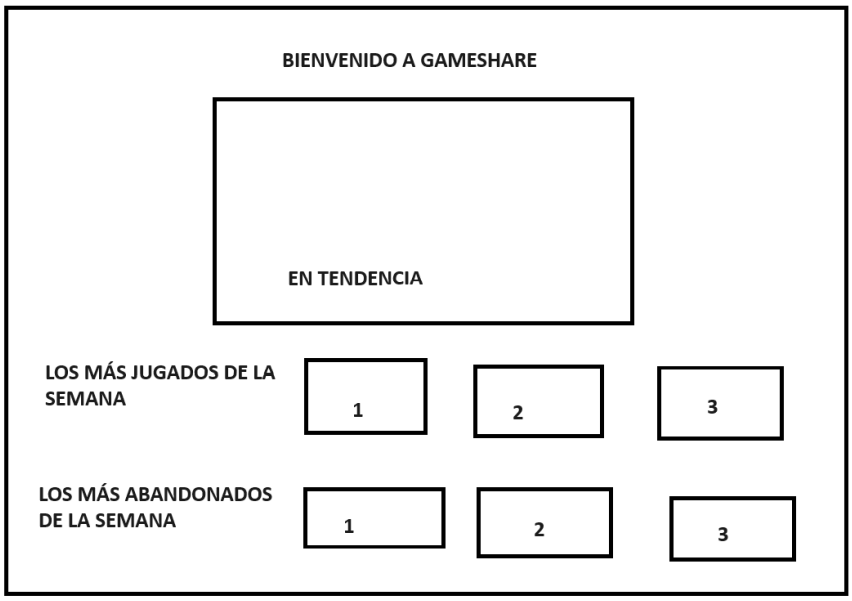

Figura 19. Boceto de la pantalla de inicio de la aplicación. (Fuente: creación propia con la herramienta Paint).

En la fase de investigación del proyecto, cuando se investigaron las distintas alternativas a la aplicación de este documento, las páginas de inicio consistían en tres cosas: la lista personal del usuario, recomendaciones personalizadas o los videojuegos más populares del momento. Para esta aplicación se decidió utilizar un acercamiento a la tercera opción. Sin embargo, al no contar con usuarios activos en esta demo para poder enseñar esa parte, se ha optado por recuperar de la base de datos diez videojuegos aleatorios y mostrar sus portadas. Una vez se tengan usuarios auténticos esta funcionalidad será sustituida por los verdaderos juegos más populares.

Por último, se consideró la idea de crear un carrusel de imágenes imitando el estilo de las aplicaciones de *streaming* populares como *Disney+*. No obstante, esta idea fue rápidamente rechazada debido a que las librerías encontradas para esta funcionalidad no funcionaban adecuadamente. Se considerará añadirlo en una actualización futura al igual que la opción anterior.

Teniendo en cuenta lo expuesto en los párrafos anteriores, la Figura 20 muestra el diseño final de la pantalla de inicio:

Figura 20. Página de inicio de la aplicación. (Fuente: Elaboración propia).

En primer lugar, se estimó necesario sustituir el 'bienvenido' mostrado en el boceto de la pantalla por un 'te damos la bienvenida', manteniendo un género neutral en las palabras. Además, se mantiene la misma estética de colores con el color azul.

5.4.2 Implementación del filtrado de la lista de videojuegos

Partiendo de la lista de videojuegos general se decidió añadir un filtro por distintos atributos de cada videojuego para que el usuario pueda utilizarlos para que solo se muestren los videojuegos de aventuras o los desarrollados por *Valve*. Se añadieron cinco filtros distintos: género, desarrollador, editor, PEGI y fecha de salida.

En primer lugar, se recuperan todos los atributos no repetidos de cada uno de los campos del filtro de todos los videojuegos ubicados en la base de datos y se asignan a una lista, menos la fecha de salida. Este campo permite la selección de dos fechas y los juegos seleccionados se ubicarán entre estas dos fechas.

Además, es importante destacar que los filtros pueden ser utilizados conjuntamente. Por ejemplo, la Figura 21 muestra los videojuegos de género 'Acción' que salieron al mercado entre el 18/06/2017 y el 18/06/2020:

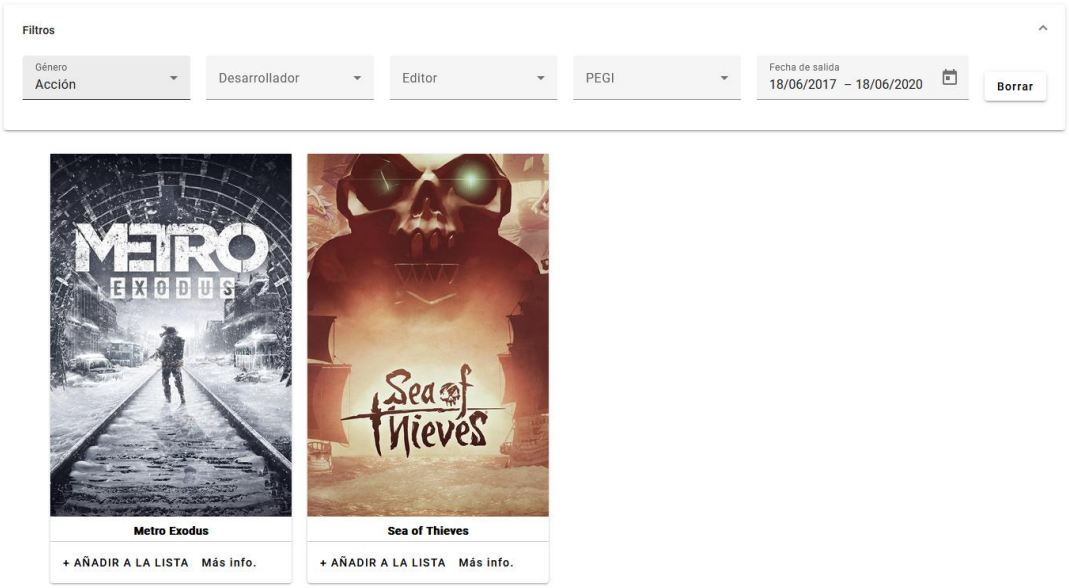

Figura 21. Muestra de videojuegos filtrados por género y fecha de salida. (Fuente: Elaboración propia).

Al contrario que en muchas aplicaciones, el filtrado ocurre de forma automática, sin tener que darle a ningún botón gracias a las tecnologías de *Angular*. Además, se ha añadido un botón de 'Borrar' para eliminar todos los filtros y volver fácilmente a la lista general.

5.4.3 Desarrollo de la funcionalidad de búsqueda por título

Un elemento esencial en cualquier aplicación de este tipo es la barra de búsqueda. Sin embargo, durante su implementación, se encontró con una limitación de *Firebase*. Al igual que con los demás componentes del filtro, la búsqueda se lleva a cabo mediante la creación de una consulta que devuelve una lista de resultados. Dicha consulta se compara con el campo 'titulo' de la colección 'juegos', y para que encuentre una coincidencia, debe respetar el mismo uso de mayúsculas y minúsculas.

Dado que esta es una característica inherente de *Firebase*, se decidió implementar una solución alternativa: colocar un aviso que indique que el campo de búsqueda distingue entre mayúsculas y minúsculas. Al igual que los filtros, la búsqueda se realiza automáticamente a medida que el usuario introduce caracteres. A continuación, la Figura 22 muestra un ejemplo de cómo buscar juegos que comiencen con las letras 'De':

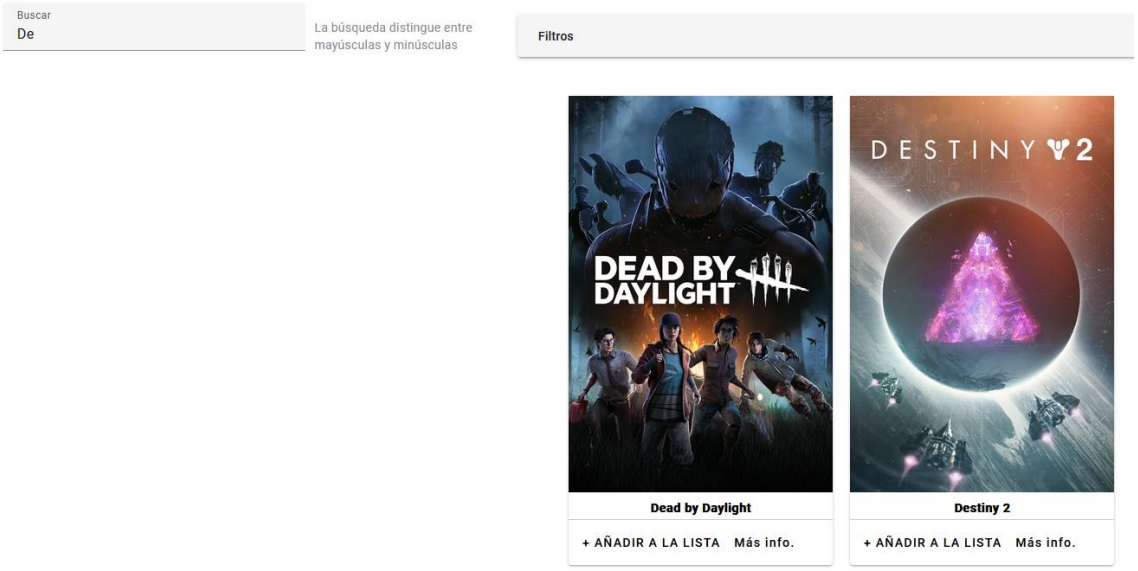

Figura 22. Muestra de videojuegos filtrados por el titulo comenzando por las letras 'De'. (FuenteElaboración propia).

5.4.4 Implementación de la lista personal de videojuegos

Como se ha podido ver en las 'tarjetas' de los videojuegos en los anteriores apartados, justo debajo del título se encuentra un botón llamado '+ AÑADIR A LA LISTA'. Si el usuario decide pulsarlo, ese videojuego se añadirá automáticamente a su lista personal de videojuegos en el estado 'Jugando'.

Una de las ventajas de *Firestore* es que permite crear colecciones y subcolecciones 'al vuelo', es decir, de forma instantánea. Cuando un jugador añade su primer videojuego a la lista personal, automáticamente se crea un subconjunto llamado 'Jugando' a su documento de usuario dentro de la colección 'usuarios'. Dentro de este subconjunto se añade un documento con el nombre del videojuego, el ID del juego y del usuario y su estado. Este estado determina en qué 'sección' se añade el videojuego cuando el usuario quiera ver su lista personal.

Cada juego puede tener en total tres estados: Jugando, Terminado y Abandonado. Cuando el usuario añade un videojuego a su lista automáticamente se añade con el estado en Jugando. El usuario puede cambiar fácilmente el estado de este seleccionando sobre el icono de Jugando, que ahora sustituye al botón de añadir a la lista. De aquí se mostrará un desplegable mostrando los distintos estados que se han comentado permitiendo cambiarlo fácilmente. Ninguno de los estados es fijo, se pueden cambiar en cualquier momento y entre estados.

En la Figura 23 se puede apreciar cómo se verían los videojuegos cuando un usuario los añada a su lista personal:

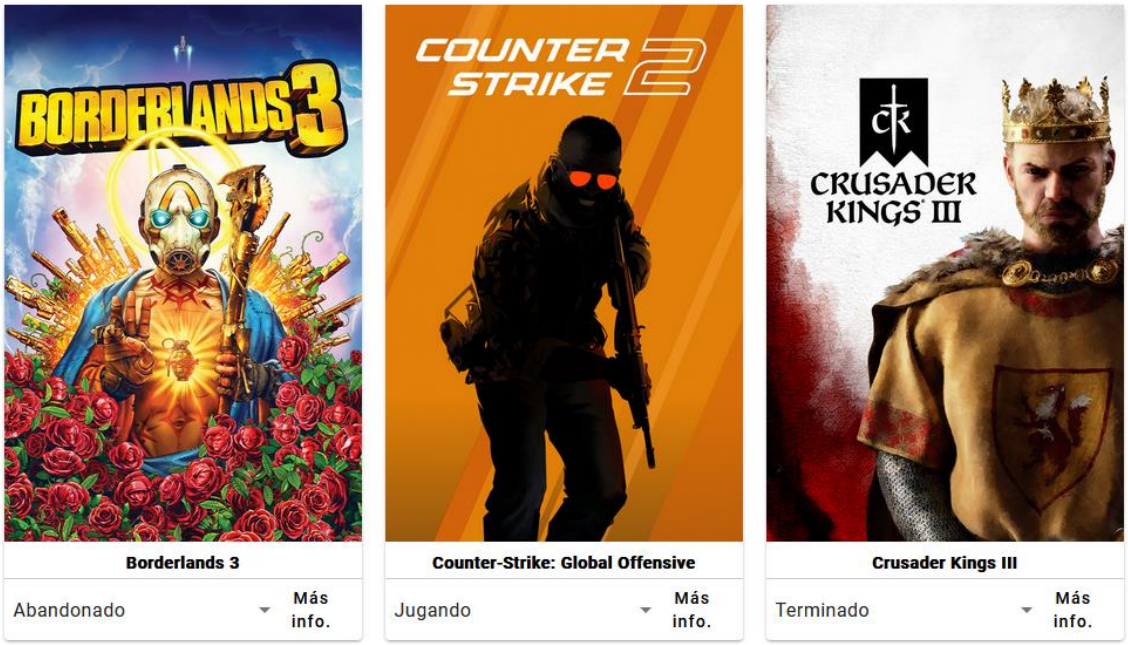

Figura 23. Ejemplo de tres videojuegos añadidos a la lista personal del usuario, cada uno con un estado diferente. (Fuente: Elaboración propia).

5.4.5 Desarrollo de la página de detalles del videojuego

A partir de las tarjetas mencionadas anteriormente se decidió crear una pantalla emergente que contuviese cierta información relacionada con el videojuego que podrían ser de importancia a la hora de jugarlo, como el desarrollador o una pequeña descripción de este. Esta información es obtenida de la base de datos de 'juegos' y mostrada al usuario siguiendo la tónica habitual de la aplicación. En la Figura 24 podemos ver un ejemplo con el videojuego '*Hades*':

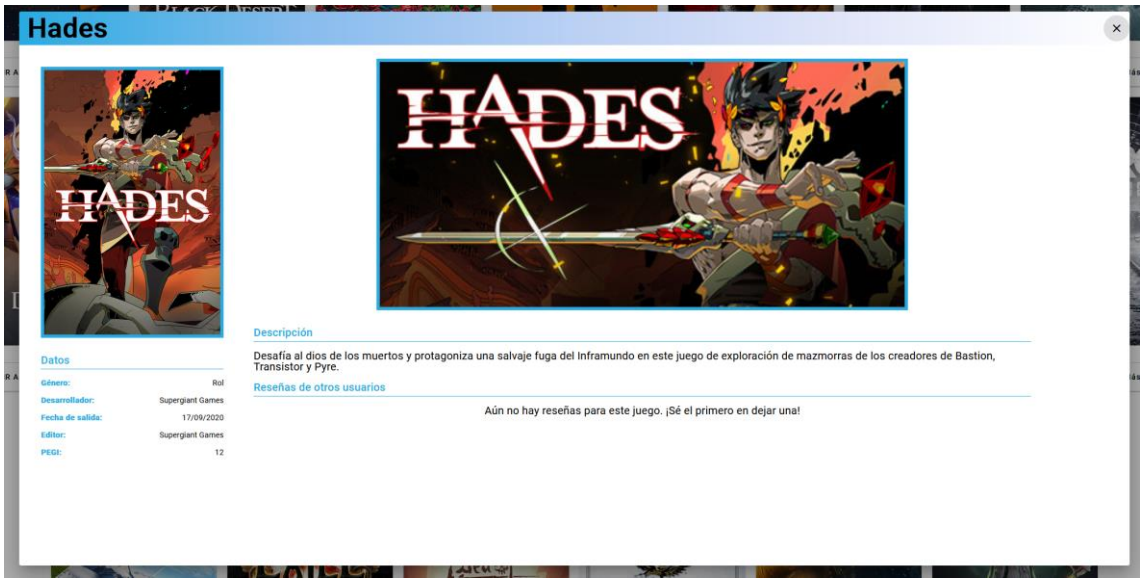

Figura 24. Pantalla de detalles del videojuego 'Hades'. (Fuente: Elaboración propia).

Además, en esta pantalla se incluyen las posibles reseñas, de las cuales se hablará en el siguiente apartado, dejadas por los usuarios. En caso de no haber se muestra un mensaje tal y como podemos ver en la captura de pantalla superior.

5.4.6 Implementación de calificaciones y reseñas de videojuegos

Como se ha mencionado anteriormente, el apartado de reseñas es de especial relevancia en la aplicación, ya que permite la compartición de experiencias entre los usuarios de la página web.

Cualquier usuario puede dejar una o varias reseñas en todos los videojuegos disponibles en la aplicación, pero este debe cumplir una condición. El usuario debe tener el videojuego en su lista para que se active la opción de dejar evaluaciones. Gracias a la tecnología de *Angular* podemos ocultar estos aspectos de la aplicación hasta que se cumplan las condiciones. Hay que destacar que, aunque el usuario no pueda escribir su propia reseña en un videojuego si este no se encuentra en su lista, sí que podrá ver las reseñas dejadas por otros usuarios en ese mismo videojuego.

Aparte de las reseñas, también se incluye un sistema de evaluación basado en estrellas, implicando que una estrella equivale a un uno y diez estrellas a un diez. Al igual que las reseñas esta opción solo se activará si se cumplen los requisitos mencionados anteriormente. Una vez el usuario escoja una nota, esta se guardará automáticamente en el documento asociado en su lista personal. Esta calificación no es invariable con lo que

si el usuario cambia de opinión puede ser modificada sin inconvenientes, sobrescribiendo la calificación en el documento de su colección 'jugando'.

En caso de cumplir las condiciones, se mostrará un cuadro de texto donde el usuario podrá escribir sus pensamientos sobre ese videojuego y publicarlo. Todas las reseñas se almacenan en una colección 'reseñas' de *Firestore* y un mismo usuario puede dejar varias reseñas de un mismo videojuego, sin límite. Cada reseña se almacenará con el ID del usuario, el ID del videojuego, la fecha y hora exactas en las que se hizo y la calificación, en caso de haber. Una vez publicada la reseña se mostrará con el nombre de usuario, la calificación, la fecha de la publicación y la reseña en sí. La Figura 25 muestra un ejemplo:

Figura 25. Pantalla de reseñas de un videojuego de la aplicación. (Fuente: Elaboración propia).

5.4.7 Desarrollo de la funcionalidad de consulta de datos personales y visualización de estadísticas

Una vez terminado el desarrollo del resto de páginas de la aplicación era necesaria una opción de consulta y modificación de datos personales para la comodidad del usuario. Además, como un extra, se decidió añadir una pequeña sección de estadística.

Utilizando un esquema parecido al usado en la pantalla de información, se creó una última pantalla de para mostrar el perfil del usuario. Como el resto de las pantallas el primer paso fue un boceto de esta, tal y como se puede observar en la Figura 26:

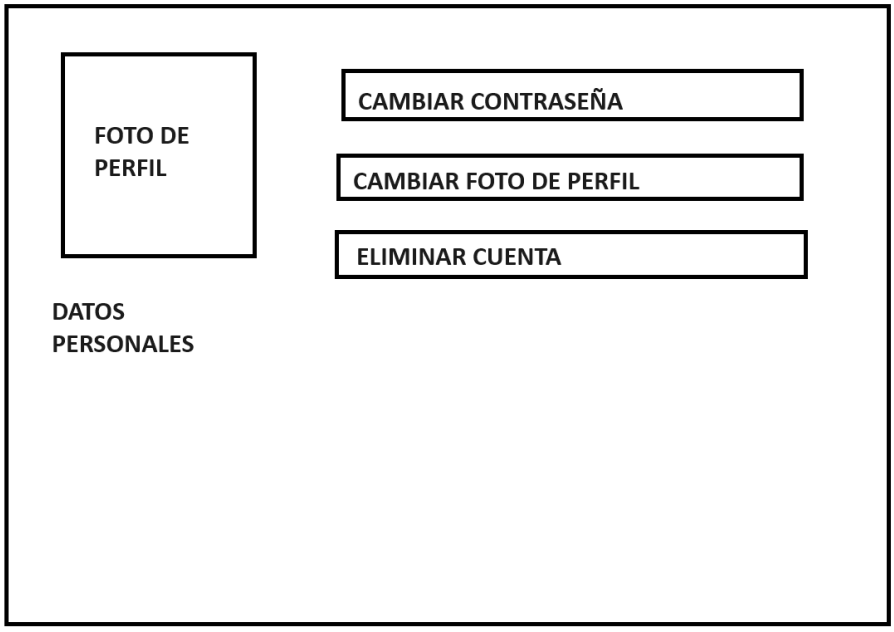

Figura 26. Boceto de la pantalla de perfil de la aplicación. (Fuente: creación propia con la herramienta Paint).

Una vez hecho el boceto se procedió a la realización de la página. En la Figura 27 se puede apreciar el resultado final:

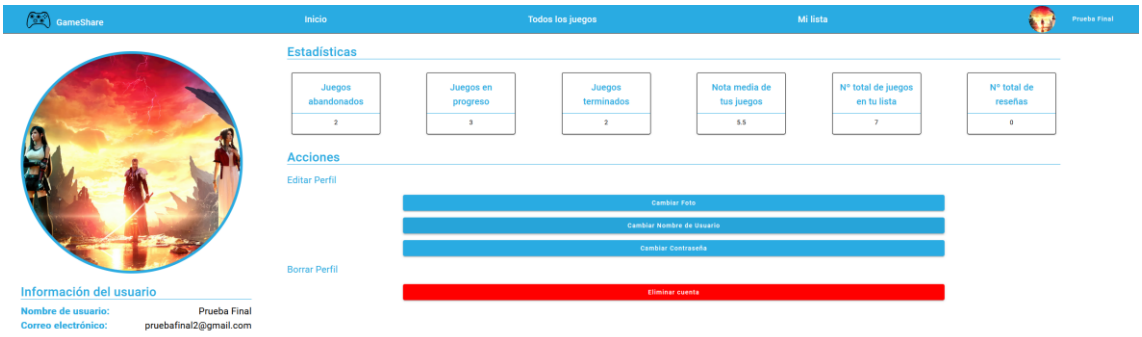

Figura 27. Pantalla perfil del usuario, (Fuente: Elaboración propia).

Tal y como se puede apreciar, en el boceto de esta página no se incluyó la mencionada sección de estadísticas, ya que fue planificada posteriormente para darle más contenido a la sección de perfil.

Estas estadísticas se nutren del documento personal de cada jugador y se actualizan automáticamente conforme cambian los datos de este. Las estadísticas elegidas para mostrar al usuario son las siguientes:

- Juegos en progreso.

- Juegos terminados.
- Juegos abandonados.
- Número total de videojuegos en la lista personal.
- Número total de reseñas.
- Nota media de videojuegos en la lista personal.

Con esta pequeña sección podemos aprovechar aún más el poderío de *Firestore*.

5.4.8 Implementación del cambio de foto de perfil, contraseña y nombre de usuario

Estas tres opciones son fundamentales en cualquier aplicación. En primer lugar, para cambiar la foto de perfil se implementó una pantalla emergente en la que el usuario podría seleccionar su futura foto, tal y como se muestra en la Figura 28::

Figura 28. Ejemplo de cambio de foto de perfil. (Fuente: Elaboración propia).

Una vez seleccionada la foto, la anterior será borrada de *Storage* para evitar problemas de espacio y en su lugar se añadirá la nueva imagen, además de sustituir su URL en el documento del usuario en *Firestore.*

Para cambiar la contraseña se decidió utilizar una ventana emergente, al igual que para el cambio de foto de perfil. Para poder realizar la acción del cambio de contraseña el usuario debe introducir la contraseña actual, introducir una nueva y repetir la contraseña nueva.

Para verificar la validez de la contraseña anterior, se simula un proceso de inicio de sesión en *Firebase*. Si este proceso es exitoso, entonces se procederá a actualizar las credenciales del usuario correspondiente en la sección de *Authentication*.

Por último, el usuario podrá cambiar su propio nombre de usuario al gusto sin ningún tipo de restricción. Una vez iniciado el proceso de cambio de nombre se abrirá una pantalla emergente con un cuadro de texto para escribir el nuevo nombre. Una vez introducido, el anterior nombre de usuario será sustituido por el nuevo en su correspondiente documento de la colección 'usuarios' de *Firestore*.

Tanto si las operaciones tienen éxito como si se produce algún fallo el usuario será notificado en la pantalla por una notificación. Veamos cómo quedaría la página de perfil con los cambios de foto de perfil y nombre de usuario en la Figura 29:

Figura 29. Ejemplo con la foto de perfil y el nombre de usuario cambiados. (Fuente: Elaboración propia).

Podemos destacar que estos cambios de foto de perfil y nombre de usuario se aplican automáticamente sin necesidad de actualizar la pantalla.

5.4.9 Desarrollo de la funcionalidad de eliminación de cuenta

Debido a que borrar una cuenta de usuario siempre es un proceso delicado, si el usuario decide hacerlo antes de nada deberá pasar una doble confirmación a través de una ventana emergente tal y como se muestra en la Figura 30:

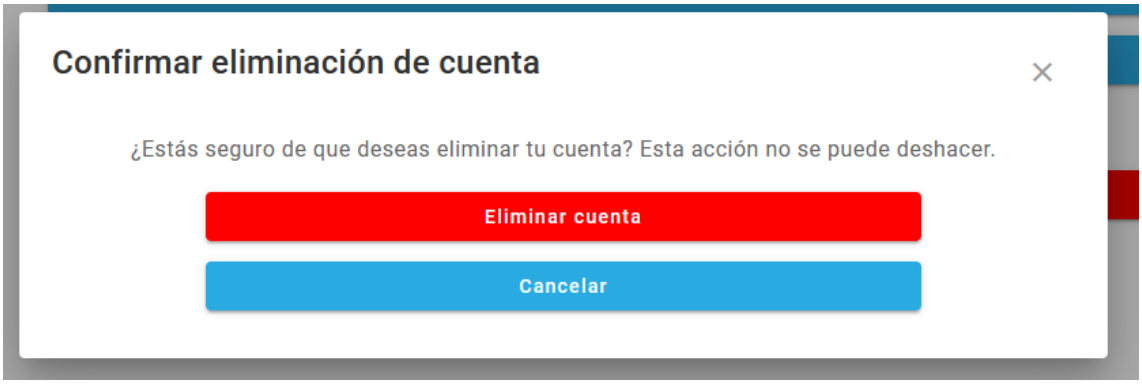

Figura 30. Confirmación eliminación de cuenta. (Fuente: Elaboración propia).

Rompiendo con el diseño general de la aplicación se decidió a propósito utilizar un color rojo en este botón para alertar al usuario de la acción que está a punto de realizar.

Si el usuario desea continuar se eliminará su documento de la colección 'usuarios', su cuenta en *Authentication* y se devolverá al usuario a la pantalla de inicio de sesión. Si intenta entrar con las credenciales de la cuenta eliminada le será imposible hacerlo.

5.4.10 Implementación del cierre de sesión

Por último, se implementó una opción para cerrar la sesión del usuario. Esta opción se mostrará debajo de la opción de ver perfil en forma de menú desplegable, tal y como se puede ver en la Figura 31:

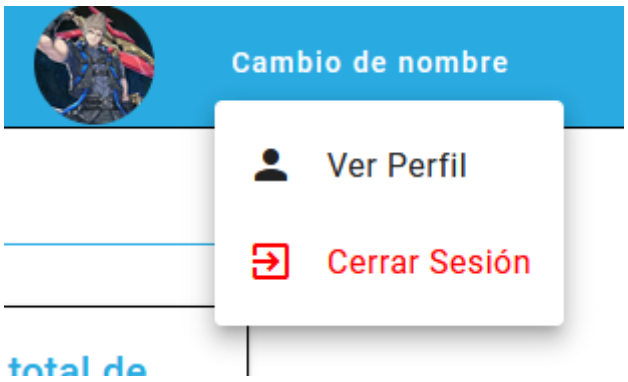

Figura 31. Desplegable mostrando el cierre de sesión. (Fuente: Elaboración propia).

Si el usuario decide cerrar sesión se llamará a una función incluida en la librería *AngularFire* que manejará el cierre de sesión y devolverá al usuario a la pantalla de inicio de sesión. El usuario podrá volver a entrar en su cuenta con sus credenciales.

5.5 *Sprint* 4

Tabla 6. Cuadro que muestra las tareas a completar del cuarto Sprint. Incluyendo un tiempo de desarrollo estimado y los requisitos funcionales y no funcionales relacionados.

TAREAS	ESTIMACIÓN	REQUSITIOS FUNCIONALES	REQUISITOS NO FUNCIONALES
Integración con *GitHub Copilot*	5 días		RNF1, RNF7,
Pruebas y corrección de errores	2 días		RNF8, RNF10
Actualización de las librerías de la aplicación	2 días		RNF1, RNF6, RNF7

5.5.1 Integración con *GitHub Copilot*

GitHub Copilot es un asistente que ayuda al desarrollador en la codificación de aplicaciones. Está basado en inteligencia artificial y fue codesarrollado entre *GitHub*, una plataforma para el alojamiento de código, y *OpenAI*, una empresa de desarrollo e investigación de inteligencia artificial.

Este asistente utiliza el código escrito por el desarrollador hasta el momento para intentar 'adivinar' el siguiente paso del desarrollo. Es capaz de desarrollar código desde cero y es fácilmente instalable en *Visual Studio Code* (Microsoft Corporation, 2024), la plataforma elegida para el desarrollo de esta página web.

Veamos un ejemplo de cómo se ha utilizado *GitHub Copilot* en esta aplicación mediante la Figura 32:

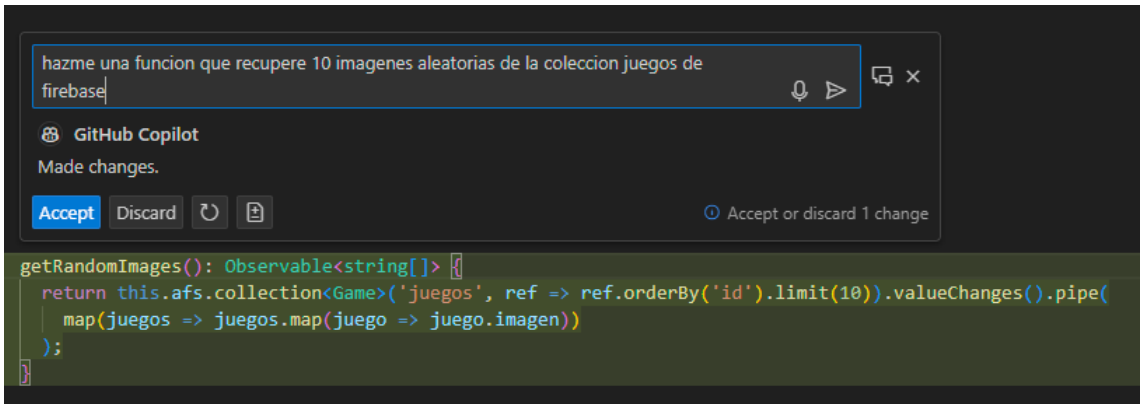

Figura 32. Ejemplo de código realizado automáticamente por GitHub Copilot. (Fuente: Elaboración propia).

Como se puede ver, en esta imagen el desarrollador introduce una *prompt* (es decir, una instrucción que se utiliza para interactuar con la IA) al asistente y este es capaz de generar código gracias a que puede utilizar el código ya escrito anteriormente por el desarrollador. Aun así, cualquier código hecho por una inteligencia artificial debe ser revisado manualmente ya que pueden contener errores.

Esta tecnología es de pago, pero permite una prueba de treinta días gratuita, la cual se ha utilizado para poder probar las capacidades de esta herramienta. Además, también se ha dado uso a otras tecnologías de inteligencia artificial como *Microsoft Copilot* (Microsoft Copilot, 2024) para la mejora del código o resolución de dudas. Esta tecnología, al contrario que *GitHub Copilot*, es gratuita.

5.5.2 Pruebas y corrección de errores

Una vez finalizado el desarrollo de la primera versión de la aplicación web, se probó exhaustivamente el proyecto. Debido a la falta de tiempo no se implementaron pruebas unitarias ni de integración.

Cabe destacar que, aunque se ha revisado al completo la aplicación en busca de posibles errores, es posible que puedan aparecer errores no contemplados en esta batería de pruebas.

5.5.3 Actualización de las librerías de la aplicación

La fase final del desarrollo implicó la actualización de todas las librerías empleadas en el proyecto a su versión más reciente disponible al momento de redactar este documento. Tras concluir este proceso, se llevó a cabo una nueva ronda de pruebas para detectar posibles fallos que pudieran haber surgido como resultado de la actualización.

6. Trabajo futuro

Una vez finalizado el desarrollo de esta primera versión de la aplicación se esgrimió una serie de objetivos a continuación, incluyendo desarrollos dejados en el tintero y pasos a futuro:

- **Implementación de listas de más jugados y carrusel de imágenes:** como se ha mencionado anteriormente, una funcionalidad que lamentablemente no se ha podido implementar correctamente en esta versión son las listas de juegos más jugados y un carrusel de imágenes mostrando distintos videojuegos tal y como se mencionó en la sección de la página de inicio de la aplicación.
- **Creación de pruebas unitarias y de integración:** mientras que se han realizado innumerables pruebas manuales de la aplicación para comprobar que los desarrollos funcionaban correctamente, lo más adecuado en este tipo de proyectos es la creación de pruebas unitarias y de integración.
- **Despliegue de la aplicación:** una vez creadas las pruebas y las funcionalidades mencionadas en el primer apartado, se publicará la aplicación en fase demo para poder probarla con amigos y conocidos y poder recibir sus recomendaciones. Además, *Firebase* cuenta con un módulo de alojamiento web llamado *Hosting*.
- **Nuevas funcionalidades:** a partir de esas recomendaciones recibidas se crearán nuevos *sprints* con nuevos desarrollos tal y como un sistema de amigos, añadir más videojuegos en la base de datos o calificación de reseñas.
- **Actualización y mantenimiento:** esta sección se dedicará a abordar los problemas señalados por los usuarios, así como a la incorporación de nuevas funcionalidades que surjan debido a la demanda o con el paso del tiempo. Además, se llevará a cabo la actualización de las librerías del proyecto cada vez que se publique una nueva versión estable.

Estos pasos representan la hoja de ruta para el futuro de la aplicación. Aunque ya se ha logrado mucho, aún queda un camino emocionante por recorrer. Con la implementación de nuevas funcionalidades, la creación de pruebas exhaustivas, el despliegue de la aplicación y su constante actualización y mantenimiento, se espera mejorar continuamente la experiencia del usuario y adaptarse a sus necesidades cambiantes.

7. Conclusiones

Este proyecto ha representado una oportunidad única para aplicar y consolidar una serie de habilidades y conocimientos adquiridos tanto en el ámbito académico como profesional. Relacionando los sub-objetivos establecidos al principio del desarrollo del proyecto podemos sacar las siguientes conclusiones:

1. **Dominio de *Firebase* y consolidación de habilidades en *Angular.*** A pesar de que *Firebase* no se estudia en el grado, he conseguido desarrollar esta aplicación utilizando esta tecnología en prácticamente todos los aspectos de base de datos de la aplicación gracias a la documentación disponible en su página web oficial. Considero que he conseguido desarrollar habilidades que se ven reflejadas en el manejo de la base de datos NoSQL de la aplicación o distintas acciones que necesitaban utilizar *Firebase*. Ha sido un desafío duro implementar una tecnología totalmente desconocida para mi persona, pero se ha conseguido con éxito.
2. **Finalización de un ciclo de desarrollo completo.** En este proyecto he podido planificar un ciclo de vida de desarrollo de software de forma completa, desde la toma y análisis de requisitos hasta el mantenimiento de la aplicación gracias a los conocimientos adquiridos en el grado. Por desgracia, debido a la falta de tiempo, este objetivo no se ha podido cumplir al completo ya que me he quedado en el desarrollo e implementación de la aplicación. Aun así, el resto de aspectos están presentes en el documento aquí expuesto, desde la toma de requisitos pasando por la planificación, el diseño y el desarrollo. Este objetivo ha sido cumplido 'a medias' debido a mi falta de experiencia en este aspecto.
3. **Implementación de habilidades profesionales en un proyecto personal.** En este proyecto he querido poner en práctica todos mis conocimientos adquiridos en los tres años aproximadamente que llevo trabajando como desarrollador web de *Angular* y *Java*. Debido a que no quería utilizar *Java* opté por utilizar *Firebase* para aprender nuevas tecnologías de cara a mi situación laboral. Podemos dar este objetivo como conseguido ya que el desarrollo de la parte de *Angular* ha sido muy rápido, permitiéndome centrarme en el reto que suponía aprender *Firebase*. Gracias a la experiencia laboral obtenida anteriormente el desarrollo ha sido mucho más sencillo y rápido además de que se han puesto en práctica lo aprendido en el entorno de trabajo, con lo que podemos dar este objetivo por conseguido.

4. **Innovación y desarrollo de funcionalidades.** El último objetivo planteado en este proyecto era el desarrollo de funcionalidades existentes en otras aplicaciones que se tuvieron en cuenta durante la fase de investigación del proyecto. La mayoría tenían una sección dedicada a reseñas de usuarios de los videojuegos y a los videojuegos más populares. Además, yo personalmente quería incluir un carrusel de imágenes para mostrar mejor los videojuegos más populares del momento. Este objetivo no ha podido cumplirse en su totalidad ya que solo he podido incluir la sección de reseñas de usuarios, teniendo que desechar el carrusel de imágenes y tendencias debido a la falta de tiempo.

En resumen, este proyecto ha servido para reforzar los conocimientos previos adquiridos, el descubrimiento de nuevas herramientas con potenciales increíbles y una firme convicción si cabe más todavía de continuar con el desarrollo de aplicaciones web.

8. Bibliografía

Angular Team. (2024). *Angular*. Obtenido de https://angular.io/

AngularFire. (2024). *AngularFire*. Obtenido de https://github.com/angular/angularfire

AngularFire. (2024). *Authentication*. Obtenido de https://github.com/angular/angularfire/blob/master/docs/auth.md#authentication

AngularFire. (2024). *Cloud Firestore*. Obtenido de https://github.com/angular/angularfire/blob/master/docs/firestore.md#cloud-firestore

Asociación Española de Videojuegos (AEVI). (2020). *Asociación Española de Videojuegos*. Obtenido de https://www.aevi.org.es/web/la-industria-del-videojuego/en-espana/

Asociación Española de Videojuegos (AEVI). (2023). *Asociación Española de Videojuegos*. Obtenido de https://www.aevi.org.es/web/sector-del-videojuego-crece-12-2022-una-facturacion-record-mas-2-000-millones-euros/

Atlassian. (2024). *Trello*. Obtenido de https://trello.com/es

BACKLOGGD. (2024). *BACKLOGGD*. Obtenido de https://www.backloggd.com/

Creately. (2024). *Creately*. Obtenido de https://creately.com/

Firebase. (2024). *Cloud Firestore*. Obtenido de https://firebase.google.com/docs/firestore

Firebase. (2024). *Cloud Storage for Firebase*. Obtenido de https://firebase.google.com/docs/storage

Firebase. (2024). *Firebase Authentication*. Obtenido de https://firebase.google.com/docs/auth

GitHub. (2024). *GitHub Copilot*. Obtenido de https://github.com/features/copilot/

Google LLC. (2024). *Firebase*. Obtenido de https://firebase.google.com/

Microsoft Copilot. (2024).

Microsoft Corporation. (2024). *Microsoft Paint*. Obtenido de https://www.microsoft.com/es-es/

Microsoft Corporation. (2024). *TypeScript*. Obtenido de https://www.typescriptlang.org/

Microsoft Corporation. (2024). *Visual Studio Code*. Obtenido de https://code.visualstudio.com/

Orellana, J. (2017). *LinkedIn*. Obtenido de https://es.linkedin.com/pulse/una-breve-introducci%C3%B3n-scrum-javier-orellana

Peinado, D. A. (2024). *GitHub*. Obtenido de GameShare_front: https://github.com/FRoZeNDanieh/GameShare_front

STASH. (2024). *STASH*. Obtenido de https://stash.games/

TV TIME. (2024). *TV TIME*. Obtenido de https://www.tvtime.com/es