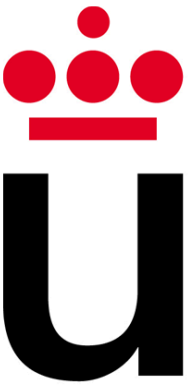

Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE COMPUTADORES

Curso Académico 2023/2024

Trabajo Fin de Grado

GameTags: Una plataforma para clasificaciones de videojuegos

Autor: Gonzalo Lobo González

Tutor: Daniel Palacios Alonso

Resumen

En la era digital actual, tenemos a nuestro alcance un universo de información sobre casi cualquier tema. Sin embargo, el acceso a estos datos no siempre es sencillo. Un ejemplo de ello es el mundo de los videojuegos. Un dato esquivo es el consenso sobre la clasificación por edades de los nuevos videojuegos en el mercado. Aunque las clasificaciones oficiales por edades están disponibles en sus respectivos sitios web, obtener una visión completa del rango de edad recomendado puede ser una tarea ardua, que requiere una extensa navegación por Internet. Y, aun así, esta información puede resultar insuficiente, ya que no toma en cuenta un factor igualmente crucial: la opinión del público. La perspectiva de aquellos que han jugado al videojuego y han experimentado de primera mano su contenido es invaluable. Lamentablemente, esta información no se refleja en toda su extensión en los informes de las entidades oficiales. Por lo tanto, aunque vivimos en un mundo lleno de información, no siempre es fácil acceder a los datos que realmente importan.

Ante esta problemática, surge una pregunta: ¿existe algún lugar donde se reúnan tanto las clasificaciones oficiales otorgadas a los videojuegos como la opinión del público que los ha jugado? Algunas opciones similares a esta idea ya existen, pero ninguna que la abarque al completo. Por eso, para este trabajo, se ha creado una plataforma que aborde directamente a esta situación planteada. Se llama GameTags.

Palabras Clave: Back-End, Front-End, Client, Servidor, Página Web, Java, Base de Datos, Opinión, Temáticas, Comentarios

Abstract

In today's world of information, access to an infinity of data on practically anything is possible. But just because this data is accessible doesn't mean it's easily accessible. In the world of video games, one such less accessible piece of data is the general consensus of the age rating assigned to video games released to the market. Although most official age ratings can be accessed through their portals, obtaining a broader picture of the recommended minimum age range is not as straightforward, requiring extensive internet navigation to eventually acquire that information. But even then, it's insufficient as it overlooks an equally important piece of data: public opinion. The opinion of those who have played the video game and experienced its contents. Information not provided by official entities in its entirety.

Faced with this problem, a question arises: is there any place where both the official ratings assigned to video games and the opinions of the public who have played them are gathered? Some options similar to this idea already exist, but none that encompass it entirely. That's why, for this project, a platform has been created to directly address this situation. It's called GameTags.

Keywords: Back-End, Front-End, Client, Server, Web Page, Java, Data Base, Judgement, Themes, Comments

Índice de contenido

Índice de figuras

Glosario

Back-End	El encargado de procesar la información aportada desde el *front-end* para realizar operaciones. A diferencia del *front-end*, el *back-end* no es visible para el usuario
Front-End	Interfaz de un portal web con el que interactúa el usuario. A través de él el usuario solicita la realización de operaciones, las cuales son gestionadas por el *back-end*
Token	Representación digital de un activo concreto, en este caso de las credenciales de un usuario
Mapper	Clase encargada de realizar la operación de conversión entre un objeto de datos y otro
API	Conjunto de definiciones y protocolos que se usan para desarrollar e integrar la comunicación entre dos aplicaciones
LOGGER	Herramienta de creación de trazas de consola para el seguimiento del flujo de una aplicación
Framework	Conjunto de herramientas, guías y prácticas predefinidas usadas como estructura base para el desarrollo de aplicaciones
JavaScript	Uno de los lenguajes de programación usados en páginas web
HTML	Lenguaje de Marcas estándar usado para la definición del significado y estructura de una página web y su contenido
REST	Estilo de arquitectura de comunicación entre sistemas distribuidos
CSS	Lenguaje usado para dar estilo a un documento HTML
DTO	Patrón de transmisión entre procesos de objetos que contienen datos para reducir el número de llamadas
DAO	Patrón de comunicación entre una aplicación y una base de datos que busca simplificar la obtención de datos de esta última
Dropdown	Elemento contextual de la interfaz de una web, usado para mostrar listas de elementos
Debuggear	Proceso por el cual el programador examina el funcionamiento del programa en vivo en busca de errores
Layout	Forma en la que se disponen los elementos de una página
Endpoint	Ruta en la que escucha un servidor para atender un tipo concreto de peticiones

Capítulo 1

Introducción

En esta sección se aporta información como orígenes, labores y funciones; sobre las principales plataformas y entidades que se centran en informar sobre los contenidos de los videojuegos, además de presentar la forma de compartir la información. Asimismo, también se informa sobre el estado actual de dichas entidades.

1.1 Apartado de historia

Este proyecto de fin de carrera es la secuela de un estudio previo titulado "Análisis y Comparativa de los Sistemas de Clasificación por Edades de Videojuegos a Nivel Mundial". En dicho estudio, se observó que la información relativa a las diversas clasificaciones asignadas a un mismo videojuego estaba dispersa en diferentes sitios web, a menudo sin detalles suficientes sobre el contenido del juego o sin especificaciones claras sobre qué esperar al jugarlo. Esta situación llevó a la conclusión de que sería ventajoso disponer de un sitio web o plataforma que centralizara toda esta información.

Si uno busca por Internet alguna página que se asemeje mínimamente a esta idea, encontrará varias que se encarguen, aunque solo en algunos tipos de productos, de esta labor, prestando especial atención a los contenidos específicos que se puede encontrar dentro de ellos, como IMDb (*Internet Movie Database*) o Common Sense. También se pueden encontrar páginas las cuales sí que actúan como bases de datos exclusivas de videojuegos donde uno de los apartados de información que almacenan son las clasificaciones por edad, aunque no entran en mucha mayor profundidad sobre los contenidos dentro de estos videojuegos, como es el caso con IGDB [27] o MobyGames [28].

La historia de **IMDb** comienza a principios de los 90, más concretamente el 17 de octubre de 1990 [7], cuando su creador, Col Needham, monta la página para poder recopilar todas las películas que había visto durante la década de los 80. Esta página la acabó compartiendo en un foro de USENET [7], previo a la creación de la WWW (*World Wide Web*), donde tuvo repercusión, ya que más usuarios hicieron lo mismo que Col y subieron las películas que ellos habían visto, creciendo considerablemente hasta que finalmente se incorporó a la WWW en enero de 1996. Un par de años más tarde, en 1998, fue comprada por Amazon [7]. La intención de Amazon era convertir IMDb en un portal online de venta de películas, aunque los esfuerzos puestos en hacer esta conversión no culminaron y desistieron de la idea, pero

sin cerrar el sitio web. Dejaron que Col siguiese dirigiendo IMDb con un cierto nivel de independencia [7] y que siguiese centrándose en lo que ya hacía y que siguiese creciendo.

En un inicio la información que se subía por cada una de las películas y series no era información muy detallada, solo la que podían obtener leyendo los créditos al final de éstas. Pero a medida que fue pasando el tiempo se expandieron, documentando también otros datos como las fechas de lanzamiento, críticas recibidas, beneficios generados, clasificaciones por edades, etc; incluso con esta expansión empezaron a documentar otros productos como los videojuegos o la música.

CommonSense Media se fundó en 2003 como una entidad independiente [2] cuyo objetivo era recomendar material adecuado para familias como películas y series de televisión, de modo que los hijos pudiesen consumir material adecuado para su edad durante la expansión de Internet. Más tarde, en 2008, extendieron su alcance, involucrándose con las escuelas para así instruir a los profesores en materia digital y que estos, a su vez, enseñaran a los estudiantes cómo hacer un buen uso de la tecnología [1].

En 2011 comenzaron un proyecto de investigación sobre la influencia que puede tener los medios en los jóvenes [1] con el objetivo de mejorar su forma de guiar y de adaptarse a los cambios sociales del momento de ahí en adelante. En 2012 ampliaron su catálogo de guía, empezando a hacer recomendaciones sobre videojuegos, páginas web y aplicaciones de dispositivos móviles [1]. Con esto buscaban cumplir sus labores principales en más medios y guiar sobre las herramientas educacionales adecuadas para niños y familias.

1.2 Apartado de mercado actual

Dentro de los muchos apartados de los que consta una entrada en la base de datos de IMDb, la que interesa para este proyecto es el apartado “Guía Para Padres”. Dentro de esta sección se detalla información como las clasificaciones por edad que ha recibido un videojuego, película o serie de televisión en todo el mundo, además de cinco subapartados (*Fig.1*) de contenidos temáticos. Estos apartados temáticos son:

- Sexo y Desnudez
- Violencia
- Lenguaje Soez
- Alcohol, Drogas y Tabaco
- Escenas Intensas o Aterradoras

Fig. 1 Página de información de un videojuego en IMDb

Por cada uno de estos subapartados se especifica el nivel general de severidad de dicho tipo de contenido presente en la obra, indicando la cantidad de opiniones que han compartido los usuarios sobre dicha temática y la cantidad de opiniones que han catalogado dicha

temática con el nivel de severidad elegido finalmente para la categoría (*Fig.2*), además de los comentarios de aquellos usuarios que, además de indicar el nivel de severidad que ellos creen, también han escrito un texto sobre los contenidos presentes de esa categoría.

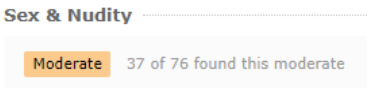

Fig. 2 Nivel de severidad de temática en IMDb

Se usan cuatro niveles de severidad para categorizar cada uno de los apartados, siendo estos niveles:

- Ninguno
- Leve
- Moderado
- Severo

Como se ha señalado anteriormente, la empresa no es la que determina la gravedad de cada tema. En cambio, son los usuarios quienes, al compartir sus puntos de vista en los comentarios, influyen en esta decisión. Posteriormente, la página realiza un análisis de las opiniones compartidas y establece el nivel de gravedad general basándose en el grado de severidad que se haya mencionado con mayor frecuencia en los comentarios.

También los usuarios tienen la opción de reportar el contenido de un comentario ajeno, como medida de control comunitaria, o incluso pueden especificar que el contenido de los textos de sus comentarios contiene espóileres (*Fig.3*), separando esos comentarios en un subapartado dentro de la categoría de temática.

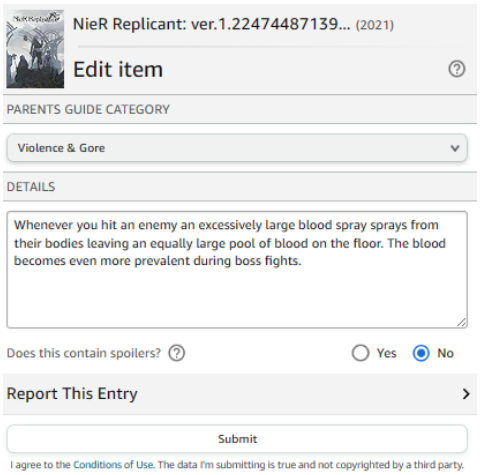

Fig. 3 Página de reporte de un comentario en IMDb

Dentro de la plataforma web de CommonSense Media existe una página por cada videojuego, película o serie de televisión que tengan registrada. Dentro de esta página el producto cuenta con un resumen dirigido a los padres, una valoración dada por CommonSense y evaluada en un rango de cero a cinco estrellas para indicar su nivel de calidad, junto con la edad recomendada por ellos para consumir dicho producto y una muy breve idea del contenido de este (*Fig.4*).

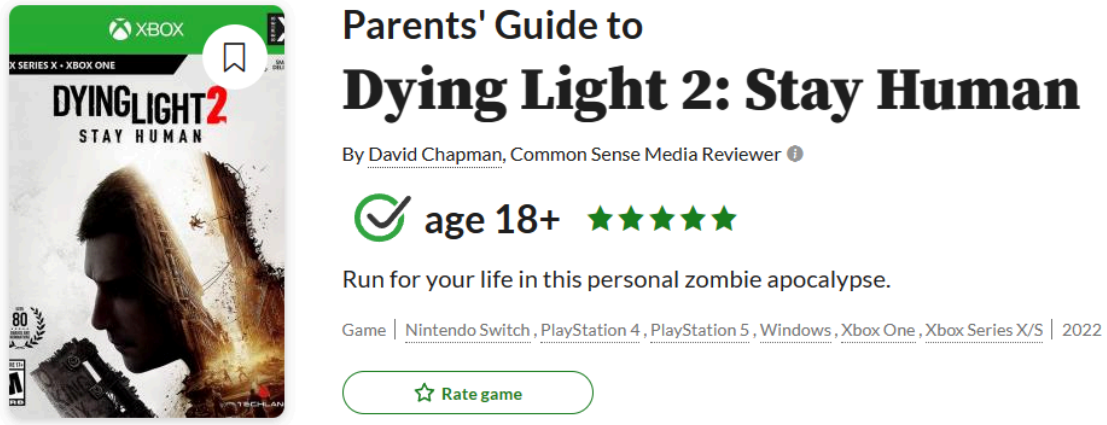

Fig. 4 Cabecera de una entrada en CommonSense Media

Después cuentan con valoraciones de severidad repartidas en ocho categorías (*Fig.5*):

- Mensajes Positivos
- Modelos de Comportamiento Positivos
- Diversidad de Representación
- Violencia y Horror
- Sexo, Romance y Desnudez
- Lenguaje Soez
- Publicidad
- Alcohol, Drogas y Tabaco

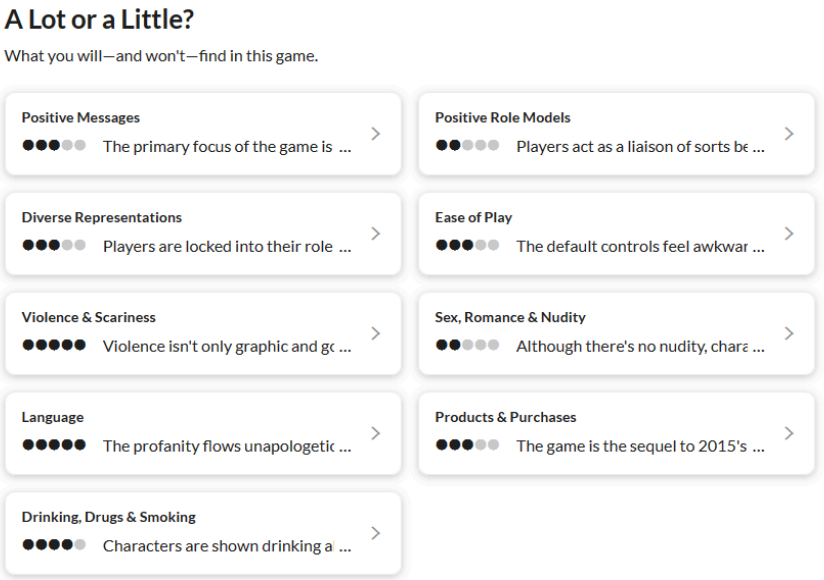

Fig. 5 Apartados de contenidos de una entrada en CommonSense Media

Dentro de cada uno de estos apartados ellos valoran en un rango de cero a cinco el nivel de severidad de dicha categoría, además de contar con un resumen del por qué cuenta con esa puntuación (*Fig.6*).

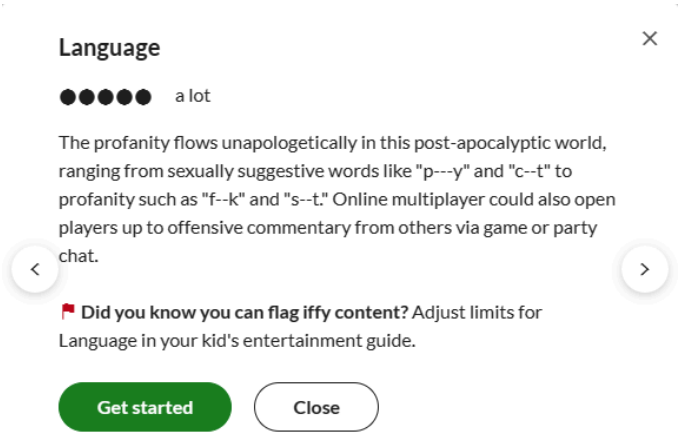

Fig. 6 Descripción del contenido de una temática

Además de esto, existe una sección de análisis de la comunidad (*Fig.7*) donde tanto los padres como los hijos dan su perspectiva sobre el producto mediante un texto, una valoración de cero a cinco estrellas de la calidad del producto y la edad mínima apta que ellos consideran para que el producto sea consumido. Las opiniones de cada grupo se encuentran separadas en dos subapartados, obteniendo así la edad mínima recomendada por cada grupo basándose en los comentarios hechos por los miembros de cada uno.

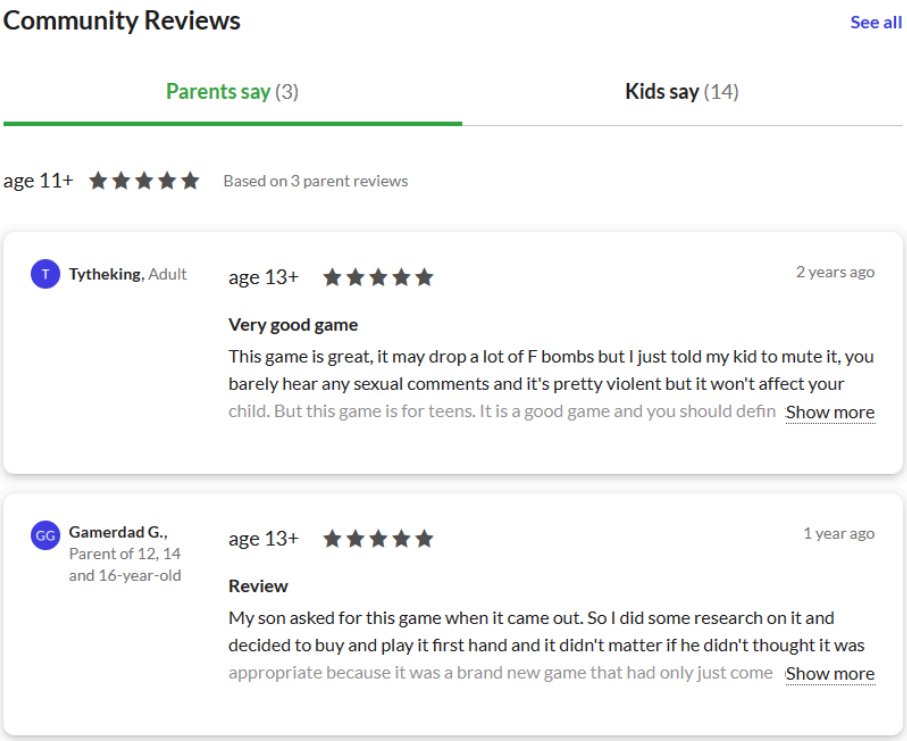

Fig. 7 Sección de comentarios de la comunidad en una entrada de CommonSense Media

1.3 Apartado de estado del arte

CommonSense Media, en su continua labor de proteger a los menores y proporcionarles una plataforma para expresarse, sigue vigente. Para el 2024, se han propuesto influir más en la política con el fin de promover leyes que defiendan los derechos y la privacidad de los menores en Internet [4]. Además, están abordando la creciente expansión de las Inteligencias Artificiales [5], estableciendo acuerdos con organizaciones internacionales como la británica NSPCC (National Society for the Prevention of Cruelty to Children) en enero de 2024 [6]. Su objetivo es negociar con las empresas tecnológicas para implementar medidas de seguridad que protejan a los menores de los posibles riesgos de estas tecnologías, al tiempo que se exploran formas en las que los menores puedan beneficiarse de su uso.

Junto con lo previamente mencionado, siguen con el desarrollo de investigaciones para acercar al público general las opiniones y las voces de los menores, como por ejemplo una investigación finalizada en enero de 2024 [3] donde tanto los votantes como los adolescentes estadounidenses compartían sus preocupaciones y perspectivas sobre el desarrollo de políticas pensadas en la protección del menor.

En el caso de IMDb no ha pasado por muchos cambios en los últimos años. Sigue actuando como una plataforma que actúa como base de datos de información sobre películas y series, y en menor medida videojuegos y música. En los últimos años ha puesto empeño en crear su propia plataforma de videos bajo demanda, lanzando Freedive en enero de 2019 [8] y recibiendo varios *rebrandings*, uno en junio del mismo año pasándose a llamar IMDb TV y otro en abril de 2022 cambiando a Amazon Freevee, siendo este último para enlazarse más con su empresa paterna.

Capítulo 2

Objetivos

Esta sección expone el problema que el presente documento aspira a abordar y resolver. Se presenta una serie de objetivos que se pretenden alcanzar para solucionar dicho problema, así como alternativas a la solución seleccionada. Además, se incluye una sección que detalla la metodología de trabajo empleada y la planificación que ha guiado el desarrollo integral del trabajo.

2.1 Descripción del problema

Como se ha observado en las secciones previas, existen sitios web que proporcionan información detallada sobre videojuegos y otros donde los usuarios pueden expresar sus opiniones sobre el contenido de un videojuego. Sin embargo, no hay una plataforma que combine estos dos aspectos y se centre exclusivamente en videojuegos.

En respuesta a esta necesidad, se ha creado una plataforma donde los usuarios pueden documentar los diversos videojuegos disponibles y añadir las calificaciones oficiales otorgadas por varias entidades de clasificación a nivel global. Además, los usuarios pueden compartir sus opiniones sobre los diferentes aspectos del juego. Esto permite a otros usuarios tomar una decisión más informada sobre si el juego es apropiado para la edad de sus hijos.

2.2 Objetivos

Los objetivos que busca solventar esta solución son:

- Otorgar a los padres, madres y tutores legales de menores un lugar que centralice todas las clasificaciones por edades oficiales dadas a los videojuegos.
- Permitir que los usuarios añadan información importante en cuanto a los contenidos de los videojuegos que no es comunicada por las entidades oficiales.
- Definir el grado de impacto de los contenidos de un videojuego de forma comunitaria.

2.3 Estudio de alternativas

A parte de la solución planteada en el apartado anterior, se podrían optar por otras soluciones.

Debido a la existencia de la **IARC** (*International Age Rating Coalition*) y el rol de ésta como entidad que une a los principales sistemas de clasificación que existen alrededor del mundo [29], la IARC podría crear una plataforma en colaboración con esas entidades para recopilar todas las clasificaciones otorgadas a los distintos videojuegos que salen al mercado y hacerlo de forma oficial, promocionando además la existencia de dicha plataforma para que el público fuese consciente de que existe un lugar donde poder consultar de forma centralizada la información sobre los contenidos de un videojuego.

Otra posible solución sería que **IMDb** y **CommonSense Media** aumentasen la relevancia de la calificación de videojuegos dentro de sus plataformas. En el caso de IMDb, que moviesen a su comunidad para que se involucrase más con la calificación de videojuegos ya que, al lado de la información aportada por ella en películas o series de televisión, la información que se puede encontrar dentro de sus páginas de videojuegos es bastante escasa. En el caso de CommonSense Media, que diesen la opción en el apartado de comentarios de padres e hijos que estos pudiesen evaluar el nivel de intensidad de los contenidos del videojuego ya que actualmente solo pueden decir la edad mínima que ellos recomiendan y un texto sobre su opinión.

2.4 Metodología empleada

El desarrollo de todo el trabajo se ha dividido en tres etapas.

La primera etapa consistió en el desarrollo del *back-end* de la plataforma (*Fig.8*). En esta etapa se definieron las entidades con las que iba a contar la plataforma, los datos de cada una y la forma en la que se relacionaban unas con otras. A parte también se definieron la cantidad de páginas que iban a formar la plataforma, las acciones que estarían a disposición del usuario y la interconexión que existirían entre unas y otras.

También se planeó y eligió la arquitectura con la que se iba a organizar el *back-end* del proyecto. Tras esto se procedió a comenzar su desarrollo, junto con tests unitarios para asegurar la robustez de las implementaciones realizadas y tests de integración para la prueba del correcto funcionamiento del *back-end* ante las futuras peticiones provenientes del *front-end*.

La segunda etapa consistió en el desarrollo del *front-end* de la plataforma (*Fig.8*). Se hizo una pequeña investigación previa sobre las distintas tecnologías disponibles para ver cuales se adecuaban más a las ideas desarrolladas e implementadas durante la etapa previa. Esto se hizo debido a la poca experiencia previa en el desarrollo de *front-ends*.

Una vez realizada la investigación, se procedió a empezar con el desarrollo técnico del *front-end*, enlazando ambas secciones y probando el funcionamiento de la integración de ambas. Una vez terminado este desarrollo y ya contando con un *front-end* funcional, se continuó con el desarrollo de su apartado estético, además de implementar funcionalidades aconsejadas durante las sesiones de refinamiento realizadas con el tutor para así conseguir un *front-end* más completo.

Tras haber finalizado con el desarrollo de ambos elementos, se pasó a comenzar la etapa de escritura (*Fig.8*). Esta etapa consistió en la elaboración del presente documento donde se desgranaría el trabajo realizado en las etapas anteriores y las opciones de continuación del proyecto en el futuro.

2.5 Planificación

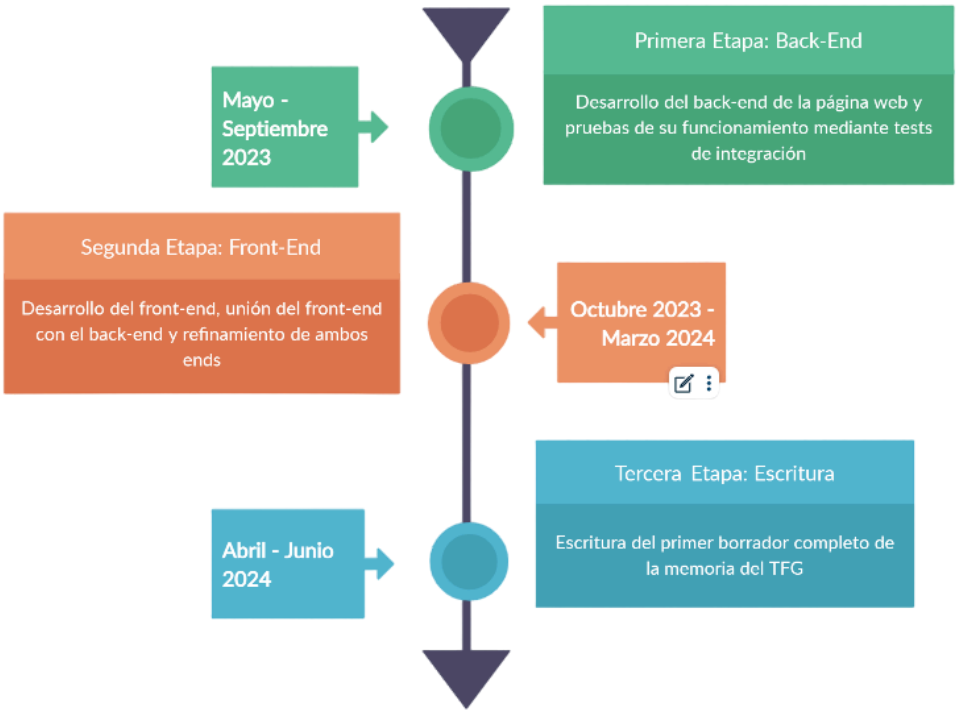

Fig. 8 Cronología del trabajo

Capítulo 3

Especificación

En esta sección del documento se definen los requisitos funcionales que busca cumplir la implementación realizada, desde la definición de los pilares de la plataforma hasta los elementos de los que se componen y cómo funcionará la plataforma desde la vista de un usuario.

El proyecto de GameTags se encuentra dividido en tres secciones: un cliente que hace de Front-End para las peticiones que realiza el usuario. Este se comunica con el Back-End, un servidor, procesando las peticiones y realizando las operaciones pertinentes, consultando a una base de datos donde se persisten los datos de negocio que necesita el servidor para realizar las operaciones que ejecuta y cuyos resultados solicita el usuario por medio del cliente (*Fig.9*).

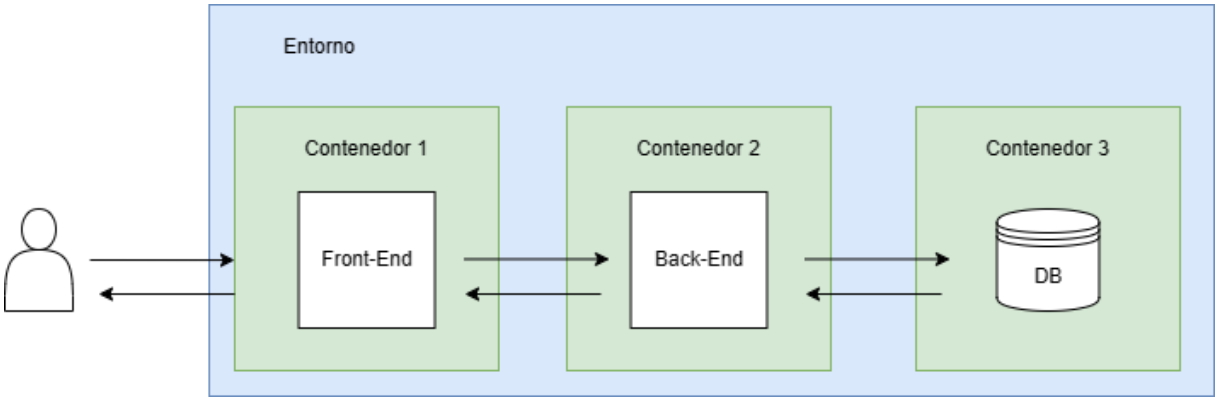

Fig. 9 Organización de GameTags

Cada uno de estos componentes de GameTags se encuentra alojado dentro de un contenedor, comunicándose con el resto de los contenedores por medio de una red compartida.

3.1 Especificación del Back-End

La lógica de negocio de la plataforma se compone de cuatro entidades: el Videojuego que es subido a la plataforma, el Usuario que sube dicho Videojuego, la Clasificación del Videojuego elegida por el Usuario y el Comentario sobre el Videojuego escrito por el Usuario *(Fig.10)*.

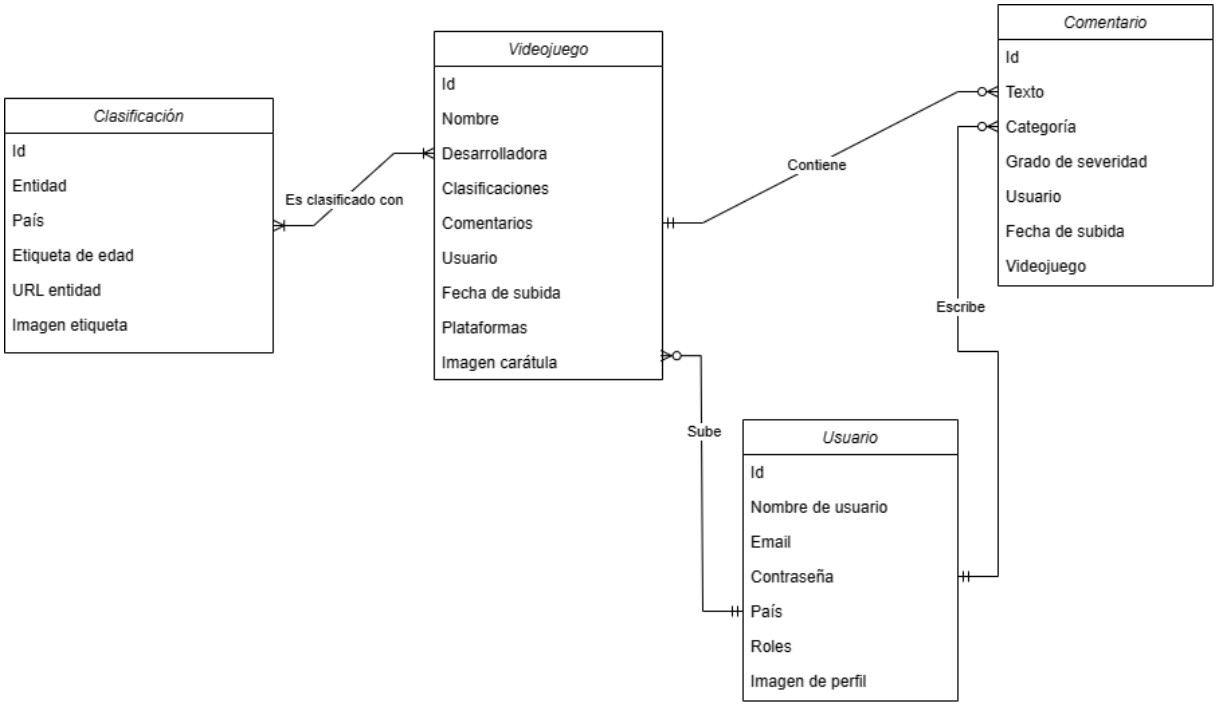

Fig. 10 Diagrama Entidad Relación de las entidades de la plataforma

El Usuario puede no subir ningún Videojuego o subir N Videojuegos a la plataforma, pudiendo ser solo un Usuario el responsable de la subida de dicho Videojuego. El Videojuego cuenta como mínimo con una Clasificación, creada durante la subida del primero o asignada a este en el caso de que ya existiese previamente dicha Clasificación, haciendo que una Clasificación se encuentre asignada a un Videojuego como mínimo y que al mismo tiempo pueda estar asignada a N Videojuegos. Por otro lado, el Usuario puede no crear ningún Comentario o crear N Comentarios, mientras que el Comentario solo puede estar relacionado con un Usuario y con un Videojuego. Sin embargo, un Videojuego puede no contener ningún Comentario o contener N Comentarios.

En el caso de que aumente la cantidad de acciones que el usuario puede realizar desde el cliente y, por ende, aumenten las operaciones y transacciones a tramitar, se requiere que el *back-end* se organice de forma modular, permitiendo la rápida integración de nuevas funcionalidades. También se requiere que la lógica de negocio no se vuelva dependiente de

las tecnologías externas usadas por el servidor, permitiendo en un futuro cambiar las tecnologías con el mínimo impacto.

Ya que, día a día, la cantidad de videojuegos aumenta considerablemente, hay muchos videojuegos, actuales y pasados, que los usuarios pueden ir añadiendo a la plataforma, más aún si se amplía el número de plataformas soportadas por la plataforma. Por esto, se requiere que la forma de persistencia de la base de datos esté especializada en la escalabilidad. Además de este requisito, el tipo de información que se persista de los videojuegos puede aumentar conforme se expanda la plataforma, requiriendo informar, por ejemplo, de la empresa dedicada a publicarlo, la fecha o fechas de lanzamiento del videojuego, entre otros datos. O también puede que se le requiera a los usuarios más información para darse de alta en la plataforma. En estos casos, la base de datos también debe ser flexible para permitir la incorporación de nuevos campos en sus entidades, penalizando lo mínimo posible a las entradas ya existentes dentro de la base de datos.

3.2 Especificación del Front-End

El *front-end* se compone de 12 páginas web, las cuales son las siguientes (*Fig.11*):

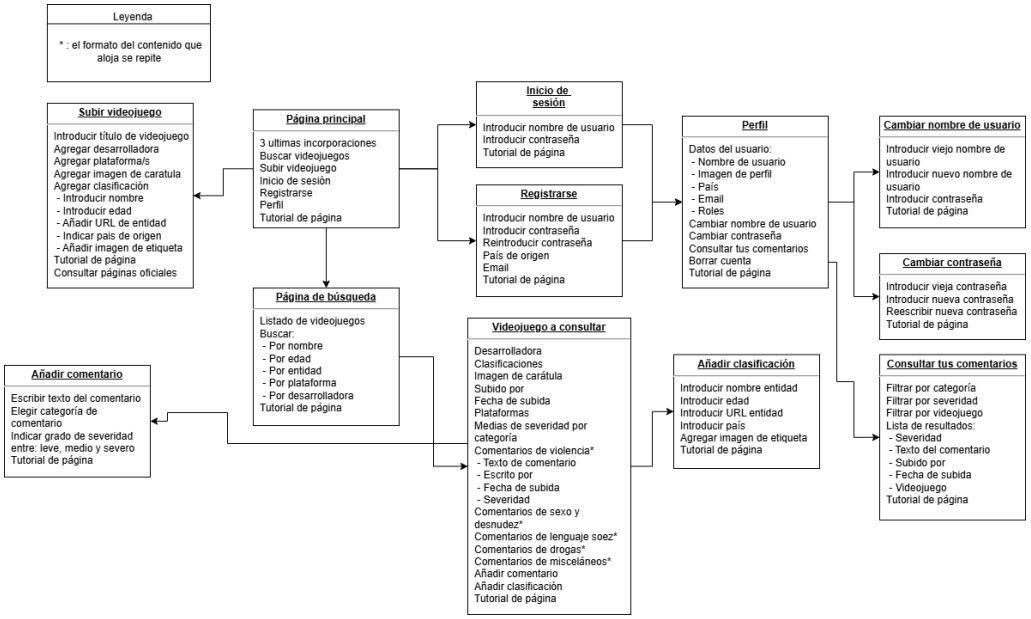

Fig. 11 Diagrama de las páginas que conforman el Front-End

Se pueden agrupar las páginas en dos grupos: un grupo de páginas de acceso general, a las que puede acceder cualquier usuario de la página, haya iniciado sesión o no y un segundo grupo de páginas a las que solo se puede acceder si el usuario ha iniciado sesión.

Dentro del primer grupo se encuentran las siguientes páginas:

- **Página principal**: es la página de acceso a la plataforma. En ella se muestran los tres últimos videojuegos añadidos a la base de datos, usando la imagen de sus carátulas para ilustrarlos.
- **Inicio de sesión**: página en la que un usuario que ya se ha dado de alta en la plataforma puede identificarse y acceder a las páginas y opciones exclusivas de los usuarios registrados.
- **Registrar nuevo usuario**: página en la que se puede dar de alta un nuevo usuario.
- **Página de búsqueda**: página en la que, usando múltiples filtros, se busca uno o varios videojuegos para consultar la información sobre él o ellos.
- **Videojuego para consultar**: página en la que se muestra toda la información perteneciente al videojuego como sus etiquetas, comentarios, valoraciones, etc.

En el segundo grupo se encuentran:

- **Perfil**: aquí se muestra la información del usuario como los roles que desempeña o su nombre de usuario, además de poder acceder a otras opciones relacionadas con su perfil.
- **Agregar nuevo videojuego**: página en la que el usuario especifica los datos de un videojuego que quiere añadir a la plataforma que no estaba previamente en ella.
- **Añadir nuevo comentario**: página en la que un usuario puede escribir un comentario sobre un videojuego, expresando su opinión sobre el contenido de una temática específica del juego o indicando los elementos para tener en cuenta sobre dicha temática.
- **Añadir nueva clasificación**: página en la que un usuario puede añadir una clasificación nueva a un videojuego ya existente.
- **Cambiar nombre de usuario**: página en la que el usuario procede a cambiar su nombre de usuario.
- **Cambiar contraseña**: página en la que el usuario puede cambiar la contraseña que usa actualmente por otra nueva.
- **Consultar tus comentarios**: aquí el usuario puede buscar los comentarios que ha realizado usando varios filtros.

Prácticamente todas las páginas cuentan con una barra de navegación en la parte superior de esta, desde donde se accede a varias de las otras páginas de la plataforma. Si el usuario ha iniciado sesión, la barra de navegación mostrará más opciones que para aquellos usuarios que no lo hayan iniciado. En todas las páginas aparece una o varias ventanas modales que introducen al usuario al funcionamiento de estas, explicando las opciones y acciones disponibles en la página o los campos que debe rellenar si quiere realizar una de esas acciones.

Desde la barra de navegación se puede acceder a la página principal, al inicio de sesión, al registro de un nuevo usuario y la búsqueda de videojuegos en el caso de que el usuario no haya iniciado sesión. Si el usuario ya ha iniciado sesión, a parte de las opciones previas

también se podrá acceder a la subida de un nuevo videojuego, al perfil del usuario y se le dará la opción de cerrar la sesión.

Para abrir la plataforma a una comunidad de usuarios más internacional, se pide que el idioma usado en la plataforma web sea el inglés.

Capítulo 4

Marco teórico

En este apartado se detallan las tecnologías usadas durante el desarrollo de GameTags tanto en el cliente como en el servidor, además del propósito que tienen varias de ellas.

4.1 Tecnologías del Back-End

El **Back-End** consiste en un servidor Tomcat [11] construido mediante Spring Boot [9] y que arranca dentro de un contenedor Docker [10]. La base de datos que usa el *back-end* es una base de datos no relacional MongoDB [13] que se arranca en un contenedor Docker distinto a la del servidor. Se ha acabado eligiendo MongoDB como modelo de base de datos debido al formato de esquema flexible del que se caracteriza, pudiendo añadir nuevos campos a las entidades en el caso de ampliarlas en el futuro, además del extenso uso que se le da dentro del mercado. Para gestionar la seguridad del servidor y la autenticación de los usuarios cuando se registran desde el *front-end* se está usando la librería específica de seguridad de Spring, Spring Security, la cual ya viene integrada junto con Spring Boot cuando se crea el proyecto. En el caso de la identificación de los usuarios cuando estos pretenden acceder a opciones reservadas para usuarios registrados, se usan *tokens* generados mediante JWT [18] y validados con Spring Security. El motivo por el que se ha acabado eligiendo JWT es para evitar el almacenamiento de los *tokens* en base de datos, haciendo de los *tokens* elementos independientes de sesiones que pueden funcionar por sí mismos.

Para poder probar las funcionalidades implementadas y la lógica de negocio se han creado múltiples pruebas unitarias con JUnit [14] y pruebas de integración a través de Postman [15]. Las pruebas de JUnit prueban la lógica de los casos de uso y los elementos que estos usan como *mappers* o servicios sin llegar a involucrar las clases de la capa de integración como controladores o repositorios, mientras que las pruebas de integración realizadas en Postman se organizan en escenarios de caso de uso, como por ejemplo un usuario añadiendo un videojuego y añadiéndole un comentario. En estas pruebas se evalúan, además de lo que prueban las pruebas unitarias, todos los elementos de integración del *back-end* como la seguridad, la creación de *tokens* para identificación o la correcta funcionalidad de la base de datos y la relación de las distintas entidades. Junto con JUnit, para la creación de *mocks* de los componentes que no estaban siendo probados, como adaptadores o puertos, se ha

usado Mockito [26]. El motivo por el que han sido elegidos JUnit y Postman para las pruebas en vez de otras tecnologías como Cucumber [16] o AssertJ [17] ha sido por ya contar con experiencia con ellos en otros proyectos.

Las peticiones al *back-end* se hacen mediante peticiones REST que conforman una API. Dicha API se documenta mediante Swagger [19], el cual crea una página en la que se pueden consultar todas las llamadas cuando el servidor del *back-end* arranca.

Para facilitar y agilizar la construcción de objetos de los distintos modelos de datos, se usa Lombok [12], el cual permite mediante anotaciones evitar escribir manualmente constructores, *getters* y *setters*, solo necesitando especificar los campos de los que constan los objetos. Para mostrar trazas de información en consola se está usando el LOGGER que viene integrado dentro de Lombok.

4.2 Tecnologías del Front-End

El **Front-End** se ha desarrollado usando como *framework* Next.js [21], el cual es un *framework* derivado de React [20] donde cada página es un archivo JavaScript donde se construye el HTML usado por la página como esqueleto y donde se implementan las funciones de JavaScript de las que hace uso para completar dicho esqueleto con datos. Para que el *front-end* realizase las peticiones REST al *back-end* se ha usado Axios [22], el cual crea un cliente y solicita y recibe información del *back-end*. El *front-end* también es arrancado dentro de un contenedor Docker dentro de la misma red en la que está el *back-end* y la base de datos.

Para desarrollar el apartado visual de las páginas, se ha usado Pico CSS [24], el cual es un *framework* de CSS con el que, mediante etiquetas de HTML, se puede definir el aspecto de los elementos de las páginas. Junto con estas etiquetas de Pico también se ha usado CSS para definir los colores de los distintos elementos de las páginas y para cambiar el diseño de estas cuando el tamaño de la pantalla cambiaba, siguiendo un diseño *responsive*.

Otra herramienta que se ha usado ha sido Intro.js [23]. Intro.js se ha usado para crear ventanas modales en cada página que actuasen como tutorial para los nuevos usuarios y les guiasen sobre las distintas opciones presentes dentro de ésta, enseñándoles así sobre su funcionamiento.

Tanto para crear y arrancar el proyecto de Next.js como para instalar los módulos de Axios, Pico CSS e Intro.js se ha usado el instalador de paquetes NPM [25].

Capítulo 5

Descripción informática

En esta sección se describe el diseño del proyecto, las entidades que lo conforman y los campos de los que se componen, además de describir las páginas web de la plataforma, su diseño y las acciones que permiten realizar al usuario. Además, se explica la organización de los archivos del proyecto, el código implementado en ellos y el propósito de cada uno de estos archivos.

5.1 Diseño

5.1.1 Diseño del Back-End

Según lo definido en el capítulo de especificación, se han creado para el modelo de negocio cuatro entidades: clasificación, videojuego, usuario y comentario.

Los distintos campos de los que se conforma cada entidad son los siguientes:

- **Clasificación**:
  - **ID**: identificador único de la clasificación.
  - **Entidad**: abreviatura del nombre completo de la entidad a la que pertenece la clasificación, por ejemplo, PEGI (*Pan European Game Information*) o BBFC (*British Board of Film Classification*)
  - **País**: país o región de procedencia de la entidad a la que pertenece la clasificación.
  - **Etiqueta de edad**: representación de la edad indicada en la etiqueta, por ejemplo, 18 para la etiqueta de +18 del PEGI o T para la etiqueta *Teen* del ESRB (*Entertainment Software Rating Board*)
  - **URL de la entidad**: URL de la página principal de la entidad a la que pertenece la clasificación.
  - **Imagen de etiqueta**: archivo de imagen con el que se ilustra la clasificación.
- **Videojuego**:
  - **ID**: identificador único del videojuego.
  - **Nombre**: nombre del videojuego
  - **Desarrolladora**: nombre de la desarrolladora del videojuego.

    - **Clasificaciones**: listado de clasificaciones relacionadas con el videojuego.
    - **Comentarios**: listado de comentarios hechos por los usuarios para opinar sobre los contenidos del videojuego.
    - **Usuario**: nombre del usuario que subió el videojuego.
    - **Fecha de subida**: fecha en la que se subió a la plataforma el videojuego, expresada en DD/MM/AAAA.
    - **Plataformas**: listado de todas las plataformas en las que se encuentra disponible oficialmente el videojuego.
    - **Imagen de la carátula**: archivo de imagen de la carátula del videojuego.
- **Usuario**:
    - **ID**: identificador único del usuario.
    - **Nombre de usuario**: nombre por el que se identifica al usuario dentro de la plataforma.
    - **Email**: correo electrónico para contactar con el usuario.
    - **Contraseña**: contraseña cifrada con la que el usuario inicia sesión.
    - **País**: país de procedencia del usuario.
    - **Roles**: lista de roles del usuario.
    - **Imagen de perfil**: archivo de imagen usado por el usuario para ser reconocido.
- **Comentario**:
    - **ID**: identificador único del comentario.
    - **Texto**: texto que contiene la opinión del usuario.
    - **Categoría**: temática de contenido a la que pertenece el comentario.
    - **Grado de severidad**: nivel de intensidad considerado por el usuario para la temática comentada.
    - **Usuario**: nombre del usuario que escribió el comentario.
    - **Fecha de subida**: fecha en la que el usuario escribió el comentario.
    - **Videojuego**: nombre del videojuego del que trata el comentario.

Para organizar la parte del *back-end* del proyecto se ha optado por usar una arquitectura hexagonal [30]. Con este modelo de arquitectura, los elementos que conforman el *back-end* se organizan en tres grupos:

- Modelo: grupo en el que se concentra toda la lógica de negocio y que es independiente del resto de grupos.
- Puertos: son los casos de uso de la aplicación, donde se realizan las operaciones requeridas en función de la petición del usuario.
- Adaptadores: son las interfaces hacia el exterior del aplicativo a través de los cuales los usuarios solicitan los casos de uso.

La elección de este tipo de arquitectura se basó en su capacidad para desacoplar el conjunto de adaptadores del resto del sistema. Esto permite modificar el modelo de la base de datos consultada por la aplicación o la forma en que se solicitan las peticiones sin alterar la funcionalidad de los casos de uso ni la lógica de negocio. Otra razón para optar por este modelo de arquitectura fue su modularidad, que facilita la construcción de los diferentes flujos de casos de uso. Esto permite identificar las clases necesarias antes de iniciar el

desarrollo, acelerando así la construcción de cada caso de uso. Además, se han seleccionado los patrones DAO (*Data Access Object*) y DTO (*Data Transfer Object*) para modelar los objetos de datos utilizados en las transacciones entre el servidor y la base de datos, y entre el cliente y el servidor, respectivamente.

Esta arquitectura se ha plasmado en el proyecto de la siguiente manera *(Fig.12)*:

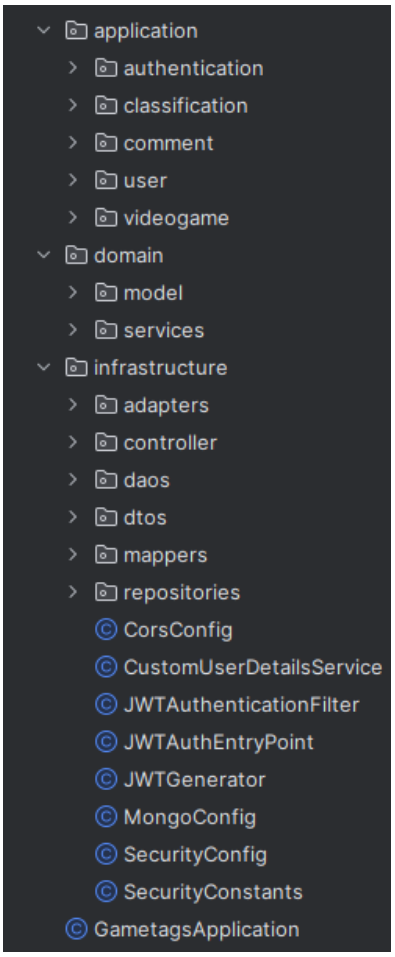

Fig. 12 Jerarquía de archivos del Back-End

Existen tres directorios principales: uno para la capa de infraestructura, otro para la capa de aplicación y un tercero para la capa del modelo.

Dentro de la capa de infraestructura se encuentran todos los archivos que corresponden a la integración del modelo con el exterior, es decir, los controladores de las peticiones REST que se solicitan desde el *front-end*, los adaptadores para realizar *queries* a la base de datos, los DTOs y DAOs usados en dichas solicitudes, los repositorios de cada una de las entidades, los *mappers* para realizar las transformaciones de los objetos a lo largo de cada flujo y las configuraciones específicas sobre la seguridad del *back-end*, la base de datos y la generación de *tokens*.

La capa de aplicación alberga todos los casos de uso que el usuario puede requerir. Cada uno de estos casos de uso se agrupa en subdirectorios correspondientes a las entidades a las que se aplican. Si un caso de uso necesita una clase adicional, existe un subdirectorio específico para ese caso de uso dentro del directorio de la entidad. Este subdirectorio contiene todos los archivos que están específicamente relacionados con ese caso de uso.

En la capa de modelo se almacenan los modelos de negocio de las entidades y los servicios que los usan para realizar las diferentes solicitudes a base de datos mediante adaptadores.

5.1.2 Diseño del Front-End

Para organizar el *front-end*, se ha seguido el formato de estructura que impone Next.js. Dicha estructura consiste en comprimir la lógica y construcción de cada página web dentro de un archivo .js, pudiendo aun así extraer a archivos de uso general funciones compartidas entre varias páginas, denominándolos servicios, y a otro subdirectorio toda la configuración sobre el estilo de las páginas realizado con CSS. Debido a esto, existe un archivo .js por cada una de las páginas definidas en los requisitos. Dentro de cada una de estas páginas se ha definido el HTML usado y las funciones específicas usadas para construir la página. En el caso de que varias páginas compartan funciones, dichas funciones se han extraído a otros archivos .js para centralizarlas y evitar la duplicación de código.

5.1.2.1 Página principal

La página principal es la de inicio, es decir, la página a la que se accede directamente desde el navegador. Se compone solamente del apartado de las últimas incorporaciones. En este apartado se muestran los tres últimos videojuegos añadidos a la plataforma (*Fig.13*)(*Fig.14*), usando la imagen de la carátula de cada videojuego para poder representarlo. Si el usuario hace clic sobre la imagen de la carátula, será redirigido directamente a la página de información del videojuego concreto donde podrá consultar el resto de su información. Si no existen al menos tres videojuegos registrados, se mostrarán los que estén disponibles, ya sea uno o dos.

La primera vez que acceda un nuevo usuario a la página o si el usuario no marcó previamente la casilla de “no volver a mostrar”, aparecerá una ventana modal explicando el funcionamiento del apartado de las últimas incorporaciones.

Fig. 13 MockUp final de la página principal en modo horizontal

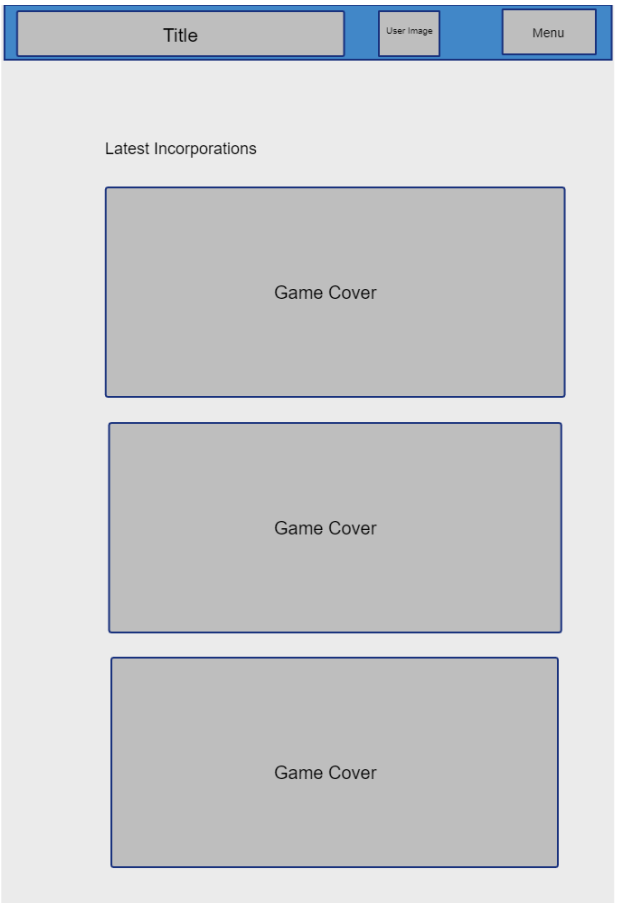

Fig. 14 MockUp final de la página principal en modo vertical

Se exploraron otras opciones de como visualizar los últimos videojuegos incorporados, por ejemplo, usar en vez de una tabla un carrusel (*Fig.15*). Junto con esta opción también se consideró que se recuperasen más de tres videojuegos, debido a la funcionalidad del carrusel. Otra idea que se tuvo fue mostrar más información del videojuego además de la carátula, como su nombre y las etiquetas de las que consta (*Fig.15*).

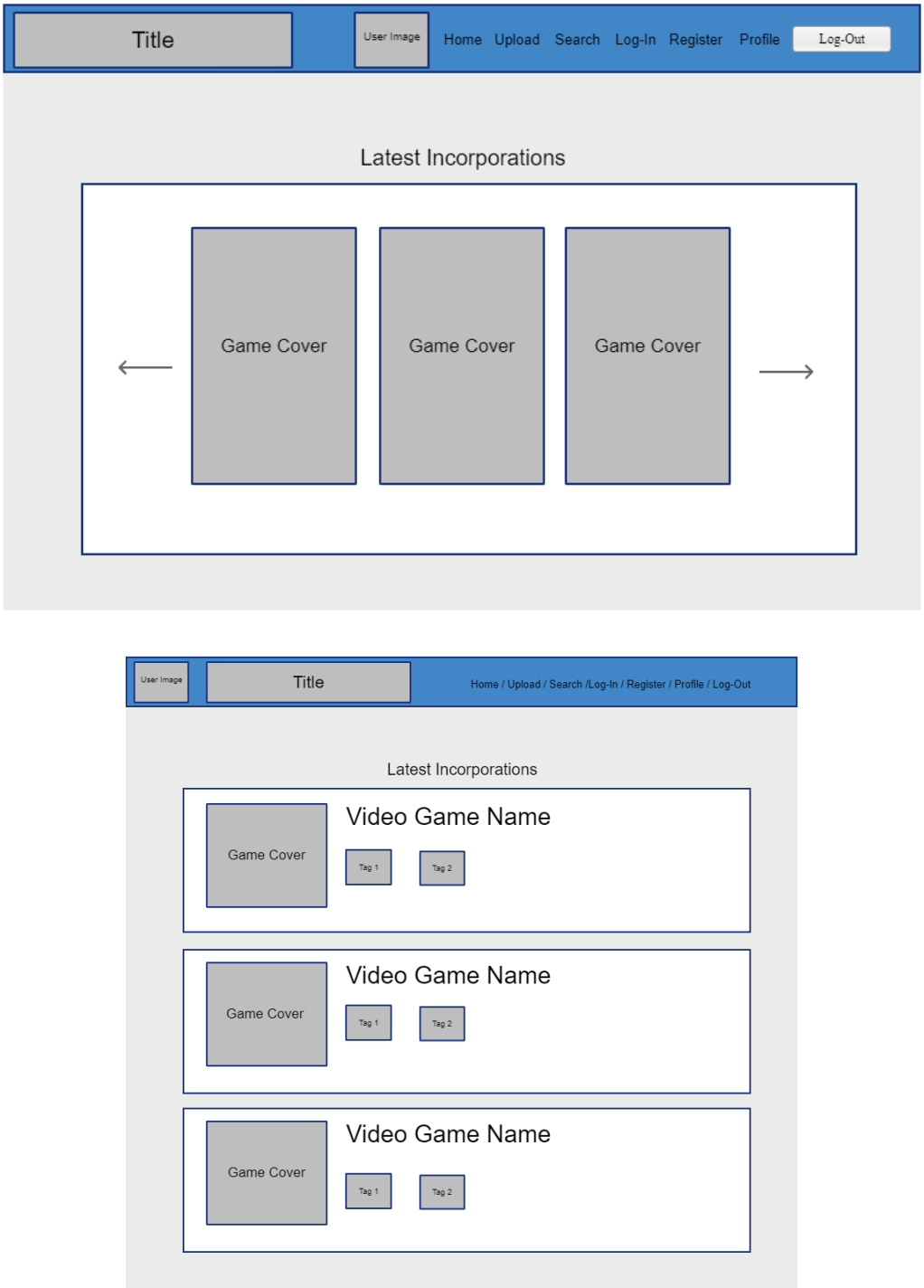

Fig. 15 MockUps de exploración de ideas de la página principal en modo horizontal

Además de hacer pruebas con la visualización en horizontal de las ideas de la página principal, también se hicieron pruebas de cómo se trasladarían en su versión vertical (*Fig.16*).

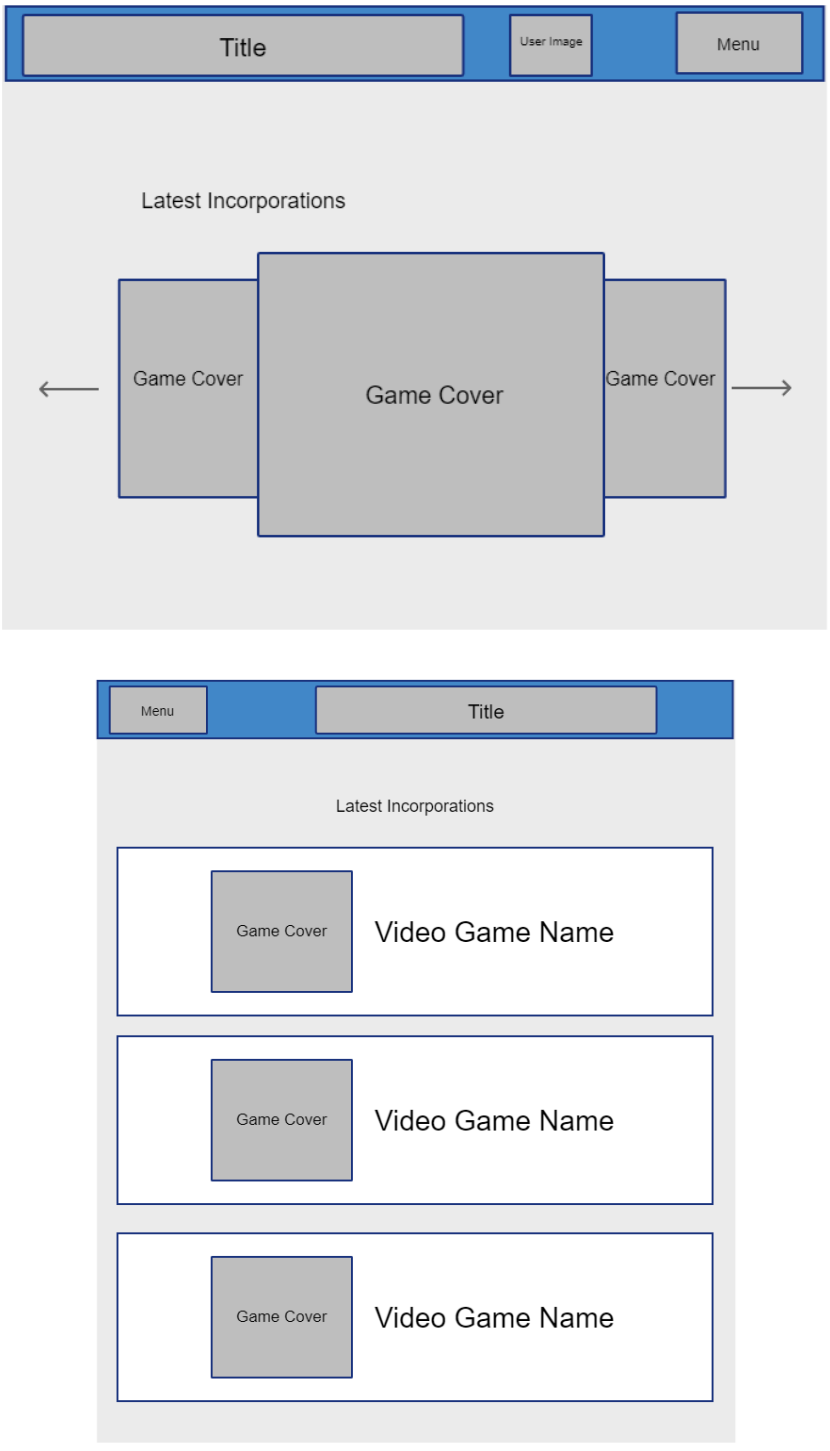

Fig. 16 Exploración del diseño de la página principal en modo vertical

5.1.2.2 Subir un nuevo videojuego

En esta página el usuario debe aportar varios datos para poder agregar un nuevo videojuego a la plataforma. Los datos del videojuego que debe aportar son (*Fig.17*):

- Nombre del videojuego
- Nombre de la desarrolladora del videojuego
- Plataforma/s en las que el videojuego está disponible de forma oficial
- Imagen de la carátula del videojuego

Además de esta información, el videojuego debe contar como mínimo con una clasificación, por eso el usuario debe de especificar también los datos de la clasificación que quiere que tenga el videojuego al ser agregado (*Fig.17*). Dichos datos son:

- Nombre de la entidad de la clasificación
- Abreviatura de la etiqueta de edad de la clasificación
- URL de la página principal de la entidad
- País al que pertenece la entidad
- Imagen de la etiqueta de edad

En el caso de que tenga dudas sobre la clasificación que quiere que acompañe al videojuego o directamente no sabe cuál es la edad del videojuego, el usuario puede consultar directamente en las páginas oficiales de las entidades que son soportadas por GameTags dicha información por medio de unos enlaces dentro de la propia página de subida del videojuego (*Fig.17*).

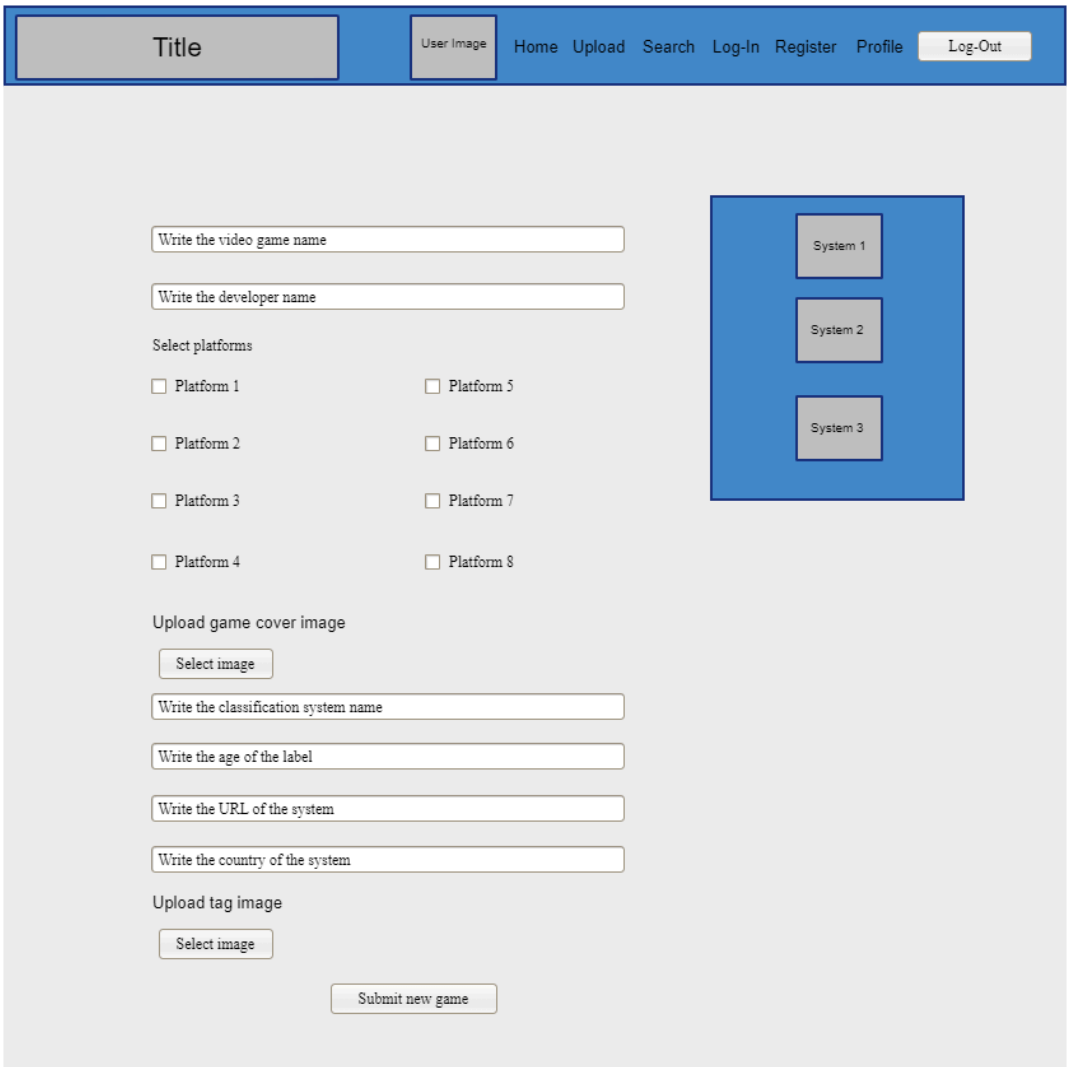

Fig. 17 MockUp final de la página de subida de un nuevo videojuego en modo horizontal

En la versión para pantallas verticales de la página, la sección donde aparecen los enlaces de las entidades oficiales soportadas se sitúa al final de la página y no a la derecha (*Fig.18*). Además, la disposición de las plataformas también cambia. En vez de aparecer en dos columnas, aparecen en una sola columna (*Fig.18*).

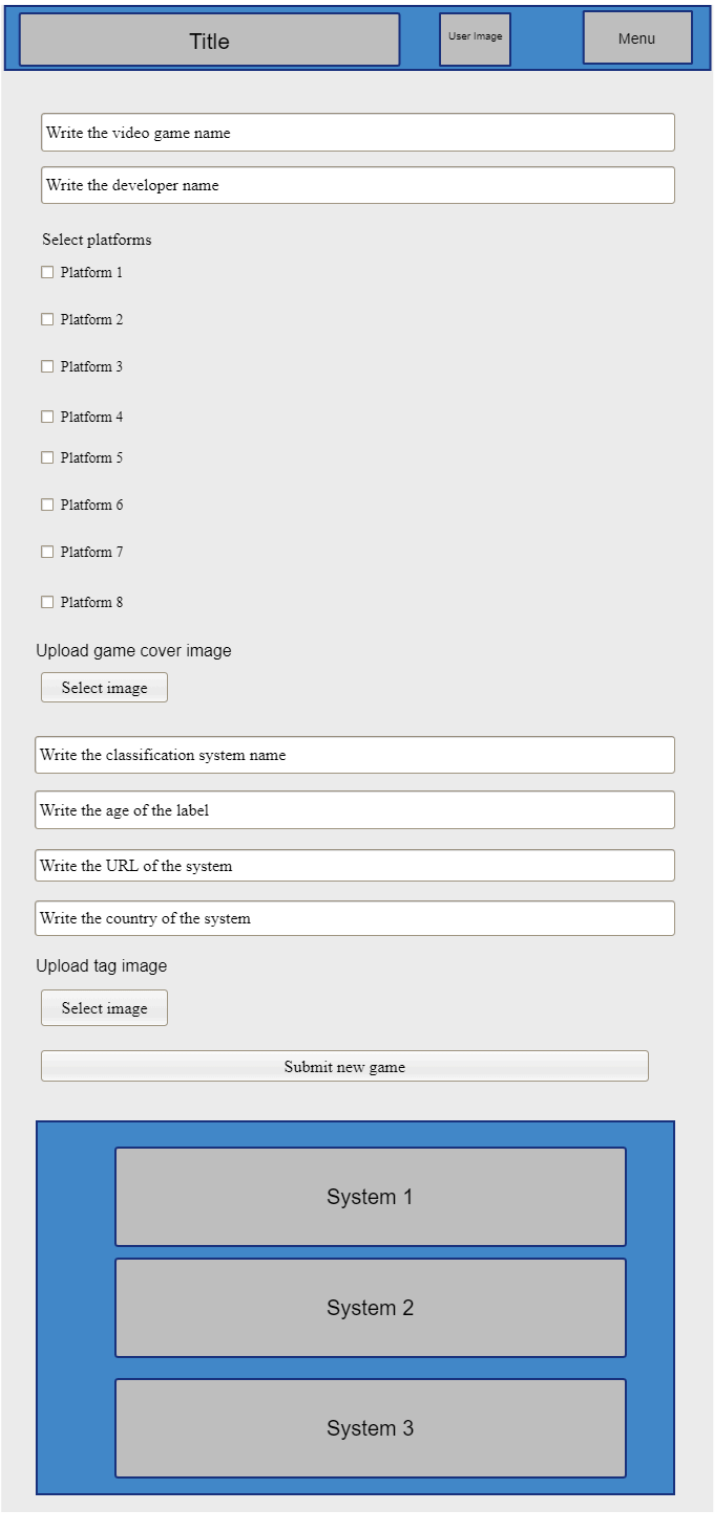

Fig. 18 MockUp final de la página de subida de un nuevo videojuego en modo vertical

Si el usuario no completa todos los campos requeridos, se desplegará una ventana emergente indicando que algunos campos están vacíos. Se solicitará al usuario que complete todos los campos para poder proceder con la operación de carga. Además, si el videojuego que el usuario intenta cargar ya está presente en la base de datos, se mostrará una ventana adicional informando que el videojuego ya existe y se sugerirá al usuario que intente agregar un videojuego diferente.

Se exploraron opciones de cómo organizar los elementos a rellenar por el usuario, tanto probando a que los campos propios del videojuego estuviesen contenidos dentro de un rectángulo y los propios de la clasificación estuviesen contenidos en otro rectángulo (*Fig.19*) como intentando ocupar más ancho de pantalla (*Fig.20*). También se exploraron ideas de cómo hacer la selección de plataformas y de etiquetas de edad, probando a tener las opciones dentro de un *dropdown* (*Fig.19*)(*Fig.20*). La idea del *dropdown* para la etiqueta de edad se pensó para facilitar al usuario la selección de esta y evitar errores gramaticales por su parte, mientras que su uso para las plataformas se pensó para compactar la apariencia y tamaño de la página. También se pensó en cambiar la forma de representación de las entidades de clasificación oficiales en la página, de representarlas por medio de sus logos a representarlas por medio de sus nombres (*Fig.19*).

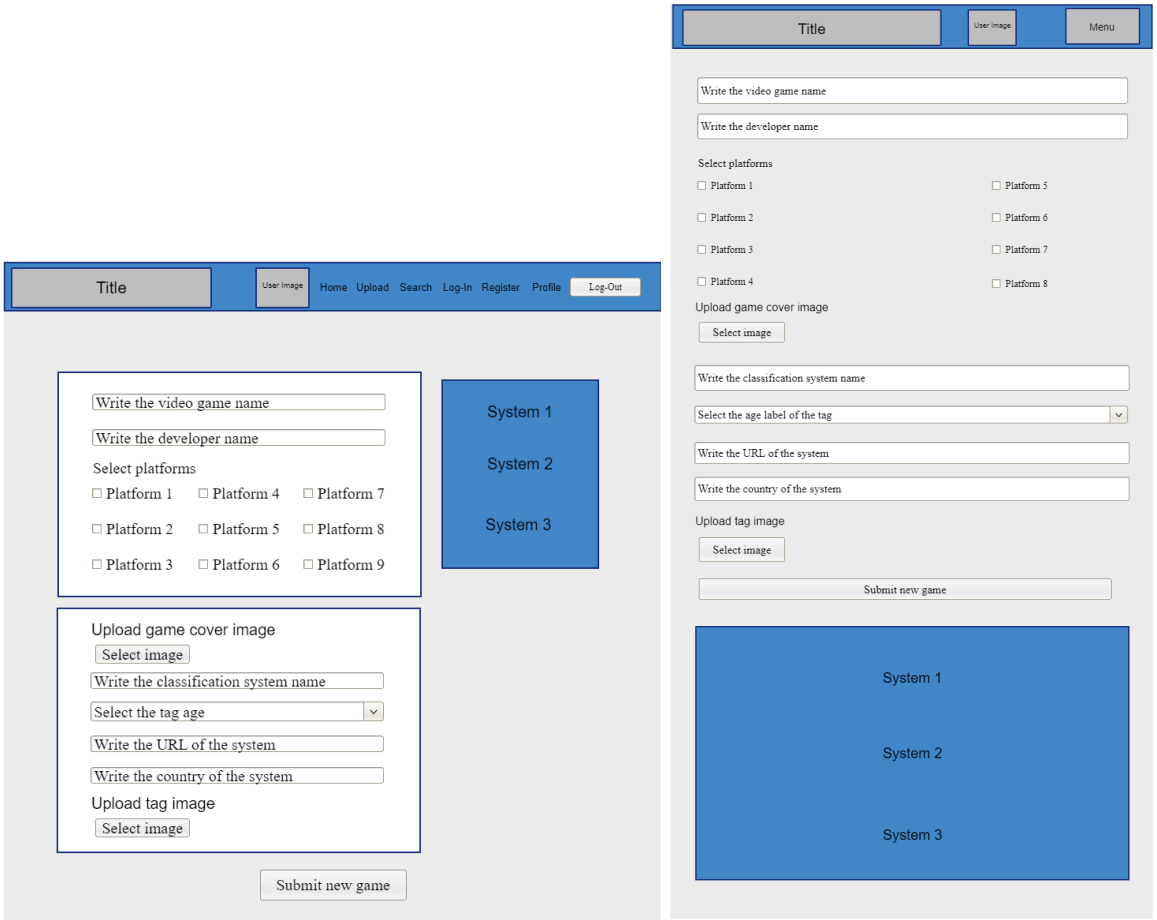

Fig. 19 MockUps de la versión horizontal y vertical de la primera exploración

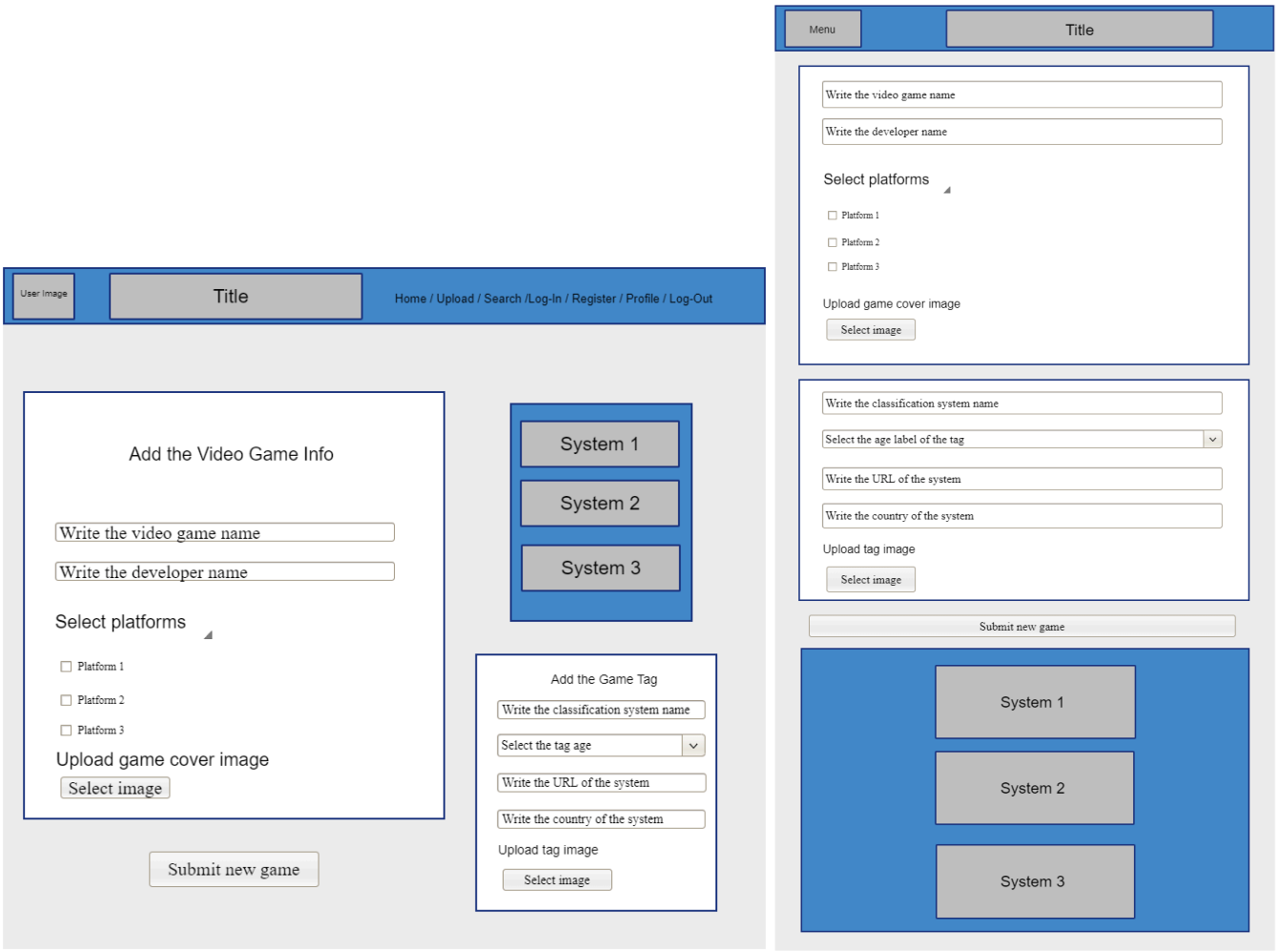

Fig. 20 MockUps de la versión horizontal y vertical de la segunda exploración

El tutorial en esta página explica al usuario el propósito de ella, cómo debería escribir la etiqueta de edad para evitar errores e indica que, en el caso de dudas, puede consultar en las páginas oficiales por medio de los enlaces en la página.

5.1.2.3 Iniciar sesión

En la página del inicio de sesión el usuario introduce dos datos: el nombre del usuario del que quiere iniciar sesión y la contraseña con la que se identifica dicho usuario (*Fig.21*).

Si la contraseña ingresada no coincide con la asociada al usuario, se rechazará el acceso. El usuario será informado del error a través de una ventana emergente y se le solicitará que lo intente de nuevo. Si el nombre de usuario ingresado no existe en la base de datos, se informará al usuario de su inexistencia también mediante una ventana emergente,

invitándole a intentarlo de nuevo. Si el usuario no ha proporcionado ninguno de estos dos datos y trata de continuar con la operación, se le impedirá hacerlo e informará con el mismo mensaje de error descrito anteriormente.

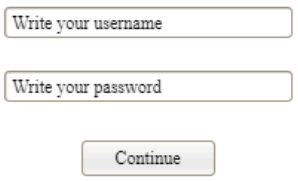

Fig. 21 MockUp final de la página del inicio de sesión

No se realizaron exploraciones de ideas alternativas al diseño original debido a la sencillez de los datos requeridos al ser necesarios solo esos dos para iniciar la sesión. Tampoco cuenta con una versión ajustada a las ventanas verticales igualmente porque solo se piden dos campos de información.

Las ventanas modales de tutorial de esta página indican los datos que hay que introducir en cada campo.

5.1.2.4 Registrar nuevo usuario

En la página de registro de un nuevo usuario se solicitan cuatro datos: el nombre de usuario que se quiere usar, la contraseña con la que se identificará este nuevo usuario, el país de procedencia del usuario y un correo electrónico de contacto. Además, hay un campo adicional para reintroducir la contraseña y verificar que no ha habido confusión en la escritura de esta (*Fig.22*).

Si falta información en alguno de los cinco campos, se mostrará una ventana emergente notificando que la operación no se pudo completar. Se instará al usuario a intentarlo de nuevo. Si la contraseña no coincide con la contraseña reescrita, aparecerá otra ventana modal informando de que la contraseña escrita en los dos campos no coincide y pidiendo al usuario que escriba la misma en ambos. Si la operación de registro del nuevo usuario se completa satisfactoriamente, aparecerá una ventana emergente informando del éxito de la

operación que redirigirá al usuario a la página principal, informando de que ahora puede iniciar sesión con ese nuevo usuario.

Fig. 22 MockUp final de la página de registro de un nuevo usuario

Similar a la página de inicio de sesión, no se consideraron ideas alternativas debido a la sencillez de los datos a ingresar. Además, no se dispone de una versión adaptada para pantallas verticales debido a la disposición de los campos en el diseño original.

Las ventanas emergentes del tutorial de esta página indican el propósito de la página y la información que debe proporcionar el usuario para llevar a cabo la operación.

5.1.2.5 Perfil del usuario

En esta página el usuario puede consultar sus datos: nombre, país, roles, correo electrónico de contacto y la imagen de perfil que usa (*Fig.23*). Además de esto puede acceder a las operaciones relacionadas directamente con su perfil, como cambiar su nombre de usuario, cambiar su contraseña, consultar los comentarios que ha escrito, cambiar de imagen de perfil y borrar su cuenta (*Fig.23*).

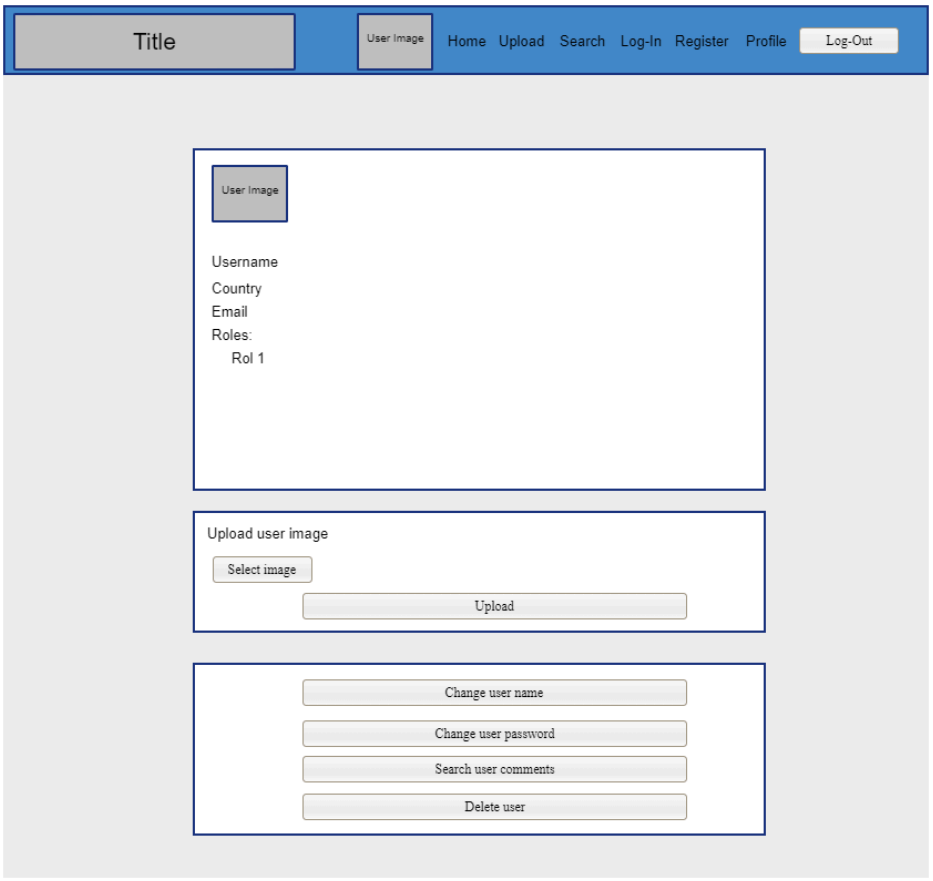

Fig. 23 MockUp final de la página del perfil del usuario

Si el usuario pretende cambiar de imagen de perfil, pero no ha seleccionado ninguna en su dispositivo, se le mostrará una ventana modal explicando que debe elegir una imagen antes de realizar la acción. Cuando vaya a borrar su cuenta, aparecerá una ventana emergente según el resultado de la operación. Si se eliminó la cuenta correctamente, la ventana le indicará de dicho éxito y le redirigirá a la página principal. En el caso de que ocurra algún error, se le indicará que ha habido un problema y se mantendrá en la página.

Cabe destacar que no existe una versión alternativa de esta página para pantallas verticales porque no era necesario por la disposición de los elementos en la idea elegida.

Se experimentó con la organización de los elementos de la página, buscando hacer secciones claras dentro de ésta (*Fig.25*) o aprovechar el ancho de la página (*Fig.24*).

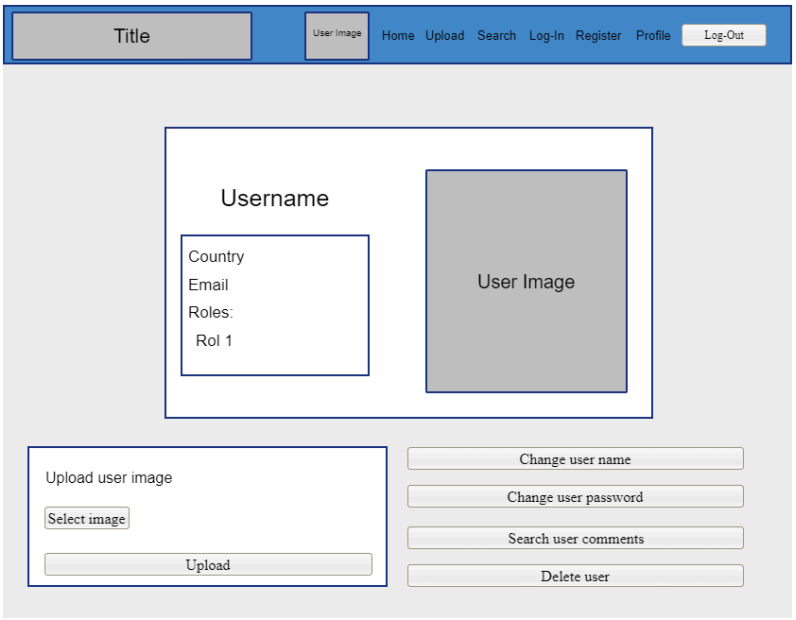

Fig. 24 MockUp de la exploración de la página de perfil aprovechando el ancho de pantalla

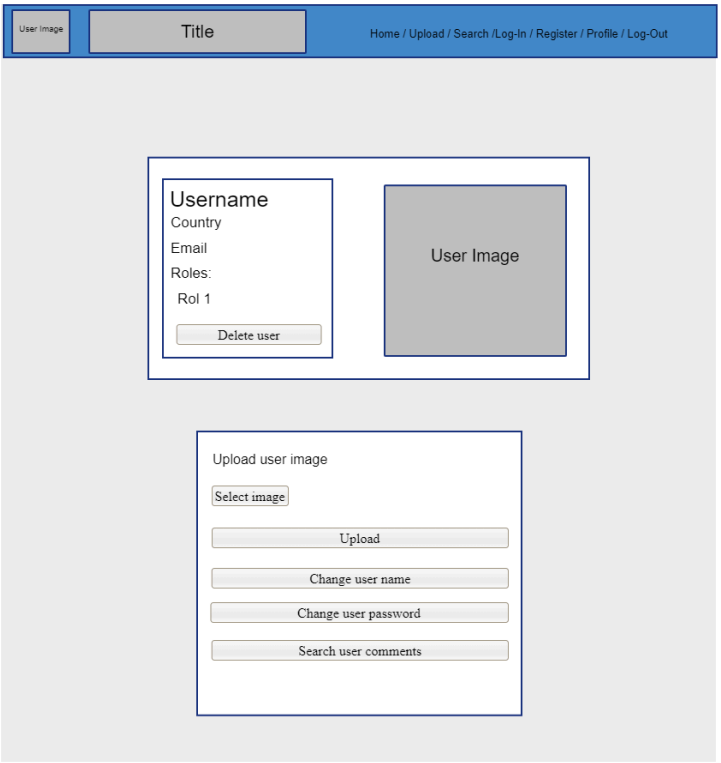

Fig. 25 MockUp de la exploración de la página de perfil agrupando los elementos en secciones

El tutorial de esta página explica al usuario el propósito de ésta, además de mostrarle las distintas opciones que tiene disponibles.

5.1.2.6 Cambiar nombre de usuario

En esta página el usuario debe introducir tres datos: el nombre de usuario que usa actualmente, el nombre de usuario que quiere empezar a usar y su contraseña como confirmación del cambio (*Fig.26*).

Si el usuario ingresa una contraseña que no coincide con la que está actualmente en uso, aparecerá una ventana emergente informándole de la discrepancia y solicitándole que lo intente de nuevo.

Por otro lado, si no rellena la información de todos los campos, aparecerá otra ventana emergente pidiéndole que rellene todos los campos para poder completar la operación. Una vez realizada dicha operación, se mostrará una ventana emergente indicando el éxito de la operación, la cual redirigirá al usuario a su página de perfil.

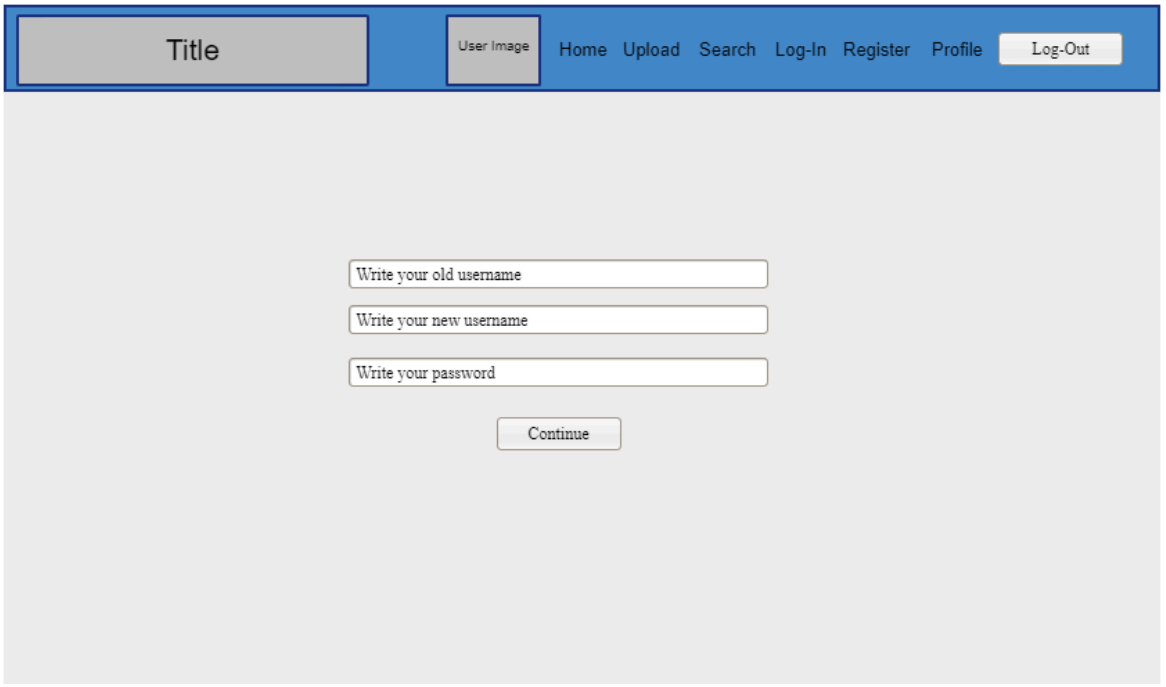

Fig. 26 MockUp final de la página de cambio del nombre de usuario en modo horizontal

No cuenta con una versión alternativa para pantallas verticales debido a la organización de los campos en la idea original para la página.

Con respecto a la exploración de alternativas sobre la página, fueron orientadas a la estética de esta, pensando en contener los campos dentro de un rectángulo para resaltar los elementos importantes de la página (*Fig.27*).

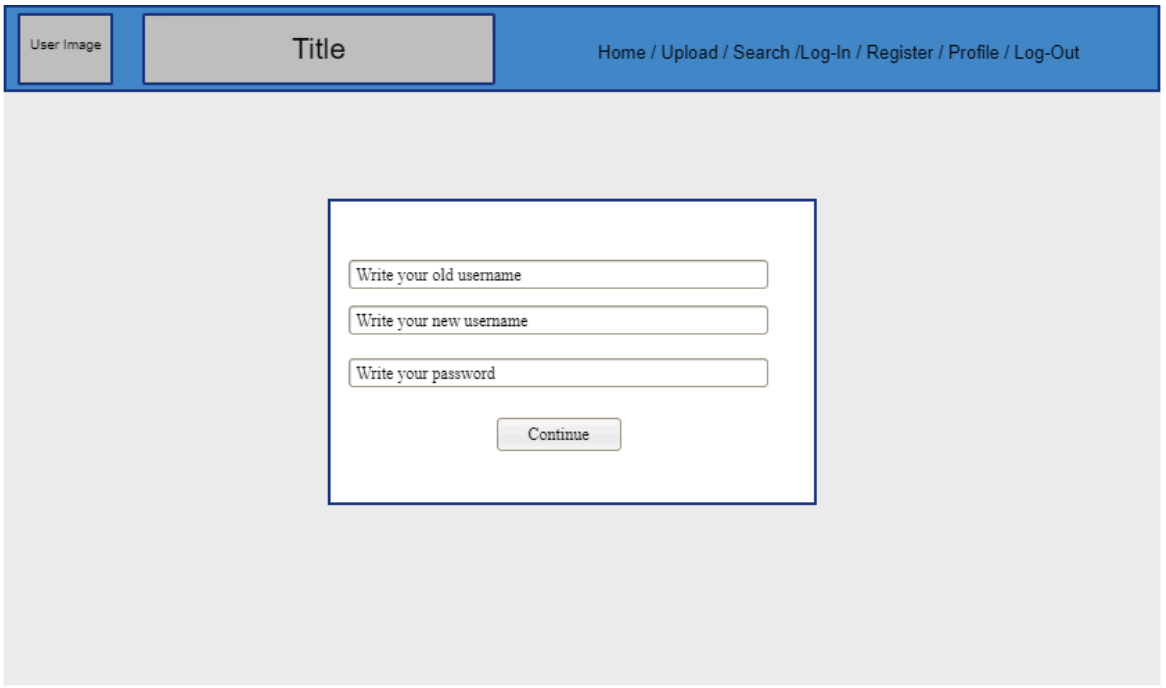

Fig. 27 MockUps de exploración de ideas de la página del cambio del nombre de usuario

El tutorial en esta página explica el objetivo de la página y los datos que el usuario debe aportar para llevar a cabo la operación.

5.1.2.7 Cambiar contraseña

En la página del cambio de contraseña, el usuario debe aportar tres datos para realizar la operación: la contraseña que usa actualmente, la contraseña que quiere empezar a usar y la contraseña actual de nuevo para confirmar la operación (*Fig.28*). En estos campos la contraseña no aparecerá "en oculto" como en el resto de los campos de contraseña que aparecen en las otras páginas, para evitar errores gramaticales no deseados en la nueva contraseña.

Si la contraseña de confirmación no coincide con la contraseña anterior, se mostrará una ventana emergente señalando el error y solicitando al usuario que introduzca la misma contraseña en ambos campos. Si no se rellenan todos los campos, una ventana emergente

aparecerá indicando la ausencia y pidiendo al usuario que rellene todos los campos para realizar la operación. En el supuesto de que la contraseña introducida en el campo de la contraseña antigua sea la misma que la introducida en el campo de confirmación, pero sea incorrecta, aparecerá otra ventana emergente informando de que la contraseña introducida es incorrecta. Si la operación se realiza correctamente, aparecerá una ventana emergente informando del éxito de la operación, redirigiendo al usuario a la página de su perfil.

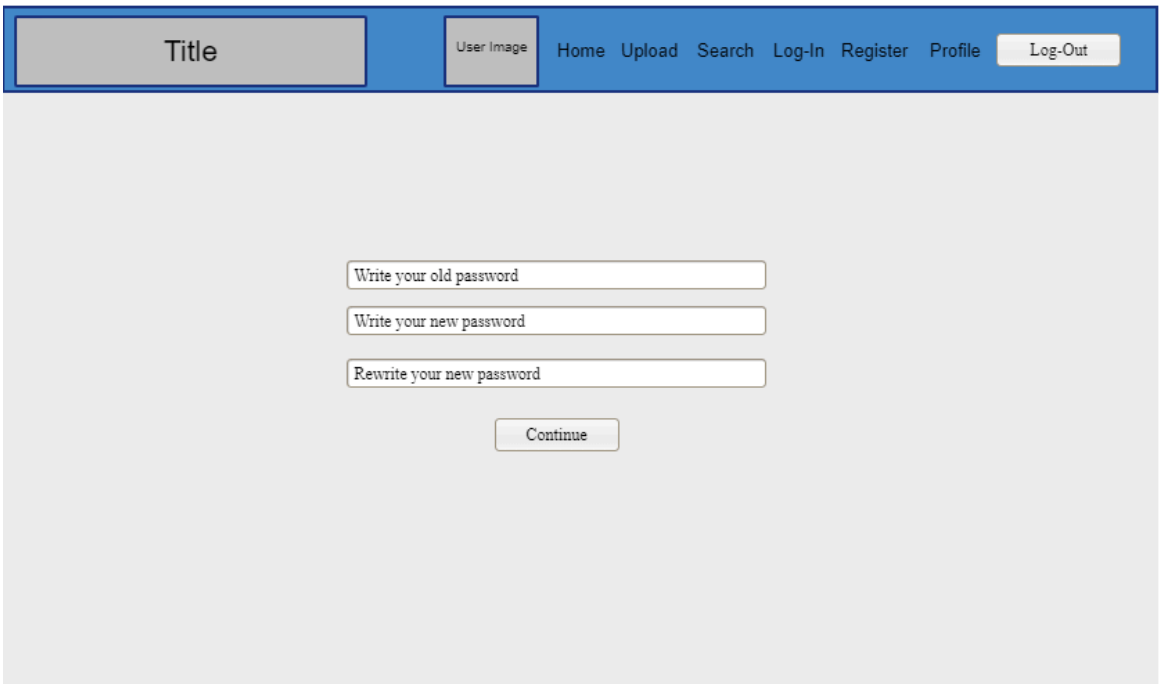

Fig. 28 MockUp final de la página del cambio de contraseña en modo horizontal

Al igual que con la página del cambio del nombre de usuario, esta página no cuenta con una versión alternativa para las pantallas verticales.

De la misma forma, la exploración de versiones alternativas de esta página solo consideró la estética de la presentación de los campos, pensando en contener los elementos dentro de un rectángulo para centrar su atención (*Fig.29*).

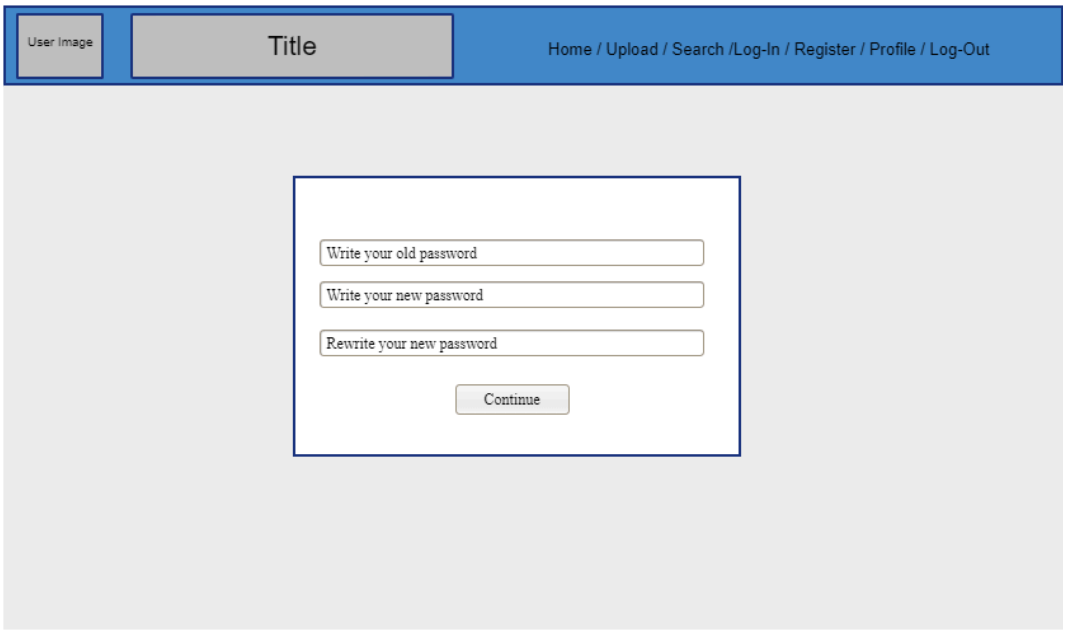

Fig. 29 MockUp de exploración de ideas de la página del cambio de contraseña en modo horizontal

Al igual que en la página del cambio del nombre de usuario, el tutorial en esta página explica el propósito de esta y la información a aportar en los campos.

5.1.2.8 Consultar comentarios del usuario

Esta página se divide en dos secciones: la primera donde el usuario elige los campos por los que quiere filtrar los comentarios a buscar y la segunda donde aparece un listado de los comentarios escritos por el usuario que cumplen el filtro elegido (*Fig.30*). Los filtros que puede usar el usuario para hacer la búsqueda son la severidad que indicó en el comentario, la categoría a la que pertenece el comentario o el nombre del videojuego para el que escribió el comentario. Solo puede elegir uno de los filtros para hacer la búsqueda, no puede combinar más de uno para refinar la búsqueda. En la sección de resultados los comentarios aparecen en forma de lista, primero mostrando su nivel de severidad con su color correspondiente, seguido del texto del comentario, la categoría del comentario, la fecha de subida y el nombre del videojuego sobre el que trata (*Fig.30*). El nombre del videojuego es un hipervínculo que redirige a la página de información del videojuego en cuestión.

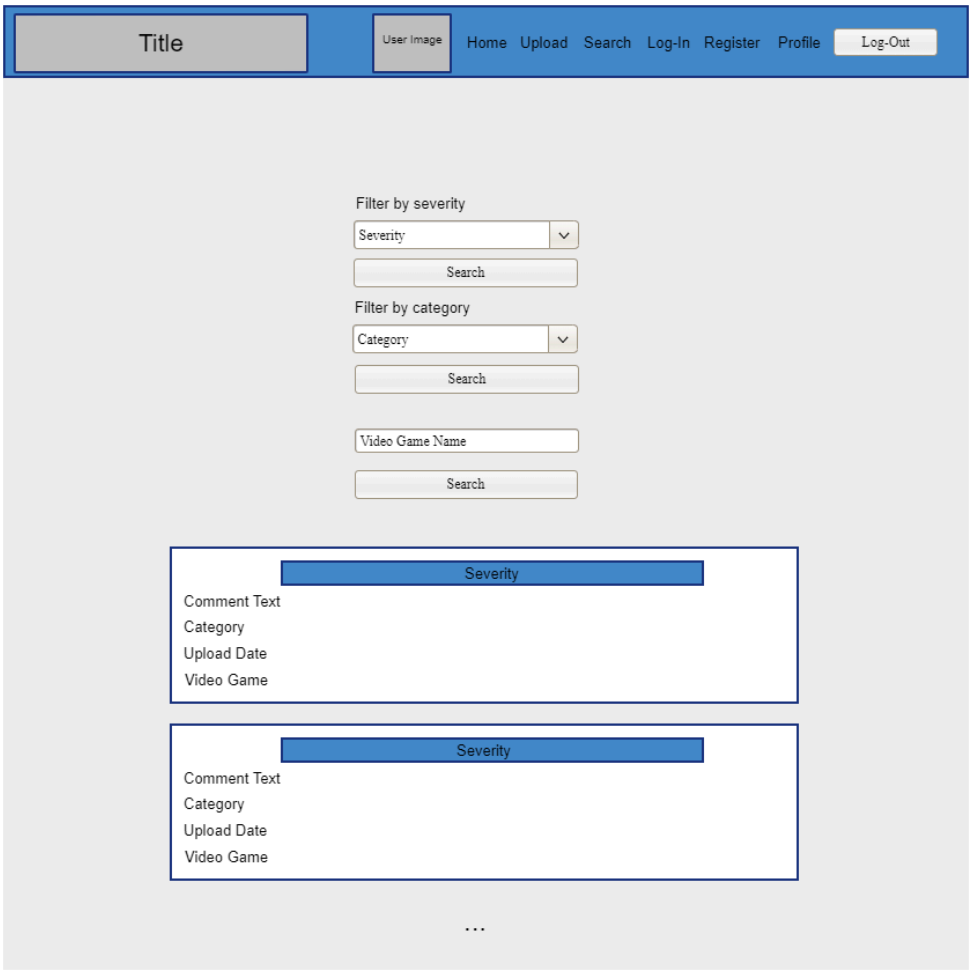

Fig. 30 MockUp final de la página de búsqueda de comentarios del usuario en modo horizontal

Si se da el caso de que el usuario no ha escrito ningún comentario que cumpla el filtro elegido, aparecerá una ventana emergente indicando que no se encontraron comentarios, recomendándole usar otro filtro de búsqueda.

Esta página no tiene una versión alternativa para pantallas verticales porque no era necesario reorganizar la información de pantalla.

Las ideas que se exploraron sobre esta página giraron en torno a usar más de un filtro para realizar la búsqueda y en reorganizar la disposición de la información dentro del comentario (*Fig.31*) y la disposición de las secciones de la página (*Fig.32*).

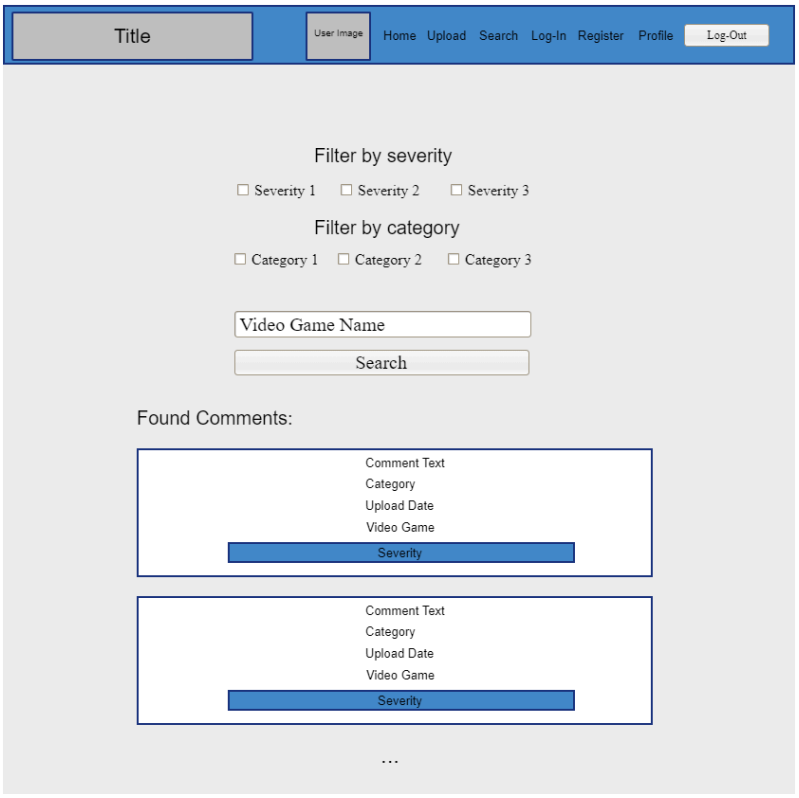

Fig. 31 MockUp del uso de más de un filtro en la búsqueda de los comentarios del usuario

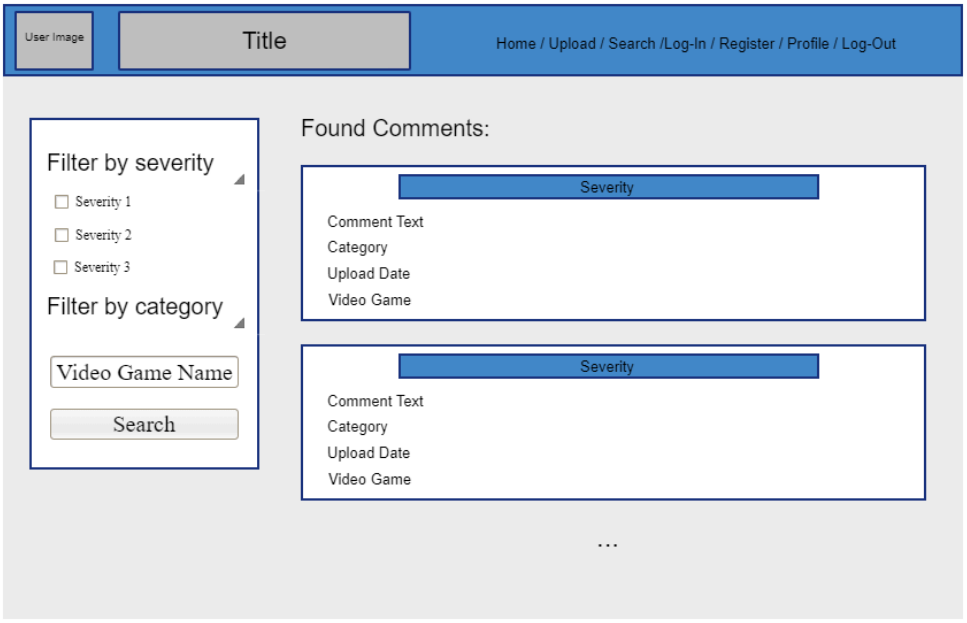

Fig. 32 MockUp de la alternativa de distribución de las secciones de la página de la búsqueda de los comentarios del usuario

El tutorial de esta página indica las dos secciones en las que se divide y el funcionamiento de cada una de ellas.

5.1.2.9 Página de búsqueda

En esta página el usuario realiza la búsqueda del videojuego o videojuegos sobre los que quiere consultar su información. Para ello, el usuario realiza la búsqueda usando uno de los siguientes filtros (*Fig.33*):

- Nombre del videojuego
- Edad
- Plataforma
- Entidad clasificadora
- Nombre de la desarrolladora

Al filtrar por edad, plataforma o entidad, solo se puede seleccionar una de las opciones disponibles. Por ejemplo, no es posible buscar videojuegos que hayan recibido tanto una calificación de +16 como de +12. En cuanto a la búsqueda por el nombre del videojuego o el nombre del desarrollador, se buscarán todos los videojuegos que incluyan el texto introducido en el campo, en lugar de aquellos cuyo nombre coincida exactamente con el texto ingresado. Una vez realizada la operación, los resultados aparecerán en la parte inferior de la página en forma de listado, mostrando una tarjeta por cada videojuego encontrado, conteniendo la imagen de la carátula del videojuego, el nombre del videojuego, el nombre de la desarrolladora y las imágenes de las etiquetas de edad del videojuego (*Fig.33*). Al hacer clic sobre el nombre del videojuego se moverá al usuario a la página de información del videojuego.

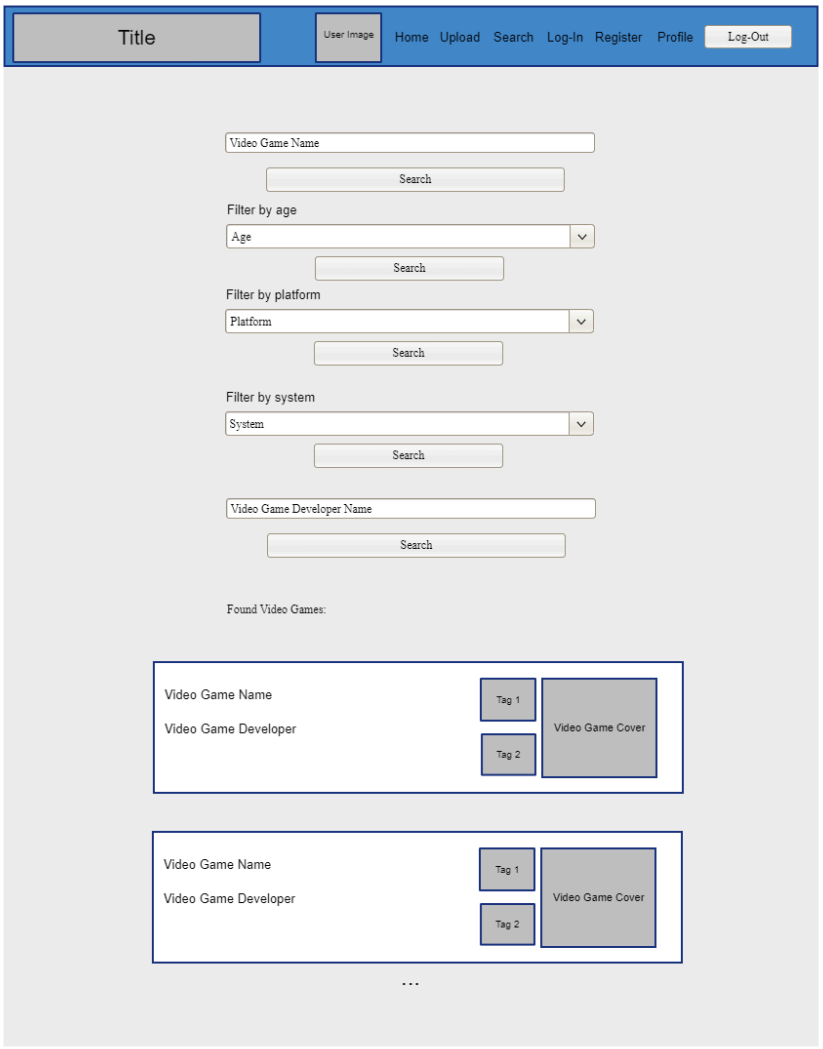

Fig. 33 MockUp final de la página de búsqueda de videojuegos en modo horizontal

La versión alternativa de esta página para pantallas verticales cambia la forma en la que se organizan los elementos de las tarjetas de los videojuegos mostrados en la sección de resultados. En esta versión solo se muestran el nombre del videojuego y la imagen de la carátula, para acomodarlo a la orientación de la pantalla (*Fig.34*).

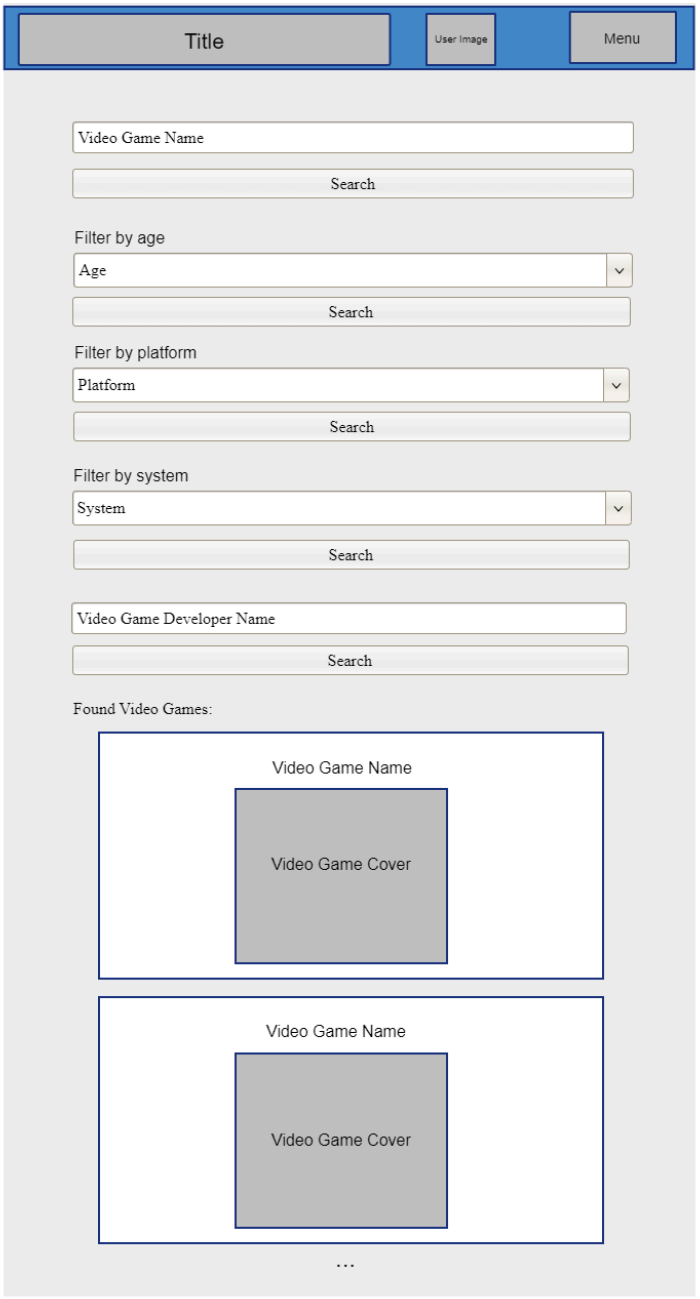

Fig. 34 MockUp final de la página de búsqueda de videojuegos en modo vertical

Si, al buscar con uno de los filtros, no se encuentra ningún videojuego que cumpla dicho criterio, aparecerá una ventana emergente informando de que no se encontró ningún videojuego, diciéndole al usuario que pruebe con otro filtro.

Se exploró la idea de cambiar la forma de organizar los elementos de la página, dividiéndolo en dos columnas en la versión para pantallas horizontales (*Fig.35*) y volver a juntarlo todo en una única columna en la versión para pantallas verticales (*Fig.36*). Además, en esta exploración también se pensó en cambiar la forma en la que se seleccionaban los filtros con

los que hacer la búsqueda, pasando a ser *dropdowns* de elección múltiple (*Fig.35*). Dentro de esta idea también se contempló cambiar la presentación de los datos de los videojuegos en las tarjetas tanto en la versión para pantallas horizontales (*Fig.35*) como para pantallas verticales (*Fig.36*), en esta última mostrando también las imágenes de las etiquetas de edad.

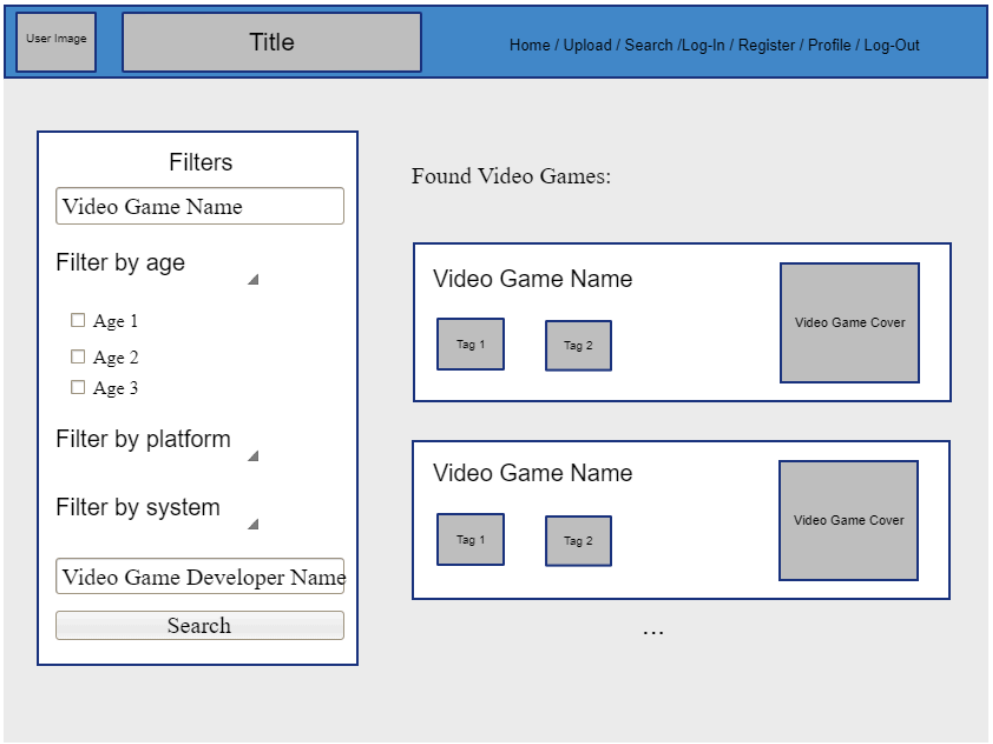

Fig. 35 MockUp de la exploración de las dos columnas en la página de búsqueda de videojuegos en modo horizontal

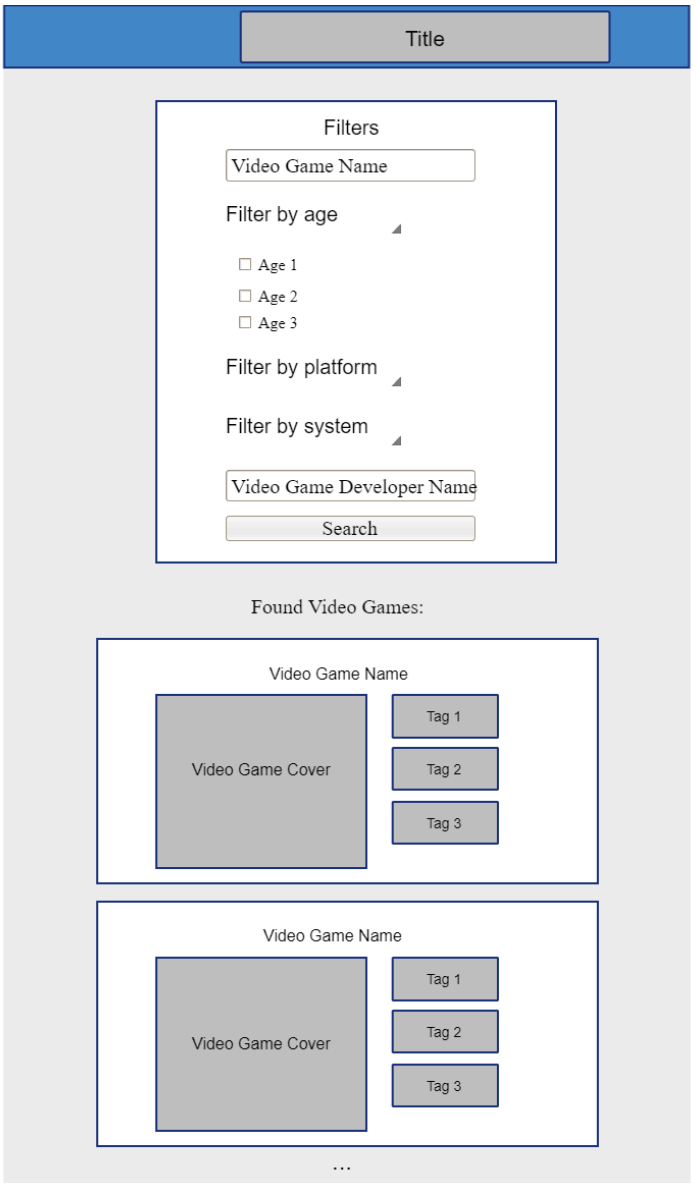

Fig. 36 MockUp de la exploración de las dos columnas en la página de la búsqueda de videojuegos en modo vertical

Otra idea que se contempló fue la de cambiar de nuevo la forma de selección de filtros para que fuese también de selección múltiple pero que en vez de usar un *dropdown* fuesen un conjunto de *checkboxes* (*Fig.37*). En la versión para pantallas verticales también se mantenía el uso de múltiples *checkbox* para la selección múltiple (*Fig.38*).

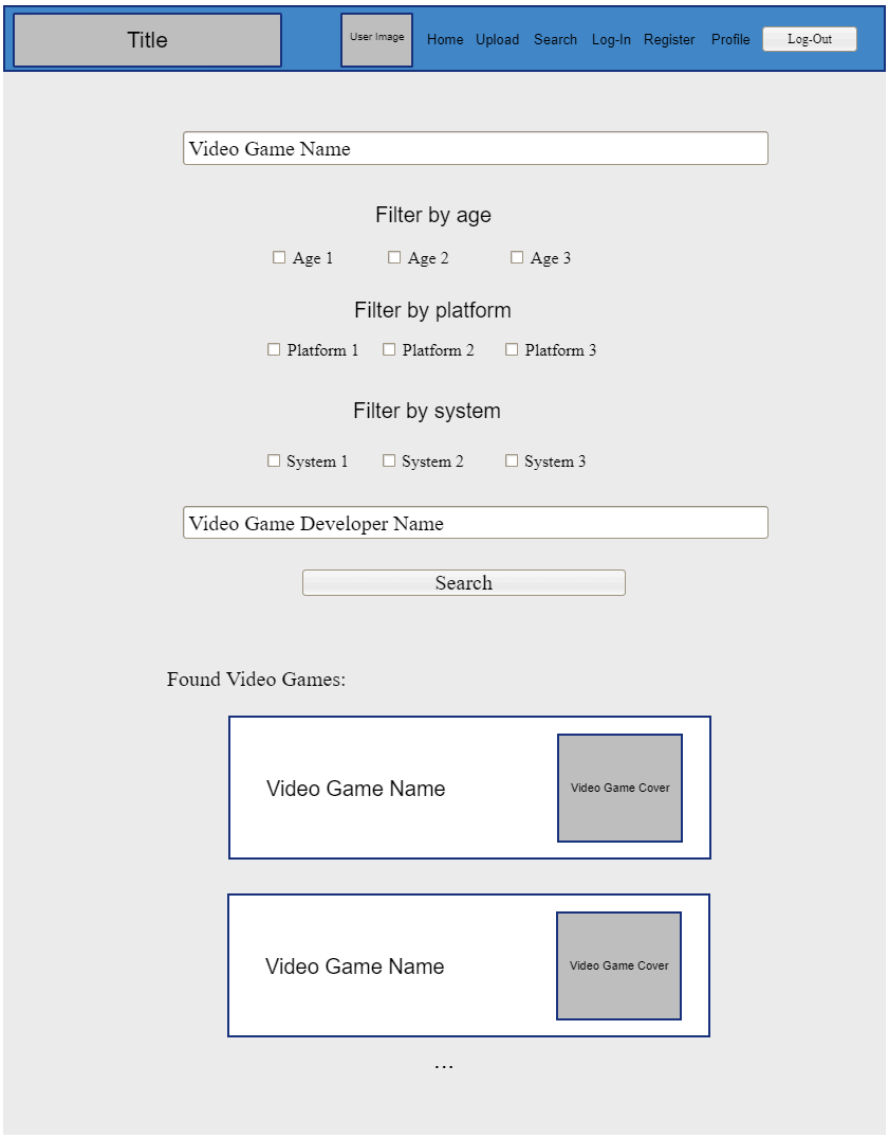

Fig. 37 MockUp de la idea de selección múltiple con *checkbox* en la página de búsqueda de videojuegos en modo horizontal

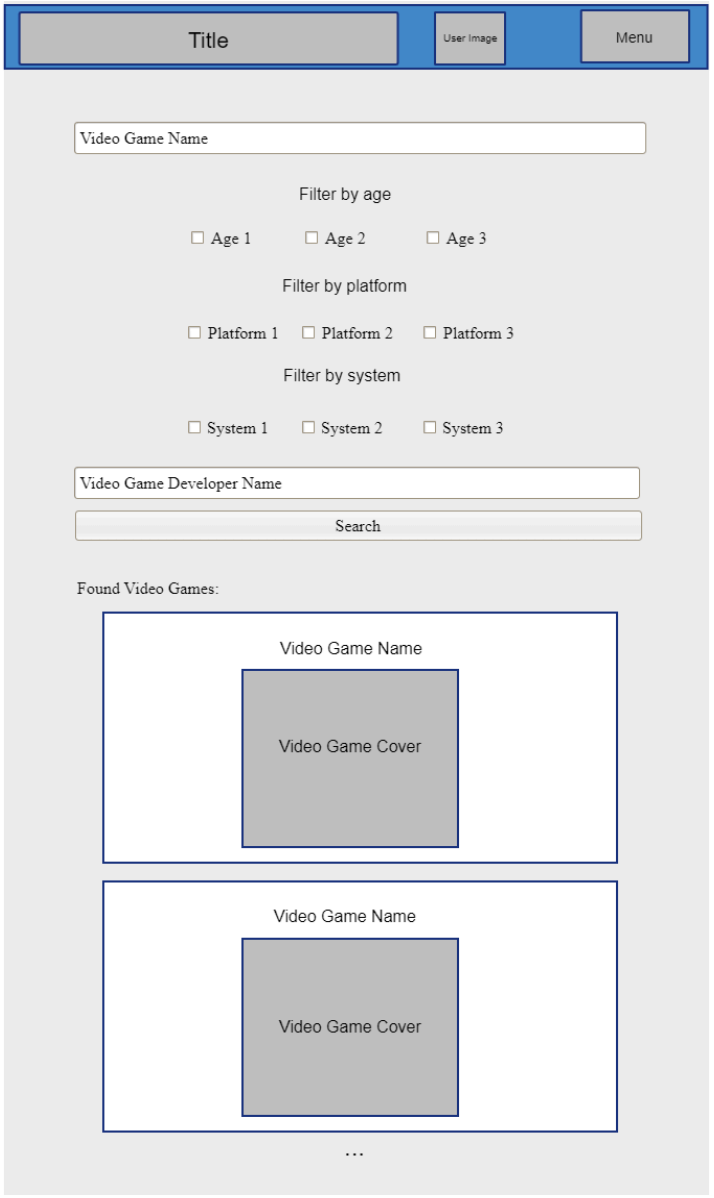

Fig. 38 MockUp del cambio a selección múltiple por checkbox en la página de búsqueda de videojuegos en modo vertical

El tutorial presente en esta página pretende ilustrar su funcionalidad y propósito. Además, orienta al usuario a través de cada filtro disponible para optimizar la búsqueda, y señala el lugar donde se visualizarán los resultados obtenidos.

5.1.2.10 Página de información del videojuego

En esta página el usuario puede consultar la información asociada al videojuego (*Fig.39*):

- Nombre del videojuego
- Nombre de la desarrolladora del videojuego
- Plataformas en las que está disponible el videojuego
- Imagen de la carátula del videojuego
- Imágenes de las etiquetas de edad del videojuego
- Fecha de subida del videojuego a la plataforma
- Nombre del usuario que subió el videojuego

También se indica la valoración media del videojuego en cada una de las temáticas de contenido. Los contenidos son (*Fig.39*):

- Violencia
- Lenguaje Soez
- Sexualidad
- Drogas
- Miscelánea

Junto con estos datos además la página contiene los comentarios realizados por los usuarios sobre cada temática previamente indicada. La información del videojuego y las medias de las categorías se agrupan dentro de una sección mientras que los comentarios se sitúan en una sección aparte. Además, si el usuario ha iniciado sesión, también podrá añadir una nueva clasificación al videojuego y crear un nuevo comentario. En la sección de los comentarios, estos se encuentran agrupados según la temática. Si se quiere ver los comentarios de una categoría específica, el usuario debe de hacer clic sobre el título de la categoría, desplegándose una lista con los comentarios. Cada comentario se encuentra alojado dentro de una tarjeta (*Fig.39*) donde se indica el nivel de severidad otorgado por el usuario para dicha categoría de contenido, un texto que explica los elementos por los cuales considera que merece dicho nivel de intensidad, la fecha en la que se subió el comentario y el usuario que subió el comentario. Si el usuario actual es el usuario que hizo el comentario, puede hacer clic sobre su nombre y lo llevará a su página de perfil. Lo mismo ocurre con el nombre del usuario en la sección de información del videojuego.

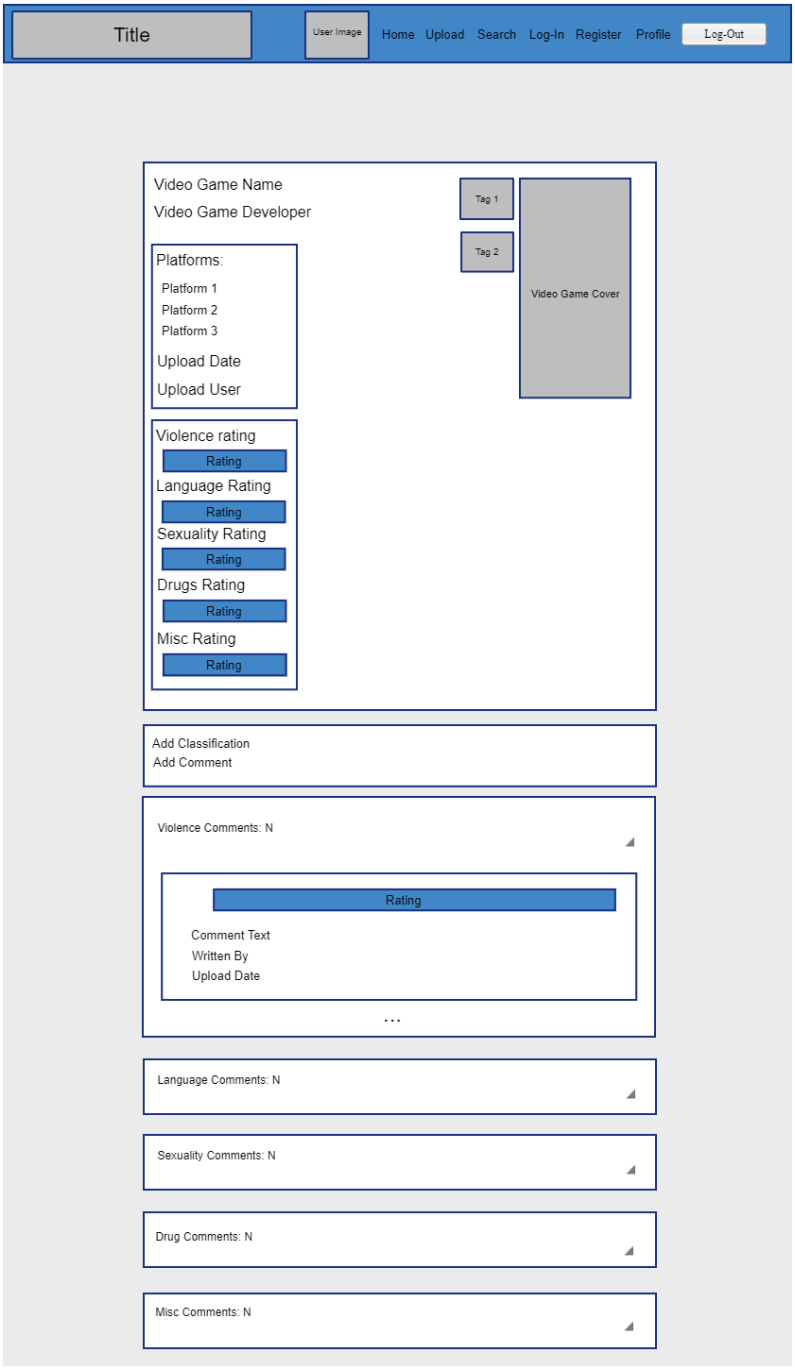

Fig. 39 MockUp final de la página de información del videojuego en modo horizontal

En la versión para pantallas verticales se ha cambiado la disposición de los elementos de la sección de información del videojuego para acomodarse mejor a las pantallas verticales, pasando a estar todo en una única columna (*Fig.40*).

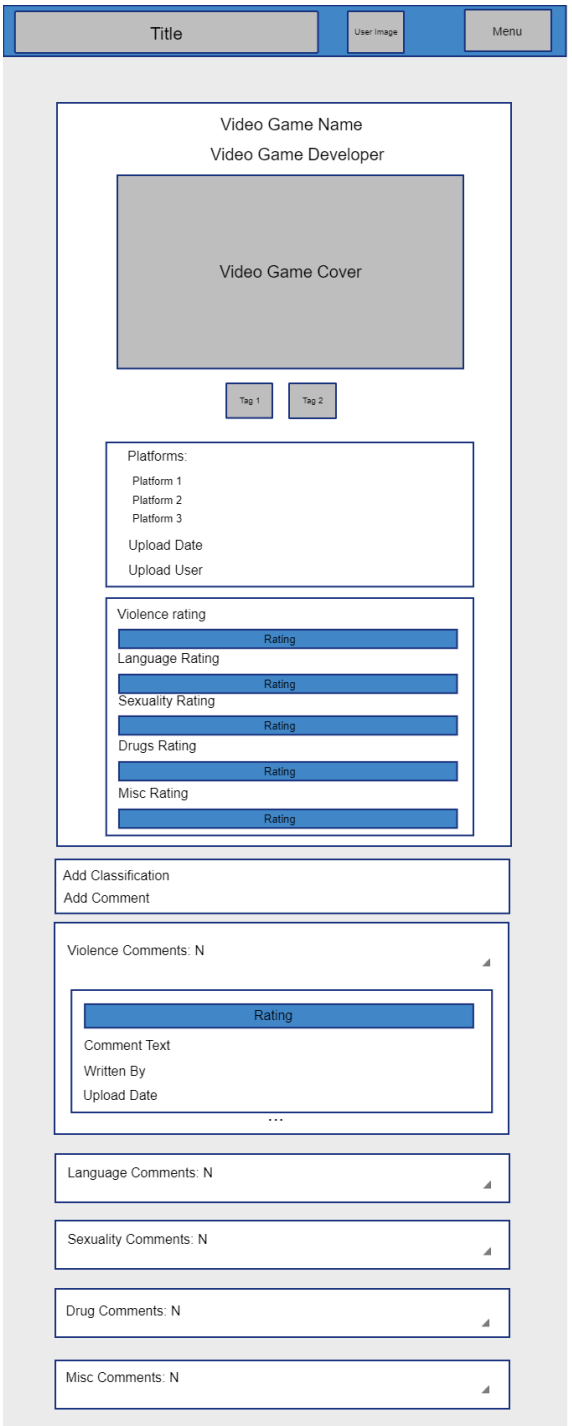

Fig. 40 MockUp final de la página de información del videojuego en modo vertical

Una alternativa que se contempló con respecto a la distribución de los elementos de esta página fue la idea de que, en la sección de información del videojuego, el nombre del videojuego y la imagen de la carátula se situasen en el centro en vez de en un lateral, mientras que la valoración media y el resto de información del videojuego, incluyendo las etiquetas de edad, estuviesen en dos columnas paralelas (*Fig.41*). Además, también se pensó

en cambiar el orden de los datos de los comentarios, descartando el uso del *dropdown* para sustituirlo por el listado completo de los comentarios, sin reducirlo (*Fig.41*). La versión para pantallas verticales se adapta a esta idea haciendo que todo esté en una misma columna (*Fig.42*). Se acabó descartando por preferir la disposición de los elementos de la versión final y porque sin el *dropdown,* la presentación de la sección de los comentarios sería demasiado incómoda.

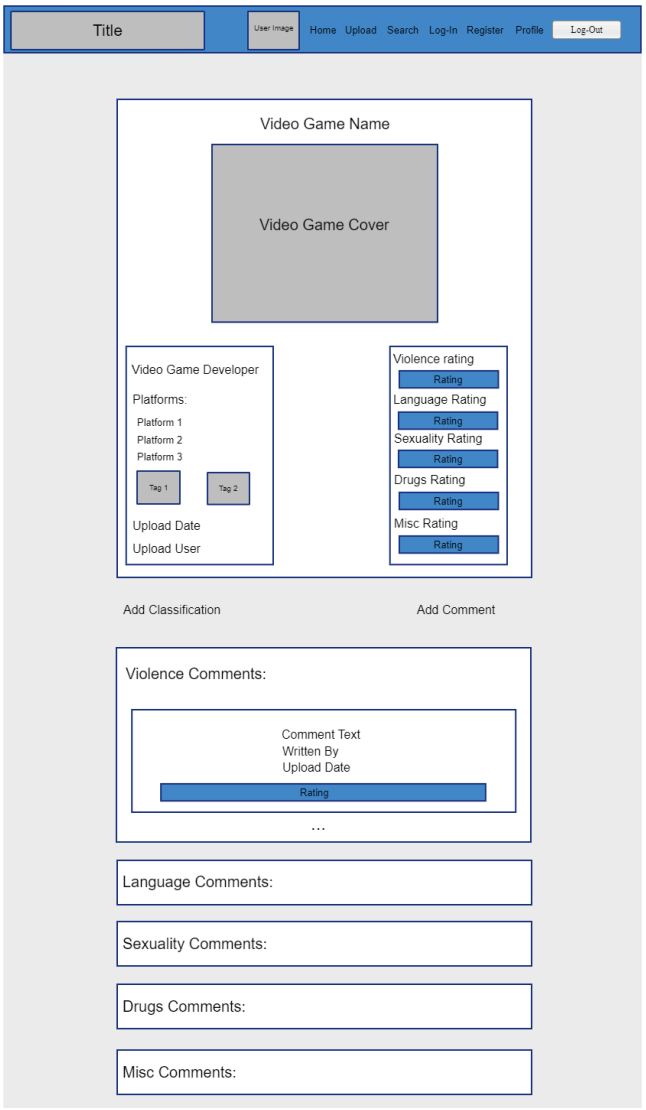

Fig. 41 MockUp de la idea de varias columnas para la información del videojuego y las medias en la página de información del videojuego en modo horizontal

Fig. 42 MockUp de la idea de varias columnas para la información del videojuego y las valoraciones medias en la página de información del videojuego en modo vertical

Otra alternativa en la que se pensó fue, en la tarjeta de información del videojuego, organizarlo en cuatro secciones que fuesen la imagen de la carátula del videojuego, las valoraciones medias de las categorías, la propia información del videojuego y las etiquetas de

edad (*Fig.43*). Además, también se jugó con la idea de no mostrar la cantidad de comentarios de cada categoría, cambiando la organización de los elementos dentro de la tarjeta de un comentario, poniendo al final el texto de éste(*Fig.43*). En la versión para pantallas verticales (*Fig.44*) se sigue usando una distribución de una sola columna para la información del videojuego, excepto con las etiquetas de edad, las cuales se muestran en paralelo a la imagen de la carátula del videojuego. Se descartó esta idea por la forma de organizar los elementos en la sección de información del videojuego.

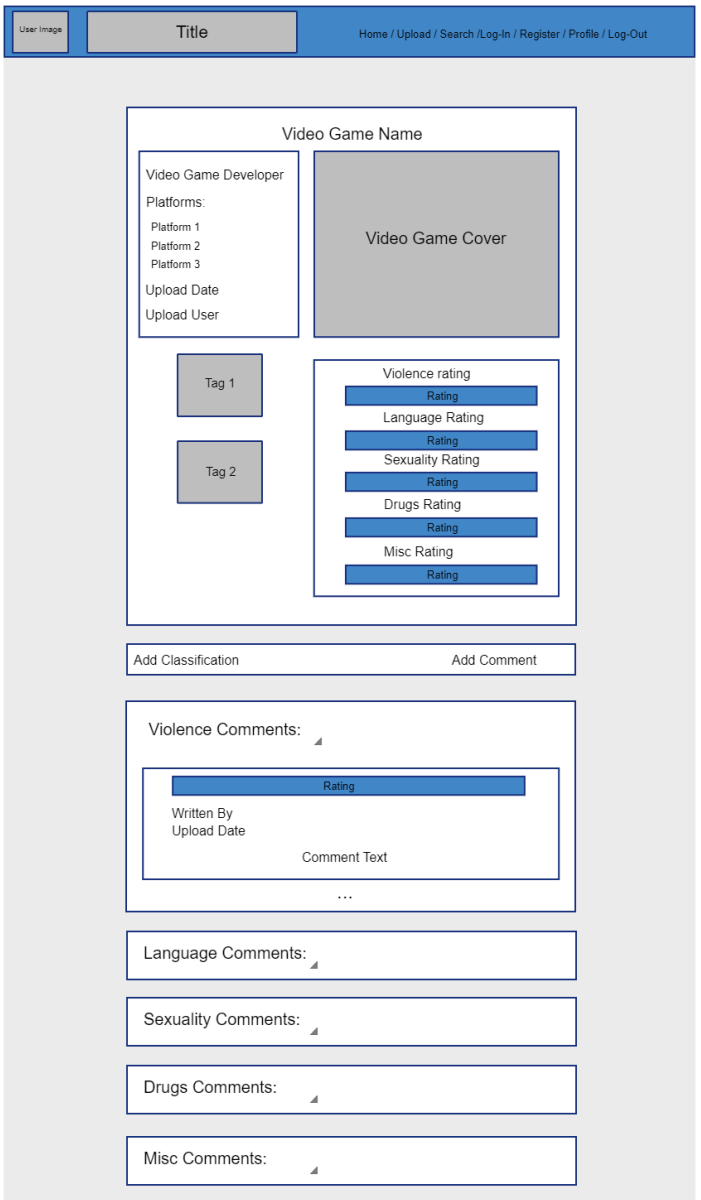

Fig. 43 MockUp de la idea de los cuatro subapartados en la página de la información del videojuego en modo horizontal

Fig. 44 MockUp de la idea de los cuatro subapartados en la página de la información del videojuego en modo vertical

El tutorial de esta página explica la información que se puede encontrar aquí y cada una de las secciones en las que se divide la página, además del tipo de información que contienen los comentarios.

5.1.2.11 Añadir nuevo comentario

En esta página el usuario introduce la información necesaria para poder añadir un comentario a un videojuego. Dicha información consiste en el texto del propio comentario, la categoría sobre la que está comentando el usuario y el nivel de severidad de ésta (*Fig.45*).

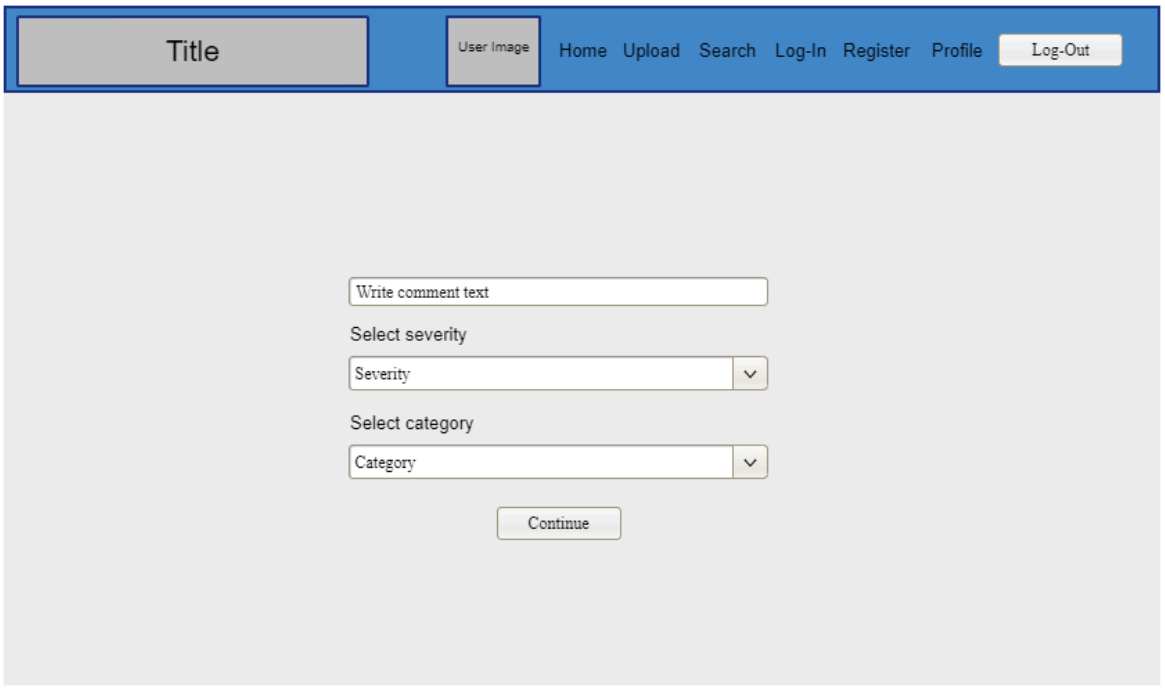

Fig. 45 MockUp final de página de creación de comentario

Si el usuario no aporta estos datos, en la página aparecerá una ventana modal indicando que faltan campos por completar.

No existe versión para las pantallas verticales debido a que no era necesario por la cantidad de información solicitada y la disposición de esta en la idea final de la versión de la pantalla horizontal.

Solo se hizo una exploración sobre la estética de la página, en la que se contenían los elementos dentro de un rectángulo (*Fig.46*) pero se acabó descartando por motivos estéticos.

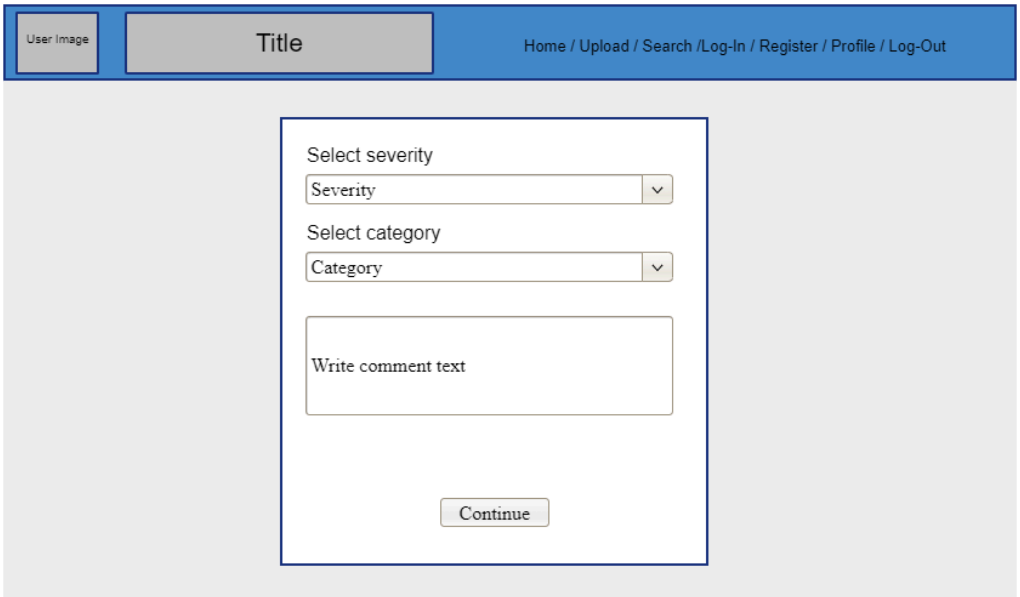

Fig. 46 MockUp de la exploración de estética de la página de creación de comentario

El tutorial de la página explica el propósito de ella y la información que debe se debe aportar para poder completar la operación de agregación.

5.1.2.12 Añadir nueva clasificación

En esta página el usuario añade una nueva clasificación a un videojuego. Si la clasificación a agregar ya existía previamente, no se duplicará, solo se añadirá al videojuego seleccionado. Si la clasificación no existe antes, entonces si se persistirá en la base de datos. Para poder llevar a cabo la operación, el usuario debe aportar la siguiente información (*Fig.47*):

- Nombre de la entidad propietaria de la clasificación
- Nombre de la clasificación en la forma en la que aparece en la etiqueta
- URL de la página principal de la entidad
- País al que pertenece la entidad
- Imagen de la etiqueta de edad

Es la misma información que se solicita cuando se sube un videojuego nuevo.

Si falta el dato en alguno de los campos, aparecerá una ventana emergente para informar al usuario de que la operación ha fallado por falta de información, instándole a que los rellene. Si, por otro lado, la clasificación que se está intentando agregar ya la tiene el videojuego, otra ventana emergente se mostrará para informar al usuario de que ya existe.

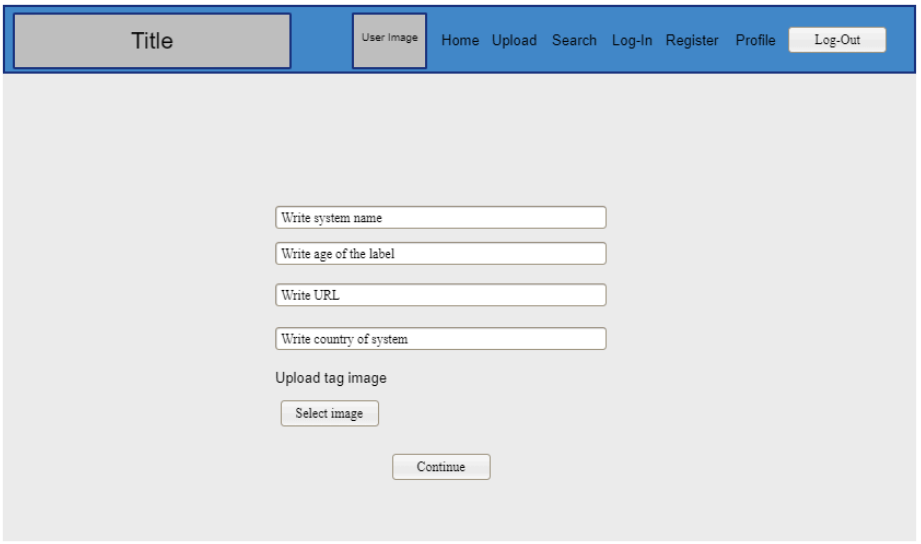

Fig. 47 MockUp final de la página de creación de una nueva clasificación

Similar a la página de generación de comentarios, no se proporciona una versión alternativa para visualización en pantallas verticales. Esto se debe a que la disposición final de los elementos no requería tal adaptación.

Se exploró la idea de cambiar la forma de introducción de información en el campo del nombre de la etiqueta, de modo que fuese un *dropdown* dinámico (*Fig.48*) en función del nombre de la entidad escrita en el campo previo, evitando así errores de escritura por parte del usuario. También se exploró modificar la estética de la página, conteniendo todos los elementos dentro de un contenedor (*Fig.49*), pero se descartó también.

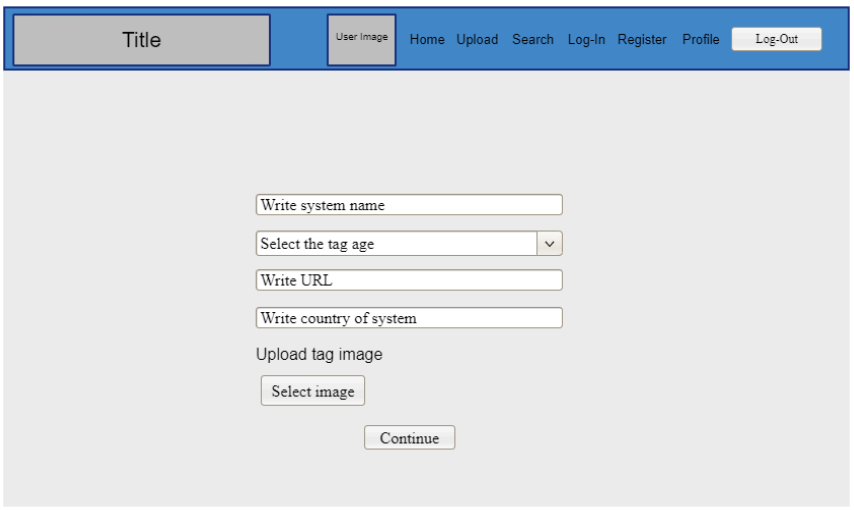

Fig. 48 MockUp de exploración de la forma de especificar el nombre de la etiqueta de edad en la creación de clasificación

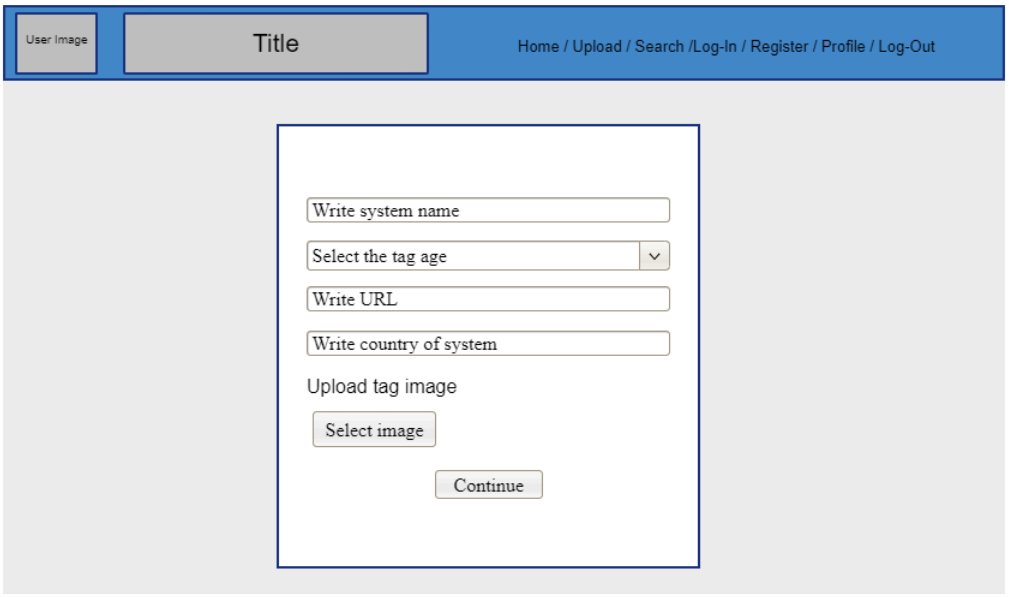

Fig. 49 MockUp de la exploración de la contención de los elementos de la página de creación de clasificación

El tutorial en esta página explica el propósito de ésta, además de explicar el formato en el que se debe de introducir el nombre de la etiqueta de edad.

5.1.2.13 Barra de navegación

La barra de navegación es uno de los componentes principales en las páginas de GameTags. Se encuentra presente en casi todas las páginas web de la plataforma, con excepción del inicio de sesión y del registro de un nuevo usuario. Se divide en tres partes (*Fig.50*):

- La imagen del logo de GameTags
- La imagen de perfil del usuario
- Las páginas accesibles

En función de si el usuario actual ha iniciado sesión o no, la barra de navegación contará con más o menos páginas disponibles. En el caso de que el usuario no haya iniciado sesión las opciones disponibles serán (*Fig.51*):

- Página principal
- Inicio de sesión
- Registrar nuevo usuario
- Buscar videojuegos

Si el usuario ha iniciado sesión, las opciones disponibles serán (*Fig.50*):

- Página principal
- Inicio de sesión
- Registrar nuevo usuario
- Buscar videojuegos
- Perfil del usuario
- Subir un nuevo videojuego
- Cerrar sesión

Además, si el usuario cuenta con una imagen de perfil personalizada, al iniciar sesión esta sustituirá a la imagen de perfil por defecto. La opción de "cerrar sesión" redirige a la página principal una vez se completa el cierre de la sesión actual con éxito.

Fig. 50 MockUp final de la barra de navegación con el usuario habiendo iniciado sesión en modo horizontal

Fig. 51 MockUp final de la barra de navegación sin haber iniciado sesión el usuario en modo horizontal

La versión alternativa para pantallas verticales cambia la forma en la que se visualizan las opciones de la barra de navegación. Son sustituidas por un icono de menú, el cual al ser pulsado despliega debajo de la barra de navegación las opciones disponibles en ese momento. Si se vuelve a pulsar el icono de menú, dicho desplegable se repliega (*Fig.52*). Se muestran las mismas opciones que en la versión para pantallas horizontales si el usuario ha iniciado sesión o no (*Fig.53*).

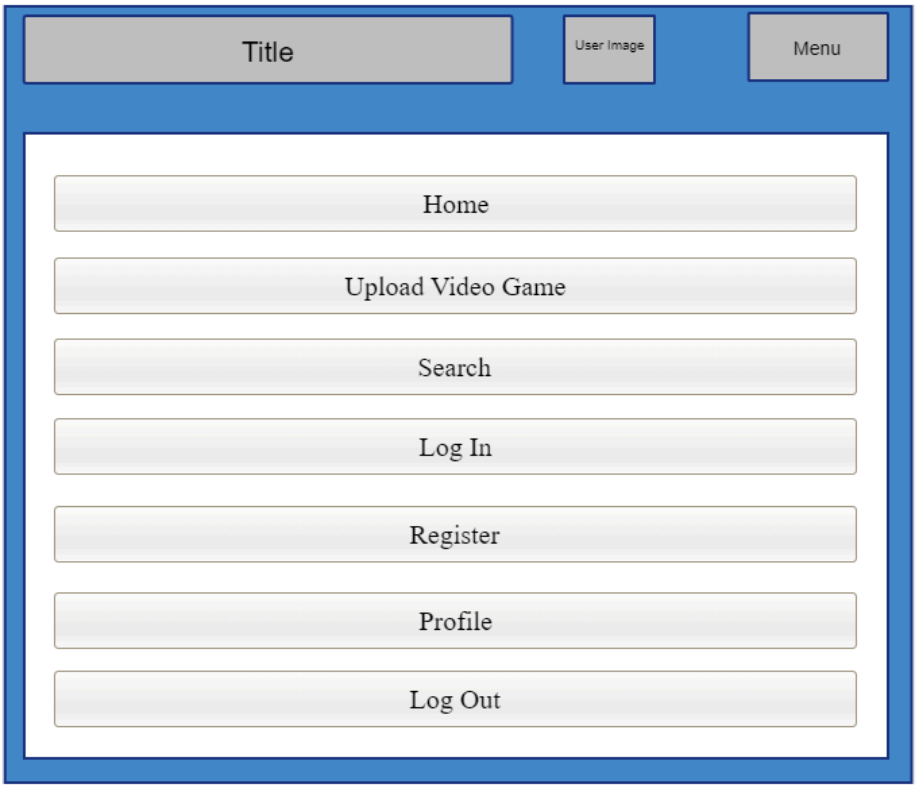

Fig. 52 MockUp final de la barra de navegación con el usuario habiendo iniciado sesión en modo vertical

Fig. 53 MockUp final de la barra de navegación con el usuario sin iniciar sesión en modo vertical

Se contempló para la versión de pantallas verticales hacer que el desplegable no saliera de la barra de navegación, sino que se extendiera desde un lateral de la página (*Fig.54*).

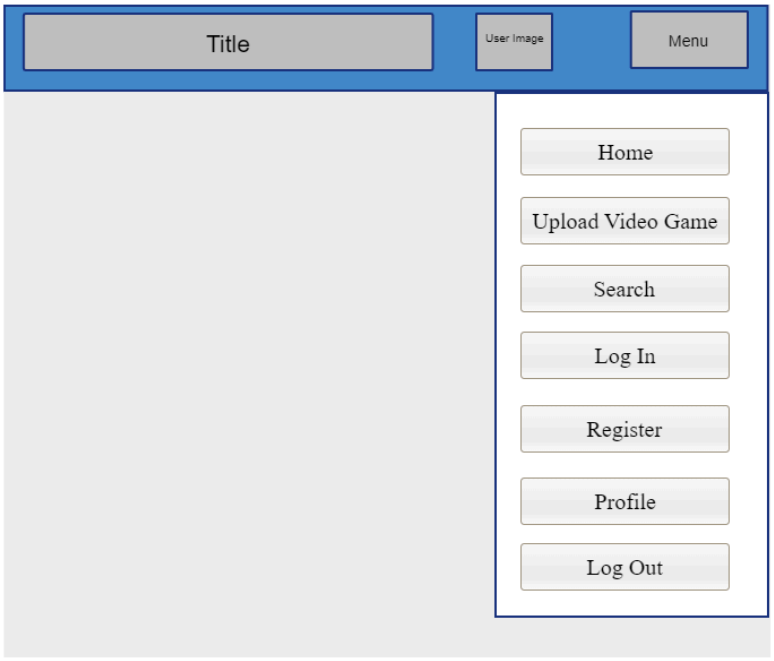

Fig. 54 MockUp de la idea del desplegable lateral en la barra de navegación en modo vertical

Otra idea que se exploró fue cambiar el orden de las secciones en la barra, poniendo primero la imagen de perfil del usuario y cambiando la forma en la que se representaban las opciones disponibles en la barra(*Fig.55*). En la versión para pantallas verticales de esta idea también aparecía el desplegable desde un lateral de la página, pero además dentro aparecía la imagen de perfil del usuario (*Fig.56*).

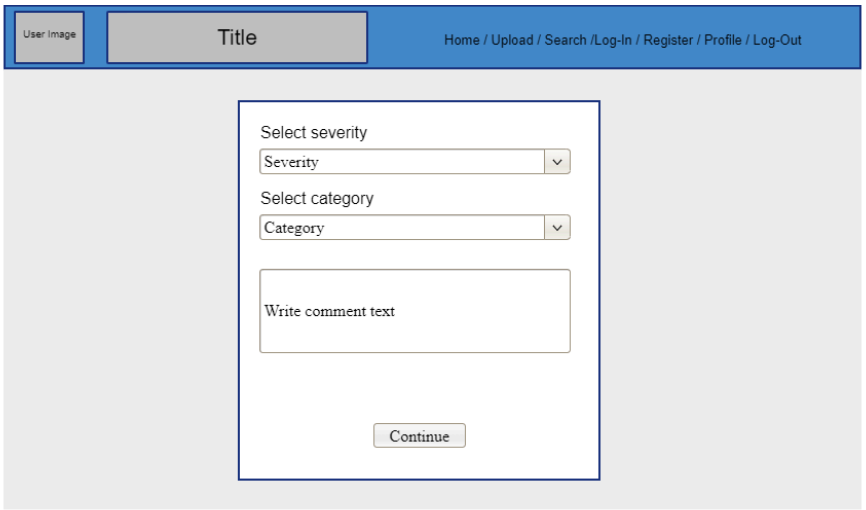

Fig. 55 MockUp de la idea de la reorganización de las partes de la barra de navegación en el modo horizontal

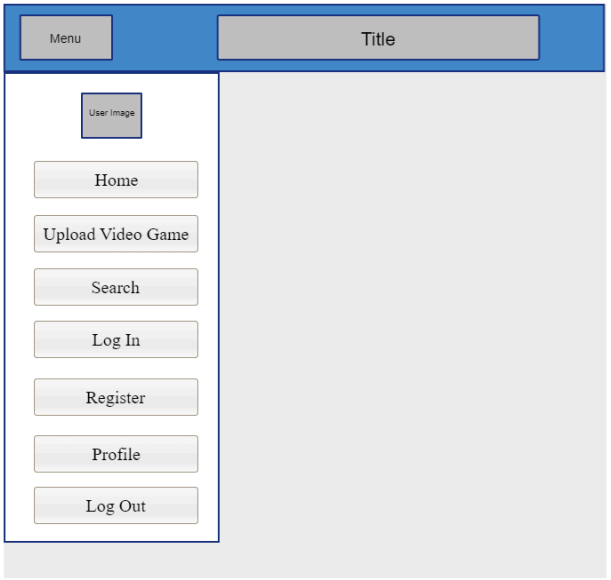

Fig. 56 MockUp de la idea de la imagen del perfil de usuario en el desplegable de la barra de navegación en el modo vertical

5.2 Desarrollo

5.2.1 Desarrollo del Back-End

Como se mencionó en apartados anteriores, el *back-end* se divide en tres subdirectorios, siguiendo el modelo de la arquitectura hexagonal: un directorio para la capa de infraestructura en el que se encuentran todos los archivos de configuración de la comunicación entre el servidor y el cliente y el servidor y la base de datos, además de los archivos que implementan dichas comunicaciones con el cliente (por medio de varias API REST) y con la base de datos (por medio de repositorios de Mongo proporcionados por el *framework* de Spring); un directorio para la capa de aplicación en la que se encuentran las clases que implementan los casos de uso que puede activar el usuario y un directorio para la capa de dominio donde se definen los modelos de datos de la lógica de negocio y los servicios que hacen de puerto entre los casos de uso y los adaptadores de la base de datos. El flujo que seguiría un caso de uso sería el siguiente (*Fig.57*):

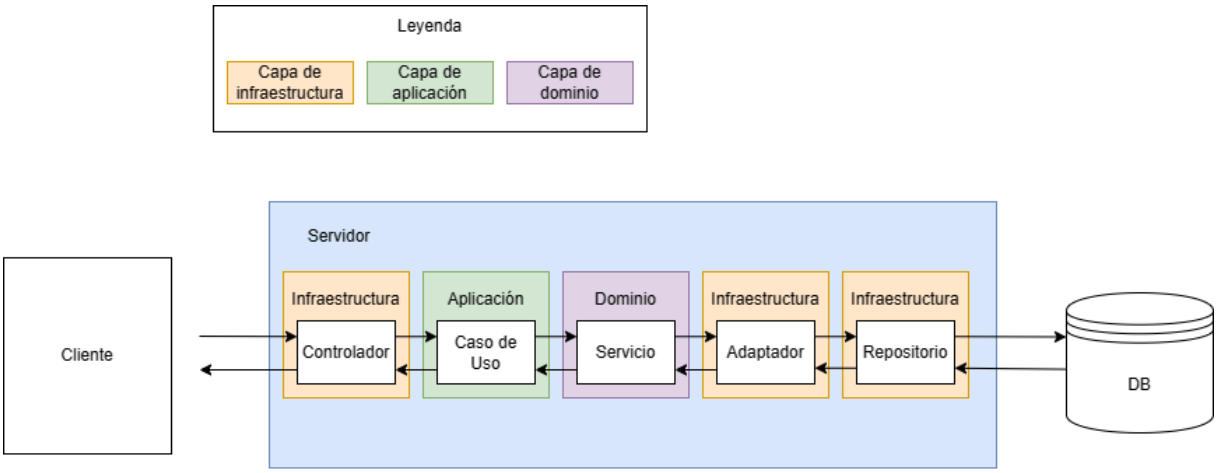

Fig. 57 Diagrama de flujo dentro del servidor de un caso de uso

Para la gestión de dependencias de clases dentro de otras clases y la instanciación de clases al arrancar el servidor se han usado las anotaciones que ofrece el *framework* de Spring. A excepción de los repositorios, todas las clases usan anotaciones de Spring para indicar el propósito general de la clase y para que Spring las construya al arrancar el servidor. Además, algunas de estas anotaciones se han usado para enlazar las clases que tiene dependencias de otras, sin necesidad de construirlas manualmente. Para ello se han marcado los objetos con la anotación *Autowired.*

5.2.1.1 Capa de Infraestructura

Dentro del subdirectorio de esta capa los archivos se han organizado en las siguientes subcarpetas:

- ***adapters***: dentro de esta carpeta hay un archivo por cada una de las entidades que existen en el servidor (*Fig.58*). La funcionalidad de estas clases es la de llamar a la clase del repositorio específica de la entidad para realizar la transacción, usando como argumento la instancia de la entidad transformada a su modelo DAO por medio de un *mapper*. La respuesta que devuelva el repositorio, en modelo DAO, se transmitirá a la clase de servicio específica de la entidad para continuar el flujo, convirtiéndola antes al modelo de datos de dominio. Se usa Lombok para la creación de los constructores.

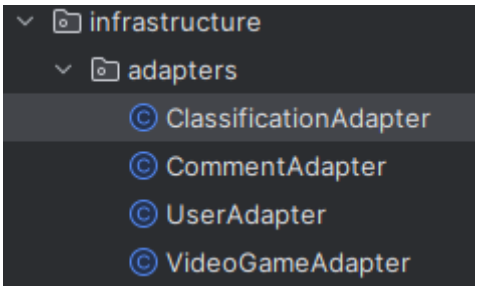

Fig. 58 Adaptadores de las entidades en la capa de infraestructura

- ***controller***: en esta carpeta se encuentran todos los controladores de las peticiones REST de cada una de las entidades, además del controlador de las peticiones de autenticación (*Fig.59*). A estas clases es a donde primero llegan las peticiones desde el cliente y desde ellas se derivan a las clases de los casos de uso, definiendo la ruta web que escuchan para empezar a procesar las peticiones, además del tipo de petición REST que gestionan. No cuentan con ningún tipo de lógica más allá del uso de *mappers* para convertir la información recibida en los objetos de entrada al modelo de datos del dominio, el cual es que usan las clases de casos de uso para realizar las operaciones; y para convertir dicho modelo de datos de dominio al DTO correspondiente y devolvérselo al cliente como respuesta. Asimismo, se especifica aquí la respuesta HTTP que devuelve al cliente en el caso de que la operación haya sido exitosa. Cada una de las peticiones que gestionan se relacionan directamente con un caso de uso de la capa de aplicación.

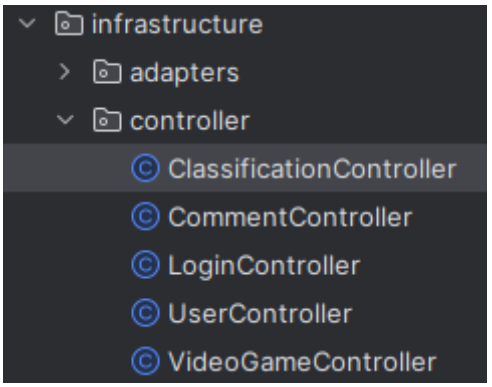

Fig. 59 Clases de controladores en la capa de infraestructura

- ***daos***: aquí se encuentran todos los archivos (*Fig.60*) donde se define el esquema de datos devuelto y recibido por la base de datos cuando se le hace una petición. Usan los mismos campos que tienen los modelos DTO y los modelos de dominio. También se define a qué colección pertenece, para poder agruparse en el repositorio. La definición de los campos se realiza usando Lombok, con lo que no es necesario implementar el constructor, los *getters* y los *setters*.

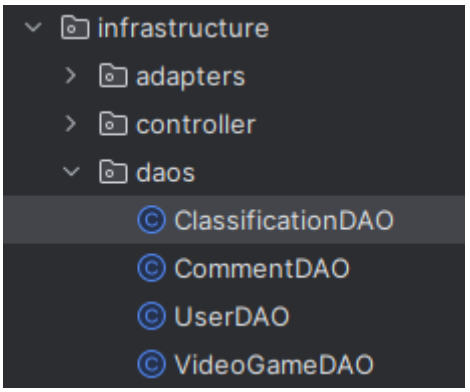

Fig. 60 Clases DAO de cada entidad en la capa de infraestructura

- ***dtos***: aquí se encuentran todos los archivos (*Fig.61*) donde se define el esquema de datos devuelto al cliente como respuesta a su petición o el esquema de datos que recibe el servidor en algunas peticiones. Estos objetos cuando se devuelven van dentro de entidades de respuesta HTTP creados por los controladores. La definición de los campos se realiza usando Lombok, con lo que no es necesario implementar el constructor, los *getters* y los *setters*, a excepción del DTO de autenticación donde sí se implementa manualmente el constructor.

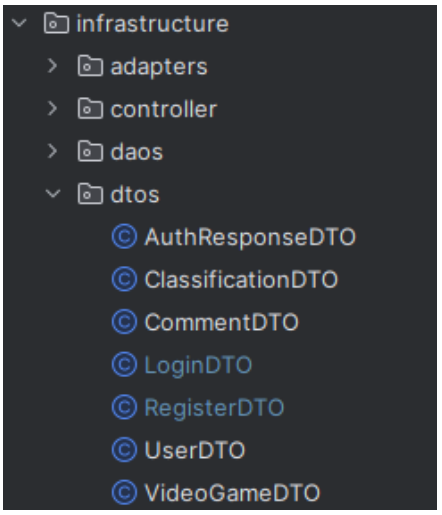

Fig. 61 Clases DTO de las entidades en la capa de infraestructura

- ***mappers***: aquí se almacenan todas las clases encargadas de hacer la conversión entre modelos de datos (de dominio a DAO, de dominio a DTO, etc) (*Fig.62*). Para no tener que implementar los constructores de estas clases, se usa Lombok. Para cada tipo de conversión usado en el servidor, existe una función. Si el objeto a convertir contiene objetos de otras entidades, el conversor llamará al conversor concreto de dicha entidad para convertir también ese objeto anidado. En el caso específico de las

funciones de conversión en la que están involucradas contraseñas (por ejemplo, cuando se registra un nuevo usuario), el conversor además cifrará la contraseña usando una clase de cifrado de contraseñas provisionada por Spring.

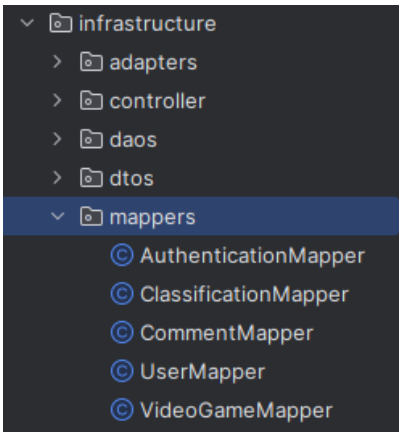

Fig. 62 Clases de conversión de modelos de datos

- ***repositories***: aquí se encuentran todas las interfaces entre el servidor y la base de datos usadas en todas las transacciones (*Fig.63*). Cada uno de los repositorios extiende de la clase *MongoRepository*, proporcionada por el *framework* de Spring, para poder indicar el modelo de datos con el que se realizan las transacciones y el tipo de repositorio que es. Se aprovecha la funcionalidad de los *Query Methods* [31] proporcionada por Spring Data (un módulo de Spring) para escribir las peticiones a la base de datos sin tener que usar el lenguaje NoSQL, solo aplicando una sintaxis a la hora de escribir el nombre del método. Además, no se han implementado operaciones fundamentales como guardar o actualizar, ya que estas funciones ya están incorporadas de manera inherente en el repositorio.

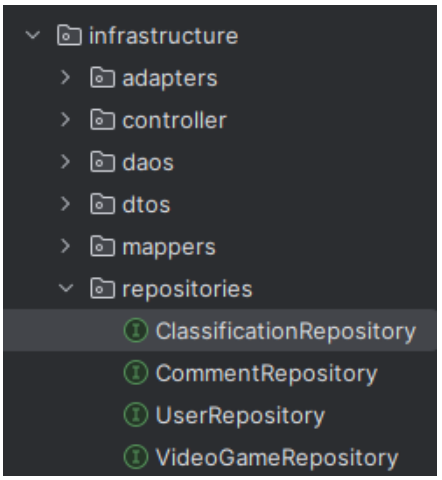

Fig. 63 Interfaces de repositorios dentro de la capa de infraestructura

Los archivos de configuración de Spring Security, Mongo, CORS [32], y la generación de *tokens* JWT no se encuentran dentro de ninguna subcarpeta de configuración debido a que el proyecto no era capaz de encontrar las configuraciones.

La clase de configuración de CORS (*CorsConfig*) (*Fig.64*) se encarga de configurar la comunicación con el cliente para evitar problemas de legitimación en las peticiones enviadas desde el cliente en el otro contenedor de Docker. Se han especificado los orígenes de petición, los tipos de petición REST y las cabeceras permitidas.

En la clase de configuración de seguridad de Spring (*SecurityConfig*) (*Fig.64*) se instancia el *AuthenticationManager* para autenticar a los usuarios por *token*, además del filtro del *token* JWT. Otra configuración realizada en esta clase es la de los filtros de seguridad antes de empezar a procesar una petición, especificando: las autoridades necesarias para acceder a los *endpoints*, el punto de entrada de autenticación, el tipo de sesión (*Stateless* por usar JWT), la habilitación de CORS, la inhabilitación de CSRF y la agregación del filtro de autenticación JWT. Además, junto con esta clase, también se ha creado una clase aparte (*SecurityConstants*) (*Fig.64*) para especificar los valores del secreto del JWT y el tipo de expiración del *token*.

La clase de configuración de Mongo (*MongoConfig*)(*Fig.64*) se usa para indicar el *endpoint* de conexión a la base de datos, la configuración que va a usar el cliente de Mongo en el servidor y el nombre de la base de datos.

Hay cuatro clases de configuración de JWT. Una de las clases (*JWTGenerator*) (*Fig.64*) se encarga de generar el *token* e implementar los métodos para validarlo y otras funcionalidades necesarias para su gestión. En otra de las clases (*JWTAuthenticationFilter*)(*Fig.64*) se implementa el filtro agregado a la cadena de filtros de seguridad en la configuración de Spring Security. En otra clase de este grupo (*CustomUserDetailsService*)(*Fig.64*) se implementa el servicio de petición de información del usuario que es usado en el filtro de JWT cuando se va a autenticar a dicho usuario. En la última clase (*JWTAuthEntryPoint*) (*Fig.64*) se define el punto de entrada de la autenticación por JWT.

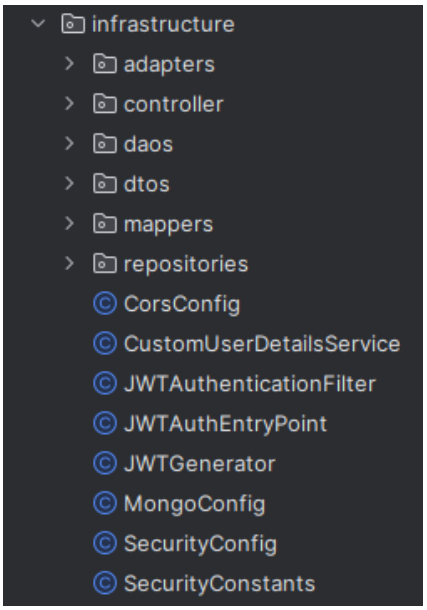

Fig. 64 Clases de configuración del servidor en la capa de infraestructura

5.2.1.2 Capa de Aplicación

En el subdirectorio, los casos de uso se han agrupado en carpetas en función de la entidad con la que trabajan. Las carpetas que existen son:

- ***authentication***: los casos de uso de esta carpeta son el caso de uso para iniciar sesión (*LoginUserUseCase*) y el caso de uso para registrar un nuevo usuario en la plataforma (*RegisterUserUseCase*) (*Fig.65*). A continuación, se describe el funcionamiento de cada caso de uso:
  - ***LoginUserUseCase***: recibe como parámetro de entrada las credenciales del usuario (nombre de usuario y contraseña). Consulta al servicio de usuarios si existe un usuario con el nombre pasado como argumento. Si existe el usuario, crea un objeto de autenticación usando dichas credenciales y especificando el rol del usuario, estableciendo dicho *token* en el contexto de seguridad. Tras esto, crea el *token* con dicho objeto de autenticación y devuelve un DTO como respuesta. Si el usuario no existe en la base de datos, lanza una excepción.
  - ***RegisterUserUseCase***: recibe como parámetro de entrada el nombre de usuario, la contraseña de este, el país al que pertenece y un correo electrónico de contacto. Consulta al servicio de usuarios si existe un usuario con el mismo nombre. Si ya existe, lanza una excepción indicando que el usuario ya existe. Si no existe, vuelve a llamar al servicio para crear un nuevo usuario con los parámetros de entrada y lo devuelve como respuesta al controlador.

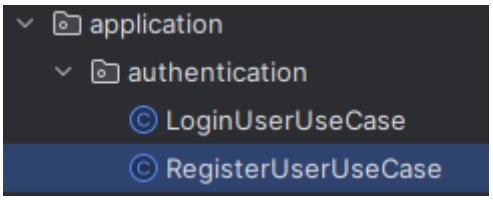

Fig. 65 Casos de uso de la autenticación en la capa de aplicación

- ***classification***: los casos de uso que se alojan dentro de esta carpeta son el caso de uso de borrado de una clasificación (*DeleteClassificationUseCase*), el caso de uso de búsqueda de todas las clasificaciones (*FindAllClassificationsUseCase*), el caso de uso para buscar una clasificación por su ID (*FindClassificationByIdUseCase*), el caso de uso para actualizar una clasificación ya existente (*UpdateClassificationUseCase*), el caso de uso para crear una nueva clasificación (*CreateClassificationUseCase*) y el caso de uso para añadir la imagen de la etiqueta a una clasificación (*AddTagImageUseCase*) (*Fig.66*). A continuación, se describe el funcionamiento de cada uno:
  - ***CreateClassificationUseCase****:* recibe como parámetro de entrada la clasificación a crear, el cual ha sido convertida a partir del objeto de entrada (*CreateClassificationInput*) en el controlador. Primero consulta al servicio de las clasificaciones para comprobar si ya existe la clasificación. Si ya existe, devuelve una excepción con el mensaje de que ya existe. Si no existe, vuelve a llamar al servicio para crearla y la devuelve como respuesta.
  - ***DeleteClassificationUseCase****:* recibe como parámetro de entrada el ID de la clasificación a borrar. Llama al servicio de las clasificaciones para comprobar si existe la clasificación con ese ID. Si existe, la borra y la devuelve como respuesta. Si no existe, lanza una excepción indicando que no existe tal clasificación. El caso de uso se ha usado con el propósito de hacer *debug*.
  - ***UpdateClassificationUseCase****:* recibe como parámetro de entrada la clasificación con los campos a actualizar ya actualizados. Se llama al servicio de clasificaciones para comprobar si existe la clasificación a actualizar. Si no existe, se lanza una excepción indicando que no existe la clasificación. Si existe, se vuelve a llamar al servicio para actualizar la entrada de la entidad en la base de datos y se devuelve como respuesta dicha entrada actualizada.
  - ***FindAllClassificationsUseCase****:* no recibe ningún parámetro de entrada. Llama al servicio de clasificaciones para buscar todas las clasificaciones almacenadas en la base de datos y devuelve como respuesta la lista de clasificaciones que recibe del servicio. El caso de uso se ha usado con el propósito de hacer *debug*.
  - ***FindClassificationByIdUseCase****:* recibe como parámetro de entrada el ID de la clasificación que se quiere buscar. Se llama al servicio de clasificaciones para buscar la clasificación que tenga el mismo ID. Si no encuentra ninguna clasificación, lanza una excepción indicando que no la encontró. Si la encuentra, la devuelve como respuesta.

- ***AddTagImageUseCase**:* recibe como parámetros de entrada el ID de la clasificación a la que se quiere añadir la imagen, el ID del videojuego para el que se había creado al principio la clasificación y el archivo de la propia imagen de la etiqueta. Primero llama al servicio de clasificaciones para buscar la clasificación a la que añadir la imagen. Si no encuentra la clasificación, para la ejecución y devuelve la clasificación creada por defecto por el servicio. Si se localiza, el archivo de la imagen se convierte al formato con el que se almacenará en la base de datos. El resultado de esta transformación se guarda en la entidad de la clasificación previamente encontrada. De esta manera, el objeto de la clasificación está preparado para ser cargado en la base de datos y actualizar la entrada que ya existe en ella. Luego, se llama al servicio de videojuegos para buscar el videojuego que debe actualizarse con la clasificación ya actualizada. Si no encuentra el videojuego, devuelve la clasificación como respuesta. Si encuentra el videojuego, coje la lista de clasificaciones que almacena el videojuego, busca la clasificación actualizada previamente y si la encuentra, le añade la imagen a la instancia de la lista. Tras esto llama al servicio de los videojuegos para actualizar el videojuego y llama al servicio de las clasificaciones para actualizar la clasificación en la base de datos, devolviendo al controlador la respuesta del servicio

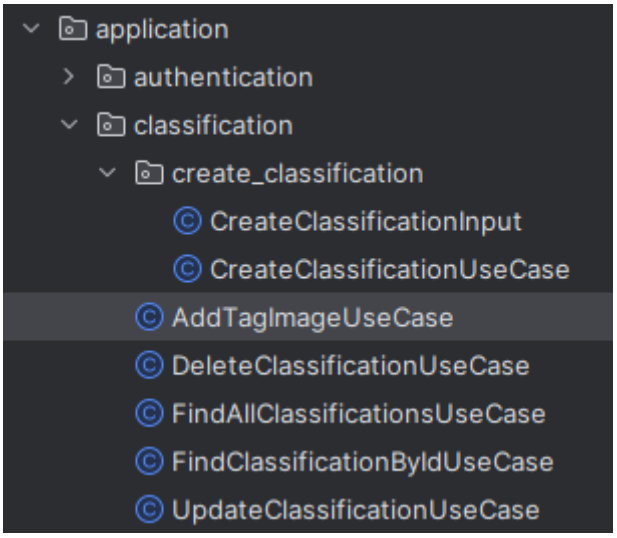

Fig. 66 Casos de uso de las clasificaciones en la capa de aplicación

- ***comment***: los casos de uso que se almacenan dentro de esta carpeta son el caso de uso para crear un comentario (*CreateCommentUseCase*), los casos de uso para filtrar los comentarios por su categoría (*FilterByCategoryUseCase*), severidad (*FilterBySeverityUseCase*) y el videojuego al que pertenecen (*FilterByVideoGameUseCase*), el caso de uso para borrar un comentario (*DeleteCommentUseCase*), el caso de uso para buscar todos los comentarios que hay en la plataforma (*FindAllCommentsUseCase*), el caso de uso para buscar un comentario concreto por su ID (*FindCommentByIdUseCase*) y el caso de uso para

actualizar un comentario (*UpdateCommentUseCase*) (*Fig.67*). A continuación, se describe el funcionamiento de cada uno:

- ***CreateCommentUseCase***: recibe como parámetro de entrada el comentario a crear. Primero se llama al servicio de usuarios para comprobar si el usuario que pretende crear el comentario existe. Además, también se llama al servicio de videojuegos para comprobar que también existe el videojuego. Si alguno de ellos no existe, salta una excepción indicando el error. Si ambos existen, se llama al servicio de comentarios para añadir el comentario a la base de datos. Después se agrega el comentario a la lista de comentarios del videojuego y se actualiza el videojuego en base de datos llamando al servicio de videojuegos. Finalmente, se devuelve el comentario creado como respuesta.
- ***FilterByCategoryUseCase***: hay dos casos de uso dentro de esta clase. Uno de ellos para filtrar los comentarios del usuario por categoría y otro para filtrar los comentarios del videojuego por categoría. El primer caso recibe como parámetro de entrada la categoría por la que filtrar. Al principio se coge el nombre de usuario del contexto de seguridad. Tras esto, se llama al servicio de usuarios para comprobar que exista el usuario con ese nombre en la base de datos. Si no existe, salta una excepción indicando que el usuario no existe. Si existe, llama al servicio de comentarios para buscar todos los comentarios del usuario que sean de la categoría deseada. Del servicio recibe una lista de comentarios que devuelve al controlador como respuesta. El segundo caso recibe el nombre del videojuego y la categoría por la que filtrar. Se llama directamente al servicio de comentarios pasando estos dos argumentos y devuelve al controlador la lista de comentarios que le devuelva el servicio. El segundo caso se ha usado para hacer *debug*.
- ***FilterBySeverityUseCase***: hay dos casos de uso dentro de esta clase. Uno de ellos para filtrar los comentarios del usuario por la severidad y otro para filtrar los comentarios del videojuego por la severidad. El primer caso recibe como parámetro de entrada la severidad por la que filtrar. Primero se coge el nombre del usuario del contexto de seguridad y se llama al servicio de usuarios para comprobar si existe el usuario con ese nombre en la base de datos. Si no existe, se lanza una excepción indicando la ausencia. Si existe, se llama al servicio de comentarios para buscar los comentarios del usuario que concuerden con el filtro. La lista de comentarios que reciba como respuesta del servicio la devolverá al controlador. El segundo caso recibe el nombre del videojuego y la severidad por la que filtrar. Se llama directamente al servicio de comentarios pasando estos dos argumentos y devuelve al controlador la lista de comentarios que le devuelva el servicio. El segundo caso se ha usado para hacer *debug*.
- ***FilterByVideoGameUseCase***: recibe como parámetro de entrada el nombre del videojuego del que se quieren buscar los comentarios. Primero se coge el nombre de usuario que está en el contexto de seguridad para después comprobar si existe o no en la base de datos el usuario con ese nombre a través del servicio de usuarios. Además, se llama al servicio de videojuegos

para comprobar que el videojuego exista. Si alguno de ellos no existe, se lanza una excepción indicando la situación. Si existen los dos, se llama al servicio de comentarios, pasando el nombre del videojuego y del usuario, para buscar los comentarios que cumplan el filtro. La lista de comentarios que devuelve el servicio se envía como respuesta al controlador.

- ***DeleteCommentUseCase***: recibe como parámetro de entrada el ID del comentario a borrar. Se llama al servicio de comentarios para comprobar si existe un comentario con ese ID en base de datos. Si no existe, se lanza una excepción indicando que no existe. Si existe, se vuelve a llamar al servicio para borrar el comentario que coincida con el ID. Se devuelve como respuesta al controlador el comentario borrado. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***FindAllCommentsUseCase***: en esta clase hay dos casos de uso: uno para buscar todos los comentarios en la base de datos y otro para buscar todos los comentarios del usuario. El primer caso de uso no recibe ningún parámetro de entrada. Llama directamente al servicio de comentarios y la lista de comentarios que recibe como respuesta la devuelve al controlador. El segundo caso de uso recibe como parámetro de entrada el nombre del usuario del que se quiere recuperar los comentarios. Se llama al servicio de comentarios pasando el nombre como parámetro y la lista de comentarios que devuelva el servicio como respuesta se devuelve al controlador. El primer caso de uso se ha usado con el propósito de hacer *debug*.
- ***FindCommentByIdUseCase***: el caso de uso recibe como parámetro de entrada el ID del comentario a buscar. Primero llama al servicio de comentarios para buscar el comentario que coincida con el ID. Si devuelve un comentario, este se devuelve al controlador como respuesta. Si no se devuelve ningún comentario, se lanza una excepción indicando que el comentario no fue encontrado. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***UpdateCommentUseCase***: el caso de uso recibe como parámetro de entrada el comentario ya actualizado que se quiere sustituir. Se llama al servicio de comentarios para comprobar si el comentario existe en la base de datos. Si no existe, se lanza una excepción indicando que no existe. Si existe, se vuelve a llamar al servicio para actualizar el comentario. Se devuelve como respuesta al controlador la respuesta que dé el servicio, que será el comentario ya actualizado en base de datos. El caso de uso se ha usado con el propósito de hacer *debug*

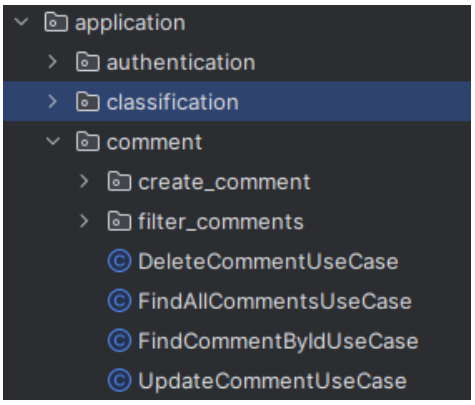

Fig. 67 Casos de uso de los comentarios en la capa de aplicación

- ***user***: los casos de uso de esta carpeta son el caso de creación del usuario (*CreateUserUseCase*), el caso de uso para actualizar la contraseña del usuario (*UpdatePasswordUseCase*), el caso de uso para actualizar el nombre de usuario del usuario (*UpdateUsernameUseCase*), el caso de uso para añadir una imagen de perfil al usuario (*AddProfileImageUseCase*), el caso de uso para borrar un usuario (*DeleteUserUseCase*), el caso de uso para buscar a todos los usuarios que existen en la plataforma (*FindAllUsersUseCase*), el caso de uso para buscar a un usuario concreto por medio de su ID (*FindUserByIdUseCase*), el caso de uso para buscar a un usuario por su nombre (*FindUserByNameUseCase*), y el caso de uso para actualizar un usuario (*UpdateUserUseCase*) (*Fig.68*). A continuación, se describe el funcionamiento de cada uno:
  - ***CreateUserUseCase***: recibe como parámetro de entrada el usuario que se quiere crear. Primero se llama al servicio de usuarios para comprobar si ya existe un usuario con el mismo nombre que el que se pretende crear. Si no existe, se vuelve a llamar al servicio para crear el usuario en base de datos y se devuelve al controlador el usuario creado. Si ya existe, se devuelve directamente el usuario ya existente. El caso de uso se ha usado con el propósito de hacer *debug*.
  - ***UpdatePasswordUseCase***: se recibe como parámetro de entrada la contraseña nueva y la contraseña antigua. Primero se recoge el nombre del usuario del contexto de seguridad y se llama al servicio de usuarios para obtener todos los datos del usuario que tenga ese nombre. Después se cifra la nueva contraseña y se crea un nuevo usuario usando los datos del que ya existen, salvo la contraseña, donde se usa la nueva contraseña ya cifrada. Si la contraseña antigua proporcionada como argumento coincide con la contraseña almacenada del usuario recuperado de la base de datos, se procede a actualizar el usuario en la misma. Esto se realiza invocando al servicio de usuarios y proporcionando el usuario actualizado. Simultáneamente, se actualiza el contexto de seguridad con el usuario modificado y se genera un nuevo *token*. Como resultado, se devuelve un DTO

de autenticación que incluye el nuevo *token* y el nombre del usuario. Si las contraseñas no coinciden, se lanza una excepción indicando que la contraseña es incorrecta.

- ***UpdateUsernameUseCase***: este caso recibe como parámetro de entrada el nuevo nombre de usuario, la contraseña que usa actualmente el usuario y un objeto *Principal* donde se almacenan los datos de la sesión del usuario. Primero se recupera el nombre de usuario del objeto *Principal* para después llamar al servicio de usuarios y recuperar los datos de dicho usuario por medio de su nombre. A continuación, se crea un usuario usando los datos recuperados antes por el servicio, a excepción del nombre de usuario, donde se usa el pasado en la entrada del caso. Tras esto, se comprueba si la contraseña pasada en la entrada coincide con la contraseña del usuario almacenada en la base de datos. Si no coinciden, lanza una excepción indicando que la contraseña proporcionada no es correcta. Si coinciden, se actualiza el nombre de usuario en otros objetos en los que se estuviese usando el nombre antiguo (comentarios y videojuegos creados por el usuario) y se llama al servicio de usuarios para actualizar al usuario en base de datos. Después, se actualiza al usuario del contexto de seguridad, creando un nuevo *token* con el nombre de usuario actualizado y se devuelve un DTO de autenticación con el nuevo *token* y nombre al controlador.
- ***AddProfileImageUseCase***: recibe como parámetro de entrada la imagen de perfil que se quiere usar y el ID del usuario al que se quiere añadir la imagen. Primero se llama al servicio de usuarios para comprobar que exista el usuario. Si no existe, se devuelve al controlador un usuario por defecto como respuesta. Si existe el usuario, se transforma el archivo de la imagen al formato con el que se guardan en base de datos. Después se guarda la imagen transformada en el usuario y se vuelve a llamar al servicio de usuarios para actualizar el usuario en base de datos, pasando como argumento al usuario con la imagen. El servicio devuelve como respuesta al usuario actualizado en base de datos y este usuario se devuelve al controlador como respuesta.
- ***DeleteUserUseCase***: recibe como parámetro de entrada el ID del usuario a borrar. Se llama al servicio de usuarios para comprobar si existe el usuario que se quiere borrar. Si no existe, se lanza una excepción indicando que no existe. Si existe el usuario, se vuelve a llamar al servicio de usuarios para borrar al usuario que tenga el ID. La respuesta que devuelve el servicio (el usuario borrado) se manda como respuesta al controlador. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***FindAllUsersUseCase***: el caso de uso no recibe ningún parámetro de entrada. Llama al servicio de usuarios para recuperar una lista de todos los usuarios presentes en la base de datos y devuelve dicha lista al controlador como respuesta. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***FindUserByIdUseCase***: recibe como parámetro de entrada el ID del usuario que se quiere buscar. Se llama al servicio de usuarios para buscar al usuario que concuerde con el ID pasado en la entrada. Si existe el usuario, este se

devuelve al controlador como respuesta. Si no, se lanza una excepción indicando que el usuario no existe.

- ***FindUserByNameUseCase***: recibe como entrada el nombre del usuario que se quiere buscar. Primero se llama al servicio de usuarios para buscar al usuario que tenga ese nombre. Si lo encuentra, lo devuelve al controlador como respuesta. Si no, lanza una excepción indicando que no lo encontró. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***UpdateUserUseCase***: recibe como parámetro de entrada el usuario a actualizar con los datos ya actualizados. Llama al servicio de usuarios para comprobar si el usuario existe en base de datos. Si no existe, lanza una excepción indicando que no lo encontró. Si lo encuentra, se vuelve a llamar al servicio de usuarios para actualizar el usuario en base de datos pasando el usuario recibido como entrada. El servicio devolverá el usuario ya actualizado y este se devolverá como respuesta al controlador.

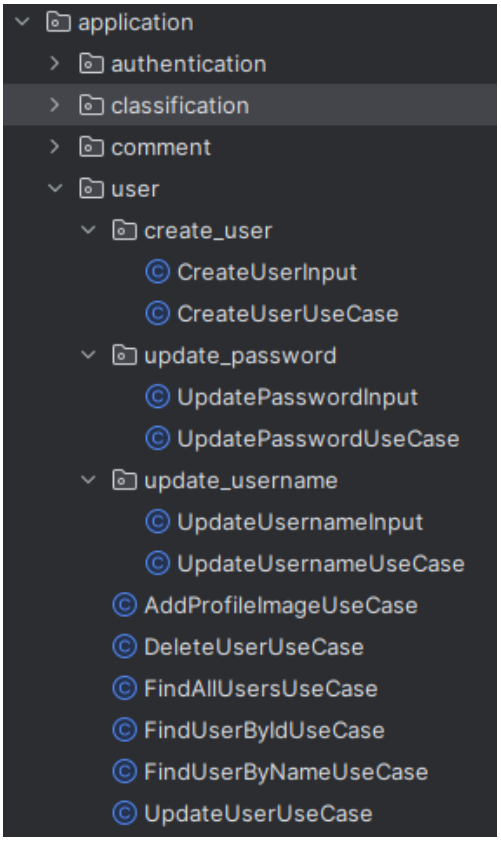

Fig. 68 Casos de uso del usuario en la capa de aplicación

- ***videogame***: en esta carpeta se encuentran el caso de uso de añadir una nueva clasificación a un videojuego existente (*AddNewClassificationUseCase*), el caso de uso de crear un nuevo videojuego (*CreateVideoGameUseCase*), los casos de uso del filtrado de videojuegos por el desarrollador (*FilterByDeveloperUseCase*), la plataforma (*FilterByPlatformsUseCase*), la entidad clasificadora

(*FilterBySystemUseCase*) y la etiqueta de edad (*FilterByTagUseCase*); el caso de uso de añadir la portada al videojuego (*AddImageUseCase*), el caso de uso para borrar un videojuego (*DeleteVideoGameUseCase*), el caso de uso para buscar todos los videojuegos de la base de datos (*FindAllVideoGamesUseCase*), el caso de uso para buscar un videojuego por su ID (*FindVideoGameByIdUseCase*), el caso de uso para buscar un videojuego por su nombre (*FindVideoGameByNameUseCase*), el caso de uso para buscar los últimos videojuegos añadidos a la plataforma (*LatestVideoGamesUseCase*) y el caso de uso para actualizar un videojuego ya existente (*UpdateVideoGameUseCase*) (*Fig.69*). A continuación, se describe el funcionamiento de cada uno:

- ***AddNewClassificationUseCase***: recibe como parámetro de entrada la clasificación nueva a añadir al videojuego y el nombre del videojuego al que se le quiere agregar la clasificación. Primero se llama al servicio de videojuegos para comprobar si existe un videojuego con el mismo nombre pasado en la entrada del caso. Si no se encuentra, se lanza una excepción indicando que no se encontró el videojuego. Si existe, entonces se comprueba si la clasificación a agregar ya existe o no en la base de datos. Si no existe, persistirá en ella. Si ya existe, se usa la clasificación almacenada en base de datos en vez de la recibida en la entrada del caso de uso. Después, si la clasificación a agregar no existe ya dentro del videojuego (para evitar duplicados), se añade. De no ser así, no se hace nada. Finalmente, se vuelve a llamar al servicio de videojuegos para actualizar el videojuego. Devolverá como respuesta el videojuego ya actualizado en base de datos y este videojuego se devolverá al controlador como respuesta.
- ***CreateVideoGameUseCase***: se recibe como parámetro de entrada el videojuego que se quiere crear. Primero se llama al servicio para comprobar si el videojuego que se quiere añadir ya existe. Si ya existe, este se devuelve como respuesta y termina el caso de uso. Si no existe, se procede a intentar añadir a la base de datos la clasificación con la que se ha creado el videojuego y actualizar la lista de clasificaciones. Si dicha clasificación no existe en base de datos, se agrega. Si existe, se usa la clasificación que está en base de datos en vez de la nueva con la que se creó el videojuego. Tras esto, se vuelve a llamar al servicio de videojuegos para agregar el videojuego a la base de datos. El servicio devolverá el videojuego ya agregado a la base de datos y este se devolverá al controlador como respuesta.
- ***FilterByDeveloperUseCase***: recibe como parámetro de entrada el nombre del desarrollador de videojuegos por el que se quiere filtrar en la búsqueda. Se llama al servicio de videojuegos para hacer la búsqueda usando el filtro pasado. El servicio devolverá una lista de videojuegos que cumplan el filtro y esta lista se devolverá al controlador como respuesta.
- ***FilterByPlatformsUseCase***: recibe como parámetro de entrada la lista de plataformas por las que se quiere filtrar la búsqueda. Se llama al servicio de videojuegos para hacer la búsqueda usando la lista de filtro pasada. El servicio devolverá una lista de videojuegos que cumplan el filtro y esta lista se devolverá al controlador como respuesta.

- ***FilterBySystemUseCase***: recibe como parámetro de entrada el nombre del sistema de clasificación por el que se quiere filtrar en la búsqueda. Se llama al servicio de videojuegos para hacer la búsqueda usando el filtro pasado. El servicio devolverá una lista de videojuegos que cumplan el filtro y esta lista se devolverá al controlador como respuesta.
- ***FilterByTagUseCase***: se recibe como parámetro de entrada la edad por la que se quiere filtrar la búsqueda de videojuegos. Según el valor de la edad a buscar, se escoge una lista de las edades equivalentes a esta en todos los sistemas de edad soportados por GameTags. Si el valor de la edad no es reconocido, termina el caso de uso. Una vez escogida una lista de edades equivalentes, se llama al servicio de videojuegos para buscar los videojuegos que usen alguna de las edades contenidas en la lista. La respuesta del servicio será una lista de videojuegos que contengan alguna de esas edades y la lista será devuelta al controlador como respuesta.
- ***AddImageUseCase***: se recibe como argumentos de entrada el archivo de la imagen de la carátula del videojuego y el ID del videojuego al que se quiere agregar la imagen. Primero se llama al servicio de videojuegos para comprobar que el videojuego con el mismo ID existe en la base de datos. Si no existe, se devuelve un objeto de videojuego vacío. Si existe, se transforma el archivo de imagen al formato con el que se guardará en base de datos. Después de esto, se guarda la imagen transformada en el videojuego y se actualiza el videojuego llamando al servicio y pasándole como argumento el videojuego con la imagen ya agregada. El servicio devolverá el videojuego ya actualizado y este será devuelto como respuesta al controlador.
- ***DeleteVideoGameUseCase***: se recibe como argumento de entrada el ID del videojuego a borrar. Se llama al servicio de videojuegos para comprobar si existe el videojuego. Si existe, se vuelve a llamar al servicio para borrar el videojuego al que corresponda el ID, devolviendo al controlador el videojuego borrado. Si no existe, lanza una excepción indicando que el videojuego a borrar no existe. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***FindAllVideoGamesUseCase***: no recibe ningún parámetro de entrada. Se llama al servicio de videojuegos para recuperar todos los videojuegos de la base de datos. La lista de videojuegos que devuelve el servicio se envía al controlador como respuesta. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***FindVideoGameByIdUseCase***: recibe como parámetro de entrada el ID del videojuego a buscar. Se llama al servicio de videojuegos para buscar el videojuego que use el ID pasado como argumento. Si lo encuentra, lo devuelve como respuesta al controlador. Si no, lanza una excepción indicando que no se encontró el videojuego. El caso de uso se ha usado con el propósito de hacer *debug*.
- ***FindVideoGameByNameUseCase***: recibe como parámetro de entrada el nombre del videojuego que se quiere buscar. Primero se llama al servicio de videojuegos para buscar los videojuegos que tengan el mismo nombre o contengan el nombre que se ha pasado en la entrada. Si la lista devuelta por

el servicio está vacía, se lanza una excepción indicando que no se ha encontrado ningún videojuego que cumpla la condición. Si la lista no está vacía, se devuelve al controlador como respuesta.

- ***LatestVideoGamesUseCase***: no recibe ningún parámetro de entrada. Se llama al servicio de videojuegos para recuperar los tres últimos videojuegos agregados a la base de datos. La lista de videojuegos que devuelva el servicio será devuelta al controlador como respuesta.
- ***UpdateVideoGameUseCase***: recibe como argumento de entrada el videojuego a actualizar con los datos ya actualizados. Primero se usa el servicio de videojuegos para comprobar que el videojuego a actualizar existe en la base de datos. Si no existe, se lanza una excepción indicando que el videojuego no existe. Si existe, se actualiza la lista de comentarios del videojuego por si se ha añadido un nuevo comentario, agregando aquellos nuevos comentarios. Tras esto, se vuelve a llamar al servicio de videojuegos para actualizar el videojuego. El servicio devolverá el videojuego actualizado en la base de datos y este será devuelto al controlador como respuesta.

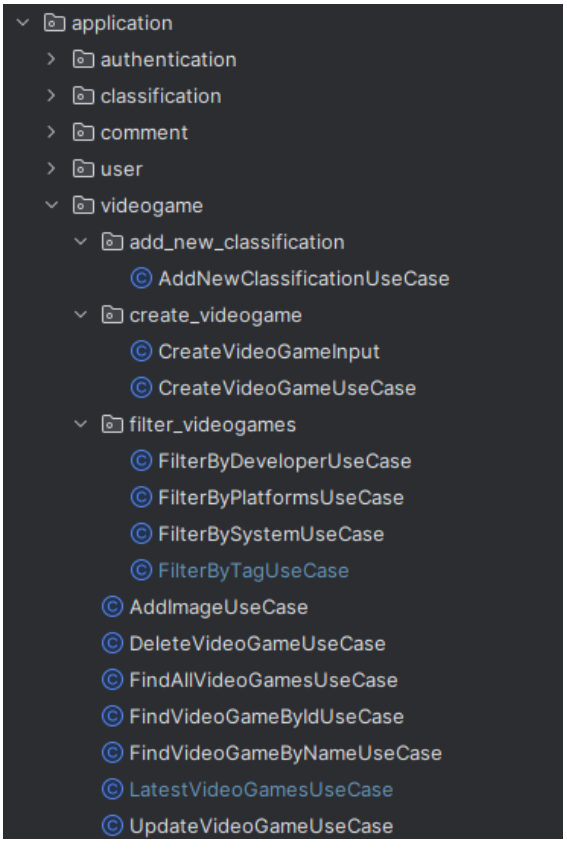

Fig. 69 Casos de uso de videojuegos en la capa de aplicación

5.2.1.3 Capa de Dominio

En este subdirectorio hay dos carpetas: una en la que se definen todos los modelos de datos del dominio y otra en la que se definen los servicios que se comunican con los adaptadores para hacer solicitudes a base de datos:

- ***model***: en esta carpeta se encuentran todas las clases en las que se definen las entidades del modelo de negocio (*Fig.70*). Para no tener que realizar la implementación de los *getters*, *setters* y constructores, se ha usado Lombok, teniendo solo que indicar los campos de los que se compone cada entidad. Además de las entidades del modelo de negocio, también están definidas las clases de datos usadas en los casos de uso de autenticación (*Fig.70*).

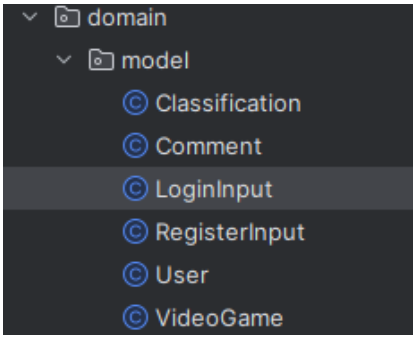

Fig. 70 Clases de los modelos de datos de negocio en la capa de dominio

- ***services***: en esta carpeta se encuentran todas las clases usadas por los casos de uso cuando quieren que la base de datos les de servicio para realizar transacciones. Estas clases transmiten estas peticiones a los adaptadores, en la capa de infraestructura. Para no tener que implementar los constructores se ha usado Lombok. Por cada tipo de entidad hay una clase de servicio (*Fig.71*) donde se implementan las funciones necesarias para cada transacción.

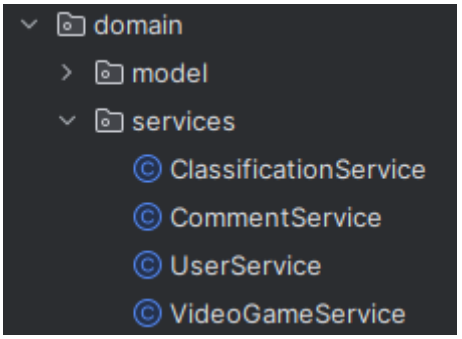

Fig. 71 Clases de servicios de la capa de dominio

5.2.1.4 Test unitarios y de integración

Las pruebas unitarias, hechas con JUnit, se han agrupado en función de la entidad sobre la que se trabaja en la clase. Por esto, existen cinco subcarpetas (*Fig.72*): *authentication*, *classification*, *comment*, *user* y *videogame*. Dentro de cada una de estas carpetas hay una prueba por cada *mapper* y caso de uso que hay en el servidor. Como las pruebas unitarias buscan probar solo el correcto funcionamiento de la lógica de negocio, no hay test unitarios de ninguna de las clases de la capa de infraestructura, a excepción de los *mappers* ya que no cuentan con ninguna dependencia externa. Tampoco se ha hecho test unitario de las clases de datos ya que estas son usadas dentro de las pruebas de los casos de uso o de los *mappers*, siendo innecesaria una prueba aparte. Aquellas clases que no estaban siendo probadas dentro del test han sido sustituidas usando Mockito.

Por la parte de las pruebas de integración, estas se han escrito usando Postman. Se ha creado una carpeta por cada una de las entidades (*Fig.73*) y dentro de cada una de estas carpetas se ha creado un caso por cada petición REST que se puede hacer con la entidad correspondiente. Estas pruebas fueron creadas en etapas tempranas del desarrollo para comprobar el funcionamiento correcto del servidor y de la base de datos. Además de estas carpetas, también se han creado varios escenarios para poder comprobar casos de flujo más complejos que lanza el usuario (*Fig.73*), como crear un videojuego y después buscarlo usando varios de los filtros disponibles o crear varios videojuegos y después recuperar los tres últimos.

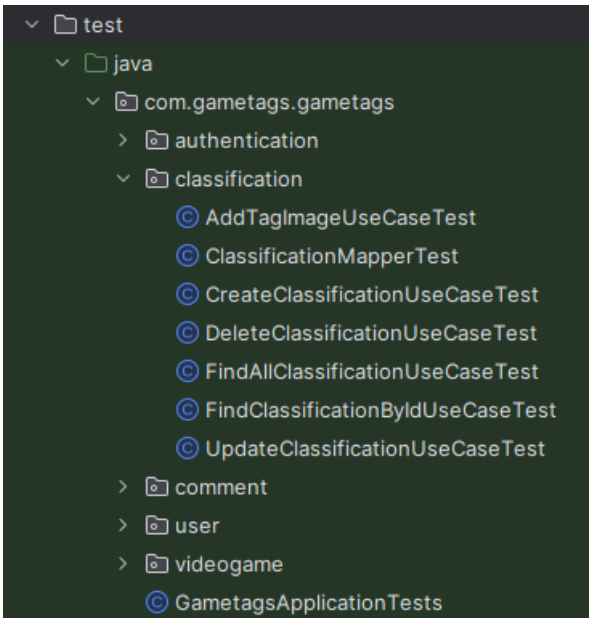

Fig. 72 Test unitarios del servidor

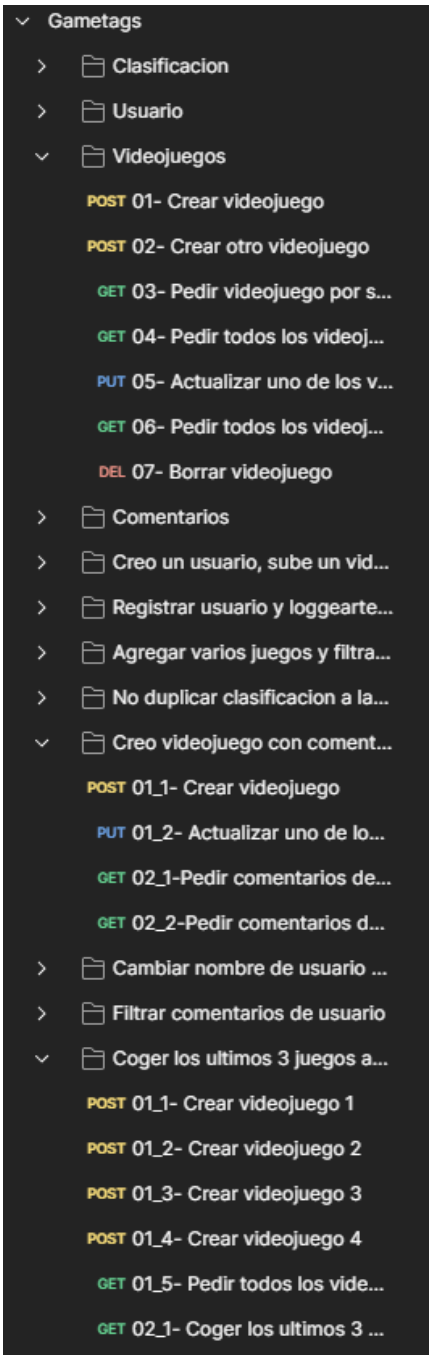

Fig. 73 Escenarios de las pruebas unitarios

5.2.2 Desarrollo del Front-End

Los archivos desarrollados para el *front-end* se han organizado siguiendo la estructura impuesta por Next.js: las páginas se encuentran dentro del directorio *pages*, los archivos de CSS dentro del directorio *styles*, los servicios usados para funciones compartidas y la

comunicación con el servidor en el directorio *service*, componentes de las páginas como el *layout* o la barra de navegación en el directorio *components* y las imágenes dentro del directorio *public*.

5.2.2.1 Pages

Dentro de este directorio varias de las páginas se agrupan en dos carpetas, mientras que el resto no se anidan dentro de ninguna carpeta (Fig.74). Las carpetas son *home* (Fig.74), donde se encuentra la página principal, y *profile* (Fig.74), donde se encuentran todas las páginas relacionadas con el usuario (iniciar sesión, la propia página de perfil del usuario, etc). Cada uno de estos archivos de páginas se divide en dos partes: la primera en la que se implementan las funciones usadas exclusivamente por la página (si comparte función con otra página ésta se habrá implementado en el servicio de *miscService*) y la segunda en la que se especifica la estructura HTML de la página, incluyendo las ventanas emergentes de las que haga uso la página.

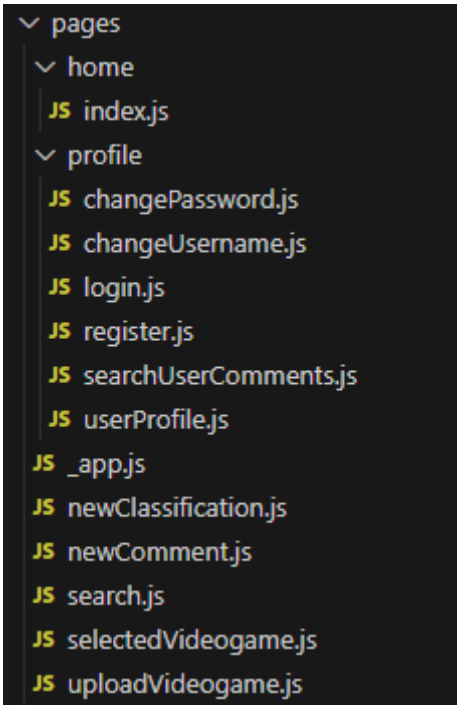

Fig. 74 Archivos de las páginas dentro del directorio *pages*

5.2.2.1.1 Índex

Esta página corresponde a la página principal de GameTags que se encuentra alojada dentro de la carpeta *home*. Solo cuenta con la implementación de una función: la llamada al servidor para recuperar los tres últimos videojuegos añadidos a la base de datos (***threeLatestAddedVideogames***), pasándole como argumento de entrada el *endpoint* al que tiene que llamar. Se llama a esta función antes de que termine de cargar la página. Junto a esta función también se habilita el tutorial de Intro.Js de la página.

5.2.2.1.2 ChangePassword

Este archivo corresponde a la página del cambio de contraseña del usuario y se encuentra alojado dentro de la carpeta *profile*. Antes de terminar de cargar la página se llama a la función ***getUserRequest*** implementada en ***miscService*** para recuperar los datos el usuario actual y se habilita el tutorial de la página. La función que usa la página y que se implementa en este archivo es la siguiente:

- ***updatePassword***: recibe como parámetros de entrada la nueva contraseña, la contraseña previa y la nueva contraseña de nuevo para verificar que no ha habido error gramatical. Antes de realizar la petición se comprueba que la información aportada por el usuario esté completa. Si no es así, se mostrará la ventana emergente indicando que falta información. Si la información está completa, comprueba que la contraseña nueva coincida con la contraseña reescrita. De no ser así, aparecerá otra ventana emergente indicando que las contraseñas no coinciden. Si coinciden, entonces manda la petición al servidor para actualizar la contraseña. Si el usuario escribió una contraseña errónea en el campo de la contraseña previa, aparecerá un error tras la respuesta del servidor indicando que la contraseña era incorrecta y que no se pudo hacer el cambio. Si el cambio se realizó satisfactoriamente, se actualiza el *token* previo con el nuevo recibido en la respuesta y se carga una ventana emergente indicando del éxito de la operación.

5.2.2.1.3 ChangeUsername

Este archivo corresponde a la página del cambio del nombre del usuario y se encuentra alojada dentro de la carpeta *profile*. Antes de que termine de cargar la página por completo, se usa la función compartida ***getUserRequest*** para recuperar los datos del usuario actual, además de activar el tutorial de la página. Solo cuenta con la implementación de una función:

- ***updateUsername***: se reciben como parámetros de entrada el nuevo nombre de usuario, el nombre de usuario actual y la contraseña para validar la operación. Primero se comprueba que la información aportada por el usuario esté completa usando la función compartida ***isCompleteInformation***. Si no está completa, aparecerá una ventana emergente indicando al usuario que falta información. Si está completa, se manda una petición al servidor para actualizar el nombre. Si la contraseña aportada por el usuario es incorrecta, no se habrá devuelto el usuario actualizado y se mostrará una ventana emergente indicando que la contraseña introducida era incorrecta. Si el usuario se ha actualizado correctamente, se actualizará el *token* con el nuevo recibido en la respuesta y se mostrará otra ventana emergente indicando que el proceso se realizó con éxito.

5.2.2.1.4 Login

Este archivo corresponde a la página encargada de gestionar el inicio de sesión del usuario en GameTags y se encuentra dentro de la carpeta *profile*. Antes de que termine de cargar la página se habilita el tutorial de esta. Esta página cuenta con una función propia:

- ***loginUser***: recibe como parámetros de entrada el *endpoint* del servidor al que realizar la petición, el nombre del usuario y la contraseña. Primero realiza la llamada al servidor usando estos datos. Tras recibir la respuesta de este, comprueba que en ella venga el *token* a usar por el usuario actual. De no ser así, se carga una ventana modal indicando que no se pudo iniciar sesión. Si hay *token*, significa que se inició sesión correctamente, por lo que se guarda en el almacenamiento local del navegador el nombre de usuario para recuperarlo en otras páginas y el *token*. Finalmente se redirige a la página del perfil del usuario.

5.2.2.1.5 Register

En este archivo se gestiona la página el registro de un nuevo usuario en la plataforma. Se almacena dentro de la carpeta de *profile*. Antes de que termine de cargar la página se habilita el tutorial de esta. Solo se hace uso de una función propia en esta página, la cual es la siguiente:

- ***registerUser***: recibe como parámetros de entrada el *endpoint* del servidor al que llamar, el nombre de usuario, la contraseña, la contraseña reescrita para validación, el país y un correo electrónico. Primero se comprueba que todos los campos recibidos estén completos usando la función compartida ***isCompleteInformation***. Si faltan

valores en algún campo, aparece una ventana modal indicando de la ausencia al usuario. Si están todos los datos necesarios, se comprueba que la contraseña aportada y la contraseña reescrita coincidan, evitando fallos gramaticales. De no coincidir, se muestra otra ventana modal indicando que las contraseñas no coinciden. Si coinciden, se realiza la petición al servidor. Si en la respuesta viene un nombre de usuario, se considera que la operación ha finalizado correctamente y se muestra una ventana modal indicando esto. Si se hace clic al botón que aparece en la ventana, redirige al usuario a la página principal, indicándole que ya puede iniciar sesión con el usuario registrado. Si no hay nombre de usuario en la respuesta, se considera que falló la operación y aparece una ventana modal indicando que hubo fallo y que lo intente de nuevo.

5.2.2.1.6 SearchUserComments

En este archivo, almacenado dentro de la carpeta *profile*, se gestiona la lógica de la página de búsqueda de comentarios del usuario por parte del cliente. Antes de que termine de cargar la página se habilita el tutorial que aparece en ella. Hace uso de dos funciones:

- ***searchBy***: recibe como parámetros de entrada la temática por la que se filtra y el valor de filtrado que se va a usar. Si la temática por la que se filtra es “categoría” o “severidad”, se hace la llamada al servidor pasándole un objeto simple en el cuerpo de la petición. Si la temática es “videojuego” se hace la llamada, pero pasando un objeto que transformado a JSON en el cuerpo de la petición. Si no hay valor en el campo por el que se filtra o si no se ha devuelto ningún comentario como respuesta, aparecerá una ventana emergente indicando que no se encontraron comentarios. Si por el contrario sí que se recibieron comentarios en la respuesta, estos se mostrarán en el listado inferior en la página usando la función ***showComments***.
- ***showComments***: recibe como parámetro de entrada una lista de comentarios. Se itera sobre la lista de comentarios si no está vacía. Por cada uno de los comentarios de la lista se devuelve un elemento *article* de HTML donde se muestra el valor del texto, de la categoría, de la fecha de subida, del nombre del videojuego que comenta y de la severidad especificada por el usuario, mediante la función compartida ***showRating*.**

5.2.2.1.7 UserProfile

Este archivo gestiona la lógica de la página del perfil del usuario, habilitando el tutorial de la página y recuperando los datos del usuario con una función ***getUser*** implementada en el archivo antes de que termine de cargar la página. Si no hay *token* guardado en el almacenamiento local, se redirige a la página principal. Este archivo se encuentra también

dentro de la carpeta *profile*. Se hace uso de varias funciones privadas implementadas en el archivo:

- ***getUser***: no recibe ningún parámetro de entrada. Recupera el nombre del usuario actual del almacenamiento local ya que el nombre se guardó cuando se inició sesión. Después se realiza la petición al servidor para recuperar el resto de los datos del usuario al que corresponde el nombre. Tras recibir respuesta del servidor, se comprueba si cuenta con valor el nombre de usuario alojado dentro del objeto, indicando que la operación se realizó correctamente. De ser así, se actualiza el nombre guardado en el almacenamiento con el nombre del objeto recibido en la respuesta y se comprueba si contenía también una imagen de usuario personalizada. De ser así, también se guarda en el almacenamiento local. Tras esto, finalmente se guarda el objeto del usuario en el objeto *useState* usado en la página.
- ***showRoles***: recibe como parámetro de entrada el usuario del que se quieren extraer los roles para mostrar. Si el usuario o los roles no están vacíos, se itera sobre la lista de roles. Por cada rol se comprueba su valor, para entonces devolver un elemento HTML distinto en función del rol que se esté examinando. Si el rol es de usuario, se devuelve un elemento HTML que indica que tiene rol de usuario, y si el rol es de administrador, se devuelve otro elemento HTML distinto indicando que es administrador.
- ***deleteUser***: recibe como parámetro de entrada los datos del usuario que se quiere borrar, es decir, el usuario actual. Se lanza la petición al servidor pasando el ID del usuario como argumento. Si la respuesta recibida es de un usuario vacío, indica que el usuario no se ha borrado y se muestra una ventana emergente indicando que el usuario no fue borrado. Si la respuesta recibida no está vacía, significa que se realizó la operación correctamente, por lo que muestra otra ventana emergente indicando el éxito de la operación y que redirige al usuario a la página principal, limpiando el almacenamiento local, incluyendo el *token* de la sesión del usuario eliminado.
- ***uploadProfileImage***: se reciben como parámetros de entrada el *endpoint* del servidor al que hacer la llamada y el archivo de imagen que se quiere guardar. Primero se comprueba que el archivo de imagen no esté vacío. Si lo está, se muestra una ventana emergente indicando que no se ha especificado ninguna imagen a subir. Si el archivo no está vacío, se realiza la solicitud mandando la imagen en el cuerpo de la solicitud. Tras esto, se recarga la página del perfil del usuario.

5.2.2.1.8 _App

Esta página es usada para poder aplicar correctamente el estilo definido en el archivo *theme.css*

5.2.2.1.9 NewClassification

En este archivo se gestiona el funcionamiento de la página que agrega una clasificación a un videojuego. Antes de que termine de cargar la página se habilita el tutorial de la página. La página hace uso de dos funciones propias, las cuales son:

- ***checkExistingClassificationInVideogame***: recibe como parámetros de entrada el *endpoint* al que realizar la solicitud y la clasificación que se quiere comprobar si ya existe o no. Realiza una petición para recuperar los datos del videojuego del que se quiere comprobar el listado de clasificaciones. Tras recibir el videojuego como respuesta, recupera el listado de clasificaciones y filtra dicho listado para comprobar si ya cuenta con la clasificación que se pretende añadir, devolviendo una lista vacía si no la usa el videojuego o una lista no vacía si lo usa ya.
- ***addClassification***: recibe como parámetros de entrada el *endpoint* al que mandar la petición, la clasificación que se quiere añadir al videojuego actual y la imagen de la etiqueta de la clasificación que se quiere añadir. Primero hace uso de la función ***checkExistingClassificationInVideogame*** para evitar duplicar la clasificación en el videojuego. Si recibe una lista con datos significa que ya está siendo usada la clasificación, por lo que se muestra una ventana emergente indicando que la clasificación ya la está usando el videojuego y termina el proceso. En el caso de que la lista esté vacía, se comprueba si los datos de la clasificación a agregar están completos por medio de la función compartida ***isCompleteInformation***. Si no está completo, aparece una ventana emergente indicando que faltan datos y termina la ejecución del proceso. Si están todos los datos, se prosigue a comprobar que el usuario haya iniciado sesión, enviándole a la página principal de ser el caso en el que no haya iniciado sesión. Si ha iniciado sesión se realiza la petición al servidor para agregar la clasificación al videojuego y persistirla en base de datos de no estar ya en ella. Tras esto, se vuelve a hacer otra petición para agregar la imagen de la etiqueta a la clasificación recién creada. Tras todo esto, se redirige al usuario a la página de información del videojuego.

5.2.2.1.10 NewComment

En este archivo se gestiona la lógica de creación de un nuevo comentario para un videojuego. Antes de que termine de cargar la página, se usa la función compartida ***getUserRequest*** para recuperar los datos del usuario, además de habilitar el tutorial de la página. Solo cuenta con una función propia:

- ***addComment***: recibe como parámetros de entrada el *endpoint* al que realizar la petición y el comentario que se quiere crear. Primero se hace uso de la función compartida ***isCompleteInformation*** para comprobar que el comentario cuenta con

todos los datos necesarios. De no ser así se mostrará una ventana emergente indicando que faltan datos, terminando el proceso de creación. Si están todos los datos, se comprueba que el usuario haya iniciado sesión. De no ser así se le redirige a la página principal. Si ha iniciado sesión, crea el objeto del comentario que debe ir en el cuerpo de la petición al servidor usando como base el comentario recibido en la entrada de la función y realiza la solicitud al servidor para crear un comentario con esos datos. Tras recibir respuesta, redirige al usuario a la página de información del videojuego.

5.2.2.1.11 Search

En este archivo se gestiona la lógica del motor de búsqueda de videojuegos de la página de búsqueda de videojuegos. Antes de que haya terminado el cargado de la página se habilita el tutorial de esta. Hace uso de varias funciones propias, las cuales son:

- ***searchVideogame***: recibe como parámetro de entrada el *endpoint* al que mandar la petición y el valor del criterio por el que se quiere filtrar. Primero se comprueba que haya valor en el dato del criterio. De no ser así se cargará una ventana emergente indicando que no se han encontrado videojuegos que cumplan el filtro, terminando el proceso. Si hay valor, se realiza la petición al servidor. Si la respuesta recibida es una lista, se comprobará si no está vacía. En el caso de que lo esté, se mostrará la ventana emergente que indica que no se encontraron videojuegos. Si no está vacía, se mostrarán los videojuegos en la parte inferior de la página. Si la respuesta recibida por el servidor no es una lista, se comprobará el ID del videojuego recibido, comprobando así si es un objeto vacío. Si no hay ID, se mostrará la ventana emergente que indica que no se ha encontrado ningún videojuego que cumpla el criterio. Si hay ID, significa que el servidor ha respondido con un videojuego que cumple el criterio, mostrándolo en la parte inferior de la página.
- ***searchVideogameByPlatforms***: recibe como parámetro de entrada el *endpoint* al que realizar la petición y la plataforma por la que se quiere filtrar. Primero se manda la petición al servidor para encontrar algún videojuego que cumpla el filtro. Similar a la otra función, la respuesta puede ser una lista de videojuegos o un solo videojuego. Se realizan las mismas verificaciones que en la otra función. Si la respuesta está vacía, se muestra una ventana emergente indicando que no se encontró ningún videojuego. El motivo por el que son dos funciones separadas a pesar de hacer lo mismo es porque para el *endpoint* de las plataformas, el filtro se espera recibir en el cuerpo de la petición, mientras que en los otros filtros se pasa como variable de ruta.

5.2.2.1.12 SelectedVideogame

Este archivo se encarga de gestionar la página que muestra la información de un videojuego. Antes de que termine de cargar, se habilita el tutorial de la página, se recuperan los datos del videojuego por medio de la función propia ***getVideogame*** y se comprueba si el usuario ha iniciado sesión con la función propia ***checkUserLogged***. Las funciones propias de la página de las que hace uso son:

- ***getVideogame***: recibe como parámetro de entrada el *endpoint* al que mandar la solicitud. Realiza la petición al servidor para recuperar los datos del videojuego del que se tiene que mostrar la información. Se recibe como respuesta una lista en la que hay un videojuego. Si dicho videojuego tiene nombre significa que se ha recuperado correctamente, guardándolo en el *useState* de la página para poder cargar sus datos.
- ***showPlatforms***: se recibe como entrada el videojuego del que se quieren extraer las plataformas en forma de lista. Lo que hace la función es iterar sobre dicho listado de plataformas, devolviendo un elemento HTML con el nombre de la plataforma para formar parte de la estructura HTML de la página.
- ***showComments***: se reciben como parámetros de entrada el videojuego del que se quieren extraer los comentarios y la categoría de comentario que se quiere mostrar. Se comprueba que haya listado de comentarios. En el caso de haberlo, se itera sobre toda la lista, buscando aquellos comentarios que pertenezcan a la categoría especificada en la entrada de la función. Si se encuentra un comentario que coincida con la categoría, se devolverá un elemento HTML *article* donde se mostrará la información del comentario como el texto, el usuario que lo subió y el enlace a su perfil en el caso de que el usuario actual sea ese usuario por medio de la función propia ***addProfileLink***, la fecha de cuando lo subió, formateada por medio de la función compartida ***fixDate*** y la valoración de la intensidad que otorgó a la categoría de contenido usando la función compartida ***showRating***.
- ***addProfileLink***: recibe en la entrada un nombre de usuario. Se comprueba que dicho nombre coincida con el nombre de usuario del usuario actual. De ser así, se transformará el elemento HTML para convertirlo en un enlace a la página de perfil de dicho usuario. Si no coinciden, no ocurre nada.
- ***calculateRating***: recibe como parámetro de entrada la categoría sobre la que se quiere calcular la valoración final de intensidad. Primero comprueba que haya un listado de comentarios sobre el videojuego recuperado al principio de la página. Si lo hay, se hace un recuento de la cantidad de comentarios que hay por nivel de severidad que pertenezcan a la categoría pasada como argumento de entrada. Tras esto, se comprueba que nivel de severidad es el que más veces se ha repetido y se devuelve un elemento HTML con el nombre de dicho nivel de intensidad.
- ***checkUserLogged***: recupera el *token* del almacenamiento local del navegador. Si resulta que cuenta con valor, significa que el usuario ha iniciado sesión y se mostrarán los enlaces para ir a las páginas de creación de comentarios y agregación de clasificaciones. Si no ha iniciado sesión, esas opciones no se mostrarán en la página.

- ***countComments***: recibe como parámetro de entrada la temática de contenido de la que se quiere realizar el conteo. Se comprueba si se ha recuperado el videojuego al principio y si cuenta con comentarios. De ser así, se filtra la lista de comentarios para quedarse solo con los comentarios que pertenezcan a la misma temática de contenido. Tras esto, se realiza el conteo del resultado de dicha filtración.

5.2.2.1.13 UploadVideogame

En este archivo se encuentra la lógica y definición de la página encargada de la subida de un nuevo videojuego a GameTags. Antes de que termine de cargar la página se habilita el tutorial de esta. Se hace uso de varias funciones específicas de la página, las cuales son:

- ***createClassification***: recibe como parámetros de entrada el nombre del sistema clasificador dueño de la clasificación, la edad de la etiqueta, la URL de la página oficial del sistema y el país del que proviene el sistema. De lo que se encarga la función es de devolver un objeto en el que se encuentren todos los datos de la clasificación que se quiere subir con el videojuego.
- ***addPlatforms***: recibe como parámetro de entrada la lista de todas las plataformas soportadas por GameTags. Se itera sobre los elementos de la lista, comprobando si el elemento HTML con el que se corresponde fue marcado por el usuario. En caso afirmativo, dicha plataforma se incorpora a la lista de plataformas que se devolverá como resultado de la función. Si el elemento HTML correspondiente no fue seleccionado, no se incluye en la lista de resultados.
- ***videogameAlreadyExists***: se recibe como parámetro de entrada el nombre del videojuego que se quiere agregar a GameTags. Se hace una petición al servidor para comprobar si el videojuego que se pretende subir ya existe o no en la base de datos del servidor. Si la respuesta recibida viene vacía, se devolverá un “falso” para indicar que el videojuego no existe actualmente en la base de datos. Si la respuesta no viene vacía, se devolverá un “verdadero” indicando que el videojuego ya existe y que no debería de volverse a agregar.
- ***addVideogame***: recibe como parámetros de entrada el *endpoint* al que realizar la petición, el videojuego que se quiere agregar a GameTags y las imágenes de carátula del videojuego y de la etiqueta de edad de la clasificación que sube con este. Primero se hace uso de la función compartida ***isCompleteInformation*** para comprobar que todos los campos tanto del videojuego como de la clasificación se han rellenado. De no ser así se mostrará una ventana emergente indicando que falta información y terminará el proceso. Si están todos los datos necesarios, se comprobará que el usuario haya iniciado sesión. De no ser así se le redirigirá a la página principal y terminará el proceso. Si ha iniciado sesión se pasará a comprobar si el videojuego ya existe usando la función propia ***videogameAlreadyExists***. Si ya existe el videojuego, se mostrará una ventana emergente indicando que el videojuego ya existe y terminará la operación. Si no existe el videojuego, se realizará la petición al servidor

para crearlo junto con la clasificación que lo acompaña. Tras esto se harán otras dos peticiones al servidor: una para agregar la imagen de la carátula al videojuego recién creado y otra para agregar la imagen de etiqueta a la clasificación que subía con el videojuego. Finalmente se redirige al usuario a la página de información del videojuego que se ha creado.

5.2.2.2 Styles

En este directorio se encuentran los archivos encargados de definir el estilo usado en las páginas de Gametags. En concreto el archivo en el que se encuentra definido el estilo es *theme.css* (Fig.75), el resto de los archivos .css se crearon junto con el proyecto y no se han borrado por ser también necesarios. Este archivo se creó usando como base el archivo que proporciona Pico CSS al ser importado al proyecto, modificando el tema usado en las páginas de GameTags para cambiar elementos como los colores usados en los títulos, en los textos de los *dropdowns* o el color de los botones cuando se mantiene el cursor encima de ellos. Además de eso, en este archivo también se define el estilo personalizado de varios elementos de las páginas, como la barra de navegación, la imagen del título de la plataforma, las imágenes que aparecen en las páginas o el tamaño de las fuentes. También se han aprovechado funcionalidades de CSS (la regla @media [34]) para poder aplicar un diseño *Responsive* a las páginas cuando el ancho de estas se reducía en ventanas más pequeñas o en dispositivos móviles, cambiando el tamaño de la fuente de las letras o moviendo los elementos de la página para acomodarse mejor al nuevo tamaño de pantalla.

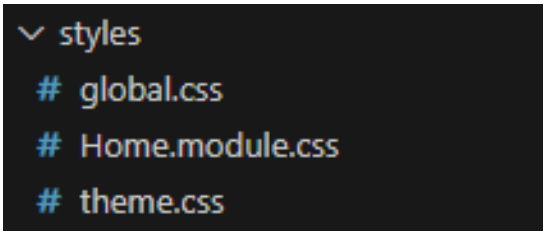

Fig. 75 Archivos CSS en el directorio de *styles*

5.2.2.3 Service

En este directorio se encuentran dos archivos: ***backendService*** y ***miscService*** (Fig.76). En el archivo ***backendService*** se encuentran la configuración para comunicarse con el servidor y la implementación de las llamadas que hace el cliente al servidor cuando el usuario ha realizado alguna acción en la que se precise la actuación del servidor. Las funciones implementadas en este archivo, usando Axios, son los cuatro tipos de peticiones REST que se pueden hacer al servidor (GET, POST, PUT y DELETE). Junto con los datos de la petición

también se envía el *token* de autenticación del usuario actual para que pueda verificar la petición OPTIONS antes de procesar la petición real. En el caso de que en la petición se envíe un archivo de imagen, en el *payload* de la llamada también se agregará el archivo de imagen. Para aquellas peticiones REST que puede hacer un usuario sin haber iniciado sesión, también cuenta con una implementación distinta de la llamada en la que no se manda el *token*. La respuesta que se reciba desde el servidor se transmitirá a la página que hizo la petición.

En el archivo ***miscService*** se encuentran implementadas aquellas funciones comunes usadas por varias páginas. Estas funciones son:

- ***showRating***: función encargada de devolver el elemento HTML correcto en función de un nivel de severidad recibido como parámetro de entrada.
- ***loadModal***: función encargada de habilitar las ventanas emergentes de las páginas.
- ***unloadModal***: función encargada de deshabilitar las ventanas emergentes de las páginas.
- ***isCompleteInformation***: función encargada de comprobar que todos los campos del objeto recibido en la entrada cuentan con valor, considerando entonces que la información está completa.
- ***fixDate***: función encargada de formatear las fechas a un formato legible.
- ***loadTagImages***: función encargada de cargar las imágenes dinámicas de las etiquetas.
- ***getUserRequest***: función usada en múltiples sitios para recuperar los datos del usuario actual.
- ***loadImage***: función encargada de cargar las imágenes dinámicas de las carátulas y del perfil del usuario.

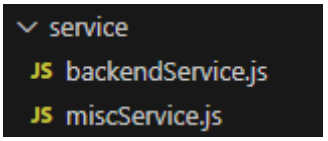

Fig. 76 Archivos de los servicios alojados dentro del directorio *service*

5.2.2.4 Components

En este directorio se encuentran aquellos archivos que no encajaban como páginas o como servicios. En concreto están los archivos que definen el *layout* que usan casi todas las páginas (*layout*) (Fig.77) y el *layout* propio de las páginas de inicio de sesión y de registro (*layout_authentication*) (Fig.77); y el archivo en el que se crea la barra de navegación (*navbar*) (Fig.77). En los dos primeros archivos solo se define el HTML padre que va a anidar al HTML de las páginas que lo van a usar. En el caso del archivo de la barra de navegación a parte de definir la estructura HTML de la barra de navegación tanto para pantallas horizontales como verticales, también se han implementado las siguientes funciones de JavaScript que son usadas por la barra de navegación:

- ***logOut***: función con la que se cierra la sesión del usuario, borrando el *token* de la sesión y redirigiendo al usuario a la página principal.
- ***loadTitle***: función con la que se habilita la imagen del logo de GameTags.
- ***getUser***: función con la que se recuperan los datos del usuario que haya iniciado sesión.
- **enableResponsiveOptions**: función con la que se despliegan y repliegan las opciones del menú de la barra de navegación cuando está en pantalla vertical.
- ***checkUserLogged***: función con la que se habilitan o deshabilitan las opciones reservadas para los usuarios que han iniciado sesión.

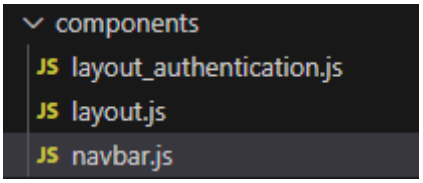

Fig. 77 Archivos del directorio *components*

5.2.2.5 Public

En este directorio se almacenan aquellas imágenes estáticas usadas en las páginas de GameTags. Las imágenes se han guardado en dos subcarpetas distintas en función del tipo de información que buscan representar. Esto quiere decir que las imágenes de los sistemas de clasificación se han guardado en una carpeta llamada *systems* (Fig.78), mientras que el resto de las imágenes como el logo de GameTags, el icono del menú usado en pantallas verticales o la imagen de perfil de usuario por defecto se han guardado en otra carpeta llamada *images* (Fig.78). Además, también siguiendo la jerarquía definida por Next.js, la imagen usada para representar la página de GameTags en la barra de ventanas del navegador (*favicon*) no se ha guardado en ninguna de esas dos carpetas. Se ha mantenido en el mismo nivel de jerarquía que las carpetas de las otras imágenes.

El *favicon* ha sido generado usando Ideogram [33], el cual es un generador de imágenes mediante IA (Inteligencia Artificial) online. Las imágenes de los sistemas de clasificación se han recogido de sus portales, pues son los logos oficiales. La imagen del menú y del perfil por defecto han sido recogidas de Internet. El resto de las imágenes se generaron cuando se creó el proyecto y no se han borrado por precaución.

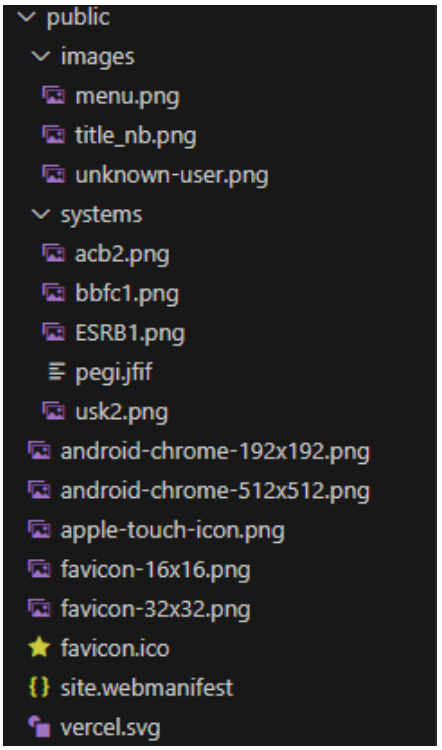

Fig. 78 Imágenes del directorio de *public*

Capítulo 6

Resultados

En este apartado se muestra el aspecto final de la plataforma web y su funcionamiento por medio de una demostración en vídeo

6.1 Resultado final

https://youtu.be/HwEhy7r2iUc

Fig. 79 Vídeo demostración de GameTags

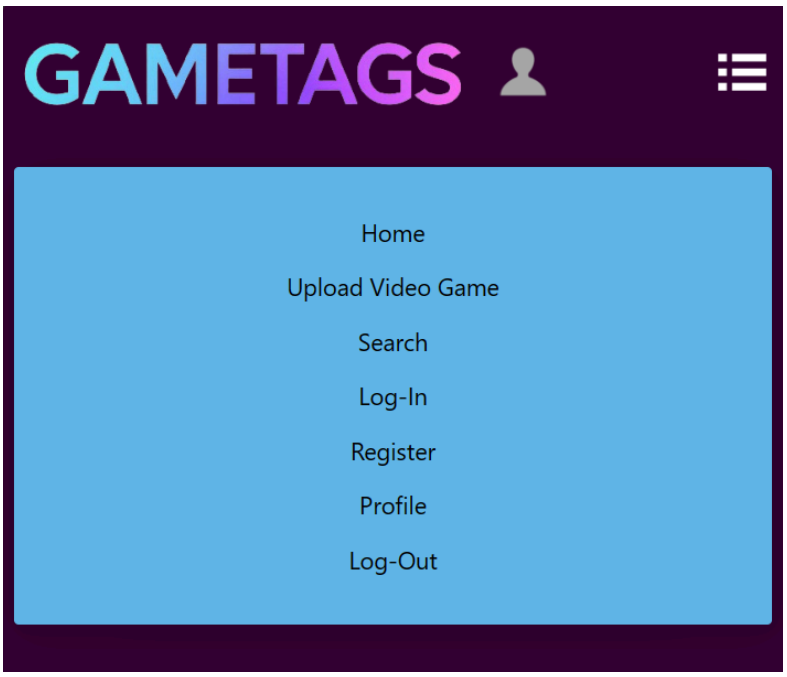

Fig. 80 Barra de navegación final en modo vertical

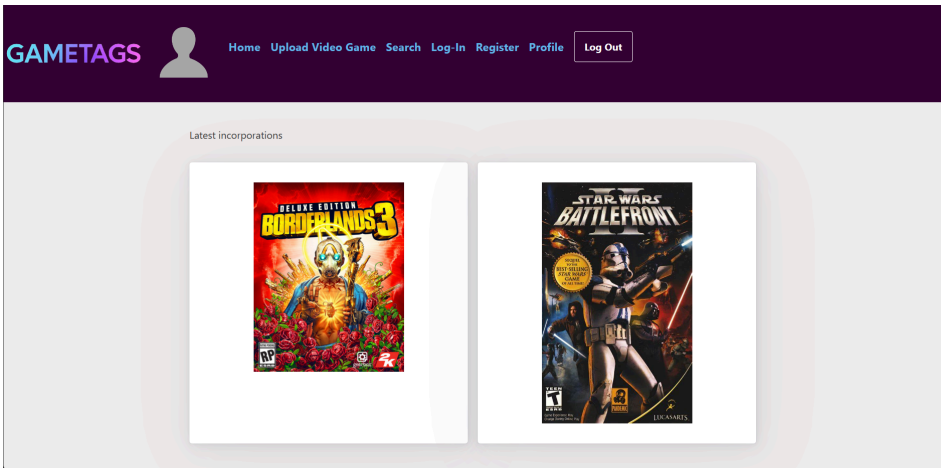

Fig. 81 Página principal final en modo horizontal

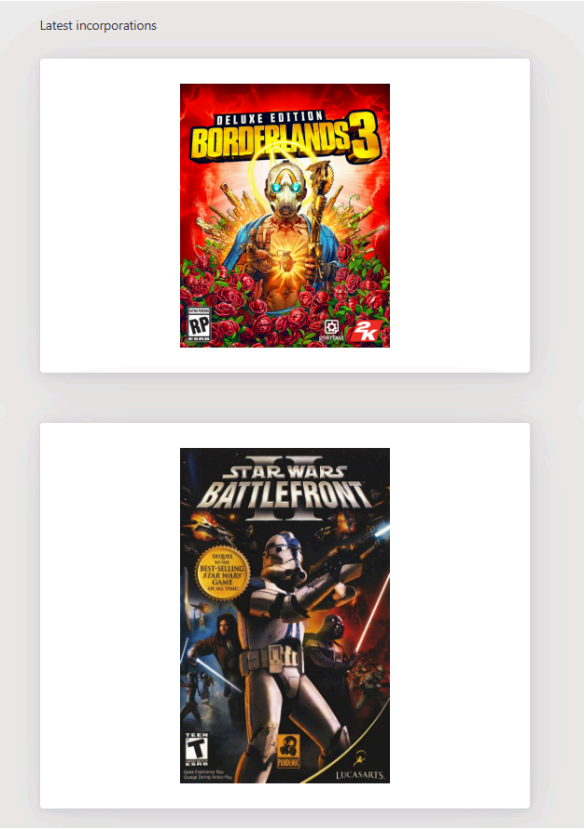

Fig. 82 Página principal final en modo vertical

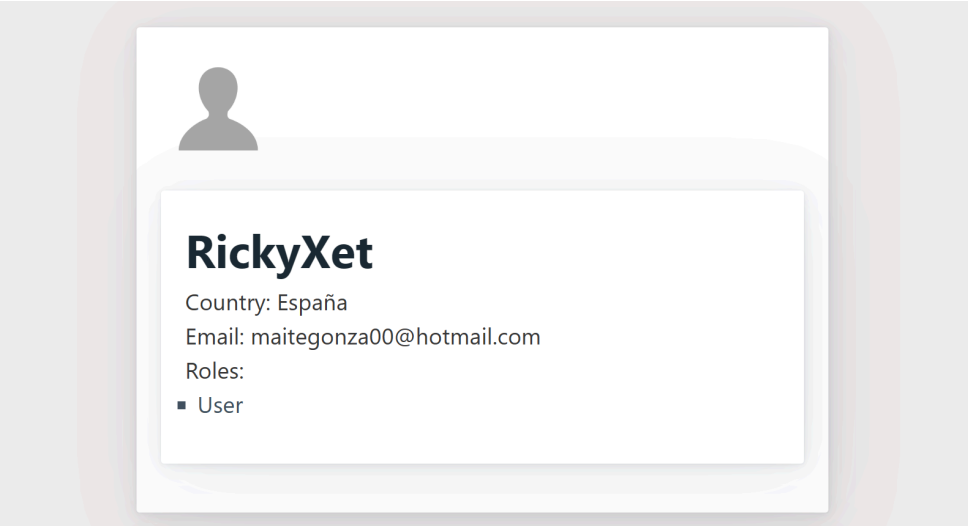

Fig. 83 Datos del usuario en la página de perfil final

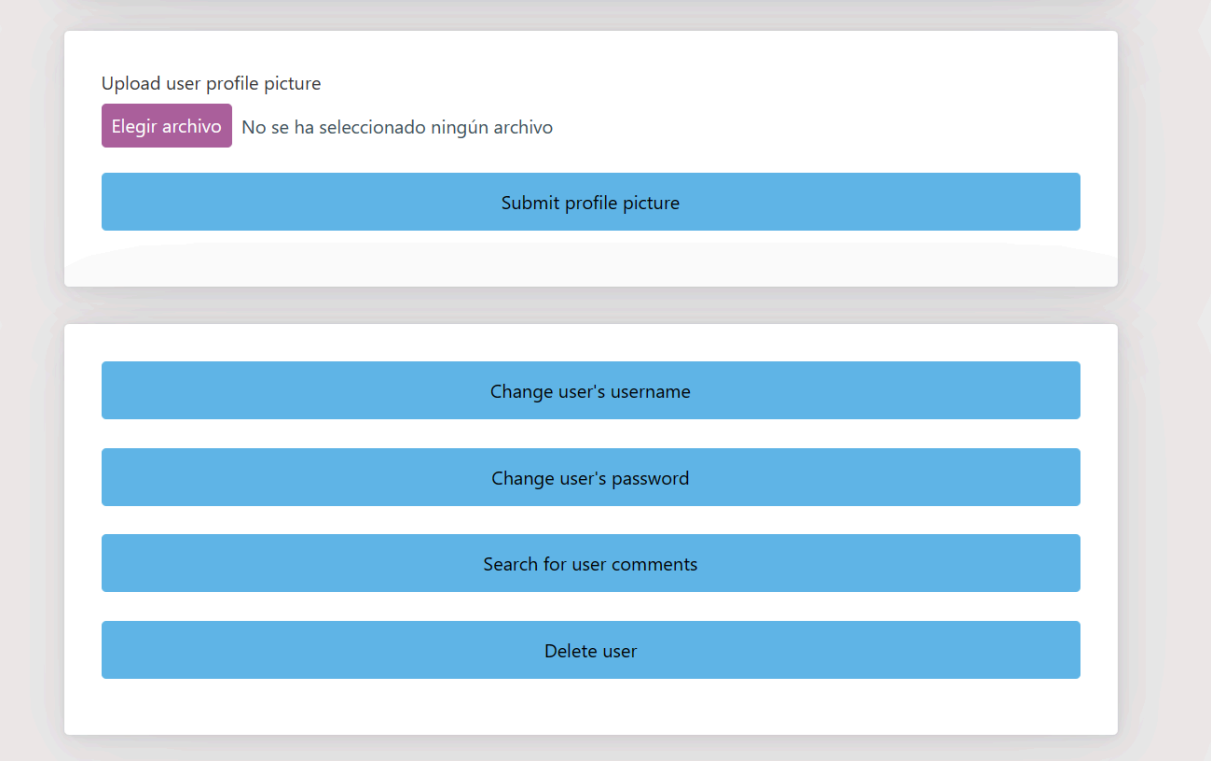

Fig. 84 Opciones en la página de perfil final del usuario

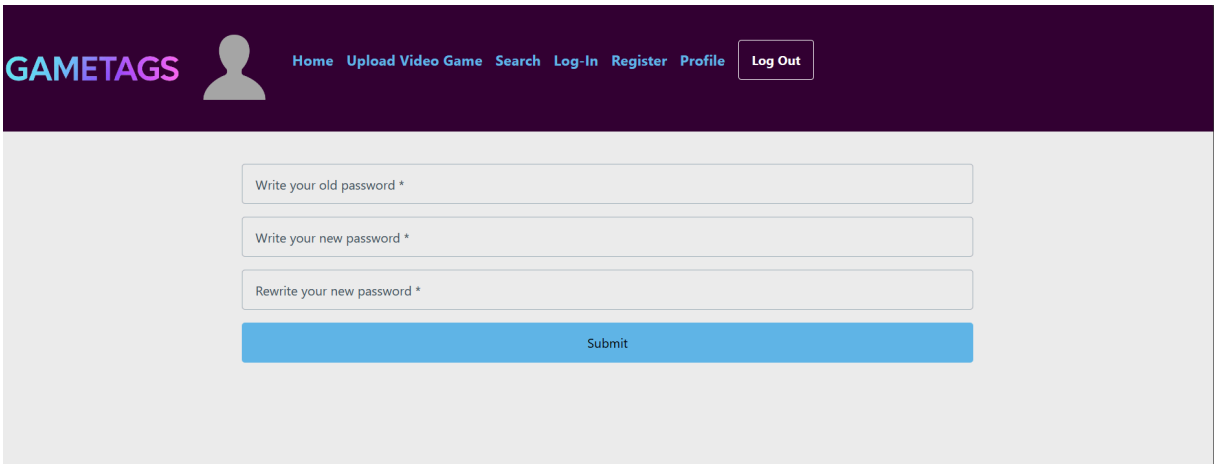

Fig. 85 Página final del cambio de contraseña del usuario

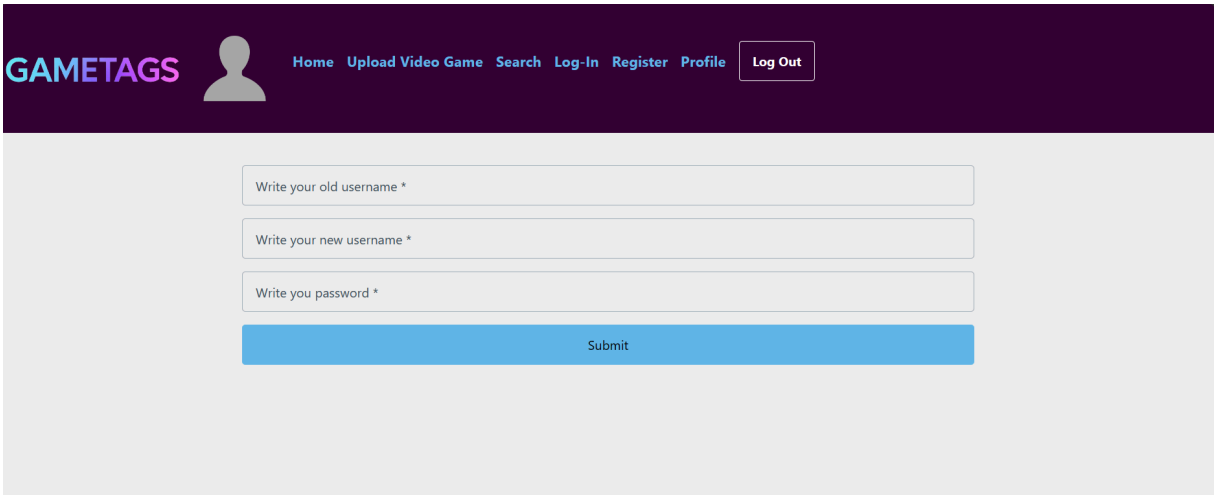

Fig. 86 Página final del cambio del nombre de usuario

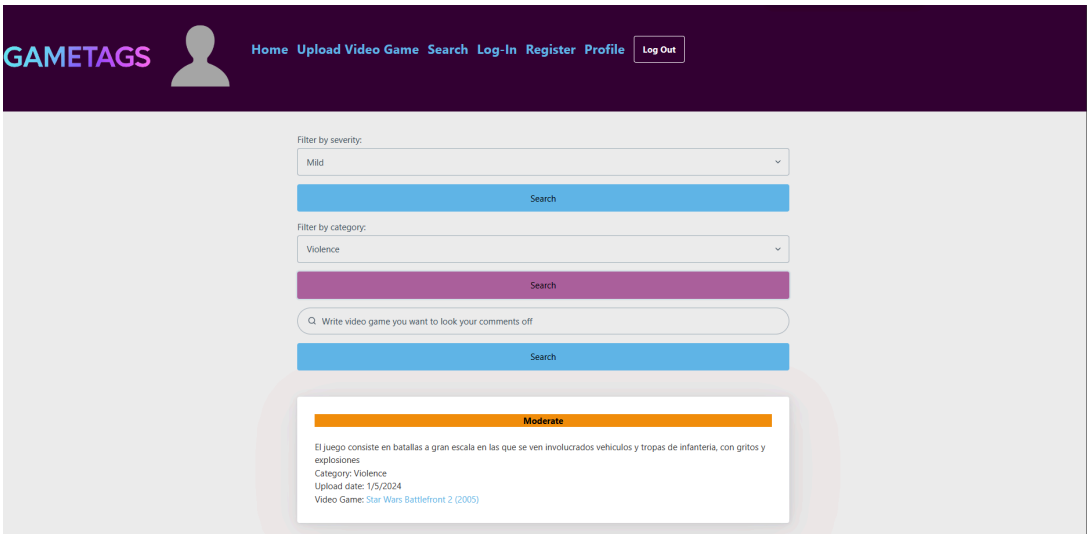

Fig. 87 Página final de búsqueda de los comentarios del usuario

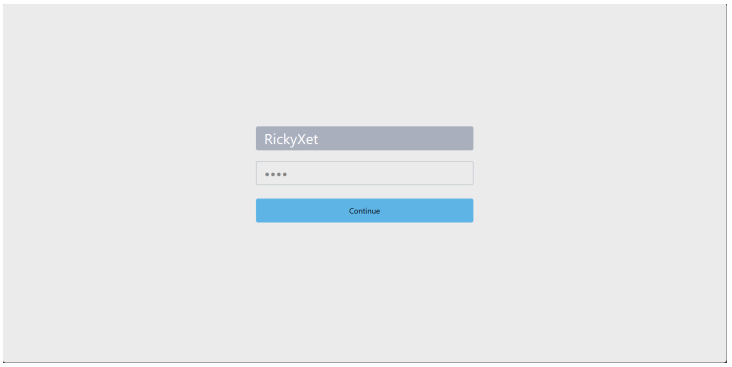

Fig. 88 Inicio de sesión final

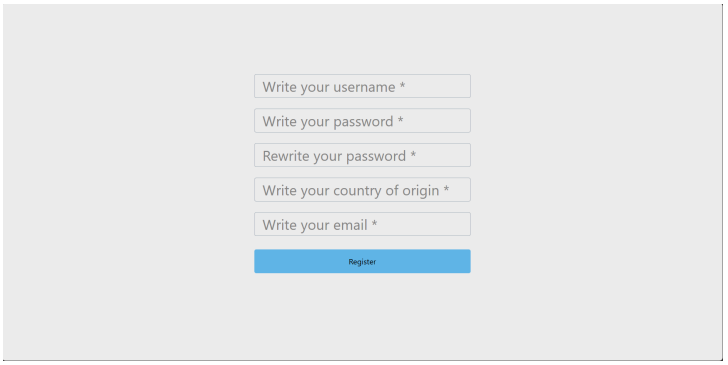

Fig. 89 Página final de registro de un nuevo usuario

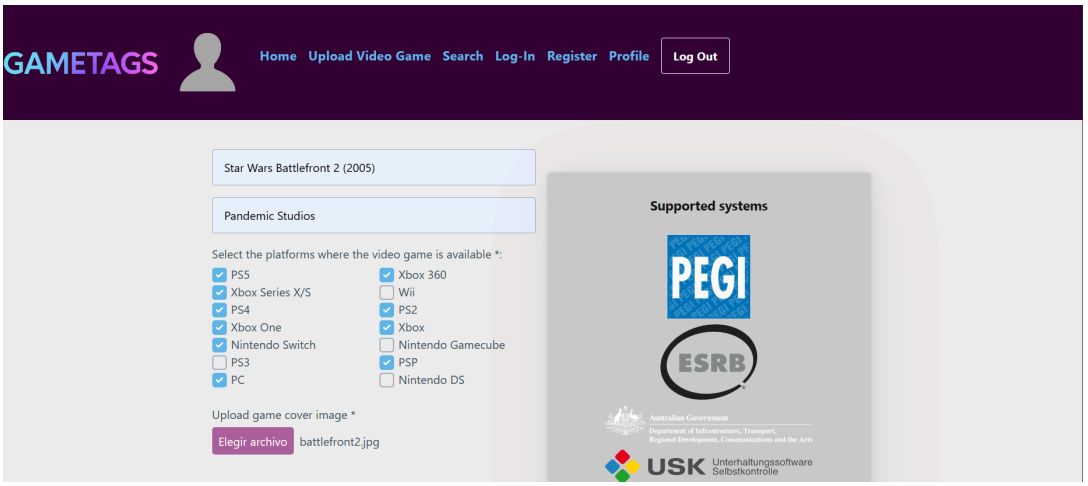

Fig. 90 Introducción de datos del videojuego a subir la página final de subida de un nuevo videojuego

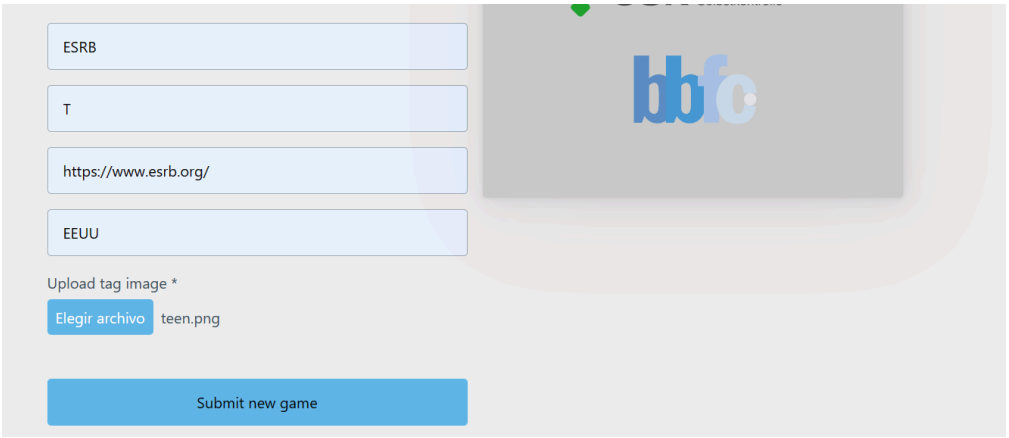

Fig. 91 Introducción de datos de la clasificación del nuevo videojuego en la página final de subida del videojuego

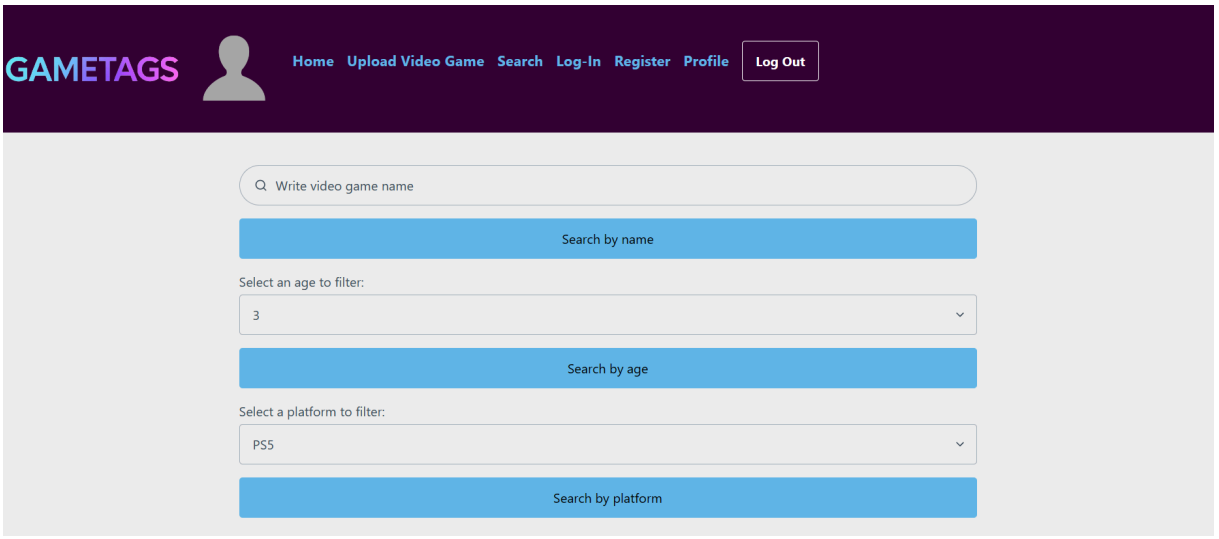

Fig. 92 Filtros de nombre, edad y plataforma de la página final de búsqueda de un videojuego en la plataforma

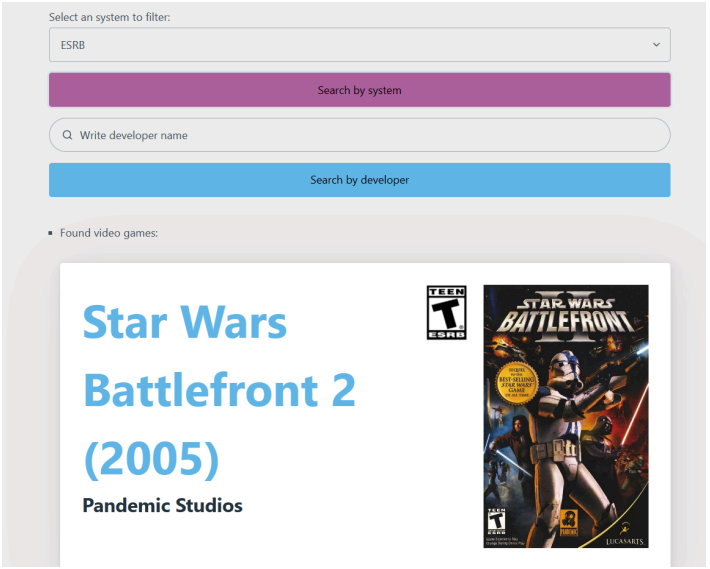

Fig. 93 Filtros de búsqueda por entidad y nombre de desarrolladora y listado de resultados de la página final de búsqueda de videojuegos

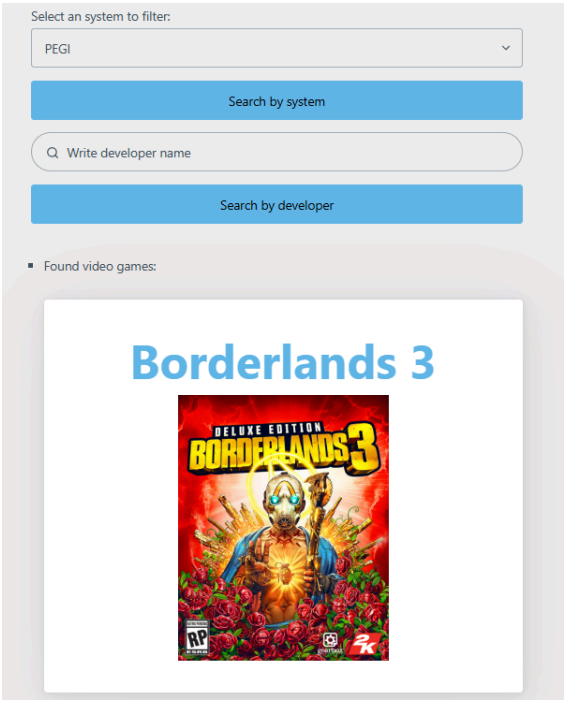

Fig. 94 Vista de resultados de la página final de búsqueda de videojuegos en modo vertical

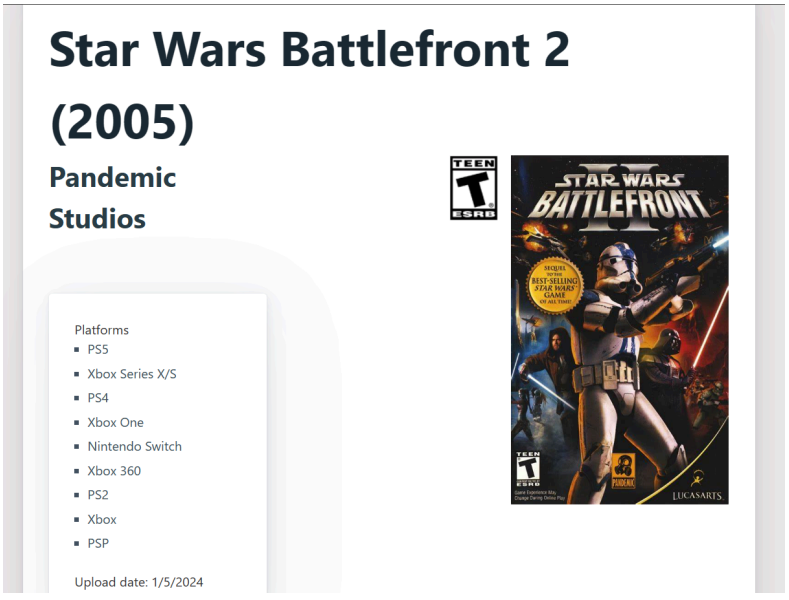

Fig. 95 Datos del videojuego seleccionado en la página final de información del videojuego

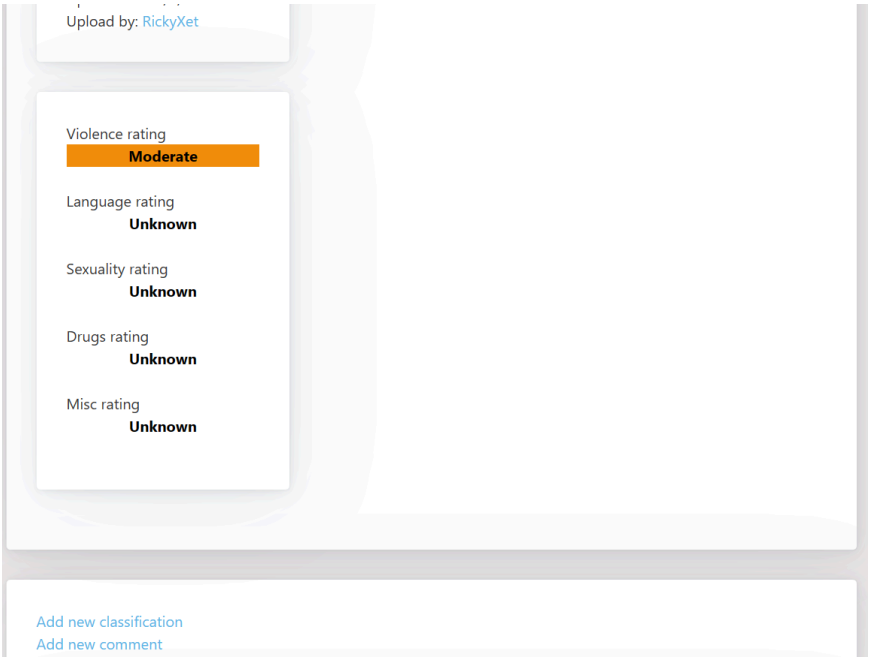

Fig. 96 Valoración media de las categorías del videojuego seleccionado en la página final de información del videojuego

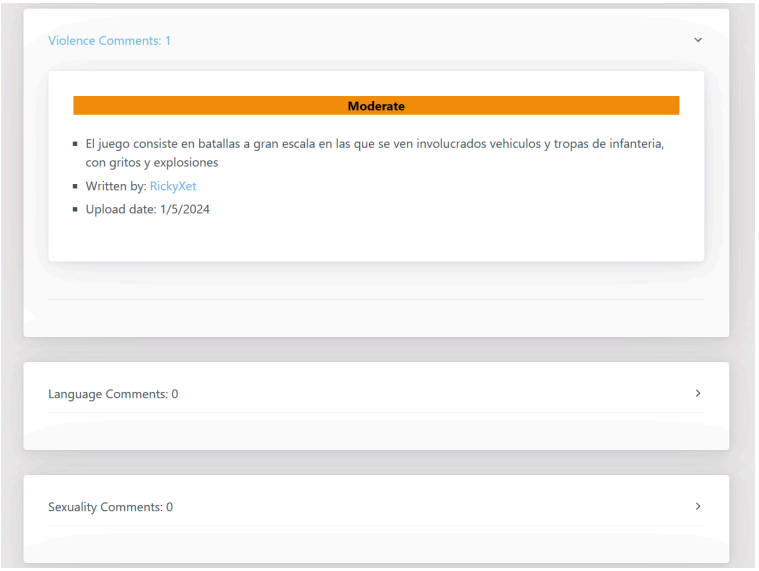

Fig. 97 Comentarios del videojuego seleccionado en la página final de información del videojuego

Fig. 98 Visualización de la información del videojuego en la página de información del videojuego en modo vertical

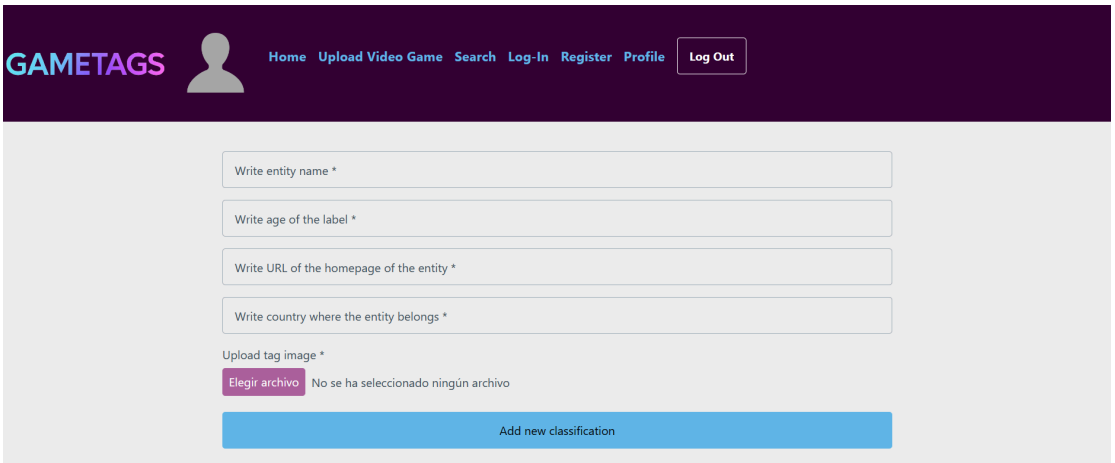

Fig. 99 Página final de la agregación de una nueva clasificación

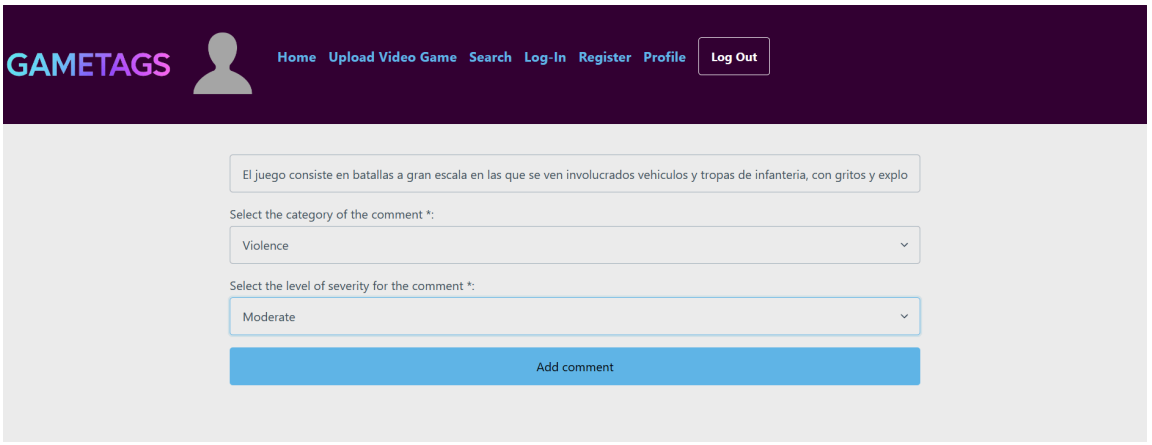

Fig. 100 Página final para la creación de un comentario

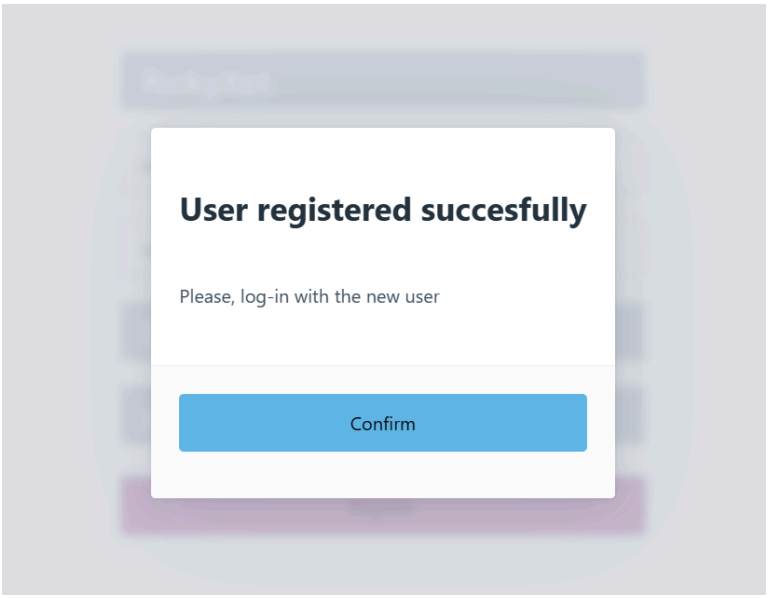

Fig. 101 Modal indicando que el usuario ha sido registrado exitosamente

Fig. 102 Modal indicando que el videojuego a subir ya existe dentro de la base de datos

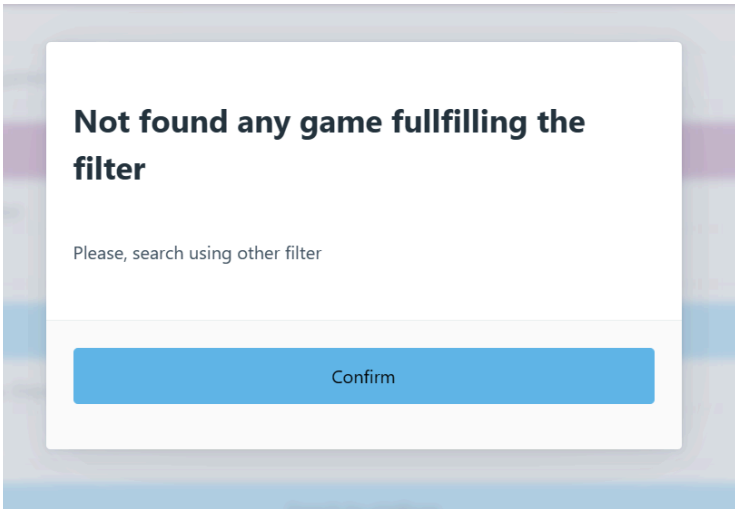

Fig. 103 Modal que aparece en la página de búsqueda de videojuegos cuando no se encuentra ninguno que cumpla las condiciones

Capítulo 7

Conclusiones

En este capítulo se describen los logros obtenidos tras la creación de GameTags, se recapitula sobre lo aprendido a lo largo de su desarrollo y los aspectos en los que se puede seguir ampliando y expandiendo en el futuro.

7.1 Logros alcanzados

A continuación se va a argumentar sobre los objetivos propuestos y como GameTags los cumple:

- *Otorgar a los padres, madres y tutores legales de menores un lugar que centralice todas las clasificaciones por edades oficiales dadas a los videojuegos*: aunque en esta primera versión GameTags no cubra todas los sistemas oficiales usados en el mercado, sí que permite a los usuarios añadir las etiquetas de varios de estos sistemas, ya sea tanto al añadir un videojuego por primera vez a la plataforma como al agregar nuevas etiquetas a videojuegos ya existentes. Además GameTags proporciona los enlaces para acceder a los sitios oficiales de estos sistemas directamente desde sus propias páginas, sin que tenga el usuario que buscarlo por su cuenta, pudiendo usarse si, por ejemplo, el usuario no se acuerda de que etiqueta tenía el videojuego para la clasificación que quiere subir junto con él.
- *Permitir que los usuarios añadan información importante en cuanto a los contenidos de los videojuegos que no es comunicada por las entidades oficiales*: la meta de las valoraciones en GameTags es que la comunidad colabore con la plataforma, compartiendo sus experiencias y conocimientos sobre los contenidos de los videojuegos por medio del sistema de comentarios. Estos conocimientos y experiencias pueden ser mucho más abundantes y específicas que el material que publican los sistemas oficiales en un sitios web, dando detalles sobre qué puede encontrarse un usuario al jugar un videojuego.
- *Definir el grado de impacto de los contenidos de un videojuego de forma comunitaria*: gracias al sistema de comentarios, GameTags es capaz de calcular el promedio de intensidad de los contenidos de los videojuegos en función de las opiniones de los usuarios, actualmente valorando cinco temáticas de contenido. Además de esto, un

usuario cualquiera puede ver estas opiniones de forma individual, ver el nivel de impacto que ha valorado cada usuario y su justificación de porqué merece ese nivel, usando un código de colores para representarlo.

7.2 Lecciones aprendidas

La falta de experiencia en el desarrollo de *front-ends* de páginas web ha ralentizado bastante el desarrollo de la plataforma. El servidor, al contar con mayor experiencia, se desarrolló en un periodo de tiempo bastante corto, entre dos y tres meses, pero el cliente requirió de mucho más trabajo, entre seis y siete meses. De haber contado con mayor experiencia y más conocimientos sobre las tecnologías de ese campo el desarrollo podría haberse equiparado al desarrollo del servidor.

Por otro lado, las plataformas que ya existen en el mercado cuentan con un gran marco de mejora con respecto a los objetivos que busca cumplir GameTags. Ya cuentan con plataformas y comunidades consolidadas que podrían ampliar no solo su público sino también sus dominios, logrando alcanzar los objetivos propuestos para este proyecto con un menor número de complicaciones que las encontradas al crear una plataforma desde cero. En el caso de IMDb, darle más protagonismo a la clasificación de videojuegos dentro de su plataforma ya que cuenta con todos los elementos para cumplir los objetivos propuestos en este trabajo. Y en el caso de CommonSense Media ya está orientado y preparado para padres, pero limita la forma en la que los usuarios pueden expresar sus perspectivas y experiencias sobre los contenidos, además de que solo les permite representar el nivel de intensidad del producto por medio de la edad mínima recomendada, sin especificar por contenido al contrario que como lo hace la opinión oficial de CommonSense.

7.3 Líneas futuras

Dado que es una plataforma de reciente creación, construida dentro de un plazo limitado, existen numerosas características e ideas que finalmente se han omitido en el producto final debido a la falta de tiempo para su desarrollo.

Como se mencionó en el apartado de diseño del *front-end*, varias características como el filtrado por múltiples campos en la página de búsqueda de videojuegos y de comentarios o la selección de la etiqueta de edad en función de la entidad escrita en el campo previo a la hora de agregar una clasificación a un videojuego existente, entre otras, son algunas características que podrían ser introducidas en futuros evolutivos de la plataforma, mejorando y ampliando las funcionalidades ya existentes. También, tomando como ejemplo

las páginas web oficiales de las entidades clasificadoras, se podría crear un selector de idioma para aumentar el rango de público potencial que hiciese uso de la plataforma, permitiendo así a usuarios que no sepan inglés hacer uso de ella. Por no mencionar que también se podrían añadir otros filtros como los niveles de impacto de los contenidos, de modo que un usuario pudiese buscar, por ejemplo, aquellos videojuegos cuyo nivel de lenguaje soez sea leve.

También se podría ampliar la información de la que se componen las entidades, añadiendo nuevos campos de datos como fechas de lanzamiento de los videojuegos o una valoración de fiabilidad de cada usuario, para dar un cierto de nivel de garantía sobre si los comentarios del usuario en cuestión son fiables, añadiendo con ello un sistema de evaluación de comentarios y almacenando esta información. Otro aspecto de ampliación sería el de integrar más sistemas oficiales de clasificación a las opciones a elegir en GameTags, ya que siguen faltando entidades reconocidas a nivel global como CERO (*Computer Entertainment Rating Organization*), el sistema de clasificación por edades japonés.

Bibliografía Web

[1] Common Sense Media for Families. (2016, 19 mayo). *Common Sense Organization History video* [Vídeo]. YouTube. Common Sense Organization History Video

[2] CommonSense Media - About Us (2024, Mayo) Our Mission | Common Sense Media

[3] Lenhart, A. (2024, 15 Marzo). Debunking myths about kids' issues. *Common Sense Media*. Debunking Myths About Kids' Issues | Common Sense Media

[4] Foley, E. (2024, 8 Marzo). Four Realities Facing Kids and Families in 2024 – and How to Change Them. *Common Sense Media*. Four Realities Facing Kids and Families in 2024 – and How to Change Them | Common Sense Media

[5] Egozi, S. (2024, 25 Enero). Giving Kids on Both Sides of the Atlantic a Seat at the AI Table. *Common Sense Media*. Giving Kids on Both Sides of the Atlantic a Seat at the AI Table | Common Sense Media

[6] *New Transatlantic Partnership Aims to Put Children's Voices at the Heart of AI Development to Address Impact and Risks*. (2024, 25 Enero.). Common Sense Media. New Transatlantic Partnership Aims to Put Children's Voices at the Heart of AI Development to Address Impact and Risks | Common Sense Media

[7] Lewis, R. (2024, 19 Mayo). *IMDB | History, Features, & Facts*. Encyclopedia Britannica. IMDb | History, Features, & Facts | Britannica

[8] Andreeva, N. (2022, 13 Abril). IMDB TV streaming service rebrands as Amazon Freevee, plans launch in Germany and 70% more originals. *Deadline*.

IMDb TV Streaming Service Rebrands As Amazon Freevee, Plans Launch In Germany And 70%More Originals

[9] Broadcom, Inc. - Spring Boot Main Page (2024, Mayo) Spring Boot

[10] Docker, Inc. - Main Page (2024, Mayo) Docker

[11] The Apache Software Foundation - Main Page (2024, Mayo) Apache Tomcat

[12] The Project Lombok Authors - Main Page (2024, Mayo) Project Lombok

[13] MongoDB, Inc. - Main Page (2024, Mayo) MongoDB

[14] The JUnit Team - Main Page (2024, Mayo) JUnit 5

[15] Postman, Inc. - Main Page (2024, Mayo) Postman

[16] SmartBear Software - Cucumber Main Page (2024, Mayo) Cucumber

[17] AssertJ - AssertJ Documentation Main Page (2024, Mayo) AssertJ - fluent assertions java library

[18] Okta, Inc. - JWT Main Page (2024, Mayo) JWT.io

[19] SmartBear Software - Swagger Main Page (2024, Mayo) Swagger

[20] Meta Open Source - React Main Page (2024, Mayo) React

[21] Vercel, Inc. - Next.js Main Page (2024, Mayo) Next.js

[22] Matt Zabriskie - Axios Documentation Main Page (2024, Mayo) Getting Started | Axios Docs

[23] Usablica - Intro.js Main Page (2024, Mayo) Intro.js

[24] Lucas Larroche - Pico CSS Main Page (2024, Mayo) Pico CSS

[25] npm, Inc. - Main Page (2024, Mayo) NPM

[26] Szczepan Faber - Mockito Main Page (2024, Mayo) Mockito

[27] Twitch - IGDB Main Page (2024, Mayo) IGDB.com

[28] MobyGames™ - Main Page (2024, Mayo) MobyGames

[29] IARC - Main Page (2024, Mayo) International Age Rating Coalition

[30] Novoseltseva, E. (2024, 19 febrero). ¿Qué es arquitectura Hexagonal o arquitectura de puertos y adaptadores? *Apiumhub*.

¿Que es arquitectura Hexagonal o arquitectura de puertos y adaptadores ?

[31] Broadcom, Inc. - Query Methods (2024, Mayo) JPA Query Methods :: Spring Data JPA

[32] *Cross-Origin Resource Sharing (CORS) - HTTP | MDN*. (2024, 8 mayo). MDN Web Docs.

Cross-Origin Resource Sharing (CORS) - HTTP | MDN

[33] Ideogram AI - Main Page (2024, Mayo) Ideogram

[34] Refsnes Data - W3Schools (2024, Mayo) CSS @media Rule