UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una plataforma web para la escena E-Sports

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Landete Zamora, Javier

Tutor/a: Valderas Aranda, Pedro José

CURSO ACADÉMICO: 2023/2024

Dedicatoria

Este proyecto va dedicado a todas esas personas que se han cruzado y quedado en mi vida, dejando una marca inolvidable y que pervivirá por siempre dentro de mí. Una marca llena de experiencias que me han hecho llegar a donde estoy junto a esas personas que me han apoyado y forjado como persona.

Primero, a mis padres y hermanos por ser las personas más importantes de mi vida y creer en mí constantemente, además de estar presentes durante toda mi trayectoria personal, educativa y profesional.

Luego, al resto de mi familia, en especial a mis abuelos por estar siempre presentes, guiarme y ayudarme en todo momento. Siempre estaréis conmigo en mi corazón.

Por último, a mis amigos del instituto con los que he crecido y compartido muchos momentos especiales de mi vida y los de la universidad porque con ellos toda la experiencia universitaria ha sido increíble y hemos pasado muy buenos momentos.

Resumen

El mundo de los videojuegos está cada vez más presente en la sociedad, consolidándose como una de las formas de entretenimiento más importantes actualmente frente a otros métodos más clásicos. Debido a esto, muchos jugadores han empezado a ir más allá de jugar y han empezado a profesionalizarse y a competir entre ellos, generando sus propios ingresos y creándose así los deportes electrónicos o *e-sports*. Por esta razón, se pretende crear un proyecto web para unir a la comunidad competitiva y social del sector y ofrecerles funcionalidades interesantes para formar una comunidad, descubrir nuevos videojuegos y jugadores, competir entre ellos, desarrollarse como jugador y aspirar a ser el mejor en este campo.

Para ello, en general, se han desarrollado una parte cliente o *frontend* para la interfaz visual y las funcionalidades del sistema, una parte servidor o *backend* que servirá como puente para gestionar las peticiones del cliente y consultar y administrar la base de datos, y por último, una base de datos que recoge todos los datos del proyecto. Toda esta arquitectura sumada a una captura de requisitos, elección de herramientas, diseño de interfaces y funcionalidades disponibles, una validación de la interfaz, entre otros factores, lograrán crear el proyecto web *E-ZONE*.

Palabras clave: web, social, *e-sports*, deportes, *online*, videojuegos, portal, interfaz, cliente, servidor, base de datos, usuario

Resum

El món dels videojocs és cada vegada més present a la societat, consolidant-se com una de les formes d'entreteniment més importants actualment davant d'altres mètodes més clàssics. A causa d'això, molts jugadors han començat a anar més enllà de jugar i han començat a professionalitzar-se i competir entre ells, generant els seus propis ingressos i creant-se així els esports electrònics o *e-sports*. Per aquesta raó, es pretén crear un projecte web per unir la comunitat competitiva i social del sector i oferir funcionalitats interessants per formar una comunitat, descobrir nous videojocs i jugadors, competir entre ells, desenvolupar-se com a jugador i aspirar a ser el millor en aquest camp.

Per això, en general, s'han desenvolupat una part client o *frontend* per a la interfície visual i les funcionalitats del sistema, una part servidor o *backend* que servirà com a pont per gestionar les peticions del client i consultar i administrar la base de dades, i per últim, una base de dades que recull totes les dades del projecte. Tota aquesta arquitectura sumada a una captura de requisits, elecció d'eines, disseny d'interfícies i funcionalitats disponibles, una validació de la interfície, entre altres factors, aconseguiran crear el projecte web *E-ZONE*.

Paraules clau: web, social, *e-sports*, esports, *online*, videojocs, portal, interfície, client, servidor, base de dades, usuari

Abstract

The world of video games is increasingly present in society, consolidating itself as one of the most important forms of entertainment today compared to other more classic methods. Because of this, many players have begun to go beyond playing and have begun to become professional and compete among themselves, generating their own income and thus creating electronic sports or *e-sports*. For this reason, the aim is to create a web project to unite the competitive and social community of the sector and offer them interesting functionalities to form a community, discover new video games and players, compete among themselves, develop as a player and aspire to be the best in this field.

To do this, in general, a client part or *frontend* has been developed for the visual interface and the system's functionalities, a server part or *backend* that will serve as a bridge to manage client requests and consult and manage the database, and finally, a database that collects all the project data. All this architecture added to a capture of requirements, choice of tools, design of interfaces and available functionalities, a validation of the interface, among other factors, will create the web project *E-ZONE*.

Key words: web, social, e-sports, sports, online, video games, portal, interface, customer, server, database, user

Índice general

Apéndices

Índice de figuras

CAPÍTULO 1
Introducción

El sector de los videojuegos está adquiriendo una mayor relevancia en la sociedad mundial y se está consagrando como una de las formas de entretenimiento, cultura y arte más importantes. De hecho, en la edición número 23 del informe *Entertainment and Media Outlook 2021-2026 España* realizado por la *PwC*, se estima que el sector crecerá un 4,6 % a nivel global y que en España, se alcanzarán los 34.092 millones de euros y habrá un 5,1 % de crecimiento del sector en 2026 [1].

Además, la industria de los videojuegos está destacando sobre otros medios de entretenimiento por lo que hablamos de una evolución enorme respecto a otras formas de entretenimiento clásicas.

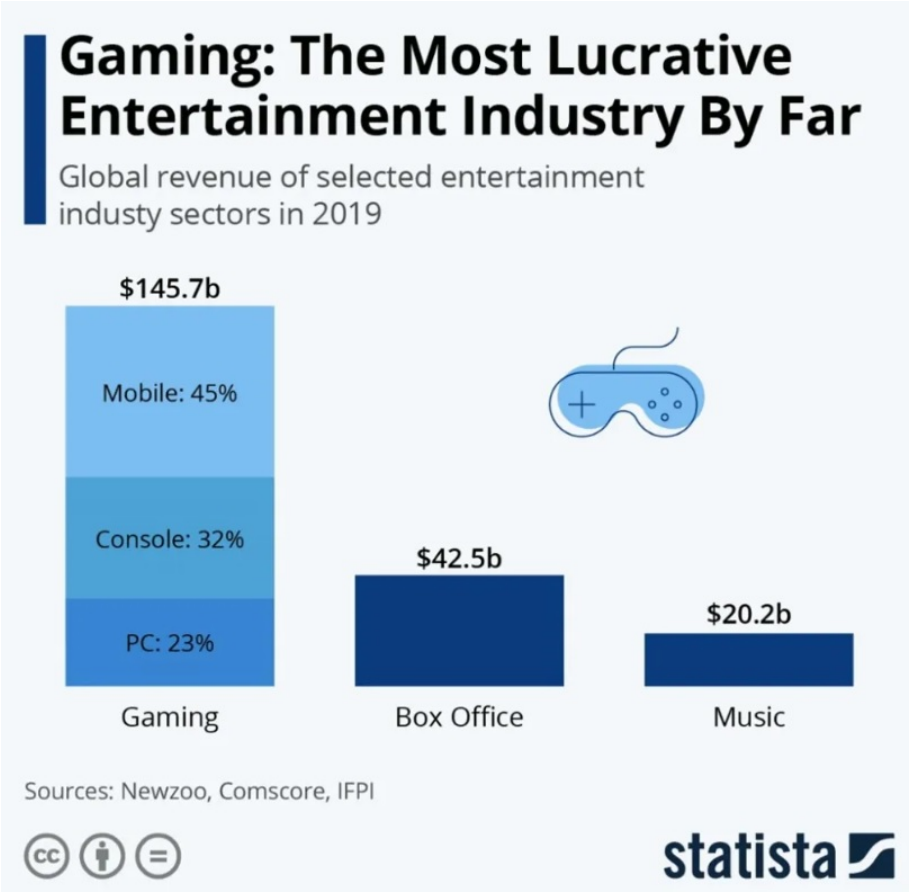

Figura 1.1: Gráfica de ingresos en la industria del entretenimiento

Como podemos ver en la gráfica de la figura 1.1, el sector del *gaming* tiene unos ingresos muy superiores, aproximadamente unos 100 billones de dolares de diferencia, respecto a la música y el cine en 2019. Esto demuestra la gran demanda que hay en esta industria y la relevancia que tiene hoy en día.

1.1 Contexto y motivación

Por todo lo comentado en la sección 1, parte de la población se ha aventurado en el mundo de los videojuegos y han empezado a interesarse en este sector. En consecuencia, muchos jugadores han empezado a mejorar y a profesionalizarse en ello, llegando a generar sus propios ingresos y a conseguir reconocimiento en la industria.

La motivación de este proyecto consiste en unir a toda la comunidad de videojuegos, sobre todo a la competitiva, y crear un espacio donde se puede: aprender sobre la escena competitiva, conocer nuevos videojuegos y jugadores, formar equipos y competir por ser el mejor. Esta unión conseguirá concentrar en un mismo espacio a todos los jugadores y creará una comunidad que aumentará en los próximos años y que mejorará entre todos.

1.2 Objetivos

El propósito principal de este proyecto TFG es crear una aplicación web para la escena competitiva de videojuegos actual. Concretamente los objetivos que busca tener la página web son:

1. Ver cuáles son los videojuegos competitivos actuales de forma detallada.
2. Ver los torneos disponibles según el videojuego seleccionado.
3. Apuntarse para participar en torneos como jugador o equipo.
4. Ver la clasificación de mejores jugadores o equipos en cada videojuego.
5. La posibilidad de que el usuario personalice su propio perfil.
6. Disponer de una tienda para comprar elementos de personalización para el perfil de usuario.
7. Ver información de otros jugadores.
8. Crear equipos.
9. Poder interactuar con la comunidad de la web.

1.3 Estructura de la memoria

Esta memoria está dividida en 9 secciones donde se detallará el proceso que se ha seguido para crear el proyecto web. A su vez, proporcionará una visión global de los pasos del proyecto y el orden seguido. Los apartados son:

1. **Introducción**, sección 1: Se introduce el proyecto, haciendo énfasis en su contexto, la motivación que hay detrás y los objetivos.
2. **Estado del arte**, sección 2: Se presentarán y se estudiarán páginas web similares a las del proyecto. Gracias a este análisis, podremos entender mejor lo que se ofrece actualmente, las funcionalidades que necesitamos y cómo podemos diferenciarnos del resto.
3. **Metodología**, sección 3: En esta sección, encontraremos diferentes formas de abordar el proyecto y cual de ellas será más útil para el desarrollo de este.
4. **Análisis de requisitos**, sección 4: Aquí, se mostrará la forma de obtención de los requisitos y se enumerarán. Además, se desarrollarán mediante casos de uso.
5. **Análisis conceptual y diseño**, sección 5: En este apartado, se explicará el diagrama de clases utilizado que relacionará todos los elementos de la web y su aplicación en la base de datos. También, se mostrará el diseño de las interfaces mediante bocetos.
6. **Desarrollo de la solución**, sección 6: Se definirá la arquitectura del sistema, en concreto, los elementos que participan, la forma de comunicación y la información que se envía. Además, se explicarán las herramientas utilizadas en el desarrollo y los lenguajes de programación. Por último, se proporcionará ejemplos representativos del sistema mediante código.
7. **Producto desarrollado**, sección 7: Se presentará mediante capturas de pantalla la página web desarrollada y las acciones disponibles para el usuario.
8. **Validación**, sección 8: Se hará una validación de la interfaz creada.
9. **Conclusiones**, sección 9: Se proporcionará un resumen que aportará una visión general del proyecto y se destacarán algunas mejoras para futuras actualizaciones.

CAPÍTULO 2
Estado del arte

Gracias al aumento constante de popularidad de los videojuegos y de su entorno competitivo, existen múltiples plataformas, páginas web o aplicaciones que aplican de alguna forma los objetivos descritos para este proyecto. Aún así, muchas de estas no consiguen llegar al público por diversas razones, ya sea porque las funcionalidades que ofrecen no consiguen encajar entre los jugadores, no adquieren la popularidad necesaria para ser una plataforma constante con una base sólida de usuarios o porque sus desarrolladores no aplican mejoras.

Ahora se va a realizar un estudio sobre proyectos similares a este y analizar si funcionan o no en la comunidad, el por qué de su éxito o no y buscar algo diferente para resaltar sobre ellas.

2.1 Estudio del estado del arte

A continuación, se presentarán algunas páginas web del sector, donde se ofrecen varios de los objetivos mencionados anteriormente, sección **1.2**. Aún así, como se verá, muchas de estas no consiguen ofrecer correctamente estas funcionalidades ni conseguir una base sólida de usuarios, además de que carecen en algunos aspectos.

2.1.1. Torneum

Para empezar, se presenta la página Torneum [**2**], en primer lugar podemos ver la página principal, figura **2.1**. Lo más importante que podemos observar sin iniciar sesión es: los videojuegos a los que se da soporte en esta plataforma, una mera información sobre ellos y una mención a las funcionalidades que aporta. Pese a ello, un usuario novel, que solo quiere mirar si esta página le puede servir para sus intereses, puede no sentirse atraído a quedarse, ya que se tiene que crear una cuenta para poder conocer con detalle el funcionamiento del producto, cuando hay otras plataformas donde no necesitas tener una cuenta para explorarla.

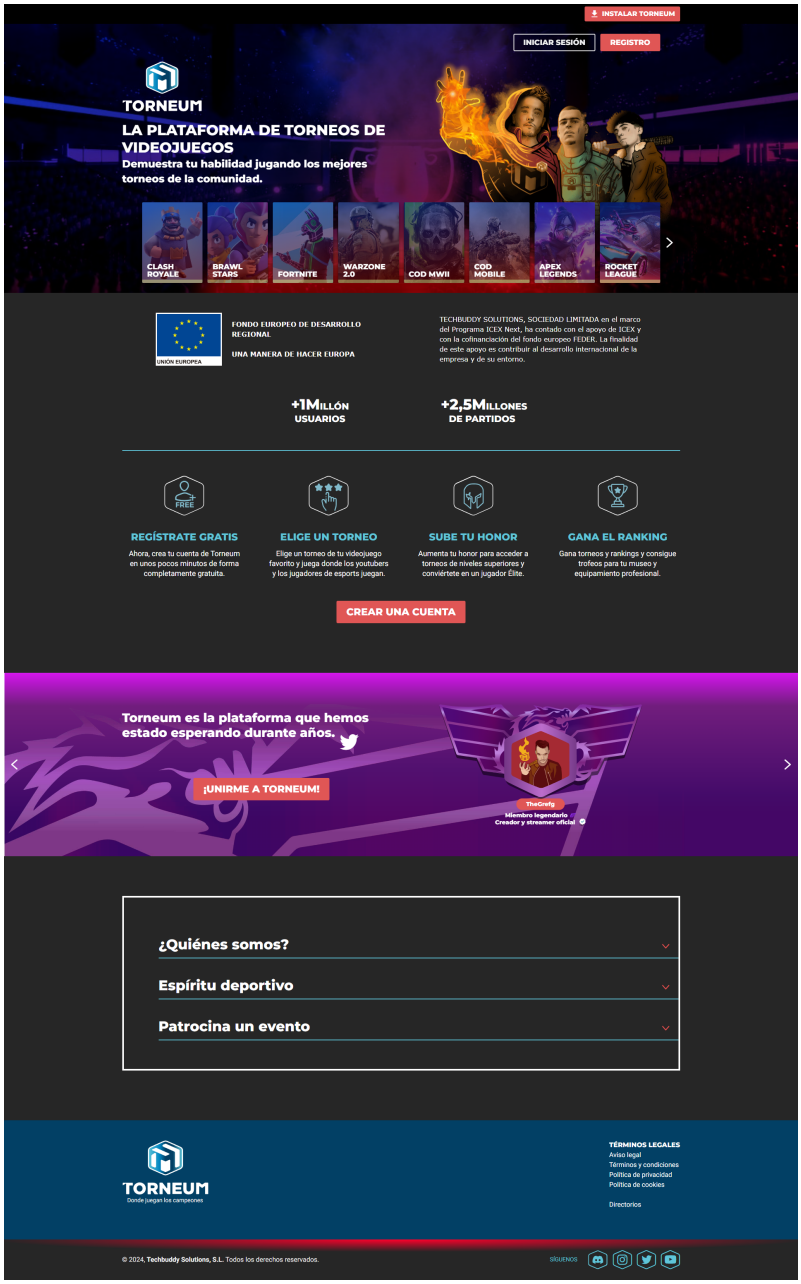

Figura 2.1: Página principal sin sesión Torneum

Una vez iniciado sesión, figura 2.2, se ven todas las posibilidades que ofrece la plataforma, como un apartado de torneos, *rankings*, premios y noticias del *gaming*, a su vez, se puede personalizar el perfil, ver los torneos en los que el usuario participa y un apartado social. En general, todos estos puntos mencionados son muy importantes en el mundo competitivo y suelen ser requeridos en este tipo de plataformas [3]. Uno de los fallos a destacar es que en el menú superior, los torneos se encuentran en la pestaña ***Jugar*** y la de noticias en ***Gaming*** y a algunos usuarios les puede resultar confuso. Por el resto, todo está bastante bien mostrado y funciona correctamente. Respecto a la base de usuarios, es pequeña ya que en el apartado de ***Rankings*** no hay muchos jugadores por videojuego, pese a que hay bastantes torneos disponibles, lo que puede no ser muy llamativo para nuevos jugadores, ya que no podrán conocer ni competir. El último punto a comentar y que puede ser una causa de lo mencionado anteriormente es que llevan desde

2022 sin actualizar el apartado de ***Gaming*** y no hay premios disponibles en el apartado de ***Premios*** por lo que podemos deducir que el mantenimiento de la página no es muy constante. Por último, la interacción que tiene el usuario con la comunidad es prácticamente nula, ya que el apartado ***Social*** de la página solo es para recibir notificaciones y no sirve para interactuar con el resto de jugadores.

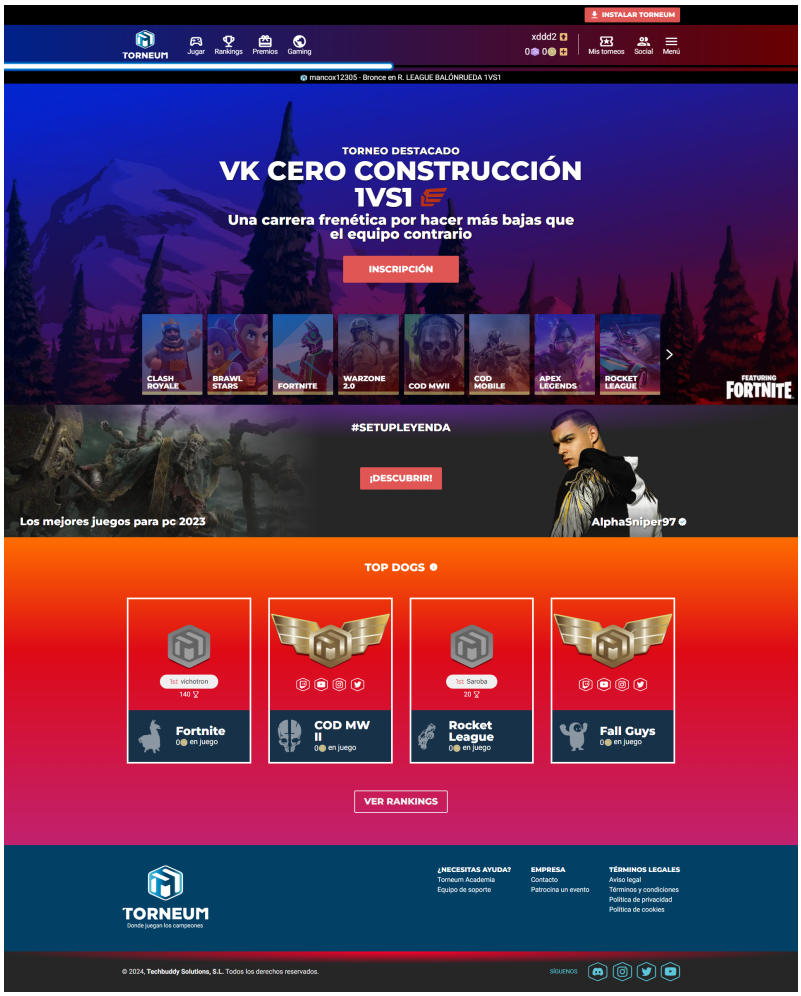

Figura 2.2: Página principal con sesión Torneum

2.1.2. ArenaGG

La siguiente plataforma a comentar es ArenaGG [4]. Al contrario de la anterior página, sección 2.1.1, esta sí deja viajar por todas las funcionalidades al usuario sin iniciar sesión, figura 2.3, también podemos ver prácticamente las mismas posibilidades en el menú y también podemos ver los videojuegos de la comunidad e información sobre ellos. Un error que hay es el apartado de torneos, apartado bastante necesario para una comunidad que busca competir. Este está muy escondido y para llegar a él hay que navegar por varias pestañas, lo que puede resultar confuso y tedioso al usuario. Otro fallo, es el mismo que en la anterior página web, la interacción con la comunidad. Aquí, tampoco puedes interactuar con la comunidad, funcionalidad muy necesaria para este tipo de clientes.

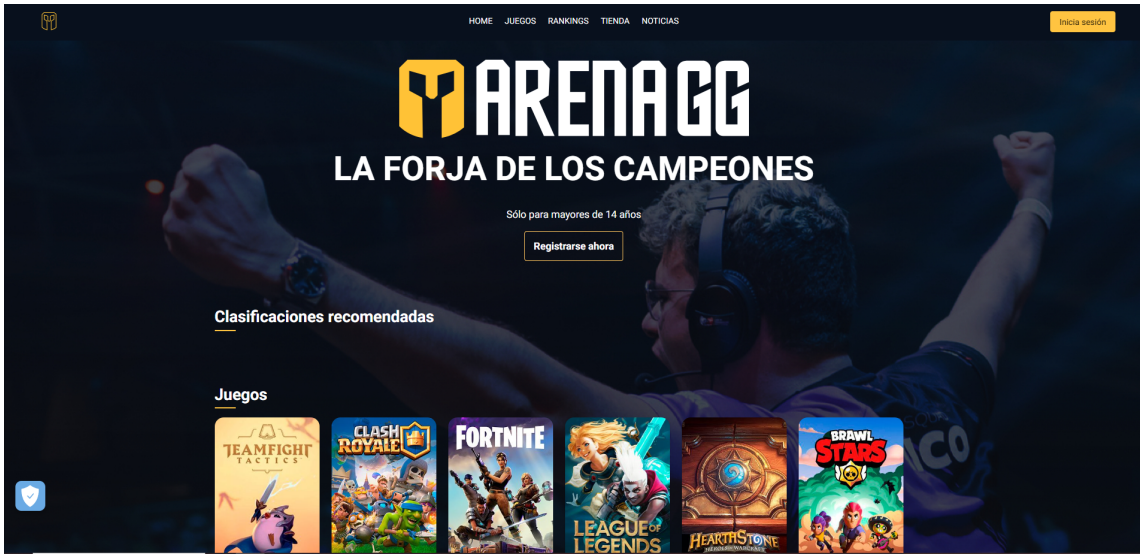

Figura 2.3: Página principal sin sesión ArenaGG

Para continuar, la página es bastante clara y muestra bastante bien todos los apartados de la página. Respecto a la base de usuarios, hay muchos jugadores inscritos en la página y en el apartado de ***Rankings***, se puede comprobar la gran comunidad que tienen algunos de los videojuegos disponibles.

Por último, cuando se inicia sesión, la página principal cambia radicalmente a esta, figura 2.4. Podría ser un fallo de modelo y diseño, ya que como página principal hubiera sido mejor adaptar la anterior, teniendo en cuenta el inicio de la sesión, figura 2.3.

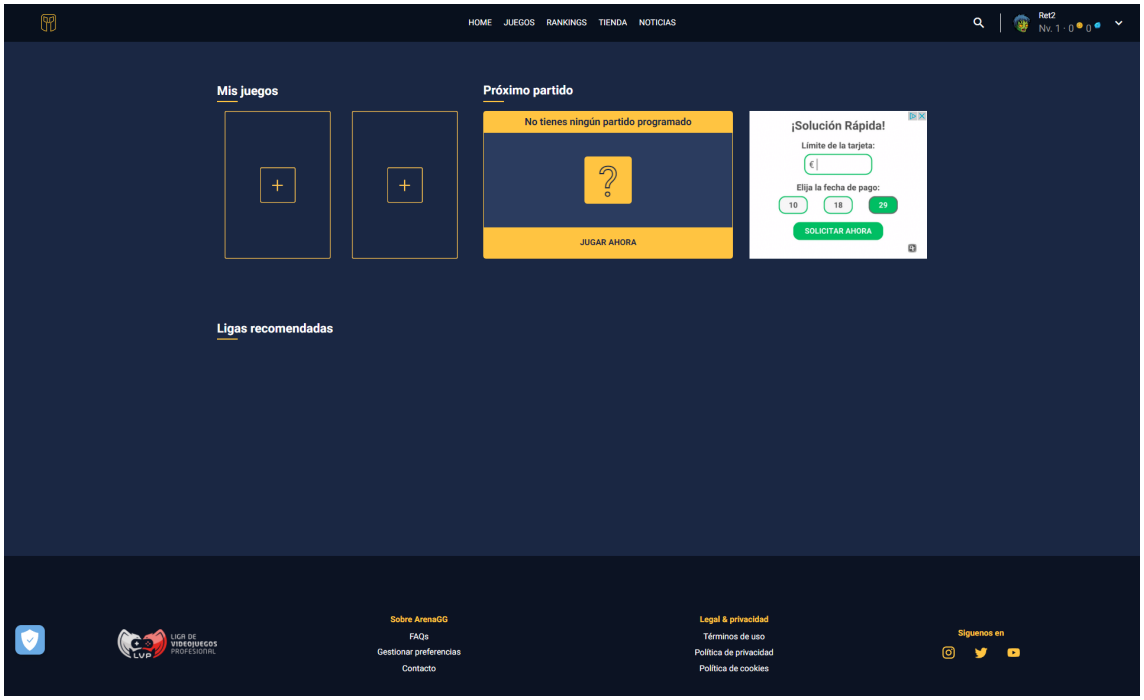

Figura 2.4: Página principal sesión ArenaGG

2.1.3. Torneos.GG

La última plataforma que vamos a mostrar es *Torneos.GG* [5]. En esta página como podemos ver en la figura 2.5, parece una página bastante completa con buen diseño pero carece de muchas funcionalidades. El usuario puede ver los videojuegos disponibles o crear o unirse a torneos, pero por ejemplo, funcionalidades básicas como la clasificación o una tienda no existen en este caso, además, tampoco hay contacto con la comunidad. Después, las pocas funcionalidades que hay están escondidas por la página y a veces es complicado llegar a ellas. Por último, no hay casi jugadores activos, ya que cuando entras a la parte de torneos no hay nadie inscrito.

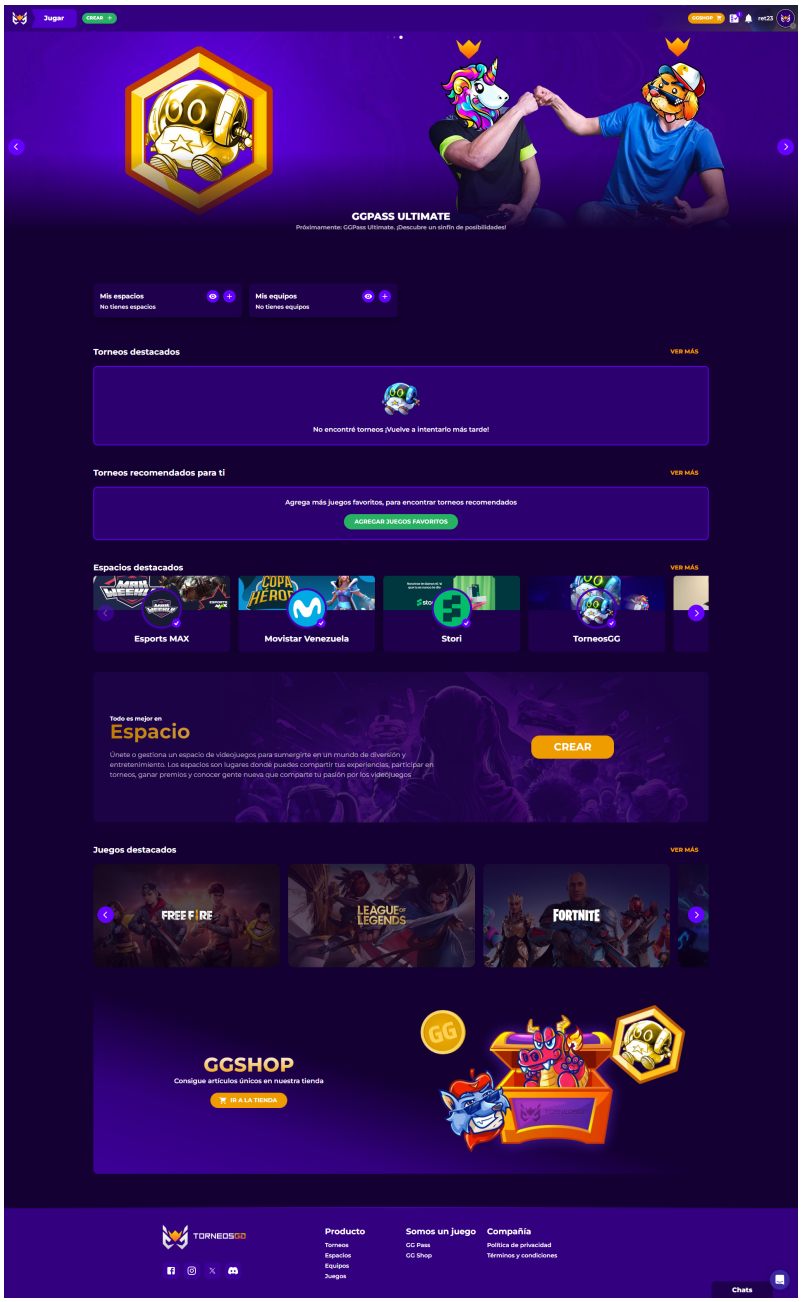

Figura 2.5: Página principal Torneo.GG

2.2 Conclusión del estudio del estado del arte

Como conclusión del estudio hecho en plataformas web parecidas a la del proyecto, podemos ver que en general ofrecen las mismas funcionalidades como los torneos o los *rankings*, además de ofrecer información sobre ellos y los videojuegos competitivos. Algunos problemas que se han observado son:

- Falta de jugadores en la web.
- Nula navegación en los nuevos usuarios.
- Falta de claridad en el menú.
- Poca constancia en el mantenimiento o actualización de la página web.
- Funcionalidades importantes escondidas.
- Mal diseño de la página principal al iniciar sesión.
- Nula participación con la comunidad

Cabe resaltar el último punto de la lista, ya que ninguna de las páginas del sector ofrece una funcionalidad relacionada con la comunidad y su desarrollo puede hacernos destacar sobre el resto.

Por todo lo anteriormente comentado, se puede empezar a plantear el desarrollo del proyecto, ya que se sabe que ofrecer, que hay que evitar o corregir y que no ofrecen el resto de competidores para que el proyecto web destaque sobre los demás.

CAPÍTULO 3
Metodología

Antes de empezar a desarrollar el producto hay que elegir la metodología que se usará para llevarlo a cabo. Este paso es clave, ya que muchas veces se pierde tiempo y dinero en el desarrollo del proyecto simplemente porque no se sabe cómo proceder durante la evolución del mismo y sus imprevistos.

Una metodología en el desarrollo *software*, consiste en un *framework* para planificar, estructurar y controlar la evolución del producto *software*, dando un enfoque sistemático a todo el proceso [6]. Por lo que basar nuestro proyecto en una metodología es un paso básico y fundamental.

Ahora, se va a consultar diferentes metodologías utilizadas actualmente, explicando cómo funcionan y lo que nos aportan. Por último, se elegirá la más adecuada para el producto a desarrollar, ya que escoger la correcta ayudará a reducir la dificultad, organizar las tareas y el proceso y perfeccionar el resultado, además de que evitará, entre otras cosas, un desarrollo complejo, retrasos y errores [7].

3.1 En cascada

La metodología en cascada es un modelo tradicional que consiste en la ejecución en forma lineal de 6 etapas, figura 3.1, donde solo se puede empezar una fase nueva cuando la anterior ha terminado, formando así una cadena de tareas de arriba a abajo, de ahí su nombre. Estas etapas son siempre las mismas, por lo que no cambiarán aunque el producto a desarrollar sea distinto.

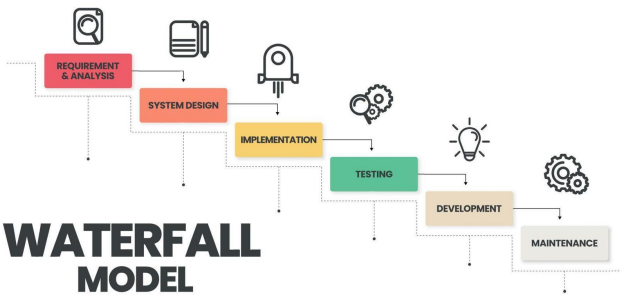

Figura 3.1: Fases metodología en cascada

Las 6 etapas son:

1. **Requisitos**: Se recogen todos los requisitos del sistema y se documentan.

2. **Diseño**: Se crea la arquitectura *software* del sistema y un diseño técnico de cómo será el proyecto en el futuro.

3. **Implementación**: Se transforman los requisitos y todo el trabajo previo en código informático.

4. **Pruebas**: Ahora, a partir de lo que se ha desarrollado, se harán unas pruebas para verificar su funcionamiento. Entre las pruebas que se hacen, habrán; pruebas unitarias, de sistema, integración y de usuario.

5. **Despliegue**: El producto pasa a implementarse y empieza a usarse por los clientes.

6. **Mantenimiento**: Corregir los errores y problemas que aparecen mientras se utiliza la solución realizada.

Las ventajas que ofrece este modelo son que los plazos y el tiempo de realización de las tareas no suele variar mucho y que la metodología es fácil de entender. Pese a esto, presenta algunos inconvenientes, como el nulo impacto del usuario durante el proceso y la propia rigidez del modelo que no admite cambios ni retroceder durante el desarrollo del producto.

3.2 Incremental

El modelo incremental consiste en la entrega sucesiva de funcionalidades del sistema creando así un desarrollo por incrementos del producto hasta llegar a la entrega final. En cada etapa se crea una nueva funcionalidad o característica lo que hace que se vea mejor la evolución del producto, respecto al modelo en cascada, sección **3.1**. Además en cada incremento realizado, el usuario puede probarlo y así los desarrolladores reciben *feedback* del proceso. Por último, cada entrega será una versión mejorada de la anterior y no siempre tiene alguna de las características definidas en la planificación.

Esta metodología tiene muchas ventajas, como hemos mencionado anteriormente, nos ofrece una visión continua y progresiva sobre la evolución del producto que resultará en un mayor conocimiento sobre el estado actual del proceso frente a otras metodologías más lineales. También, es más flexible ya que nos da la libertad de desarrollar las funcionalidades en el orden que necesitemos sin tener que seguir un orden prefijado, además de que podemos incluir características no planteadas al comienzo. Por último, la retroalimentación que nos dan los usuarios es de gran ayuda para solucionar problemas y añadir elementos.

3.3 Metodologías ágiles

Son un grupo de metodologías muy empleadas actualmente por las mejores empresas que nos dan una visión diferente de cómo desarrollar el *software*. Destacan por la capacidad de adaptar la forma de trabajo a las condiciones del proyecto, al entorno y a los problemas, consiguiendo así mayor elasticidad y rapidez a la hora de realizar el proyecto [8]. También, consiguen dotar a los equipos de trabajo de autonomía y eficacia en la gestión y desarrollo de los proyectos.

Existen muchos tipos de metodologías ágiles, algunas de las más importantes son:

- ***Scrum***: Consiste en segmentar todo el proyecto en tareas pequeñas, cada una con su fase de análisis, desarrollo y *testing*. Es muy útil cuando el producto a desarrollar es muy complejo y se compone de muchas fases.
- ***Kanban***: Está constituido por un tablero visible para todo el equipo, con tres partes: tareas pendientes, en proceso y terminadas. Sus puntos fuertes son la planificación y la mejora del rendimiento.
- ***Extreme Programming***: Trata de impulsar las relaciones entre trabajadores y clientes. Como resultado, el trabajo en equipo y la comunicación entre los trabajadores y los clientes mejoran.

3.4 Metodología escogida: Incremental

Después de comentar y explicar las metodologías más utilizadas, el modelo a seguir para el proyecto será la **metodología incremental**. Esta técnica es la que más se ajusta a lo que necesita el proyecto y al modo de desarrollar pensado. Al no ser un proyecto muy grande ni al tener clientes disponibles constantemente, las metodologías ágiles no podían ser una opción. El modelo en cascada tiene una estructura demasiado rígida y siempre es necesario hacer cambios y volver hacia atrás en este tipo de proyectos, por lo que tampoco serviría. En cambio, la incremental, nos permite ir desarrollando por funcionalidades cada aspecto del proyecto web en el orden que se quiera y necesite la aplicación. Además, gracias a la herramienta *GitHub*, cuando se ha acabado una funcionalidad y se ha sabido que funciona perfectamente, se ha subido a la rama principal, simulando así una entrega por incrementos. También, durante el proceso de realización del proyecto se ha necesitado ir hacia atrás y remodelar.

En la siguiente figura 3.2, podemos ver cómo se muestran las subidas al *GitHub* simulando los incrementos. En el caso de este proyecto, se iban subiendo los cambios cuando se avanzaba de forma significante, se terminaba alguna funcionalidad o se hacía algún ajuste que no se había tenido en cuenta previamente, siempre y cuando todo funcione o avance correctamente. Como podemos ver en la figura 3.2, en los nombres de las subidas de cambios, se comenta el estado del proyecto para que se sea consciente del progreso del proyecto.

En conclusión, se ha elegido la metodología incremental, por todo lo comentado anteriormente. Pese a esto, antes de empezar con ese modelo, se han recogido

unos requisitos y se ha hecho una labor de diseño, para ayudar a desarrollar cada entrega de la metodología escogida.

Figura 3.2: Subidas a *GitHub*

CAPÍTULO 4
Análisis de requisitos

A continuación, se explicará que métodos se han usado para capturar los requisitos que debe tener la página web. Luego, estos se listarán y a partir de ellos se crearán casos de uso que describirán cómo el usuario realiza los requisitos planteados.

4.1 Captura de requisitos

En esta sección se determinarán los métodos usados para recabar los requisitos necesarios para la web. Este paso es primordial, ya que de aquí, se obtendrán las funcionalidades y características necesarias para montar el proyecto y hacerlo llamativo para los usuarios.

El primer método que se ha usado es la observación. A partir de lo visto en otras páginas web del mismo tipo, en la sección 2, se ha observado que normalmente estos proyectos ofrecen a los usuarios la posibilidad de participar en torneos, situarse en el *ranking* y conseguir objetos, ya sea mediante una tienda o premios. Además, dan la posibilidad de personalizar el usuario, crear o pertenecer a equipos y enseñar qué juegos tienen en su biblioteca. Todo esto son características a tener en cuenta, ya que si varias aplicaciones web disponen de ellas es porque a los usuarios les interesa.

El siguiente procedimiento utilizado es la búsqueda por Internet. Para reforzar la conclusión sacada del anterior método, se ha recurrido a averiguar cómo se debe hacer una página web sobre *e-sports*. A partir de la búsqueda realizada, se ha podido concluir que la parte de torneos es algo esencial, de hecho, los usuarios valoran mucho la posibilidad de participar en ellos y poder ver sus detalles. También, la clasificación tanto de jugadores como de equipos, según el videojuego, motiva a muchos usuarios a jugar y a superarse por lo que sería interesante añadirlo a la aplicación web. Otro punto a destacar sería conseguir elementos mediante la existencia de una tienda en la propia web. También es importante la elección de los videojuegos, escogiendo juegos actuales con una amplia base de jugadores, que hará que se interesen aún más en la web. Por último, hay que remarcar que poder conocer e interactuar con otros usuarios sería un gran aporte para ellos [3] [9].

Continuando con la búsqueda de requisitos, anteriormente se ha comentado que los videojuegos elegidos para nuestro proyecto es bastante importante, ya que si cogemos algunos que sean actuales y populares entre los jugadores, nuestra página web llamará más la atención. En la investigación se ha podido ver que los juegos de *PC* son los más populares entre la comunidad y los géneros más famosos son: *Multiplayer online battle arena* o *MOBA*, *shooters* ya sean en primera o tercera persona, *battle royale*, estrategia o deportes [10]. Ejemplos de videojuegos que cumplen con estas características son: *League of Legends*, *Fortnite*, *Counter Strike 2*, *Overwatch 2*, *Dota 2* y *Call of Duty Modern Warfare 3* [10] [11]. Algunos de estos juegos los podemos ver ordenados de arriba a abajo según su relevancia, en la figura 4.1.

Figura 4.1: *E-sports* actuales

El último método que se ha usado son los cuestionarios. Este modelo nos permite saber de forma directa qué quieren y opinan los usuarios sobre determinados temas. De esta forma, al desarrollar el producto, se tendrá en cuenta para acercar al usuario aún más la página web.

Para esta aplicación web, se ha desarrollado un cuestionario de *Google* de 16 preguntas, y se ha entregado para su realización a un grupo de personas interesadas en los videojuegos y en su mundo competitivo. Las preguntas junto a sus resultados están disponibles en el apéndice A. Las cuestiones se han elaborado por grupos, es decir, que cada grupo de preguntas pertenece a una temática en concreto.

Las primeras tres preguntas del formulario, figuras A.1, A.2 y A.3, tratan sobre los videojuegos que se pondrán en la web. Como podemos ver, la mayoría de usuarios están al día sobre los videojuegos actuales, por lo que es un tema de interés en la comunidad y debe estar fácilmente accesible en la web. También, los dispositivos donde más juegan son el ordenador, el teléfono móvil y la Nintendo Switch, por lo que debemos elegir videojuegos que se ajusten a estos dispositivos. Para terminar esta parte, vemos que los aspectos de información más relevantes son: descripción, género y plataformas donde están disponibles, por lo que lo introduciremos en nuestro proyecto.

Las siguientes 4 preguntas, figuras **A.4**, **A.5**, **A.6** y **A.7**, gestionan la parte más competitiva de la web, los torneos y las clasificaciones. Como se puede observar, casi ningún usuario ha ido ni ha participado en un torneo, pero a la mitad de ellos les gustaría por lo que deberíamos dejar esa pestaña accesible en más de un sitio de la página. Después, al apuntarse a un torneo tanto si es de forma individual como en equipo, valoran que la información general, como en qué consiste o las fechas, su estado, premios y modalidad esté visible por lo que hay que integrarlo en la web. Por último, muchos de ellos conocen a algún jugador o equipo importante, lo que nos da a entender que tienen interés en este mundo.

Las preguntas de las figuras **A.8** y **A.14**, resuelven las dudas de qué información les interesa sobre los jugadores. En el entorno competitivo, a los usuarios les interesa, las victorias que han conseguido, los videojuegos donde participan, su participación en torneos y los integrantes de los equipos. Si es un entorno más casual, les interesa su información general como nombre de usuario o nacionalidad, sus estadísticas de juego, sus videojuegos y los equipos donde participan. Por todo esto, se podría crear una pestaña para información sobre jugadores.

La cuestión de la figura **A.9**, nos muestra que los usuarios no compran mucho por tiendas virtuales por lo que a la tienda no se le dará tanta importancia. De todos modos, se creará la tienda de forma fácil y accesible a los usuarios con dinero virtual que se conseguirá en la propia página.

Las próximas figuras, **A.10**, **A.11**, **A.12** y **A.13**, exponen que la mayoría de usuarios han conocido a alguien a través de Internet y que están al día de las opiniones de alguna comunidad. Además, usan bastante las redes sociales para informarse sobre este mundo, en concreto, Whatsapp e Instagram son las más recurridas. Con ello podemos concluir la introducción de un apartado que simule una red social en la página web. Por ejemplo, podríamos combinar las dos redes más utilizadas para crear unos mensajes, más propio de Whatsapp, que se puedan colgar en la red como si fueran publicaciones de Instagram. Aparte de eso, también se podría permitir personalizar el usuario mediante elementos como avatares o *banners*, haciendo que se parezca aún más a una red social.

Para finalizar este apartado, las dos últimas figuras, **A.15** y **A.16**, servirán para el diseño y creación de la aplicación web. La primera enseña que el modo oscuro es muy utilizado por los usuarios, por lo que la web podría crearse con tonalidades de ese tipo para captar más la atención. Por último, han escrito algunos problemas que suelen tener con las web. Los contratiempos que más han señalado son que algunas páginas web son lentas, la mala organización o poca claridad de alguna de ellas y que la estructura no está bien definida, por lo que podemos tomar en cuenta todo esto para que no pase en este proyecto.

4.2 Requisitos iniciales

A partir de las conclusiones sacadas en el anterior apartado **4.1**, vamos a listar los requisitos iniciales que debe tener el proyecto:

- Los nuevos usuarios se podrán registrar.
- Usuarios con una cuenta creada, podrán iniciar sesión.

- Los usuarios ya autenticados podrán cerrar su sesión.
- Se podrá ver los videojuegos soportados junto a su información.
- Los usuarios podrán ver los torneos y su información según el videojuego escogido y apuntarse a ellos.
- Se podrá ver la clasificación de los mejores jugadores y equipos según el juego escogido.
- Los usuarios podrán colgar *posts* en la parte de comunidad, filtrar los mensajes por temática y borrar los suyos propios.
- También, podrán comprar objetos en la tienda como avatares y *banners* para la personalización de usuario.
- El usuario podrá personalizar su perfil.
- El usuario tendrá la opción de ver su información personal y competitiva, además de su historial.
- Los usuarios podrán ver los equipos en los que participan y crear nuevos.
- Por último, podrán ver información de otros jugadores.

En las siguientes secciones, tendremos más en detalle estos requisitos y cómo el usuario los lleva a cabo.

4.3 Casos de uso

Ahora, antes de empezar a explicar los casos de uso uno por uno, se ha elaborado un diagrama de casos de uso en *draw.io*, figura **4.2**, que muestra de forma visual qué acciones pueden realizar los usarios. En ella podemos observar que hay dos tipos de usuarios, usuarios autenticados y no autenticados. Las líneas simples que salen desde los usuarios a los casos de uso, son los que pueden realizar estos. Al tener una flecha que va desde el usuario autenticado al que no lo está, el primero, recibe todos los métodos que hace el segundo usuario por lo que el autenticado puede realizar todas las funciones del sistema. Además de esto, tenemos dos casos más el *include* y el *extend*, el primero indica una acción que se realiza automáticamente después de otra, un ejemplo, es al hacer el registro, que el sistema iniciará sesión seguidamente. El otro caso, el *extend*, es una opción que da el sistema al realizar una acción, pero no es obligada. Por ejemplo, ver los videojuegos y su información, nos da como opción ver los torneos o la clasificación.

Antes de empezar los casos de uso, el proyecto web está pensado para que a partir de la página principal y cuatro páginas adicionales (torneos, *rankings*, comunidad y tienda), aparezca o se tenga la opción de ejecutar las funcionalidades del sistema. Muchas de estas funcionalidades se llevarán a cabo en ventanas emergentes que se irán abriendo y cerrando, por lo que cuando se mencione que se abre una ventana emergente, si hay otra activa, esta se cerrará para dar paso

Figura 4.2: Diagrama de casos de uso

a la nueva. Además, si presionamos fuera de estas ventanas, desaparecerán y el proceso parará. A continuación, vamos a enseñar los casos de uso del sistema:

Caso de uso: **Registrarse**

- **Descripción**: El usuario se puede registrar en la aplicación rellenando los datos del formulario.
- **Precondición**: El usuario no debe tener cuenta.
- **Secuencia principal**:
  1. El nuevo usuario presiona el botón de '**REGISTRARSE AHORA**' disponible en la pantalla principal.
  2. Rellena todos los campos del formulario (nombre de usuario, contraseña, correo electrónico, país, fecha de nacimiento y género) y presiona el botón de '**Crear cuenta**'.
- **Alternativas/Errores**:
  - 2.1. Si el usuario no ha rellenado todos los campos, se le avisará para que los rellene.
  - 2.2. Si el usuario ingresa un nombre de usuario que ya existe, le mostrará un error indicándoselo.

  - 2.3. Si la contraseña que pone, es de menos de 6 caracteres o no tiene mayúscula, se le avisará.
  - 2.4. Si el correo que ingresa no es un correo o ya está en la base de datos, se avisará al usuario.
  - 2.5. Si el usuario es menor de 18 años no podrá crearse una cuenta, por lo que se le avisará.
- **Postcondiciones**: Si el usuario ha rellenado correctamente todos los campos, el sistema mostrará una ventana emergente avisando de que su cuenta ha sido creada y se iniciará sesión automáticamente.

Caso de uso: **Iniciar sesión**

- **Descripción**: El usuario introduce su nombre de usuario y contraseña para acceder a su cuenta.
- **Precondición**: El usuario debe tener una cuenta.
- **Secuencia principal**:
  1. El usuario presiona el botón de '**INICIAR SESIÓN**' a la derecha del menú superior de la página.
  2. Rellena su nombre de usuario y contraseña del formulario que se ha abierto y le da al botón de '**Entrar**'.
- **Alternativas/Errores**:
  - 2.1. Si las credenciales no son correctas se le avisará al usuario.
- **Postcondiciones**: Si las credenciales son correctas, el usuario iniciará sesión y aparecerá su nombre junto a sus monedas y el perfil del usuario a la derecha del menú superior.

Caso de uso: **Cerrar sesión**

- **Descripción**: El usuario quiere cerrar su sesión.
- **Precondición**: El usuario debe estar autenticado.
- **Secuencia principal**:
  1. El usuario presiona su nombre a la derecha del menú superior y se abre el menú de usuario.
  2. En el menú de usuario, presiona el último botón del menú llamado '**Cerrar sesión**'.
- **Alternativas/Errores**: -

- **Postcondiciones**: Al cerrar sesión, volverá a salir el botón de iniciar sesión en el menú superior y el de registrarse en la página principal.

Caso de uso: **Ver los videojuegos y su información**

- **Descripción**: El usuario verá los videojuegos de la página web junto a su descripción, género, plataforma y portada.
- **Precondición**: -
- **Secuencia principal**:
  1. El usuario viaja hacia abajo en la pantalla principal y ve los juegos y su información.
- **Alternativas/Errores**:
  - 1.1. Si el usuario presiona las flechas o los puntos del carrusel puede mostrar un videojuego diferente al actual.
- **Postcondiciones**: -

Caso de uso: **Ver los torneos disponibles según el videojuego y su información**

- **Descripción**: El usuario podrá ver los torneos según el videojuego seleccionado y su información.
- **Precondición**: -
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Torneos**' disponible en el menú superior.
  2. Después, el usuario verá un botón para filtrar el videojuego del que corresponderá el torneo, por defecto es 'League of Legends', y sus respectivos torneos representados en un desplegable cerrado con el título del mismo.
  3. Si se hace *click* en uno de los torneos, se abrirá el desplegable y podremos ver un botón para inscribirse y toda la información: tipo, modalidad, lugar, jugadores inscritos, día y fecha de comienzo y recompensa.
- **Alternativas/Errores**:
  - 1.1 Se puede acceder también mediante el carrusel de videojuegos, presionando el botón '**VER TORNEOS**'. Además según el juego del carrusel, al dar al botón, filtrará por este automáticamente.

- 2.1. Si el usuario presiona el botón para filtrar por videojuego, podrá cambiar este y por ende los torneos disponibles según el juego.

- **Postcondiciones**: -

Caso de uso: **Inscribirse a torneos**

- **Descripción**: El usuario podrá inscribirse tanto individualmente como en equipo en los torneos.
- **Precondición**: El usuario debe estar autenticado, además si es un torneo en equipo, debe estar en uno correspondiente al videojuego del torneo.
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Torneos**' disponible en el menú superior.
  2. Después, el usuario verá un botón para filtrar el videojuego del que corresponderá el torneo, por defecto es 'League of Legends', y sus respectivos torneos representados en un desplegable cerrado con el título del mismo.
  3. Si se hace *click* en uno de los torneos, se abrirá el desplegable y podremos ver un botón para inscribirse.
  4. Si el tipo del torneo corresponde a '**Individual**', al presionar el botón '**INSCRIBIRSE**' aparecerá una ventana emergente preguntando si quiere inscribirse al torneo junto a dos botones para seguir o salir de la inscripción.
  5. El usuario presionará el botón '**Sí**' y le saldrá una ventana que le confirma la inscripción.
- **Alternativas/Errores**:
  - 4.1 Si el tipo corresponde a '**Equipo**', al presionar el botón '**INSCRIBIRSE**', saldrá una ventana para elegir el equipo deseado y dos botones '**Inscribirse ahora**' y '**Cancelar**'.
  - 4.2 Si el tipo corresponde a '**Equipo**' y no tiene un equipo para este videojuego, al intentar inscribirse, saldrá una ventana de error diciendo que no tiene equipos disponibles.
  - 5.1 En el caso del tipo '**Individual**', si el usuario no quiere inscribirse al final y quiere cancelar, al presionar el botón '**No**', saldrá de la ventana y el proceso se parará.
  - 5.2.1 El usuario elige el equipo y para inscribirlo, presiona el botón '**Inscribirse ahora**' y se confirma la inscripción mediante una ventana emergente.
  - 5.2.2 Si el usuario presiona el botón '**Cancelar**', la inscripción se frena y se sale de la ventana.

- **Postcondiciones**: El botón cambiará de color a un tono grisáceo y pondrá inscrito, en el caso de ser un equipo si vuelve a intentar inscribir al equipo, saldrá una ventana de error diciendo que el equipo ya ha sido inscrito.

Caso de uso: **Ver la clasificación de los mejores jugadores y equipos según el videojuego**

- **Descripción**: El usuario podrá ver unas tablas con la clasificación de los mejores jugadores y equipos en cada videojuego.
- **Precondición**: -
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Rankings**' disponible en el menú superior.
  2. Al instante, el usuario verá una tabla de clasificación con los mejores jugadores junto a su posición y sus victorias, según el videojuego elegido en el selector de este, por defecto es 'League Of Legends'. También verá un botón con dos opciones donde pone 'JUGADORES' y 'EQUIPOS', que servirá para mostrar, como dice el botón, los mejores jugadores o los mejores equipos, por defecto, está seleccionado la primera opción comentada.
- **Alternativas/Errores**:
  - 1.1 Se puede acceder también mediante el carrusel de videojuegos, presionando el botón '**VER RANKINGS**'. Además según el juego del carrusel, al dar al botón, filtrará por este automáticamente.
  - 2.1. Si el usuario presiona el botón para filtrar por videojuego, podrá cambiar este y en consecuencia la tabla de clasificación.
  - 2.2 Si el usuario presiona la parte de 'EQUIPOS' en el botón de elección entre jugadores y equipos, cambiará la tabla y en vez de indicarse los jugadores, se mostrarán los mejores equipos, mostrando también su posición y sus victorias.
- **Postcondiciones**: -

Caso de uso: **Ver *posts* de la comunidad**

- **Descripción**: El usuario podrá ver las publicaciones de la comunidad.
- **Precondición**: -
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Comunidad**' disponible en el menú superior.

2. Al momento, el usuario verá todas las publicaciones de la comunidad ordenadas de más a menos reciente, en concreto, verá el autor de estas, su contenido, su fecha de publicación y la categoría a la que pertenecen, además de un botón de añadir y filtrar *posts*.

- **Alternativas/Errores**: -
- **Postcondiciones**: -

Caso de uso: **Filtrar *posts***

- **Descripción**: El usuario podrá filtrar por categoría las publicaciones.
- **Precondición**: El usuario debe estar autenticado.
- **Secuencia principal**:

1. El usuario presiona el botón de '**Comunidad**' disponible en el menú superior.
2. Al momento, el usuario verá todas las publicaciones de la comunidad, un botón de añadir y uno de filtrar *posts*. El usuario presiona el botón '**Aplicar Filtro**'.
3. Se abrirá una ventana preguntando por cuales temáticas se quiere filtrar, se podrá elegir entre una o más y estas son: general, torneos, tienda, *rankings*, videojuegos y comunidad. También habrán dos botones '**Aplicar**' y '**Cancelar**'.
4. El usuario elegirá las temáticas que quiere y presionará el botón '**Aplicar**'.

- **Alternativas/Errores**:

  - 3.1 Si durante la elección de categorías el usuario quiere salir del proceso, presionará el botón '**Cancelar**' y saldrá del proceso.

- **Postcondiciones**: Los *posts* cambiarán a los del filtro seleccionado.

Caso de uso: **Añadir *posts***

- **Descripción**: El usuario podrá añadir *posts* en la sección de comunidad.
- **Precondición**: El usuario debe haber iniciado sesión.
- **Secuencia principal**:

1. El usuario presiona el botón de '**Comunidad**' disponible en el menú superior.
2. Al momento, el usuario verá todas las publicaciones de la comunidad, un botón de añadir y uno de filtrar *posts*. El usuario presiona el botón '**Añadir Post**'.

3. Se abre una ventana con un área de texto para escribir el contenido del *post* y unos botones para elegir la categoría del post, se podrá elegir uno o más, estas categorías son: general, torneos, tienda, *rankings*, videojuegos y comunidad. Además, habrá un botón de añadir y cancelar.
4. El usuario escribe el contenido de la publicación y elige sus categorías. Después, presiona el botón de '**Añadir**'.

- **Alternativas/Errores**:
  - 4.1 Si el usuario intenta añadir el *post* y no ha escrito nada en el contenido de este, la ventana emergente le dará un aviso comentándole el error.
  - 4.2 Si el usuario no elige al menos una categoría e intenta publicar el *post*, la ventana emergente mostrará un aviso diciendo que debe seleccionar una categoría al menos.
  - 4.3 Si el usuario presiona el botón '**Cancelar**', la ventana se cerrará y el proceso parará.
- **Postcondiciones**: El *post* del usuario se añadirá a la comunidad y saldrá en pantalla el primero al ser el más reciente.

Caso de uso: **Borrar sus *posts***

- **Descripción**: El usuario podrá borrar las publicaciones creadas por él mismo.
- **Precondición**: El usuario debe haber iniciado sesión y haber publicado al menos un *post*.
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Comunidad**' disponible en el menú superior.
  2. Al momento, el usuario verá todas las publicaciones de la comunidad, un botón de añadir y uno de filtrar *posts*. Además, podrá ver en sus propias publicaciones de la comunidad un botón rojo con forma de papelera. El usuario hace *click* en este.
  3. En consecuencia, sale una ventana emergente, preguntando si quiere borrar el *post*, con dos botones: '**Sí, elimínalo**' y '**Cancelar**'.
  4. El usuario presiona el botón que pone '**Sí, elimínalo**' y sale otra ventana anunciando que el post se ha borrado.
- **Alternativas/Errores**:
  - 4.1 Si el usuario presiona el botón de '**Cancelar**', el proceso se detiene.
- **Postcondiciones**: La publicación eliminada no sale en la comunidad.

Caso de uso: **Ver productos de la tienda**

- **Descripción**: El usuario podrá ver los artículos disponibles en la tienda.
- **Precondición**: -
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Tienda**' en el menú superior de la página.
  2. Al entrar, podrá ver que la tienda tiene dos tipos de productos: avatares y *banners*, junto a sus respectivos precios.
- **Alternativas/Errores**: -
- **Postcondiciones**: -

Caso de uso: **Comprar productos en la tienda**

- **Descripción**: El usuario podrá comprar los artículos de la tienda.
- **Precondición**: El usuario debe haber iniciado sesión y tener monedas suficientes.
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Tienda**' en el menú superior.
  2. Al entrar, podrá ver dos tipos de productos: avatares y *banners*, junto a sus precios contenidos en un botón cada uno.
  3. El usuario presiona un botón de estos para comprar un producto.
  4. Al instante, sale una ventana emergente, preguntando si quiere comprar el objeto, junto a dos botones: '**Sí, comprar**' y '**Cancelar**'.
  5. El usuario presiona el botón de '**Sí, comprar**' y sale una ventana de confirmación de compra.
- **Alternativas/Errores**:
  - 5.1 Si el usuario presiona el botón de '**Sí, comprar**' y no tiene suficiente dinero (se puede ver a la izquierda del nombre del usuario en el menú superior), le saldrá una ventana emergente de error, diciendo que no tiene suficiente dinero.
  - 5.2 Si el usuario presiona el botón de '**Cancelar**', el proceso de comprar el artículo se detiene y se sale de la ventana emergente.
- **Postcondiciones**: El producto se compra y donde estaba el precio del artículo, ahora sale un símbolo que muestra que el producto ha sido comprado.

Caso de uso: **Ver información de su perfil**

- **Descripción**: El usuario podrá ver la información de su perfil (información personal, competitiva y sus videojuegos).
- **Precondición**: El usuario debe haber iniciado sesión.
- **Secuencia principal**:
  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.
  2. El usuario presiona el botón de '**Perfil**' y se abre una ventana emergente donde se puede ver toda la información del usuario. Esta información será: Información personal (nombre de usuario, país, fecha de nacimiento, correo y género), información competitiva (torneos jugados y ganados) y sus videojuegos.
- **Alternativas/Errores**: -
- **Postcondiciones**: -

Caso de uso: **Cambiar videojuegos del usuario**

- **Descripción**: El usuario podrá actualizar su biblioteca de videojuegos.
- **Precondición**: El usuario debe haber iniciado sesión.
- **Secuencia principal**:
  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.
  2. El usuario hace *click* en el botón de '**Perfil**' y se abre una ventana emergente donde se puede ver toda la información del usuario, en la parte final de esta información, verá una sección de sus videojuegos donde puede elegir cuales videojuegos de los que da soporte la página tiene en su biblioteca. Además de un botón llamado '**Actualizar Videojuegos**'.
  3. El usuario elige los videojuegos que tiene, presiona el botón y sale otra ventana emergente anunciando que la lista de juegos ha sido actualizada.
- **Alternativas/Errores**: -
- **Postcondiciones**: Cuando se vuelve a abrir el perfil del usuario, los videojuegos que haya seleccionado antes, se mostrarán con un *checkbox* activado y los que no, será uno igual pero desactivado.

Caso de uso: **Personalizar perfil**

- **Descripción**: El usuario tendrá la opción de personalizar su perfil mediante avatares y *banners*.
- **Precondición**: El usuario debe haber iniciado sesión y preferiblemente haber comprado algún objeto en la tienda para que pueda cambiar el avatar y el *banner* predeterminados.
- **Secuencia principal**:
  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.
  2. El usuario accede al botón de '**Personalización**' y se abre una ventana emergente donde sale el avatar y el banner en uso por el usuario y dos botones para cambiar estos.
  3. Presiona el botón '**Cambiar avatar**' y le sale otra ventana emergente con los avatares que tiene, poniendo el borde en verde en el que usa actualmente y un botón de '**Confirmar**' y otro de '**Cancelar**'.
  4. El usuario elige el avatar que quiere usar y le da al botón de '**Confirmar**'. La acción se ejecuta y se confirma mediante otra ventana emergente que anuncia que ha sido actualizado.
- **Alternativas/Errores**:
  - 3.1 Presiona el botón '**Cambiar banner**' y se abre otra ventana con todos los avatares que tiene el usuario y el que está en uso con borde verde, junto a los botones de confirmar y cancelar.
  - 4.1 Si estamos con los *banners*, el usuario selecciona el deseado y presiona el botón '**Confirmar**' y la acción se ejecuta y se confirma mediante una ventana emergente que anuncia la actualización hecha.
  - 4.2 Si el usuario presiona el botón '**Cancelar**', en los dos casos, la ventana se cerrará y el proceso parará.
- **Postcondiciones**: El avatar o el *banner* se cambian y cuando entras al apartado de personalización el avatar y *banner* en uso se han cambiado a los escogidos.

Caso de uso: **Ver sus equipos y su información**

- **Descripción**: El usuario podrá ver los equipos a los que pertenece junto a información de estos.
- **Precondición**: El usuario debe haber iniciado sesión y pertenecer a un equipo.

- **Secuencia principal**:

  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.
  2. El usuario accede al botón de '**Equipos**' y se abre una ventana emergente donde salen dos botones: '**Mis equipos**' y '**Crear equipo**'.
  3. Presiona el botón '**Mis equipos**' y se abre otra ventana emergente con un selector donde puede elegir entre todos sus equipos y dos botones: '**Buscar**' y '**Cancelar**'.
  4. El usuario elige uno de sus equipos y presiona el botón '**Buscar**'. Al instante, sale una ventana emergente con el nombre del equipo, sus participaciones, victorias, su videojuego y sus integrantes. Además salen dos botones para administrar el equipo y el botón '**OK**' para cerrar la ventana.
  5. El usuario mira la información del equipo y presiona el botón '**OK**' para acabar con el proceso.

- **Alternativas/Errores**:

  - 3.1 Si el usuario no tiene equipos, se abrirá una ventana emergente de error, comunicando al usuario que no tiene equipos.
  - 4.1 Si el usuario presiona '**Cancelar**', se sale de la ventana y el proceso para.

- **Postcondiciones**: -

Caso de uso: **Crear equipos**

- **Descripción**: El usuario podrá crear un equipo nuevo.
- **Precondición**: El usuario debe haber iniciado sesión.
- **Secuencia principal**:

  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.
  2. El usuario accede al botón de '**Equipos**' y se abre una ventana emergente donde salen dos botones: '**Mis equipos**' y '**Crear equipo**'.
  3. Presiona el botón '**Crear equipo**' y se abre otra ventana emergente con un formulario, donde se debe poner el nombre del equipo y el videojuego del equipo. Y dos botones: '**Crear equipo**' y '**Cancelar**'.
  4. El usuario rellena todos los campos y presiona el botón '**Crear equipo**', se realiza la acción y se abre una ventana emergente confirmando la acción.

- **Alternativas/Errores**:
  - 4.1 Si el usuario no rellena todos los campos, al darle al botón '**Crear equipo**', saldrá un aviso para que rellene todos los campos.
  - 4.2 Si el nombre de equipo que pone el usuario ya está en uso, al presionar '**Crear equipo**', saldrá un aviso de que el nombre de equipo ya está en uso.
  - 4.3 Si el usuario presiona '**Cancelar**', la ventana se cierra y el proceso para.
- **Postcondiciones**: El equipo se crea y si entras a la opción de '**Mis equipos**', en el menú de equipos, saldrá en el selector.

Caso de uso: **Abandonar equipos**

- **Descripción**: El usuario podrá abandonar un equipo.
- **Precondición**: El usuario debe haber iniciado sesión y debe pertenecer a un equipo.
- **Secuencia principal**:
  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.
  2. El usuario accede al botón de '**Equipos**' y se abre una ventana emergente donde salen dos botones: '**Mis equipos**' y '**Crear equipo**'.
  3. Presiona el botón '**Mis equipos**' y se abre otra ventana emergente con un selector donde puede elegir entre todos sus equipos y dos botones: '**Buscar**' y '**Cancelar**'.
  4. El usuario elige uno de sus equipos y presiona el botón '**Buscar**'. Al instante, sale una ventana emergente con la información del equipo y un botón para cerrar la ventana llamado '**OK**' y dos botones para administrar el equipo, uno verde llamado '**Invitar a un jugador**' y otro rojo de nombre '**Abandonar equipo**'.
  5. El usuario presiona el botón '**Abandonar equipo**', se realiza la acción y sale una ventana emergente confirmando que el usuario lo ha abandonado.
- **Alternativas/Errores**: -
- **Postcondiciones**: El usuario abandona el equipo y si entra al apartado de sus equipos, ya no se encuentra en el selector.

Caso de uso: **Invitar a jugadores**

- **Descripción**: El usuario podrá invitar a nuevos usuarios a sus equipos.
- **Precondición**: El usuario debe haber iniciado sesión y debe pertenecer a un equipo.
- **Secuencia principal**:
  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.
  2. El usuario accede al botón de '**Equipos**' y se abre una ventana emergente donde salen dos botones: '**Mis equipos**' y '**Crear equipo**'.
  3. Presiona el botón '**Mis equipos**' y se abre otra ventana emergente con un selector donde puede elegir entre todos sus equipos y dos botones: '**Buscar**' y '**Cancelar**'.
  4. El usuario elige uno de sus equipos y presiona el botón '**Buscar**'. Al instante, sale una ventana emergente con la información del equipo y un botón para cerrar la ventana llamado '**OK**' y dos botones para administrar el equipo, uno verde llamado '**Invitar a un jugador**' y otro rojo de nombre '**Abandonar equipo**'.
  5. El usuario presiona el botón '**Invitar a un jugador**', y sale un selector para seleccionar al usuario que se quiere invitar y dos botones: '**Invitar**' y '**Cancelar**'.
  6. El usuario selecciona el usuario al que quiere invitar y presiona el botón '**Invitar**'. Al hacerlo, se hará la acción y sale una ventana emergente anunciando que has invitado al jugador.
- **Alternativas/Errores**:
  - 6.1 Si el usuario presiona '**Cancelar**', el proceso parará y se saldrá de la pantalla.
- **Postcondiciones**: Al usuario invitado, le saldrá la invitación en la primera ventana de la parte de '**Equipos**' disponible en el menú de usuario. Además, no aparecerá otra vez en el selector de invitación al equipo.

Caso de uso: **Aceptar/Rechazar invitación**

- **Descripción**: El usuario podrá aceptar o rechazar las invitaciones a los equipos.
- **Precondición**: El usuario debe haber iniciado sesión y debe haber sido invitado a algún equipo.
- **Secuencia principal**:
  1. El usuario presiona su nombre a la derecha del menú superior y se abre un desplegable con cuatro botones: '**Perfil**', '**Equipos**', '**Personalización**' y '**Cerrar sesión**'.

2. El usuario accede al botón de '**Equipos**' y se abre una ventana emergente donde salen dos botones: '**Mis equipos**' y '**Crear equipo**'. Si el usuario ha sido invitado a algún equipo, aparecerá una sección abajo llamada '**Invitaciones**'. En ella aparecerán los nombres de los equipos a los que se les ha invitado junto a un botón verde con un símbolo de confirmar y otro rojo con un símbolo de rechazar.
3. Si el usuario quiere aceptar, presiona el botón verde y se realiza la acción de entrar en el equipo. Todo esto se confirma mediante una ventana emergente que confirma la acción.

- **Alternativas/Errores**:
  - 3.3 Si el usuario por otro lado quiere rechazar la invitación, presionará el botón rojo y no entrará en el equipo. También aparece una ventana confirmando que no se ha aceptado la invitación.
- **Postcondiciones**: Si el usuario acepta, figurará como uno de sus integrantes y si entra en la parte de sus equipos, lo verá.

Caso de uso: **Ver información de otros jugadores**

- **Descripción**: El usuario podrá ver información de otros jugadores.
- **Precondición**: No hay precondición como tal, ya que cualquier usuario puede hacerlo, pero hay determinadas ventanas que no se pueden acceder sin sesión donde se puede hacer esta acción también.
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Rankings**' disponible en el menú superior.
  2. Al realizar esta acción se ven la clasificación de los mejores jugadores que se compone de la posición, nombre de usuario y victorias. Para ver la información de otros usuarios, el usuario actual hace *click* el nombre de uno de estos jugadores.
  3. Al presionarlo, sale una ventana con toda la información de ese usuario. Esta información se compone de: su nombre de usuario, información personal (fecha de nacimiento, género y país), información competitiva (torneos jugados y ganados y equipos a los que pertenece) y los videojuegos de su biblioteca. Además del avatar y *banner* que usa actualmente.
- **Alternativas/Errores**:
  - 1.1 El usuario entra en uno de sus equipos en la parte de '**Mis equipos**' disponible en la sección de '**Equipos**' del menú de usuario que está disponible al iniciar sesión, cuando presiona su nombre en el menú superior.

- 2.1 Siguiendo con el 1.1. En la parte de información de ese equipo, están los integrantes y le da *click* a uno de ellos para ver su información.
- 1.2 El usuario presiona el botón de '**Comunidad**' disponible en el menú superior.
- 2.2 Después del 1.2, se pueden ver todas las publicaciones de la comunidad junto con el nombre del usuario que las ha hecho. El usuario actual, entra a ver la información de un usuario al darle al nombre de uno de los publicadores.
- 1.3 Si el usuario está viendo información de otro equipo, si este quiere estudiar a sus integrantes, presiona uno de sus nombres.

- **Postcondiciones**: -

Caso de uso: **Ver información de otros equipos**

- **Descripción**: El usuario podrá ver información de otros equipos.
- **Precondición**: No hay precondición como tal, ya que cualquier usuario puede hacerlo, pero hay determinadas ventanas que no se pueden acceder sin sesión donde se puede hacer esta acción también.
- **Secuencia principal**:
  1. El usuario presiona el botón de '**Rankings**' disponible en el menú superior.
  2. Al realizar esta acción se ven la clasificación de los mejores jugadores junto a un botón para cambiar el videojuego y otro para cambiar de jugadores a equipos. El usuario presiona este último y se listan los mejores equipos, donde se indica: la posición, el nombre del equipo y las victorias. Para ver la información de uno de estos, el usuario aprieta uno de sus nombres.
  3. Al hacerlo, sale una ventana donde se indica: el nombre del equipo, el videojuego donde compite, los torneos jugados y ganados y los integrantes.
- **Alternativas/Errores**:
  - 1.1 El usuario entra en la sección de '**Equipos**' del menú de usuario disponible al presionar su nombre en el menú superior de la página, siempre que haya iniciado sesión.
  - 2.1 Siguiendo con el 1.1. Si el jugador ha sido invitado, saldrá una sección de '**Invitaciones**' donde aparece el nombre del equipo y los botones para aceptar o rechazar la petición. Si antes de entrar en él, quiere conocer el equipo, el usuario aprieta el nombre del equipo.
  - 1.2 Ahora, si el usuario está viendo el perfil de otro usuario, verá los equipos a los que pertenece. El usuario presiona uno de estos nombres de equipo y se abre la información del equipo.
- **Postcondiciones**: -

CAPÍTULO 5
Análisis conceptual y diseño

Una vez explicado cómo funciona y que posibilidades ofrece nuestra aplicación web, nos vamos a ocupar de definir la parte no visible que hace que todas las funcionalidades definidas anteriormente funcionen correctamente. Para ello, en primer lugar se mostrará el diagrama de clases que define los objetos de la web y sus relaciones, en segundo lugar, se mostrará la base de datos utilizada y por último los bocetos que se han realizado previamente a la construcción del portal web para formar el aspecto de este.

5.1 Diagrama de clases

Como se ha indicado, se va a comenzar dando un vistazo al diagrama de clases, hecho con *draw.io*.

En la figura **5.1** y **5.2**, podemos ver el diagrama en cuestión separado para visualizarlo mejor. En ellas se pueden ver las clases que lo componen y sus relaciones entre ellas, además de los propios atributos de cada uno, donde cada una de estas clases representa los objetos que se manejan en la web.

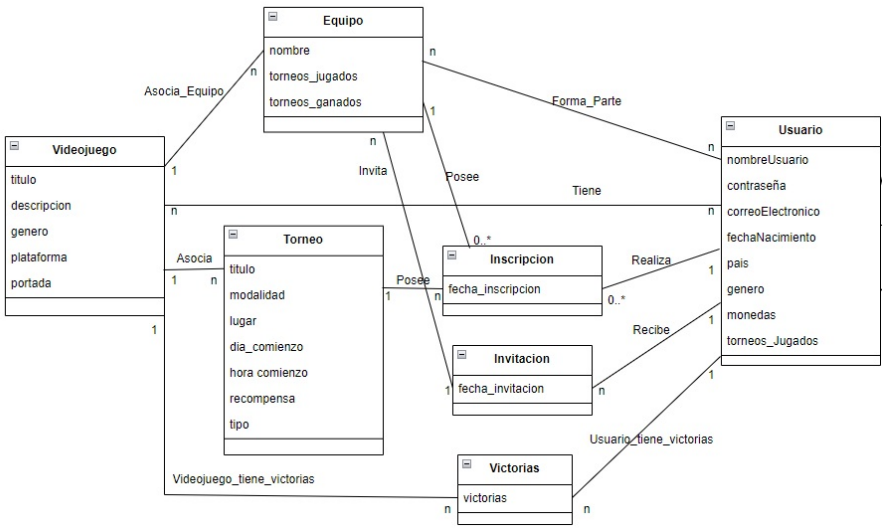

Figura 5.1: Diagrama de clases parte 1

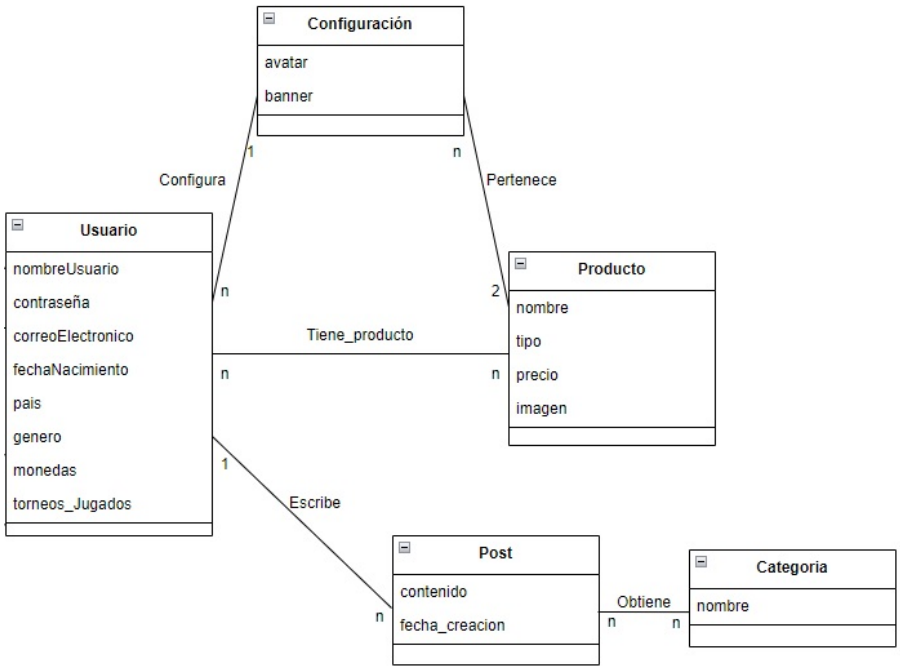

Figura 5.2: Diagrama de clases parte 2

5.2 Modelo de base de datos

Ahora, se mostrará la base de datos. Primero, se ha elegido el sistema de gestión de base de datos. En este caso, se ha optado por *PostgreSQL*, un gestor de base de datos relacionales y orientado a objetos. Las razones por las que se ha cogido son: es de código abierto, por lo que no está bajo el control de ninguna entidad, su fácil configuración, funciona con *SQL* y su gran comunidad [12].

Después de elegir el gestor de base de datos, hay que elegir la mejor herramienta que nos facilite el uso del gestor. La elección para el proyecto web ha sido *DBeaver*, una herramienta gratuita y multiplataforma de código abierto. Esta herramienta nos proporciona muchas posibilidades, una visión clara y rápida de todos los datos, tablas y relaciones y por último, al ser de código abierto, es una buena opción para nuestro proyecto [13]. También, pese a que la interfaz aporta muchas opciones y puede resultar algo abrumadora, al tener experiencia con la herramienta no será un problema. Por último, la visualización de datos en las tablas se muestra de forma clara. Aquí tenemos la interfaz, figura 5.3.

A continuación, vamos a detallar cada tabla de la base de datos y sus atributos o columnas:

La tabla '**usuarios**', hace referencia a los usuarios de la página.

- '**id**': Número de identificación numérico automático para los usuarios.
- '**nombreusuario**': Nombre del usuario.
- '**contraseña**': La contraseña encriptada del usuario.
- '**correoelectronico**': El correo del usuario.

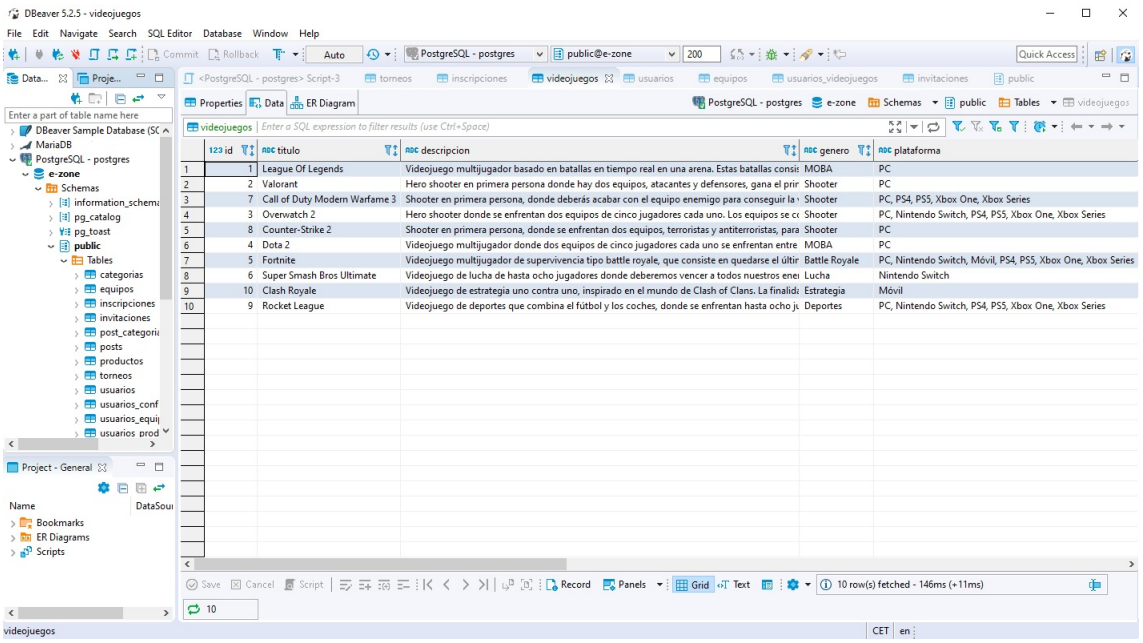

Figura 5.3: Interfaz *DBeaver*

- '**fechanacimiento**': La fecha de nacimiento del usuario.
- '**pais**': El país de donde es el usuario.
- '**genero**': El género del usuario.
- '**monedas**': Las monedas actuales del usuario.
- '**torneos_jugados**': Los torneos jugados por el usuario.

La tabla '**videojuegos**', hace referencia a los videojuegos elegidos para la página.

- '**id**': Número de identificación numérico automático para los videojuegos.
- '**titulo**': Título del videojuego.
- '**descripcion**': Descripción breve del videojuego.
- '**genero**': El género del videojuego.
- '**plataforma**': Lista de plataformas donde está disponible el videojuego.
- '**portada**': Una cadena de caracteres en base 64 donde está guardada la imagen de la portada.

La tabla '**productos**', hace referencia a los productos disponibles.

- '**id**': Número de identificación numérico automático para los productos.

- '**nombre**': Nombre del producto.
- '**tipo**': El tipo del producto, actualmente solo hay avatares y *banners*.
- '**precio**': El precio del producto.
- '**imagen**': Cadena de caracteres en base 64 donde se guarda la imagen del producto.

La tabla '**torneos**', hace referencia a los torneos.

- '**torneo_id**': Número de identificación numérico automático para los torneos.
- '**titulo**': Título del torneo.
- '**videojuego_id**': El *id* del videojuego del torneo.
- '**modalidad**': La modalidad del torneo.
- '**lugar**': El lugar donde se hace el torneo, actualmente son todos *online*.
- '**dia_comienzo**': El día de comienzo del torneo.
- '**hora_comienzo**': La hora de comienzo del torneo.
- '**recompensa**': La recompensa por ganar el torneo.
- '**tipo**': Se refiere a qué si el torneo es individual o en equipo.

La tabla '**equipos**', hace referencia a los equipos hechos por los usuarios.

- '**id**': Número de identificación numérico automático para los equipos.
- '**nombre**': Nombre del equipo.
- '**videojuego_id**': El id del videojuego del equipo.
- '**torneos_jugados**': Los torneos jugados por el equipo.
- '**torneos_ganados**': Los torneos ganados por el equipo.

La tabla '**posts**', hace referencia a las publicaciones de la comunidad.

- '**id**': Número de identificación numérico automático para las publicaciones.
- '**contenido**': Contenido de los *posts*.
- '**autor_id**': El *id* del usuario que lo ha publicado.
- '**fecha_creacion**': Fecha de cuando se publicó el *post*.

La tabla '**inscripciones**', hace referencia a las inscripciones a los torneos por parte de usuarios o equipos.

- '**id**': Número de identificación numérico automático para la inscripción al torneo.
- '**usuario_id**': *Id* del usuario inscrito siempre que el tipo del torneo sea 'Individual', sino será *NULL*.
- '**equipo_id**': El *id* del equipo inscrito siempre que el tipo del torneo sea 'Equipo', sino será *NULL*.
- '**torneo_id**': *Id* del torneo al que se relaciona la inscripción.
- '**fecha_inscripcion**': La fecha en la que se ha inscrito el usuario o equipo al torneo.

La tabla '**invitaciones**', hace referencia a las invitaciones hechas por los equipos para invitar a unirse a nuevos usuarios.

- '**id**': Número de identificación numérico automático para la invitación al equipo.
- '**equipo_id**': *Id* del equipo que está invitando.
- '**usuario_id**': El *id* del usuario que está siendo invitado a unirse.
- '**fecha_invitacion**': La fecha en la que se ha enviado la invitación.

La tabla '**categorias**', hace referencia a los tipos de categorías que pueden tener los *posts*.

- '**id**': Número de identificación numérico automático para el tipo de categoría.
- '**nombre**': Nombre de la categoría.

La tabla '**usuarios_videojuegos**', hace referencia a los videojuegos que tiene un usuario. Un usuario puede tener más de un videojuego por lo que los dos *id* que hacen referencia a los usuarios y videojuegos de sus respectivas tablas, forman la clave primaria.

- '**usuario_id**': El *id* del usuario al que se hace referencia.
- '**videojuego_id**': El *id* del videojuego al que se hace referencia.

La tabla '**usuarios_productos**', hace referencia a los productos que tiene un usuario. Un usuario puede tener más de un producto por lo que los dos *id* que hacen referencia a los usuarios y productos de sus respectivas tablas, forman la clave primaria.

- '**usuario_id**': El *id* del usuario al que se hace referencia.
- '**producto_id**': El *id* del producto al que se hace referencia.

La tabla '**usuarios_configuracion**', hace referencia a la configuración que tiene un usuario. Un usuario tiene como configuración, un avatar y un *banner*, por defecto tiene los base, por lo que siempre tiene una configuración activa. Los dos objetos mencionados, son productos, pertenecientes a la tabla del mismo nombre, por lo que se relacionará dos veces con esta tabla para sacar sus *id* y luego con la tabla usuario, ya que un usuario tiene una configuración. Por esto último, la única clave primaria en este caso es el *id* del usuario.

- '**usuario_id**': El *id* del usuario al que se hace referencia.
- '**avatar_id**': El *id* del producto de tipo 'Avatar' al que se hace referencia.
- '**banner_id**': El *id* del producto de tipo 'Banner' al que se hace referencia.

La tabla '**usuarios_equipos**', hace referencia a los equipos a los que pertenece un usuario. Un usuario puede estar en más de un equipo por lo que los dos *id* que hacen referencia a los usuarios y equipos de sus respectivas tablas, forman la clave primaria.

- '**usuario_id**': El *id* del usuario al que se hace referencia.
- '**equipo_id**': El *id* del equipo al que se hace referencia.

La tabla '**victoriastorneo_videojuego_usuario**', hace referencia a las victorias que tiene un usuario dentro de un videojuego. Un usuario puede tener en varios videojuegos un distinto número de victorias, por lo que los dos *id* que hacen referencia a los usuarios y videojuegos de sus respectivas tablas, forman la clave primaria.

- '**usuario_id**': El *id* del usuario al que se hace referencia.
- **'videojuego_id**': El *id* del videojuego al que se hace referencia.
- '**victorias**': El número de victorias del usuario en un videojuego.

La tabla '**post_categorias**', hace referencia a las categorías que tiene un *post*. Un usuario puede tener en varias categorías dentro de una misma publicación, por lo que los dos *id* que hacen referencia a los *posts* y categorías de sus respectivas tablas, forman la clave primaria.

- '**post_id**': El *id* del *post* al que se hace referencia.
- '**categoria_id**': El *id* de la categoría a la que se hace referencia.

Podemos ver el resultado de todas las tablas en el diagrama de clases hecho por el programa de la base de datos, figura **5.4**.

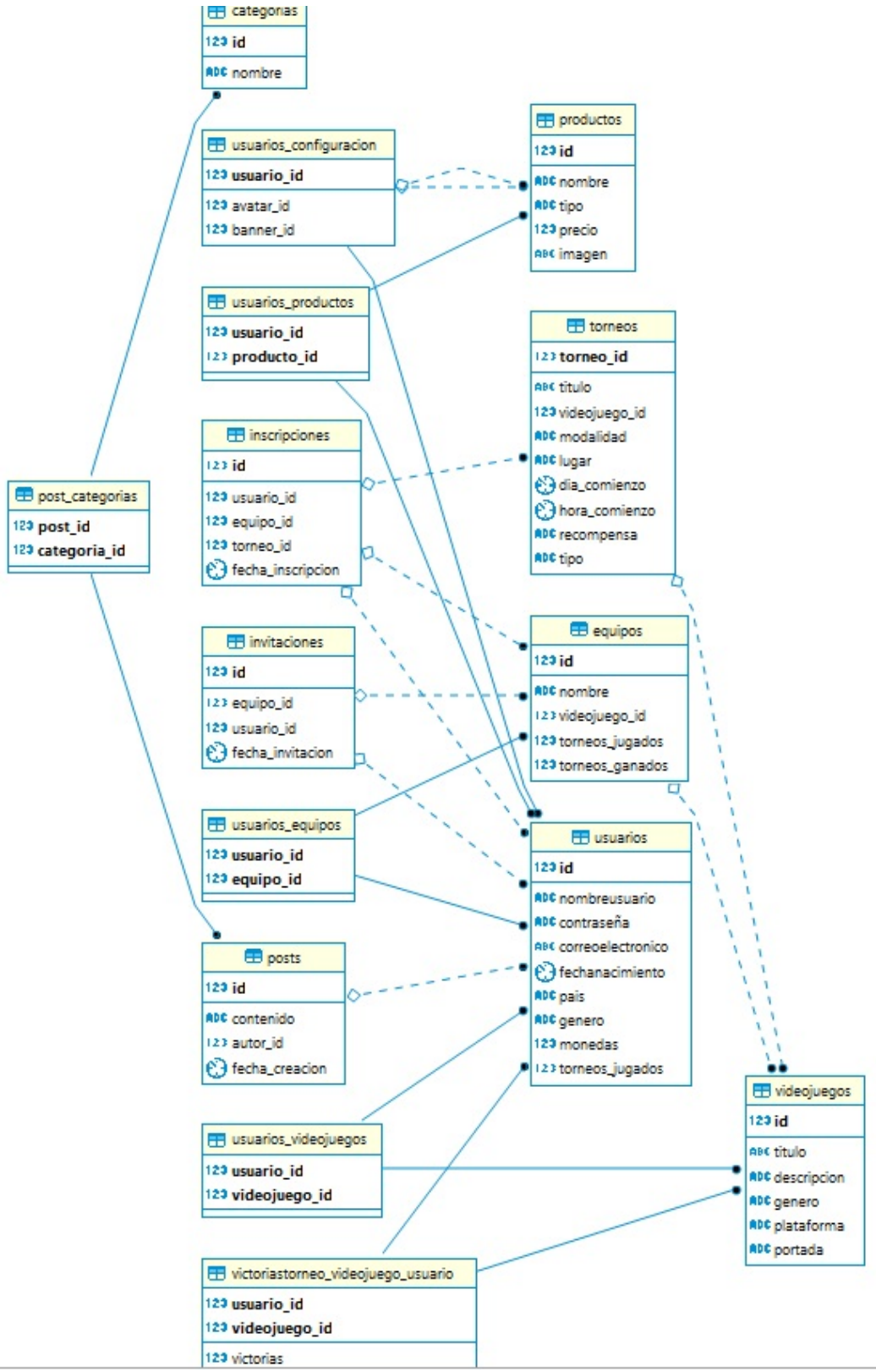

Figura 5.4: Diagrama de clases de la base de datos

5.3 Bocetos de las interfaces

Para terminar este capítulo, se comentarán y explicarán todos los aspectos de diseño y los bocetos de las interfaces realizadas para el proyecto web.

Antes de realizar los bocetos, hay que tener en cuenta que las interfaces deben ser sencillas de usar y el estilo que tengan debe ir a favor del usuario, para no confundir ni estropear la experiencia de este. También, se han diseñado mensajes de retroalimentación que saldrán después de realizar determinadas acciones para que el usuario conozca si se ha producido o no un error. Además, pese a que ahora solo se presenten los bocetos para la web, las interfaces deben ser responsivas y adaptables a cualquier tamaño o dispositivo. Por último, aunque las tonalidades de los colores puedan cambiar en un futuro, la idea es realizar los bocetos con colores oscuros como el negro y el gris junto a otros colores que los complementen como el verde y el blanco, ya que en la fase de requisitos, se detectó que los usuarios usaban frecuentemente el modo oscuro, sección 4.1.

El primer boceto que se va a mostrar no es de una página, sino de un elemento que ayudará a identificar el proyecto y que es esencial, el **logo**, figura 5.5. Como podemos ver, el nombre del proyecto web es ***E-ZONE***, una combinación de palabras entre zona en inglés y *e-sports* y a su lado un icono de un mando típico de videojuegos con una llama detrás, representados con los colores verde, blanco y azul oscuro, respetando la temática cromática descrita anteriormente.

Figura 5.5: Logo del proyecto web

Ahora, se van a empezar a mostrar las interfaces creadas para la plataforma web. La herramienta web que se ha usado es *Moqups*, una página que permite entre otras cosas crear prototipos de interfaces. Como ya se mencionó anteriormente, el proyecto tiene la página principal y otras cuatro páginas adicionales (torneos, *rankings*, comunidad y tienda), que tendrán cada una sus respectivos bocetos de interfaces. El resto de bocetos corresponderán a ventanas emergentes que se van abriendo para realizar y confirmar acciones o comunicar fallos de determinadas acciones. Estas ventanas se colocarán encima de la página que se esté usando en ese momento (una de las cinco mencionadas), en mitad de la pantalla, por lo que en los bocetos solo se mostrará la ventana emergente y un marco de color negro claro simulando la página que queda detrás.

La primera que se va a mostrar es la **página principal** tanto con la **sesión iniciada**, figura 5.7, como **sin ella**, figura 5.6. Esta será la única que se mostrará de dos maneras, ya que de esta forma podemos ver de una vez el botón de iniciar sesión y registrarse cuando la sesión no está iniciada y es donde más diferencias podemos ver entre una página y otra, ya que en el resto de páginas secundarias solo hay cambios menores.

Lo común entre las dos páginas iniciales, es el diseño del menú superior y sus cuatro botones para ir a otras partes de la página, la parte de videojuegos competitivos que constará de un carrusel con información de estos, donde si se presiona alguna de las flechas o círculos se cambiará de videojuego. Por último, vemos una sección de información para animar a los usuarios a empezar su camino en la página.

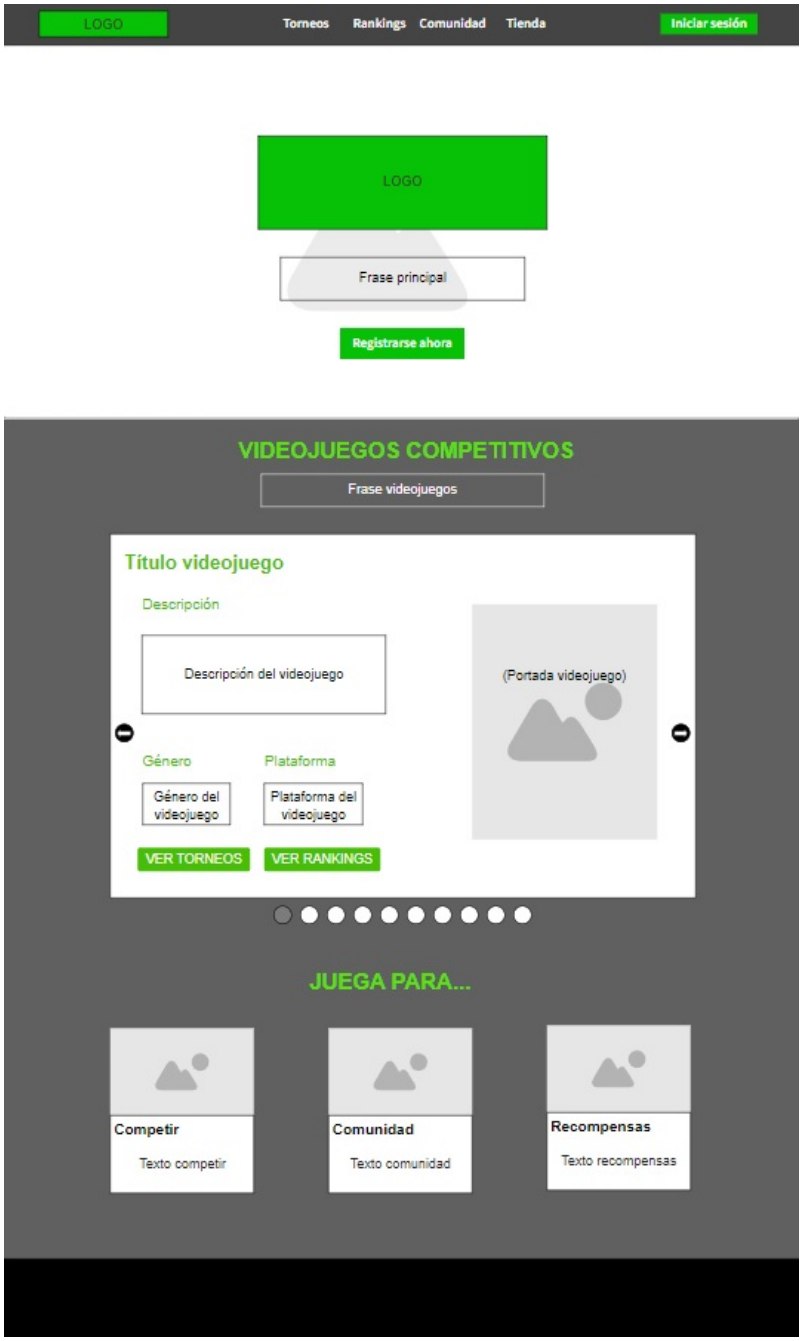

Figura 5.6: Boceto página principal sin sesión iniciada

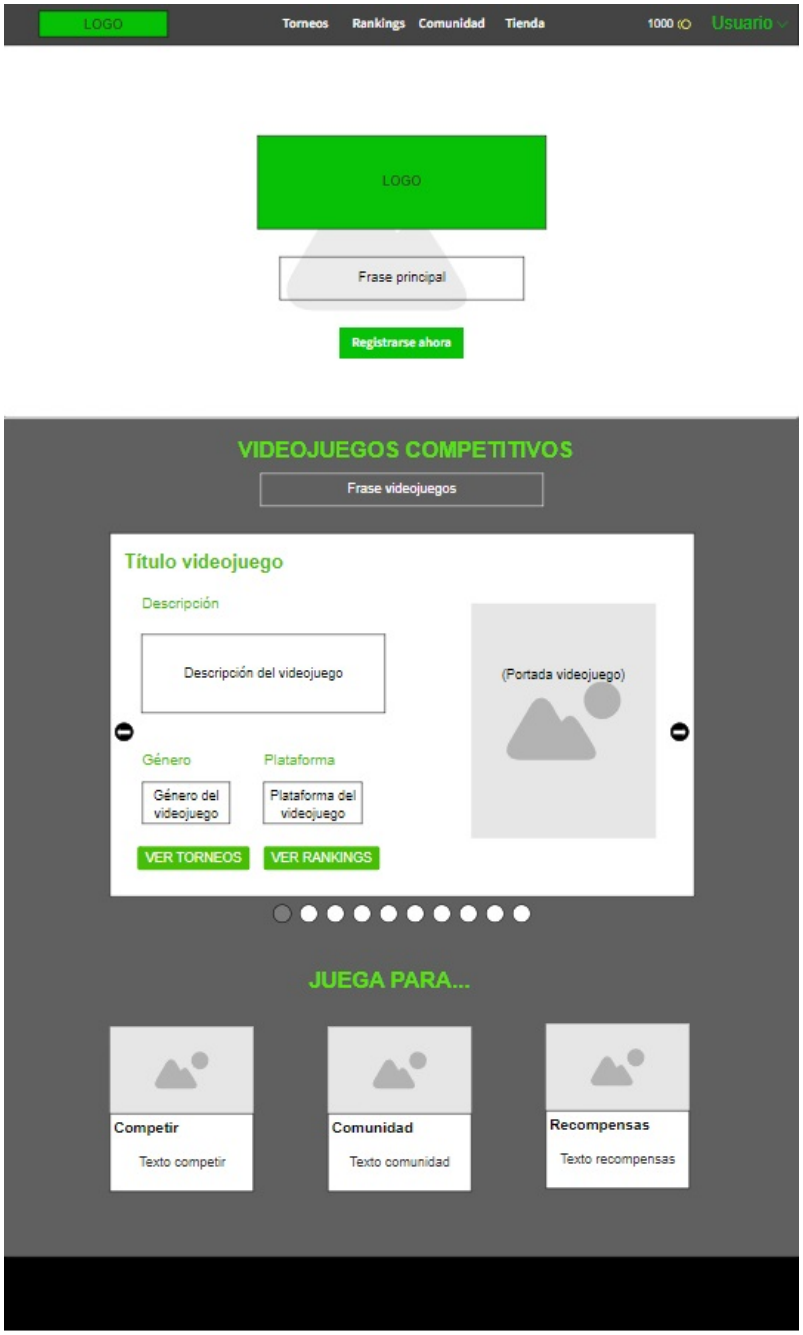

Figura 5.7: Boceto página principal con sesión iniciada

A partir de ahora, el resto de páginas tendrán la sesión iniciada. Si se presiona el nombre del usuario del menú superior, aparecerá el **menú de usuario**, aquí su boceto, figura 5.8. El siguiente boceto corresponde con el de **torneos**, figura 5.9, donde podemos ver los torneos disponibles actualmente con toda su información según el videojuego que se elija en el selector.

Figura 5.8: Boceto menú de usuario

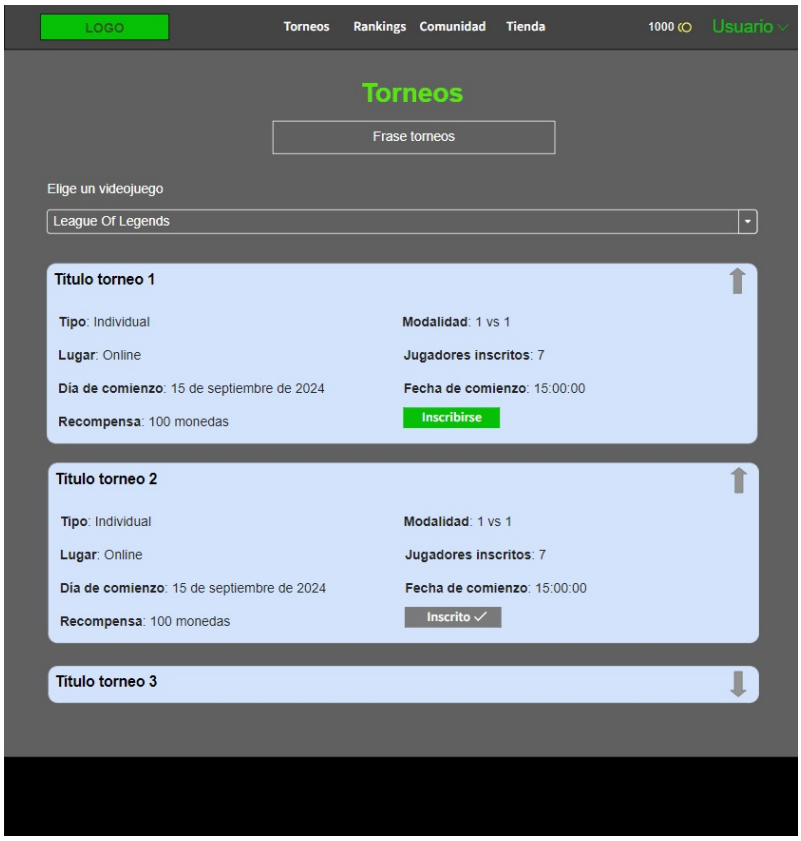

Figura 5.9: Boceto página torneos

El próximo es el de ***rankings***, figura 5.10, donde podemos ver los mejores jugadores según el videojuego elegido y si se pulsa en equipos, se seleccionará este y en vez de jugadores, serán equipos los que salgan en la pantalla.

Ahora, se presenta el de **comunidad**, figura 5.11, donde están las publicaciones de cada usuario, junto a los botones de filtrar y añadir *post*, además si la publicación es tuya, saldrá *Tú* en vez del nombre del usuario y una papelera roja que sirve de botón para borrar la publicación.

Y por último, el boceto de la parte de **tienda**, figura 5.12, en ella vemos los productos disponibles que son avatares y *banners* con su precio cada uno. Si uno de los productos ya está comprado, saldrá con una marca como se ve en el segundo avatar de la primera fila.

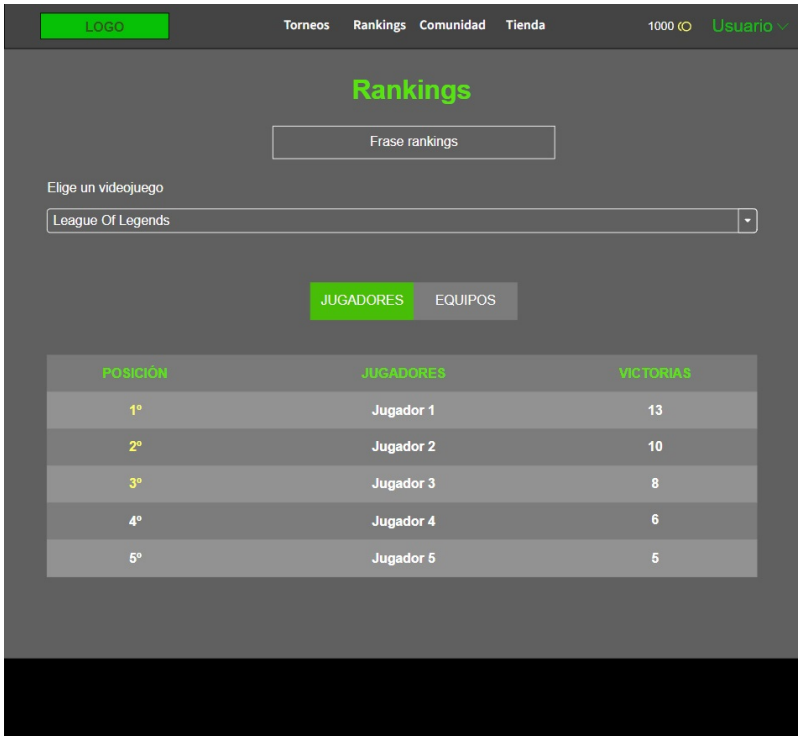

Figura 5.10: Boceto página *rankings*

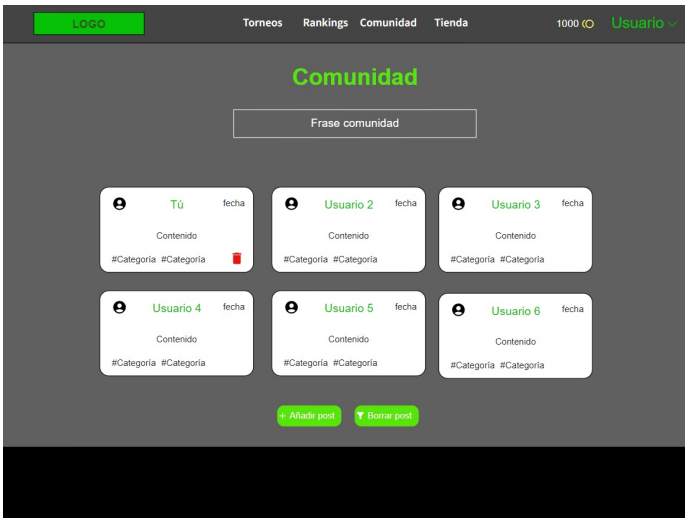

Figura 5.11: Boceto página comunidad

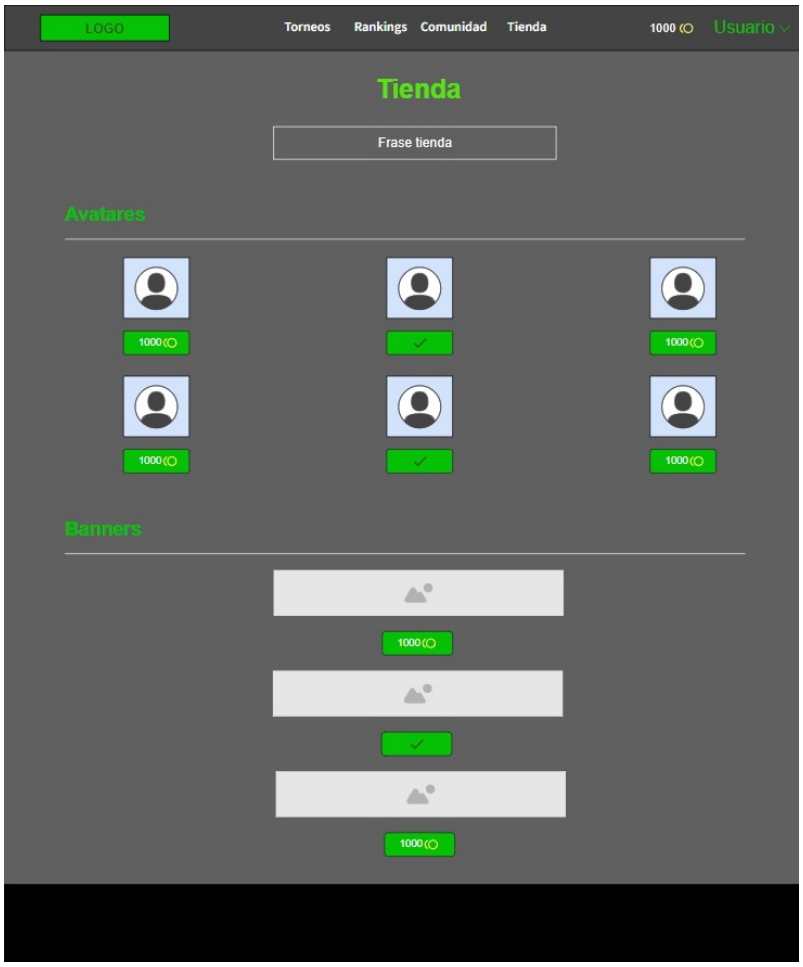

Figura 5.12: Boceto página tienda

A continuación, se presentarán los bocetos de las **ventanas emergentes**. Los primeros serán los de **iniciar sesión**, figura 5.13 y **registrarse**, figura 5.14. Donde en cada uno se verán unos campos a rellenar y sus botones para ejecutar la acción.

Figura 5.13: Boceto iniciar sesión

Figura 5.14: Boceto registrarse

Ahora, viene un boceto que nos servirá para varias partes del proyecto, figura 5.15. Se usará para preguntar al usuario sobre la **confirmación o no de diferentes acciones**. Después tenemos la ventana para **inscribir un equipo al torneo**, figura 5.16, la de **añadir un *post***, figura 5.17 y la de **filtrar un *post***, figura 5.18.

Figura 5.15: Boceto preguntas

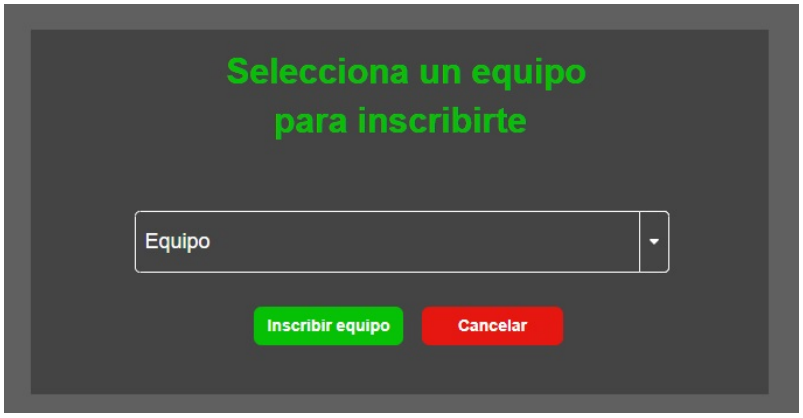

Figura 5.16: Boceto inscribir equipo al torneo

Figura 5.17: Boceto añadir *post*

Figura 5.18: Boceto filtrar *post*

Para continuar, tenemos los bocetos de las ventanas relacionados con el menú del propio usuario. Primero tenemos el **perfil del usuario**, figura 5.19, donde podemos ver su información y la opción de actualizar su biblioteca de videojuegos. Luego, tenemos el **menú de equipos**, donde el usuario puede elegir si ver sus equipos o crear uno nuevo, figura 5.20. Si elige **crear uno nuevo**, este es el boceto, figura 5.21, de la ventana que saldrá para elegir el nombre y el videojuego del equipo. Por otro lado, si elige **ver sus equipos**, se pretende que salga una ventana como esta, figura 5.22, donde puede elegir uno de sus equipos. Cuando le da a uno de ellos, este será el **perfil del equipo**, figura 5.23. Si quiere **invitar a alguien a su equipo**, este es el boceto de la pantalla que saldrá, figura 5.24. Por último, si el usuario quiere **personalizar su perfil**, esta será la ventana que se ha diseñado para que pueda cambiar su avatar o *banner*, figura 5.25. Si quiere **cambiar el avatar**, este será su boceto, figura 5.26 y **si es el *banner***, será este otro, figura 5.27. En los dos, el objeto que tiene un borde verde, será el que está usando actualmente.

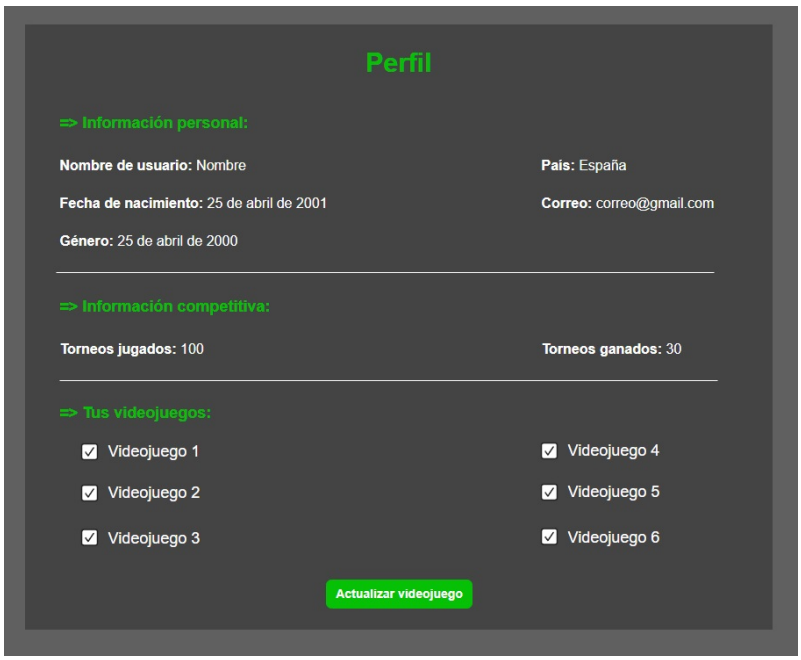

Figura 5.19: Boceto perfil del usuario

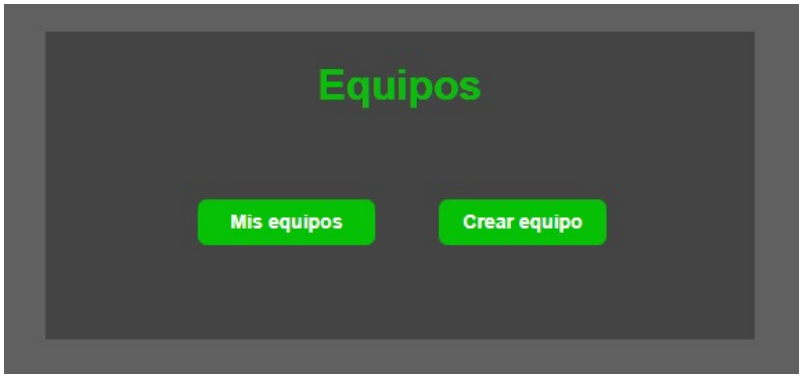

Figura 5.20: Boceto menú de equipos

Figura 5.21: Boceto crear equipo

Figura 5.22: Boceto ver los equipos del usuario

Figura 5.23: Boceto perfil de equipo del usuario

Figura 5.24: Boceto invitar jugador

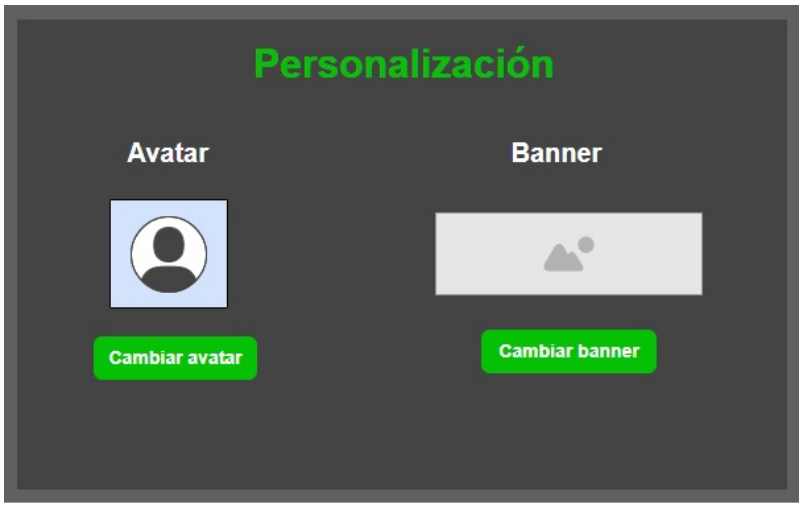

Figura 5.25: Boceto menú personalización

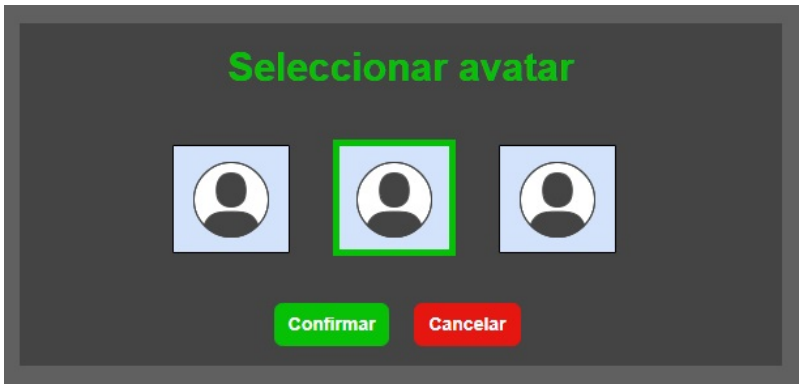

Figura 5.26: Boceto seleccionar avatar

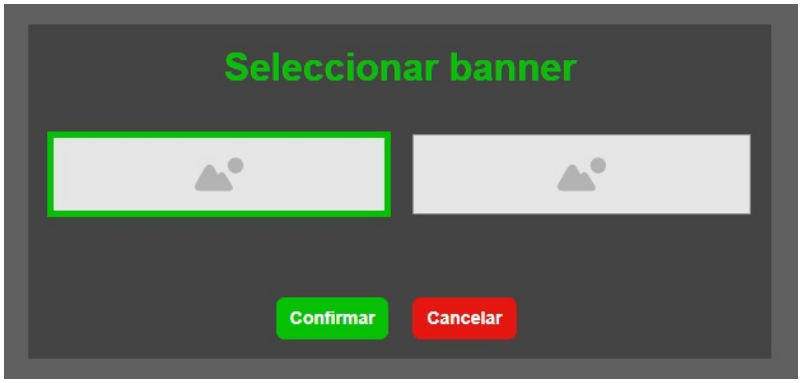

Figura 5.27: Boceto menú personalización

Los últimos bocetos a mostrar serán cuatro. Los dos primeros son para comunicar una **acción realizada con éxito**, figura 5.28 y para **mostrar un error** al realizar una acción, figura 5.29. Estas dos ventanas nos serán útiles para informar al usuario acciones realizadas o no realizadas, por lo que serán utilizadas en más de una ocasión. Las otras dos, son para **ver los perfiles de otros usuarios**, figura 5.30 y **de otros equipos**, figura 5.31.

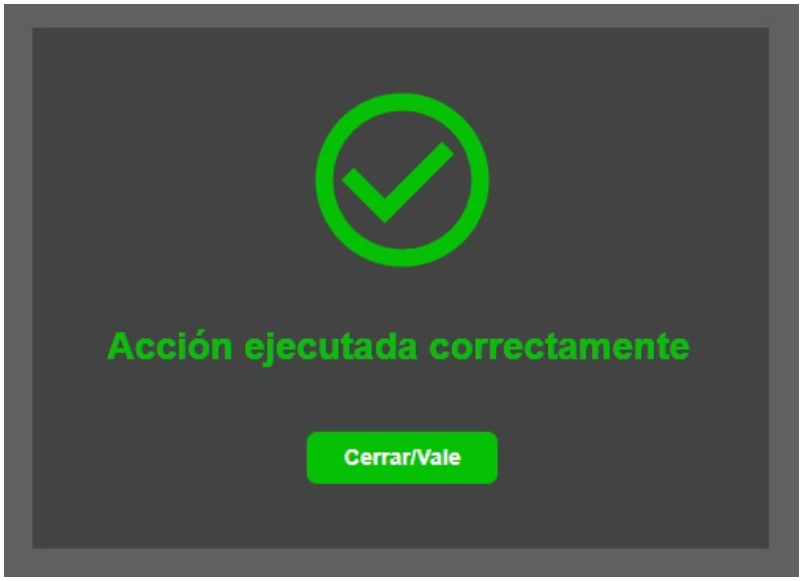

Figura 5.28: Boceto acción realizada exitosamente

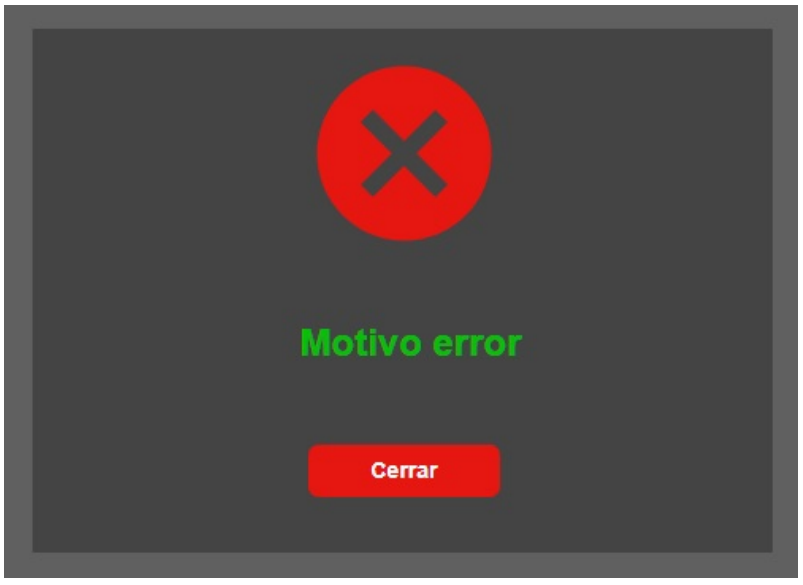

Figura 5.29: Boceto error en la acción

Figura 5.30: Boceto perfil de otro usuario

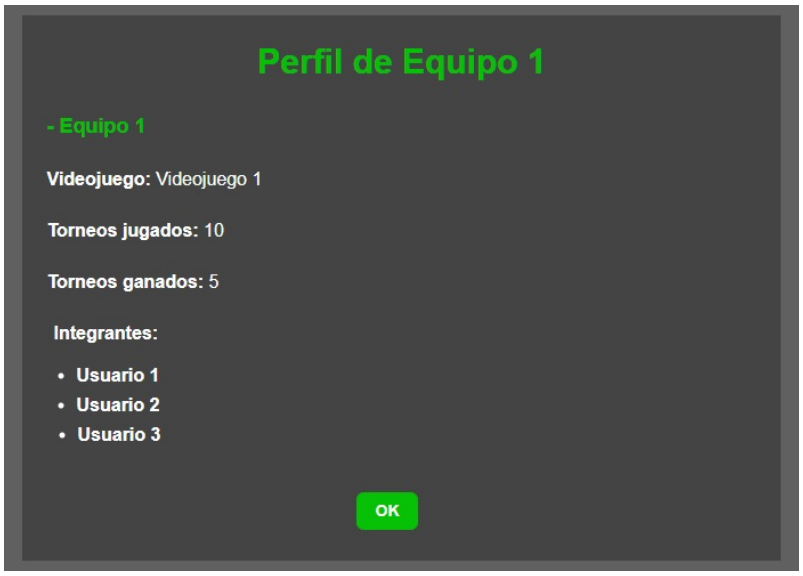

Figura 5.31: Boceto perfil de otro equipo

CAPÍTULO 6
Desarrollo de la solución

En el siguiente capítulo se desarrolla toda la arquitectura del sistema, la forma de comunicación entre partes del proyecto, las herramientas, los lenguajes de programación usados y la información que se envía. Además, se adjuntarán piezas de código para mostrar cómo se han hecho las partes principales del proyecto web.

6.1 Arquitectura

Respecto a la estructura del proyecto, se ha decidido realizar una arquitectura en tres niveles, figura 6.1, donde está el cliente o *frontend*, el servidor o *backend* y la base de datos. Esta estructura es típica entre aplicaciones tradicionales donde el cliente se comunica con el servidor por medio de la aplicación [14]. En esta arquitectura, el cliente se comunica con el servidor y viceversa, y la base de datos se comunica también con el servidor y viceversa.

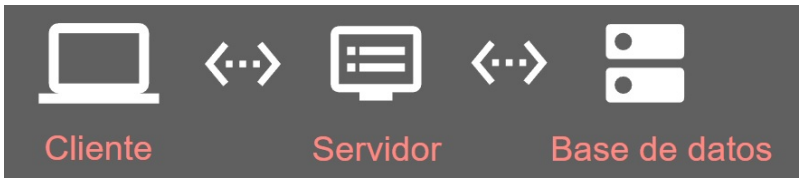

Figura 6.1: Arquitectura del sistema

- **Cliente**: Se trata de la parte visual y que interactúa directamente con el usuario, es decir la interfaz de la aplicación. Mediante acciones hechas en la interfaz, el cliente se comunica con el servidor mediante peticiones a este para recibir las respuestas a estas acciones.
- **Servidor**: La capa intermedia se centra en comunicarse con los dos extremos y satisfacer las peticiones del cliente. En concreto, procesará las peticiones del clientes a la base de datos y enviará la respuesta de este al cliente.
- **Base de datos**: Este último nivel se encarga de administrar los datos de la aplicación.

La comunicación que se utiliza entre los tres niveles en el proyecto son **conexiones HTTP y REST**. **HTTP** es uno de los protocolos de comunicación más usado actualmente que permite comunicarse con el servidor de la web. Esta conexión es posible gracias a: la URL proporcionada para acceder a los datos, las cabeceras que son información adicional que puede ser útil para la petición y los métodos HTTP. Además al no gestionar el estado de las conexiones, es un protocolo bastante seguro.

Hablando de los **métodos HTTP**, hay muchísimos de ellos, aunque en este proyecto solo se usan los más comunes, en concreto tres:

- **GET**: Este primer método es útil para recibir datos del servidor.
- **POST**: Sirve para enviar datos al servidor y recibir, modificar o crear datos a partir de estos.
- **DELETE**: Este último método ayuda a eliminar totalmente algún dato.

Pasando a las conexiones por servicio **REST**, este tipo de conectividad tiene de base el protocolo HTTP. De hecho, su propósito es el mismo y utiliza los mismos elementos y métodos para la conexión. Pese a esto tiene una diferencia y es que se puede personalizar la URL mediante campos que identifiquen los recursos y así aplicar la acción que se quiera hacer a ese objeto en concreto.

Por último, la información que se envía entre las tres capas viaja en forma de **JSON**. **JSON** es un formato de intercambio de datos muy usado hoy en día. Es un formato simple y bastante fácil de leer y entender por el usuario y la máquina, además es independiente a cualquier tipo de lenguaje de programación, por lo que es soportado por la mayoría de estos. Es muy popular debido a que es fácil de implementar en cualquier lado y es fácil de entender, frente a otros formatos de intercambio de datos como XML que son más robustos y complicados de utilizar.

6.2 Contexto tecnológico

Ahora se va a tratar el contexto tecnológico del proyecto. En este apartado se presentarán las herramientas utilizadas en el portal web que aún no se han presentado y que se dividen en: **herramientas globales**, **herramientas del *frontend* o del cliente** y **herramientas del *backend* o del servidor**. Por último, se comentarán también los **lenguajes de programación** usados.

6.2.1. Herramientas globales

En este apartado, se presentarán las herramientas que se han usado tanto en la parte del cliente como en la del servidor:

- **Visual Studio Code**: Un editor de código usado tanto para el *frontend* como el *backend*. Las razones por las que se ha usado este editor frente a otros son: la gran variedad de extensiones que nos ofrece la aplicación para facilitar el

trabajo, un terminal para ejecutar comandos y la experiencia que ya se tenía previamente con esta.

- **Github**: Es un controlador de versiones muy utilizado actualmente, que sirve para ir subiendo los cambios del proyecto y controlar la evolución de este. Además, se pueden crear *branches* en el programa para empezar el desarrollo de algún aspecto del proyecto y si al final es convincente, poder añadir este a la rama principal. Por último, nos permite retroceder si se necesita.

6.2.2. Herramientas *frontend*

Las herramientas que componen la parte del cliente son:

- **React**: Biblioteca de JavaScript de código abierto, que ayuda a los desarrolladores a crear aplicaciones con interfaces de usuario de una sola página. Esta idea se centra en crear un proyecto donde todas las iteraciones que se hacen en este, hagan que parezca que nunca se pasa de una página a otra, ya que una vez que carga la página no se vuelve a recargar. Además, es útil para crear aplicaciones con datos o interfaces que están en constante cambio, ya que con herramientas de esta biblioteca como *useState*, *useEffect* o *useContext*, se pueden guardar estados de determinados objetos, crear condiciones, entre otras cosas, para modificar el DOM virtual que provocará que en la página solo cambien las partes necesarias sin volver a cargar todo el proyecto.
- **SweetAlert2**: O más conocido como **Swal2**, es una herramienta para crear ventanas emergentes personalizadas para la web. Los atractivos de esta herramienta son: su diseño simple pero elegante y con grandes opciones de personalización, el ahorro de código que supone, ya que tiene la capacidad de crear ventanas complejas mediante poco código y la gran variedad de funcionalidades que se pueden agregar a las ventanas y que se puede combinar con los lenguajes de programación del proyecto.
- **MUI**: O más conocido como **Material UI**, es una biblioteca que contiene una gran variedad de componentes propias de las interfaces de usuario. Algunos de estos componentes son: botones, menús, elementos de selección, formularios, entre muchos otros. Pese a que muchos de estos, ya se encuentran por defecto sin la librería, con esta tenemos unos componentes más atractivos visualmente con múltiples opciones de estilos y funcionalidades, que de otra forma sería muy complicado de tener.
- **Font Awesome**: Es una biblioteca que tiene una gran variedad de iconos que se pueden personalizar con CSS. Es muy útil cuando se quieren usar iconos para representar o acompañar a determinadas acciones.

6.2.3. Herramientas *backend*

Ahora, las herramientas relacionadas con el servidor son:

- **Node.js**: Entorno de ejecución en JavaScript que permite ejecutar código de este tipo en entornos que no forman parte del cliente. Es una herramienta orientada a eventos que permite rendir la aplicación en tiempo real. Además, gracias a esta herramienta se pueden importar y añadir fácilmente a nuestro proyecto librerías hechas por otros desarrolladores.

- **Express**: Un marco *backend* de trabajo de Node.js que sirve para desarrollar aplicaciones web y *APIs* de forma sencilla. Destaca porque es capaz de crear rutas para la *API* y ejecutar métodos cuando se llama a estas rutas en la parte del cliente.

- **Body-parser**: *Middleware* que ayuda a manipular los cuerpos de las solicitudes HTTP que entran en aplicaciones web que usan Express. Básicamente, recoge el cuerpo de la solicitud y lo convierte a otros formatos como JSON .

- **Pg**: Módulo que permite utilizar una base de datos en *PostgreSQL* desde aplicaciones que usan Node.js. En general, te permite conectarte a la base de datos y administrarla.

- **Cors**: Otro *middleware* útil para configurar y gestionar las reglas de seguridad entre el intercambio de datos entre aplicaciones web y *APIs* en proyectos que usan Express. En resumen, Cors utiliza encabezados HTTP para especificar un origen de datos y cargarlos en este, también, es útil cuando una aplicación necesita utilizar datos datos o recursos de un dominio distinto al de la aplicación.

- **Jsonwebtoken**: Librería que usa un recurso llamado token web JSON o comunmente llamado **JWT** que se utilizan para transmitir y representar información entre dos partes codificadas como un JSON [15]. Otra funcionalidad que aporta a las aplicaciones web y es utilizado en este proyecto, es el tema de la autenticación. Esta autenticación se crea mediante una clave secreta guardada en el *backend*, que se asocia al id del usuario y crea un token que identifica al usuario y se guarda en el navegador.

- **Bcrypt**: Herramienta de encriptación de contraseñas. A partir de la contraseña proporcionada, utiliza una función de *hashing* con un factor de aleatoriedad llamado *sal* que provoca que las claves encriptadas que se crean para la contraseña sean siempre diferentes, incluso cuando la contraseña sea la misma.

6.2.4. Lenguajes de programación

En este punto tenemos los lenguajes de programación usados en el portal web:

- **HTML**: Lenguaje de marcado de hipertexto que sirve para definir la estructura de la página. El código que usa se llaman etiquetas y con ellas se crean todo el contenido de las páginas como: párrafos, listas, tablas, hiperenlaces... Gracias a este lenguaje, se ha creado toda la estructura y contenido del proyecto.

- **JavaScript**: Lenguaje de programación orientado a objetos que permite crear métodos complejos. En el aspecto de las páginas web, es muy útil para crear funciones e iteraciones con el usuario.
- **CSS**: Lenguaje centrado en el diseño y estilo de los elementos de la página. En el proyecto se ha utilizado con el propósito de conseguir que la página sea más atractiva, además, se combina con HTML a la perfección.
- **SQL**: Lenguaje utilizado para gestionar operaciones en la base de datos. En concreto, se realizarán consultas para obtener datos o modificar la base de datos.

6.3 Ejemplos de código

En esta sección se encontrarán fragmentos de código del proyecto y comentarios sobre los mismos. Ya que el proyecto es muy grande y contiene muchos archivos, es inviable poner todo el código en el documento porque abrumaría y confundiría mucho al lector. En consecuencia, se mostrarán las partes más interesantes del proyecto junto al código donde se muestran todas las herramientas y lenguajes de programación usados.

Lo primero que se muestra es el **archivo correspondiente al *backend***. Aquí se realiza la conexión a la base de datos y todas las configuraciones para tratar las peticiones del cliente al servidor por medio de las direcciones *API* creadas gracias a la herramienta Express y cómo a partir de estas rutas se piden los datos. En concreto, se muestra la importación de casi todas las herramientas del *backend* comentadas anteriormente y como se usan algunas de ellas, el resto de herramientas *backend* se explicarán en el proceso de iniciar sesión. Como se ve en el primer fragmento, se le añade a *app* el express y se hace que utilice como *middleware* el cors y el body-parser. Además, se configura el archivo, para que el servidor escuche en el puerto 3001, mediante la constante *PORT* que es igual a 3001 y el *app.listen* del final del documento que ejecuta la escucha en ese puerto. Por último, tenemos un ejemplo de cómo se crea y ejecuta una ruta en Express para enviar una encuesta a la base de datos y enviar su respuesta al cliente. En este caso, se está pidiendo los avatares de la tienda, por lo que será un *app.get*, si correspondiera a un *POST* o un *DELETE*, sería *app.post* o *app.delete* respectivamente. Después, se define la encuesta SQL que se hará en la base de datos, gracias al *pool*, que explicaremos en el siguiente código a este, se puede conectar a la base de datos y realizar la petición. Por último, mediante *res.json*, se devuelve la respuesta al cliente y si no se consigue, se enviara un error 500 y un mensaje de error.

```
1 const express = require('express'); //Importa Express
2 const bodyParser = require('body-parser'); //Importa body-parser
3 const pool = require('./configBD'); // Importa la pool a la base de
    datos.
4 const cors = require('cors'); //Importa cors
5 const jwt = require('jsonwebtoken'); //Importa jsonwebtoken
6 const bcrypt = require('bcrypt'); //Importa bcrypt
7
8 const app = express();
9 const PORT = process.env.PORT || 3001;
```

```
10
11 const claveSecreta = 'mIcLaV3s3cR3t4yMuyL4rg4Yd1f1c1Ld3g3n3r4r'; //
       Clave para el token del login
12
13 // Habilitar cors como middleware
14 app.use(cors());
15
16 // Middleware donde las solicitudes hechas, se analizan como JSON
17 app.use(bodyParser.json());
18
19 //Resto de funciones app
20
21 //Obtener los avatares de la tienda
22 app.get('/getAvataresTienda', async (req, res) => {
23   try {
24     const result = await pool.query("SELECT * FROM productos WHERE tipo
           = 'Avatar'");
25     res.json(result.rows);
26   } catch (error) {
27     console.error('Error al obtener los avatares', error);
28     res.status(500).json({ message: 'Error al obtener los avatares' });
29   }
30 });
31
32 //Resto de funciones app
33
34 app.listen(PORT, () => {
35     console.log(`Servidor API iniciado en el puerto \${PORT}`);
36 \});
```

El siguiente fragmento de código, como se ha dicho antes, pertenece a la ***pool* de la conexión a la base de datos**. En este, se importa el pg para conectarse a la base de datos PostgreSQL y se empieza a realizar la configuración de la conexión. La configuración consta del nombre de usuario, la dirección anfitrión, el nombre y contraseña de la base de datos y por último, el puerto que corresponde con el predeterminado de PostgreSQL.

```
1 const { Pool } = require('pg');
2
3 // Conectarse a la base de datos
4 const configBD = new Pool({
5   user: 'postgres',
6   host: '192.168.1.10',
7   database: 'e-zone',
8   password: 'cloros',
9   port: 5432, // Puerto predeterminado de PostgreSQL
10 });
11
12 module.exports = configBD;
```

Ahora, se va a presentar la **forma de iniciar sesión** y **cómo se han encriptado sus contraseñas** para que sean seguras. Primero, se ha diseñado un *app.post* con la dirección *API* para el inicio de sesión en el archivo *backend*. En este método, primero sacamos todos los usuarios de la base de datos y comprobamos si el usuario que quiere iniciar sesión, está en la lista, si no lo está, retornará un error. Luego, se comparará la contraseña proporcionada por el usuario, con la que está encriptada dentro de la base de datos con el módulo bcrypt. Si la comparación es

correcta, se generará un *token* JWT que se firmará con la clave secreta definida al principio del archivo con el *id* del usuario y que durará una hora. Al generarse este *token* se devolverá al cliente. Si el proceso falla se devolverá un error.

```
app.post('/login', async (req, res) => {
  try {
    const { nombreUsuario, password } = req.body;

    // Obtener todos los usuarios
    const queryResult = await pool.query("SELECT * FROM usuarios");

    const users = queryResult.rows;

    // Buscar al usuario en users
    let user;
    for (let i = 0; i < users.length; i++) {
      if (users[i].nombreusuario === nombreUsuario) {
        user = users[i];
        break;
      }
    }

    // Si el usuario no existe, retorna un error
    if (!user) {
      return res.status(401).json({ error: 'Credenciales incorrectas'
          });
    }

    // Compara la password proporcionada con la password almacenada con
        bcrypt
    const match = await bcrypt.compare(password, user.password);

    // Si la password no es correcta, se devuelve un error
    if (!match) {
      return res.status(401).json({ error: 'Credenciales incorrectas'
          });
    }

    // Genera un token JWT
    const token = jwt.sign({ userId: user.id }, claveSecreta, {
       expiresIn: '1h' });

    // Se devuelve el token al cliente
    res.json({ token });
  } catch (error) {
    console.error('Error:', error);
    res.status(500).json({ error: 'Hubo un error en el servidor' });
  }
});
```

Para completar el anterior código, se va a **mostrar cómo se han encriptado las contraseñas**. Para ello, se importa el módulo bcrypt y se realiza un método para este propósito. Primero se genera un *salt*, que consiste en un valor aleatorio. Después de la generación de este, se produce un *hash* con la contraseña y el *salt* y se devuelve la solución de esto que corresponderá a la contraseña encriptada. Podemos ver un ejemplo del resultado en la figura 6.2.

```
import bcrypt from 'bcryptjs';
```

```
async function encriptarPassword(password) {
    try {
        // Genera un salt aleatorio
        const salt = await bcrypt.genSalt(10);

        // Encripta la password con el salt
        const hash = await bcrypt.hash(password, salt);

        return hash;
    } catch (error) {
        console.error('Error al encriptar la password:', error);
        throw error;
    }
}

export default encriptarPassword;
```

nombreusuario	contraseña
Link	$2a$10$MmrC/5.tlhw5be06MGWt6eQpTNbqP8YUBhERUnYbKJ4/NqizYgGg6
JovenGamer23	$2a$10$wMRoFsrU37N/MI1N1DROMevpKOeAHaVQdfsw5Dx4o5L1FCyAfwCva
Cheems98	$2a$10$aYQzdbcEGn4Xf1schm0daOTJEtc3yRQ113u15cbUfD2mlGxAbfgRa
DarkCheems	$2a$10$hswqiDwqEpp8eYUEfFKpEu1575mV5cf.eqdJ2qMqldQdGshmCZZCe

Figura 6.2: Ejemplo usuario y contraseña encriptada base de datos

Antes de enseñar código del *frontend*, es interesante resaltar que se ha creado un **contexto** gracias a la herramienta *createContext* de React, que nos permite definir variables o métodos que pueden ser propagados a otros componentes sin necesidad de pasar los datos a cada componente de forma manual. De esta forma si en esos componentes se usa la herramienta *useContext* de React, podrán usar los elementos propagados. En concreto, en este proyecto, se han creado cinco datos que se refieren a variables y métodos que se podrán usar en los componentes que se incluyan en *children*. Los cinco elementos que se crean, versan sobre el inicio de sesión (tanto cuando haces el inicio de sesión normal, como cuando te registras, que inicias sesión automáticamente) y el cierre de sesión, que hacen cambiar el valor de *isLoggedIn* que definirá si el usuario ha iniciado sesión mediante los valores *true* o *false*.

```
import React, { createContext, useState } from 'react';

import Login from '../components/metodos/login';

import cerrarSesion from '../components/metodos/cerrarSesion';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const login = () => {
    Login(() => setIsLoggedIn(true));
  };

  const login_Register = () => {
```

```
    setIsLoggedIn(true);
  }

  const logout = () => {
    cerrarSesion(() => setIsLoggedIn(false));
  };


  return (
    <AuthContext.Provider value={{ isLoggedIn, setIsLoggedIn,
        login_Register, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};
export default AuthContext;
```

Como apunte, el *children* del anterior código que son los componentes que podrán usar el contexto, se refiere a todos los componentes de la aplicación. En el siguiente código, podemos ver el **archivo *App.js*** que es donde se define la estructura del proyecto y sus componentes principales. Como podemos ver, primero se importa el contexto y el *NavBar*, correspondiente al menú superior, que estará disponible en todos los componentes. Luego, se implementan los componentes con la información de cada parte de la página. Como solo puede haber uno activo a la vez, se crean con un componente *Route*, que permitirá asignar una ruta a cada componente y si queremos cambiar de componente, tendremos que viajar a esta. Por último, se define el *Footer*, disponible también en todos los componentes.

```
import React, { useState } from "react";
import Navbar from "./components/Navbar";
import Main from "./components/Main";
import Footer from "./components/Footer";

import { BrowserRouter as Router, Route, Routes } from 'react-router-
    dom';

import Torneos from './components/supPaginas/torneos';
import Rankings from './components/supPaginas/rankings';
import Tienda from './components/supPaginas/tienda';
import Comunidad from './components/supPaginas/comunidad';

import { AuthProvider } from './conexionBD/AuthContext';

function App() {

  const [actualizarNavbar, setActualizarNavbar] = useState(false);

  const actualizarNavbarDespuesDeCompra = () => {
    setActualizarNavbar(prevState => !prevState);
  };

  return (
    <AuthProvider>
      <Router>
        <Navbar actualizarNavbar={actualizarNavbar} />
          <Routes>
            <Route path="/" element={<Main />} />
```

```
            <Route path="/torneos" element={<Torneos />} />
            <Route path="/rankings" element={<Rankings />} />
            <Route path="/tienda" element={<Tienda actualizarNavbar={
                actualizarNavbarDespuesDeCompra} />} />
            <Route path="/comunidad" element={<Comunidad />} />
          </Routes>
        <Footer />
      </Router>
    </AuthProvider>
  );
}

export default App;
```

Ahora vamos a mostrar código relacionado con la parte del cliente. Se va a empezar enseñando la **pantalla principal** o como se llama dentro del código, el componente *Main*. Antes de ver el código, cabe indicar que como el código es largo se van a obviar los *import* para reducir el tamaño. No obstante en herramientas que aún no se han visto importadas, dejaremos el primero a modo de ejemplo para que se vea su importación y uso.

Como se puede observar, el componente *Main*, tiene otros dos componentes, ***Header*** y ***Services***. El primer componente, que corresponde al segundo código, podemos observar el uso del *useContext* visto por primera vez en un archivo, que recoge las constantes *isLoggedIn* y *login_Register* del contexto, además, podemos ver un componente sacado de Material UI, en este caso, un botón y la importación de un archivo CSS para añadir estilos. A continuación, podemos ver que hay una imagen que tendrá encima el botón para registrarse si el usuario no ha iniciado sesión. Este efecto, dependerá si *isLoggedIn* o el elemento *token*, que se asigna al iniciar sesión, del almacenamiento local son verdaderos o falsos. En el tercer código, correspondiente a *Services*, se importa la herramienta Font Awesome junto a un icono llamado *faGamepad*, para ponerlo al lado de un párrafo. Después, se importa el carrusel de videojuegos realizado, y se crean unas tarjetas con información de la página.

```
function Main() {
  return (
    <>
      <main>
        <Header />
        <Services />
      </main>
    </>
  );
}
export default Main;
```

```
import React, { useContext } from "react";

import './estilos/PantallaPrincipal.css';

import Button from '@mui/material/Button';

import AuthContext from '../conexionBD/AuthContext';

//Resto de import
```

```
10
11 function Header() {
12
13   const { isLoggedIn } = useContext(AuthContext);
14
15   const { login_Register } = useContext(AuthContext);
16
17   const token = localStorage.getItem('token') != null;
18
19   const handleRegistrarseClick = () => {
20     Registrarse(() => login_Register());
21   };
22
23   return (
24     <header>
25       <div class="div_Imagen_Principal">
26         <img src={logo} alt="Logo E-ZONE" class="foto_Principal"></img>
27         <h2 className="texto_Principal">Unete a la mejor comunidad de
               videojuegos y compite contra los mejores</h2>
28         {!isLoggedIn && !token && (
29           <Button variant="contained" style={{ backgroundColor: 'green'
                 , marginTop: '750px', marginLeft: '43%', fontSize: '18px'
                  }} onClick={handleRegistrarseClick} >
30             Registrarse ahora
31           </Button>
32         )}
33       </div>
34     </header>
35   );
36 }
37 export default Header;
```

En ***Services***, como hay mucha código y no se ve bien en el documento, se han quitado muchos *div* creados y dos elemenos *Card*, para que se vea el código de forma ordenada y limpia. En consecuencia, se ha dejado la oración con el icono de Font Awesome y una *Card* de Material UI con más elementos de esta herramienta dentro, junto a código CSS aplicado en el propio archivo que se puede identificar dentro de las etiquetas HTML con las letras *sx*, a modo de ejemplo para ver cómo se haría.

```
1 //Resto de import
2
3 import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
4 import { faGamepad } from '@fortawesome/free-solid-svg-icons';
5
6 //Resto de import
7
8 function Services() {
9   return (
10     <div className="container services">
11         <h2 className="main-title text-center">VIDEOJUEGOS COMPETITIVOS
               </h2>
12         <p className="texto-Servicios">Los juegos de nuestra comunidad
               <FontAwesomeIcon icon={faGamepad} /></p>
13         <Carrusel />
14         <h2 className="main-title text-center">JUEGA PARA...</h2>
15         <div className="card-cover">
16             //Muchos div creados
```

```
                <Card sx={{ maxWidth: 345 }}>
                    <CardMedia sx={{ height: 140 }}
                        image={competir}/>
                    <CardContent>
                        <Typography gutterBottom variant="h5" component
                            ="div">
                            <b>Competir</b>
                        </Typography>
                        <Typography variant="body2" color="text.
                            secondary">
                            Compite en varios videojuegos de forma
                                profesional en torneos en solitario o
                                en equipo y ocupa las listas de los
                                mejores jugadores.
                        </Typography>
                    </CardContent>
                </Card>
                //Resto de elementos Card
            //Cierre de todos los div
        </div>
    </div>
  );
}
export default Services;
```

Después, se va a enseñar el archivo ***Navbar***, que corresponde al menú superior, donde se puede ver cómo se viaja de un componente a otro y cómo se gestiona según el inicio de sesión. Para ello, el pedazo de código elegido es lo que retorna el archivo. En él, podemos ver unas etiquetas *Link* de React que redirigirán al componente del enlace situado en *to*, al hacerle *click*. Después, dependiendo si el usuario ha iniciado sesión, se mostrará el botón para iniciar sesión y sino, las monedas del usuario junto al nombre del usuario que contendrá el menú de usuario llamado *PerfilNavBar*, que contendrá cuatro funcionalidades: perfil, donde puede ver su perfil y gestionar sus videojuegos, equipos, donde podrá ver, gestionar y crear equipos, personalización, donde podrá personalizar su perfil y cerrar sesión. La última parte del menú superior, también se ha descartado ya que no es importante.

```
//Resto de codigo
return (
    <nav className="navbar">
      <div className="container">
        <div className="row">
          <ul className="bar">
            <li style={{ position: 'absolute', left: 30 }}>
              <Link to="/"><img src={logo} alt="Logo E-ZONE" class="
                  foto_Logo"></img></Link>
            </li>
            <li>
              <Link to="/torneos">Torneos</Link>
            </li>
            <li>
              <Link to="/rankings">Rankings</Link>
            </li>
            <li>
              <Link to="/comunidad">Comunidad</Link>
            </li>
```

```
            <li>
              <Link to="/tienda">Tienda</Link>
            </li>
            {isLoggedIn || token ? (
              <>
              <li style={{ justifyContent: 'flex-end', position: '
                  absolute', right: 50, bottom: 10, display: 'flex',
                  alignItems: 'center' }}>
                <span style={{ color: 'white', marginRight: '5px',
                    marginBottom: '10px' }}>{monedas}</span>
                <FontAwesomeIcon icon={faCoins} style={{ color: 'yellow
                    ', marginTop: '-8px' }} />
                <PerfilNavBar />
              </li>
              </>
            ) : (
              <li style={{ justifyContent: 'flex-end', position: '
                  absolute', right: 0 }}>
                <Button variant="contained" onClick={handleLoginClick}
                    style={{ backgroundColor: 'green', marginRight: '
                    100px' }}>
                  Iniciar sesion
                </Button>
              </li>
            )}
          </ul>
          //Resto de codigo
        </div>
      </div>
    </nav>
  );
```

Por último, se va a enseñar la creación de una **ventana emergente** con SweetAlert2. Para eso, se ha elegido el **archivo *infoEquipos.js***, donde se realiza la función de ver la información de otro equipo. Entonces, cuando se accede a este archivo, lo primero que se realiza, que no hemos analizado aún en este apartado, es una petición al *backend*. En este caso, se realiza una petición *POST* a la dirección *http://localhost:3001/getInfoEquipo* con el nombre del equipo en el cuerpo de la solicitud y se guarda la respuesta en *equipoDatos*, esa respuesta se usa después, para agregar en la ventana toda la información del equipo. A continuación, llega el código relacionado con Swal2. Se empieza con un *Swal.fire* que hace que se abra la ventana emergente. El contenido y estilo que vendrá en la ventana son los campos que se encuentran a continuación, como el título, el html, correspondiente al contenido, el color del fondo o el ancho. El campo *didOpen* se ejecutará después de que la ventana esté totalmente visible por lo que será lo último que se añada. Para finalizar, si hay algún error, en vez de esta ventana, se abrirá otra de error. Se han quitado algunas partes para hacer más visible el código.

```
//Los import

const infoEquipos = async (equipo) => {
  try {

    const response = await fetch('http://localhost:3001/getInfoEquipo',
        {
      method: 'POST',
```

```
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ equipo }),
    });

    //Codigo por si hay un error

    const data = await response.json();
    const equipoDatos = data[0];

    const integrantesLista = equipoDatos.integrantes_nombres.map(
        integrante => {
      if (integrante === localStorage.getItem('username')) {
        return `<li style="color: #01bf71;">${integrante}</li>`;
      } else {
        return `<li style="color: white; cursor: pointer;" data-usuario
            ="${integrante}">${integrante}</li>`;
      }
    }).join('');

    Swal.fire({
      title: `<strong>Perfil de ${equipo}</strong>`,
      html: `
        <div style="margin-bottom: 20px; text-align: left">
            <h3 style="color: #01bf71;">- ${equipoDatos.equipo_nombre
                }</h3>
            <p style="color: white;"><b>Videojuego:</b> ${equipoDatos.
                videojuego_titulo}</p>
            <p style="color: white;"><b>Torneos Jugados:</b> ${
                equipoDatos.torneos_jugados}</p>
            <p style="color: white;"><b>Torneos Ganados:</b> ${
                equipoDatos.torneos_ganados}</p>
            <p style="color: white;"><b>Integrantes:</b></p>
            <ul>${integrantesLista}</ul>
        </div>
      `,
      background: '#333333',
      customClass: {
        title: 'titulo_Registro',
      },
      confirmButtonColor: '#01bf71',
      width: '700px',
      didOpen: () => {
        document.querySelectorAll('li[data-usuario]').forEach(item => {
          item.addEventListener('click', () => {
            const usuario = item.getAttribute('data-usuario');
            infoUsuario(usuario);
          });
        });
      }
    });
  } catch (error) {
    console.error('Error:', error);
    Swal.fire({
      title: 'Error',
      icon: 'error',
      //Codigo del Swal2 de error
    });
```

```
61     }
62 };
63
64 export default infoEquipos;
```

CAPÍTULO 7
Producto desarrollado

A continuación se van a mostrar capturas de la interfaz final del proyecto. Además, se mostrará cómo a partir de estas capturas, se llevarán a cabo los casos de uso definidos en el apartado 4.3. Como aclaración, hay muchas ventanas emergentes de pregunta, confirmación, error y mensajes de aviso en ellas, que salen en muchos casos de uso y prácticamente son idénticas entre ellas a excepción del texto, por lo que cuando se enseñe una de estas ventanas por primera vez no se volverá a enseñar más adelante y solo se hará alusión a la primera con sus diferencias.

Caso de uso: **Registrarse**

Un usuario sin cuenta se puede registrar en la pantalla principal presionando el botón '**REGISTRARSE AHORA**', figura 7.2. Al presionarlo, aparece una ventana emergente para el registro donde se pueden rellenar los datos de la cuenta: nombre de usuario, contraseña, país, fecha de nacimiento, correo electrónico y género, figura 7.1. Si todo es correcto al presionar '**Crear cuenta**', saldrá una ventana emergente donde se confirma la cuenta creada y se inicia sesión automáticamente, como se puede comprobar a la derecha del menú superior que pone el nombre de la nueva cuenta y sus monedas, figura 7.3. Si hay algún error de los definidos en el caso de uso, saldrá un mensaje de aviso, pero con el error específico de cada uno, figura 7.4.

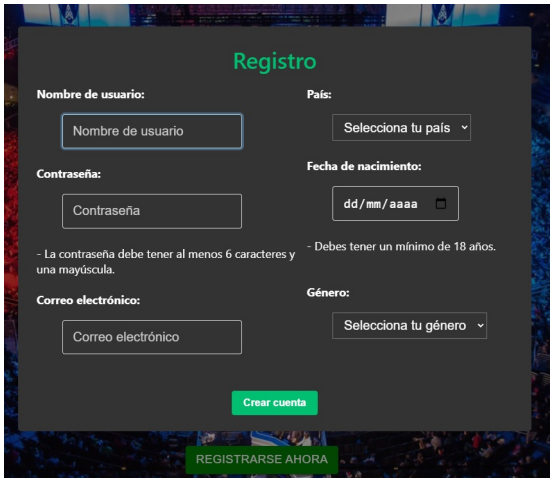

Figura 7.1: Interfaz ventana emergente registro

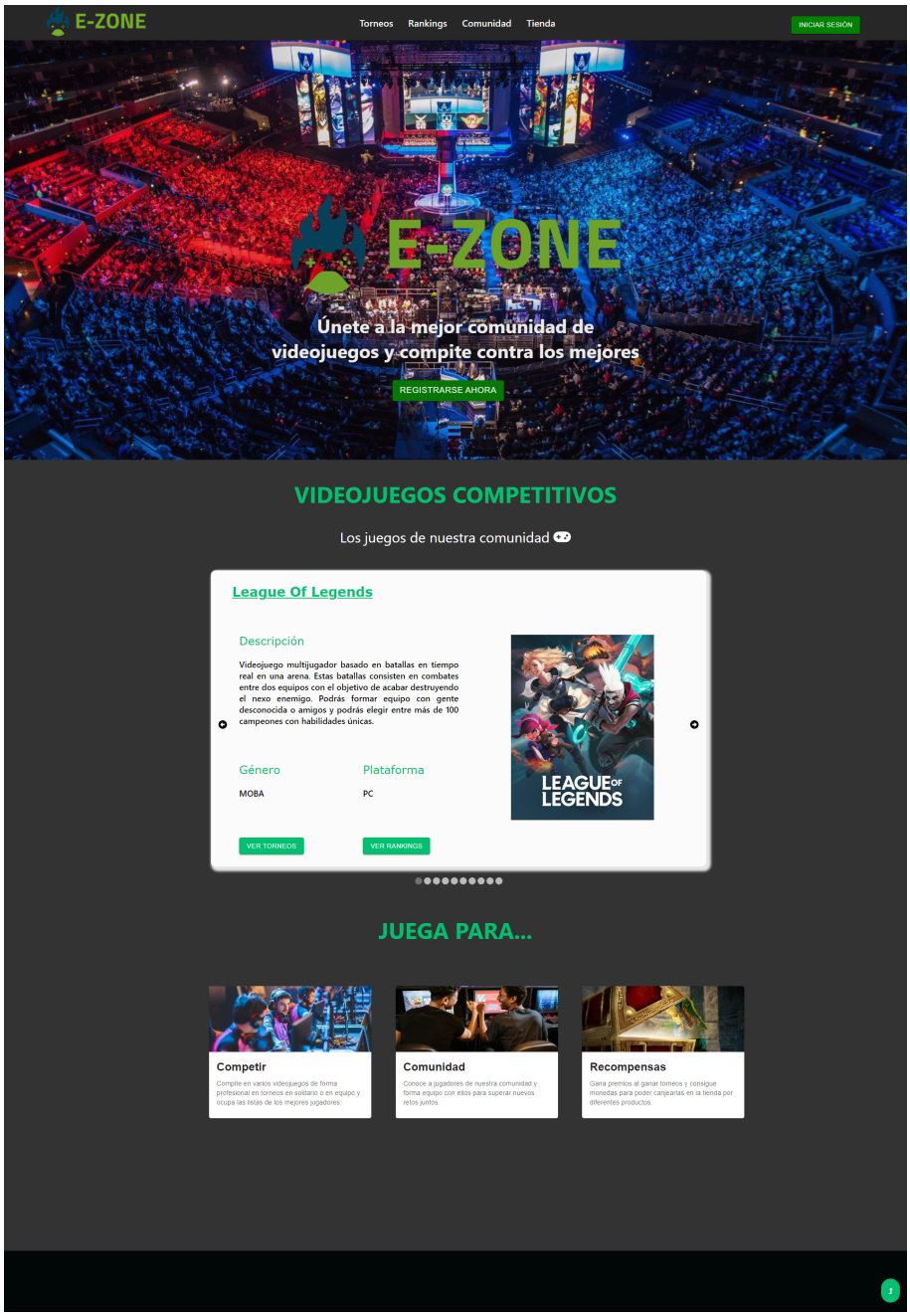

Figura 7.2: Interfaz pantalla principal sin sesión

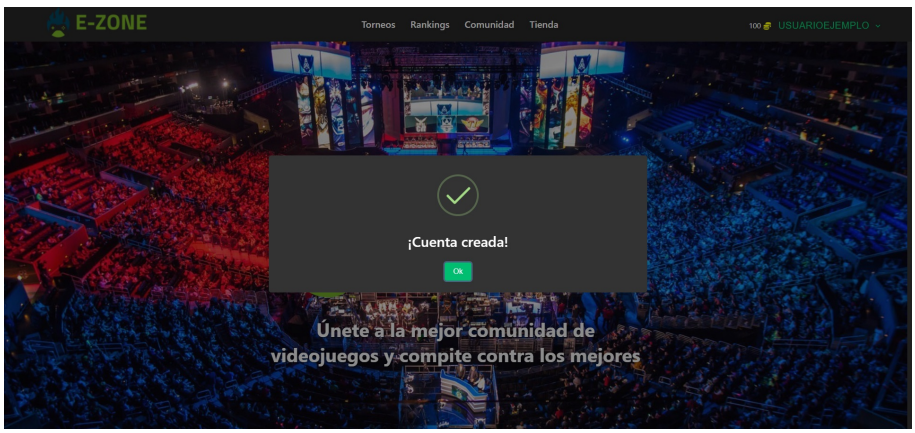

Figura 7.3: Interfaz ventana emergente confirmación

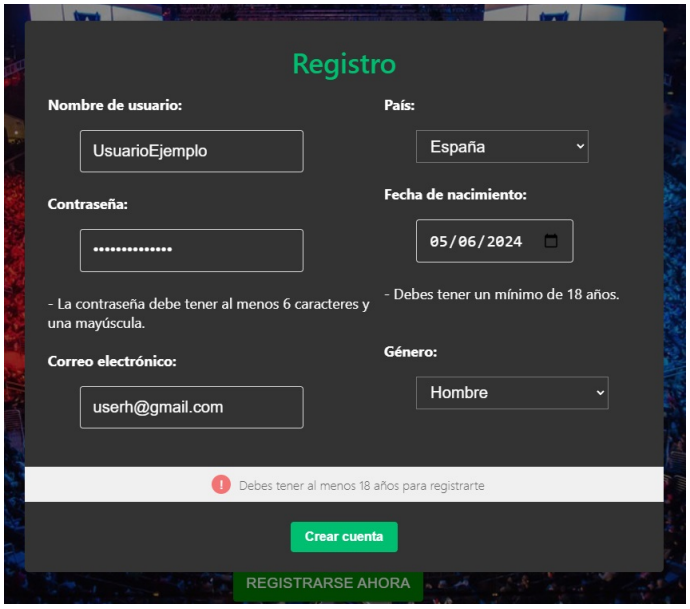

Figura 7.4: Interfaz ventana emergente con mensaje de aviso

Caso de uso: **Iniciar sesión**

Para iniciar sesión el usuario se sitúa en la pantalla principal, aunque se puede realizar en cualquier parte de la página sin que haya iniciado sesión previamente, ya que se encuentra en el menú superior a la derecha, figura 7.2. Al presionar el botón '**INICIAR SESIÓN**', sale la siguiente ventana para rellenar el nombre de usuario y la contraseña, figura 7.5. Al rellenar el formulario, se presiona el botón '**Entrar**' y si es correcto, se iniciará sesión y se pondrá su nombre y sus monedas en el menú superior, además, se activarán las funciones de los usuarios con sesión iniciada. Si hay un error con las credenciales, saldrá un aviso en la propia ventana de iniciar sesión, parecido al de la figura 7.4, pero avisando que hay un error de credenciales.

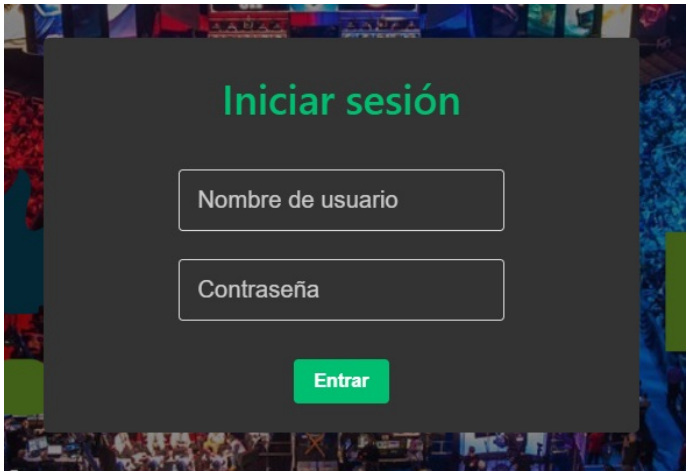

Figura 7.5: Interfaz ventana emergente iniciar sesión

A partir de ahora, partiremos de la base de que el usuario ha iniciado sesión previamente.

Caso de uso: **Cerrar sesión**

Esta acción se podrá ejecutar desde cualquier lado de la página ya que se encuentra en el menú superior al abrir el menú de usuario al presionar su nombre, figura 7.7. Este caso se ejecutará, a partir de la pantalla principal con la sesión iniciada, figura 7.6. Para llevar a cabo la acción, se hará *click* en la parte del menú de usuario, que pone '**Cerrar sesión**' y la sesión se cerrará, dando como resultado, la página principal sin sesión de nuevo, figura 7.2.

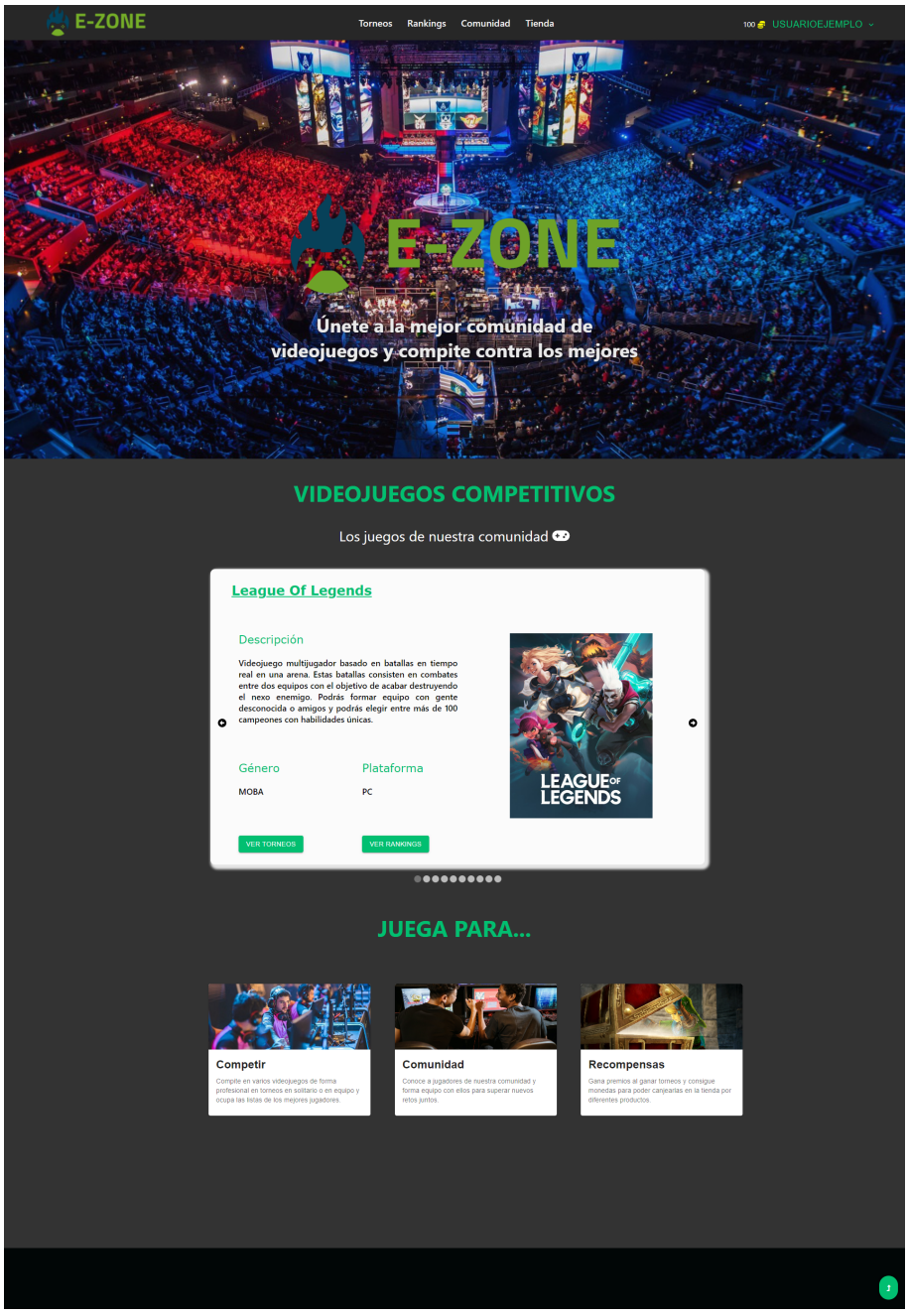

Figura 7.6: Interfaz pantalla principal con sesión

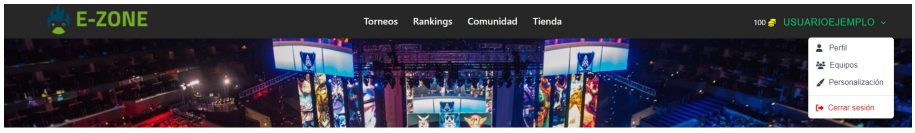

Figura 7.7: Interfaz menú de usuario

Caso de uso: **Ver los videojuegos y su información**

Si un usuario quiere ver los videojuegos que da soporte la página y su información, lo único que tendrá que hacer es viajar hacia la parte inferior de la pantalla principal, hasta la **sección de videojuegos competitivos** donde hay un carrusel, figura 7.6. En él se podrá ver: título, descripción, plataformas, género y portada, junto a los botones para ir a sus torneos y *rankings*. Actualmente hay 10 videojuegos y el carrusel solo enseña uno a la vez, si el usuario quiere ver otro, simplemente presionará el botón de la izquierda, derecha o los puntos del carrusel y el juego cambiará y con él la información del carrusel. En la siguiente figura 7.8, se puede ver más de cerca el carrusel junto a la información de un juego.

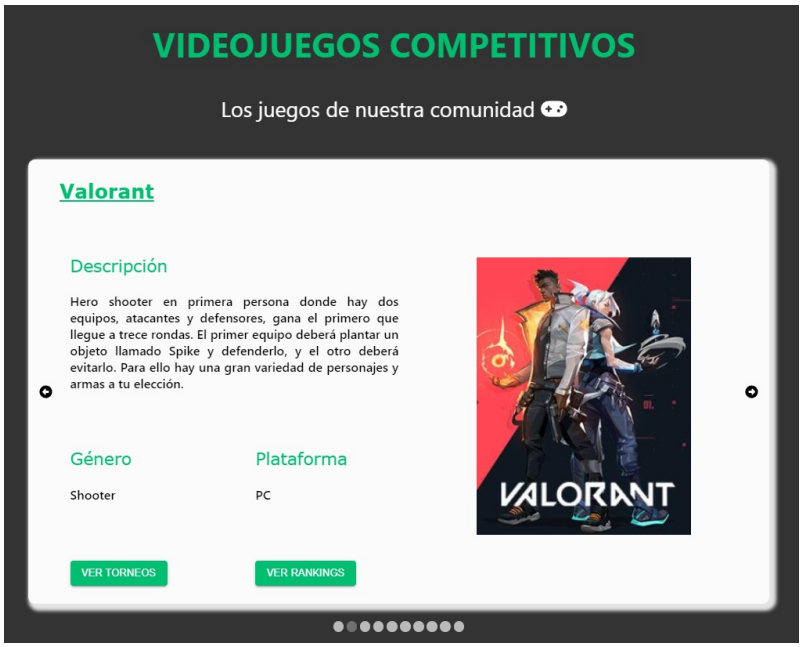

Figura 7.8: Interfaz carrusel de videojuegos

A partir de ahora, nos situaremos en las páginas secundarias. Estas páginas contienen más de un caso de uso, por lo que se van a aprovechar los pasos que se hacen para cada una para no ir repitiendo lo mismo en cada funcionalidad.

Caso de uso: **Ver los torneos disponibles según el videojuego y su información**

Si el usuario quiere ver los torneos disponibles y ver su información, si empieza en la pantalla principal, figura 7.6, presionará en el menú superior el apartado de '**Torneos**' y aparecerá la sección de torneos, figura 7.9. En ella, podemos ver los torneos disponibles del videojuego que aparece en el selector, si se quiere cambiar

el videojuego, se pulsa el selector y el usuario puede cambiar el juego. Para ver la información del torneo y el botón de inscripción, se presionará el desplegable del nombre del torneo. Se podrá ver: tipo, modalidad, lugar, jugadores inscritos, día y fecha de comienzo y recompensa.

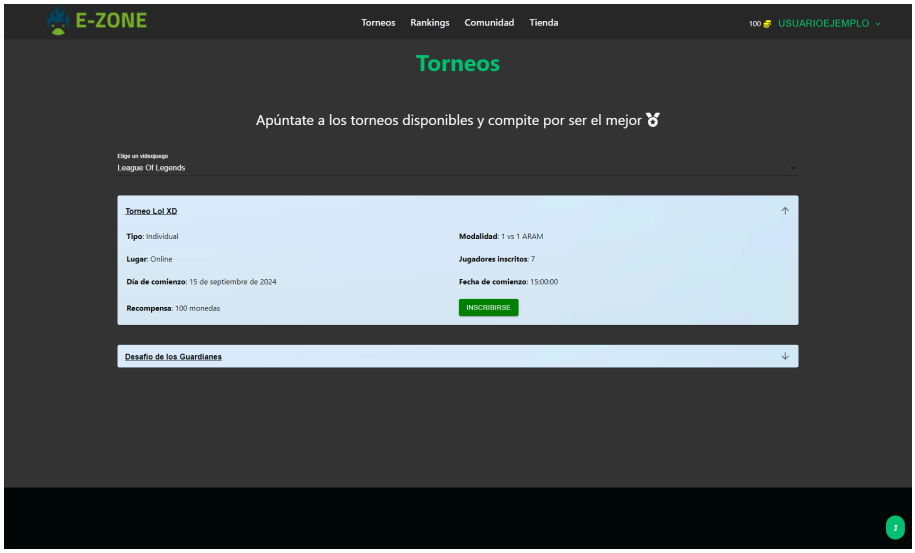

Figura 7.9: Interfaz torneos

Caso de uso: **Inscribirse a torneos**

Si el usuario decide inscribirse en un torneo de forma individual, tendrá que buscar un torneo que tenga de tipo: '**Individual**' y presionar el botón **INSCRIBIRSE**. Al hacerlo, aparecerá una ventana emergente de pregunta, figura 7.10, si quiere inscribirse presionará el botón verde, si no el rojo, haciendo que el usuario salga de la ventana emergente. En el caso de inscribirse, al hacerlo aparecerá una ventana de confirmación de este tipo, figura 7.3, pero poniendo '**¡Inscripción hecha!**'.

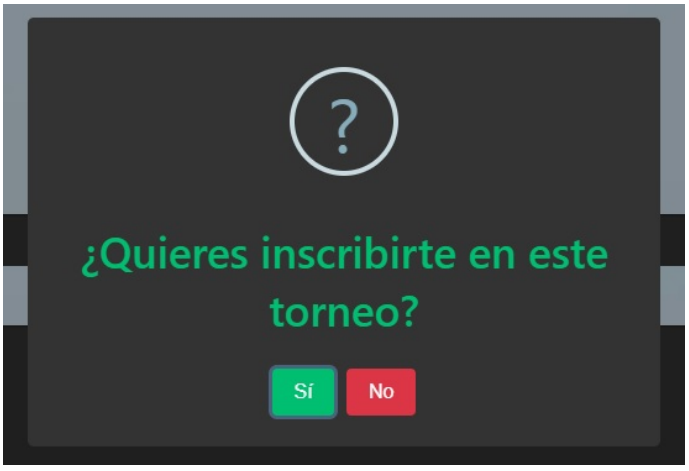

Figura 7.10: Interfaz ventana emergente pregunta

Por otro lado, si quiere inscribirse como equipo, buscará un torneo de tipo: '**Equipo** y presionará el botón **INSCRIBIRSE**. Al instante, aparecerá una ventana emergente con un selector para elegir con cual de sus equipos quiere inscribirse, figura 7.11, si presiona el botón verde lo inscribirá y si presiona el rojo, saldrá de

la ventana. Al darle al verde, aparecerá una ventana de confirmación, figura 7.3, pero con '**¡Inscripción hecha!**'.

Figura 7.11: Interfaz ventana emergente inscripción equipo

Al inscribirse a un torneo individual, el botón cambiará a gris y pondrá '**Inscrito**'. Si intenta el usuario volver a inscribir a un equipo que ya está inscrito, aparecerá una ventana emergente de error, figura 7.12.

Figura 7.12: Interfaz ventana emergente error

Caso de uso: **Ver la clasificación de los mejores jugadores y equipos según el videojuego**

Para este caso, si el usuario está en la pantalla principal, figura 7.6, presiona en el menú superior el apartado de '**Rankings**'. Al hacerlo, se abre este apartado, figura 7.13. En él se puede ver la clasificación de los mejores jugadores según el videojuego del selector, se puede cambiar al presionar el selector. Si se quiere ver el de los mejores equipos, se presionará el botón '**EQUIPOS**' y pasarán a salir los mejores equipos según el videojuego.

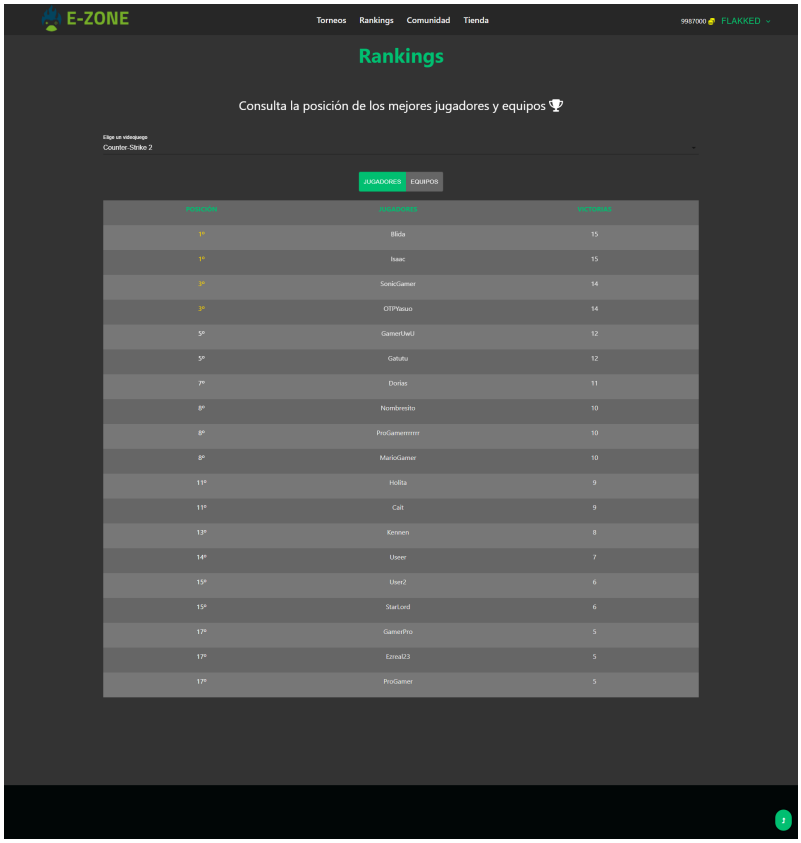

Figura 7.13: Interfaz *rankings*

Caso de uso: **Ver *posts* de la comunidad**

Para que el usuario vea las publicaciones del apartado de comunidad, si empieza en la pantalla principal, 7.6, presionará el apartado '**Comunidad**' del menú superior. En consecuencia se abrirá el apartado, figura 7.14, donde podrá ver los *posts* colgados por los usuarios, además tendrá la opción de añadir y filtrar *posts* mediante unos botones que tiene debajo de las publicaciones y podrá borrar sus propios *posts* mediante un botón rojo con forma de papelera. Sobre la información que hay en las publicaciones, encontraremos: autor, contenido, fecha de publicación y categorías.

Caso de uso: **Filtrar *posts***

Si el usuario quiere filtrar las publicaciones, presionará el botón '**Aplicar filtro**'. Al presionarlo, saldrá esta ventana, figura 7.15, donde se puede filtrar por categorías. El usuario selecciona las categorías que quiere y presiona '**Aplicar**'. Al hacerlo, por pantalla aparecerán los *posts* que tienen esas categorías. Si el usuario durante el proceso de elegir categorías, quiere cancelar el proceso, presionará '**Cancelar**'.

Caso de uso: **Añadir *posts***

Para añadir un *post*, el usuario presionará el botón '**Añadir Post**'. Inmediatamente, aparecerá esta ventana, figura 7.16. En ella podremos rellenar el contenido del *post* y elegir las categorías de este junto a un botón '**Cancelar**', para parar el proceso y otro '**Añadir**' para publicarlo. Si presionamos el botón de añadir y no

se ha escrito contenido o elegido alguna categoría para el *post*, saldrá un mensaje de aviso, como el de la figura 7.4, para avisar del error. Si todo es correcto, aparecerá una ventana de confirmación como el de la figura 7.3 y el *post* saldrá en la pantalla.

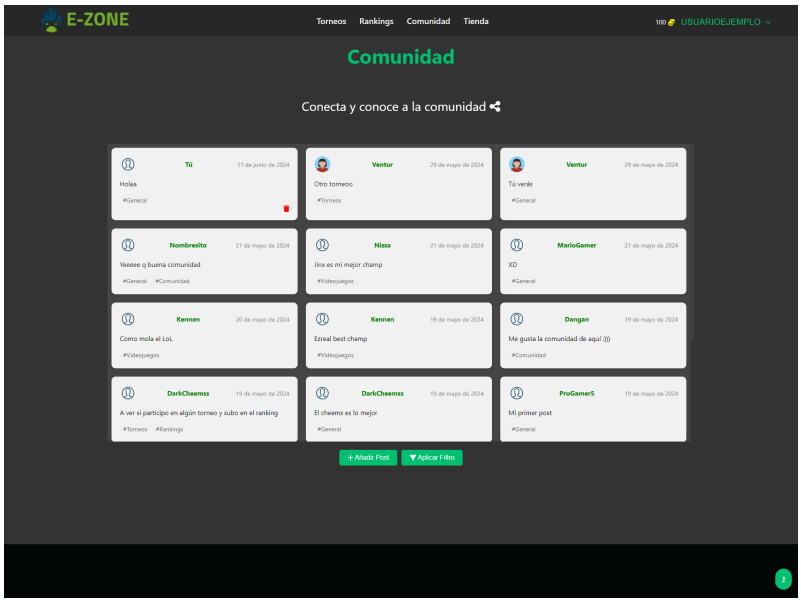

Figura 7.14: Interfaz comunidad

Figura 7.15: Interfaz aplicar filtro

Caso de uso: **Borrar sus posts**

Si el usuario quiere borrar uno de sus *posts*, presionará el botón rojo con forma de papelera y le saldrá una ventana de pregunta parecida a la ventana de la figura 7.10. En ella le pregunta si de verdad quiere borrarlo, si al final el usuario no quiere, presionará el botón rojo y saldrá del proceso y si quiere, el verde. Si decide borrarlo presionará este último y aparecerá una ventana de confirmación como el de la figura 7.3, anunciando que el *post* se ha borrado y ya no saldrá en este apartado.

Figura 7.16: Interfaz añadir *post*

Caso de uso: **Ver productos de la tienda**

Respecto a este caso de usuario, si el usuario se sitúa en la pantalla principal, figura 7.6, accederá a la tienda, presionando '**Tienda**' en el menú superior. Al acceder, figura 7.17, verá todos los productos de la tienda, actualmente habrán seis avatares y tres *banners* con sus respectivos precios. Si alguno de los productos ya ha sido comprado, tendrá una marca en el botón en vez del precio.

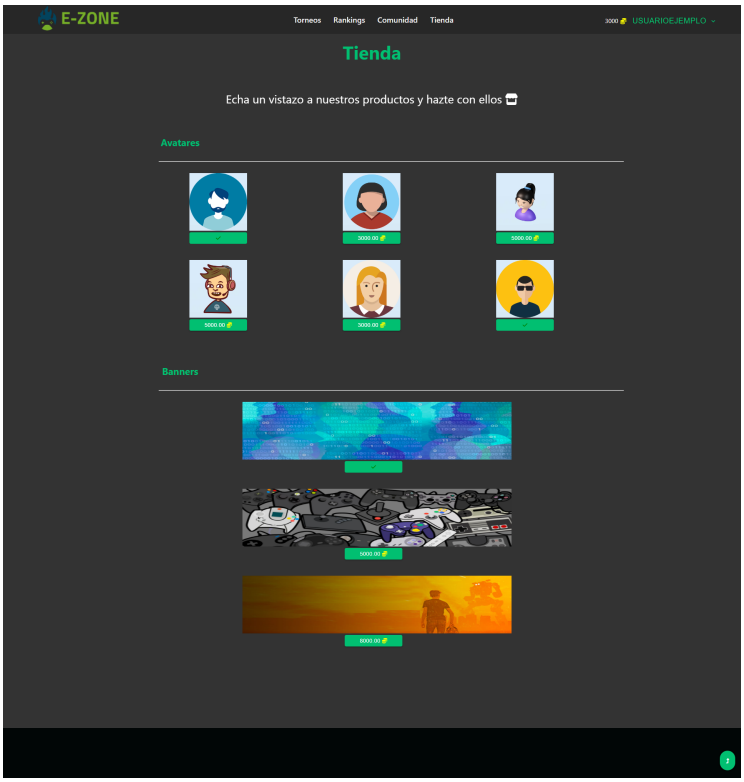

Figura 7.17: Interfaz tienda

Caso de uso: **Comprar productos en la tienda**

Si un usuario quiere comprar un producto, primero tendrá que ver si tiene suficientes monedas, lo podrá ver en el menú superior al lado de su nombre en

cualquier parte de la página. Para comprar, el usuario debe hacer *click* en el botón donde pone el precio del producto que quiere. Al hacerlo, mostrará una ventana de pregunta, parecida a la figura 7.10, preguntando si quiere comprar el producto. Si no quiere, presionará el botón rojo y si quiere, presionará el verde y aparecerá una ventana de confirmación, como la de la figura 7.3, confirmando la compra y se sustituirá el precio por la marca mencionada antes. Si el usuario no tiene dinero y le da al botón verde, aparecerá una ventana de error, como la de la figura 7.12, avisando de que el usuario no tiene dinero.

Caso de uso: **Ver información de su perfil**

En este caso, para que el usuario vea la información de su perfil, presionará su nombre en el menú superior desde cualquier sección de la página. Al hacerlo, se abrirá el menú de usuario, figura 7.7. Para acceder a su perfil, presionará '**Perfil**' y aparecerá esta ventana, figura 7.18. En ella, podrá ver su información personal (nombre, país, fecha de nacimiento, país y género), competitiva (torneos jugados y ganados) y una sección para elegir sus videojuegos.

Figura 7.18: Interfaz perfil del usuario

Caso de uso: **Cambiar videojuegos del usuario**

Para actualizar sus videojuegos, solo tendrá que presionar los *checkbox* de los videojuegos que tiene y presionar el botón '**Actualizar Videojuegos**'. Al hacerlo, se confirmará la acción con una ventana emergente parecida a la figura 7.3 y volverá a salir la misma ventana del perfil pero con los juegos actualizados.

Caso de uso: **Aceptar/Rechazar invitación**

Si el usuario quiere aceptar o rechazar una invitación, accederá al menú de usuario, desde el menú superior en cualquier parte de la página, figura 7.7. Para el propósito entrará a '**Equipos**' y aparecerá esta ventana, figura 7.19, si tiene

invitaciones saldrá como se ve y si no, no saldrá la parte de invitaciones de la pestaña. Para aceptar la invitación del equipo, en la sección '**Invitaciones**', se presionará el botón verde y si no, el rojo, en cualquiera de los dos casos saldrá una ventana de confirmación de la acción como en la figura 7.3, anunciando que se ha aceptado o rechazado.

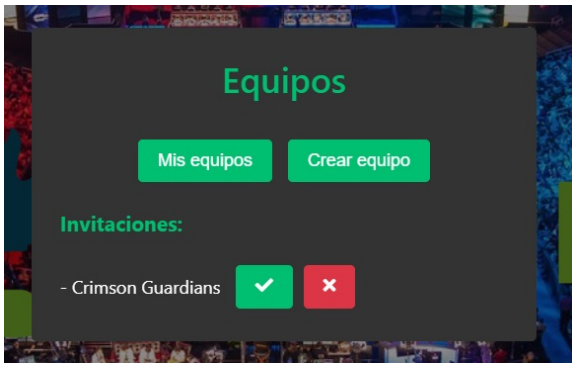

Figura 7.19: Interfaz principal equipos

Caso de uso: **Ver sus equipos y su información**

Si el usuario quiere ver sus equipos e información de estos, en la misma ventana de antes, figura 7.19, presionará el botón '**Mis equipos**', si no tiene equipos, saldrá una ventana error, como la de la figura 7.12, anunciando la situación, y si tiene, saldrá esta ventana, figura 7.20. Si el usuario quiere cancelar el proceso presionará '**Cancelar**' y saldrá de la ventana. Por otro lado, si quiere seguir, seleccionará un equipo en el selector y pulsará '**Buscar**'. Al hacerlo, saldrá esta ventana, figura 7.21, donde sale: nombre, participaciones, victorias, videojuego e integrantes del equipo, además de dos botones para gestionar el equipo. Si el usuario pulsa '**OK**', saldrá de la ventana.

Caso de uso: **Abandonar equipos**

Para abandonar un equipo, nos situamos en la última ventana que se ha comentado, figura 7.21, y presiona el botón rojo '**Abandonar equipo**'. Al presionarlo, saldrá una ventana de confirmación con la acción escrita, parecida a la figura 7.3.

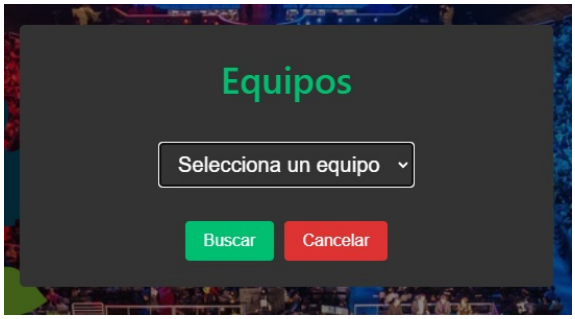

Figura 7.20: Interfaz seleccionar uno de sus equipos

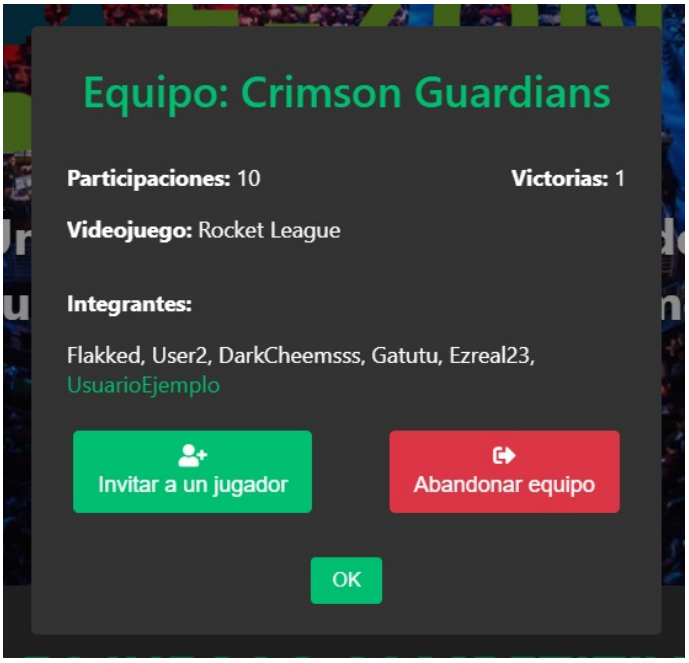

Figura 7.21: Interfaz información equipo

Caso de uso: **Invitar a jugadores**

Para invitar a un jugador al equipo, estamos en la última ventana emergente comentada, figura 7.21, y se presiona '**Invitar a un jugador**', al hacerlo, saldrá esta ventana para seleccionar un jugador, figura 7.22. El usuario selecciona uno y pulsa '**Invitar**', al instante sale una ventana de confirmación como el de la figura 7.3, anunciando la invitación, si el usuario no quiere invitarle al final, presionará el botón '**Cancelar**'.

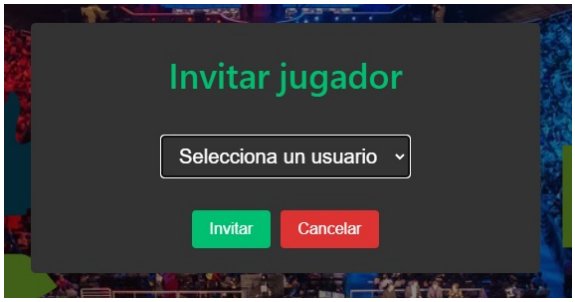

Figura 7.22: Interfaz invitar a un jugador

Caso de uso: **Crear equipos**

Para crear un equipo, el usuario se vuelve a situar en la figura 7.19 y pulsa '**Crear equipo**'. Al hacerlo, saldrá esta ventana, figura 7.23, donde habrá que elegir el nombre del equipo y el videojuego del equipo. Si el usuario no rellena ningún nombre o el nombre que ha puesto ya está en uso, saldrá un aviso con estos problemas dentro de la propia ventana, como el de la figura 7.4. Si el usuario presiona '**Cancelar**', saldrá de la ventana y si presiona '**Crear**', cuando los datos son correctos, saldrá una ventana de confirmación, anunciando la creación del equipo, como el de la figura 7.3.

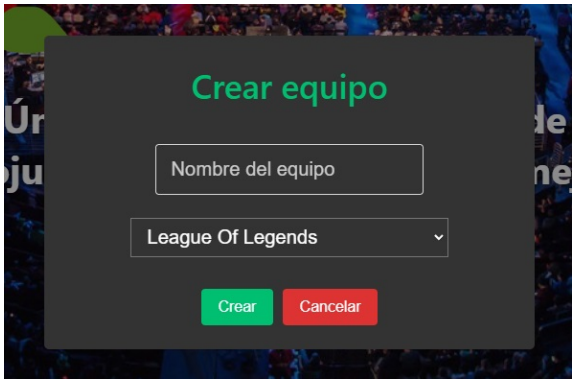

Figura 7.23: Interfaz crear equipo

Caso de uso: **Personalizar perfil**

Para que el usuario realice este caso de uso, tiene que acceder a la parte de '**Personalización**' del menú de usuario, figura 7.7, desde el menú superior en cualquier parte de la página. Al hacerlo, aparece la figura 7.24, donde se ve el avatar y el *banner* que se usa actualmente por el usuario y dos botones para cambiar estos dos.

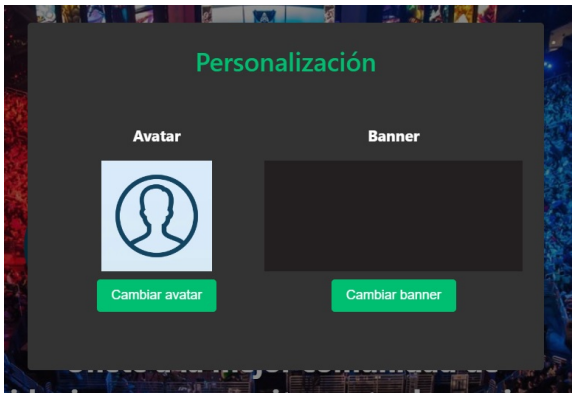

Figura 7.24: Interfaz personalización

Si quiere cambiar el avatar, pulsará el botón '**Cambiar avatar**' y aparecerá otra ventana para cambiarlo, figura 7.25, donde se ven todos los avatares que tiene, más el que usa actualmente con un borde verde. Si quiere cambiarlo, pulsará otro avatar y presionará el botón '**Aceptar**', a este nuevo avatar al pulsarlo se le pasará el borde verde. Si el usuario quiere cancelar el proceso pulsará '**Cancelar**' y saldrá de la ventana. Por otro lado, si el usuario quiere cambiar el *banner*, pulsará '**Cambiar banner**' en la figura 7.24. Al pulsarlo se abre esta ventana, figura 7.26, y cambiarlo funciona igual que con el caso del avatar.

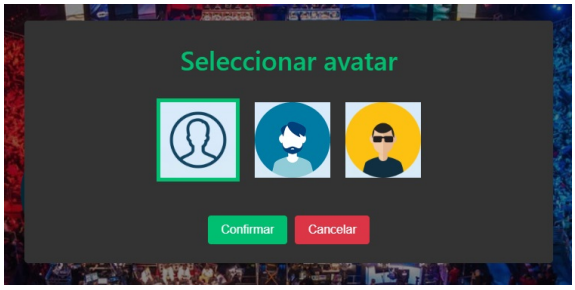

Figura 7.25: Interfaz cambiar avatar

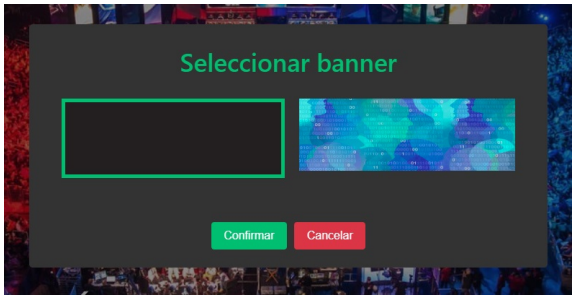

Figura 7.26: Interfaz cambiar *banner*

Caso de uso: **Ver información de otros jugadores**

Para que el usuario vea el perfil de otro usuario puede hacerlo desde diferentes sitios. Uno de ellos es viajar a la sección de *rankings* desde el menú superior, apareciendo esta ventana, figura 7.13. En ella podremos ver los mejores jugadores o equipos de cada videojuego. Si el usuario quiere ver el perfil de uno de ellos, pulsará su nombre y se abrirá esta ventana, figura 7.27, donde tiene: nombre, avatar, *banner*, información personal (fecha de nacimiento, género y país), información competitiva (torneos jugados y ganados y sus equipos) y sus videojuegos. También se puede acceder a esta funcionalidad pulsando los nombres de los jugadores en la pestaña comunidad, figura 7.14, en la sección para ver información de uno de sus equipos, figura 7.21 y en la sección de ver otros equipos, figura 7.28.

Caso de uso: **Ver información de otros equipos**

Para el último caso de uso, el usuario también se puede situar en la parte de *rankings*, accesible desde el menú superior, figura 7.13. Después, el usuario cambiará a ver los mejores equipos, pulsando el botón '**Equipos**'. Por último cuando quiera ver la información de uno de ellos, pulsará en su nombre y saldrá esta ventana, figura 7.28. En ella verá el videojuego, los torneos jugados y ganados y sus integrantes. También se puede acceder a esta funcionalidad presionando el nombre del equipo en la ventana general de la funcionalidad de equipos del menú de usuario, siempre que le hayan invitado a algún equipo, figura 7.19 y cuando ve el perfil de otro usuario, figura 7.27.

Figura 7.27: Interfaz perfil de otro usuario

Figura 7.28: Interfaz perfil de otro equipo

CAPÍTULO 8
Validación

A continuación, se va a hablar del proceso de validación realizado para el proyecto web. Esta parte es muy importante ya que será la que certifique que la interfaz que usan los usuarios es correcta en diseño y funcionalidades. Durante el desarrollo se ha pedido a gente cercana que analizara la página web cada cierto tiempo para detectar fallos de diseño y funcionalidades. Aún así, se necesita un método más formalizado para reflejarlo en el proyecto y este será los **diez principios sobre la heurística de Nielsen [16]**. Cumplir estos principios asegura un incremento significativamente positivo en la mejora de experiencia de usuario en la interfaz. Además, te ayuda a crear interfaces más accesibles, llamativas, eficientes y fáciles de usar por los usuarios.

Ahora vamos a enumerar y explicar los diez principios de la heurística de Nielsen y observar si el sistema los cumple:

1. **Visibilidad del estado del sistema**: El usuario debe saber donde está o que acción está realizando dentro de la web. El sistema cumple este requisito, ya que tanto en las secciones de la página como en las ventanas, hay un título grande en verde al comienzo de cada uno donde se dice dónde o qué está haciendo el usuario.

2. **Relación entre el sistema y el mundo real**: Este principio consiste en que la página tenga elementos reconocibles por el usuario que faciliten el uso de la web aunque sea la primera vez del usuario en la página. En este caso, en las secciones de la página y en los botones, hay iconos que cumplen esta función.

3. **Control y libertad del usuario**: El sistema dejará al usuario el control de ejecutar y cancelar acciones en todo momento. Las ventanas donde se realizan las acciones del proyecto, suelen tener un botón de cancelar para detener el proceso. También existe la posibilidad de pulsar fuera de la ventana que tendrá el mismo efecto que el botón de cancelar.

4. **Consistencia y estándares**: Como se indica, la página debe tener consistencia y seguir un orden respecto al diseño. Gracias al Material UI, los archivos *CSS* y la paleta de colores elegida, la página mantiene su estilo y orden en todo momento.

5. **Prevención de errores**: Consiste en prevenir que los usuarios hagan errores en sus acciones o avisar al usuario antes de realizar alguna acción importante. En este proyecto, los errores manuales que pueda cometer el usuario, se avisan mediante un mensaje de aviso en las ventanas. Además en determinadas acciones importantes, precio a la acción, emerge una ventana preguntando al usuario por su realización.

6. **Reconocimiento antes que recuerdo**: Trata de mostrar al usuario herramientas, iconos u otros elementos que haga intuir al usuario la acción que va a realizar. En el proyecto hay muchas herramientas e iconos que sirven para esto.

7. **Flexibilidad y eficiencia de uso**: Este principio consiste en adaptar las interfaces para todo tipo de usuarios mediante herramientas que aporten rapidez y facilidad. Como este proyecto no tiene interfaces ni ventanas muy complejas, esto no se ha tenido en cuenta.

8. **Estética y diseño minimalista**: Consiste en crear una interfaz que sea llamativa y que priorice la experiencia del usuario. Mediante los elementos de Material UI, la paleta de colores y la estructura de la web se ha conseguido una interfaz limpia y agradable.

9. **Ayudar a los usuarios a reconocer, diagnosticas y recuperarse de errores**: Como dice, ayuda al usuario a reconocer y solucionar errores. En ventanas donde hay formularios como el de registro o iniciar sesión, si rellenan algún campo con un valor no correcto, se muestra claramente el error y como solucionarlo.

10. **Ayuda y documentación**: El último principio consiste en aportar ayuda y documentación sobre algunos elementos en el proyecto web. Pese a que se ofrece todo tipo de ayuda para informar y salir de errores, no hay ningún tipo de documentación oficial ya que se ha considerado que no es necesaria.

En resumen, como se ha podido comprobar, se han cumplido todos los principios de Nielsen. Esto significa que la interfaz se ha construido correctamente, aunque siempre se puede mejorar algún aspecto como el de 'Ayuda y documentación'.

CAPÍTULO 9
Conclusiones

Una vez que se ha desarrollado y explicado todas las fases del proyecto, vamos a exponer un resumen general del mismo y algunas mejoras para futuras actualizaciones.

Debido al crecimiento constante del sector de los videojuegos como forma de entretenimiento y el interés generado por estos, muchos jugadores empezaron a profesionalizarse y a interesarse por los deportes electrónicos o *e-sports*, por lo que se ha propuesto y creado una aplicación web para unir a toda la comunidad competitiva de videojuegos. Antes de desarrollar la solución, se hizo un estudio de aplicaciones web parecidas a la propuesta para saber qué ofrecer y cómo diferenciarse. También, se expusieron varias metodologías para desarrollar el producto, seleccionando la metodología incremental debido a que era la más apropiada.

Comenzando por el desarrollo propio, primero se recogieron los requisitos que debía tener la web para funcionar, mediante herramientas como la búsqueda de información y los cuestionarios. Posteriormente, se definieron las funcionalidades mediante casos de uso y los objetos de la aplicación mediante diagramas de clase. Estos objetos se guardaron en una base de datos de *PostgreSQL*, formando diferentes tablas. Además, se hicieron bocetos de la interfaz, dando así una idea de la estructura y la forma gráfica.

A continuación, se detalló la arquitectura del sistema, todos los tipos de herramientas tanto generales, como de *frontend* y *backend*, y los lenguajes de programación usados. Luego, se mostró el código de las partes más relevantes del proyecto donde se están empleando las herramientas y lenguajes de programación descritos. Después, se exponen capturas de todas las interfaces de la web y cómo funcionan. Y por último, se hizo una validación de la interfaz.

Como conclusión, tanto los objetivos como el desarrollo han sido logrados de forma favorable. En futuras actualizaciones se podría desarrollar una interfaz para administradores, con el objetivo de hacer su trabajo más sencillo. También, se podrían desarrollar nuevas funcionalidades y publicar el portal web, puesto que ha sido desarrollado de forma local. Y para finalizar, lo más importante, seguir valorando la opinión de los usuarios y manteniendo activa la web para que más jugadores se unan a ella.

Bibliografía

[1] Informe de PwC sobre la industria del entretenimiento en España, 3 de noviembre de 2022. Consultado en https://www.pwc.es/es/sala-prensa/notas-prensa/2022/informe-entertainment-media-outlook-2022-2026.html.

[2] Página web Torneum. Consultado en https://torneum.com/.

[3] *How to Make an Esports Tournament Website?* Consultado en https://www.devteam.space/blog/how-to-develop-an-esports-tournament-website/#:~:text=Any%20esports%20website%20should%20allow,to%20buy%20esports-related%20merchandise.

[4] Página web ArenaGG. Consultado en https://www.arenagg.com/es/arena.

[5] Página web Torneos.GG. Consultado en https://torneos.gg/.

[6] Las 5 Mejores Metodologías de Desarrollo de Software. Consultado en https://gooapps.es/2022/10/27/las-5-mejores-metodologias-de-desarrollo-de-software/.

[7] Metodologías de desarrollo de software: ¿qué son? Consultado en https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html.

[8] Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa. Consultado en https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/.

[9] *Best Tips to Build E-Sports Websites*. Consultado en https://www.strikingly.com/content/blog/e-sports/.

[10] *Top 10 E-Sports Games in 2024*. Consultado en https://esportbet.com/games/.

[11] *The Best Esports Games for 2024*. Consultado en https://www.pcmag.com/picks/best-esports-games.

[12] Qué es PostgreSQL y sus principales ventajas. Consultado en https://ayudaleyprotecciondatos.es/bases-de-datos/que-es-postgresql-ventajas/.

[13] Cómo usar Dbeaver Community en el modelado de datos. Consultado en https://keepcoding.io/blog/usar-dbeaver-community/.

[14] ¿Qué es la arquitectura de tres niveles? Consultado en https://www.ibm.com/es-es/topics/three-tier-architecture.

[15] Introducción a los tokens web JSON Consultado en https://auth0.com/es/learn/json-web-tokens.

[16] Los 10 principios de Nielsen | Usabilidad web Consultado en https://digitalvar.es/articulos-diseno-web/10-principios-de-nielsen/.

APÉNDICE A
Formulario de requisitos

En este apéndice tenemos las capturas del formulario hecho por *Google Forms* para la captura de requisitos.

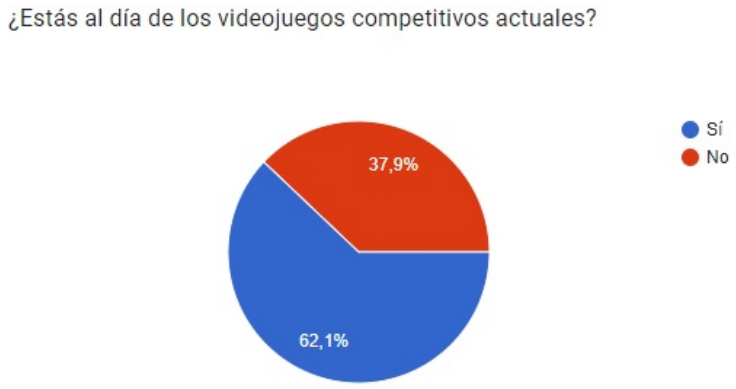

Figura A.1: Formulario pregunta 1

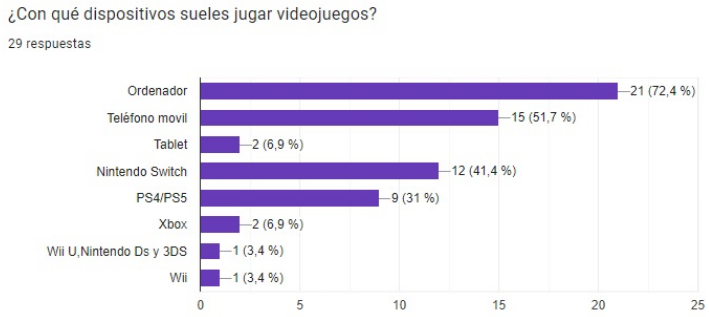

Figura A.2: Formulario pregunta 2

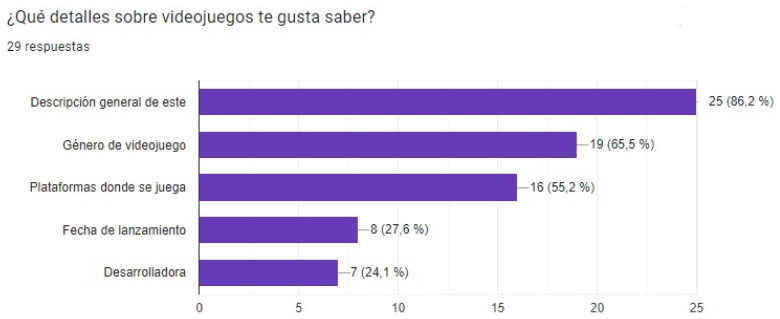

Figura A.3: Formulario pregunta 3

Figura A.4: Formulario pregunta 4

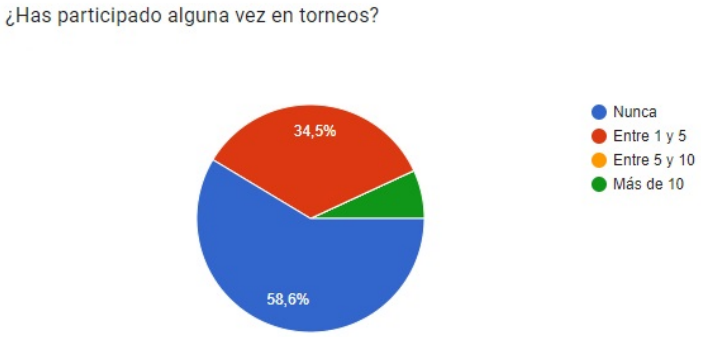

Figura A.5: Formulario pregunta 5

Figura A.6: Formulario pregunta 6

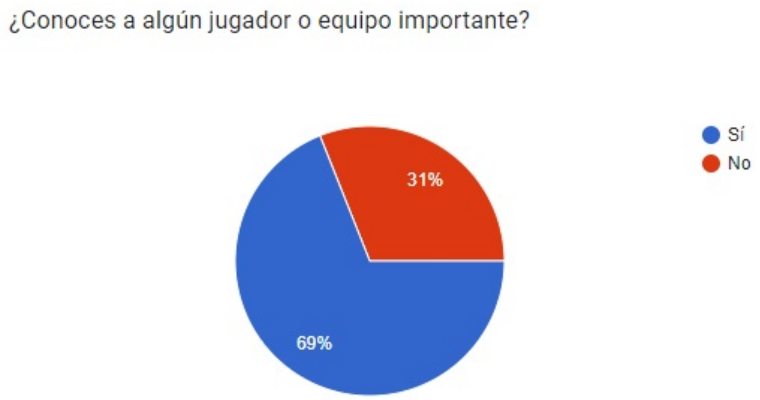

Figura A.7: Formulario pregunta 7

Figura A.8: Formulario pregunta 8

Figura A.9: Formulario pregunta 9

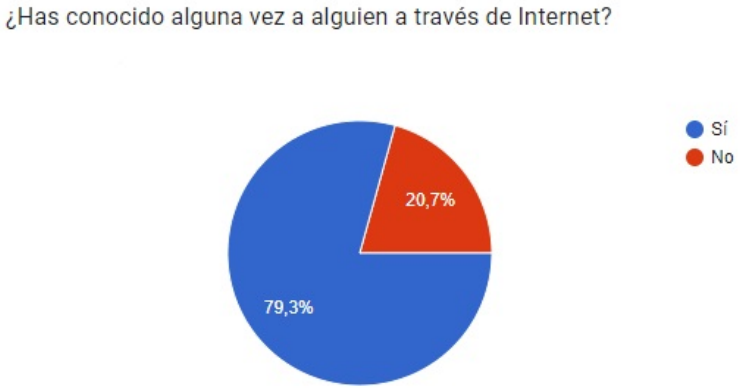

Figura A.10: Formulario pregunta 10

Figura A.11: Formulario pregunta 11

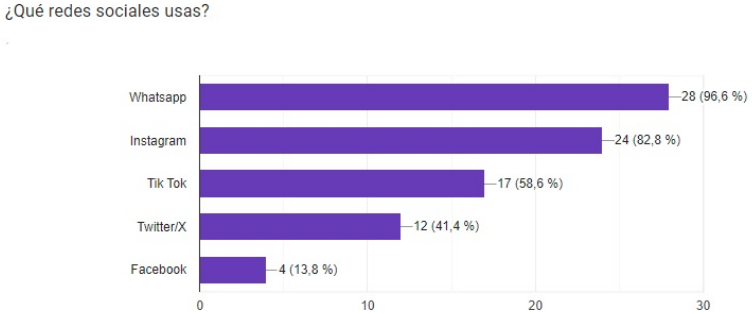

Figura A.12: Formulario pregunta 12

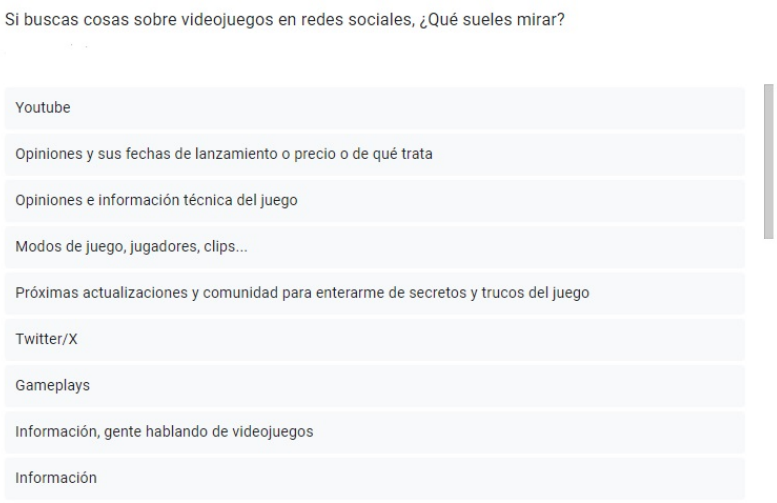

Figura A.13: Formulario pregunta 13

Figura A.14: Formulario pregunta 14

Figura A.15: Formulario pregunta 15

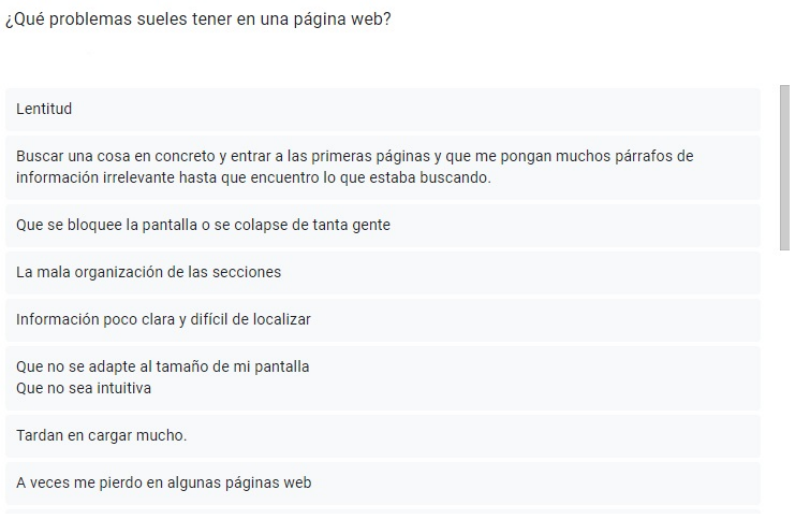

Figura A.16: Formulario pregunta 16

APÉNDICE B
Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**			X	
ODS 5. **Igualdad de género.**		X		
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**			X	
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG con los ODS.

Este proyecto está relacionado con el mundo competitivo de los videojuegos y los deportes electrónicos con la finalidad de formar una comunidad, competir y evolucionar en esta escena. A simple visita, parece que no esté relacionado con ninguno de los Objetivos de Desarrollo Sostenible (ODS). Sin embargo, la aplicación web desarrollada contribuye a estos objetivos de forma positiva, en concreto a cuatro: *Educación de calidad*, *Igualdad de género*, *Energía asequible y no contaminante* e *Industria, innovación e infraestructuras*.

Empezando con el Objetivo de Desarrollo Sostenible número cuatro, ***Educación de calidad***, este proyecto tiene como objetivo la formación y desarrollo de una comunidad competitiva en el mundo de los deportes electrónicos. Por tan-

to, la construcción de estas competiciones parte de una educación de calidad que han de cumplir los jugadores. Esta educación de los jugadores se adquiere de una enseñanza del deporte con principios y valores como el trabajo en equipo, la solidaridad, la superación personal, disciplina, tolerancia, entre muchos otros.

A continuación, el quinto Objetivo de Desarrollo Sostenible, ***Igualdad de género***, cuyo objetivo implica que todas las personas tengan los mismos derechos, recursos y oportunidades independientemente de su identidad de género. En la comunidad que se crea alrededor de *E-ZONE* se cumplen estos valores y objetivos. Por ejemplo, cuando un nuevo usuario quiere registrarse en la web, tiene la libertad de inscribirse con el género con el que se sienta identificado.

Respecto al Objetivo de Desarrollo Sostenible número siete que es ***Energía asequible y no contaminante***, *E-ZONE* trata de un proyecto *online* por tanto, se descarta el uso del papel y otros materiales para promocionar o dar información sobre el proyecto web desarrollado que pueden llegar a contaminar. Por otro lado, no se descarta que en un futuro patrocinadores o colaboradores que se unan al proyecto promuevan proyectos medioambientales y conciencien a la comunidad mediante elementos y acciones relacionados con el mundo de los videojuegos como torneos o elementos de personalización extras.

Por último, tenemos el noveno Objetivo de Desarrollo Sostenible, ***Industria, innovación e infraestructuras***, este objetivo es el más importante, pues *E-ZONE* pertenece a una industria que está en constante crecimiento que obliga a que se desarrollen más proyectos innovadores de forma libre. Durante la creación del proyecto se han usado herramientas abiertas a toda la industria, ya que son de código abierto, por lo que se está dando a conocer al público instrumentos realizados por los trabajadores y usuarios de la comunidad. Además, la alta participación en el mundo competitivo de los videojuegos conlleva que la infraestructura industrial siga perfeccionándose, generando así más reconocimiento y aportando cada vez más innovación a este campo.

En conclusión, el proyecto *E-ZONE* está relacionado con el mundo de los videojuegos y los deportes electrónicos con el propósito de la creación de una comunidad centrada en el entorno competitivo, que consigue contribuir de forma favorable a los Objetivos de Desarrollo Sostenible (ODS). En concreto, *E-ZONE* es un proyecto que educa en valores, en igualdad, con bajo impacto en el medio ambiente y que proyecta un futuro innovador y de desarrollo para el mundo.