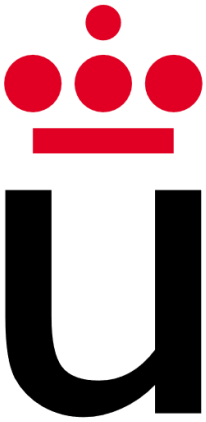


Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN DISEÑO Y DESARROLLO DE VIDEOJUEGOS

Curso Académico 2022/2023

Trabajo Fin de Grado

Desarrollo de una plataforma de distribución de videojuegos llamada DoubleTapParty

Autores: Miguel Ángel Arcones Ríos y Daniel Muñoz Serrano

Director: Julio Guillén García

Quisiéramos agradecer a todas las personas que nos han presentado su ayuda en esta recta final de la titulación.

En primer lugar, a nuestro tutor Julio por aceptar un proyecto como este y por todos los consejos que nos has ido dando.

También agradecer a nuestros amigos y familiares por apoyarnos incondicionalmente durante todo el proceso.

Muchas gracias a todos.

Resumen

En este trabajo, se diseña y se desarrolla una plataforma web de distribución de videojuegos donde es necesario la utilización de al menos dos dispositivos para su uso, un dispositivo a modo de pantalla y otro dispositivo a modo de controlador.

Esta plataforma también tendrá una gestión social, permitiendo a los usuarios añadir amigos e invitarlos a jugar ya que albergará videojuegos tanto de un solo jugador como de varios. Otra característica a destacar es la posibilidad de los usuarios de subir creaciones propias a la plataforma.

Como estudio teórico previo al desarrollo de la herramienta, se ha realizado una investigación sobre los diferentes *frameworks* posibles para el desarrollo de páginas web, las bibliotecas de comunicaciones de tiempo real existentes y los proveedores de servicios en la nube. También se ha realizado un análisis de las plataformas de distribución de videojuegos actuales para reconocer aquellas características y funcionalidades importantes.

Por último, se han realizado dos pruebas con personas físicas, una prueba individual para comprobar las comunicaciones entre el controlador y el visor, y otra con múltiples personas físicas, para comprobar la estabilidad en juegos multijugador. También se realizaron pruebas de rendimiento y esfuerzo en el servidor.

Palabras Clave

Móvil

Ordenador

Plataforma de distribución de videojuegos

Videojuegos

Amistad

Grupos

Social

Multijugador

.Net Core

SignalR

Abstract

In this work, a web-based video game distribution platform is designed and developed, where the use of at least two devices is necessary: one device as a display and another device as a controller.

This platform will also have social management capabilities, allowing users to add friends and invite them to play, as it will host both single-player and multiplayer video games. Another notable feature is the ability for users to upload their own creations to the platform.

As a theoretical study prior to the tool's development, research has been conducted on various possible *frameworks* for web page development, existing real-time communication libraries, and cloud service providers. An analysis of current video game distribution platforms has also been carried out to identify important features and functionalities.

Lastly, two tests have been conducted with individuals: an individual test to verify the communication between the controller and the display, and another test with multiple individuals to assess stability in multiplayer games. Performance and server load tests were also performed.

Keywords

Mobile

Computer

Video Game Distribution Platform

Video Games

Friendship

Groups

Social

Multiplayer

.Net Core

SignalR

Índice de Contenidos

Índice de Ilustraciones

Índice de Tablas

Glosario

API

Application Programming Interface se refiere a un conjunto de reglas y protocolos que permiten la comunicación entre diferentes aplicaciones de software. Básicamente, una API define la forma en que dos programas pueden interactuar entre sí.

Back-end

Parte de un sistema o aplicación que se encarga de la lógica de negocio, el procesamiento de datos y la interacción con bases de datos u otros sistemas. Es la capa "detrás" de la interfaz de usuario y se encarga de manejar la lógica y la funcionalidad interna de una aplicación.

Benchmarks

Pruebas o mediciones realizadas para evaluar y comparar el rendimiento de hardware, software o sistemas en términos de velocidad, eficiencia, capacidad, precisión u otros aspectos relevantes.

Blob

Binary Large Object es un tipo de dato utilizado para almacenar datos binarios en una base de datos o sistema de almacenamiento. Un Blob puede contener cualquier tipo de información binaria, como imágenes, audio, video, documentos, etc.

Cookies

Son pequeños archivos de texto que los sitios web guardan en el navegador del usuario con el fin de recordar información específica sobre la visita y permitir ciertas funcionalidades. Estas cookies contienen datos como preferencias del usuario, información de inicio de sesión, datos de seguimiento y otros detalles relevantes.

Crosspad

Se refiere a una serie de botones dispuestos en forma de cruz o una almohadilla direccional que se utiliza para controlar el movimiento en juegos. Es comúnmente encontrado en mandos de juegos para consolas como PlayStation, Xbox y Nintendo.

Front-end	Se refiere a la parte de un sistema o aplicación que interactúa directamente con los usuarios. Es la capa visible y accesible para los usuarios finales, que incluye la interfaz de usuario, la presentación visual y la interacción con los elementos visuales.
Gameplay	Se refiere a la grabación o reproducción de la experiencia de juego de un videojuego. Es común que los jugadores graben sus partidas mientras juegan y compartan esas grabaciones en forma de videos llamados *gameplays*.
Host	Cualquier dispositivo o sistema que es capaz de comunicarse y ofrecer servicios a través de una red. Puede ser una computadora, un servidor, un enrutador, un dispositivo *IoT* u otro dispositivo similar.
Joystick	Dispositivo de entrada utilizado en los mandos de juegos para controlar la dirección o movimiento en los juegos. Consiste en una palanca que se puede mover en diferentes direcciones para controlar la posición del personaje o los objetos en el juego.
Layout	Disposición o diseño visual de los elementos en una interfaz gráfica o página web. Define cómo se organizan y se presentan los componentes visuales, como botones, texto, imágenes y otros elementos, en un espacio determinado.
Long Polling	Técnica utilizada en programación web para lograr una comunicación bidireccional entre un servidor y un cliente de manera asíncrona. A diferencia de las solicitudes HTTP tradicionales, donde el cliente realiza una solicitud al servidor y espera una respuesta inmediata, con *Long Polling* el servidor puede enviar una respuesta al cliente de manera asíncrona cuando haya nuevos datos disponibles.

Rating	Sistemas de calificación o valoración que permiten a los usuarios expresar su opinión o evaluación sobre un producto, servicio, contenido o cualquier otro elemento presente en un sitio web.
Server-Sent Events	Tecnología basada en el protocolo HTTP que permite al servidor enviar actualizaciones o eventos en tiempo real al cliente de manera unidireccional. Proporciona una forma sencilla y eficiente de enviar información desde el servidor al navegador sin la necesidad de que el cliente realice solicitudes periódicas.
SQL	*Structured Query Language*, es un lenguaje de programación diseñado para administrar y manipular bases de datos relacionales. Es utilizado para gestionar la estructura de las bases de datos, realizar consultas para recuperar, insertar, actualizar y eliminar datos, así como para definir permisos y controlar la seguridad de la información almacenada en las bases de datos.
Streaming	Tecnología y método de distribución de contenido multimedia que permite reproducir archivos de audio, video o cualquier tipo de datos en tiempo real, sin necesidad de descargarlos por completo antes de su reproducción. El *streaming* se utiliza ampliamente en plataformas de transmisión en vivo, servicios de música, videos a la carta y otras aplicaciones de entretenimiento en línea.
URL	*Uniform Resource Locator*, es una dirección única que se utiliza para identificar y acceder a recursos en la web. Básicamente, es la forma estándar de especificar la ubicación de un recurso, como una página web, una imagen, un archivo, un servicio, entre otros, en Internet.
WebSockets	Tecnología de comunicación bidireccional en tiempo real que permite una conexión persistente entre un servidor y un cliente a través de un único socket TCP. A diferencia del modelo HTTP tradicional, donde el cliente realiza solicitudes al servidor y este responde, en WebSockets la comunicación puede ser iniciada tanto por el cliente como por el servidor, lo que facilita una comunicación bidireccional.

Capítulo 1

Introducción

Descripción del problema

Este proyecto tiene como objetivo la creación de una plataforma de distribución de videojuegos en línea que permita a los jugadores utilizar sus teléfonos móviles como controladores tanto para la navegación en la plataforma como para los videojuegos alojados en ella. Esta plataforma se diferencia de otras por su capacidad para utilizar dispositivos móviles como controladores. Además, el proyecto incluye el desarrollo de un videojuego multijugador que aprovechará al máximo las funcionalidades de la plataforma web.

Para jugar en la plataforma, los usuarios necesitan conectarse a la web a través de un navegador en su ordenador personal (*Personal computer - PC*) y en su teléfono móvil. La pantalla del PC mostrará la aplicación principal y los juegos, mientras que desde el teléfono móvil se visualizará un controlador que permitirá a los jugadores interactuar con la plataforma y los juegos. Además, los usuarios pueden conectarse a sesiones de amigos que ya tengan un PC conectado, lo que les permitirá jugar entre ellos.

Motivación

En la actualidad, el mercado de los videojuegos se encuentra en constante crecimiento, y esto ha dado lugar a la aparición de numerosas plataformas de distribución digital de videojuegos. Muchas de estas plataformas se enfocan en ofrecer juegos de grandes compañías, mientras que otras se enfocan en ofrecer juegos independientes. En este contexto, surge la motivación de desarrollar una plataforma enfocada en los juegos multijugador local para desarrolladores independientes, diferenciándose del resto en la capacidad de controlar la aplicación y los juegos a través de un dispositivo móvil, el cuál otorga mayor versatilidad al desarrollador pudiendo configurar el controlador del teléfono de la manera más conveniente para su juego.

Es por eso que el desarrollo de una plataforma con este enfoque puede ser una gran oportunidad para los desarrolladores y algo novedoso para los jugadores. Esta plataforma puede ofrecer herramientas y recursos que les permitan crear y publicar sus juegos, y brindar a los jugadores un lugar donde puedan encontrar y disfrutar de una amplia variedad de juegos multijugador local de alta calidad.

Además, al ofrecer una plataforma unificada, se facilita el acceso a estos juegos para los usuarios, lo que a su vez puede aumentar la popularidad y generar beneficios a los desarrolladores. Esto puede ser especialmente importante para los desarrolladores

independientes que pueden tener dificultades para promocionar sus juegos y llegar a una audiencia más amplia.

Objetivos

Investigación

- Estudiar las plataformas de distribución de videojuegos actuales, análisis de sus características y funcionalidades.
- Evaluar las diferentes tecnologías de desarrollo de software posibles para el desarrollo de la aplicación.
- Desarrollar un método de comunicación en tiempo real entre distintos dispositivos.

Arquitectura

- Construir la aplicación en una arquitectura descentralizada, en la que cada juego se ejecute en un servicio web diferente.
- Implementar la aplicación dentro de un proveedor de servicios de la nube y aprovechar sus capacidades.
- Adquirir un dominio y enlazarlo a la aplicación desarrollada.

Desarrollo

- Crear un código escalable.
- Crear un *framework* que permita desarrollar fácilmente juegos para incorporarlos en la aplicación.
- Proveer a la aplicación de juegos externos incorporando las funcionalidades desarrolladas.
- Permitir incorporar juegos *HTML5* y *WebGL*.
- Evaluar y testear la aplicación.

Planificación inicial

En el desarrollo de proyectos, contar con una planificación inicial sólida es vital para garantizar un resultado final dentro de la fecha límite establecida. Para facilitar esta planificación, se ha utilizado un diagrama de *Gantt* que expone de manera clara las principales tareas a realizar a lo largo del desarrollo del proyecto. Se estimó que el desarrollo completo tomará aproximadamente 7 meses, sin tener en cuenta el desarrollo de la memoria.

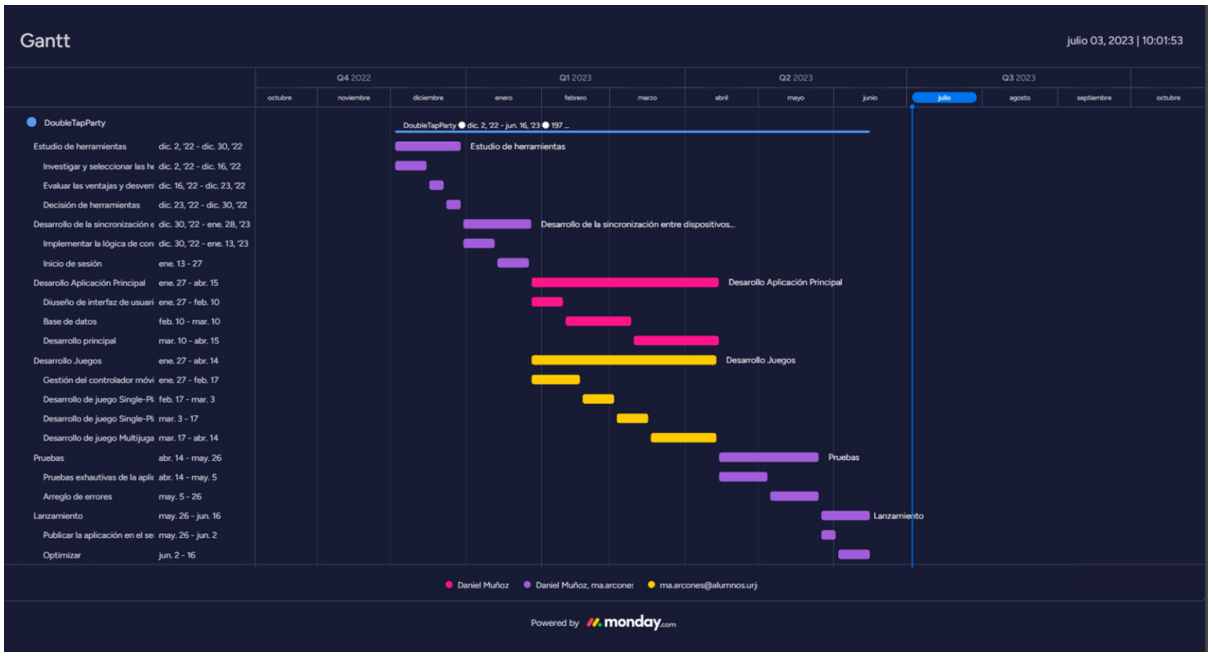


Ilustración 1. Diagrama Gantt del proyecto

Reparto del trabajo

El desarrollo se ha separado en tres partes, el encargado del desarrollo de la aplicación principal será Daniel Muñoz, los servicios de los juegos será desarrollado por Miguel Ángel Arcones y una parte común compuesta por los sistemas de comunicación entre dispositivos y configuración de proyectos, despliegues y tecnologías. Estas tareas serás explicadas con detalle a lo largo del capítulo 4.

Metodología empleada

Teniendo en cuenta que el equipo estaba compuesto por solo dos personas se decidió utilizar un enfoque ágil de desarrollo Kanban. Este enfoque permitió realizar ajustes en tiempo real a medida que surgían nuevos requisitos o prioridades cambiantes, al no estar atados a un marco de trabajo rígido.

Para garantizar la eficiencia en el desarrollo, se hizo uso del diagrama *Gantt* que se había creado durante la planificación inicial. Esto permitió establecer prioridades para las tareas predecesoras de las tareas dependientes, evitando cuellos de botella durante el desarrollo.

Se comenzó por crear un tablero *Kanban* en la aplicación *Trello*, utilizando los estados de tarea como *Backlog*, *To do*, *Doing*, *Review* y *Done*. Se han definido las tareas a partir de los requisitos iniciales, se les asignaron prioridades y se repartieron las tareas en función de su ámbito.

Debido a las circunstancias laborales de los miembros del equipo, se estableció una frecuencia de reuniones recurrentes cada dos semanas para unificar los avances de cada rama de desarrollo y discutir la posibilidad de añadir nuevos requisitos y tareas al proyecto.

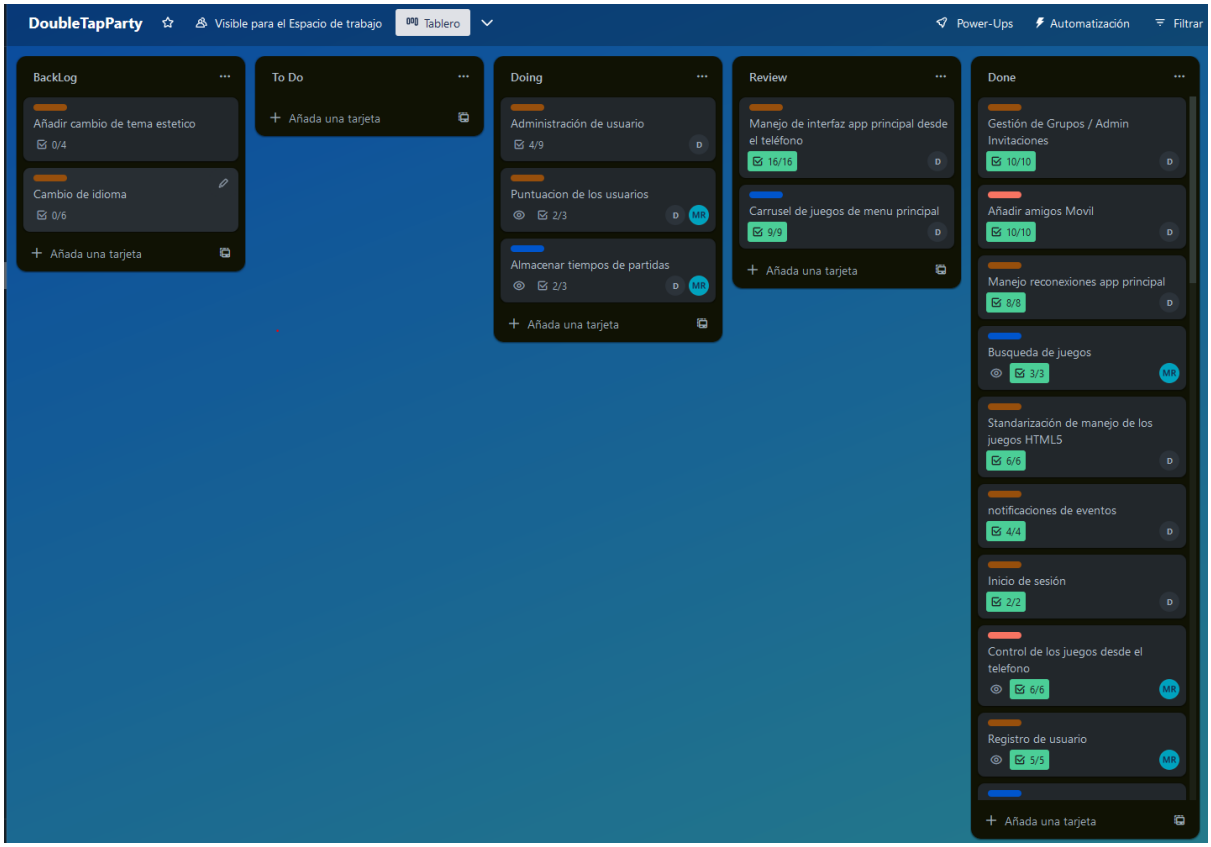


Ilustración 2. Tablero Kanban del proyecto

Por último, se utilizó *Repos* de *Azure Devops* como herramienta de repositorio compartido y control de versiones, y *SourceTree* como interfaz gráfica para optimizar el proceso de desarrollo.

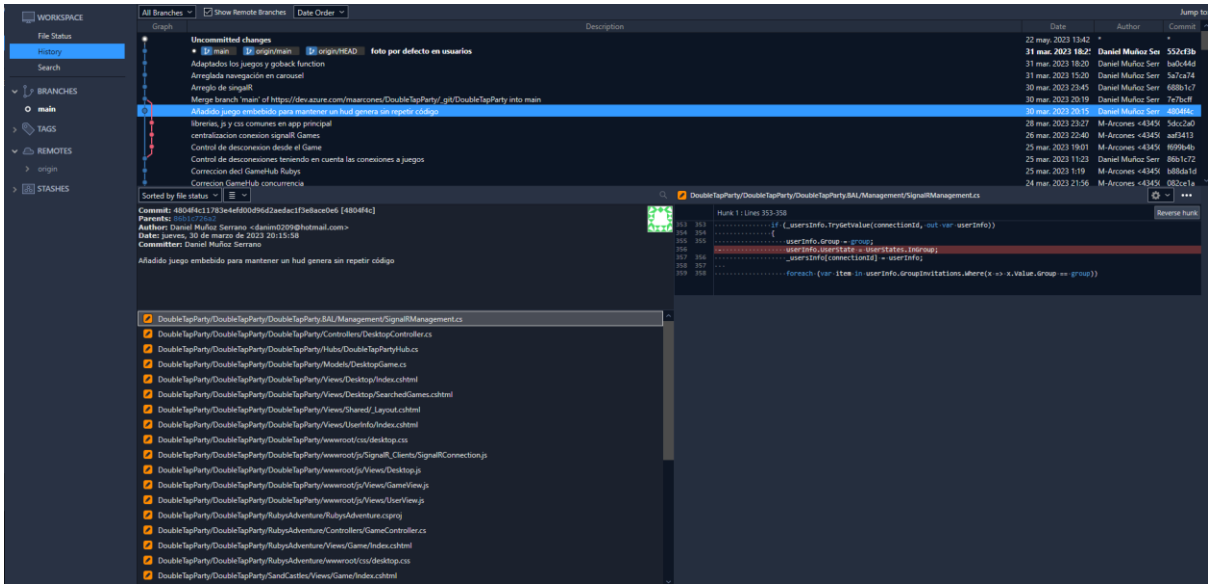


Ilustración 3. Repositorio visto desde *SourceTree*

Estructura de la memoria

La memoria se estructura en 3 secciones:

1. Análisis del desarrollo: En los dos primeros capítulos se define el problema, la metodología para abordarlo y los objetivos que se desean cumplir durante el desarrollo. También se expone el estudio previo realizado previo al diseño y desarrollo de la aplicación, incluyendo los casos de estudio y alternativas planteadas en el desarrollo.
2. Desarrollo: Los capítulos 3 y 4 se centran en explicar en detalle diferentes aspectos de la aplicación desarrollada. En el capítulo 3 se muestra el diseño y uso final de la aplicación y en el capítulo 4 se realiza una descripción informática de la implementación.
3. Conclusiones: Por último, los capítulos 5 y 6 presentan los resultados finales del proyecto, en qué medida se han cumplido los objetivos planteados y las lecciones aprendidas durante el proyecto. Además, se exploran posibles líneas de mejora a futuro que podrían seguirse después de finalizar el proyecto.

Capítulo 2

Marco Teórico

Evolución histórica de las plataformas de distribución de videojuegos

La evolución histórica de las plataformas de distribución de videojuegos ha experimentado muchas etapas de desarrollo y transformación.

En las décadas de 1970 y 1980, se dieron los primeros pasos en esta industria. En 1972 apareció *Magnavox Odyssey*, la primera plataforma de videojuegos disponible para el uso doméstico con un precio de $100 y lograron vender alrededor de 350.000 unidades entre 1972 y 1975. (Bedi, 2019)

Ilustración 4. Magnavox Odyssey

En 1977, apareció la consola *Atari 2600* (originalmente llamada VCS - *Video Computer System*), considerada el origen de la industria actual del videojuego. Esta consola introdujo el formato de cartuchos intercambiables, permitiendo a los usuarios adquirir nuevos juegos y sentando las bases para la distribución de juegos en formato físico. (Lendino, 2018)

Ilustración 5. Atari 2600

A principios de los años 80, la industria de los videojuegos se vio afectada por la denominada "crisis de los videojuegos". Durante esta crisis el mercado se saturó con una gran cantidad de juegos de baja calidad, provocando una caída drástica en las ventas y los ingresos de los videojuegos. A raíz de esto, muchas empresas se vieron obligadas a cerrar o declararse en quiebra, provocando una crisis económica para la industria. Esta crisis llevó a una reevaluación de las prácticas comerciales y de producción, buscando ofrecer a los usuarios productos de mayor calidad y con experiencias de juego más satisfactorias. La crisis de los videojuegos estableció la importancia de la calidad de los juegos como un factor determinante para la distribución. (Belli & López Raventós, 2008)

A pesar de la crisis, en 1983 *Nintendo* lanzó el *Nintendo Entertainment System* (NES) obteniendo un éxito clave para revitalizar la industria y restaurar la confianza de los consumidores en los videojuegos, ya que presentaba juegos de alta calidad y una experiencia de juego más inmersiva. *Nintendo* implementó un estricto control de calidad y un programa de licencias para los desarrolladores de juegos de *NES*, permitiendo mantener altos estándares de calidad y garantizar que los juegos lanzados en la plataforma cumplieran con los requisitos de *Nintendo*. La *NES* fue también la plataforma que presentó algunas de las franquicias más reconocidas hasta el día de hoy: *Super Mario Bros*, The *Legend of Zelda* y *Metroid*. Otro dato a destacar de la *NES* era su sistema de cartuchos intercambiables como formato de distribución de videojuegos, permitiendo una mayor flexibilidad y expansión del catálogo de juegos. (Diskin, 2004)

Ilustración 6. Nintendo Entertainment System (NES)

El periodo final de la década de 1980 y principios de la década de 1990 fue un periodo de intensa competencia entre *Sega* con el *Sega Genesis* (también conocido como *Mega Drive*) y *Nintendo* con el *Super Nintendo Entertainment System* (SNES). Ambas consolas presentaban mejoras significativas en términos de gráficos y sonidos en comparación con las generaciones anteriores.

Sega Genesis se lanzó en 1988, destacando por su procesador de 16 *bits* y su capacidad para mostrar colores más vivos en pantalla. Otra mejora que trajo consigo fue el concepto de accesorios, como el *Sega CD* y el *Sega 32X*, ampliando así las capacidades de la consola y ofreciendo experiencias de usuario adicionales. (Scullion, 2021)

Ilustración 7. Sega Genesis

SNES se lanzó en 1990, contando también con un procesador de 16 *bits,* pero presentaba mayor detalle en el apartado gráfico. *SNES* también contaba con un chip de sonido personalizado, ofreciendo un sonido mejorado y una experiencia de audio más inmersiva.

Ilustración 8. Super Nintendo Entertainment System (SNES)

La competencia entre ambas compañías provocó el lanzamiento de numerosos juegos icónicos para sus respectivas consolas, encontrando el *Super Mario World*, *The Legend of Zelda: A Link to the Past* y *donkey Kong Country* en *SNES* y *Sonic the Hedgehog*, *Streets of Rage* y *Phantasy Star* en *Sega Genesis*.

Ilustración 9. Videojuegos icónicos de Sega y Nintendo

Esta rivalidad llevó a un avance significativo en la calidad y la diversidad de los juegos disponibles en el mercado.

En 1994 se *Sega* lanzó *Sega Channel*, siendo la primera plataforma de distribución digital de videojuegos, permitiendo a los usuarios descargar y jugar una amplia selección de juegos directamente a su consola *Sega Genesis* a través de una conexión por cable, eliminando la necesidad de comprar cartuchos físicos. (Foster & Oppermann, 1996)

Ilustración 10. Sega Channel

Durante la segunda mitad de la década de 1990 se produjo un avance significativo en la representación gráfica de los videojuegos, pasando de unos gráficos bidimensionales a unos tridimensionales. Otra característica importante de esta época es la introducción de los *CD-ROM* como medio de almacenamiento, ofreciendo una mayor capacidad en comparación con los cartuchos utilizados hasta el momento. (Wolf & Perron, 2005)

En 1994 se lanzó la *PlayStation* de *Sony* con mucho éxito en el mercado de las consolas, desafiando a las compañías dominantes con sus respectivas consolas, el *Sega Saturn* y la *Nintendo 64*.

Ilustración 11. Videoconsola PlayStation, Sega Saturn y Nintendo 64

A finales de la década de 1990 y principios de la década de 2000 se produjo un gran avance en el rendimiento de las consolas, apareciendo la *PlayStation 2*, la *Xbox* y el *GameCube*, ofreciendo unos gráficos mejorados, mayor potencia de procesamiento y mayor capacidad de almacenamiento en comparación con sus predecesores. En esta etapa también se produjo una introducción del juego en línea y el juego multijugador en red ya que la *PlayStation 2* y la *Xbox* permitían a los jugadores conectarse a internet y jugar en línea. (Belli & López Raventós, 2008)

Ilustración 12. Videoconsola PlayStation 2 y Xbox

Durante esta etapa también se realizó una diversificación en las opciones de distribución, ya que además del formato *DVD*, se comenzaron a utilizar nuevas formas de distribución como la descarga digital y las tiendas en línea, apareciendo *Xbox Live* de *Microsoft* y *PlayStation Network* de *Sony*. (Parente, 2012)

Ilustración 13. Plataformas de descargas digital de PlayStation y Microsoft

Otras plataformas de distribución de videojuegos que han ido apareciendo hasta la actualidad serían *Battle.net* de *Blizzard* lanzada en 1996, *Steam* de *Valve Corporation* lanzado en 2003, *Origin* de *Electronic Arts* (EA) lanzado en 2011 y *Epic Games Store* de *Epic Games* lanzada en 2018.

Ilustración 14. Plataformas actuales de distribución digital

Complementos teóricos

Los sistemas de distribución de videojuegos emplean diferentes arquitecturas en función de sus necesidades y objetivos específicos. Cada una de estas arquitecturas tiene sus propias ventajas e inconvenientes, lo que permite adaptarse a distintos escenarios y requisitos. Se pueden encontrar las siguientes arquitecturas:

Descarga Directa

La arquitectura de descarga directa es la forma más común en la distribución de videojuegos. En este enfoque, los usuarios acceden al videojuego a través de un servidor centralizado, desde donde deben descargar el juego completo en sus dispositivos antes de poder disfrutarlo.

Las ventajas que ofrece este tipo de arquitectura son las siguientes:

1. Acceso inmediato al juego: Una de las principales ventajas de la descarga directa es que los usuarios pueden acceder al juego de inmediato una vez que se ha completado la descarga. No es necesario esperar la entrega física del juego o realizar una instalación a través de discos. Esto proporciona una experiencia de acceso más rápida y sencilla a los jugadores.

2. Mayor control de calidad y seguridad: Al utilizar la arquitectura de descarga directa, los desarrolladores y editores de juegos tienen un mayor control sobre la calidad y la seguridad del contenido. Pueden realizar pruebas exhaustivas antes del lanzamiento y asegurarse de que los usuarios obtengan la mejor experiencia posible. Además,

permite implementar medidas de seguridad para proteger el juego contra la piratería y el acceso no autorizado.
3. Actualizaciones y contenido adicional: La descarga directa también facilita la distribución de actualizaciones, parches y contenido adicional del juego. Los desarrolladores pueden lanzar correcciones de errores, mejoras de rendimiento, nuevas características y expansiones de contenido directamente a través de la propia plataforma de descarga. Esto permite a los jugadores mantener sus juegos actualizados y acceder a nuevas experiencias sin tener que admitir una nueva versión física del juego.

Los inconvenientes que trae consigo esta arquitectura son:

1. Requisitos de ancho de banda y almacenamiento: La descarga directa de juegos requiere una conexión a internet estable y de alta velocidad para descargar archivos de gran tamaño. Esto puede ser un problema para aquellos usuarios con conexiones de internet lentas o limitadas. Además, los juegos descargados ocupan espacio de almacenamiento en el dispositivo del usuario, lo que puede ocasionar problemas en dispositivos con poca memoria de almacenamiento.

2. Tiempo de descarga: El tiempo necesario para la descarga de un juego completo puede ser considerable, especialmente para aquellos con conexiones de internet más lentas ya que puede ocasionar una espera larga y prolongada antes de poder disfrutar del juego.

3. Falta de resistencia a caídas de conexión: La descarga directa puede ser problemática en caso de interrupciones o caídas de conexión a internet. Si la conexión se pierde durante la descarga, es posible que el usuario deba reiniciar la descarga, ocasionando un sentimiento de frustración para el usuario.

La descarga directa es una arquitectura utilizada en los sistemas de distribución de videojuegos que ofrece acceso inmediato al juego y proporciona un mayor control de calidad, pero requiere de una buena conexión, puede ocasionar tiempos de descarte prolongados y puede ser susceptible a interrupciones de conexión. (Vaudour & Heinze, 2020)

Streaming de videojuegos

El *streaming* de videojuegos se trata de una arquitectura de distribución que permite a los jugadores disfrutar de sus videojuegos sin la necesidad de descargarlos por completo en sus dispositivos. Esto se lleva a cabo mediante la transmisión en tiempo real desde servidores remotos a través de conexión a internet. Otra característica a destacar de este tipo de arquitectura es que no requiere de espacio de almacenamiento, ya que el juego no se descarga en el dispositivo del usuario, sino que se corre directamente en el servidor remoto.

Las principales ventajas que ofrece estas arquitecturas son:

1. Acceso instantáneo a un catálogo de juegos: Los jugadores tienen acceso a una amplia biblioteca virtual de videojuegos sin necesidad de comprarlos individualmente. Pueden escoger cualquiera dentro de la biblioteca y lanzarlo al instante eliminando las esperas por descarga.

2. Actualizaciones automáticas: Esta arquitectura ofrece la automatización de actualizaciones y parches ya que estos se ejecutan directamente en los en los servidores remotos sin la necesidad de que el jugador tenga que realizar ninguna acción.

3. No se requiere espacio de almacenamiento: Dado que los videojuegos no se descargan en el dispositivo del usuario, no necesita de espacio de almacenamiento elevado para disfrutarlo. Esto es beneficioso para aquellos dispositivos con memoria de almacenamiento limitada.

4. Amplia compatibilidad: Este tipo de arquitectura puede ser compatible con una amplia variedad de dispositivos, desde ordenadores y consolas hasta dispositivos móviles y *smarts TVs*, permitiendo al usuario disfrutar de los juegos sin adquirir un hardware específico.

Algunos inconvenientes de aplicar la arquitectura de *streaming* de videojuegos son:

1. Requisitos de ancho de banda y latencia: El *streaming* de videojuegos requiere de una conexión a internet rápida y estable para transmitir el juego sin problemas. Los usuarios con conexiones lentas o inestables pueden experimentar retrasos, congelaciones de pantalla o calidad de imagen reducida. Además, la latencia puede afectar a la respuesta del juego empeorando la experiencia del jugador.

2. Dependencia de la conectividad a internet: El *streaming* de videojuegos depende de una conexión a internet constante y fiable, si la conexión se interrumpe, puede afectar negativamente a la experiencia de juego, causando interrupciones o la desconexión completa.

La arquitectura de *streaming* de videojuegos ofrece acceso instantáneo a un catálogo de juegos sin necesidad de descargarlos por completo y proporciona una experiencia de juego rápida y flexible en diferentes dispositivos. Sin embargo, requiere una conexión a internet estable y rápida, y puede experimentar limitaciones en términos de ancho de banda y latencia. (Wu et al., 2014) (Xuew et al., 2014)

Redes de distribución de contenido

Las redes de distribución de contenido (*Content Delivery Network* - CDN) se centran en la colocación de copias del contenido en servidores distribuidos en todo el mundo. Estos servidores, llamados cachés, están estratégicamente ubicados cerca de los usuarios finales

para reducir la latencia y mejorar la velocidad de descarga. Cuando un usuario solicita contenido, la red CDN redirige la solicitud al servidor más cercano que almacena una copia en caché del contenido solicitado, en lugar de tener que buscarlo en un servidor central.

Las ventajas que ofrece esta arquitectura son:

1. Mejora la velocidad de carga: Al utilizar una red de servidores de caché distribuidos estratégicamente en diferentes ubicaciones geográficas, la CDN puede ofrecer una mejora significativa en la velocidad de carga de los videojuegos. Los servidores de caché almacenan copias del contenido cerca de los usuarios finales, lo que reduce la latencia y mejora la experiencia de juego al acelerar la entrega del contenido.

2. Mayor disponibilidad: La CDN aumenta la disponibilidad del contenido al distribuirlo en múltiples servidores de caché. Si un servidor experimenta problemas técnicos o se encuentra sobrecargado, los usuarios pueden acceder al contenido desde otro servidor cercano. Esto garantiza una mayor disponibilidad del juego y evita interrupciones para los usuarios.

3. Reducción de la carga en el servidor central: Al distribuir el contenido en servidores de caché ubicados estratégicamente, la CDN puede reducir significativamente la carga en el servidor central. Esto permite un mejor rendimiento y capacidad de respuesta del servidor central, ya que se encarga de tareas más críticas, como la gestión de sesiones de juego o el procesamiento de datos del usuario.

Los inconvenientes de aplicar esta arquitectura son:

1. Costo: La implementación de una CDN puede requerir una inversión significativa en infraestructura y recursos. Mantener y administrar una red de servidores de caché distribuidos puede ser costoso, especialmente para desarrolladores o empresas más pequeñas. El costo puede aumentar a medida que se agregan más ubicaciones de servidores de caché para cubrir una mayor área geográfica.

2. Actualizaciones y sincronización: La distribución de contenido en caché puede presentar desafíos en cuanto a la actualización y sincronización del contenido. Cuando se realizan actualizaciones o cambios en el juego, es necesario asegurarse de que todos los servidores de caché estén actualizados y sincronizados adecuadamente. De lo contrario, los usuarios podrían enfrentar problemas de inconsistencia en el contenido o experimentar versiones antiguas del juego.

La arquitectura de distribución de contenido en caché (CDN) se basa en la colocación estratégica de copias de contenido en servidores distribuidos globalmente, mejorando la velocidad de descarga y realizando una distribución más eficiente pudiendo gestionar un mayor volumen de carga. Sin embargo, llevar a cabo esta arquitectura es más costosa y la actualización de contenido en caché necesita emplear más tiempo. (Vakali & Pallis, 2003) (Johnson et al., 2001)

Red de pares (*Peer-to-Peer*)

La arquitectura *Peer-to-Peer* (P2P) se basa en una red descentralizada en la que los usuarios comparten y descargan el contenido directamente entre sí, sin depender de un servidor central. De esta forma se crea una comunidad de pares donde cada usuario puede actuar como cliente, descargando contenido, y como servidor (compartiendo contenido).

Las principales ventajas de utilizar la arquitectura P2P son:

1. Distribución eficiente de la carga: En esta arquitectura, todo el volumen de descarga se distribuye entre los propios usuarios lo que reduce la carga en los servidores centrales. Esto provoca una descarga más rápida y eficiente cuando una gran cantidad de usuarios descargan el mismo contenido.

2. Escalabilidad: A medida que incrementa el número de usuarios que comparten contenido dentro de la red P2P, aumentan las fuentes de carga disponibles. Esto facilita la distribución del contenido a un mayor número de usuarios sin sobrecargar la red o los servidores centrales.

3. Resistencia de la red: La naturaleza descentralizada de la red P2P permite una mayor resistencia. Si un nodo se desconecta o deja la red, los demás nodos pueden continuar compartiendo y descargando contenido entre sí, lo que garantiza una mayor disponibilidad y estabilidad en la distribución de videojuegos.

Los mayores inconvenientes que trae consigo esta arquitectura son:

1. Dependencia de la disponibilidad de otros usuarios: En la arquitectura P2P, la velocidad de descarga puede verse afectada si hay una escasez de usuarios que compartan el contenido deseado. Si no hay suficientes fuentes disponibles, la velocidad de descarga puede disminuir.

2. Variabilidad en la calidad de descarga: La calidad de la descarga puede variar dependiendo de la conexión y el rendimiento de los pares con los que se está descargando. Si un usuario tiene una conexión lenta o inestable, esto puede afectar negativamente la velocidad y calidad de la descarga. Además, algunos usuarios pueden tener limitaciones en su capacidad de carga, lo que puede limitar la disponibilidad y velocidad de descarga del contenido.

La arquitectura P2P presenta una distribución eficiente del volumen de la carga, escalabilidad a medida que van aumentando el número de usuarios y resistencia de la red al no depender de nodos individuales. Sin embargo, es totalmente dependiente del número de usuarios activos y la calidad de la descarga depende en gran medida de la calidad de conexión y rendimiento de los pares. (Milojicic et al., 2003)

Casos de estudio

Se realizaron análisis y evaluaciones de tres aplicaciones de distribución de videojuegos *Steam*, *Itch.io* y *minijuegos.com* con el objetivo de estudiar sus características y determinar el modo en el que se desea incorporar en la aplicación final.

Menú principal y selección de juegos

Se evaluaron las opciones disponibles para implementar un menú intuitivo y atractivo que permita a los usuarios seleccionar y acceder a los juegos de manera sencilla

Steam

La página de inicio no presenta un menú principal con una lista de juegos disponibles, ya que su enfoque principal es la venta de juegos. En su lugar, la página de inicio se compone de información sobre juegos destacados de la tienda. Los apartados principales de esta pantalla son los siguientes:

- "*Destacados y recomendados*": Esta sección a modo de carrusel ofrece recomendaciones personalizadas para cada jugador, basadas en su historial de usuario y las ventas de la plataforma. Aquí se destacan los juegos más relevantes y apropiados para cada jugador en particular.
- "*Ofertas especiales*": En este apartado se muestran juegos que tienen descuentos durante un período de tiempo específico. Estas ofertas permiten a los usuarios aprovechar descuentos especiales en juegos seleccionados.
- "*Explorar por categoría*": Esta sección presenta las principales categorías de juegos disponibles en la plataforma. Los usuarios pueden explorar y navegar por las categorías para encontrar juegos que se ajusten a sus preferencias.

Estos elementos en la página de inicio están diseñados para brindar a los usuarios una visión general de los juegos destacados, ofrecer recomendaciones personalizadas y permitirles explorar diferentes categorías. De esta manera, se busca facilitar la experiencia de compra y descubrimiento de juegos en la plataforma.

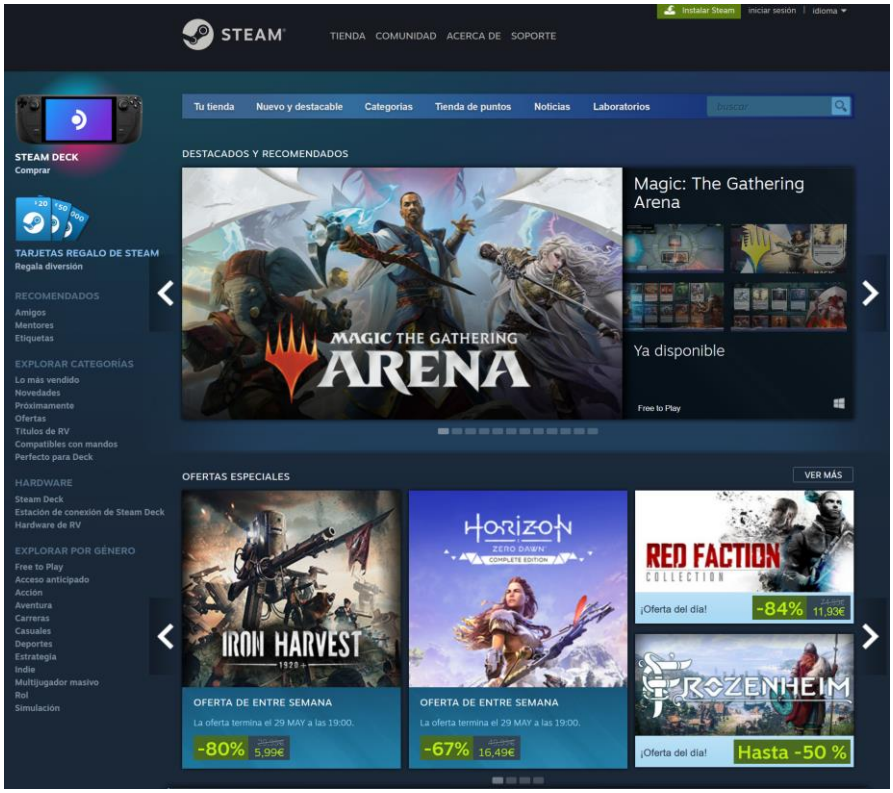


Ilustración 15. Página principal de Steam

Minijuegos.com

En la página de inicio, se encuentra una cabecera que muestra una selección de juegos destacados. Justo debajo, se encuentra un apartado que muestra los últimos juegos jugados por el usuario. A partir de ahí, la página continúa con otros apartados que agrupan los juegos por categoría, como por ejemplo *"juegos nuevos"*, *"juegos multijugador"*, *"juegos de acción"*, etc.

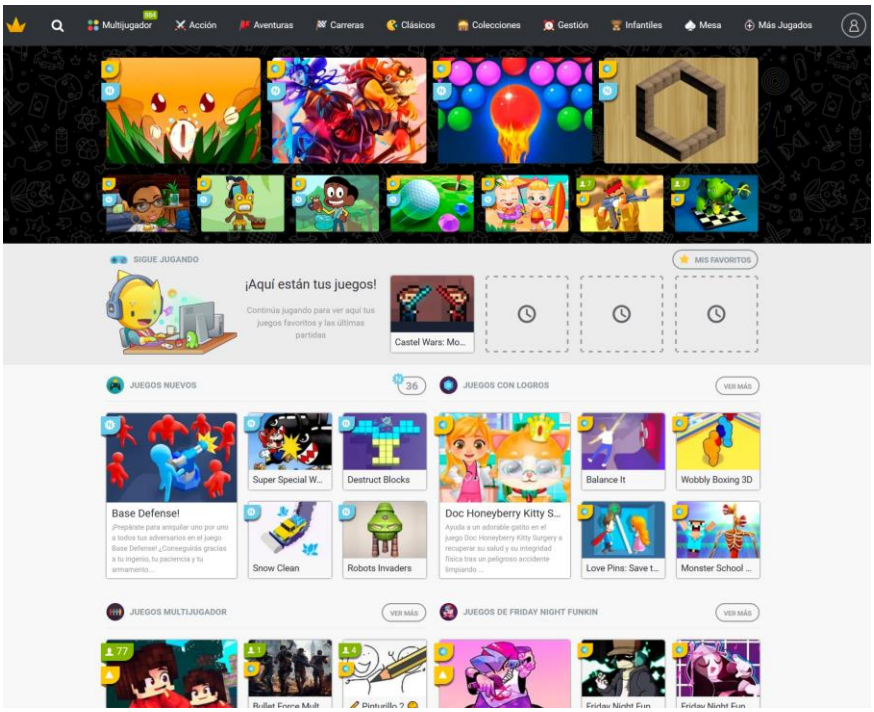


Ilustración 16. Página principal de Minijuegos.com

Itch.io

La página de inicio presenta una disposición similar a la de minijuegos, con una cabecera con un juego destacado y sucesivos apartados que agrupan juegos por categorías *"Latest Featured Games"*, *"Fresh Games"*, etc.

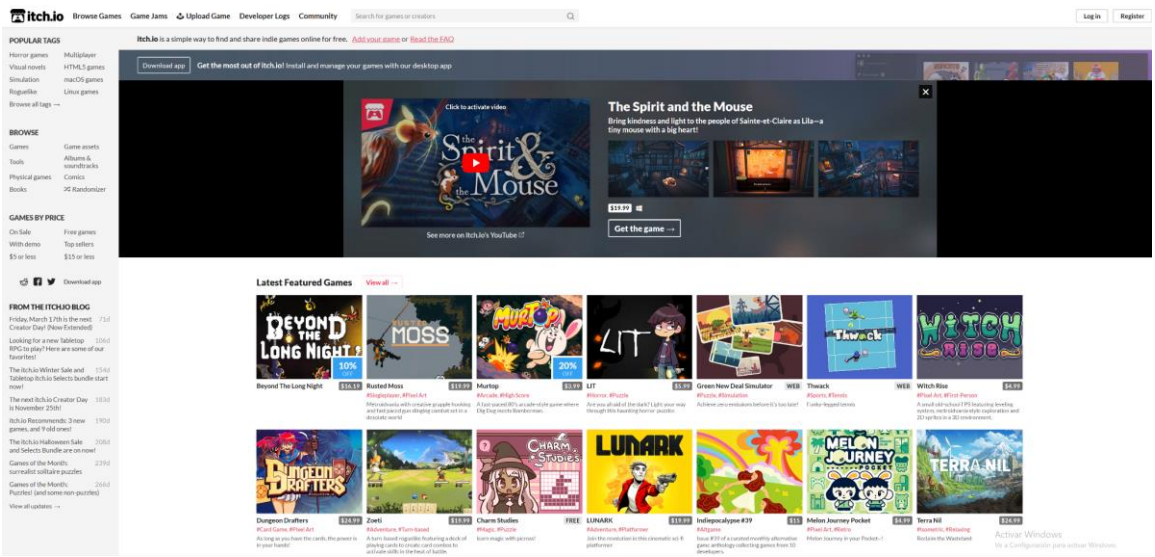


Ilustración 17. página principal de Itch.io

Información del juego y del jugador

Se consideraron las funcionalidades necesarias para proporcionar información detallada sobre cada juego.

Steam

Desde la plataforma de *Steam* se presenta la pantalla de información del juego dividiéndola en 4 subapartados.

El primero de ellos son videos, imágenes y texto promocional junto con un breve resultado de las valoraciones de los usuarios de la plataforma, así como datos del desarrollador y fecha de lanzamiento del juego.

Ilustración 18. Pantalla de información de juego de Steam – Información promocional

A continuación, muestra 2 subapartados uno principal con las diferentes opciones de compra del juego y otro lateral con diferentes especificaciones del juego como las diferentes características de *Steam* que contempla el juego, los idiomas en los que se permite jugar o la clasificación de *PEGI* a la que pertenece.

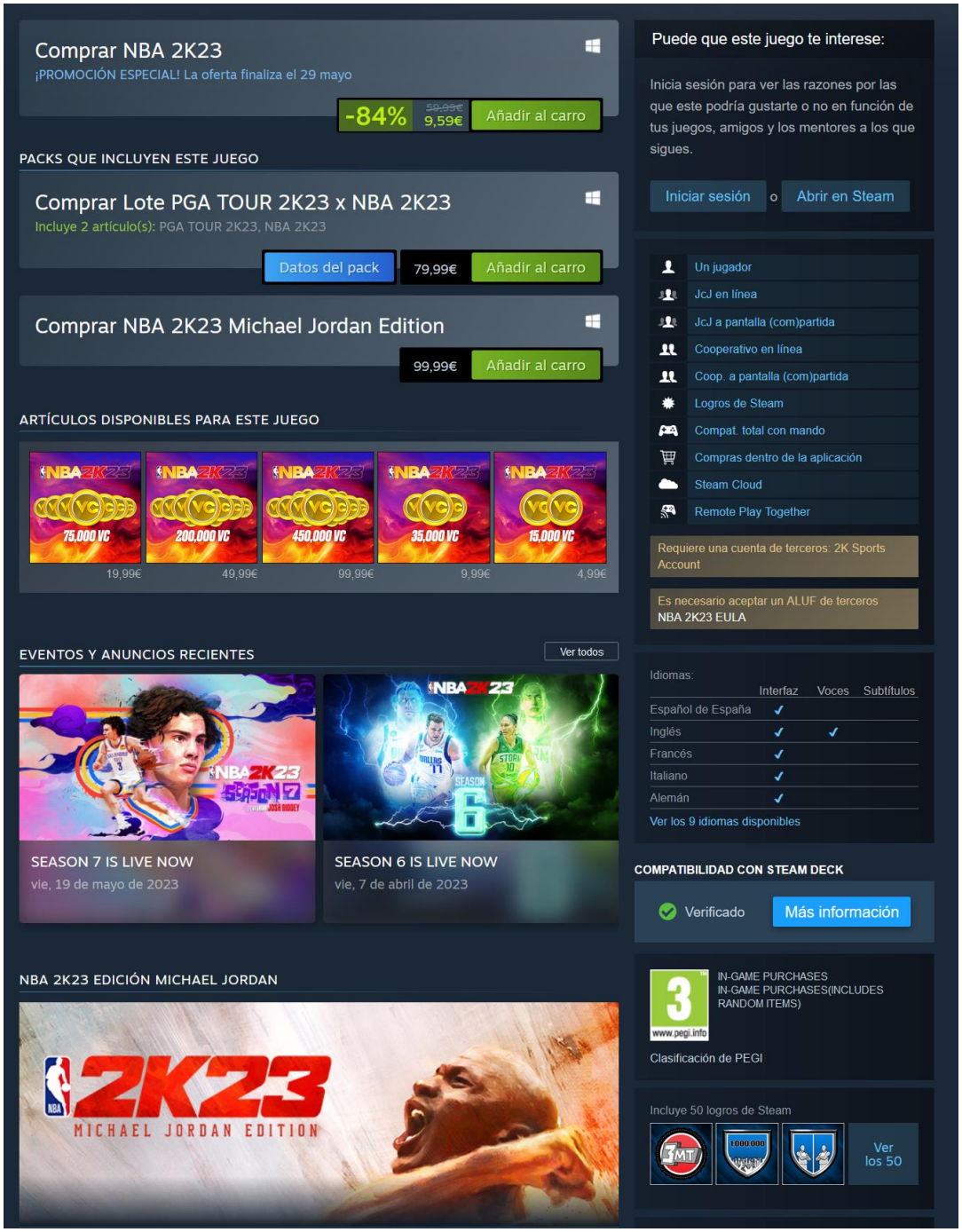
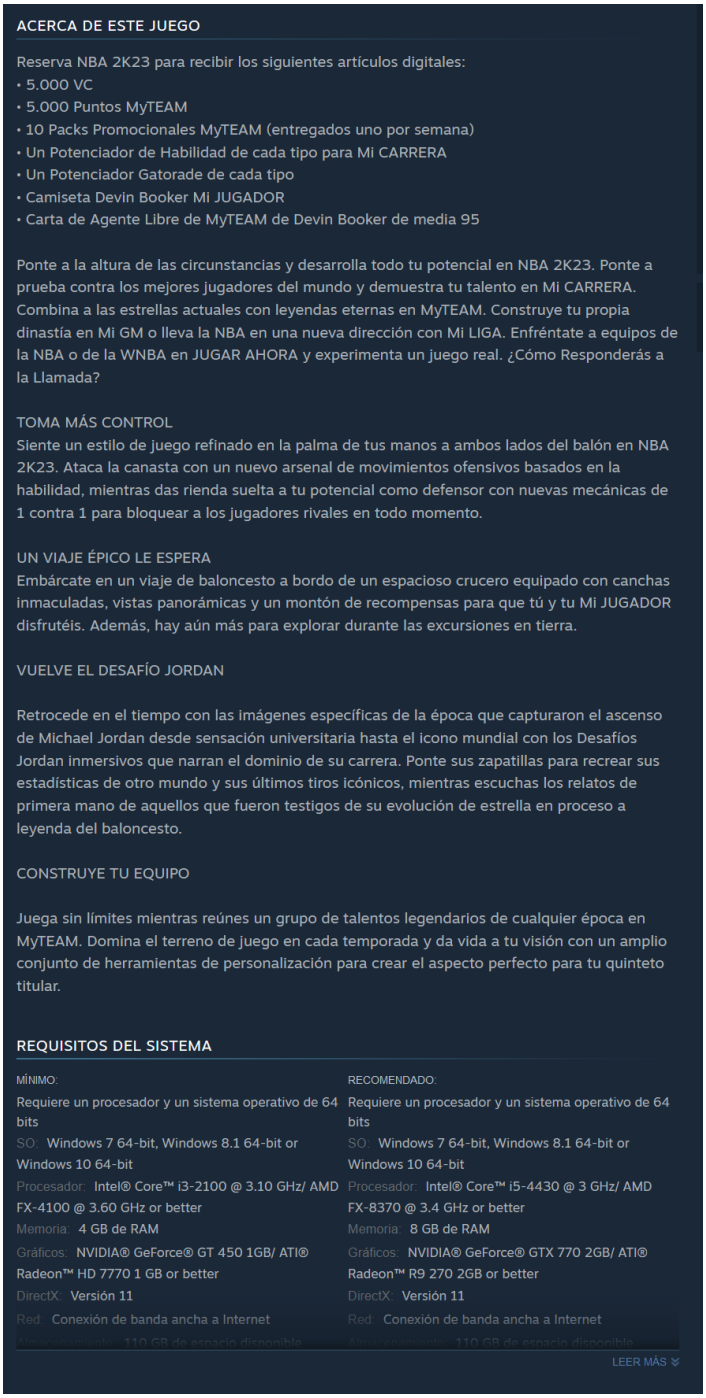


Ilustración 19. Pantalla de información de juego de Steam - Opciones de compra

Posteriormente incluye una descripción del juego, juntos con los requisitos mínimos para poder ejecutar este juego en el equipo:



ACERCA DE ESTE JUEGO

Reserva NBA 2K23 para recibir los siguientes artículos digitales:
• 5.000 VC
• 5.000 Puntos MyTEAM
• 10 Packs Promocionales MyTEAM (entregados uno por semana)
• Un Potenciador de Habilidad de cada tipo para Mi CARRERA
• Un Potenciador Gatorade de cada tipo
• Camiseta Devin Booker Mi JUGADOR
• Carta de Agente Libre de MyTEAM de Devin Booker de media 95

Ponte a la altura de las circunstancias y desarrolla todo tu potencial en NBA 2K23. Ponte a prueba contra los mejores jugadores del mundo y demuestra tu talento en Mi CARRERA. Combina a las estrellas actuales con leyendas eternas en MyTEAM. Construye tu propia dinastía en Mi GM o lleva la NBA en una nueva dirección con Mi LIGA. Enfréntate a equipos de la NBA o de la WNBA en JUGAR AHORA y experimenta un juego real. ¿Cómo Responderás a la Llamada?

TOMA MÁS CONTROL
Siente un estilo de juego refinado en la palma de tus manos a ambos lados del balón en NBA 2K23. Ataca la canasta con un nuevo arsenal de movimientos ofensivos basados en la habilidad, mientras das rienda suelta a tu potencial como defensor con nuevas mecánicas de 1 contra 1 para bloquear a los jugadores rivales en todo momento.

UN VIAJE ÉPICO LE ESPERA
Embárcate en un viaje de baloncesto a bordo de un espacioso crucero equipado con canchas inmaculadas, vistas panorámicas y un montón de recompensas para que tú y tu Mi JUGADOR disfrutéis. Además, hay aún más para explorar durante las excursiones en tierra.

VUELVE EL DESAFÍO JORDAN

Retrocede en el tiempo con las imágenes específicas de la época que capturaron el ascenso de Michael Jordan desde sensación universitaria hasta el icono mundial con los Desafíos Jordan inmersivos que narran el dominio de su carrera. Ponte sus zapatillas para recrear sus estadísticas de otro mundo y sus últimos tiros icónicos, mientras escuchas los relatos de primera mano de aquellos que fueron testigos de su evolución de estrella en proceso a leyenda del baloncesto.

CONSTRUYE TU EQUIPO

Juega sin límites mientras reúnes un grupo de talentos legendarios de cualquier época en MyTEAM. Domina el terreno de juego en cada temporada y da vida a tu visión con un amplio conjunto de herramientas de personalización para crear el aspecto perfecto para tu quinteto titular.

REQUISITOS DEL SISTEMA

MÍNIMO:	RECOMENDADO:
Requiere un procesador y un sistema operativo de 64 bits	Requiere un procesador y un sistema operativo de 64 bits
SO: Windows 7 64-bit, Windows 8.1 64-bit or Windows 10 64-bit	SO: Windows 7 64-bit, Windows 8.1 64-bit or Windows 10 64-bit
Procesador: Intel® Core™ i3-2100 @ 3.10 GHz/ AMD FX-4100 @ 3.60 GHz or better	Procesador: Intel® Core™ i5-4430 @ 3 GHz/ AMD FX-8370 @ 3.4 GHz or better
Memoria: 4 GB de RAM	Memoria: 8 GB de RAM
Gráficos: NVIDIA® GeForce® GT 450 1GB/ ATI® Radeon™ HD 7770 1 GB or better	Gráficos: NVIDIA® GeForce® GTX 770 2GB/ ATI® Radeon™ R9 270 2GB or better
DirectX: Versión 11	DirectX: Versión 11
Red: Conexión de banda ancha a Internet	Red: Conexión de banda ancha a Internet
Almacenamiento: 110 GB de espacio disponible	Almacenamiento: 110 GB de espacio disponible

LEER MÁS

Ilustración 20. Pantalla de información de juego de Steam - Detalles del juego y requisitos del sistema

Y por último comentarios de los usuarios acerca del juego junto con su voto, separando las reseñas por su utilidad y fecha de publicación:

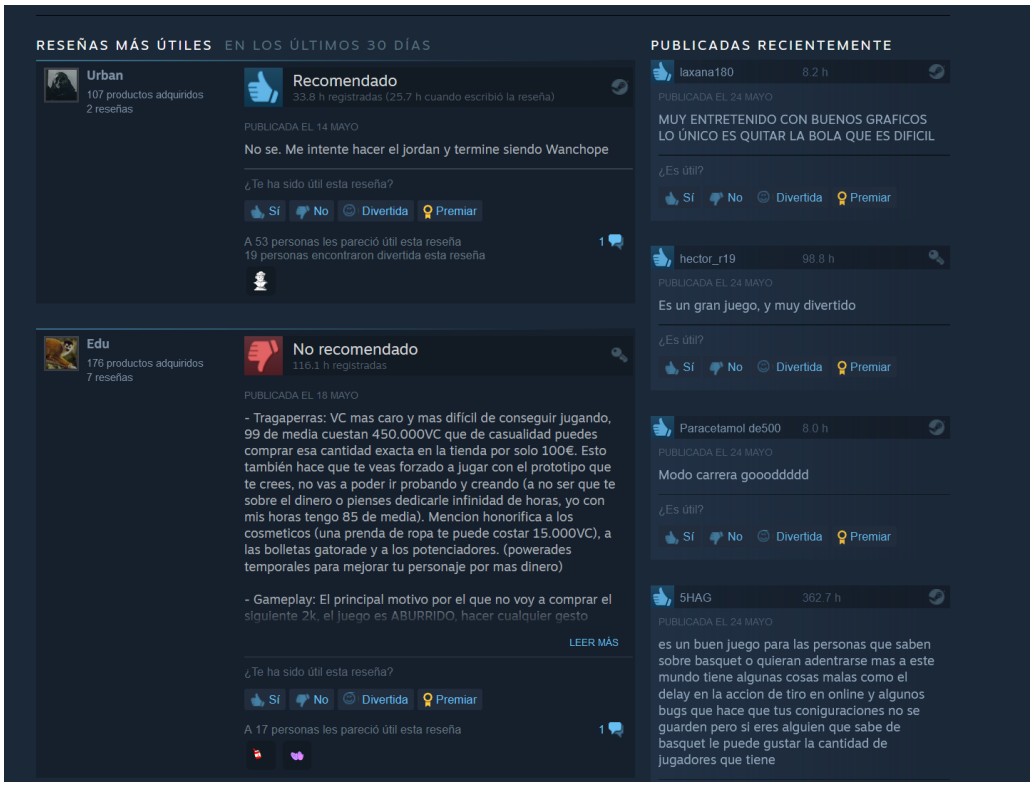


Ilustración 21. Pantalla de información de juego de Steam - Reseñas

Minijuegos.com

Dentro de la página de información del juego contiene una cabecera con el icono del juego, su nombre, las categorías a las que pertenece, el número de partidas jugadas al juego y en el lado izquierdo de la cabecera el *rating* del juego por los usuarios de la plataforma. Justo debajo de esta cabecera se encuentra la ventana del juego.

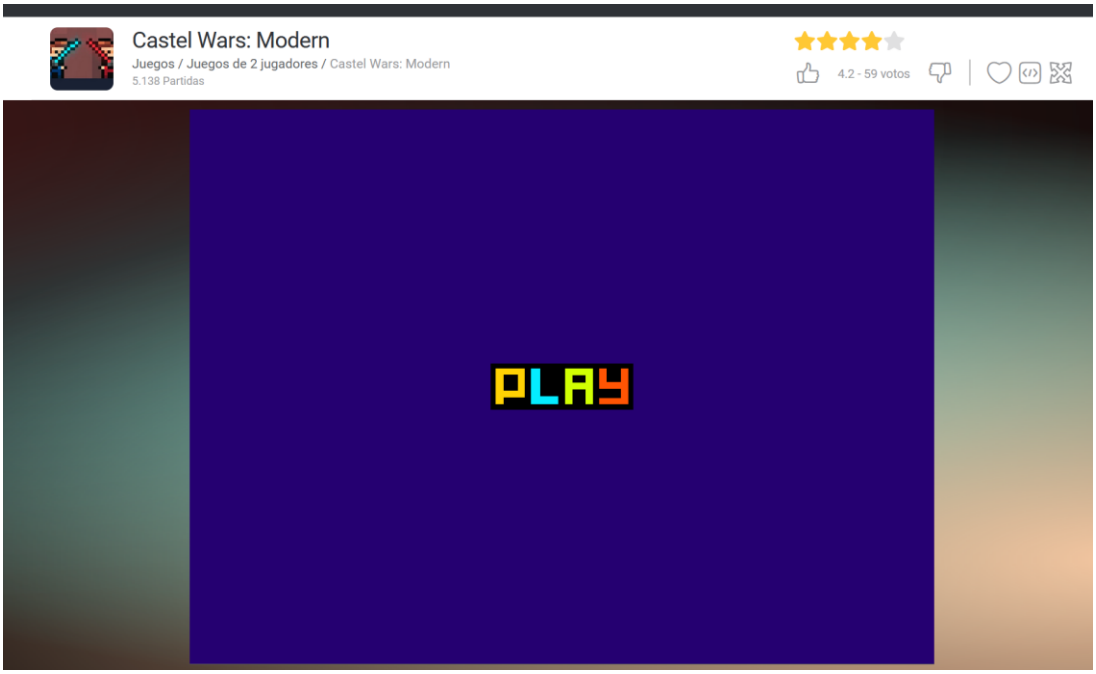


Ilustración 22. Pantalla de información de juego de Minijuegos.com – Juego

Debajo de la ventana del juego se encuentra el apartado explicativo de como jugar al juego que contiene un breve texto acerca del juego, información de los controles y videos con *gameplays* del juego.

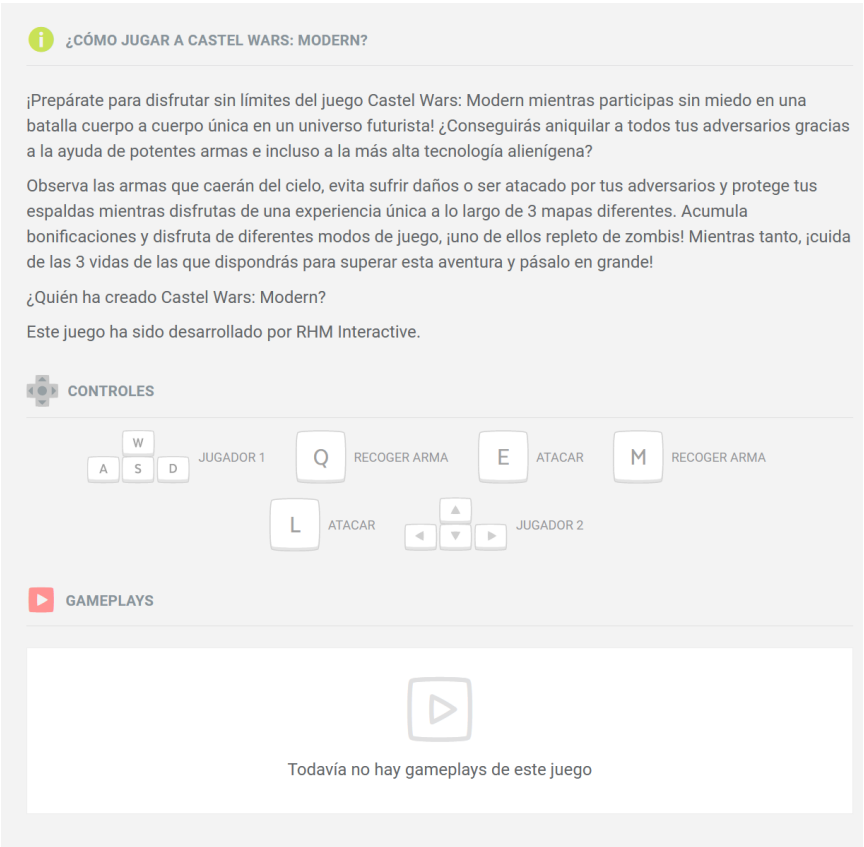


Ilustración 23. Pantalla de información de juego de Minijuegos.com - Controles

Por último, se encuentra la sección de comentarios en la que se permite ordenarlos por comentarios destacados o fecha de envío.

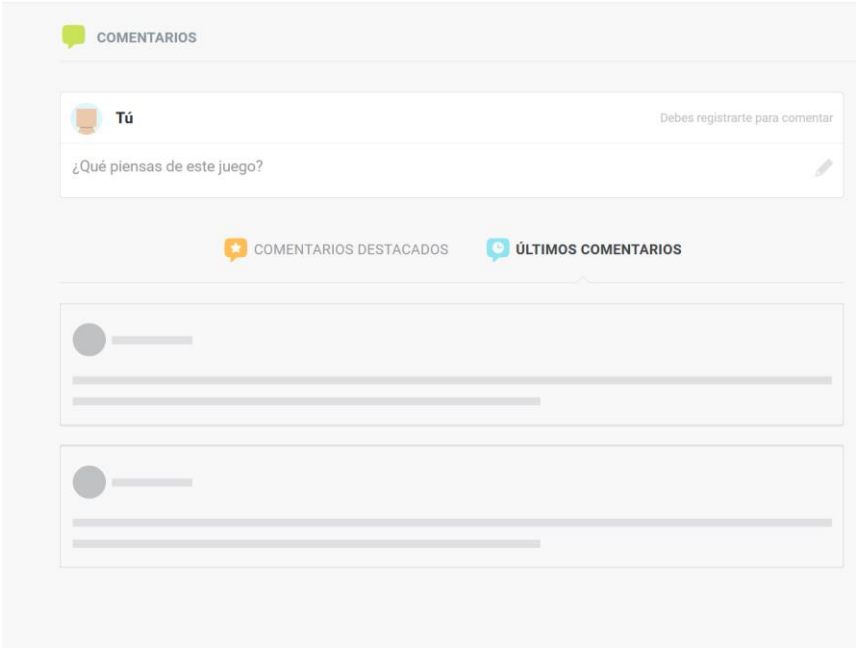


Ilustración 24. Pantalla de información de juego de Minijuegos.com - Reseñas

Itch.io

En *itch.io*, las páginas de cada juego son personalizables y tienen características estéticas distintas. Dependiendo del tipo de juego, ya sea jugable en la propia web o que requiera ser descargado, la cabecera de la página será diferente.

En el caso de los juegos que se pueden jugar directamente en la web, es posible incluir una cabecera con una imagen personalizada. Esta imagen puede servir como un elemento visual atractivo para presentar el juego y a continuación se coloca el juego, el cual se iniciará al pulsar el botón de *"run game"*.

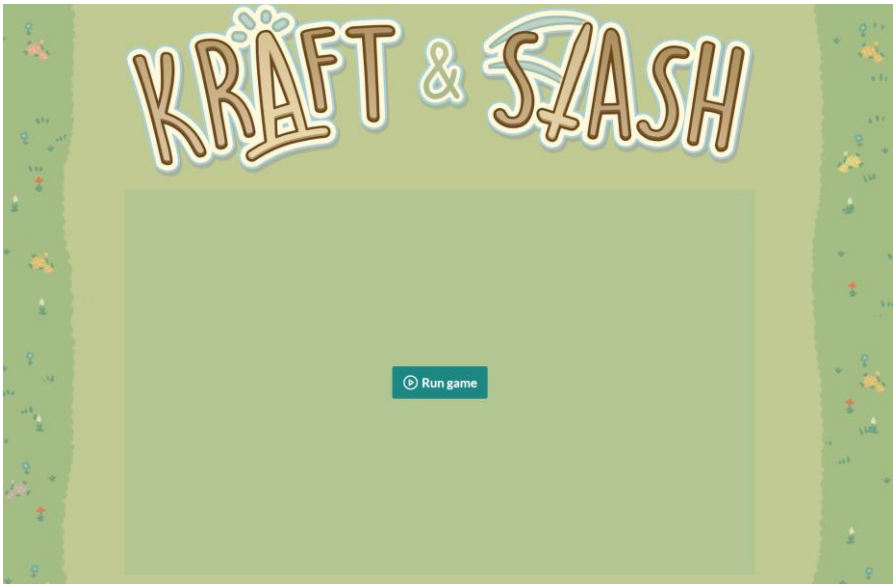


Ilustración 25. Pantalla de información de juego de Itch.io – Juego web

Para la versión descargable se dispone igualmente de la posibilidad de incluir una imagen personalizada en la cabecera y debajo se dispondrá un botón para descargar o comprar el juego, según sea de pago o gratuito.

Ilustración 26. Pantalla de información de juego de Itch.io - Juego descargable

En el siguiente apartado *itch.io* permite al desarrollador introducir la información que considere sobre el juego pudiendo incorporar imágenes y videos

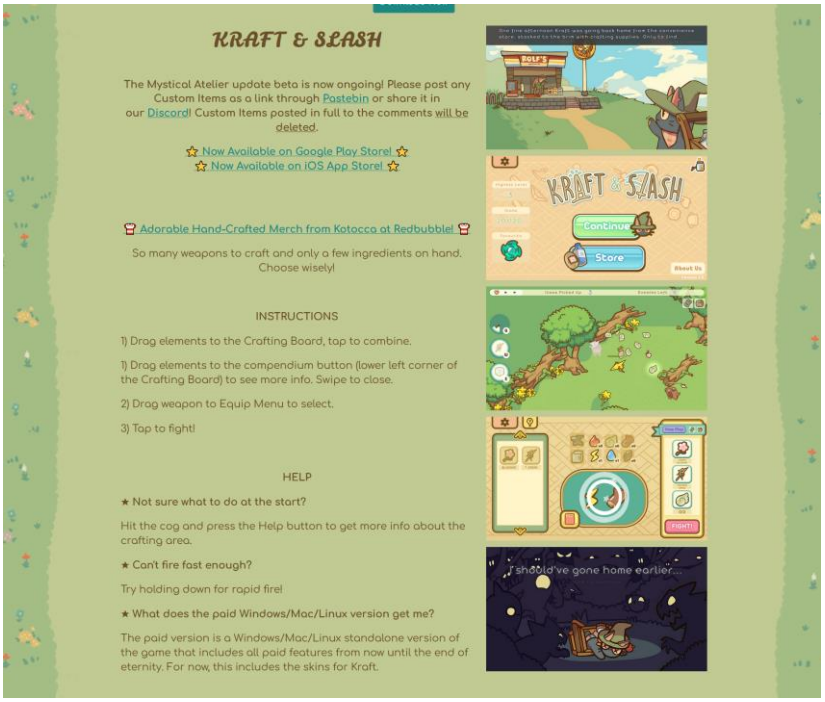


Ilustración 27. Pantalla de información de juego de Itch.io - Información del juego

Y al final de la página dispone del tablón de comentarios, para los usuarios de la plataforma:

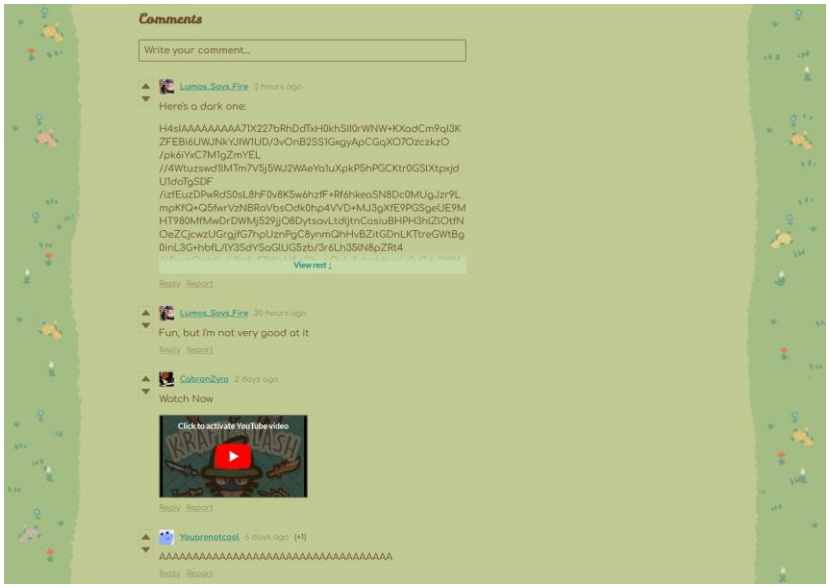


Ilustración 28. Pantalla de información de juego de Itch.io – Reseñas

Gestión de amistades

Se analizaron las características que permiten a los usuarios conectarse y establecer relaciones de amistad dentro de la plataforma

Steam

Steam ofrece varias funciones relacionadas con las amistades entre usuarios:

- Agregar amigos: Los usuarios pueden agregar a otros usuarios como amigos en *Steam*. Esto permite establecer conexiones y mantener un seguimiento de las actividades de los amigos en la plataforma.
- Chat y mensajería: *Steam* proporciona un sistema de chat y mensajería integrado que permite a los amigos comunicarse entre sí. Los usuarios pueden chatear individualmente o en grupos.
- Invitar a jugar: Los usuarios pueden invitar a sus amigos a unirse a ellos en partidas multijugador. *Steam* proporciona la funcionalidad de invitar a amigos a través de notificaciones o invitaciones en el juego.

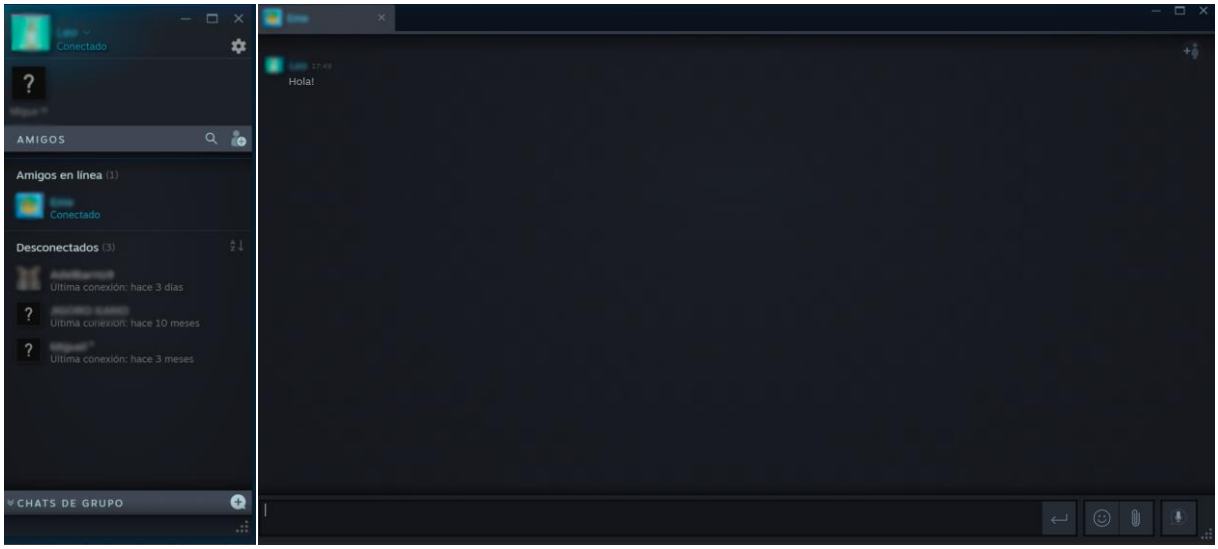


Ilustración 29. Apartado social de Steam

Minijuegos.com

Minijuegos.com al igual que Steam tiene las funciones de agregar amigos y de chatear con ellos, pero no dispone de opciones para invitar a jugar. *Minijuegos.com* dispone de un menú lateral de gestión social y ventanas emergentes para búsqueda de amigos y chat:

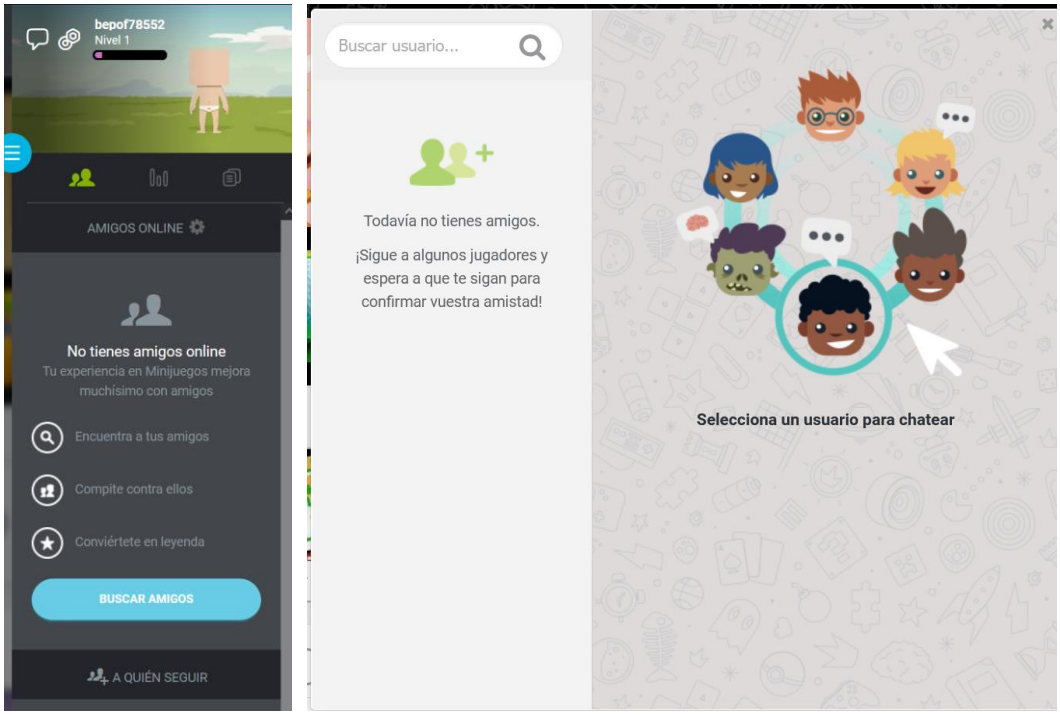


Ilustración 30. Apartado social de Minijuegos.com

Itch.io
No dispone de funciones de amistad entre usuarios.

Información del usuario

Se examina la implementación de perfiles de usuario, donde los jugadores pueden personalizar su información, establecer preferencias y ver su historial de juegos y logros.

Steam

Dentro del perfil de usuario en *Steam*, se encuentra una sección superior donde se muestra información personal del usuario, como la imagen de perfil, el nombre, la ubicación geográfica y el nivel de la cuenta.

Justo debajo de esta sección, se encuentra un área dedicada a la actividad del usuario. En la parte principal, se muestra información sobre la actividad más reciente, como los últimos juegos utilizados y el total de horas dedicadas. En el lateral derecho, se muestran las insignias obtenidas por el usuario, así como datos relacionados con el uso de *Steam*, como el número de juegos, videos y reseñas, además de información sobre los amigos del usuario.

Por último, se encuentra una sección reservada para comentarios de otros usuarios a ese perfil.

Este perfil de usuario puede ser modificado a partir de objetos obtenidos al usar la aplicación, pudiendo modificar distintos aspectos como los fondos, los marcos o la imagen de perfil.

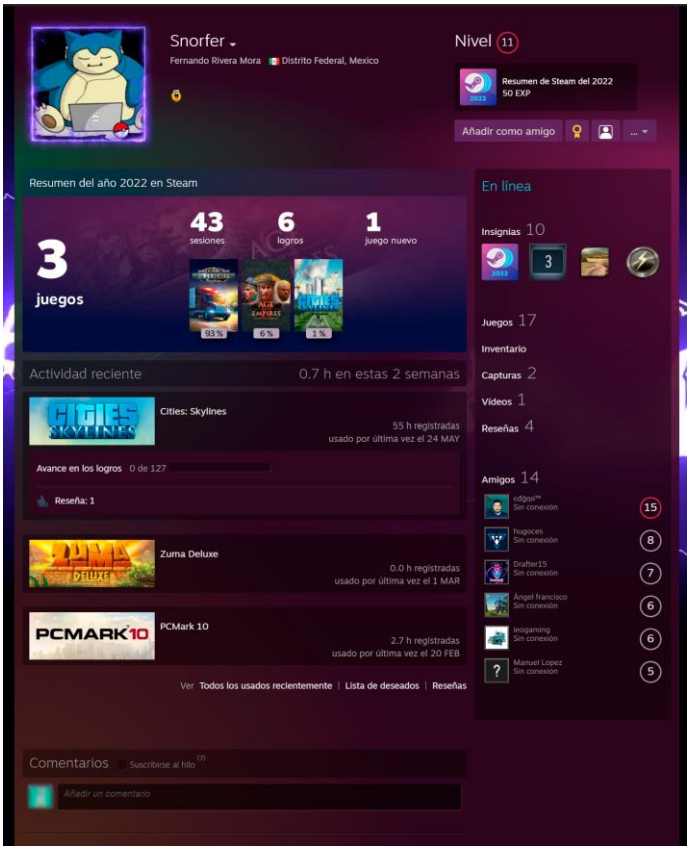


Ilustración 31. Pantalla de información del usuario de Steam

Minijuegos.com

En el perfil de usuario se tiene acceso a la ventana de edición del avatar y al registro de la actividad, trofeos, logros, jugadores seguidos y juegos favoritos. También se dispone de una barra de progreso con tu nivel actual y tu posición del ranking global de jugadores.

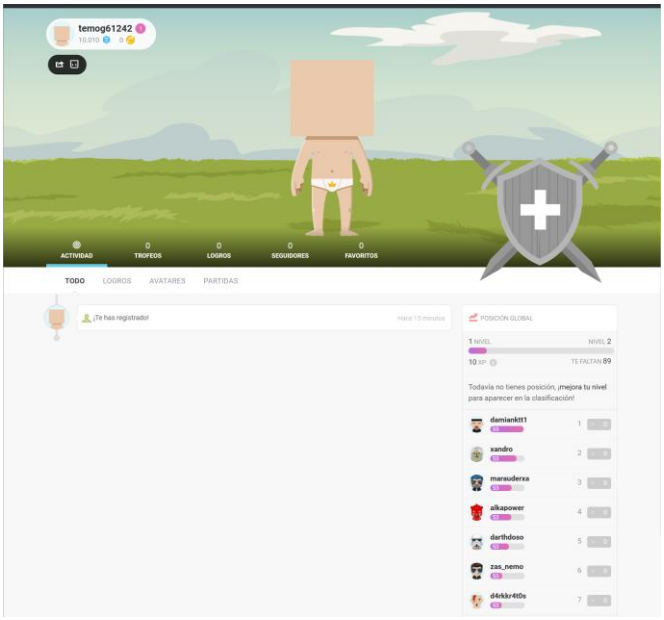


Ilustración 32. Pantalla de información del usuario de Minijuegos.com

Itch.io

Creado para una comunidad de desarrolladores, las páginas de perfil de usuario son páginas personalizables en la que se incluyen los juegos aportados a la plataforma.

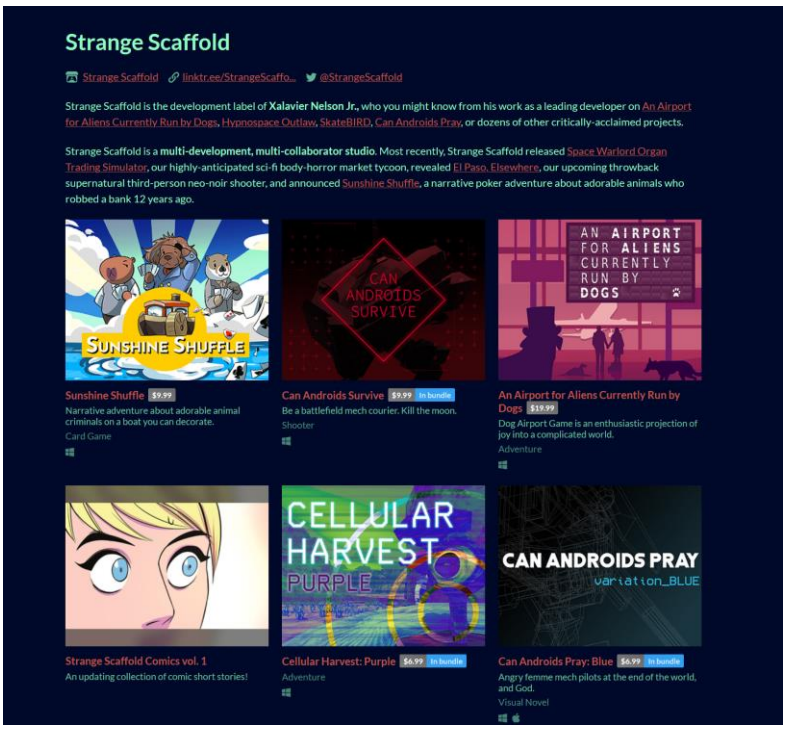


Ilustración 33. Pantalla de información del usuario de Itch.io

Conclusiones

Las conclusiones obtenidas tras el análisis realizado sobre las aplicaciones *Steam*, *Itch.io* y *minijuegos.com* son las siguientes:

En el menú principal al igual que en *Steam*, el primer elemento que se encuentra es un carrusel con una serie de juegos destacados y a continuación, basándose en el diseño de *itch.io* y *minijuegos*, deben disponerse los juegos de manera horizontal y separados por categorías.

Para las pantallas de información de los juegos se combinarán elementos de las tres aplicaciones. Un primer elemento a modo de ficha como en *Steam*, con descripciones e información del juego, pero en él sé que muestre la portada del juego como realiza *Itch.io*. Un sistema de valoración por votos, pero con tres niveles, información sobre el tiempo de uso de los juegos por parte de los usuarios y un sistema de reseñas por usuarios en la parte final de la página.

Adoptar los métodos de menú lateral para las acciones sociales de la aplicación y las ventanas emergentes para realizar acciones determinadas.

Estudio de alternativas

La primera elección que se tuvo que tomar a la hora de abordar el proyecto fue la de seleccionar un *framework* en el que desarrollar la aplicación. Para seleccionar un *framework* de desarrollo para aplicaciones web, se tuvieron que considerar ciertos requisitos clave. Uno de ellos era que la curva de aprendizaje no fuera demasiado pronunciada, lo que significaba que se buscó un *framework* que fuera fácil de aprender y utilizar para los desarrolladores. Además, era importante que contara con una comunidad activa y sólida, ya que esto aseguraba que se tuvieran recursos y soporte disponibles en caso de enfrentar desafíos durante el desarrollo del proyecto.

En base a estos criterios, se realizó una comparativa entre tres frameworks: *Ruby on Rails*, *.NET* y *Django*.

Aspecto	***Ruby on Rails***	***.NET***	***Django***
Logo			

Lenguaje	*Ruby*	*C#, F#*	*Python*
Tipo de Framework	*Back-end* y *Front-end*	*Back-end* y *Front-end*	*Back-end* y *Front-end*
Patrón de Diseño	MVC (Modelo-Vista-Controlador)	MVC y MVVM	MTV (Modelo-Plantilla-Vista)
Enrutamiento	Propio sistema de enrutamiento (*Rails Router*)	*ASP.NET Routing*	*URLconf* (sistema de enrutamiento de *Django*)
Base de Datos	Admite varios proveedores de bases de datos	Admite varios proveedores de bases de datos	Admite varios proveedores de bases de datos
ORM	*ActiveRecord*	*Entity Framework*	*Django ORM*
Escalabilidad	Altamente escalable	Altamente escalable	Altamente escalable
Comunidad	Gran comunidad de desarrollo y soporte	Gran comunidad de desarrollo y soporte	Gran comunidad de desarrollo y soporte
Curva de Aprendizaje	Puede requerir un tiempo de aprendizaje inicial. Desconocimiento total del lenguaje.	Algunos miembros del equipo los usan profesionalmente. Lenguaje conocido por ambos desarrolladores.	Puede requerir un tiempo de aprendizaje inicial. Lenguaje poco usado por ambos desarrolladores.

Tabla 1. Tabla comparativa de frameworks

Teniendo en cuenta estos aspectos, se evaluaron las características de cada uno de los *frameworks* y se decidió que *.NET* era el que mejor se ajustaba a las necesidades, sobre todo basándose en el criterio de curva de aprendizaje.

La siguiente decisión que se tomó fue seleccionar una biblioteca de comunicación en tiempo real para habilitar el envío de mensajes en tiempo real entre dispositivos móviles y PCs. Durante la búsqueda de opciones para estas funcionalidades, se evaluaron dos bibliotecas: *SignalR* y *Socket.IO*.

Aspecto	*SignalR*	*Socket.IO*
Logo		
Lenguaje	Principalmente enfocado en el ecosistema *.NET*	Principalmente enfocado en *JavaScript*
Plataforma	Compatible con *.NET*, *ASP.NET* y otras plataformas	Compatible con *Node.js* y navegadores web
Protocolos	Admite *WebSockets*, *Server-Sent Events* (SSE), *Long Polling* y más	Admite *WebSockets*, *Server-Sent Events* (SSE), *Long Polling* y más
Soporte de navegadores	Amplio soporte de navegadores modernos	Amplio soporte de navegadores modernos
Integración	Integración estrecha con el ecosistema .NET	Integración flexible con diferentes frameworks y lenguajes, principalmente con Node.js

Tabla 2. Tabla comparativa de bibliotecas de comunicación en tiempo real

Se optó por utilizar *SignalR* debido a que no se encontraron diferencias significativas entre los protocolos de comunicación soportados por ambas bibliotecas y los navegadores capaces de manejarlos por igual. Sin embargo, se consideró que la integración con *.NET* y el ecosistema de *Microsoft* era mucho más sencilla utilizando *SignalR*.

La última decisión antes de comenzar con el desarrollo fue la selección del proveedor de servicios en la nube para alojar la aplicación y aprovechar otros servicios disponibles, como bases de datos. Se realizaron comparaciones entre los servicios en la nube de *Amazon Web Services* (*Servidor Privado Virtual (VPS), Precios De Alojamiento Web-Amazon Lightsail-Amazon Web Services*), *Google Cloud Platform* (*Precios | App Engine*) y *Azure* (*Precios De App Service*).

Aspecto	AWS	GCP	Azure
Logo			
Propiedad	*Amazon.com*	*Google LLC*	*Microsoft*

Lanzamiento	2006	2008	2010
Popularidad	Ampliamente utilizado y popular en la industria	Creciendo rápidamente en popularidad	Ampliamente utilizado y popular en la industria
Escalabilidad	Escalabilidad vertical y horizontal	Escalabilidad vertical y horizontal	Escalabilidad vertical y horizontal
Soporte de .NET	Si	Si	Si
Uso	Ampliamente utilizado en diversas industrias	Ampliamente utilizado en diversas industrias	Ampliamente utilizado en diversas industrias

Tabla 3. Tabla comparativa de proveedores de servicios en la nube

Otro aspecto crítico que se debe considerar en esta comparativa, y que no se ha incluido en la tabla anterior, es el factor del precio. Cada una de las plataformas ofrece una amplia flexibilidad y diversas opciones en función de la capacidad de procesamiento, el volumen de datos y la cantidad de accesos. Para realizar la comparativa, únicamente se han comparado las versiones más básicas ofertadas por las distintas plataformas

App Service Plan F1* de *Azure: Es gratuito, permite crear 10 aplicaciones web, móviles o interfaces de programación de aplicaciones (*Application programming interface* - API) y permite 1GB de espacio en disco. Tiene deshabilitadas opciones como son el dominio personalizado y la escalabilidad automática.

***Amazon Lightsail*:** 3 meses gratuitos, posteriormente 3.5 USD/mes por un Servidor Virtual *Linux/Unix* con 512 MB de memoria procesador de 1 núcleo disco SSD de 20GB y 1TB de Transferencia.

***Google Cloud App Engine F1*:** Límite de memoria de 256MB, Limite de CPU, 600MHz. Precios de almacenamiento Blob () y base de datos no incluidos.

Con toda esta información, se decidió que el plan ***F1* de *Azure*** se ajustaba mejor a las necesidades para reducir costos. Ofrecía un servicio con las especificaciones mínimas de manera gratuita, además, a través de la universidad, se tiene la posibilidad de solicitar una suscripción a *Azure* por $100 durante un año, disponiendo de multitud de servicios gratuitos disponibles como bases de datos, almacenamiento Blob, etc. Esta ventaja económica fue determinante para tomar la decisión de utilizar *Azure* como plataforma en la nube.

Capítulo 3

Diseño de la aplicación

Introducción

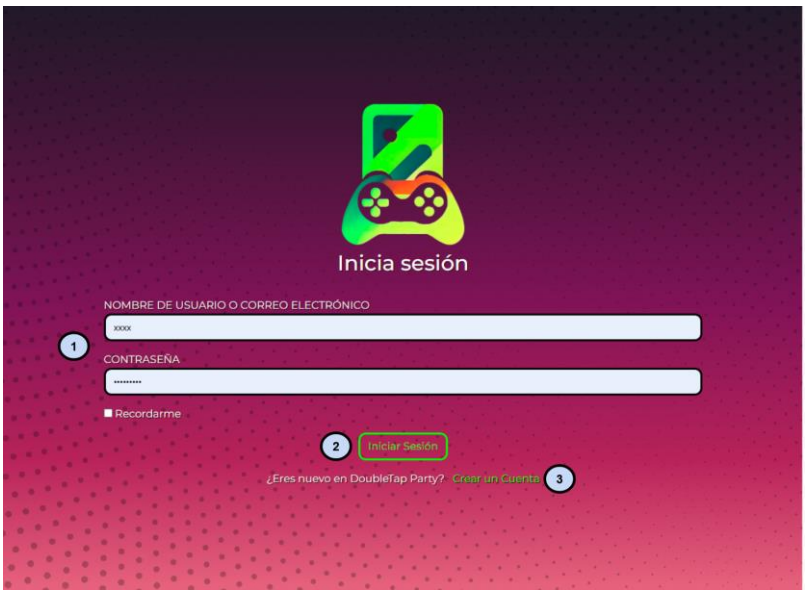
DoubleTapParty es una plataforma de juegos web que ofrece a los jugadores la posibilidad de utilizar sus teléfonos inteligentes como controladores mientras usan un ordenador como pantalla. En lugar de depender de los tradicionales mandos de consola, los jugadores acceden a los juegos y controlan la acción a través de sus teléfonos.

Para empezar a jugar en *DoubleTapParty*, los jugadores sólo necesitan contar con una conexión a Internet activa en su dispositivo móvil y en el ordenador. Al acceder al sitio web de *DoubleTapParty* y autenticarse tanto en el ordenador como en el teléfono, se establece una conexión entre ambos dispositivos, permitiendo que el teléfono controle las aplicaciones del ordenador. Además, los jugadores pueden unirse a salas con otros usuarios que sean amigos suyos en la plataforma. Una vez que los dispositivos están conectados, los jugadores pueden disfrutar de los juegos multijugador de la plataforma.

En este capítulo se detallará el diseño de la aplicación a través de las partes que lo conforman: aplicación de escritorio, aplicación móvil, páginas de administración, juegos y el juego multijugador *Carta contra la humanidad*.

Aplicación de escritorio

Al acceder a la aplicación desde el PC obliga a loguearte, para ello es necesario ingresar el nombre de usuario o bien el correo electrónico junto con su contraseña. Una vez introducidos usuario y contraseña, se chequeará que coinciden y que no existe ninguna otra sesión abierta para esa cuenta en un PC. En caso de que no se disponga de cuenta puede crear una a través del enlace situado en el texto “*Crear una cuenta*”.



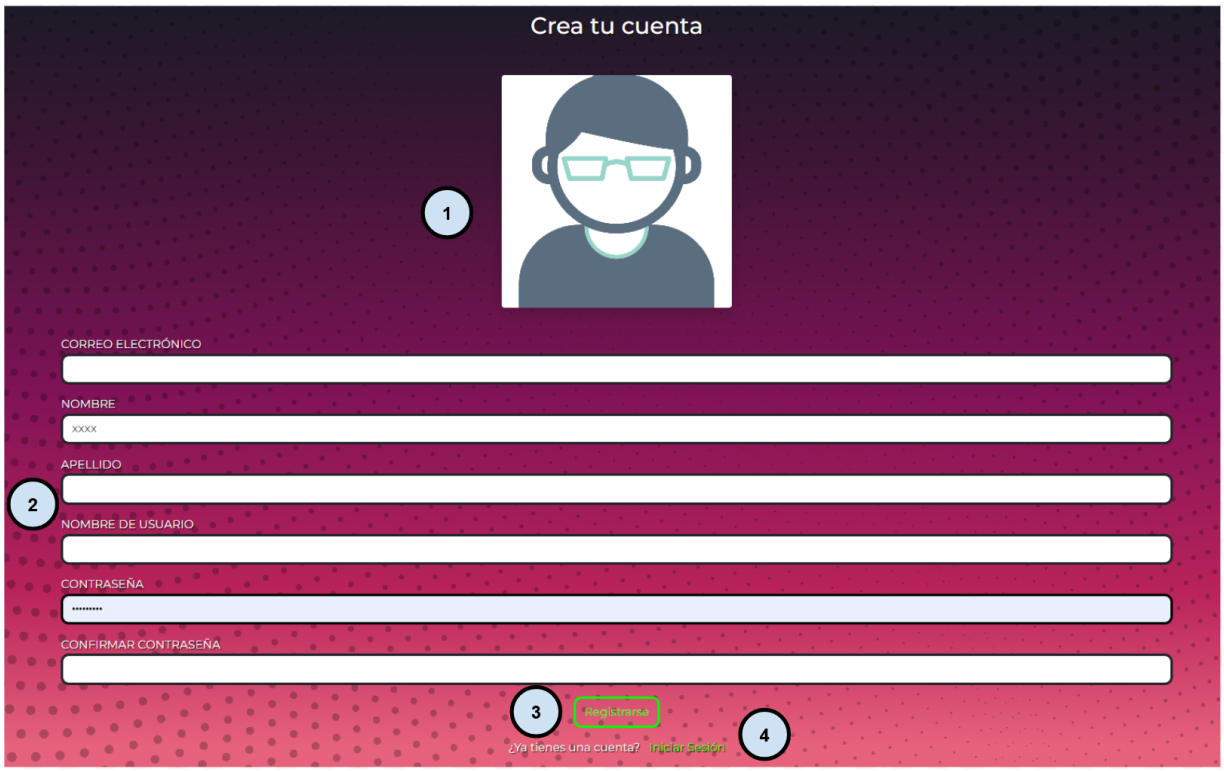
1) Entrada de credenciales
2) Botón de inicio de sesión
3) Enlace para crear nuevo usuario

Ilustración 34. Pantalla de inicio de sesión

En la página de registro se encuentran diferentes campos de texto para ingresar la información del usuario, así como la opción de adjuntar una imagen o foto de perfil. Al final de la página se encuentra un botón para completar el registro y un enlace para volver a la pantalla de inicio de sesión.

Una vez que se haya proporcionado toda la información requerida y se pulse el botón de "*Registrarse*". La aplicación verificará que tanto la dirección de correo electrónico como el nombre de usuario no estén ya registrados en la plataforma. Si alguno de estos datos ya existe, se informará del error.

Si todos los datos son válidos, la aplicación procederá a crear tu cuenta de usuario. En caso de que haya algún error adicional con los datos proporcionados, se notificará para corregirlos.



1) Imagen de usuario
2) Entradas de datos necesarios del usuario
3) Botón de Registrarse
4) Enlace para volver al Inicio de Sesión

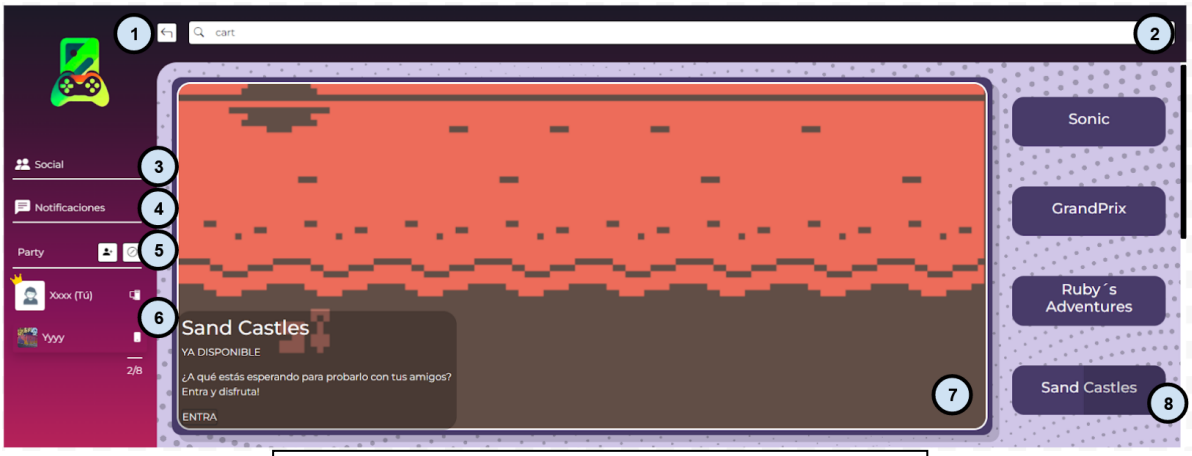
Ilustración 35. Pantalla de registro

Después de iniciar sesión, se accederá al menú principal de la aplicación. En este menú, se encuentra una barra de navegación en la parte superior, en el lateral izquierdo de la página se ubica la barra dedicada a las funciones sociales y por último en la parte central de la página se sitúa el menú principal de la aplicación que muestra los juegos disponibles. Las barras de navegación y de funciones sociales son compartidas por la mayoría de las vistas de la aplicación.

La barra de navegación cuenta con el botón de retroceso y la barra de búsqueda de juegos.

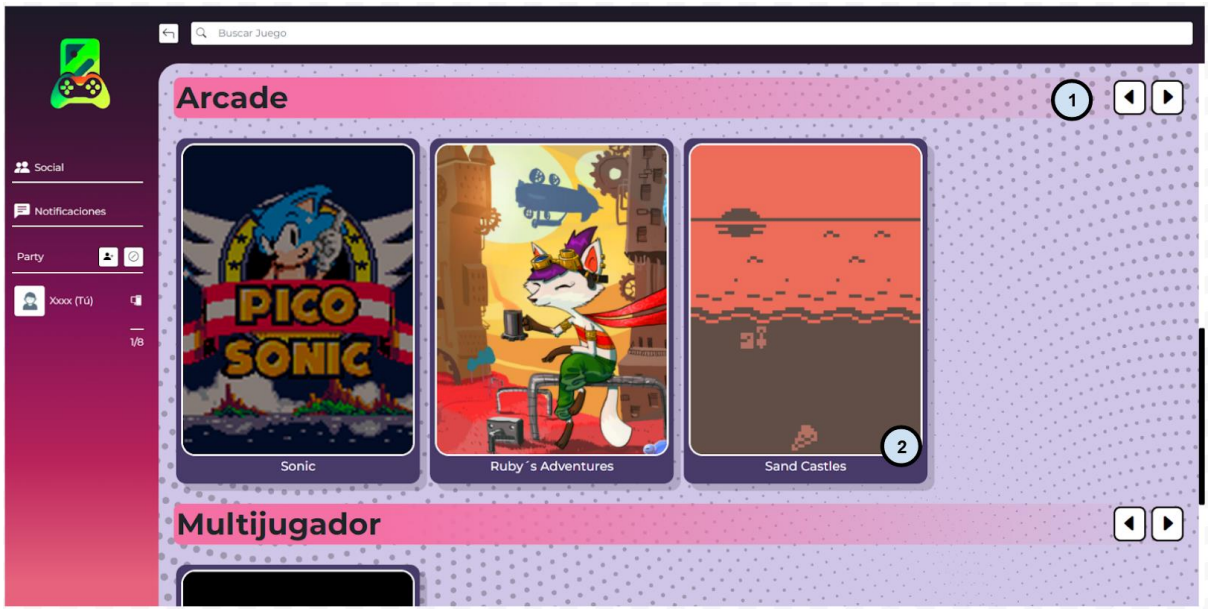
En la barra de funciones sociales está formada por los botones "*Social*", con el que buscar gente para añadir amigos, el botón "*Notificaciones*", con el que se informa si se tiene alguna invitación a grupo o solicitudes de amistad, por último, se encuentra el apartado "*Party*" con las opciones disponibles de la sala tales como invitar amigos, disolver sala y el listado de usuarios conectados a la sala.

El menú de juegos de la aplicación está formado por un carrusel con los juegos destacados y el listado de juegos agrupados por categorías.



1) Botón para volver a atrás
2) Entrada para buscar juegos por nombre
3) Botón para abrir el panel social
4) Botón para abrir las notificaciones del usuario
5) Botones para invitar a amigo al grupo o abandonarlo
6) Listado de usuarios en el grupo
7) Juego actual del carrusel
8) Listado de juegos destacados que se muestran en el carrusel

Ilustración 36. Pantalla principal – Carrusel



1) Categoría
2) Listado de juegos con la categoría

Ilustración 37. Pantalla principal – Juegos por categorías

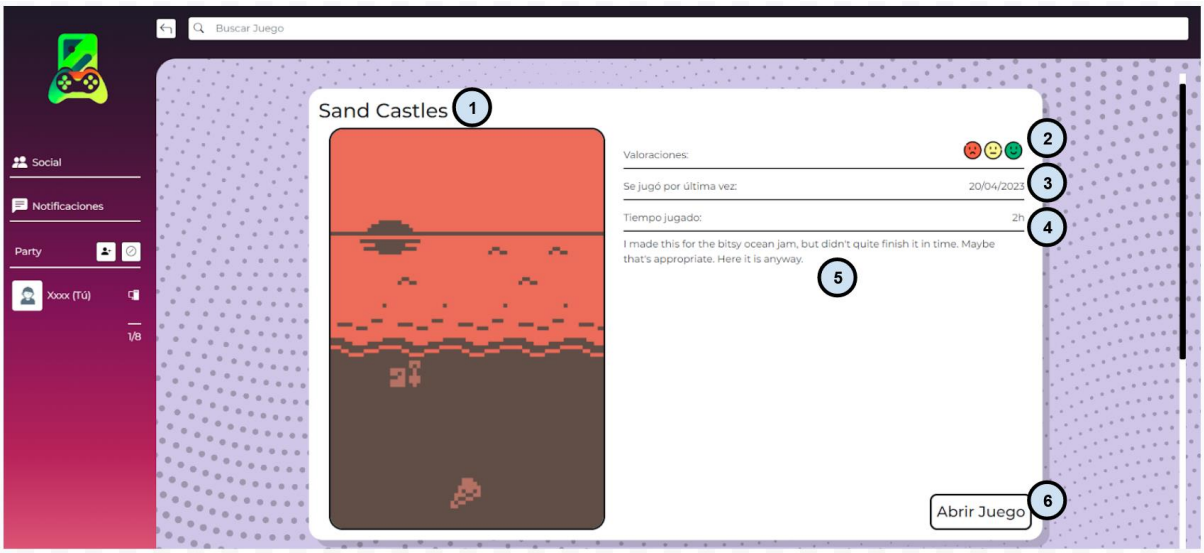
Una vez se ha seleccionado un juego del menú, se abrirá la página con la información de este. La página por defecto de la información del juego está dividida en tres partes: ficha del juego, cuadro para escribir tu reseña y el listado de los últimos comentarios sobre el juego.

Una vez se ha seleccionado un juego del menú, se abrirá la página correspondiente con la información detallada del mismo. La página de información del juego se divide en tres secciones principales: la ficha del juego, el espacio para escribir tu reseña y el listado de los últimos comentarios sobre el juego.

En la ficha del juego se proporcionan detalles importantes como el título, la portada del juego, las valoraciones, la fecha de la última vez que se jugó, el tiempo total de juego acumulado y una descripción del juego. También se encuentra el botón que permite abrir el juego.

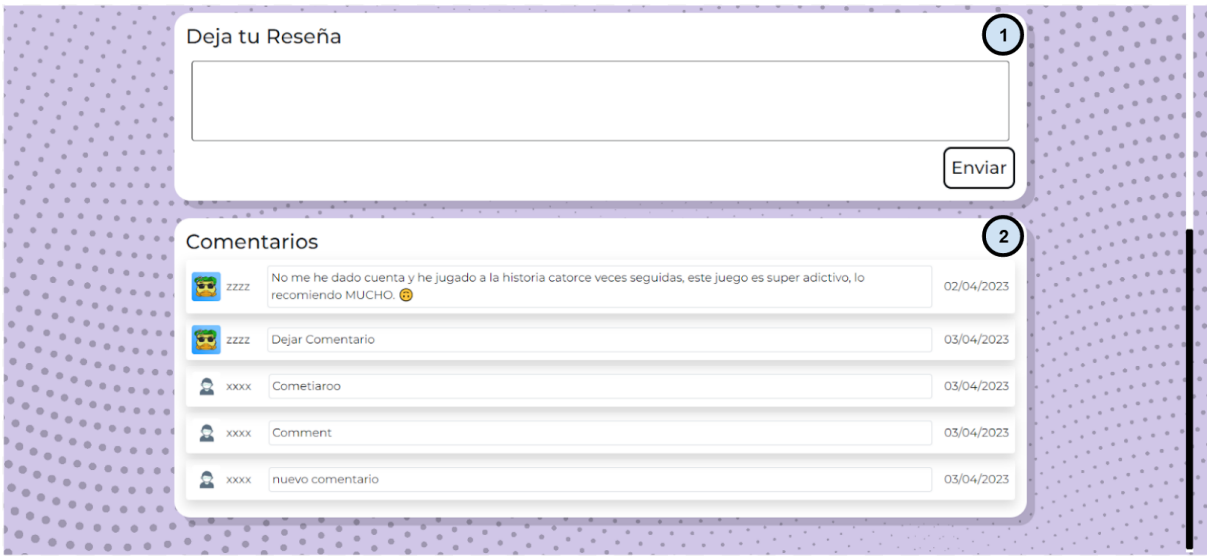
El cuadro destinado a escribir tu reseña permite expresar tus opiniones y experiencias sobre el juego. Aquí se puede compartir impresiones del juego, destacar aspectos positivos o mencionar áreas de mejora.

Por último, se encuentra el listado de los últimos comentarios sobre el juego. Esta sección muestra las opiniones y reseñas más recientes de otros usuarios.



1) Título del juego
2) Valoraciones del juego
3) Última vez jugado
4) Tiempo jugado
5) Descripción del juego
6) Botón para lanzar el

Ilustración 38. Pantalla de juego - Información principal

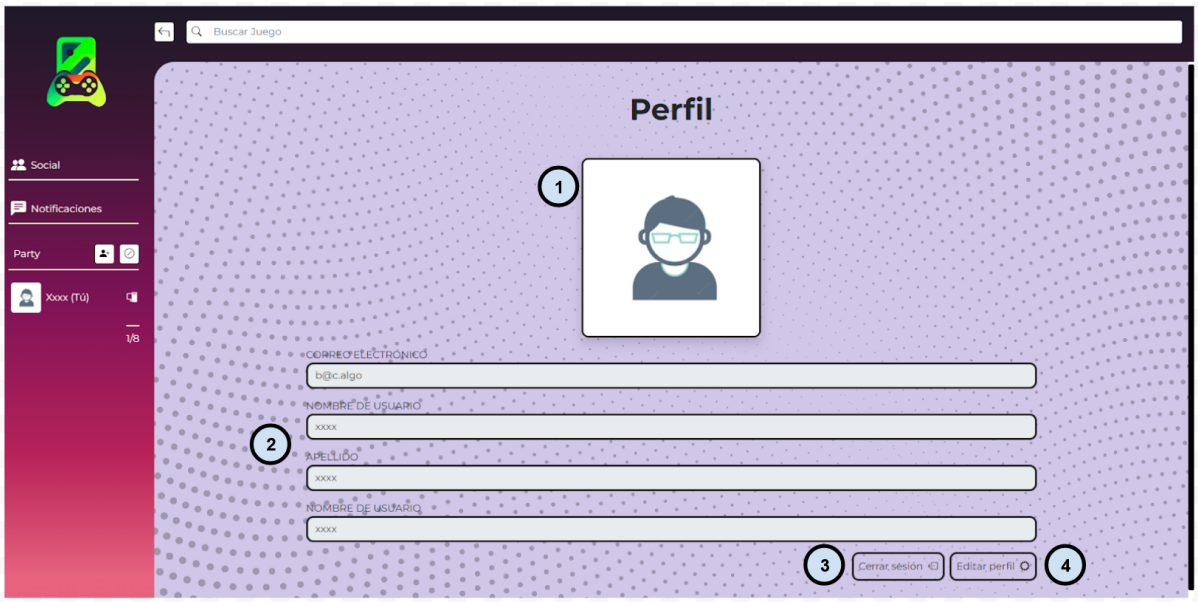


1) Entrada de texto para dejar un comentario
2) Listado de comentarios

Ilustración 39. Pantalla de juego – Reseñas

Esta vista siempre puede ser modificada por los propios desarrolladores que quieran subir a la plataforma sus creaciones con una vista más personalizada para ellos.

Otra página parte de la aplicación principal es la del perfil del usuario. A esta página se accede pulsando el icono en el listado de usuario de la sala y contiene información del perfil del usuario y los botones de cerrar sesión y editar los datos del perfil.



1) Foto de perfil
2) Información del usuario
3) Botón de cerrar sesión
4) Botón para editar el perfil

Ilustración 40. Pantalla de usuario

Por último, se encuentra la página de edición de perfil, donde se modifica la información del usuario.

1) Foto de perfil
2) Entradas de datos necesarios del usuario
3) Botón de cerrar sesión

Ilustración 41. Pantalla de actualizar perfil

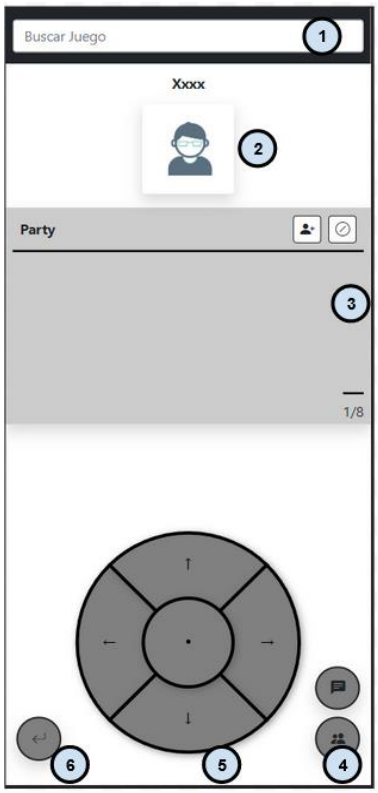
Aplicación móvil

Si se entra a la aplicación desde un dispositivo móvil se accederá a la misma página de login de la versión de escritorio, pero ajustada a la pantalla del dispositivo. Una vez dentro de la aplicación móvil se mostrará el controlador remoto de la aplicación. Este controlador contiene y permite el acceso a todos los elementos de la aplicación de escritorio desde el teléfono.

En la parte superior de la página se encuentra la barra de búsqueda de juegos y justo debajo la imagen del nombre de usuario y foto de perfil del usuario logueado.

En el centro de la página, se ubicará la información de la sala en la que se encuentra. Aquí, se encuentran el acceso a los botones de añadir amigos a la sala y abandonar la sala.

La parte inferior de la página está formada por los botones de navegación y los de acceso a las notificaciones y a añadir amigos. Dentro de los botones de navegación se encuentran el botón de retorno, que funciona de la misma manera que en la aplicación de escritorio y la cruceta de navegación, que permite moverse y acceder a los diferentes elementos de la web.



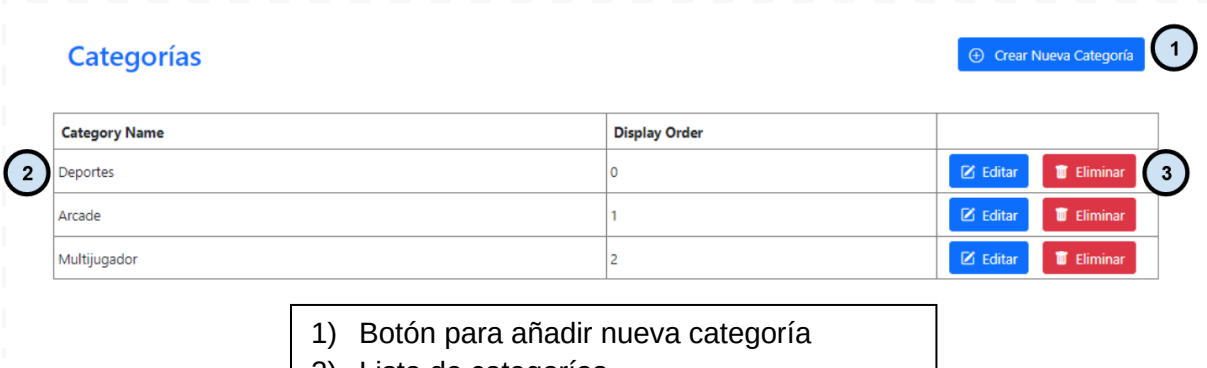
1) Entrada de texto para búsqueda de juegos
2) Imagen de usuario y botón para abrir la pantalla de información del usuario
3) Grupo del jugador con botón de invitar amigo, botón de abandonar grupo actual y listado de usuarios en el grupo actual
4) Botón de abrir notificaciones y botón de social
5) Botonera de navegación
6) Botón de vuelta a atrás

Ilustración 42. Pantalla principal de la aplicación móvil

Administración

Con el fin de simplificar las tareas administrativas relacionadas con la gestión de juegos, se han desarrollado diversas vistas que permiten la creación y edición de juegos, así como de sus categorías.

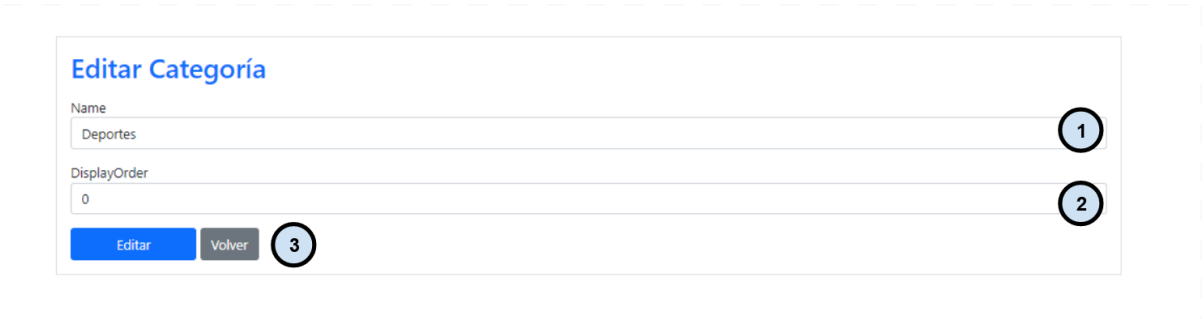
La vista de administración de las categorías contiene el listado de todas las categorías existentes en la aplicación y las opciones de añadir o editar categorías.



1) Botón para añadir nueva categoría
2) Lista de categorías
3) Botones de editar o eliminar categoría

Ilustración 43, Pantalla principal del administrador de categorías

Las vistas de creación y edición de categorías son similares, en ellas se encuentran los apartados para rellenar *"Category Name"* y *"DisplayOrder"*, donde se añaden el nombre de la categoría y el orden en el que será mostrada en el menú principal de la aplicación.



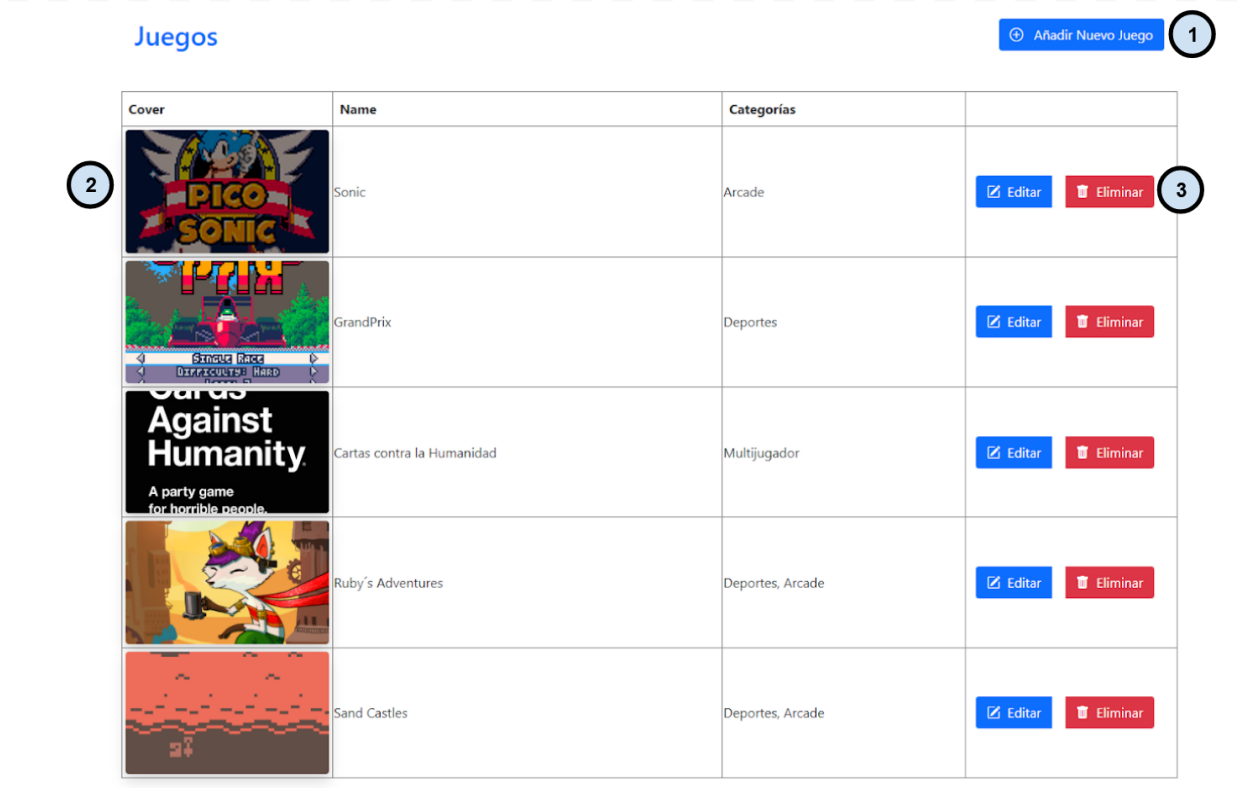
1) Entrada para el nombre de la categoría
2) Entrada para el orden para mostrar la categoría
3) Botones para editar la categoría o volver

Ilustración 44. Pantalla de edición de categorías

1) Entrada para el nombre de la categoría
2) Entrada para el orden para mostrar la categoría
3) Botones para crear la categoría o volver

Ilustración 45. Pantalla de creación de categorías

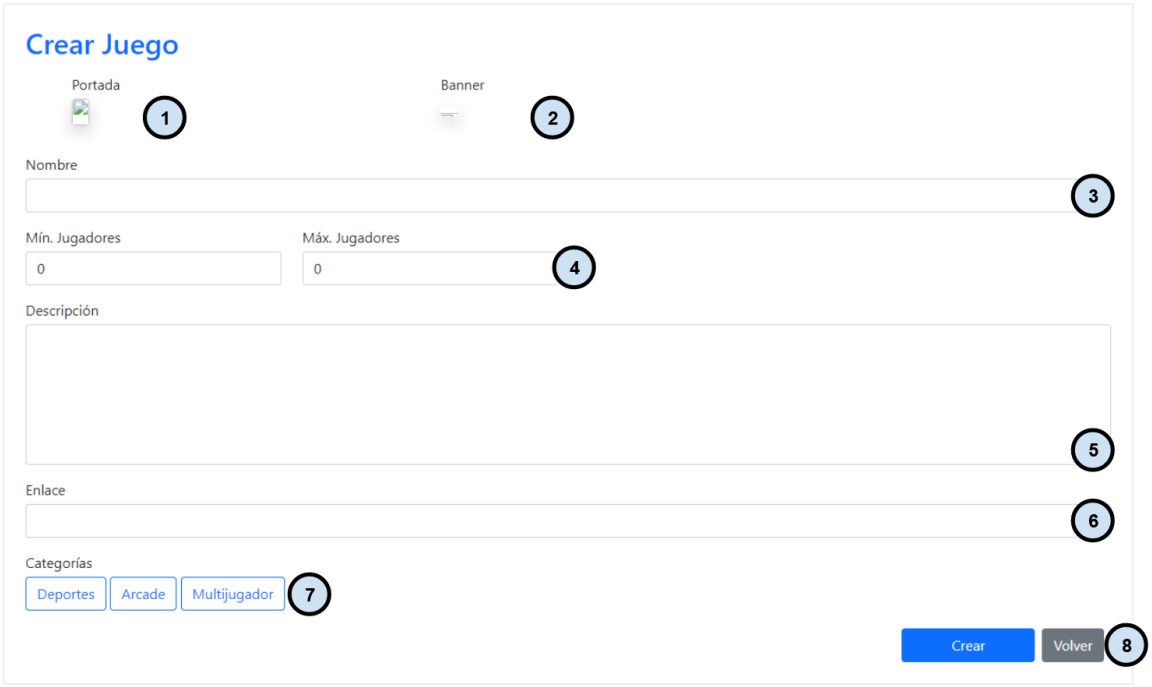
La vista de administración de juegos contiene el listado de todos los juegos de la plataforma y permite dar de alta nuevos juegos, editar la información de los ya existentes o eliminarlos.



1) Botón de añadir juego
2) Listado de juegos
3) Botones para editar y eliminar un juego

Ilustración 46. Pantalla principal de administración de juegos

En las vistas de creación y edición de juegos se solicita la información necesaria para mostrar los juegos en la aplicación.



1) Portada del juego
2) Banner del juego
3) Nombre del juego
4) Números mínimos y máximos del juego
5) Descripción del juego
6) Enlace al directorio virtual
7) Categorías del juego
8) Botones de crear y volver

Ilustración 47. Pantalla de creación de juego

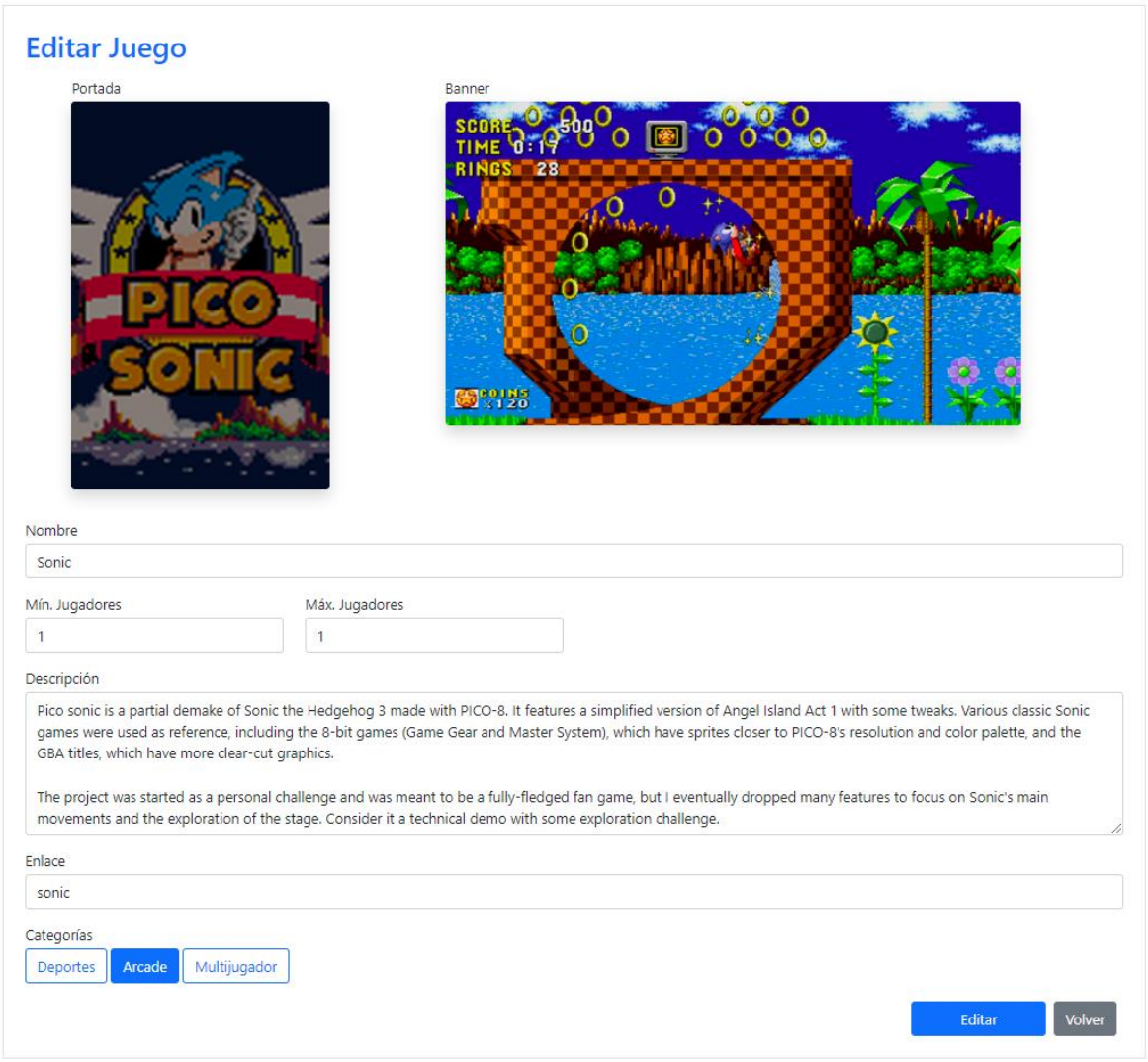


Ilustración 48. Pantalla de edición de juego

Juegos

La aplicación cuenta con una variedad de juegos para un solo jugador desarrollados por estudios externos con licencia MIT (*Massachusetts Institute of Technology* - MIT). Al seleccionar un juego, se cargará la página correspondiente en la aplicación de escritorio, mientras que en el dispositivo móvil se mostrará el controlador del juego asociado.

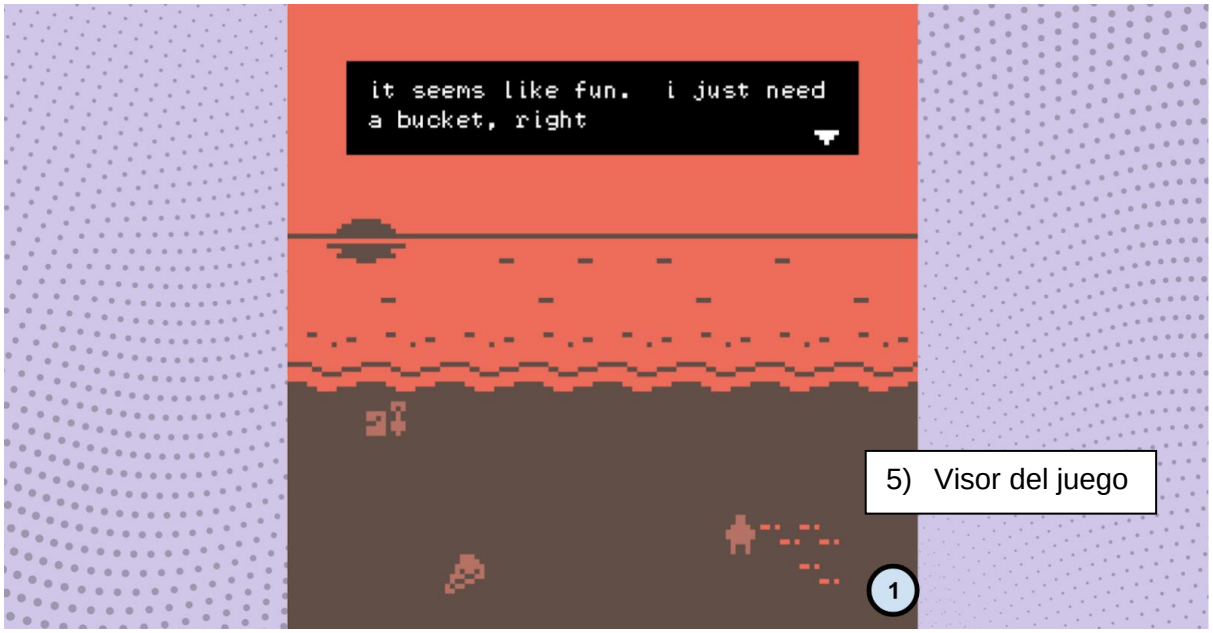
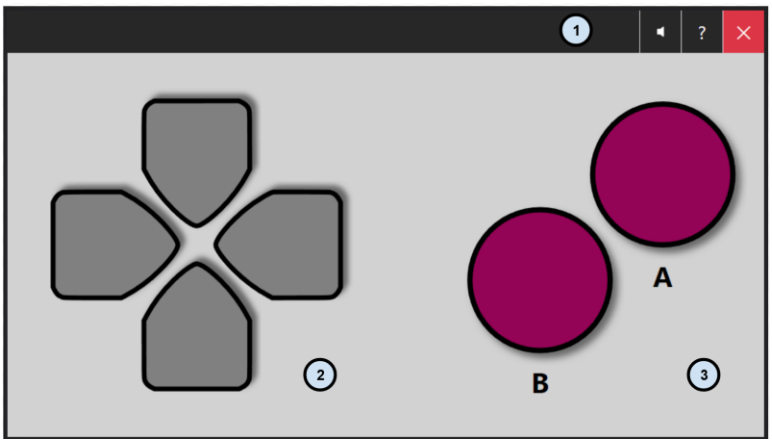


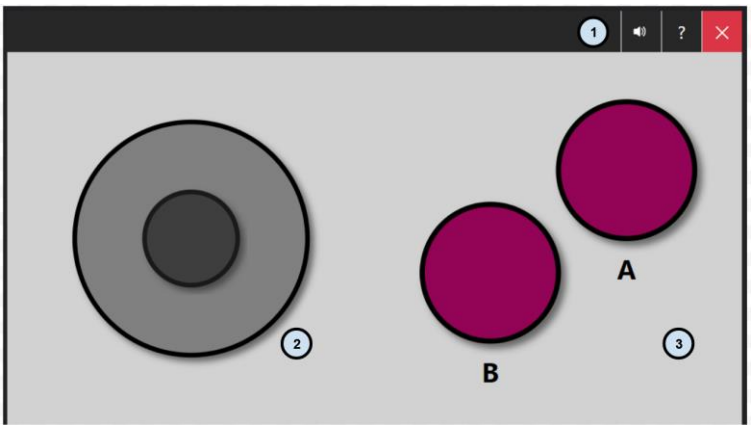
Ilustración 49. Pantalla principal del juego lanzado

Se han diseñado dos controladores estándar que los desarrolladores pueden utilizar al publicar sus juegos, o bien, tienen la opción de crear un controlador personalizado específico para su juego.



1) Botones de silenciar juego, ayuda contextual y salir del juego
2) Cruceta de movimiento
3) Botones de acción

Ilustración 50. Controlador tipo Crosspad



1) Botones de silenciar juego, ayuda contextual y salir del juego
2) Joystick de movimiento
3) Botones de acción

Ilustración 51. Controlador tipo Joystick

Juego Cartas contra la humanidad.

Se optó por desarrollar un juego que aprovechará las funcionalidades multijugador que aporta la plataforma. Para ello se desarrolló una versión del juego *Cards Against Humanity* publicado por *Cards Against Humanity LLC* bajo la licencia *CC-BY-NC-SA 2.0*.

Cards Against Humanity es un juego cuyo objetivo es crear combinaciones de cartas que generen respuestas divertidas, sorprendentes o chocantes.

El juego consta de dos tipos de cartas: las cartas negras y las cartas blancas. Las cartas negras contienen frases o preguntas incompletas, mientras que las cartas blancas tienen palabras o frases para completar las respuestas.

Durante cada ronda, un jugador actúa como juez y selecciona una carta negra, la lee en voz alta y la muestra a los demás jugadores. Los demás jugadores deben seleccionar una o varias cartas blancas de su mano para completar la frase o responder a la pregunta de la carta negra.

Una vez que todos los jugadores han elegido sus cartas blancas, se entregan al juez de forma anónima. El juez mezcla las respuestas y las lee en voz alta, incluyendo la carta negra original. Luego, el juez selecciona la respuesta más divertida, absurda o impactante, y el jugador que proporcionó esa respuesta gana un punto.

El papel de juez pasa a la siguiente persona en cada ronda, y el juego continúa hasta que los jugadores decidan finalizarlo o alcanzar un número predeterminado de puntos.

En la versión desarrollada no existe la figura del juez, desde la aplicación de escritorio se visualizará y escuchará la pregunta o frase incompleta, una vez acabado el tiempo de respuesta o cuando todos los jugadores hayan respondido se realizará una votación entre todos los jugadores para elegir la respuesta ganadora.

El juego se ha desarrollado en varias vistas, a continuación, se explicará el funcionamiento del juego a partir de las vistas que lo componen.

Pantalla de inicio

Cuando se accede al juego se cargará en todos los dispositivos unidos a la sala la pantalla de inicio del juego, esta pantalla es diferente para la aplicación de escritorio, el propietario de la sala o el resto de los dispositivos móviles conectados.
Las aplicaciones de escritorio mostrarán el título del juego y un texto informativo indicando el número de jugadores pendiente de conectarse o que se está esperando a que el anfitrión de comienzo la partida.

Ilustración 52. Página de inicio del juego Cartas contra la Humanidad versión escritorio

En la pantalla del teléfono del anfitrión se muestra, en el momento que todos los usuarios estén conectados, un pequeño menú con las opciones de iniciar partida, un selector del número de preguntas y un botón de salir del juego.

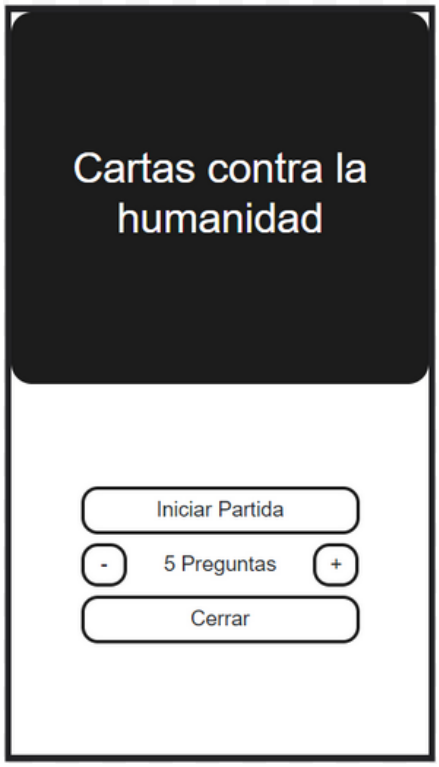


Ilustración 53. Página de inicio del juego Cartas contra la Humanidad versión móvil anfitrión.

El resto de los jugadores verán una pantalla similar a la aplicación de escritorio con la información de los jugadores pendientes de conectar o que se está esperando al anfitrión a que comience el juego.

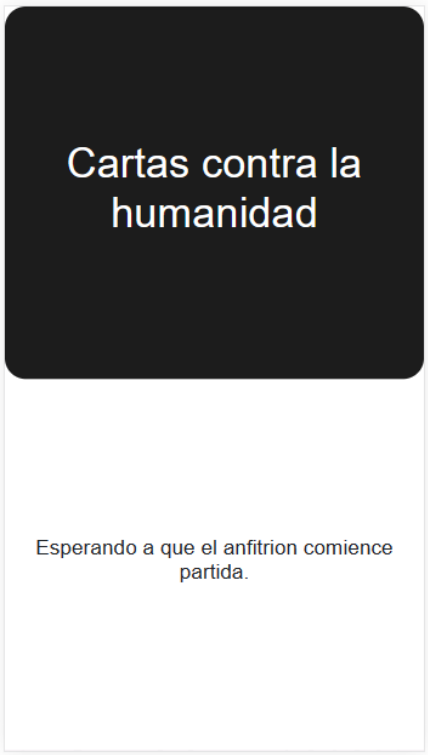


Ilustración 54. Página de inicio del juego Cartas contra la Humanidad versión móvil invitado.

Pantalla de pregunta

Una vez comienza el juego, en la pantalla de la aplicación de escritorio se mostrará la pregunta o frase a completar por los jugadores, la cuenta atrás para responder y el listado de los jugadores indicando si han enviado o no su respuesta.

1) Tiempo restante
2) Usuario sin responder
3) Usuario con respuesta enviada

Ilustración 55. Pantalla de selección de respuesta de Cartas contra la Humanidad en la versión de escritorio

En el dispositivo de los jugadores, se mostrará un conjunto de cartas dispuestas en un scroll horizontal. Estas cartas han sido previamente repartidas de manera aleatoria a cada jugador de la partida y podrán ser utilizadas para responder a la pregunta planteada. Para seleccionar la carta que se desea utilizar, simplemente se debe pulsar sobre ella. Tras seleccionar una carta, el texto de esta rellenará el valor de la caja de texto "*Hueco 1*". En caso de haber varios huecos a rellenar en la frase que aparece en la aplicación de escritorio, al pulsar sobre la carta aparecerá una ventana emergente que pregunta el hueco en el que se quiere responder.

1) Botones de ayuda contextual y salir del juego
2) Título de sección
3) Carta de respuesta
4) Listado de cartas del mazo del usuario
5) Hueco de respuesta
6) Botón de envío de respuesta
7) Modal de respuesta múltiple

Ilustración 56. Pantalla de selección de respuesta de Cartas contra la Humanidad en la versión móvil

Pantalla de votación

Una vez finalizada la fase de contestar a la pregunta se pasará a la selección de la respuesta favorita. En esta fase los jugadores deben elegir la respuesta que más les guste de las que han enviado el resto de los jugadores.
En la página de la aplicación de escritorio se dispondrán las respuestas de cada jugador ordenadas aleatoriamente y clasificadas con un número según el orden en que aparecen.

Ilustración 57. Página de votación de respuestas versión escritorio

En los teléfonos se mostrará las opciones de votación de cada respuesta. Los jugadores tienen deshabilitada la capacidad de marcar su respuesta. En caso de que se desee desmarcar una respuesta será necesario volver a pulsar sobre la respuesta seleccionada.
Cuando la partida está formada por más de 6 jugadores se deberá votar dos opciones en lugar de una.

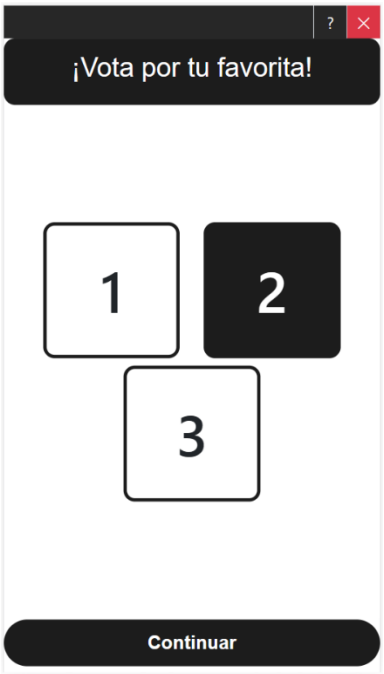


Ilustración 58. Página de votación de respuestas versión móvil

Pantalla de resultados de la votación

Al terminar la votación se accederá a los resultados. En la pantalla de la aplicación de escritorio se desvelará a quién pertenecía cada respuesta mostrando la foto y el nombre del usuario en el lado izquierdo de cada respuesta. El resultado de la votación aparecerá con una animación que rellenará el porcentaje de votos conseguido por cada jugador.

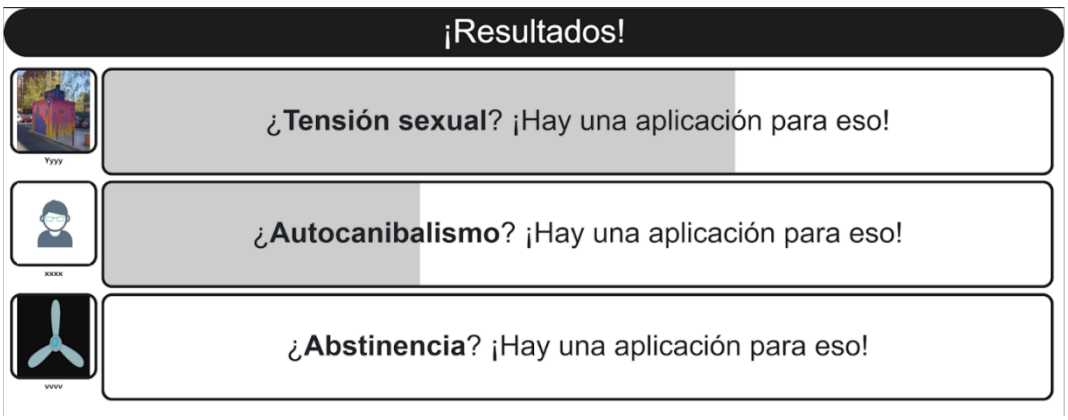


Ilustración 59. Página de resultados de la votación versión escritorio

En la pantalla de cada jugador únicamente tendrán un texto informativo y un botón de continuar para pulsar cuando quieran pasar a la siguiente vista.

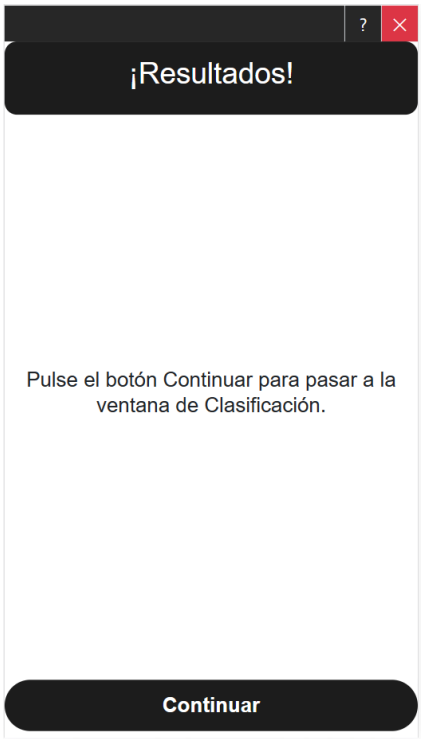


Ilustración 60. Página de resultados de la votación versión móvil

Pantalla de clasificación global

La pantalla de clasificación global muestra con una animación las puntuaciones de la partida ordenadas de mayor a menor.

Ilustración 61. Página de pantalla de clasificación global versión escritorio

La pantalla de clasificación de los dispositivos móviles es similar a la de resultados de votación, formada por el texto informativo y un botón de continuar. En caso de ser la última pregunta al pulsar se volverá a la vista de comenzar partida.

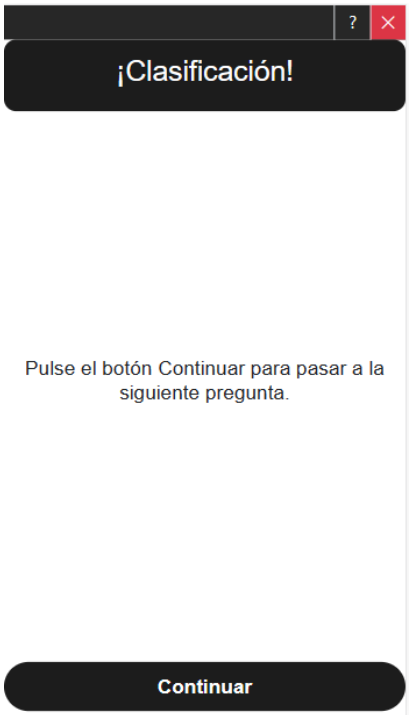


Ilustración 62. Página de pantalla de clasificación global versión móvil

La distribución de los puntos dependerá del número de jugadores de la partida siguiendo la siguiente manera:
Para partidas de menos de 6 jugadores, se repartirá 3 puntos al ganador de la ronda. En caso de haber un empate en la primera posición 1 a cada jugador empatado en primera posición.
Para las partidas de 6 o más jugadores se repartirá 5 puntos entre todos los ganadores redondeando el reparto hacia arriba al entero más cercano. Quienes hayan quedado segundos también sumarán puntos, siempre y cuando no haya habido un empate en primera posición. Se asignarán 2 puntos entre los jugadores que hayan quedado en segunda posición redondeando el reparto hacia arriba al entero más cercano.

Pantalla de espera

La pantalla de espera se muestra una vez que el usuario ha realizado su acción y debe esperar a que el resto de los jugadores realice la suya. Únicamente está compuesta por un texto informativo. La siguiente pantalla se cargará cuando todos los jugadores hayan terminado o el contador llegue a cero en el caso de la pantalla de preguntas.

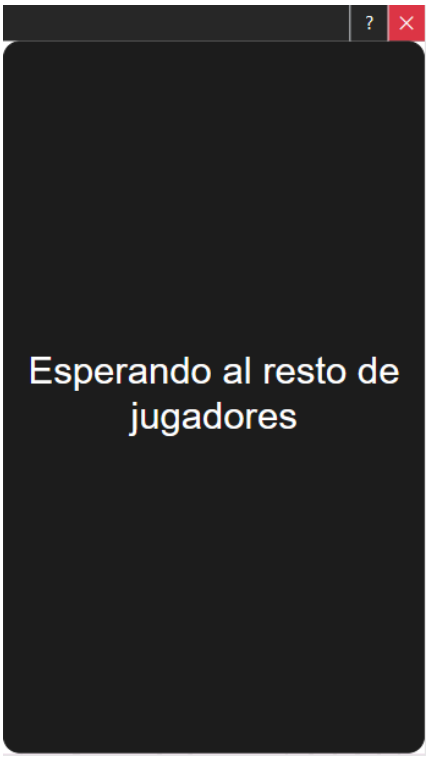


Ilustración 63. Pantalla de espera

Capítulo 4

Descripción informática

Herramientas

Durante el proceso de desarrollo de la aplicación se utilizado diversas herramientas software las cuales se indican a continuación:

Visual Studio 2022 es hasta la fecha la última versión de un entorno de desarrollo integrado (*Integrated development environment* - IDE) desarrollado por *Microsoft*. Es una herramienta muy popular utilizada por desarrolladores de software para crear una variedad de aplicaciones, incluyendo aplicaciones web, de escritorio, móviles o aplicaciones para la nube.

Azure DevOps es una plataforma integral de colaboración y gestión del ciclo de vida de desarrollo de software ofrecida por *Microsoft*. Proporciona una serie de herramientas y servicios que permiten a los equipos de desarrollo planificar, desarrollar, probar y entregar software de manera más eficiente.

SourceTree es una herramienta de cliente de escritorio gratuita y popular para la gestión de repositorios de código. Es desarrollada por *Atlassian* y se utiliza principalmente como una interfaz gráfica de usuario (*Graphical user interface* - GUI) para trabajar con sistemas de control de versiones, como *Git* y *Mercurial*.

Azure es una plataforma de computación en la nube desarrollada por *Microsoft*. Proporciona un conjunto integral de servicios y herramientas para la creación, implementación y administración de una amplia gama de aplicaciones y servicios en la nube.

Teams es una plataforma de colaboración y comunicación basada en la nube desarrollada por *Microsoft*. Está diseñada para facilitar la comunicación y la colaboración en equipo, especialmente en entornos de trabajo remotos o distribuidos.

WinSCP (*Windows Secure Copy*) es un programa de *software* de código abierto y gratuito que proporciona una interfaz gráfica de usuario para transferir archivos entre un cliente y un servidor remoto utilizando los protocolos de transferencia de archivos Protocolo de copia segura (*Secure Copy Protocol* - SCP) y Protocolo de transferencia de archivos SSH (*SSH File Transfer Protocol* - SFTP).

Universidad Rey Juan Carlos | Escuela Técnica Superior Ingeniería Informática

Análisis de requisitos

Aplicación web

Requisitos Funcionales aplicación principal

ID	Nombre	Descripción
#RF1	Registro de usuarios	Permite a los usuarios crear una cuenta en la aplicación para guardar su progreso, puntuaciones y personalización.
#RF2	Verificación de email	La aplicación verifica si el email introducido durante el registro tiene el formato correcto.
#RF3	Comprobación cuenta ya registrada	La aplicación verifica si el email introducido ya está dado de alta.
#RF4	Inicio de sesión	Permitir a los usuarios iniciar sesión en sus cuentas utilizando sus credenciales registradas.
#RF5	Inicio de sesión múltiple	Impide la conexión de un usuario de manera simultánea con el mismo tipo de dispositivo.
#RF6	Detección del tipo de dispositivo	La aplicación detecta el dispositivo con el que está conectado (escritorio o móvil).
#RF7	Diferentes pantallas de inicio según dispositivo	La aplicación mostrará el menú principal cuando el usuario se loguea desde el ordenador y el controlador cuando se loguea desde móvil.
#RF8	Selección de juegos	Proporcionar a los usuarios una variedad de juegos para elegir.
#RF9	Carrusel de juegos	En la cabecera del menú principal se presentarán los últimos juegos añadidos dentro de un carrusel.
#RF10	Agrupación de juegos	Los juegos se mostrarán en el menú principal agrupados por su categoría.
#RF11	Mostrar juegos en el menú	Los juegos deben de mostrarse con la portada del juego y su título.
#RF12	Número máximo de jugadores	Los juegos tienen un número máximo de jugadores al que pueden jugar a la vez.

#RF13	Número mínimo de jugadores	Los juegos tienen un número mínimo de jugadores al que pueden jugar a la vez.
#RF14	Descripción del juego	Los juegos tienen una descripción informativa.
#RF15	Portada del juego	Los juegos tienen una portada.
#RF16	Portada carrusel	Los juegos tienen una portada para el carrusel.
#RF17	Vista de juegos	Los juegos tienen una vista donde se ve toda la información del juego.
#RF18	Reseñas	Los usuarios pueden dejar reseñas de los juegos.
#RF19	Mostrar reseñas	Las reseñas se muestran en la vista del juego.
#RF20	Orden reseñas	Las reseñas se deben mostrar en orden decreciente de fecha de alta.
#RF21	Tiempo jugado	El sistema debe almacenar el tiempo jugado en cada partida por cada usuario.
#RF22	Mostrar tiempo jugado	El sistema debe mostrar el tiempo total jugado por cada usuario a un juego.
#RF23	Calificación de juego	El jugador puede calificar el juego.
#RF24	Mostrar calificación del juego	El sistema debe de mostrar en la vista del juego la calificación otorgada por los usuarios.
#RF25	Solicitudes de amistad	Los usuarios pueden enviar solicitudes de amistad a otros usuarios.
#RF26	Aceptar solicitud de amistad	Los usuarios pueden aceptar solicitudes de amistad a otros usuarios.
#RF27	Denegar solicitud de amistad	Los usuarios pueden denegar solicitudes de amistad a otros usuarios.
#RF28	Creación de salas de juego	Permitir a los usuarios crear salas virtuales cuando están conectados con escritorio y móvil.
#RF29	Solicitudes de unión a sala	Los usuarios pueden enviar solicitudes de unión a sala.

#RF30	Aceptar solicitud de unión a sala	Los usuarios pueden aceptar solicitudes de unión a sala.
#RF31	Denegar solicitud de unión a sala	Los usuarios pueden denegar solicitudes de unión a sala.
#RF32	Iniciar partida	Los propietarios de una sala pueden iniciar una partida.
#RF33	Mostrar usuarios de la sala	El sistema mostrará los usuarios conectados a la sala.
#RF34	Mostrar propietario de la sala	El sistema marcará qué usuario conectado a la sala es el propietario.
#RF35	Administrar sala	El usuario puede deshacer la sala o expulsar usuarios de las salas.
#RF36	Controlador móvil	El escritorio puede ser controlado desde un móvil logueado con la misma cuenta.
#RF37	Cruceta de control	El móvil tendrá una cruceta con la que controlar el movimiento dentro de los elementos del escritorio.
#RF38	Administrador sala móvil	Desde el móvil se tendrá acceso directo a los elementos de administración de la sala.
#RF39	Amistad móvil	Desde el móvil se tendrá acceso directo a los elementos de amistad.
#RF40	Buscar juegos	Se podrá realizar una búsqueda de juegos desde la aplicación.
#RF41	Gestión de perfiles de usuario	Permitir a los usuarios editar sus perfiles y cambiar la foto de perfil.
#RF42	Cambio contraseña	Permitir a los usuarios editar su contraseña de acceso.

Tabla 4. Requisitos funcionales de la aplicación principal

Requisitos no funcionales aplicación principal

ID	Nombre	Descripción
#RNF1	Usabilidad	La página web debe ser intuitiva y fácil de usar, con una interfaz de usuario clara y coherente
#RNF2	Rendimiento	La página web debe tener un tiempo de carga rápido y ser capaz de manejar un alto volumen de usuarios simultáneos sin retrasos significativos. Debe garantizar un rendimiento eficiente y una respuesta rápida a las acciones del usuario.
#RNF3	Escalabilidad	La página web debe ser capaz de adaptarse y escalar para manejar un crecimiento futuro en términos de tráfico y usuarios. Debe ser capaz de gestionar de manera efectiva la carga adicional sin comprometer el rendimiento.
#RNF4	Seguridad	La página web debe implementar medidas de seguridad sólidas para proteger la información personal de los usuarios, como contraseñas y datos privados.
#RNF5	Compatibilidad con navegadores y dispositivos	La página web debe ser compatible con una amplia gama de navegadores web populares, como *Chrome*, *Firefox*, *Safari* y *Edge*. Además, debe ser responsive y adaptable a diferentes tamaños de pantalla y dispositivos, como computadoras de escritorio, laptops, tabletas y teléfonos inteligentes.
#RNF6	Mantenibilidad	La página web debe ser fácil de mantener y actualizar. Debe tener una arquitectura modular y un código limpio y bien estructurado que facilite la implementación de cambios y mejoras futuras.
#RNF7	Tolerancia a fallos	La página web debe ser capaz de manejar de manera robusta situaciones inesperadas, como caídas del servidor, pérdida de conexión a internet o errores de sistema. Debe tener mecanismos de recuperación y una capacidad adecuada para mitigar los efectos de los fallos.

Tabla 5. Requisitos no funcionales de la aplicación principal

Desarrollo de juegos

Requisitos funcionales desarrollo de juegos

ID	Nombre	Descripción
#RF1	Mantener sala aplicación Principal	Al iniciar un juego se debe mantener la sala de la aplicación principal
#RF2	Desconexiones cierre navegador	La aplicación debe controlar las desconexiones por cierre del navegador.
#RF3	Reconexiones del navegador	La aplicación debe controlar las reconexiones a la aplicación.
#RF4	Control número mínimo jugadores al cerrar conexión	La aplicación debe de controlar las desconexiones de los miembros de los grupos y cerrar el juego si no hay suficientes jugadores
#RF5	Control del cierre del anfitrión y de las sesiones de escritorio.	La aplicación debe de controlar las desconexiones de los miembros de los grupos y cerrar el juego si se desconecta el usuario anfitrión o todas las sesiones de escritorio.
#RF6	Detección anfitrión	La aplicación debe de identificar el usuario anfitrión de un grupo.
#RF7	Detección sesiones de escritorio	La aplicación debe identificar a los usuarios con la aplicación de escritorio.
#RF8	Detección móvil	La aplicación debe de identificar los usuarios móviles.
#RF9	Juegos *HTML5*	La aplicación de escritorio debe poder reproducir juegos en *HTML5*.
#RF10	Juegos *WebGL*	La aplicación de escritorio debe poder reproducir juegos en *WebGL.*
#RF11	Audio al comenzar el juego	La aplicación de escritorio debe de comenzar a emitir audio del juego a iniciarse.
#RF12	Transmisión de acciones del móvil	La aplicación de escritorio debe poder recibir las señales del móvil y enviarlas al juego.
#RF13	Transmisión de configuración al juego	La aplicación de escritorio debe poder recibir señales de salir del juego o silenciar el juego y transmitirlo a la página.

#RF14	Controlador *Joystick* y *Crosspad*	Debe de haber dos controladores de movimiento por *Crosspad* por *Joystick*.
#RF15	Transmisión fuerza X, Y *Joystick*	El controlador por cruceta debe de enviar la fuerza en ejes X e Y para controlar el movimiento.
#RF16	Botón ayuda móvil	El controlador móvil debe de tener un botón de ayuda que indique los controles del juego.
#RF17	Botón audio móvil	El controlador móvil debe de tener un botón audio que active o silencie el sonido del juego.
#RF18	Botón salir móvil	El control móvil debe de tener un botón de salida que permita cerrar el juego.
#RF19	Confirmar salir móvil	Al pulsar el botón de salida debe asegurarse de que desea salir.

Tabla 6. Requisitos funcionales del desarrollo de la parte de juegos de la aplicación

Requisitos no funcionales desarrollo de juegos

ID	Nombre	Descripción
#RNF1	Usabilidad	La página web debe ser intuitiva y fácil de usar, con una interfaz de usuario clara y coherente
#RNF2	Rendimiento	La página web debe tener un tiempo de carga rápido y ser capaz de manejar un alto volumen de usuarios simultáneos sin retrasos significativos. Debe garantizar un rendimiento eficiente y una respuesta rápida a las acciones del usuario.
#RNF3	Escalabilidad	La página web debe ser capaz de adaptarse y escalar para manejar un crecimiento futuro en términos de tráfico y usuarios. Debe ser capaz de gestionar de manera efectiva la carga adicional sin comprometer el rendimiento.
#RNF4	Seguridad	La página web debe implementar medidas de seguridad sólidas para proteger la información personal de los usuarios, como contraseñas y datos privados.

#RNF5	Compatibilidad con navegadores y dispositivos	La página web debe ser compatible con una amplia gama de navegadores web populares, como *Chrome*, *Firefox*, *Safari* y *Edge*. Además, debe ser responsive y adaptable a diferentes tamaños de pantalla y dispositivos, como computadoras de escritorio, laptops, tabletas y teléfonos inteligentes.
#RNF6	Mantenibilidad	La página web debe ser fácil de mantener y actualizar. Debe tener una arquitectura modular y un código limpio y bien estructurado que facilite la implementación de cambios y mejoras futuras.
#RNF7	Tolerancia a fallos	La página web debe ser capaz de manejar de manera robusta situaciones inesperadas, como caídas del servidor, pérdida de conexión a internet o errores de sistema. Debe tener mecanismos de recuperación y una capacidad adecuada para mitigar los efectos de los fallos.
#RNF8	Tipos de archivo	La página web debe permitir el envío al navegador de archivos de audio, imagen y WebGL.
#RNF8	Eliminación de códigos repetidos	Los códigos comunes de los proyectos de los juegos deben estar en un único proyecto común al que se pueda acceder desde ellos.

Tabla 7. Requisitos no funcionales del desarrollo de la parte de juegos de la aplicación

Desarrollo Cartas contra la humanidad

Requisitos funcionales Cartas contra la humanidad

ID	Nombre	Descripción
#RF1	Esperar a jugadores	El sistema debe esperar a que todos los jugadores hayan cargado el juego.
#RF2	Editar partida	El sistema debe permitir al propietario de la sala editar la configuración de la partida.
#RF3	Iniciar partida	El sistema debe permitir al propietario iniciar la partida.

#RF4	Finalizar la partida	El sistema debe permitir al propietario finalizar la partida.
#RF5	Obtener cartas	El sistema debe leer un archivo con las cartas del juego.
#RF6	Barajar cartas	El sistema debe ordenar aleatoriamente las cartas.
#RF7	Repartir cartas de jugador	El sistema debe repartir aleatoriamente 10 cartas a cada jugador.
#RF8	Mostrar cartas de pregunta	El sistema debe mostrar aleatoriamente una carta de pregunta en cada ronda.
#RF9	Mostrar cartas de jugadores	El sistema debe mostrar todas las cartas del jugador en sus dispositivos.
#RF10	Número de huecos	El sistema debe de obtener el número de huecos a responder de cada carta y permitir al usuario contestar con las cartas necesarias
#RF11	Cuenta atrás	El sistema debe de asignar un tiempo máximo para responder
#RF12	Votación	El sistema debe de permitir a los usuarios votar por las respuestas que más le gusten a excepción de la suya.
#RF13	Mostrar respuestas	El sistema debe de mostrar las respuestas a las tarjetas de manera aleatoria.
#RF14	Contar votos	El sistema debe recopilar los votos de los jugadores y verificar los ganadores.
#RF15	Mostrar resultados votación	El sistema debe mostrar los resultados de la votación.
#RF16	Asignar puntuación	El sistema debe calcular la puntuación de cada ronda y sumarla al global de la partida.
#RF17	Mostrar clasificación	El sistema debe de mostrar la clasificación de la partida ordenada en orden decreciente de puntos.
#RF18	Final de ronda	El sistema debe de controlar si ha finalizado la partida o no, en caso de que haya finalizado devolver a los jugadores al menú inicial, en caso contrario iniciar una nueva ronda.
#RF19	Obtener jugadores de la partida	El sistema debe obtener los jugadores de la partida de la aplicación principal.

Tabla 8. Requisitos funcionales del juego Cartas contra la humanidad

Requisitos no funcionales Cartas contra la humanidad

ID	Nombre	Descripción
#RNF1	Usabilidad	La página web debe ser intuitiva y fácil de usar, con una interfaz de usuario clara y coherente
#RNF2	Rendimiento	La página web debe tener un tiempo de carga rápido y ser capaz de manejar un alto volumen de usuarios simultáneos sin retrasos significativos. Debe garantizar un rendimiento eficiente y una respuesta rápida a las acciones del usuario.
#RNF3	Diseño visual atractivo	El juego debe tener un diseño visual atractivo y agradable y una interfaz de usuario atractiva. Esto contribuirá a una experiencia de juego más envolvente y atractiva para los jugadores.
#RNF4	Seguridad	La página web debe implementar medidas de seguridad sólidas para proteger la información personal de los usuarios, como contraseñas y datos privados.
#RNF5	Compatibilidad con navegadores y dispositivos	La página web debe ser compatible con una amplia gama de navegadores web populares, como *Chrome*, *Firefox*, *Safari* y *Edge*. Además, debe ser responsive y adaptable a diferentes tamaños de pantalla y dispositivos, como computadoras de escritorio, laptops, tabletas y teléfonos inteligentes.
#RNF6	Tolerancia a fallos	La página web debe ser capaz de manejar de manera robusta situaciones inesperadas, como caídas del servidor, pérdida de conexión a internet o errores de sistema. Debe tener mecanismos de recuperación y una capacidad adecuada para mitigar los efectos de los fallos.

Tabla 9. Requisitos no funcionales del juego Cartas contra la humanidad

Universidad Rey Juan Carlos | Escuela Técnica Superior Ingeniería Informática

Casos de Usos

Aplicación Web

Caso de Uso 1: Entrar en la web

Pasos:

1) Abrir un navegador.
2) Introducir la URL de la página web: *"https://pocketbox.azurewebsites.net"*.

Caso de Uso 2: Iniciar sesión

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 1*.
2) Se debe introducir las credenciales de la cuenta correctamente y pulsar el botón de *"Iniciar Sesión"*.

Pasos posibles:

3) Si el usuario no tiene cuenta, deberá pulsar en el enlace *"Crear una Cuenta".*
4) Deberá introducir su correo electrónico, su nombre de usuario, su nombre y una contraseña.
5) Volver al paso 2.

Caso de Uso 3: Abrir un juego

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 1* en un ordenador o dispositivo móvil.
2) Se deben realizar los pasos descritos en el *Caso de Uso 2* en el dispositivo elegido en el paso 1.
3) Se deben realizar los pasos descritos en el *Caso de Uso 1* en un dispositivo diferente al del paso 1, si ha elegido ordenador, se debe hacer con un dispositivo móvil y viceversa.
4) Se deben realizar los pasos descritos en el *Caso de Uso 2* en el dispositivo elegido en el paso 3 con las mismas credenciales utilizadas en el paso 2.
5) Se debe escoger el juego:
    a) Utilizar los botones de desplazamiento del dispositivo móvil.
    b) Utilizar el ratón del ordenador.
6) Se debe entrar en el juego:
    a) Utilizar el botón de selección del dispositivo móvil.
    b) Realizar un *click* en el juego con el ratón del ordenador.
7) Lanzar el juego:
    a) Utilizar los botones de desplazamiento y de selección del dispositivo móvil para pulsar el botón "Iniciar".
    b) Utilizar el ratón para pulsar el botón "Iniciar".

Pasos posibles:

8) Búsqueda de juego por nombre
    a) Utilizar los botones de desplazamiento y selección y pulsar la barra de búsqueda y mediante el teclado del propio dispositivo escribe el nombre del juego a buscar.
    b) Utilizar el ratón del ordenador para pulsar la barra de búsqueda y escribir en el teclado del ordenador el juego a buscar.
9) Pulsar el botón de buscar.
10) Volver al paso 5).

Caso de Uso 4: Enviar solicitud de amistad

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 1*.
2) Se deben realizar los pasos descritos en el Caso de Uso 2 en el mismo dispositivo. del paso 1
3) Pulsar el botón *"Social"* para abrir el modal de *"Amigos"*.
4) Escribir en el panel de búsqueda el nombre de usuario del jugador buscado.
5) Pulsar el botón del jugador para enviar la solicitud.

Caso de Uso 5: Enviar invitación al grupo

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 1*.
2) Se deben realizar los pasos descritos en el *Caso de Uso 2* en el mismo dispositivo.
3) Pulsar el botón de invitar amigos alojado en el apartado de *"Party"*
4) Seleccionar al jugador o jugadores añadidos a la lista de amigos que se quieran invitar

Juegos

Caso de Uso 6: Salir de un juego

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 3* en un ordenador y en el dispositivo móvil.
2) En el dispositivo móvil pulsar en la cruz roja en la parte superior derecha de la pantalla.
3) Confirmar que desea salir.

Caso de Uso 7: Seleccionar el modo de juego de 10 preguntas en el juego Cartas contra la humanidad

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 3* en un ordenador y en el dispositivo móvil seleccionando el juego *Cartas contra la Humanidad*.

2) Desde el teléfono del anfitrión pulsar sobre las flechas hasta que marque la opción de 10 preguntas.

Caso de Uso 8: Realizar una votación en el juego Cartas contra la Humanidad

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 3* en un ordenador y en el dispositivo móvil seleccionando el juego *Cartas contra la Humanidad*.
2) Esperar a que todos los dispositivos estén preparados.
3) Desde el teléfono del anfitrión pulsar sobre Iniciar partida.
4) Seleccionar la carta o combinación de cartas que desea jugar.
5) Pulsar en el botón enviar respuesta.
6) Esperar a que todos los jugadores hagan su selección.
7) Seleccionar el número de respuesta asociado a la respuesta indicada en la aplicación de escritorio.
8) Pulsar en el botón *"Continuar"*.

Caso de Uso 9: Visualizar puntuación de la partida en *Cartas contra la Humanidad*

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 8* en un ordenador y en el dispositivo móvil seleccionando el juego *Cartas contra la Humanidad*.
2) Esperar a que todos los jugadores realicen su votación.
3) Pulsar en el móvil el botón continuar después de visualizar los resultados de la votación.
4) Esperar a que todos los jugadores pulsen el botón *"Continuar"*.

Caso de Uso 10: Consultar controles en un juego

Pasos:

1) Se deben realizar los pasos descritos en el *Caso de Uso 3* en un ordenador y en el dispositivo móvil.
2) Desde el móvil pulsar en la barra de herramientas sobre el botón *"?"*.
3) Leer la información del modal.

Arquitectura de la aplicación

La siguiente imagen muestra la arquitectura de la aplicación web. Toda la aplicación está alojada dentro de un grupo de recursos de Azure.

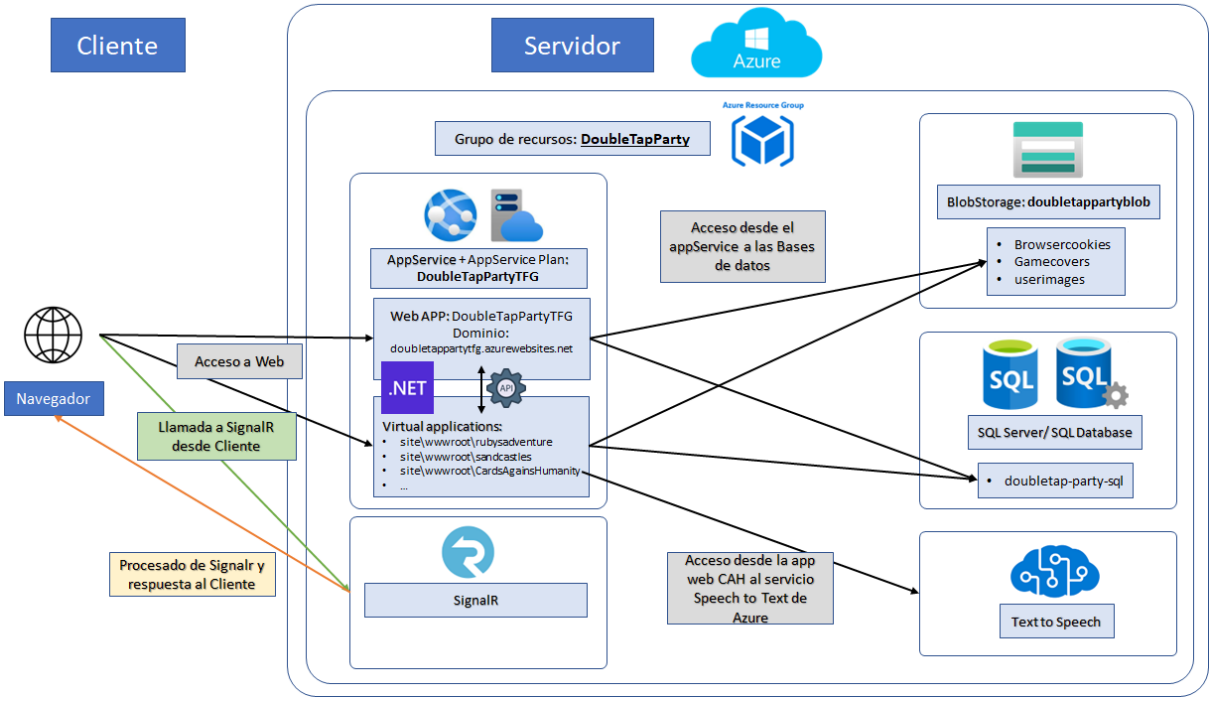


Ilustración 64. Esquema de la arquitectura de la aplicación

Los servicios asociados a este grupo de recursos incluyen:

- *BlobStorage*: Sistema de almacenamiento de datos en formato *Blob*. Este sistema se compone de tres contenedores: uno para imágenes de usuario, otro para imágenes de juegos y otro que contiene las cookies de autenticación.
- *SQL Server* y *SQL Database*: Sistema gestor de bases de datos en la nube. En este caso, se ha creado un único esquema en el que se almacena la información permanente de la aplicación.
- *Text to Speech*: Servicio de *Azure Cognitive Service*, permite convertir texto en audio.
- *AppService* + *AppServicePlan*: *AppService* es una plataforma de hospedaje de servicios web, el cual está asociado al *AppServicePlan*, que es el plan de pago contratado para levantar el servicio.

La aplicación principal se encuentra alojada en el *AppService*, y dentro de este se ejecutan múltiples aplicaciones virtuales asociadas a cada juego. Este enfoque proporciona una mayor robustez al sistema, ya que permite aislar posibles fallos, a excepción de aquellos que ocurran en la aplicación principal. Cuando un usuario se conecta a la web, la aplicación principal responde y genera la página web para el usuario. Una vez que el usuario inicia sesión y

accede a cualquier juego, será la aplicación virtual asociada a ese juego la que responda y proporcione la funcionalidad correspondiente.

Tanto las aplicaciones virtuales como la aplicación principal están programadas en *.NET*. Las aplicaciones virtuales se comunican con la aplicación principal a través de *APIs* programadas en la aplicación principal.

La comunicación en tiempo real con el servidor se realiza a través de *SignalR*, por ejemplo, al enviar una solicitud de amistad, moverse por el menú principal o controlar un juego.

Flujo de navegación

Se han generado diagramas para mostrar el flujo de navegación entre pantallas de toda la aplicación.

Aplicación Escritorio

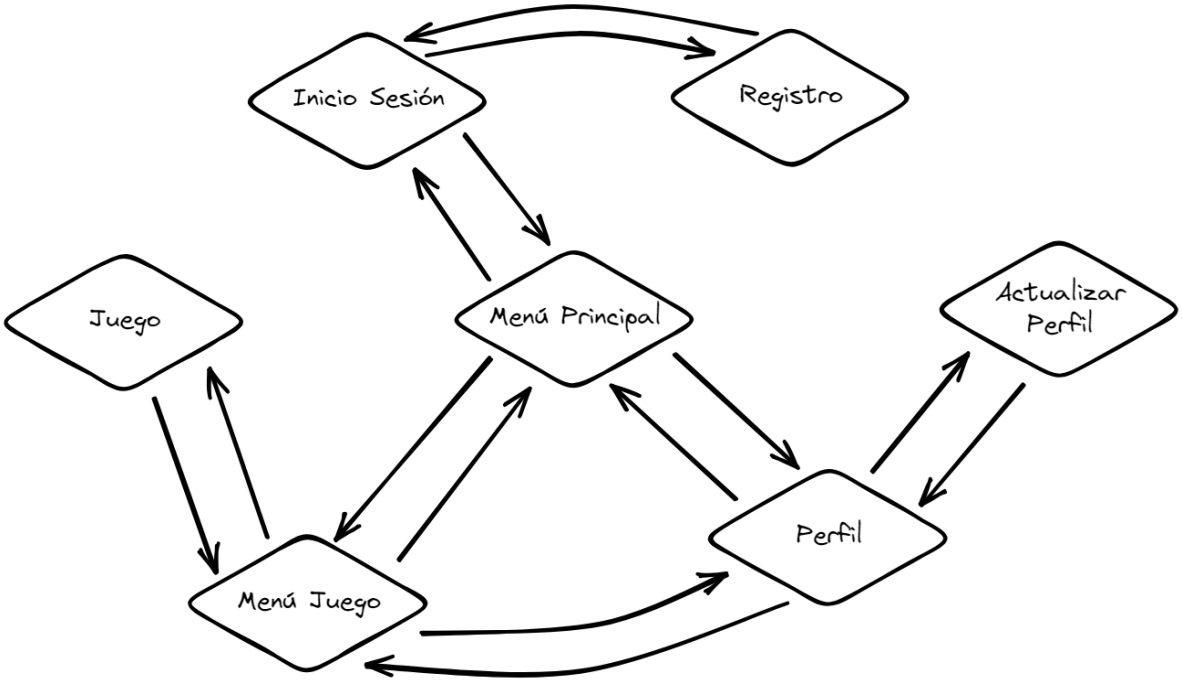


Ilustración 65. Diagrama de navegación de la versión de escritorio

Aplicación Móvil

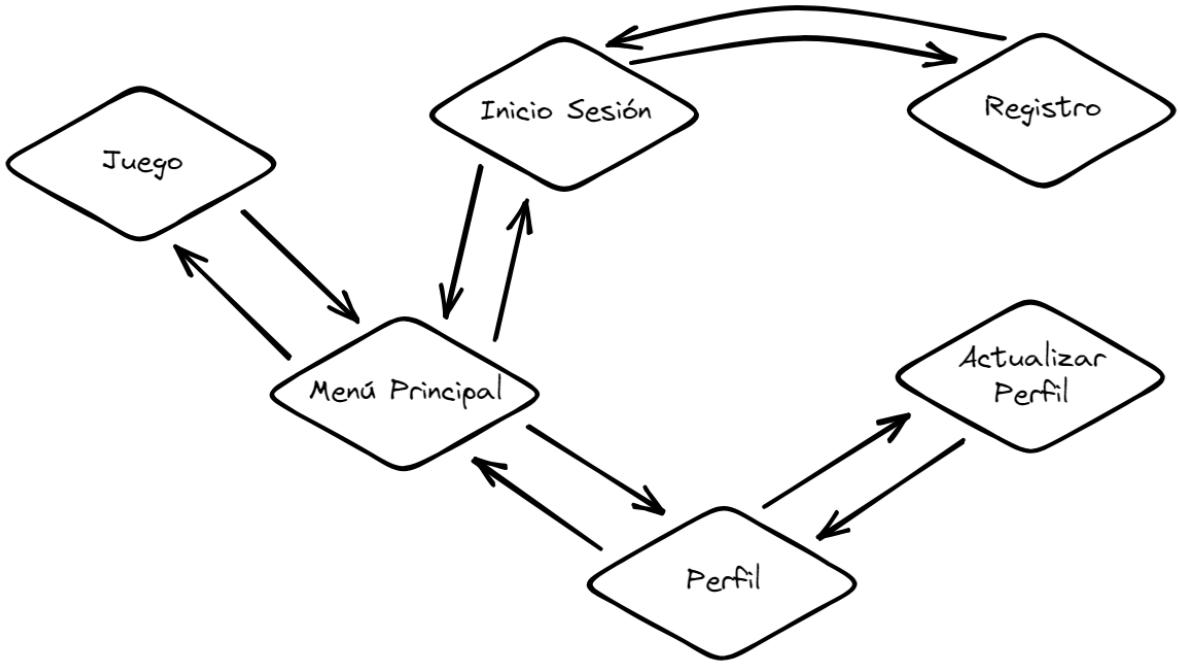


Ilustración 66. Diagrama de navegación de la versión móvil

Cartas contra la Humanidad

Escritorio

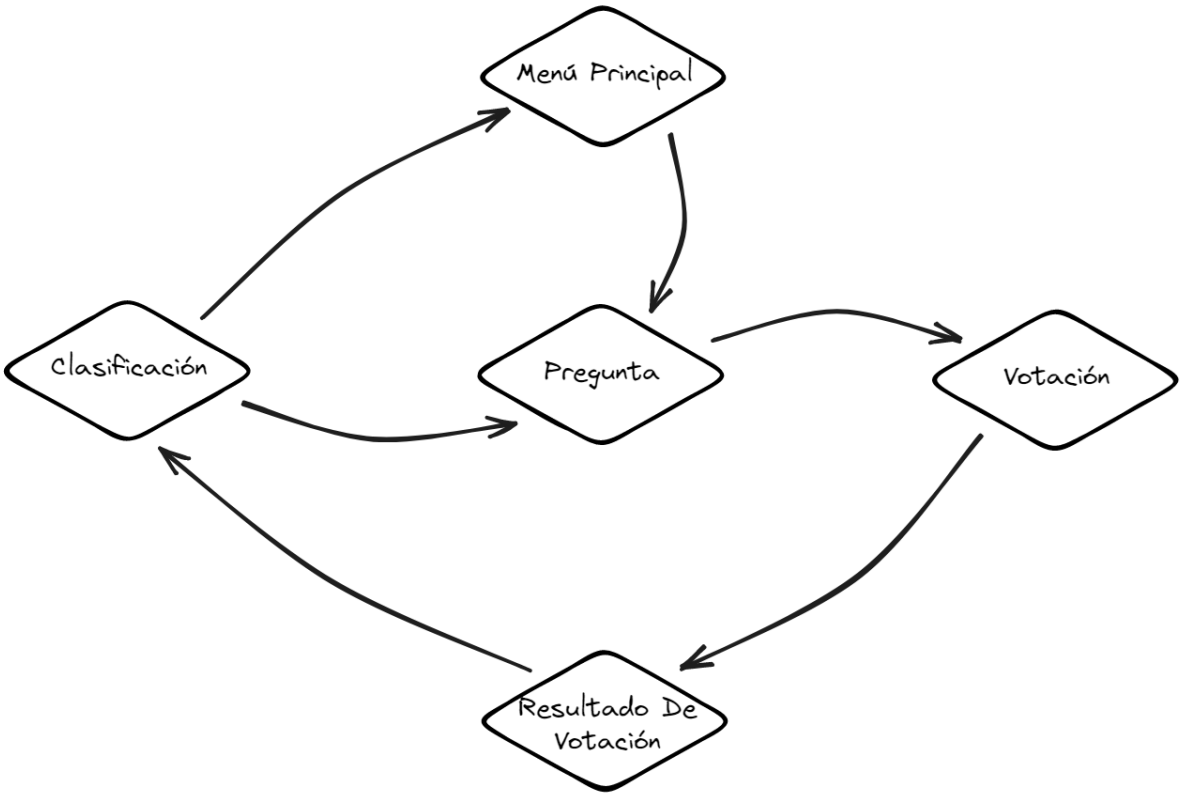


Ilustración 67. Diagrama de navegación del juego Cartas contra la Humanidad de la versión de escritorio

Móvil

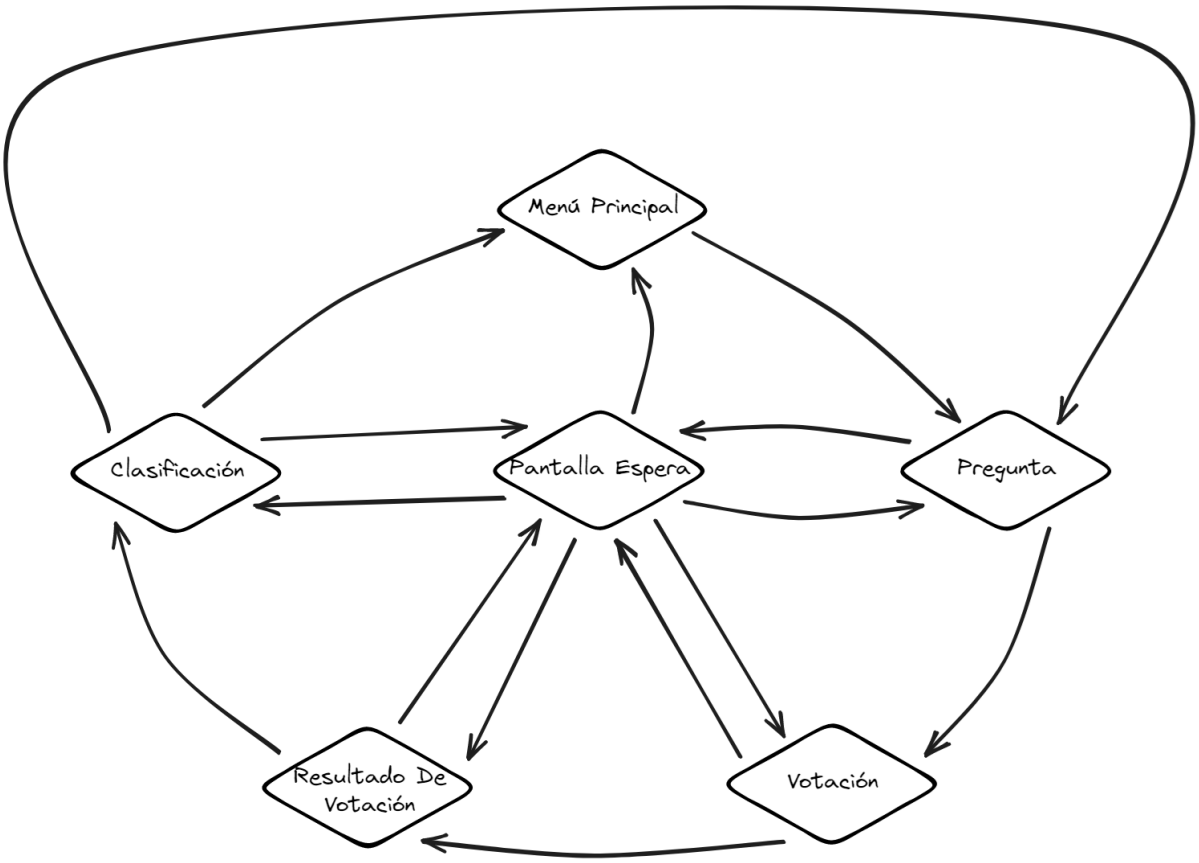


Ilustración 68. Diagrama de navegación del juego Cartas contra la Humanidad de la versión móvil

Gestión de datos

En este apartado se explicará el proceso de gestión de datos de la aplicación, centrándose en los tipos de datos de interés, cómo se almacenan y cómo se manejan.

Se debe diferenciar dos tipos de datos de interés, los permanentes y los volátiles. Los datos permanentes son aquellos que se necesitan siempre en la aplicación, como pueden ser los perfiles de los usuarios, los juegos, los amigos, etc. Los datos volátiles son aquellos que solo son necesarios mientras esté activa la sesión del usuario como las invitaciones a grupos.

Datos Permanentes

Para el almacenamiento de estos tipos de datos, se ha decidido utilizar *SQL Server* de *Azure* como sistema de gestión de base de datos (*Data base management system* - DBMS).

Con el objetivo de simplificar la creación y administración de la base de datos, se ha empleado el enfoque *Code First* en *.NET* (Gorman, 2021). *Code First* permite definir entidades y relaciones directamente en el código de la aplicación, evitando la necesidad de escribir

consultas en la propia base de datos. Esto brinda flexibilidad para evaluar y cambiar el *DBMS* en el futuro sin demasiado esfuerzo.

Se ha dedicado atención individual a la definición de cada una de las entidades, incluyendo la especificación de la clave primaria, las claves foráneas cuando sea necesario, y todas las propiedades con sus respectivos tipos de datos. Además, en el contexto de la base de datos utilizando el enfoque *Code First*, se han establecido relaciones entre las entidades para lograr un mejor control sobre ellas. Asimismo, se han generado índices en aquellas propiedades que lo requieran, con el fin de mejorar el rendimiento del sistema.

```
4   namespace DoubleTapParty.DAL.Models
5   {
        17 referencias
6       public class Comment
7       {
8           [Key]
            4 referencias
9           public Guid IdComment { get; set; }
            5 referencias
10          public string Text { get; set; }
            5 referencias
11          public DateTime Date { get; set; }
12
13          [ForeignKey("User")]
            1 referencia
14          public Guid IdUser { get; set; }
            8 referencias
15          public User User { get; set; }
16
17          [ForeignKey("Game")]
            1 referencia
18          public Guid IdGame { get; set; }
            3 referencias
19          public Game Game { get; set; }
20
21          [ForeignKey("Parent")]
            1 referencia
22          public Guid? IdParent { get; set; }
            1 referencia
23          public Comment? Parent { get; set; }
            1 referencia
24          public ICollection<Comment> Replies { get; set; }
25      }
26  }
```

Ilustración 69. Código de ejemplo de clase del modelo

```
0 referencias
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    #region User

    modelBuilder.Entity<User>()
        .HasIndex(u => u.Nickname)
        .IsUnique();

    modelBuilder.Entity<User>()
        .HasIndex(u => u.Email)
        .IsUnique();

    #endregion

    #region Friendships

    modelBuilder.Entity<Friendship>()
        .HasKey(fr => new { fr.UserId, fr.FriendId });

    modelBuilder.Entity<Friendship>()
        .HasOne<User>(fr => fr.User)
        .WithMany(u => u.Friendships)
        .HasForeignKey(fr => fr.UserId)
        .OnDelete(DeleteBehavior.ClientCascade);


    #endregion

    FriendshipsRequest
```

Ilustración 70. Código de ejemplo de configuración del modelo

Las entidades que constituyen el modelo de datos y cómo se relacionan entre ellas están definidas en el siguiente diagrama:

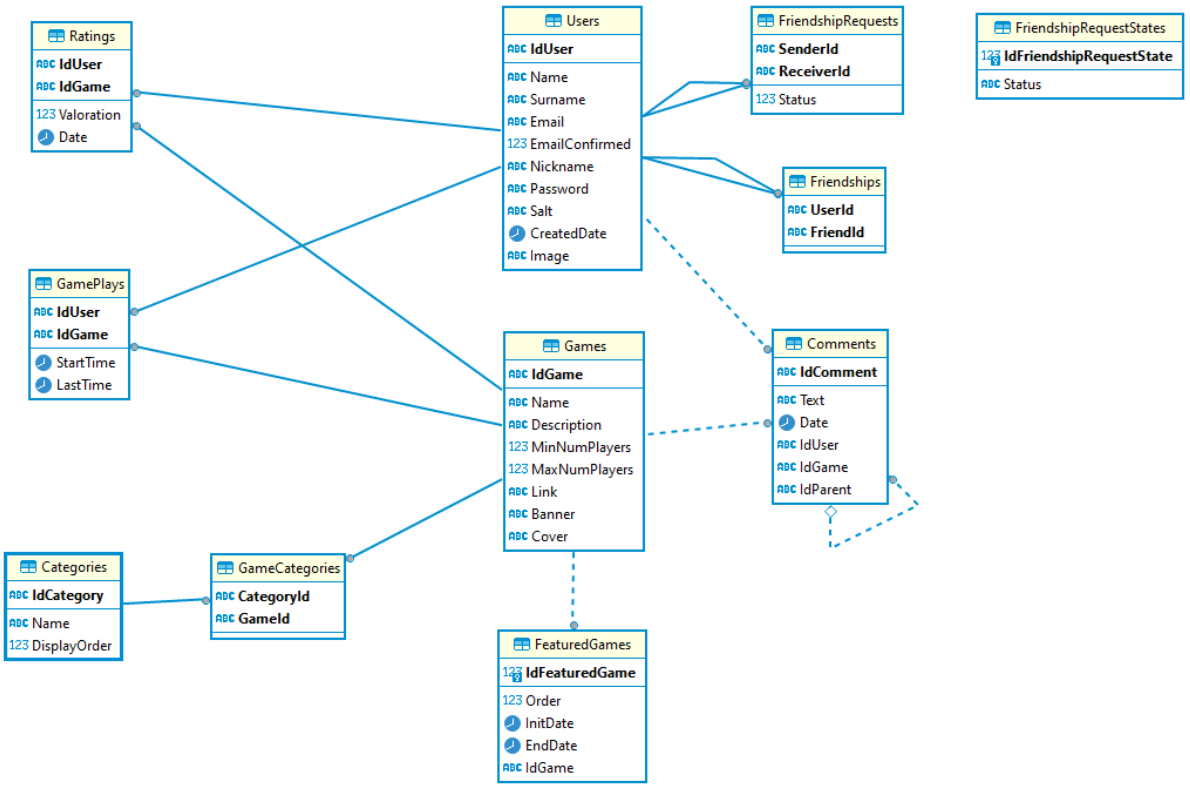


Ilustración 71. Diagrama ER de la base de datos

Datos Volátiles

Para gestionar eficientemente los datos volátiles en la aplicación, se han desarrollado administradores especializados. Estos administradores utilizan *"ConcurrentDictionaries"* para almacenar y gestionar los datos mientras las sesiones de los usuarios están activas en la plataforma. La implementación de estos administradores tiene como objetivo evitar el almacenamiento innecesario de datos temporales en la base de datos principal, lo que a su vez mejora el rendimiento general del sistema. En la aplicación, se pueden identificar tres tipos de administradores encargados de esta tarea.

Administrador de la Aplicación Principal

El administrador de la aplicación principal es el encargado de gestionar la información relevante asociada a la sesión activa del usuario. Este administrador almacena y administra los siguientes datos:

- *Group*: Contiene la clave del grupo al que pertenece el usuario y el id del usuario anfitrión.
- *CurrentGame*: Indica el nombre del juego actual en el que se encuentra el usuario.
- UserState: Indica el estado actual del usuario, ya sea *"Desconectado"*, *"Conectado"*, *"En Grupo"* o *"Jugando"*.
- *InMobile*: Indica si el usuario tiene un dispositivo móvil conectado.
- *InDesktop*: Indica si el usuario tiene un dispositivo de escritorio conectado.
- *GroupInvitations*: Almacena todas las invitaciones a grupos que el usuario ha recibido durante su sesión activa en cualquier dispositivo.

Administrador de los Juegos

El administrador de los juegos gestiona la información necesaria para permitir la comunicación entre los dispositivos de un grupo que se encuentre dentro de un juego. Este administrador almacena y administra los siguientes datos:

- *StartGame*: Fecha de inicio de la sesión actual del juego para cada usuario.
- *EndGame*: Fecha de fin de la sesión actual del juego para cada usuario.
- *IsMobile*: Indica si el usuario tiene un dispositivo móvil conectado.
- *IsDesktop*: Indica si el usuario tiene un dispositivo escritorio conectado.
- *NumTotalDevice*: Número total de dispositivos conectados a un grupo.

Administrador de Cartas Contra la Humanidad

Dentro del juego *Cartas contra la Humanidad* se manejan datos volátiles propios para la gestión del juego, aparte de los datos comentados en el punto anterior para administrar las comunicaciones entre dispositivos:

- *PlayerCardsList*: Listado de las cartas de usuario.
- *QuestionsList*: Listado de las cartas de pregunta.
- *GameState*: Estado del juego

- *DictionaryUserAnswers*: Diccionario con las respuestas de los jugadores de una partida.
- *DictionaryVotes*: Diccionario con los votos de los jugadores de una partida.
- *ListUserCardslist*: Listado de las cartas actuales de cada jugador.
- *UserContinueResultVotation*: Indica si el usuario ha pulsado continuar en la pantalla de votación.
- *UserContinueNextQuestion*: Indica si el usuario ha pulsado continuar en la pantalla de siguiente pregunta.
- *Score*: Puntuación de cada usuario.
- *NumQuestion*: Numero de la pregunta actual.
- *NumQuestionsGame*: Número total de preguntas que tiene una partida.

Implementación

Desarrollo de la Aplicación Web

Desde un primer momento se ha intentado desarrollar un proyecto donde escalar y añadir nuevas funcionalidades sea relativamente sencillo. Para ello, la aplicación principal se ha desarrollado en una arquitectura de tres capas: Capa de presentación (*DoubleTapParty*), Capa de Negocio (*DoubleTapParty.BAL*) y Capa de Datos (*DoubleTapParty.DAL*).

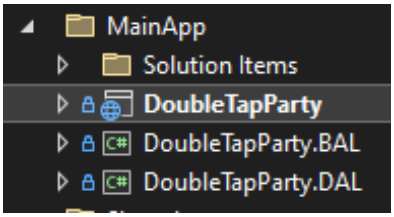


Ilustración 72. Distribución de las capas del proyecto

Capa de Presentación

La capa de presentación es la interfaz de usuario de la aplicación, es la parte en la que los usuarios interactúan directamente. El objetivo principal de esta capa es proporcionar una experiencia visual y funcional correcta, capturar las acciones del usuario y transmitir información con la capa de negocio.

Para el desarrollo de esta capa se ha utilizado el patrón de diseño Modelo-Vista-Controlador (*Model-View-Controller* - MVC) con el fin de separar y organizar la lógica en tres componentes principales: el modelo, la vista y el controlador.

1. Modelo (*Model*)

El modelo representa los datos de la aplicación. Cada vista de la página web requiere de un modelo para su correcto funcionamiento. El modelo no tiene conocimiento de cómo se

representan o se utilizan sus datos, su función principal es garantizar la integridad y consistencia de los datos.

Todos los modelos que se han creado para el correcto funcionamiento de la aplicación se muestran en la imagen siguiente:

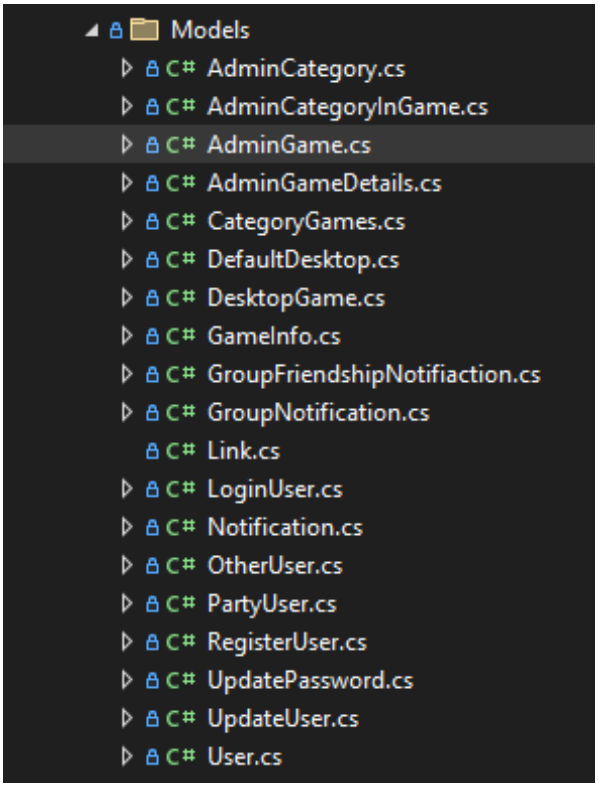


Ilustración 73. Modelos

2. Vista (*View*)

La vista es la parte responsable de la representación visual de los datos y de la interacción con el usuario. Muestra la información al usuario y permite que este interactúe con la aplicación. La vista es la encargada tanto de la representación de los datos como de la disposición de los elementos de la interfaz de usuario, los colores, las fuentes, etc. La vista no realiza ningún procesamiento ni manipulación de los datos, para ello se comunica con el controlador a la hora de realizar acciones o solicitar actualizaciones en el modelo.

Todas las vistas que se han creado para la aplicación se muestran en la imagen siguiente organizadas en diferentes secciones:

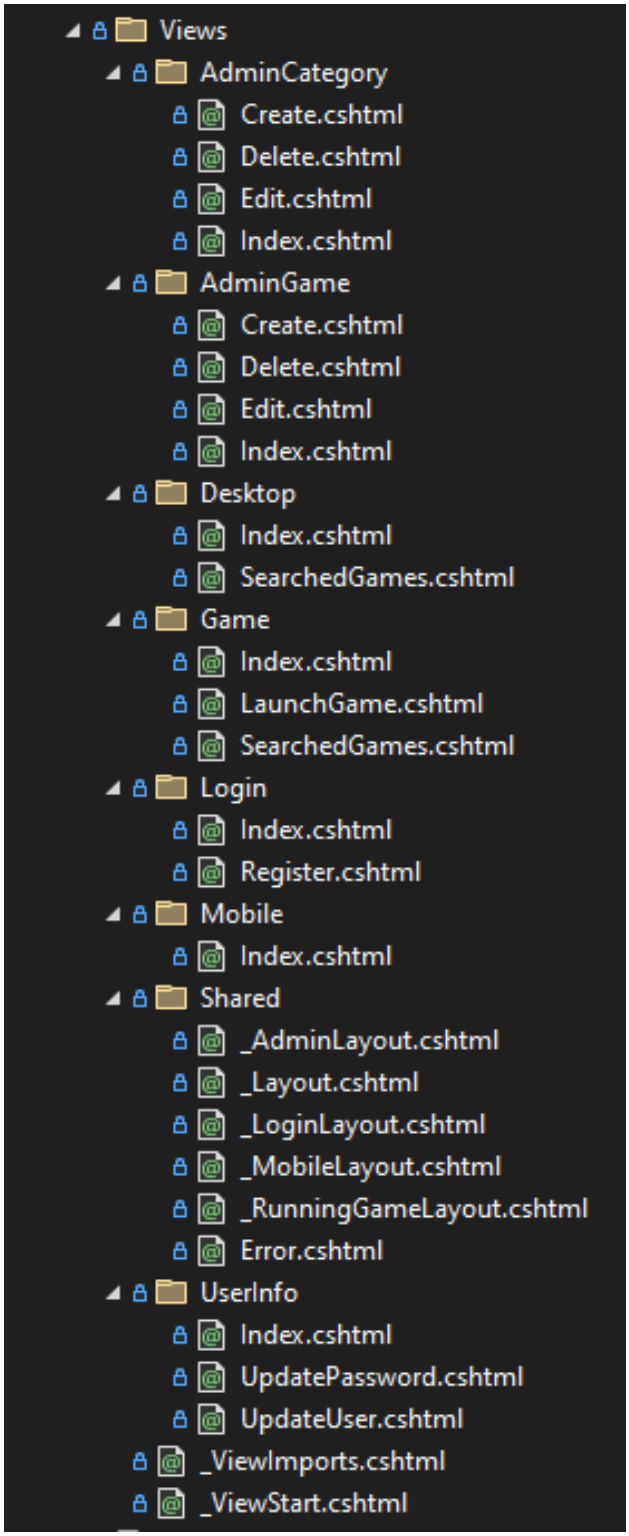


Ilustración 74. Vistas

Una parte a destacar es que todas las vistas permiten tener un diseño predeterminado, de manera que las páginas que tengan una estructura similar pueden basarse en el mismo. Estas estructuras se encuentran en la carpeta *Shared*.

3. Controlador (*Controller*)

El controlador actúa como intermediario entre el modelo y la vista. Es el responsable de manejar las interacciones del usuario y coordinar las acciones entre el modelo y la vista. El controlador recibe las entradas del usuario desde la vista y las procesa, determinando qué acciones deben realizarse en el modelo. Después de actualizar el modelo, puede ordenar el refresco de la página para reflejar los cambios. El controlador también es el encargado de la lógica de la página, dirigiendo las acciones y asegurando que las acciones se ejecuten correctamente.

Se ha generado un controlador por cada una de las secciones de la carpeta *Views* a excepción de *Shared* que no necesita de controlador.

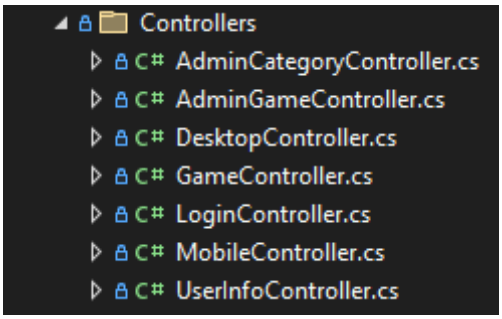


Ilustración 75. Controladores

Capa de Negocio

La capa de negocio, también conocida como capa lógica o capa de la aplicación, se encuentra en el centro de la arquitectura en tres capas. Esta es la capa encargada del procesamiento de datos, haciendo de intermediario entre la capa de presentación y la capa de datos.

En esta aplicación se van a gestionar tres tipos de datos diferentes, usuarios, categorías y juegos, por lo que se han generado una clase de negocio para cada tipo. De esta forma se consigue representar las reglas y las operaciones de negocio para cada entidad.

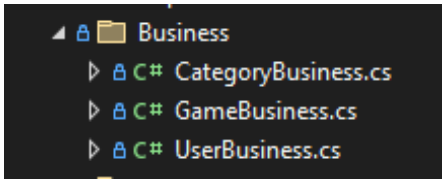


Ilustración 76. Clases de negocio

Para lograr una separación adecuada de responsabilidades y facilitar la inyección de dependencias, se han creado las interfaces correspondientes a cada una de las clases de negocio. Estas interfaces definen todos los métodos y operaciones necesarias para manipular los datos de cada tipo de dato. De esta manera se consigue una mayor flexibilidad y

extensibilidad en el código ya que permite cambiar o agregar implementaciones de las clases de negocio sin afectar directamente a la presentación.

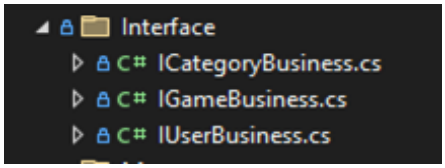


Ilustración 77. Interfaces de negocio

En la capa de negocio también se han creado un apartado de servicios para que estos sean accesibles en toda la aplicación. Se ha desarrollado un servicio de ficheros (*FileService*) que se encarga de subir, actualizar y eliminar ficheros en la nube; y un servicio de seguridad (*SecurityService*) que se encarga de cifrar contraseñas.

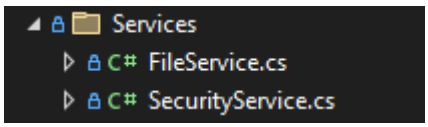


Ilustración 78. Servicios

El manejador de conexiones de usuarios también se encuentra en esta capa, pero este apartado se desarrollará en **Desarrollo de Comunicaciones entre Dispositivos**.

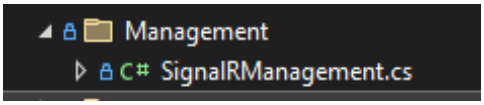


Ilustración 79. Gestor de SignalR

Capa de Datos

La capa de datos es la responsable de gestionar el almacenamiento y acceso a los datos de la aplicación.

Tal y como se explica en el apartado de **Datos Permanentes**, Se ha empleado el enfoque *Code First* para la gestión de los datos, generando todos los modelos con todos los datos necesarios y configurando el contexto de la base de datos para montarla.

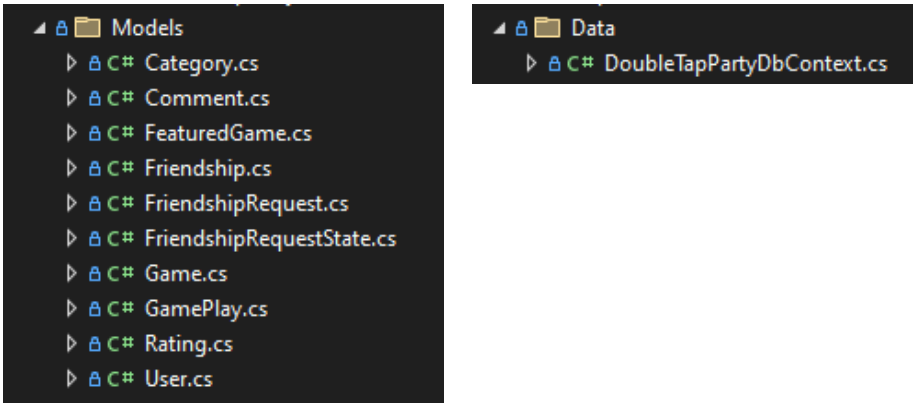


Ilustración 80. Modelos y Contexto de la base de datos

Para facilitar el acceso a datos y garantizar la separación de responsabilidades se ha desarrollado un sistema de repositorios.

Se ha implementado un patrón de diseño llamado Unidad de Trabajo (*Unit Of Work*) para agrupar varias operaciones relacionadas en una funcionalidad común, permitiendo realizar transacciones a la base de datos y mantener coherencia de los datos en un contexto específico. Esta capa, además, encapsula todos los repositorios proporcionando una funcionalidad que devuelve el repositorio buscado en función del tipo de dato que se introduce, evitando de esta forma la inyección de dependencias de todos los demás repositorios.

Se ha generado un repositorio por defecto para proporcionar las funcionalidades básicas de acceso a datos, esto incluye las operaciones de obtener todos los elementos, obtener elemento por ID, crear, actualizar y eliminar. El resto de los repositorios heredan de él, añadiendo las operaciones necesarias para cada entidad en particular, permitiendo adaptar los repositorios a las necesidades específicas y evitando la duplicidad de código.

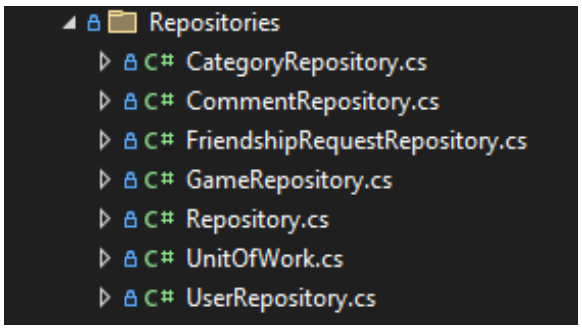


Ilustración 81. Repositorios

Para lograr una separación adecuada de responsabilidades y facilitar la inyección de dependencias, se han creado las interfaces correspondientes a cada uno de los repositorios. Estas interfaces definen todos los métodos y operaciones necesarias para manipular los datos de cada tipo de dato. De esta manera se consigue una mayor flexibilidad y

extensibilidad en el código ya que permite cambiar o agregar operaciones en los repositorios sin afectar directamente al resto de código que depende de ellos.

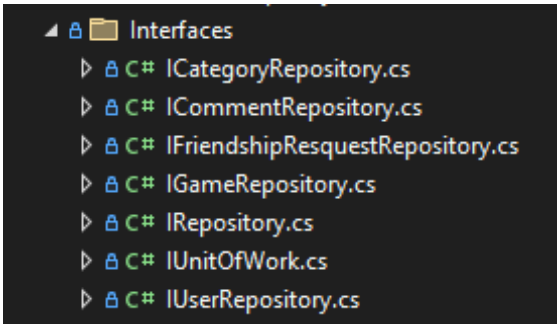


Ilustración 82. Interfaces de los repositorios

Desarrollo de Videojuegos

GameHub

Como ya se ha descrito en el apartado de arquitectura de la aplicación, cada juego se despliega en una aplicación virtual dentro del *AppService* de la aplicación principal, para poder realizar este despliegue cada juego se ha desarrollado como un proyecto en *.NET* que posteriormente se ha publicado en una aplicación virtual.

La intención a la hora de desarrollar el apartado de alojamiento de los juegos dentro de la aplicación fue que, aun siendo independiente todo el código propio de la aplicación fuera compartido y que pudiese adaptarse al código propio de cada juego introducido por los equipos de desarrollo externo.

Bajo esta idea, se ha desarrollado la clase abstracta *AbstrGameHub* y la clase estática *GameHubManager*. Para centralizar códigos y evitar problemas por la duplicidad de códigos, estas clases, que son utilizadas por todos los proyectos *.NET* de cada juego, se han incorporado dentro de la librería de Clases *SharedGameClass*. Dentro de esta librería se encuentran también las clases con los conectores a *APIs* externas y las clases de los modelos usados en la comunicación entre los juegos y la aplicación principal.

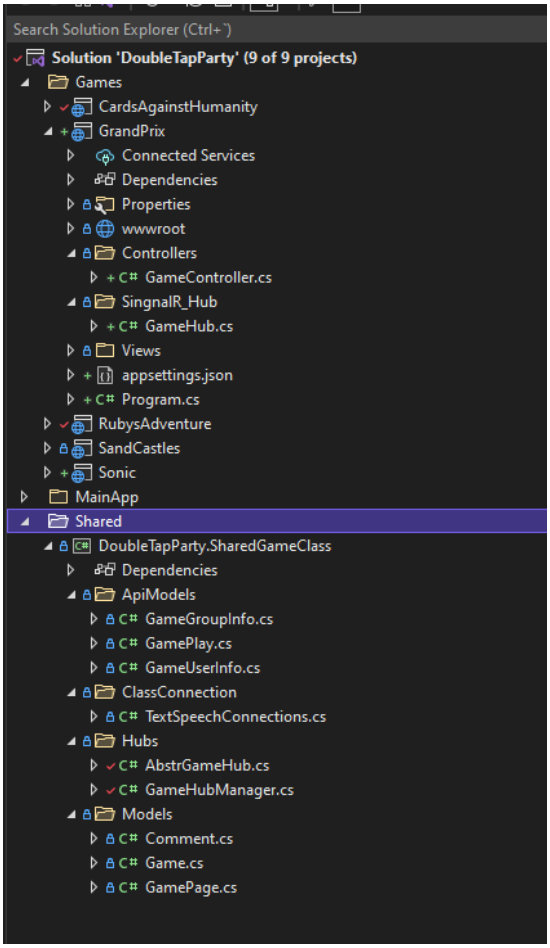


Ilustración 83. Visualización de la estructura del proyecto de un juego y de la librería de clases SharedGameClass.

Tanto la clase *AbstrGameHub*, como la clase *GameHubManager* se encargan de manejar las conexiones de los dispositivos a las vistas de los juegos.

La clase *AbstrGameHub* hereda de la clase *Hub* de *SignalR* y contiene los métodos de comunicación con la parte del cliente. En ella, se encuentran dos métodos sobrescritos de la clase *Hub*: *OnConnectedAsync* y *OnDisconnectedAsync*. Estas funciones son llamadas cuando se inicia y finaliza una conexión desde el cliente. Además, la clase *AbstrGameHub* contiene las llamadas a los métodos del *GameHubManager* y maneja las concurrencias entre conexiones y desconexiones de dispositivos de un mismo grupo y usuario.

También contiene un listado de funciones para manejar las pulsaciones de los botones estándar, el control del audio del juego y funciones virtuales vacías para manejar los cambios de estado de la conexión del juego específicamente para cada uno.

La clase *GameHubManager* contiene los métodos de gestión de conexiones y grupos en cada juego y las llamadas a las *APIs* para la comunicación con la aplicación principal.

Para cada proyecto de juego se dispone de una clase *GameHub* que hereda de la clase *AbstrGameHub*. Esta clase podrá ser modificada en función de las necesidades de cada

juego. En los ejemplos incorporados para la memoria, el juego *CardsAgainstHumanity* se ha utilizado como ejemplo de adaptación de esta clase para incorporar la capacidad de juego multijugador en la aplicación. El resto de los juegos son juegos individuales y no han necesitado adaptar esta clase.

Controladores estándar de juegos

Para incorporar el uso de controladores estándar dentro de los juegos en la parte del dispositivo móvil, únicamente será necesario aplicar el layout *_GameControllerMobileLayout* en el fichero *Mobile.cshtml* de la vista del juego y para elegir el tipo de controlador será necesario modificar el *Html.RenderPartial* por *CrossPadController* o *JoystickController*.

Las funciones *JavaScript* de conexión y de invocación a los métodos del *GameHub*, se encuentran en un directorio común para todos los juegos dentro de la aplicación principal. En estas funciones se encuentra el inicio y fin de las pulsaciones de los botones, las de los cambios de estado dentro del juego y las relativas al audio.
Dentro del código exclusivo para el controlador del *joystick* cabe destacar la función *makeDraggable* (Collingridge) que controla el movimiento de las etiquetas "*svg*" que componen el joystick y llama a la función *MoveJoystick* que envía el movimiento al *GameHub*.

Para evitar que la tasa de refresco del evento *JavaScript "touchmove"* sature de envíos al *GameHub*, se controla desde esta función las llamadas a *MoveJoystick*, de manera que únicamente se llama cada 60 ms enviando el último posicionamiento del *joystick*.

La incorporación desde el lado del dispositivo de escritorio es similar, aplicando el layout *_GameControllerDesktopLayout* en la vista de escritorio cargará los códigos requeridos para permitir el funcionamiento. Lo único que será necesario es la creación e incorporación de un fichero *JavaScript* con el nombre *[NombreJuego]Desktop.js*, que contendrá las funciones de simulación de presionar y soltar las teclas y que deberá ser editado para seleccionar las teclas que espere recibir el juego.

```
document.dispatchEvent(new KeyboardEvent('keyup', {
    key: "Arrow Down",
    keyCode: 40,
    code: "KeyArrow Down",
    which: 40,
    shiftKey: false,
    ctrlKey: false,
    metaKey: false
}));
```

Ilustración 84. Ejemplo del código de simulación de pulsación de teclas

Controlador para juego *WebGL*

La simulación de presión detallada en el punto anterior solo es válida para los juegos *HTML5*, en el juego *WebGL* desarrollado con *Unity*, los eventos de simulación de presión no tienen ningún efecto sobre el juego.
Para poder realizar la comunicación entre el dispositivo móvil y el juego se tuvo que añadir un enlace entre el código *JavaScript* del navegador y el código del juego *Unity* (*Unity User Manual (2019.4 LTS) - Unity Manual*). La creación de este enlace se compone de tres archivos. Dos ficheros que se encuentran en el proyecto *Unity JavaScriptHook.cs* y *jsfunctions.jslib*, y el fichero *[NombreJuego]Desktop.js* situado dentro del proyecto *.NET*.

El fichero *jsfunctions.jslib* contiene los métodos que contiene las funciones *JavaScript* que se ejecutan en el navegador y son llamadas desde la aplicación *WebGL*, en este caso únicamente se dispone de una función que llama a un método de *SignalR* que envía el estado del audio del juego al *GameHub*, pasando como parámetro el estado.

```
1  mergeInto(LibraryManager.library, {
2
3      sendAudioState: function (state) {
4          SendAudioStateToMobile(state);
5      },
6
7  });
```

Ilustración 85. Código de llamada a funciones JavaScript desde Unity

En el fichero *JavaScriptHook.cs* se encuentran los métodos llamados desde los códigos *JavaScript* del navegador, encargados de realizar las acciones de mover al personaje o simular la presión de los botones o cambiar el estado del audio.

Dentro del fichero se encuentran las siguientes líneas de código encargadas de declarar el método externo *sendAudioState*, situado en el fichero *jsfunctions.jslib*, La anotación *[DllImport("__Internal")]* indica que este método es una llamada a una función externa definida en el entorno de ejecución donde se está ejecutando el proyecto *WebGL*. El atributo *__Internal* se utiliza en *WebGL* para indicar que la función se encuentra en el entorno del navegador o en otro contexto externo. Esta declaración solo se ejecuta si se cumplen las condiciones de que el proyecto se está ejecutando para *WebGL* y no en el editor de *Unity*.

```
10      #if UNITY_WEBGL && !UNITY_EDITOR
11          [DllImport("__Internal")]
12          private static extern void sendAudioState(bool state);
13      #endif
14
```

Ilustración 86. Declaración de funciones externas únicamente desde Unity WebGL

En el siguiente fragmento de código se muestra la llamada a la función del fichero *jsfunctions.jslib*, con las misma condición de declaración del método externo mostrado anteriormente.

```
0 referencias
public void RequestAudioState()
{
    #if UNITY_WEBGL && !UNITY_EDITOR
        sendAudioState(AudioListener.pause);
    #endif
}
```

Ilustración 87. Llamada a funciones externas únicamente desde WebGL

El código *[NombreJuego]Desktop.js* incluye las llamadas a los métodos de *JavaScriptHook.cs*. En la imagen se observa el uso de la sentencia *unityInstance.SendMessage* para llamar a los métodos *JoystickMove* y *PressA*. Es relevante tener en cuenta que esta llamada solo permite enviar un único parámetro. Si se necesita enviar más de un valor, es necesario crear y enviar un archivo .json que contenga el conjunto de variables a transmitir.

```
//Funcion stopPressRight
function unityJoystickMovement(x, y) {
    var obj = new Object();
    obj.x = x;
    obj.y = y;
    var movementJson = JSON.stringify(obj);
    unityInstance.SendMessage('JavascriptHook', 'JoystickMove', movementJson);
};

//Funcion onPressA
function PressedA() {
    console.log("PASAAAA A" + conditionA);
    if (conditionA) {
        console.log("Ejecuta");
        unityInstance.SendMessage('JavascriptHook', 'PressA');
    }
};
```

Ilustración 88. Llamadas a métodos Unity desde JavaScript

Cards Against Humanity

El desarrollo del juego *Cards Against Humanity* comparte el modo de implementación del resto de juegos incorporados en la aplicación, con la diferencia de que al ser un juego multijugador el *GameHub* ha sido adaptado a las necesidades del juego, añadiendo métodos que controlan el estado de la partida y de los jugadores y que realizan las llamadas a las funciones JavaScript en función de ellos.

También se ha creado la clase *CardsAgainstHumanityManager*, para el manejo de la información de las partidas, el control del número de partidas simultáneas, el control de concurrencia entre jugadores de una misma partida y el acceso a las *APIs*.

Dentro de la clase *CardsAgainstHumanityManager* cabe destacar el código de barajado de *Fisher-Yates* (Fisher & Yates, 1963, 26-27), un algoritmo utilizado para ordenar de manera aleatoria los elementos de una lista o matriz. Se utiliza a la hora de reordenar la lista de respuestas y preguntas a repartir entre los jugadores y también para reordenar las respuestas a una pregunta, antes de enviarla a las aplicaciones de escritorio para mostrarla.

El algoritmo opera de la siguiente manera:

1. Comienza con una lista de elementos que deseas mezclar.
2. Comienza desde el último elemento de la lista y avanza hacia el primero.
3. En cada paso, selecciona un elemento aleatorio de los elementos que aún no han sido seleccionados y lo intercambia con el elemento actual.
4. Continúa avanzando hacia el primer elemento hasta que hayas llegado al principio de la lista.

```
2 references
public void Shuffle(List<int> list)
{
    //Fisher-Yates shuffle:
    RNGCryptoServiceProvider provider = new RNGCryptoServiceProvider();
    int n = list.Count();
    byte[] _uint32Buffer = new byte[4];
    Random rnd = new Random();
    while (n > 0)
    {
        provider.GetBytes(_uint32Buffer);
        UInt32 rand = BitConverter.ToUInt32(_uint32Buffer, 0);
        Int64 max = (1 + (Int64)UInt32.MaxValue);
        Int64 remainder = max % list.Count();
        if (rand < max - remainder)
        {
            n--;
            int k = (Int32)((rand % list.Count()));
            int value = list[k];
            list[k] = list[n];
            list[n] = value;
        }
    }
}
```

Ilustración 89. Algoritmo de Fisher-Yates

Y también cabe comentar el código de la llamada a la *API* de *TextToSpeech* de *Azure*. Este método será llamado siempre que se quiera reproducir un mensaje de texto y generará un token utilizado para llamar a la *API* desde *JavaScript*. Este token debe de ser generado siempre que se necesite crear un audio, ya que tiene una corta duración. También debe de

llamarse desde el servidor, ya que para generar el token necesita acceder a una clave propia de la cuenta *Azure* de la aplicación que no debe compartirse.

```
/// <summary>
/// Crea un objeto TextSpeechInfo para reproducir el texto de las respuestas en el desktop
/// </summary>
/// <param name="httpClient"></param>
/// <param name="texto"></param>
/// <param name="url"></param>
/// <param name="key"></param>
/// <returns></returns>
1 reference
public static async Task<TextSpeechInfo> GetInfoSpeechData(HttpClient httpClient, string texto, string url, string key)
{
    var token = "Error";
    var request = new HttpRequestMessage
    {
        Method = HttpMethod.Post,
        RequestUri = new Uri(url),
        Content = new StringContent("application/json")
    };
    httpClient.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", key);

    HttpResponseMessage response = await httpClient.SendAsync(request);

    if (response.IsSuccessStatusCode)
    {
        HttpContent responseContent = response.Content;
        var reader = new StreamReader(await responseContent.ReadAsStreamAsync());
        token = reader.ReadToEndAsync().Result;
    }

    return new TextSpeechInfo(token,texto);
}
```

Ilustración 90. Llamada a la API TextToSpeech para la obtención del token

Una vez generada se enviará mediante *SignalR* al cliente que realizará la llamada por el token a la aplicación y reproducirá el sonido:

```
//Solicita informacion la info al servidor para reproducir las respuestas
connection.on("SendTextSpeechInfo", function (textSpeechData) {
    var objClassification = JSON.parse(JSON.stringify(textSpeechData));
    textToSpeech(objClassification.token, objClassification.url, objClassification.text, objClassification.speecher);
});

//Solicita el sonido y lo reproduce
async function textToSpeech(accessToken, ttsEndpoint, text, voice) {
    audioPlayer = document.getElementById('audio-player');
    const userAgent = navigator.userAgent;
    console.log(userAgent);

    const response = await fetch(`${ttsEndpoint}`, {
        method: 'POST',
        headers: {
            'Authorization': `Bearer ${accessToken}`,
            'Content-Type': 'application/ssml+xml',
            'X-Microsoft-OutputFormat': 'audio-16khz-32kbitrate-mono-mp3',
            'User-Agent': `${userAgent}`,
        },
        body: `<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='en-US'>
            <voice name='${voice}'>${text}</voice>
          </speak>`,
    });

    if (!response.ok) {
        throw new Error(`Text to speech failed with status code ${response.status}.`);
    }
    const blob = await response.blob();

    const url = URL.createObjectURL(blob);
    audioPlayer.src = url;
    audioPlayer.play();
    return url;
}
```

Ilustración 91. Llamada a la API TextToSpeech para la obtención de audio

Desarrollos Comunes

Desarrollo Comunicaciones entre Dispositivos

Signalr

Para proporcionar una funcionalidad de comunicación en tiempo real entre usuarios y entre el usuario y la plataforma, se ha utilizado la librería *SignalR*. (Ingebrigtsen, 2013)

Para lograr esto, se ha creado un *Hub* general en la aplicación web y *Hubs* individuales para cada juego que contienen todas las llamadas necesarias para utilizar todas las características de la plataforma. El objetivo de estas llamadas es controlar todos los eventos posibles, como las conexiones y desconexiones de los usuarios y en qué dispositivos se realizan, las entradas de texto para lograr una sincronización entre dispositivos, los eventos de enviar comentarios y de abrir o cerrar un juego.

Como se explicó en la sección de *"Gestión de Datos"* en *"Datos volátiles"*, se han creado gestores de datos para las aplicaciones del proyecto encargadas de gestionar los datos de los usuarios. De esta manera, para comunicar cualquier información entre usuarios, incluyendo el propio usuario en otros dispositivos, ya se tiene almacenado el identificador único de sus sesiones activas, lo que permite enviar información de manera prácticamente instantánea.

De todos los eventos de los diferentes Hubs, es importante destacar la gestión de conexiones y desconexiones de los usuarios, ya que esto presentaba problemas al cambiar de vista. Por defecto, cada vez que se refresca la página o se cambia de vista se produce un evento de desconexión y conexión que provocaba la eliminación de los datos temporales del usuario.

Para solucionar este problema, se creó un diccionario concurrente de semáforos, de modo que cada dispositivo conectado pueda gestionar si se trata de una desconexión real o simplemente de un cambio de vista o refresco de la página. Para lograr este control, los semáforos no tienen ningún recurso asociado. El evento de desconexión espera durante un tiempo fijo a que el semáforo esté liberado. Si el tiempo de espera se supera y el semáforo sigue bloqueado, significa que ha habido una desconexión del usuario, por lo que se procede a eliminar todos sus datos temporales. El evento de conexión libera el semáforo. De esta manera, al ejecutarse ambos eventos siempre que haya un refresco o carga de una nueva vista, se puede controlar de manera precisa si se trata de una desconexión total o no.

```
/// <summary>
/// Gestiona las desconexiones de los usuarios
/// </summary>
/// <param name="exception"></param>
/// <returns></returns>
0 referencias
public override async Task OnDisconnectedAsync(Exception exception)
{
    var userSemKey = GetUserSemaphoreKey();

    if (_userSemaphores.TryGetValue(userSemKey, out var semKey))
    {
        //Si se trata de una reconexión
        if (await semKey.WaitAsync(_RECONNECTION_TIME)) { }
        //Si se trata de una desconexión
        else
        {
            var deviceType = Context.User.Claims.FirstOrDefault(x => x.Type == ClaimTypes.Role).Value == "Desktop"

            await SignalRManagement.DisconnectUser(Context.UserIdentifier, deviceType);

            _userSemaphores.TryRemove(userSemKey, out var sem);

            var group = await SignalRManagement.GetUserGroup(Context.UserIdentifier);
            var usersInGroup = await SignalRManagement.GetAllUsersIdInGroup(group);
            if (usersInGroup != null)
            {
                await Clients.Users(usersInGroup).SendAsync("UserDeviceDisconnected", Context.UserIdentifier);
            }
        }
    }
}
```

Ilustración 92. Función de desconexión de SignalR

```
/// <summary>
/// Gestiona las conexiones de los usuarios
/// </summary>
/// <returns></returns>
0 referencias
public override async Task OnConnectedAsync()
{
    var deviceType = Context.User.Claims.FirstOrDefault(x => x.Type == ClaimTypes.Role).Value == "Desktop" ? SignalRM

    var userSemKey = GetUserSemaphoreKey();

    //Si ya está conectado
    if (_userSemaphores.TryGetValue(userSemKey, out var semKey))
    {
        semKey.Release();
        //var group =await SignalRManagement.GetUserGroup(Context.UserIdentifier);
        //JoinInGroup(group);
    }
    //Si no está conectado
    else
    {
        //await ResetNotifications();
        _userSemaphores.TryAdd(userSemKey, new SemaphoreSlim(0));

        await SignalRManagement.ConectUser(Context.UserIdentifier, deviceType);

        //await Clients.Others.SendAsync("UserConnected", Context.UserIdentifier);

        var group = await SignalRManagement.GetUserGroup(Context.UserIdentifier);
        var usersInGroup = await SignalRManagement.GetAllUsersIdInGroup(group);
        if (usersInGroup != null)
        {
            await Clients.Users(usersInGroup).SendAsync("UserDeviceConnected", Context.UserIdentifier);
        }
    }

}
```

Ilustración 93. Función de conexión de SignalR

APIs

En numerosas vistas de la aplicación, existen datos que pueden actualizarse mientras el usuario se encuentra en ellas. Sin embargo, si esta actualización se realiza desde el controlador de la vista, provoca un refresco en la página que puede afectar negativamente la experiencia del usuario.

Para resolver este problema, se han creado tres *APIs* que permiten realizar llamadas desde *JavaScript* para actualizar los datos sin necesidad de refrescar la página. Estas *APIs* se encargan de añadir, eliminar o modificar información según sea necesario después de recibir la actualización.

Se han desarrollado tres *APIs* para distribuir las llamadas y clasificarlas según el tipo de información que manejan. De esta manera, se cuenta con una *API* dedicada a proporcionar

información sobre los juegos, otra *API* encargada de brindar datos relacionados con los usuarios y una tercera *API* específicamente diseñada para validar las entradas de formularios.

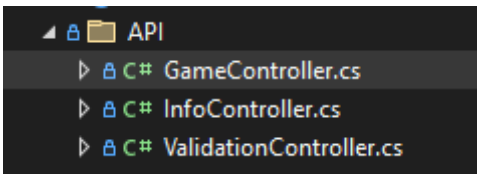


Ilustración 94. APIs

Es importante resaltar que tanto la *API* de juegos como la *API* de información de usuarios están protegidas mediante la cookie de autenticación. Esto significa que solo se puede acceder a esos datos si el usuario ha iniciado sesión en la página web. De esta manera, se garantiza que únicamente los usuarios autorizados puedan acceder y manipular la información relacionada con los juegos y los datos de los usuarios.

Despliegue y tecnologías

En el siguiente punto se detallarán los métodos de implementación y configuración de las tecnologías utilizadas en el proyecto.

Configuración de los proyectos de juegos

La idea fue buscar el modo de publicación de los proyectos asociados a los juegos conservando la misma dirección de dominio.

La primera opción que se encontró fue la de publicar los juegos en otros *AppServices*, pero tenía múltiples limitaciones. La versión gratuita de *Azure* utilizada únicamente permitía almacenar diez *AppServices* distintos y la dirección web de acceso era diferente para cada uno de ellos. Para solventar el problema de las URLs distintas se planteó la idea de obtener un dominio web propio y enlazar a los *AppServices* de los juegos como subdominios del principal. Pero esto también supuso problemas explicados en el punto *"Dominio personalizado al servicio en Azure*" de este mismo apartado.

Por lo que la solución encontrada finalmente fue crear directorios virtuales dentro del *AppService* principal. Para ello se debe acceder a la pestaña *Path mapping* del apartado *Configuration* del *AppService* y en el punto *Virtual Applications and directories* crear tantos directorios virtuales como juegos a publicar.

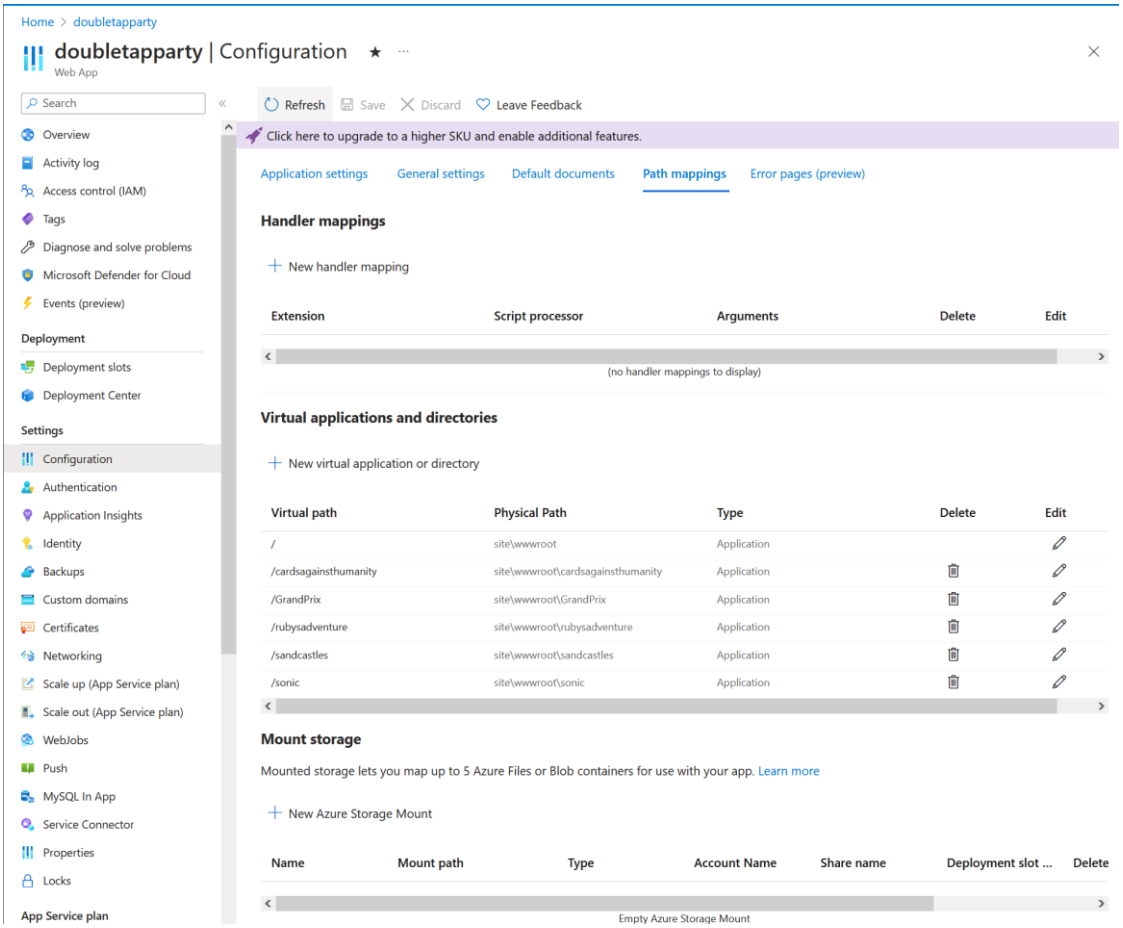


Ilustración 95. Configuración del servidor - Directorios virtuales

Esto se traduce en el servidor como un directorio en la carpeta *wwwroot* de la aplicación, donde se publican las aplicaciones de los juegos. Accediendo a ellos como si fueran un ruta dentro del dominio principal *"https://[DominioWeb].net/[Nombre del juego]"*.

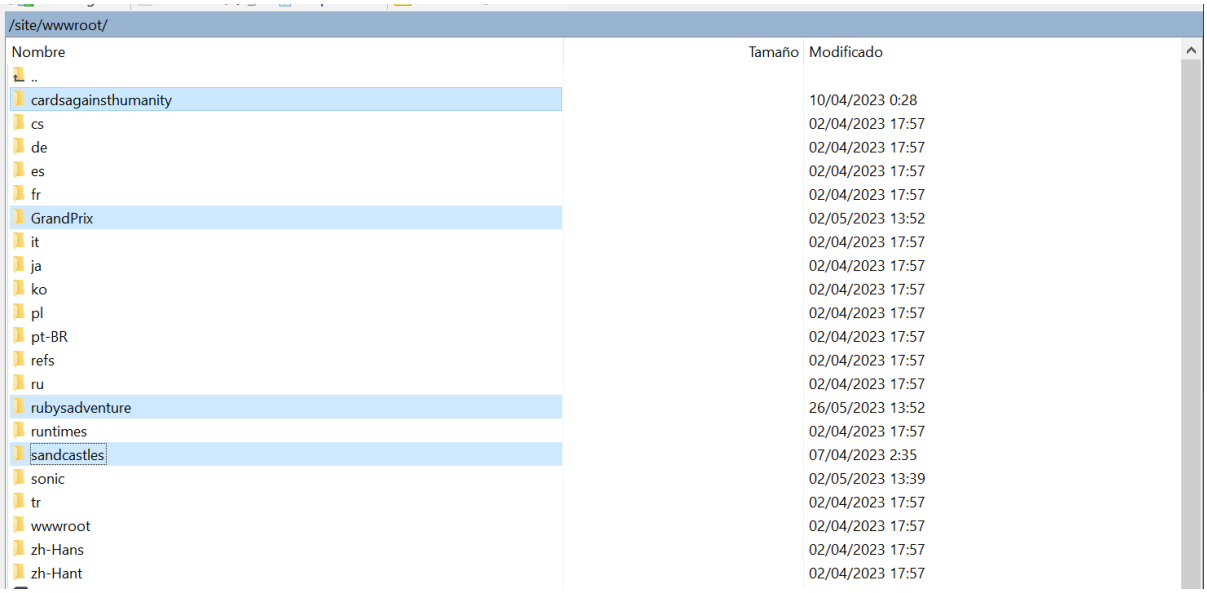


Ilustración 96. Directorio wwwroot

Tras realizar la publicación del proyecto sobre el directorio virtual, fue necesario modificar el archivo *web.config* del proyecto, cambiando el valor por defecto *AspNetCoreModuleV2* del atributo *modules* del elemento *add* por el valor *AspNetCoreModule*, para permitir acceder a los proyectos publicados en los directorios virtuales.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <handlers>
        <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModule" resourceType="Unspecified"
        />
      </handlers>
      <aspNetCore processPath="dotnet" arguments=".\RubysAdventure.dll" stdoutLogEnabled="false"
      stdoutLogFile="\\?\%home%\LogFiles\stdout" hostingModel="inprocess" />
    </system.webServer>
  </location>
</configuration>
<!--ProjectGuid: 9f4c79d3-468e-44e7-87e3-492c719fb84c-->
```

Ilustración 97. Web.config de un proyecto por defecto

Sistema de inicio de sesión único

El proyecto principal cuenta con un servicio de autenticación, con el que impide la navegación a las vistas en las que no cuenten con autorización para conexiones a la web en las que no haya habido un inicio de sesión. Tras crear los directorios virtuales y publicar los servicios web sobre ellos, se comprobó que si a las vistas se le añadía la necesidad de estar autorizados era imposible acceder aun cuando se hubiera logueado desde la aplicación principal.

En lugar de obligar a iniciar sesión de nuevo para acceder a las vistas de los juegos decidió solucionar este problema configurando Sistema de inicio de sesión único (*Single Sign-On* - SSO) que es un sistema de autenticación que permite a los usuarios acceder a múltiples aplicaciones o servicios digitales utilizando un único conjunto de credenciales de inicio de sesión.

El sistema de *SSO* escogido fue el basado en *Cookies*, para ello se configuró el sistema de protección de datos para conservar las claves en un contenedor específico en *Azure Blob Storage*, se definieron un nombre de nombre de cookie *DoubleTapPartyCookie* y se redirigió a la ruta *"/login"* en caso de no haber iniciado sesión. Estas configuraciones se llevaron a cabo tanto en el proyecto principal como en los proyectos de los juegos de la siguiente manera:

```
//coockie authentication

BlobContainerClient container = new BlobContainerClient(
    builder.Configuration.GetValue<string>("BlobStorageConnection:ConnectionString"), "browsercookies");
await container.CreateIfNotExistsAsync();

BlobClient blobClient = container.GetBlobClient("keys.xml");

builder.Services.AddDataProtection()
    .PersistKeysToAzureBlobStorage(blobClient)
    //.PersistKeysToFileSystem(new DirectoryInfo(@"C:\temp\keys"))
    .SetApplicationName("DoubleTapParty");

builder.Services.AddAuthorization();
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
.AddCookie(options =>
{
    options.Cookie.Name = "DoubleTapPartyCookie";
    options.LoginPath = new PathString("/Login");
    options.ExpireTimeSpan = TimeSpan.FromMinutes(20);
    options.AccessDeniedPath = "/Forbidden/";
    options.LogoutPath = "/Account/Logout";
});


//SignalR
builder.Services.AddSignalR();
```

Ilustración 98. Configuración del sistema de protección de datos de la aplicación principal

```
//coockie authentication

BlobContainerClient container = new BlobContainerClient(
    builder.Configuration.GetValue<string>("BlobStorageConnection:ConnectionString"), "browsercookies");
await container.CreateIfNotExistsAsync();

BlobClient blobClient = container.GetBlobClient("keys.xml");

builder.Services.AddDataProtection()
    .PersistKeysToAzureBlobStorage(blobClient)
    //.PersistKeysToFileSystem(new DirectoryInfo(@"C:\temp\keys"))
    .SetApplicationName("DoubleTapParty");

builder.Services.AddAuthorization();
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
.AddCookie(options =>
{
    options.Cookie.Name = "DoubleTapPartyCookie";
    options.Events = new CookieAuthenticationEvents()
    {
        OnRedirectToLogin = (context) =>
        {
            context.HttpContext.Response.Redirect($"/Login");
            return Task.CompletedTask;
        }
    };
    options.ExpireTimeSpan = TimeSpan.FromMinutes(20);
    //options.AccessDeniedPath = "/Forbidden/";
    //options.LogoutPath = "/Account/Logout";
});
```

Ilustración 99. Configuración del sistema de protección de datos de la aplicación móvil

En las siguientes imágenes se puede observar como la *cookie* con nombre *DoubleTapPartyCookie*, se mantiene durante la navegación por la web, tanto en la aplicación principal como en los juegos:

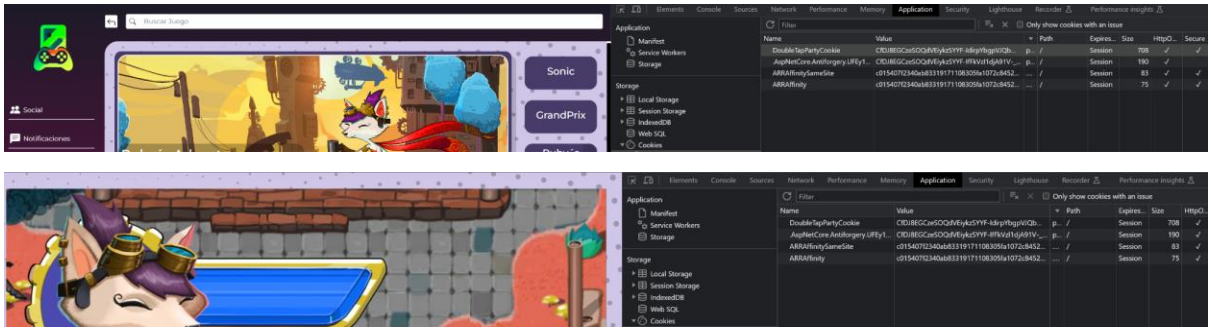


Ilustración 100. Visualización de la cookie de autenticación en diferentes pantallas

Pero al eliminar la *cookie* se vuelve a la página de inicio de sesión:

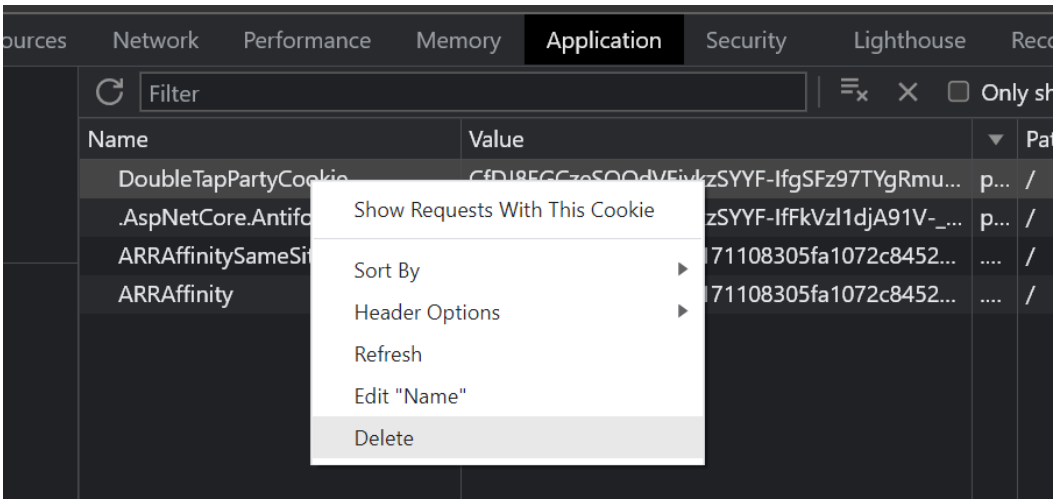


Ilustración 101. Eliminación de la cookie de autenticación

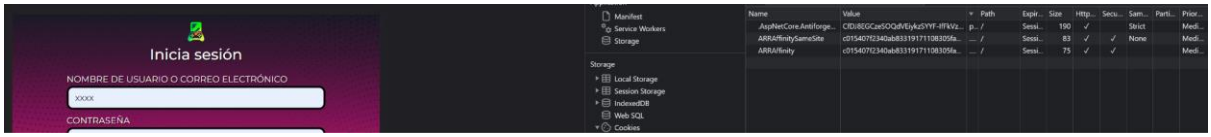


Ilustración 102. Vuelta al login por falta de cookie de autenticación

Publicación del *AppService*

El proceso de publicación de un proyecto desde el código *Visual Studio 2022* ha sido el siguiente.
Desde el portal de *Azure* se accede al *AppService* donde se desea publicar y se descarga el perfil de publicación.

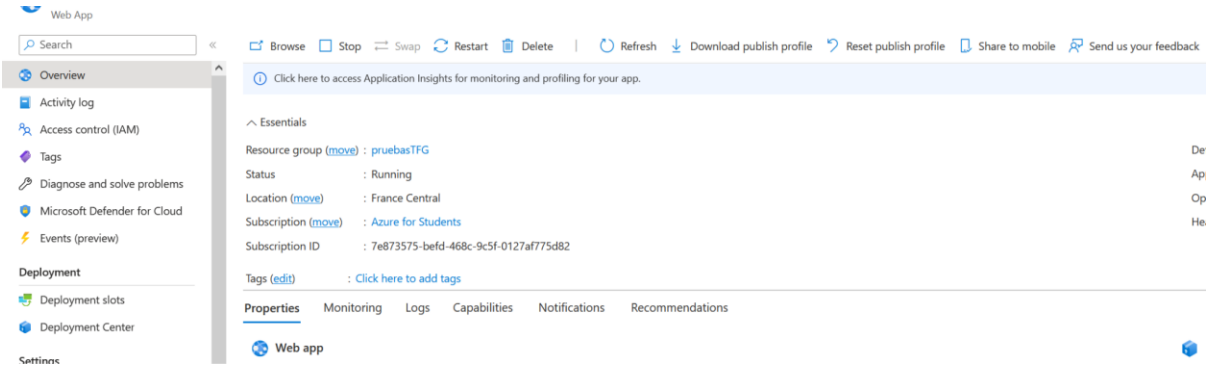


Ilustración 103. AppService de Azure

En *Visual Studio 2022* se pincha en el proyecto a publicar y se selecciona la opción publicar.

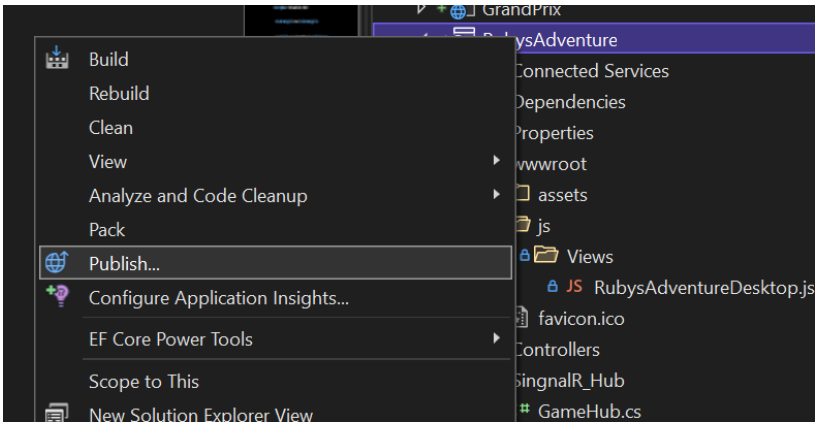


Ilustración 104. Publicación del proyecto

En la pestaña de publicación se selecciona la opción de nuevo y a continuación importar perfil, seleccionando el perfil descargado desde *Azure*. En caso de que el proyecto se desee publicar sobre un directorio virtual se selecciona la opción editar de la pestaña de publicación y se añade en la opción "*Nombre del sitio*" de la pestaña de "*Conexión*" el nombre del sitio más "/" y el nombre del directorio virtual. Tras esto se presionará el botón de publicar.

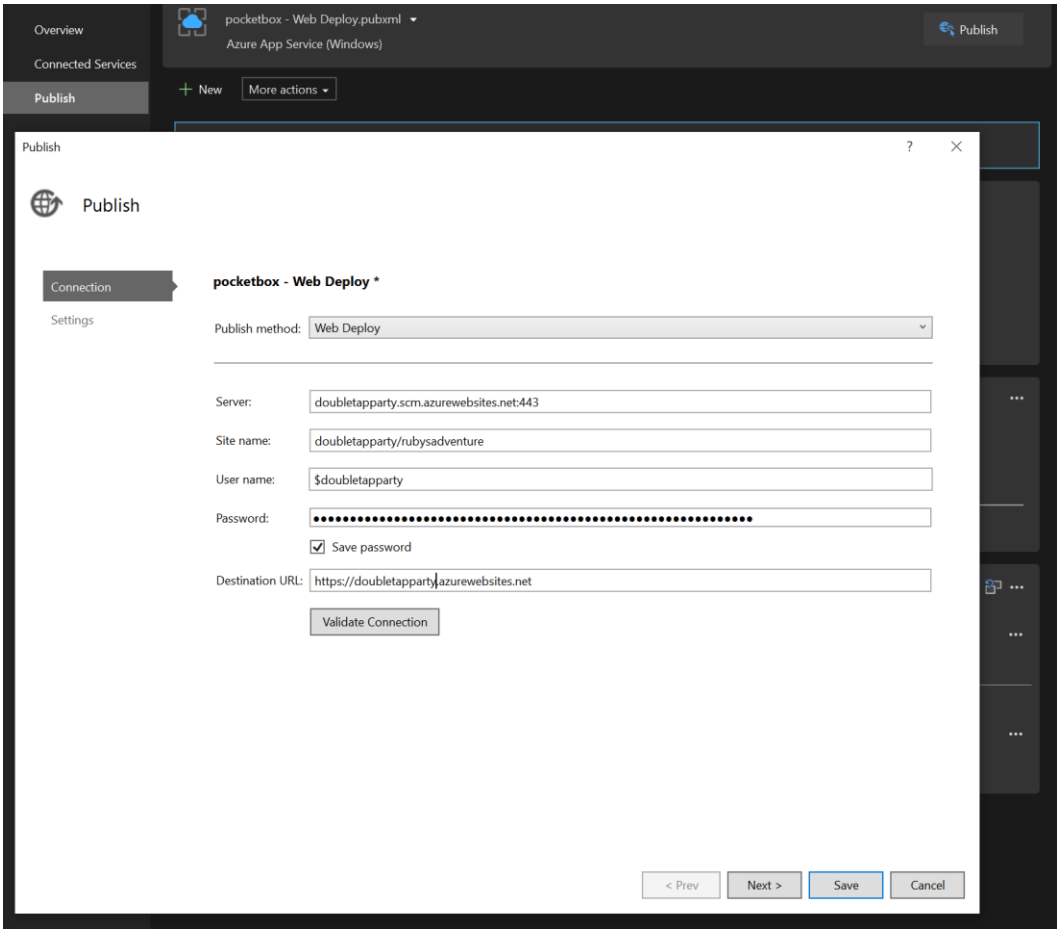


Ilustración 105. Introducción de credenciales del servidor

Envío de archivos *.unityweb* a los navegadores

Cuando se publicó el juego *Unity Ruby's Adventur*e, al acceder desde el navegador se producía un error, ya que no era capaz de encontrar los archivos *.unityweb* necesarios para la ejecución del juego. El motivo es porque *.NET* en su configuración por defecto no permite el envío de archivos *.unityweb*, para ello se añadieron las siguientes líneas de código para permitir el mime type *.unityweb* dentro de los tipos permitidos para enviar.

```
//Permite el envio al navegador de archivos .unityweb
app.UseFileServer();
StaticFileOptions option = new StaticFileOptions();
FileExtensionContentTypeProvider contentTypeProvider = (FileExtensionContentTypeProvider)option.ContentTypeProvider ??
new FileExtensionContentTypeProvider();
contentTypeProvider.Mappings.Add(".unityweb", "application/octet-stream");
option.ContentTypeProvider = contentTypeProvider;
app.UseStaticFiles(option);
```

Ilustración 106. Configuración para permitir archivos.unityweb

Se adquirió el dominio *doubletapparty.fun* desde la web *nominalia.com*, para modificar la ruta de acceso a la web.

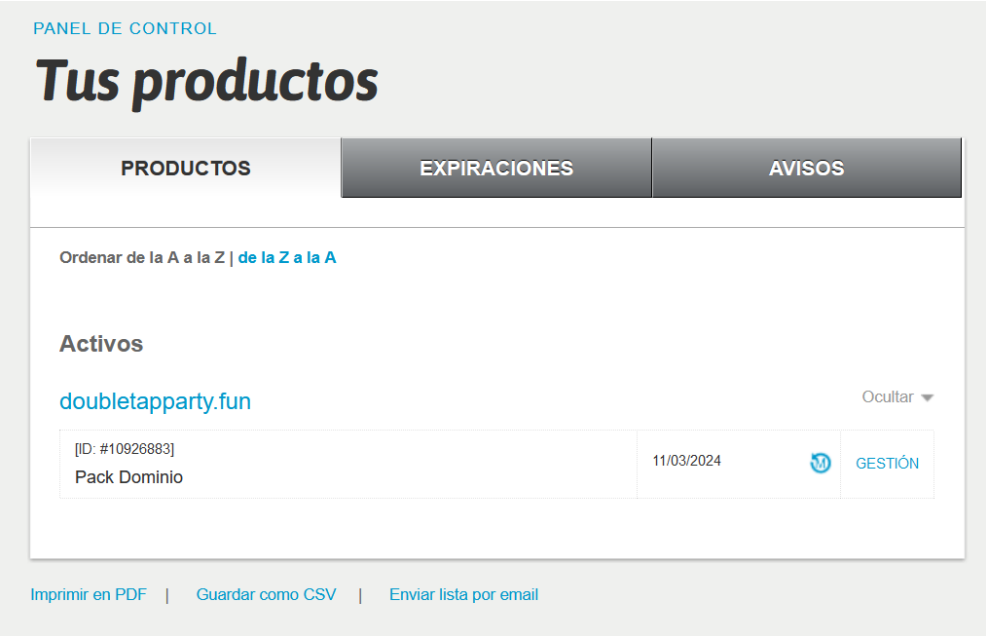


Ilustración 107. Dominio propio en nominalia.com

Para que *Azure* permitiera poder añadir un dominio personalizado al servicio en *Azure*, fue necesario cambiar el plan del servicio del *D1* al *Basic B1* para que se activará la opción de *"Añadir dominio personalizado"*.

El primer paso fue añadir la dirección del dominio personalizado

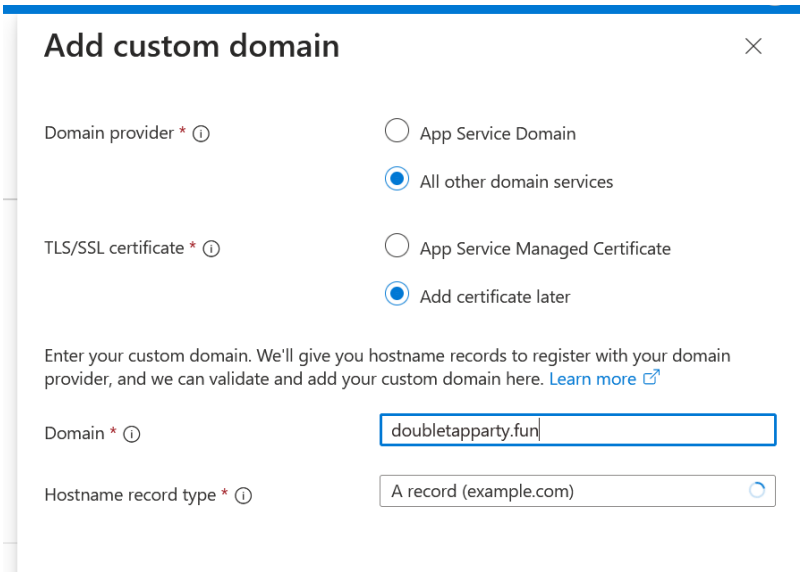


Ilustración 108. Asignación de dominio propio en Servidor Azure

A continuación, se solicitó la validación de propiedad del dominio y se añadieron los registros de nombres de host en la configuración del Sistema de nombres de dominio (*Domain Name System* - DNS) del dominio.

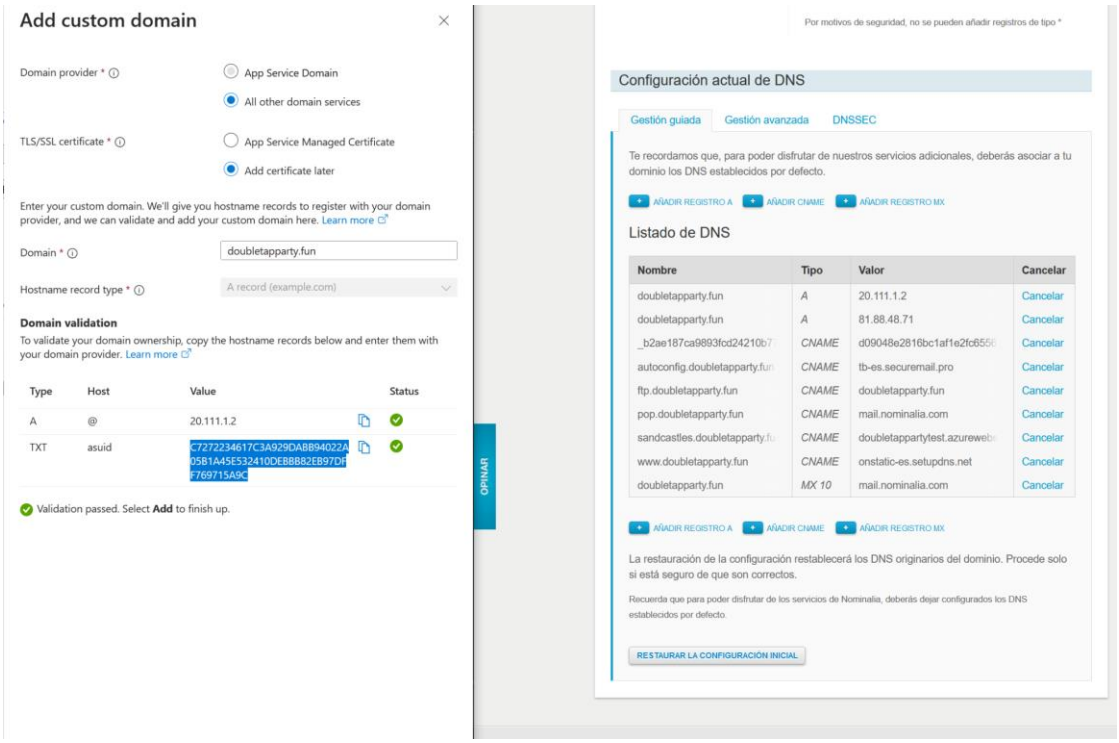


Ilustración 109. Acreditación de dominio de Azure y configuración del listado de DNS

Tras ellos se añadiría el dominio personalizado asociado al servicio, pero con un indicador de que el dominio no tenía una vinculación *TLS/SSL*, por lo que al acceder a la URL del dominio aparecería una advertencia en el navegador de que la conexión no era segura.

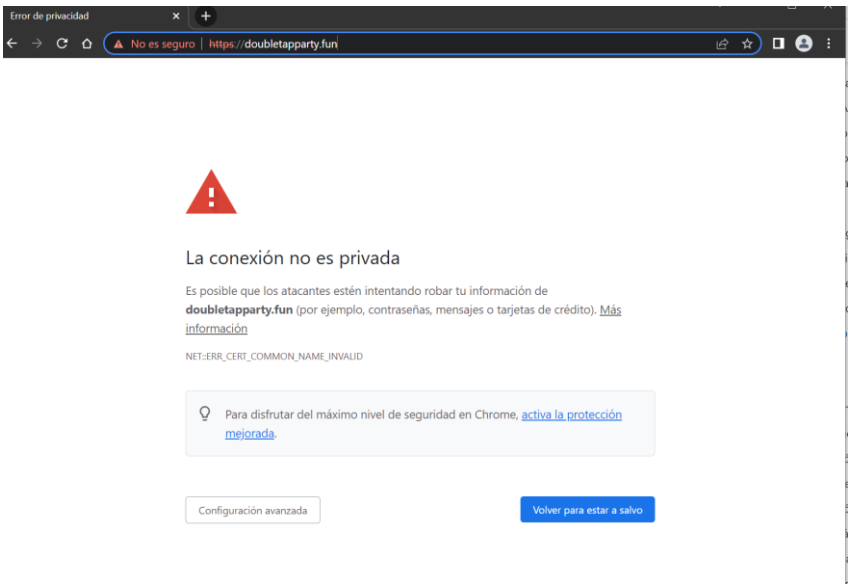


Ilustración 110. Error de seguridad

Y aunque se aceptará el riesgo, en algunos navegadores mostraría la siguiente información.

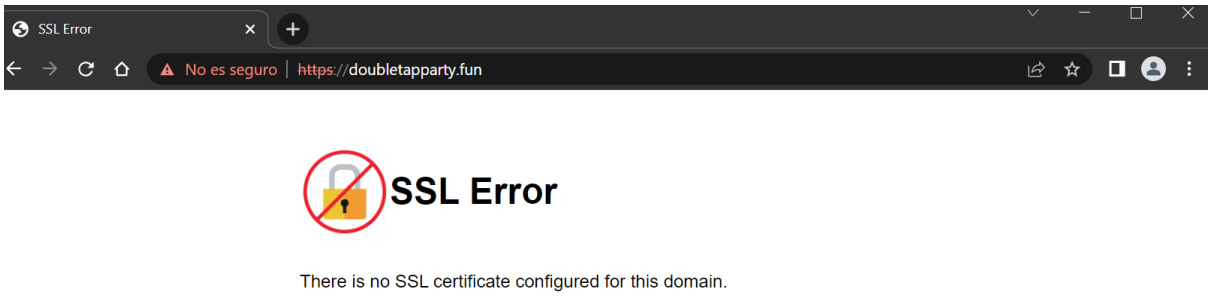


Ilustración 111. Error de certificado SSL

Posteriormente se intentó generar un certificado de manera gratuita e instalarlo en el servidor, pero las advertencias y mensajes continuaban apareciendo, por lo que se descartó el uso de momento.

Capítulo 5

Validación

Resultado final

El resultado final consiste en una página web accesible desde dispositivos móviles y ordenadores, brindando a los usuarios una experiencia completa. Los usuarios tienen la capacidad de navegar por la página y aprovechar todas las funcionalidades ofrecidas por la aplicación.

Después de la comparativa realizada en el apartado *"Casos de usos"* es necesario destacar el resultado en esas partes de la plataforma.

Menú principal y selección de juegos

En la página de inicio se encuentra una cabecera que muestra una selección de juegos destacados. Seguidamente se encuentran diferentes apartados con los juegos agrupados por categorías.

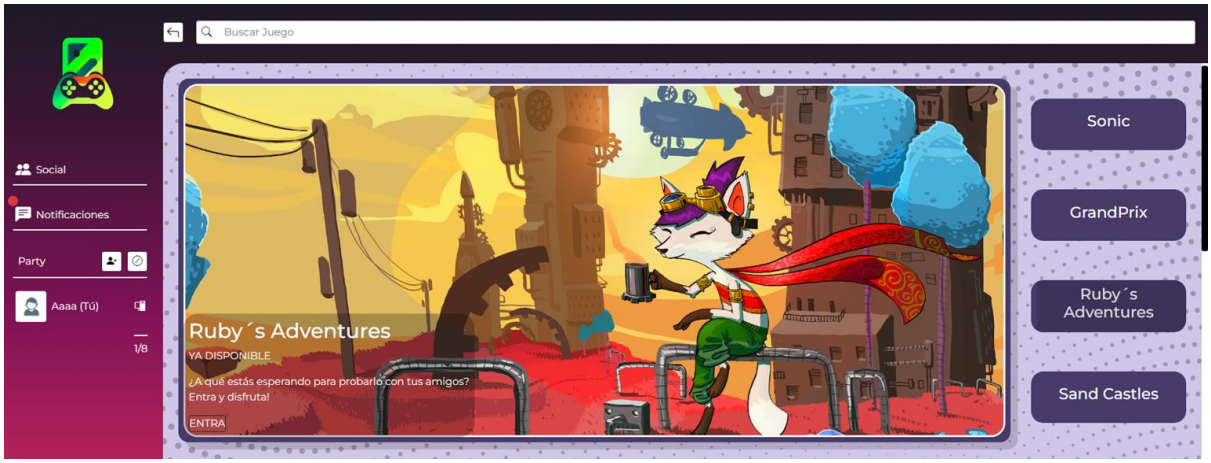


Ilustración 112. Pantalla principal con la sección de juegos destacados

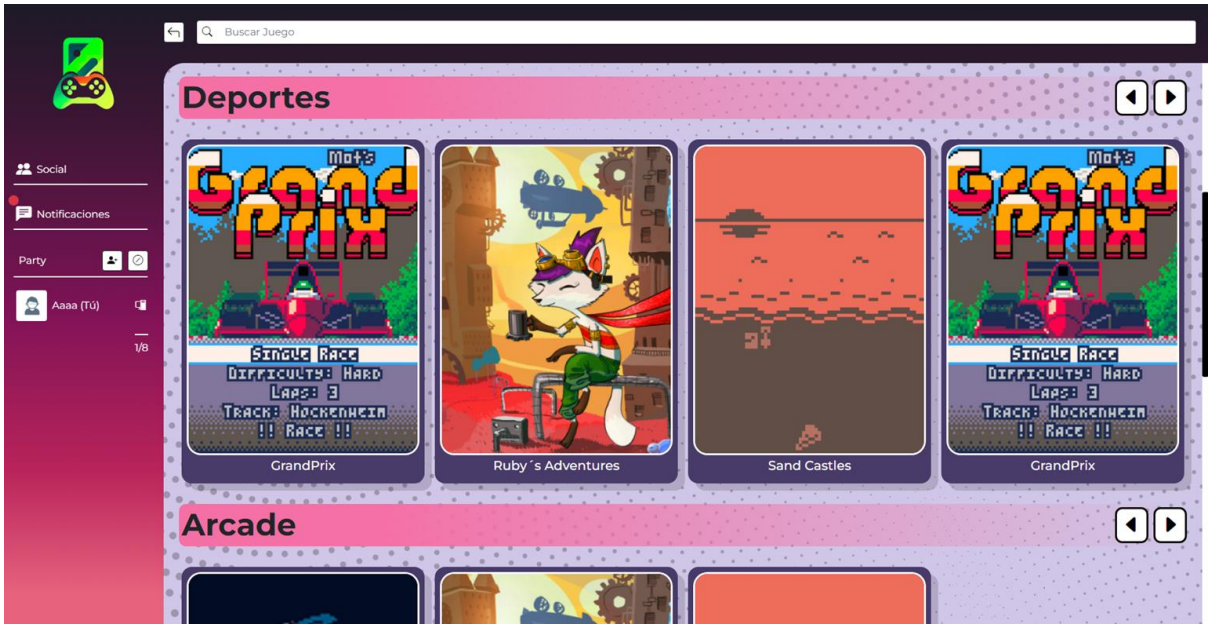


Ilustración 113. Pantalla principal con los juegos agrupados por categoría

Esta página principal también puede cambiarse si se utiliza la barra de búsqueda ya que desplegará únicamente aquellos juegos cuyos nombres coincidan con el texto buscado.

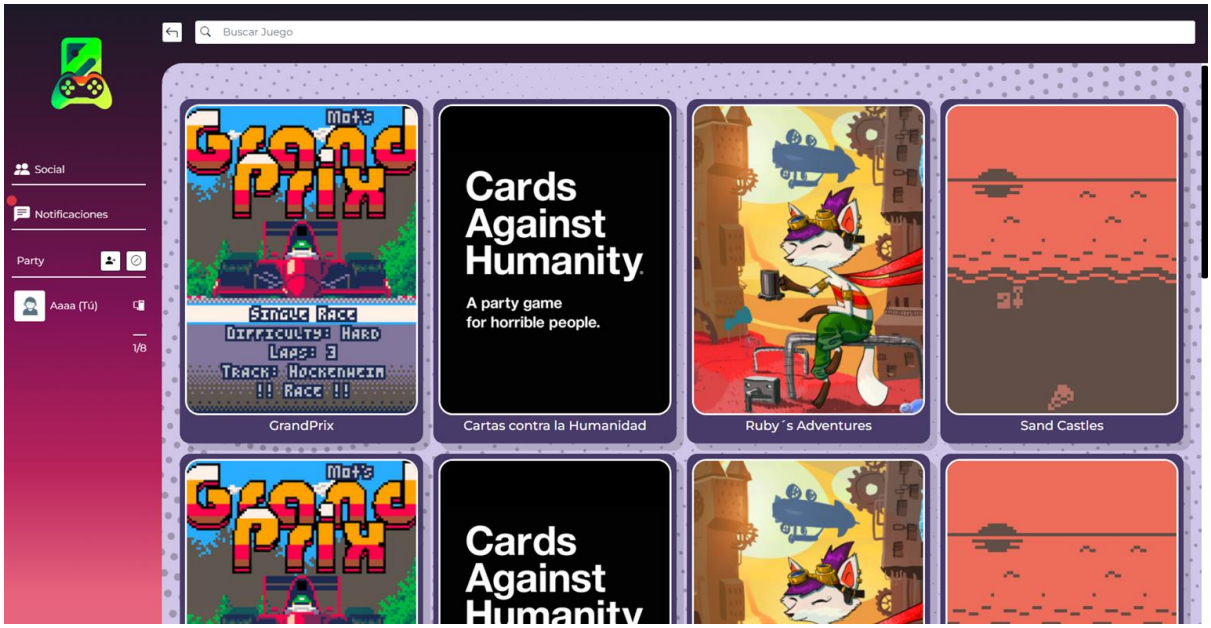


Ilustración 114. Pantalla principal con juegos filtrados del menú principal con una búsqueda realizada

Información del juego y del jugador

En la pantalla de información del juego, se ha creado un panel que presenta la portada del juego a la izquierda, mientras que a la derecha se muestran diversos datos relevantes. Entre estos datos, se incluye la valoración recibida del juego por parte de la comunidad,

representada mediante iconos faciales. También se muestra la fecha más reciente en la que el usuario lanzó el juego, así como el tiempo total invertido por el usuario en ese juego. Por último, se ofrece una descripción detallada del juego.

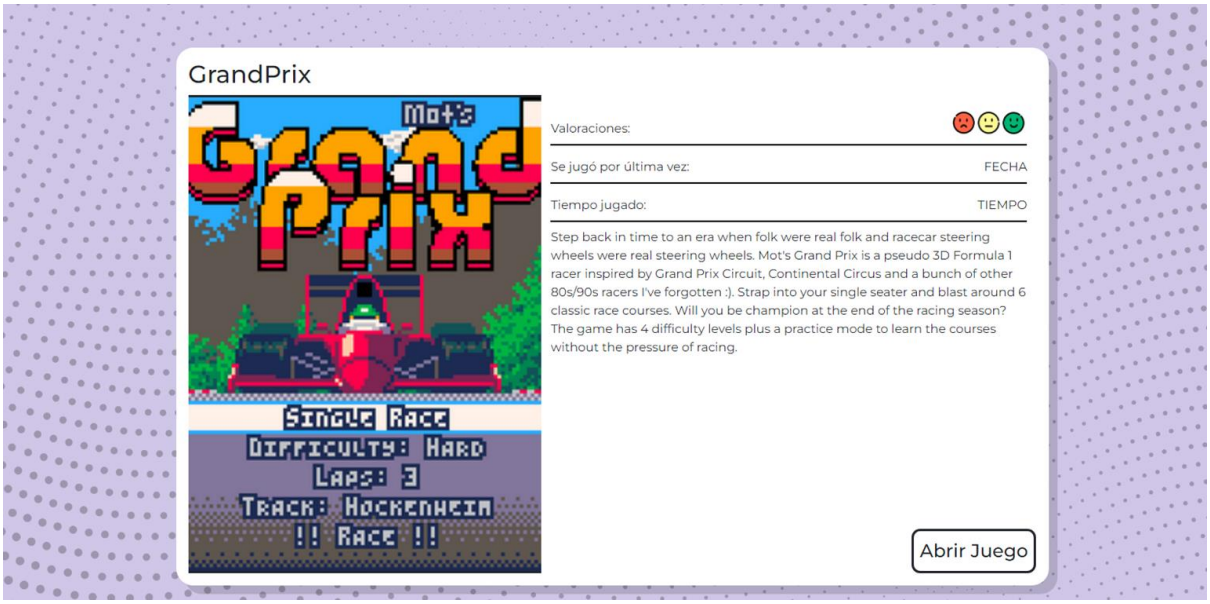


Ilustración 115. Pantalla de juego - Información

Seguidamente, el usuario encuentra la sección de comentarios donde podrá escribir su reseña del juego y ver los comentarios de otros usuarios ordenados por fecha.

Ilustración 116. Pantalla del juego – Reseñas

Gestión de amistades

Las funcionalidades que ofrece la plataforma relacionada con las amistades entre usuarios son las siguientes:

- Agregar Amigos: Los usuarios pueden agregar a otros usuarios como amigos permitiendo establecer y mantener un seguimiento de las actividades de los amigos en la plataforma.
- Invitar a jugar: Los usuarios pueden crear grupos e invitar a sus amigos para jugar a juegos multijugador.

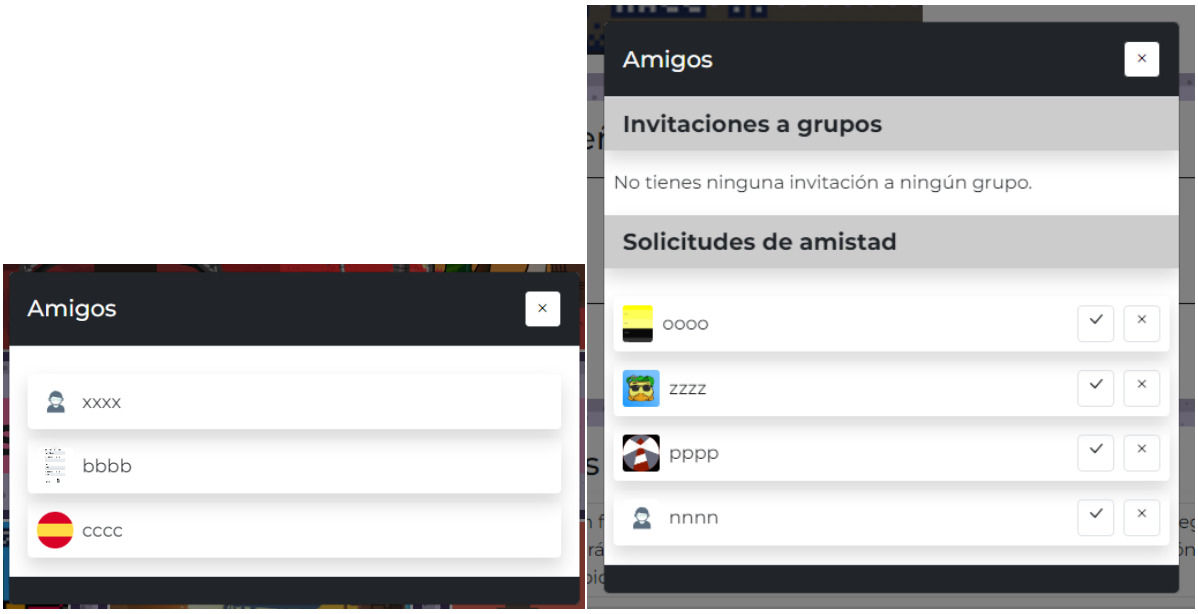


Ilustración 117. Ventanas de notificaciones y de invitación a amigos

Información del usuario

Dentro del perfil de usuario, el usuario puede editar toda su información y cerrar la sesión del juego. De esta forma, si el usuario decide cambiar de nombre, *nickname* o de correo electrónico el sistema le otorga una opción de actualizarlo.

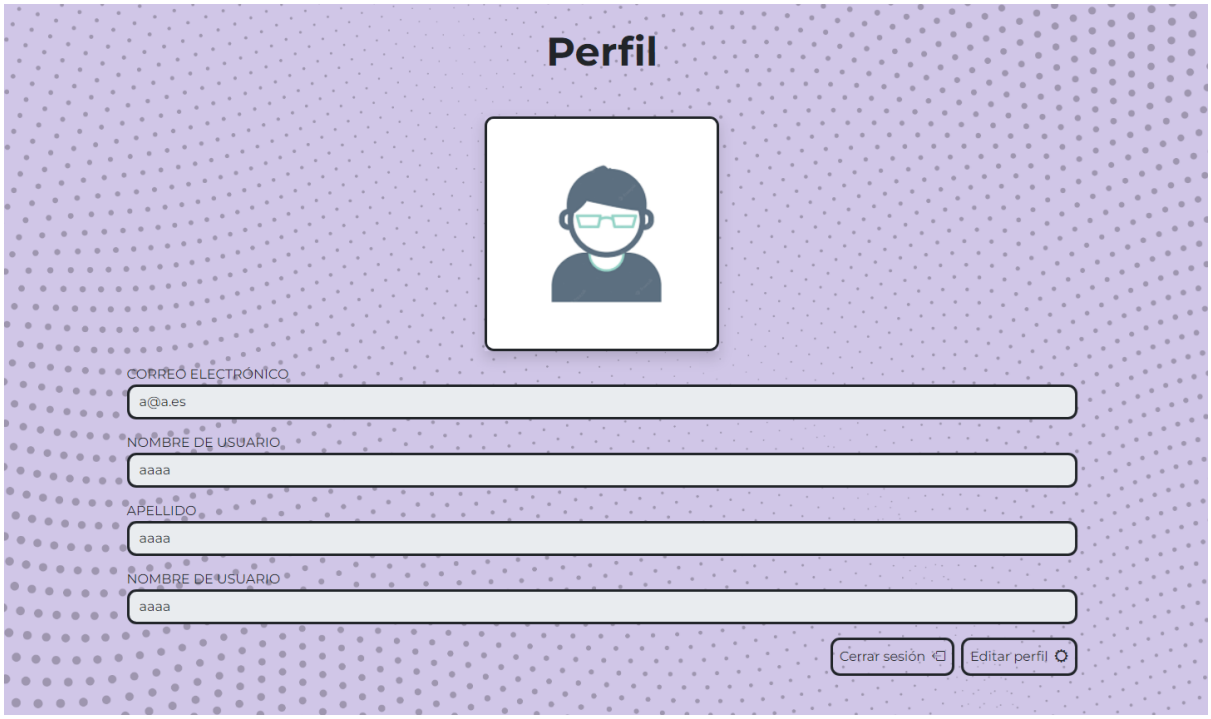


Ilustración 118. Pantalla de información del usuario

Ejemplos prácticos de uso

Para mostrar el funcionamiento de la aplicación y proporcionar a los usuarios una experiencia desde cero, se ha decidido realizar dos pruebas.

Uso de la aplicación

La primera de ellas se ha dejado la aplicación a una persona ajena al desarrollo dejándola libre de realizar cualquier cosa dentro de ella con una sola excepción, esa persona no podía probar el juego multijugador *"Cartas contra la Humanidad"* ya que era una prueba individual.

Para esta prueba se ha escogido un usuario que frecuenta juegos tanto de *PlayStation* como de *PC*.

Esta persona, una vez indicada la página web y la necesidad de utilizar el teléfono móvil para que pueda disfrutar de los juegos, lo primero que ha hecho ha sido crear su propio usuario desde el ordenador, y una vez creado ha iniciado sesión en el móvil con el mismo.

Con el móvil conectado, ha navegado por la página principal, abriendo el primer juego dentro del apartado de deportes, *GrandPrix*. Una vez abierta la página de detalle del juego, ha buscado el botón de abrir juego y lo ha lanzado.

Después de jugar dos partidas, decidió salir del juego. Una vez hecho esto tuvo un problema de conexión y se desvincularon los dispositivos. Este error fue apuntado y se arregló refrescando el navegador de ambos dispositivos.

El usuario volvió a la página principal y decidió pulsar los botones de navegación del *Banner* de los juegos destacados, eligiendo *Ruby´s Adventures* y abriendo la página del detalle del juego pulsando en el propio banner.

Dentro de la página del detalle, lo primero que hizo fue escribir un comentario y seguidamente lanzar el juego. Después de jugar durante un rato a ese juego, se decidió parar la prueba y pedir opinión.

Opinión

- “La navegación se siente muy bien, una pena que se desconectara el teléfono del ordenador, pero por lo demás está bastante bien.”

Cartas contra la Humanidad

En la segunda prueba se ha creado un grupo con alrededor de cinco personas para lanzar el juego de *Cartas contra la Humanidad* con varias personas simultáneamente con el fin de garantizar la jugabilidad del juego y la estabilidad.

Para esta prueba se escogió un grupo variado de personas que frecuentan videojuegos con grados muy diferentes.

Una vez creado el grupo el juego se lanzó sin problemas, creó la sesión y esperó a que se conectaran todos los dispositivos. Se creó una partida de diez rondas, pero en la tercera ronda un teléfono se desconectó y se decidió parar la partida y relanzarla de nuevo para que todos los usuarios pudieran jugar.

La segunda partida se lanzó de igual manera que la primera, una vez creado el grupo, y se volvió a elegir una partida de diez rondas. En esta ocasión no hubo ningún fallo a la hora de ejecutarla y todos los usuarios pudieron disfrutarla de principio a fin.

Una vez terminada la segunda partida, los jugadores decidieron jugar una tercera, para este caso, como no hubo que rehacer el grupo, una vez finalizada la partida se lanzó otra partida, pero de cinco rondas. En este caso, el mismo dispositivo móvil que falló en la primera partida, también falló. En este caso, los jugadores decidieron terminar la partida sin ese jugador.

El dispositivo móvil que fallaba al parecer tuvo problemas de cobertura durante la prueba, perdiendo la conexión y desconectándose del servidor.

Opiniones

- “Lo de que el ganador de cada ronda se elija por votaciones me parece maravilloso”
- “Está genial para pasar el rato con tus amigos”
- “¿No podemos jugar una partida más?”
- “La única pega que le pondría sería que quizá, el tiempo a la hora de elegir la respuesta es muy corto cuando se tiene más de una opción”

Benchmarks

Se han realizado diferentes pruebas de estrés en la página web. En la primera prueba se ha realizado una conexión de 50 usuarios virtuales a la página web enviando solicitudes a su punto de conexión durante 120 segundos.

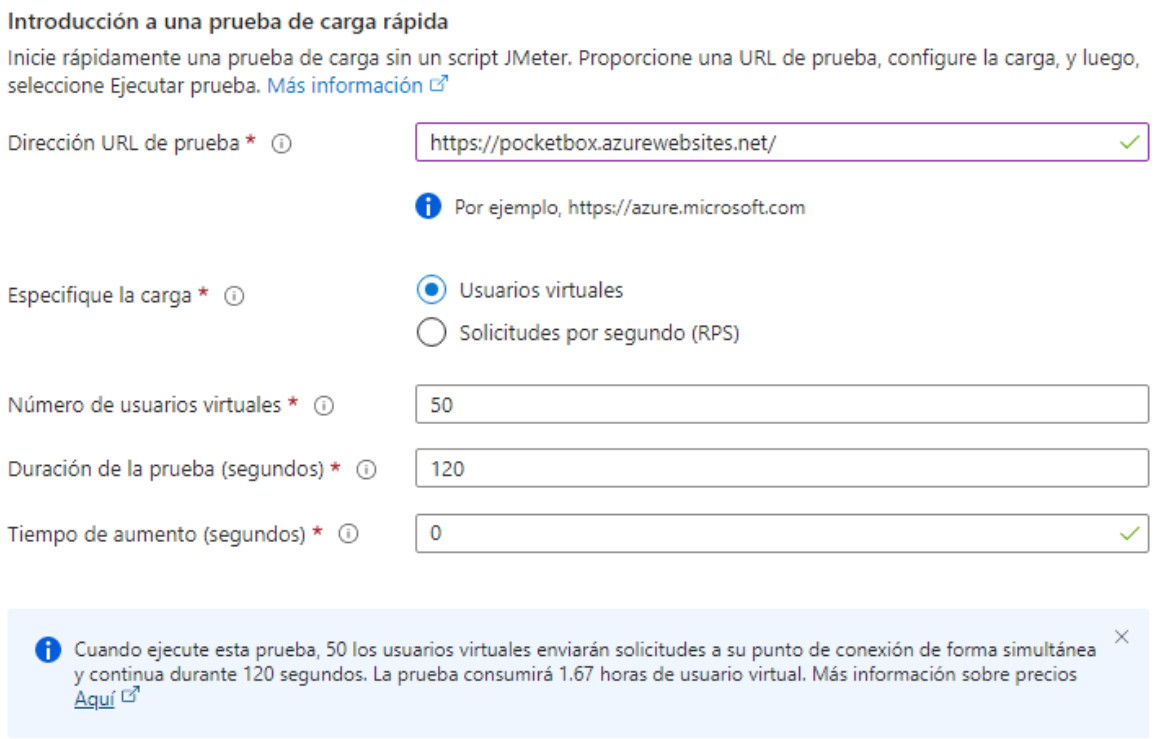


Ilustración 119. Configuración de las pruebas de estrés con 50 usuarios virtuales

En el resultado de esta prueba se encuentran varios gráficos que indican el número de usuarios conectados durante la prueba, el tiempo de respuesta de la página web, las solicitudes por segundo y los errores que han surgido durante la prueba.

El dato más destacable en la prueba es el primer tiempo de respuesta de la página web, con un tiempo de 6.99 segundos, el tiempo de respuesta del resto de cargas baja drásticamente.

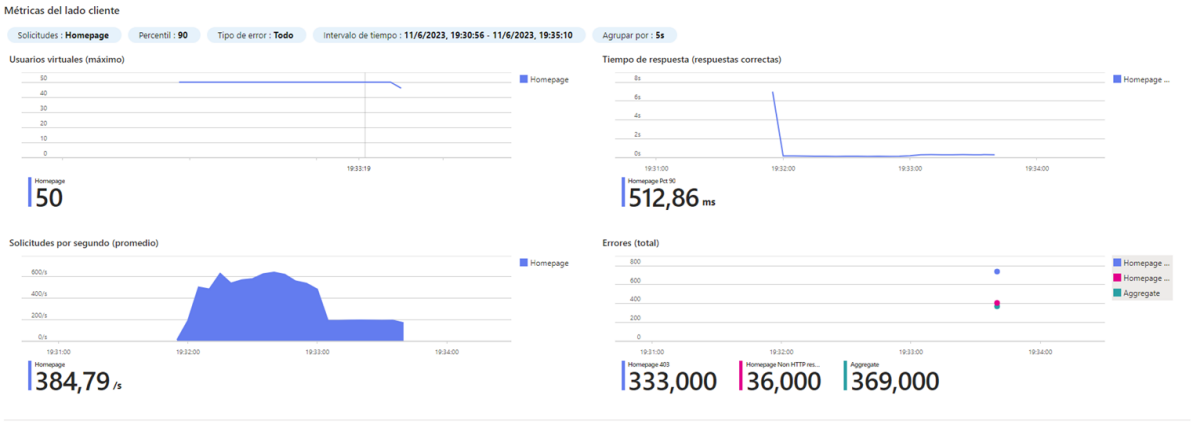


Ilustración 120. Resultados de la prueba con 50 usuarios virtuales

Debido al plan de *Azure* contratado, al estar inactiva la web debido a que ha hecho ninguna solicitud de acceso durante un periodo de tiempo relativamente largo, esto provoca que la primera carga tarde más, ya que vuelve a activar la web quitándole del reposo forzado.

También se encuentran otros gráficos que indican las métricas del estado del motor de carga. En este caso, se ha ejecutado la prueba con un solo motor, con que ha utilizado aproximadamente un 12.08% de su CPU, un 13.27% de memoria y 2.04 MB/s de uso de red alojando a los 50 usuarios virtuales.

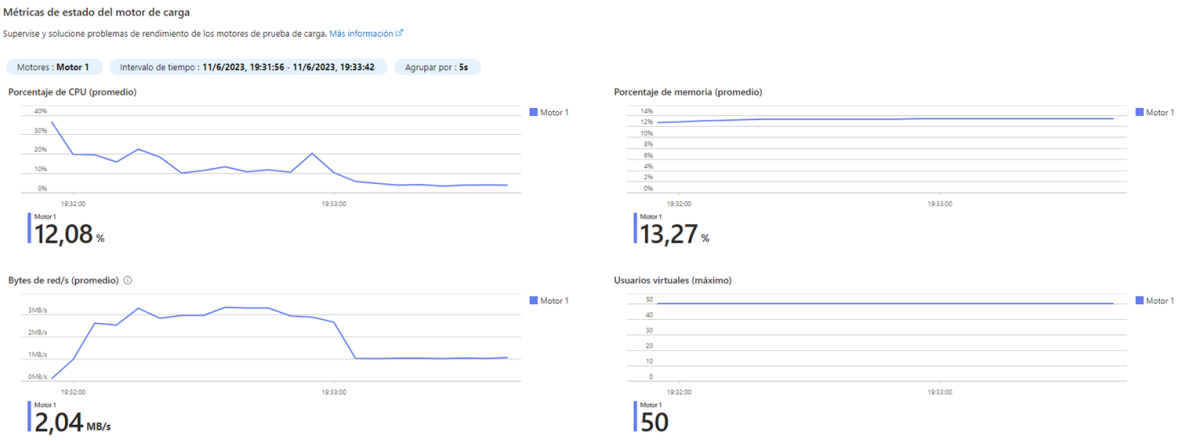


Ilustración 121. Métricas del motor de carga de la prueba con 50 usuarios virtuales

La segunda prueba que se ha realizado se ha hecho con el número máximo de usuarios virtuales que permitía la plataforma, 500. El resto de los datos sigue siendo el mismo que en la prueba anterior.

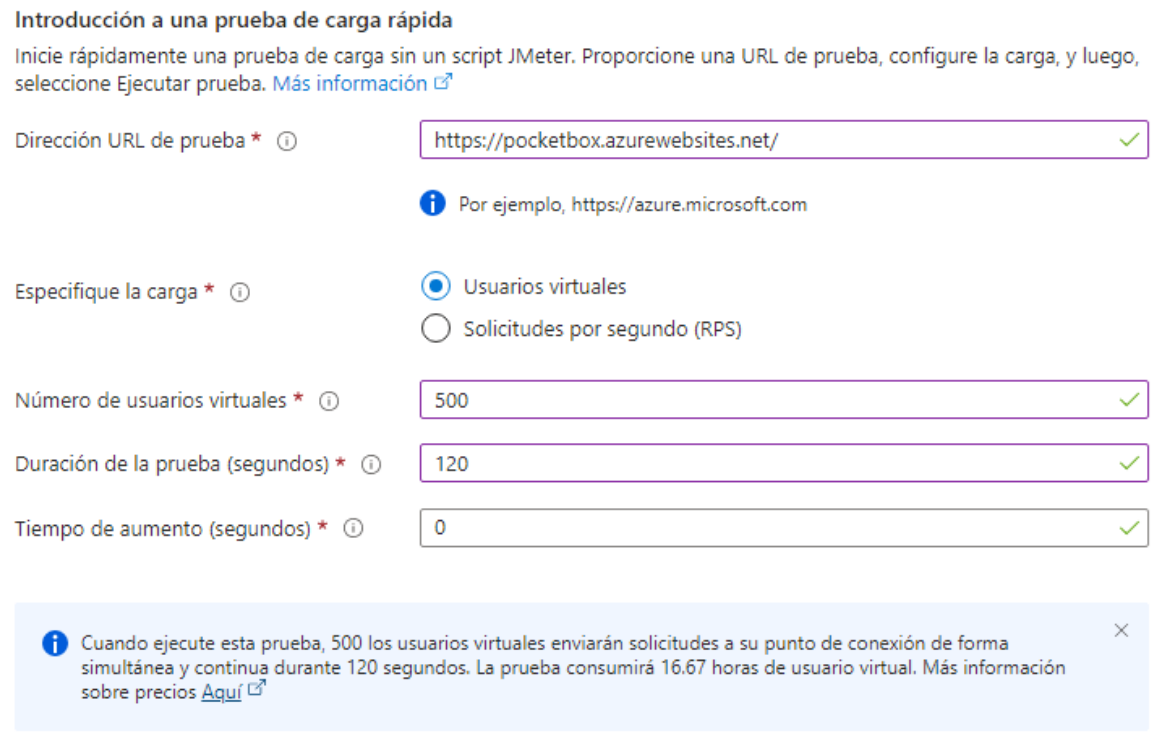


Ilustración 122. Configuración de las pruebas de estrés con 100 usuarios virtuales

Del resultado obtenido en esta segunda prueba se puede concluir que con el aumento de solicitudes iniciales y de peticiones a lo largo del tiempo, se produce un aumento en el tiempo de respuesta de la página web. El tiempo de respuesta de la primera solicitud, solicitud que rompe la inactividad de la página web, es de 13,26 segundos, el resto baja también drásticamente, pero a un tiempo mayor de respuesta debido a la mayor carga de trabajo.

Otro dato a destacar es que, en el periodo final de la prueba, 96 usuarios virtuales han tenido problemas y se han desconectado, pudiéndose reconectar 59 tras ese error.

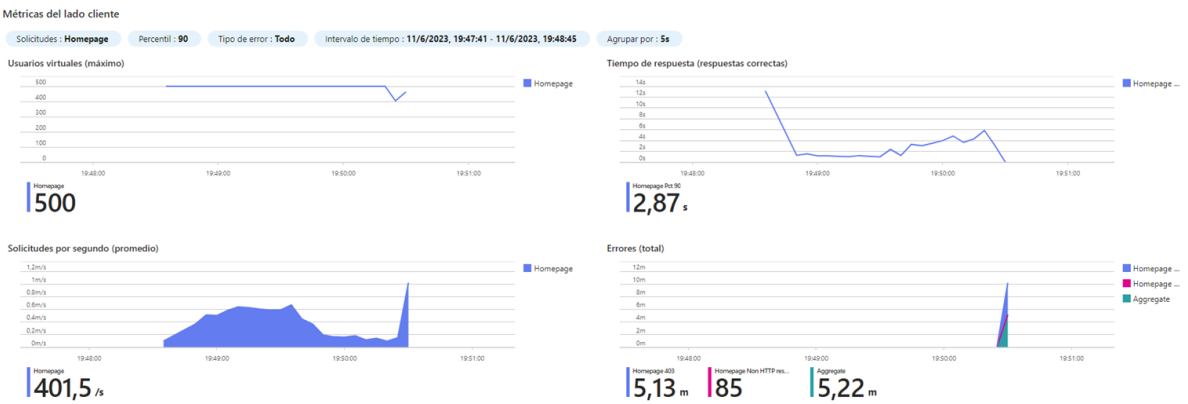


Ilustración 123. Resultados de la prueba con 100 usuarios virtuales

En los gráficos de las métricas del estado del motor de carga se observa que para esta segunda prueba se han necesitado dos motores, con que ha utilizado aproximadamente un 21.09% y 22.17% de su CPU, un 21.4% y 13.03% de memoria y 749.43 MB/s y 758.2 MB/s de uso de red alojando a los 250 usuarios virtuales cada uno.

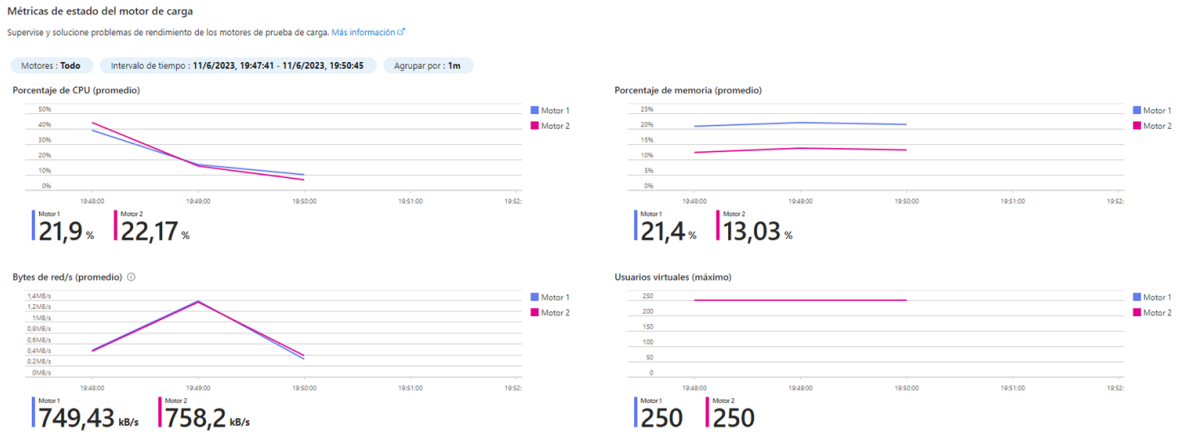


Ilustración 124. Métricas del motor de carga de la prueba con 100 usuarios virtuales

Como conclusión a los resultados obtenidos en las pruebas se valora que un plan menos limitado mitigaría el número de errores de respuesta de la aplicación, la latencia y la tasa de respuesta inicial, debido a la mejora en los recursos del sistema contratado.

Name	ACU/vCPU	vCPU	Memory (GB)	Remote Storage (GB)	Scale (instance)	SLA	Cost per hour	Cost per month
Dev/Test (For less demanding workloads)								
Free F1	60 minutes/day...	N/A	1	1	N/A	N/A	**Free**	**Free**
Shared D1	240 minutes/da...	N/A	1	1	N/A	N/A	**0,013 USD**	**9,49 USD**
Basic B1	100	1	1.75	10	3	99.95%	**0,094 USD**	**68,62 USD**
Basic B2	100	2	3.5	10	3	99.95%	**0,188 USD**	**137,24 USD**
Basic B3	100	4	7	10	3	99.95%	**0,375 USD**	**273,75 USD**
Production (For most production workloads)								
Standard S1	100	1	1.75	50	10	99.95%	**0,125 USD**	**91,25 USD**
Premium v3 P1V3	195	2	8	250	30	99.95%	**0,334 USD**	**243,82 USD**
Premium v3 P2V3	195	4	16	250	30	99.95%	**0,668 USD**	**487,64 USD**
Premium v3 P3V3	195	8	32	250	30	99.95%	**1,336 USD**	**975,28 USD**
Standard S2	100	2	3.5	50	10	99.95%	**0,25 USD**	**182,50 USD**
Standard S3	100	4	7	50	10	99.95%	**0,50 USD**	**365,00 USD**
Premium P1	100	1	1.75	250	20	99.95%	**0,375 USD**	**273,75 USD**
Premium P2	100	2	3.5	250	20	99.95%	**0,75 USD**	**547,50 USD**
Premium P3	100	4	7	250	20	99.95%	**1,50 USD**	**1095,00 USD**
Premium v2 P1V2	210	1	3.5	250	30	99.95%	**0,25 USD**	**182,50 USD**
Premium v2 P2V2	210	2	7	250	30	99.95%	**0,50 USD**	**365,00 USD**
Premium v2 P3V2	210	4	14	250	30	99.95%	**13,933 USD**	**10170,944 USD**

Ilustración 125. Listado y descripción de los planes de Azure

Capítulo 6

Conclusiones

Objetivos alcanzados

Estudiar las plataformas de distribución de videojuegos actuales, análisis de sus características y funcionalidades

Como se muestra en el capítulo 2 en el apartado de *"Casos de estudio"*, se llevó a cabo un análisis de las plataformas más exitosas en la actualidad, centrándose en las necesidades y funcionalidades que se deseaban incorporar en la plataforma que se estaba construyendo. El proyecto desarrollado es el resultado de la integración de estas funcionalidades. **Este objetivo se ha cumplido al 100%.**

Evaluar las diferentes tecnologías de desarrollo de software posibles para el desarrollo de la aplicación

Al estudiar las posibles tecnologías para llevar a cabo el proyecto, en el apartado de *"Estudio de alternativas"* del capítulo 2, se evaluaron las ventajas y desventajas de cada una de ellas. La elección realizada ha sido la más adecuada, ya que no ha generado ninguna restricción para alcanzar los objetivos de desarrollo establecidos. La combinación de todas estas tecnologías ha permitido una perfecta interrelación entre ellas, encajando de manera óptima y facilitando el desarrollo del proyecto. **Este objetivo se ha cumplido al 100%**

Desarrollar un método de comunicación en tiempo real entre distintos dispositivos

Uno de los puntos críticos para la viabilidad del proyecto era encontrar una forma de comunicación en tiempo real entre dos conexiones a una misma web. Como se explica en el capítulo 2 en el apartado *"Estudio de alternativas"*, la elección de utilizar *SignalR* como biblioteca para este propósito resultó crucial, ya que su integración en el código fue simple y los resultados superaron todas las expectativas. **Este objetivo se ha cumplido al 100%.**

Construir la aplicación en una arquitectura descentralizada, en la que cada juego se ejecute en un servicio web diferente

En el capítulo 4 en el apartado *"Despliegues y tecnologías"* se detalla cómo se ha construido una arquitectura basada en servicios virtuales que proporciona a la aplicación la robustez deseada sin un coste extra en infraestructura. **Este objetivo se ha cumplido al 100%.**

Implementar la aplicación dentro de un proveedor de servicios de la nube y aprovechar sus capacidades

Tal como se explica en el capítulo 4 en el apartado de *"Arquitectura de la aplicación"*, todos los servicios utilizados por la aplicación se encuentran dentro del sistema de *Azure*. Esto ha facilitado la construcción de la arquitectura de la aplicación y ha permitido añadir funcionalidades extras no planteadas en el diseño previo. **Este objetivo se ha cumplido al 100%**

Adquirir un dominio y enlazarlo a la aplicación desarrollada

Conforme se detalla en el capítulo 4 en el apartado *"Dominio personalizado al servicio en Azure*", este objetivo se completó adquiriendo un dominio, configurándolo, vinculándolo a la web de *Azure* e instalando los certificados de seguridad, pero se decidió no hacer usó de él, ya que para poder vincular el dominio a la web de *Azure* era necesario cambiar el plan del *AppService* a uno de mayor presupuesto, además el certificado de seguridad instalado no funcionaba correctamente. **Este objetivo se ha cumplido al 20%**

Crear un código escalable

En el capítulo 4 en el apartado *"Desarrollo de la Aplicación Web"* se explica cómo la arquitectura de la aplicación está basada en Arquitectura de 3 capas (*Presentation*, *Businness Access Layer*, *Data Access Layer*), lo cual permite una escalabilidad bastante fácil, pero en un futuro estaría bien pasar a una arquitectura Hexagonal (Martínez et al., 2021) con el fin de tener el código lo más organizado y escalable posible. **Este objetivo se ha cumplido al 75%.**

Crear un *framework* que permita desarrollar fácilmente juegos para incorporarlos en la aplicación

Como se muestra en el capítulo 4 en los apartados "*GameHub"*, *"Controladores estándar de juegos"* y *"Controlador para juego WebGL"* se explica el desarrollo de un *framework* para simplificar el trabajo de los desarrolladores a la hora de incorporar las funcionalidades de la plataforma e incluir el juego en el sistema. Este objetivo se ha completado parcialmente, por motivos de complejidad en el desarrollo únicamente se ha tenido en cuenta los juegos de un solo jugador. **Este objetivo se ha cumplido al 50%.**

Proveer a la aplicación de juegos externos incorporando las funcionalidades desarrolladas

Aprovechando el *framework* desarrollado, se han incorporado juegos de licencia MIT agregando las funcionalidades de la aplicación y permitiendo el manejo del juego desde el controlador estándar. **Este objetivo se ha cumplido al 100%.**

Permitir incorporar juegos *HTML5* y *WebGL*

En el capítulo 4 en el apartado *"Desarrollo de Videojuegos"*, se han incorporado juegos dentro de la aplicación los dos tipos de juego. Para los juegos desarrollados en *HTML5* fue necesario simular la pulsación de las teclas y conectarla a las llamadas de *SignalR*. Para los juegos en

WebGL resultó un poco más complejo pues se tuvo que codificar un puente entre el navegador y el juego. **Este objetivo se ha cumplido al 100%.**

Evaluar y testear la aplicación

Tal como se muestra en el capítulo 5 en el apartado *"Benchmarks"*, utilizando el servicio de *Azure* "*Azure Load Testing*" se han realizado pruebas de rendimiento y eficiencia sobre el plan de Azure contratado. **Este objetivo se ha cumplido al 100%.**

Lecciones aprendidas

Durante el desarrollo del proyecto, se han encontrado diversos desafíos y aprendido lecciones valiosas que han influido en el enfoque y en futuros proyectos. A continuación, se presenta un resumen de cosas a mejorar y lecciones aprendidas durante el desarrollo.

Planificación y diseño inicial: La planificación y el diseño inicial fue insuficiente. Al no tener un amplio conocimiento previo de los entornos de desarrollo, no se dedicó el esfuerzo suficiente para definir los requisitos y se decidió ir realizando esta definición a medida que se fue conociendo y dominando el entorno. Como resultado, se necesitaron hacer cambios y ajustes significativos en etapas posteriores del proyecto, lo que afectó al aumento del esfuerzo y del tiempo dedicado al desarrollo.

Control de versiones: Aunque se ha utilizado un control de versiones durante el desarrollo, somos consciente de que no haberlo utilizado de la manera más correcta y en alguna ocasión se ha visto perdido parte del desarrollo realizado.

Pruebas: Se han realizado pruebas durante el desarrollo, pero por falta de tiempo y planificación no se ha destinado suficiente tiempo y recursos para pruebas exhaustivas. Como resultado, se identificaron algunos errores y problemas en etapas posteriores, lo que requirió un esfuerzo adicional para corregirlos. Para proyectos futuros será necesario reservar un tiempo en la realización de pruebas para evitar problemas en etapas posteriores del desarrollo.

Comunicación: A pesar de los problemas de los desarrolladores para establecer reuniones de seguimiento. La comunicación interna que se ha llevado a cabo durante el desarrollo ha sido buena, no obstante, podría haber sido más efectiva en determinados momentos del desarrollo. Hubo ocasiones en las que la falta de comunicación clara y regular entre los miembros del equipo provocó malentendidos y obligando a rehacer trabajos realizados.

Flexibilidad y adaptabilidad: Al tener poca experiencia con las herramientas de desarrollo, cada poco tiempo surgían diferentes problemáticas o desafíos a los que enfrentarse. Se aprendió a modificar el enfoque para adaptarse a las posibilidades del entorno de trabajo.

Arquitectura y sistemas: Ambos desarrolladores disponemos de poca o nula experiencia diseñando y administrando sistemas, por lo que toda la parte de administración ha supuesto

un estrés extra y una gran cantidad de conocimiento nuevo adquirido. Se aprendió y decidió cómo aplicar los conceptos de seguridad, escalabilidad, rendimiento y robustez desde un punto de vista de administración de sistemas dentro de la plataforma *Azure* de la cual no había experiencia previa.

En conclusión, la realización de este proyecto ha implicado un esfuerzo considerable en diversos aspectos. Sin embargo, ha sido una gran oportunidad para adquirir nuevos conocimientos y habilidades en tecnologías y áreas de la informática que no fueron abordadas durante la formación universitaria, así como en el ámbito de la gestión del proyecto.

Impacto del proyecto

Se ha analizado el impacto del proyecto en diferentes niveles:

Impacto social

A nivel social, el impacto de la aplicación web dependerá directamente de la cantidad y la calidad de los juegos que se ofrezcan en la plataforma. Si se consigue obtener una amplia variedad de juegos, la aplicación será capaz de generar interés de usuarios de diferentes edades y gustos.

La posibilidad de disfrutar de juegos a través de dispositivos móviles y una web de un ordenador, sin requerir *hardware* adicional o descargas de programas proporcionará una experiencia de juego accesible para los usuarios. Esto permitirá que la aplicación web llegue a un público más amplio y diverso, incluyendo a aquellos que no poseen consolas de juegos o que no pueden permitirse invertir en equipos costosos.

Para asegurar el éxito y la difusión de la aplicación, será fundamental implementar estrategias promocionales. Se deberán llevar a cabo campañas de marketing dirigidas tanto a los potenciales usuarios como a la comunidad de desarrolladores de juegos. Esto implica utilizar diversos canales de comunicación, como redes sociales, blogs especializados y eventos relacionados con la industria de los videojuegos, para dar a conocer la aplicación web y resaltar sus características únicas y atractivas.

Impacto económico

DobleTapParty ofrece a los usuarios una alternativa económica para disfrutar de juegos cooperativos con tus amigos. En lugar de invertir en costosos sistemas de consolas de juegos o *hardware* adicional, los usuarios pueden acceder a una amplia variedad de juegos de calidad a través de sus dispositivos móviles existentes. Esto reduce la barrera económica de entrada y permite que un mayor número de personas disfrute de juegos sin tener que hacer grandes gastos.

La plataforma ofrece una oportunidad para que desarrolladores independientes publiquen y obtengan beneficios de sus propios juegos. Esto significa que los usuarios tendrán acceso a

una amplia gama de juegos creados por desarrolladores menos conocidos pero talentosos. Al apoyar a estos desarrolladores, los usuarios pueden disfrutar de experiencias de juego únicas y contribuir directamente a la economía de los creadores independientes.

Sostenibilidad y escalabilidad

Durante el diseño de la arquitectura de la aplicación, se consideró la posible necesidad de escalar los sistemas en un futuro como consecuencia de un aumento en la demanda de recursos. Esto podría ocurrir debido al crecimiento del número de juegos alojados en la plataforma o al aumento en la cantidad de usuarios que utilizaran la aplicación. Para abordar este escenario, se evaluaron y se valoraron los servicios en la nube que ofrecen la flexibilidad necesaria para ajustar los requisitos del sistema y garantizar su capacidad de respuesta a medida que crece.

La elección de servicios en la nube permite adaptar los recursos del sistema de manera dinámica, lo que resulta esencial para manejar eficientemente el aumento en la demanda. Estos servicios en la nube ofrecen la capacidad de escalar horizontalmente, es decir, agregar más servidores o recursos informáticos según sea necesario. Esto asegura que la aplicación satisface las necesidades de almacenamiento, capacidad de procesamiento y respuesta a solicitudes a medida que más juegos se añaden a la plataforma y más usuarios se suman a la comunidad.

Líneas futuras

A pesar del trabajo realizado el proyecto no se encuentra concluido y todavía existen múltiples funcionalidades no implementadas al completo o descartadas a la hora de la realización del TFG por falta de tiempo. Estas mejoras y ampliaciones podrían fortalecer la funcionalidad, el rendimiento y la usabilidad de la aplicación, así como también explorar nuevas características y adaptaciones a diferentes entornos.

A continuación, se detallarán algunas de las funcionalidades y aspectos en los que se podrían realizar trabajos futuros para mejorar y ampliar la aplicación:

- Selección de un proveedor de servicios en la nube: Aunque para el desarrollo se hubiese seleccionado *Azure* como proveedor de servicios con el plan gratuito, en el momento en que se desee realizar el despliegue definitivo de la aplicación será necesario seleccionar un servicio y un plan, ya que el plan gratuito actual tiene múltiples carencias, como por ejemplo la imposibilidad de conectar el *AppService* con un nombre de dominio o la cuota de accesos. Por ello, una de las primeras líneas a decidir sería esta, debido a que un cambio en el proveedor conllevaría una migración de los servicios utilizados.

- *Framework* para desarrollo de juegos multijugador: Para facilitar la incorporación de nuevos juegos por parte de desarrolladores externos se vio la necesidad de incorporar un *framework* que facilitará la incorporación de los juegos dentro de la aplicación, pero por falta de tiempo no se incluyó funcionalidades para adaptar juegos multijugador.

Otra de las líneas para solucionar este problema sería incluir estas funcionalidades al *framework*.

- Mayor cantidad de juegos: Para obtener una mayor masa de usuarios es necesario ofrecer una amplia variedad de juegos, de esta manera sería importante ir introduciendo nuevos juegos en el catálogo.

- Monetización de la aplicación: Otro aspecto importante a tratar para que la aplicación sea viable, es generar una fuente de ingresos a partir de la aplicación. Se debe valorar y desarrollar un método de monetización.

- Plataforma de subida de juegos: Es necesario desarrollar una plataforma que permita a los desarrolladores subir sus juegos y que se desplieguen de manera automática.

- Adaptación de la aplicación para cada usuario: Añadir opciones de idioma y modificaciones de la interfaz en favor de una mayor accesibilidad no implementada en el prototipo inicial.

- Aumentar la cantidad de datos que se recogen de los usuarios: Actualmente se recogen únicamente los datos del usuario necesarios para la utilización de la aplicación. Se podrían añadir información adicional con el objetivo de vender la información relevante o capturar información que permita realizar análisis de uso de la web.

- Adaptar la aplicación a *Smart Tv*: Se ha comprobado que actualmente no es posible ejecutar la aplicación web desde *Smarts Tvs*. Una futura mejora sería encontrar un método que permita ejecutar la aplicación desde este tipo de dispositivos.

- Monetizar la tecnología de vinculación móvil con una pantalla: Aprovechando la tecnología desarrollada de control de pantalla a través de móvil buscar y explotar otros usos diferentes al aplicado en el proyecto.

Modelo de negocio

DoubleTapParty es una plataforma web de videojuegos que ofrece a los usuarios una amplia variedad de juegos emocionantes. Al unirse a la plataforma, los usuarios tienen acceso a una selección de juegos disponibles de forma gratuita desde el principio. Estos juegos iniciales pueden jugarse de inmediato sin restricciones.

Además de estos juegos, *DoubleTapParty* ofrece una amplia gama de videojuegos adicionales que se pueden jugar utilizando una moneda virtual llamada *“Moneda”*. Las *Monedas* se obtienen de forma periódica a medida que pasa el tiempo o viendo anuncios en la plataforma llegando siempre a un límite máximo de *Monedas*.

La plataforma también cuenta con una segunda moneda virtual llamada "*Billete*". Los *Billetes* pueden ser comprados con dinero real o pueden ser adquiridos mediante logros en los juegos, participación en eventos especiales y otras actividades dentro de la plataforma como añadir comentarios o valorar juegos. Los *Billetes* se utilizan para adquirir elementos estéticos para personalizar la experiencia de juego, así como para canjear por beneficios especiales.

Si un usuario desea eliminar la necesidad de usar Monedas para jugar a un juego en particular, puede optar por comprarlo mediante el uso de *Billetes*. Al comprar un juego, el usuario adquiere acceso ilimitado y sin restricciones a ese juego específico, sin tener que gastar Monedas para jugarlo en el futuro.

La plataforma contará con publicidad de tal forma que la visualización de anuncios ofrece una forma adicional de obtención de *Monedas* en pequeñas cantidades para aquellos usuarios que no deseen esperar a que se regeneren.

DoubleTapParty tiene un programa de ofertas de videojuegos, ofreciendo promociones semanales que reduzcan el costo de *Billetes* de los juegos promocionados y algún juego libre de *Monedas* durante esa semana. Este modelo de negocio busca incentivar la participación de los usuarios, generar ingresos a través de compras con *Billetes* y mantener un equilibrio entre la generación de ingresos y la satisfacción de los usuarios.

Otro aspecto a tener en cuenta es el programa de incentivos de referidos ya que se ofrecen recompensas a los usuarios por invitar a sus amigos a unirse a la plataforma. Cuando un usuario invita a un amigo y este se une a la plataforma, ambos reciben *Billetes* a modo de recompensa, fomentando así el crecimiento de la base de usuarios y fortaleciendo la comunidad

Finalmente, hay que destacar que existen suscripciones premium disponibles que se pueden adquirir con *Billetes*. Estas suscripciones ofrecen a los usuarios acceso al catálogo completo de juegos sin restricciones de *Monedas* durante un periodo de tiempo limitado.

Bibliografía

Django: The web framework for perfectionists with deadlines. Retrieved July 5, 2023, from https://www.djangoproject.com/

Ruby on Rails — A web-app framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern. Retrieved July 5, 2023, from https://rubyonrails.org/

Microsoft Learn: Build skills that open doors in your career. Retrieved July 5, 2023, from https://learn.microsoft.com/en-us/

Bedi, J. (2019, 2 4). Ralph Baer: An interactive life. *Human Behavior and Emerging Technologies*, *1*(1), 18-25.

Belli, S., & López Raventós, C. (2008). Breve historia de los videojuegos. *Athenea Digital. Revista de Pensamiento e Investigación Social*, (14), 159-179.

Diskin, P. (2004). Nintendo Entertainment System Documentation. *Tokyo: Nin-tendo*.

Fisher, S. R. A., & Yates, F. (1963). *Statistical tables for biological, agricultural and medical research* (6º ed.). Oliver & Boyd.

Foster, W., & Oppermann, T. (Eds.). (1996). *The Information Superhighway and Private Households: Case Studies of Business Impacts*. Physica-Verlag HD.

Gorman, B. L. (2021). *Practical Entity Framework Core 6: Database Access for Enterprise Applications*. Apress.

Ingebrigtsen, E. (2013). *SignalR: Real-time Application Development*. Packt Pub.

Johnson, K. L., Carr, J. F., Day, M. S., & Kaashoek, M. F. (2001). The measured performance of content distribution networks. *Computer Communications*, *24*(2), 202-206.

Lendino, J. (2018). *Adventure: The Atari 2600 at the Dawn of Console Gaming*. Steel Gear Press.

Martínez, J. C., Henao, C., Henao, F., & Zapata, E. (2021). Utilización de Arquitecturas Limpias para Trabajo con Buenas Prácticas en la Construcción de Aplicaciones Java. *Innovación Digital y Desarrollo Sostenible-IDS*, *1*(2), 133-140.

Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., & Xu, Z. (2003). *Peer-to-Peer Computing*.

Obtenga información sobre .NET | Tutoriales gratuitos, vídeos, cursos y mucho más. Microsoft .NET. Retrieved July 5, 2023, from https://dotnet.microsoft.com/es-es/learn

Parente, D. (2012). Pasado, presente y futuro de los videojuegos. *Bit*, (190), 9.

Precios | App Engine. Google Cloud. Retrieved July 5, 2023, from https://cloud.google.com/appengine/pricing?hl=es

Precios de App Service. Microsoft Azure. Retrieved July 5, 2023, from https://azure.microsoft.com/es-es/pricing/details/app-service/windows/

Scullion, C. (2021). *The Sega Mega Drive and Genesis Encyclopedia: Every Game Released for Sega's 16-Bit Console*. Pen & Sword Books Limited.

Servidor privado virtual (VPS), precios de alojamiento web-Amazon Lightsail-Amazon Web Services. Amazon AWS. Retrieved July 5, 2023, from https://aws.amazon.com/es/lightsail/pricing/

SVG Tutorials. Peter Collingridge. Retrieved July 5, 2023, from https://petercollingridge.co.uk/tutorials/svg/

Unity User Manual (2019.4 LTS) - Unity Manual. Unity - Manual. Retrieved July 5, 2023, from https://docs.unity3d.com/es/

Vakali, A., & Pallis, G. (2003). Content delivery networks: Status and trends. *IEEE Internet Computing*, *7*(6), 68-74.

Vaudour, F., & Heinze, A. (2020). Software as a service: Lessons from the video game industry. *Global Business and Organizational Excellence*, *39*(2), 31-40.

Wolf, M. J., & Perron, B. (2005). *Introducción a la teoría del videojuego*. Formats: revista de comunicació audiovisual.

Wu, D., Xue, Z., & He, J. (2014). iCloudAccess: Cost-effective streaming of video games from the cloud with low latency. *IEEE Transactions on Circuits and Systems for Video Technology*, *24*(8), 1405-1416.

Xuew, Z., Wu, D., He, J., Hei, X., & Liu, Y. (2014). Playing high-end video games in the cloud: A measurement study. *IEEE Transactions on Circuits and Systems for Video Technology*, *25*(12), 2013-2025.

Ludografía

Mot's Grand Prix	Videojuego	Tom Mulgrew, 2022
Sandcastles	Videojuego	Eevee, 2018
Pico Sonic	Videojuego	Komehara, 2023
Ruby's Adventure: 2D Beginner	Videojuego	Unity Technologies, 2018
Cards Against Humanity	Juego de mesa	Cards Against Humanity LLC, 2011