(VectorStream)

DESARROLLO DE UNA PLATAFORMA DE STREAMING MULTIDISPOSITIVO DE VIDEOJUEGOS

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Agosto de 2022

Autor:

Jesús Manuel García Prieto

Tutores:

Diego J. Valdeolmillos Villaverde

Alfonso González Briones

Francisco Pinto Santos

AUTORIZACIÓN

Dr. D. Alfonso González Briones profesor del departamento de Informática y Automática de la Universidad de Salamanca, D. Diego Valdeolmillos Villaverde personal laboral de AIR-Institute y D. Francisco Pinto Santos Personal Docente e Investigador de la Universidad de Salamanca

HACEN CONSTAR

que el presente documento titulado “Desarrollo de una plataforma de streaming multidispositivo de videojuegos” ha sido realizado por Jesús Manuel García Prieto, con DNI 70966306J y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática.

Y para que así conste a efectos oportunos.

En Salamanca, a martes, 6 de agosto de 2022

RESUMEN

En una actualidad en la que la industria de la tecnología se encuentra en constante auge y evolución es solo lógico que, con el paso del tiempo, surjan nuevas formas de juntar el ocio con tecnología. En su día, los videojuegos fueron una de estas nuevas maneras de comprender el ocio y la diversión y, al igual que todas las otras que existen, no han sido inmunes a esta evolución sin fin.

El juego en streaming es una de las últimas reinvenciones de la industria del videojuego y su aparición no hace más que demostrar lo dicho previamente.

Eliminando la barrera de las especificaciones del ordenador del usuario, las plataformas de streaming de videojuegos logran proporcionar este servicio con tan solo una buena conexión a internet, proporcionando así una solución a todos aquellos que no pueden permitirse una computadora de última gama o un ordenador de gama gaming.

Este proyecto nace con la intención de investigar esta nueva forma de ocio, entender su funcionamiento y la tecnología que permite su correcta utilización.

El primer punto fundamental del mismo será entender la tecnología WebRTC, que abarca toda la complejidad y funcionalidad de capturar y enviar en formato de vídeo la pantalla de una computadora a otra. Gracias a esto y a la captura de inputs se puede crear una plataforma de streaming en la nube que abstraiga al usuario del hecho de estar jugando a un videojuego simulado.

Con el sistema completamente desarrollado se podrá soportar a grandes cantidades de usuarios simultáneos, cada uno de ellos utilizando la plataforma de manera individual sin notar ninguna colisión entre ellos, creando así una auténtica plataforma de streaming de videojuegos que ofrezca a los usuarios, de la manera más intuitiva y eficiente, los servicios que se cabe esperar de una plataforma de este tipo.

Palabras clave: *Streaming, input, ocio, gaming, video.*

ABSTRACT

Living on a present where the technology industry is suffering a steady growth and evolution it's just a matter of fact that, with time, there will be new ways of merging recreation and technology. Back in the day, videogames were one of those new ways of acknowledging leisure and fun and, just like all of the other ones, they weren't immune to this never-ending evolution.

Game streaming is one of these new reinventions of the videogame industry, and its emergence has done nothing more than proving that entertainment keeps reinventing itself.

By removing the user's computer specifications barrier, videogame streaming platforms manage to provide this service simply by having a stable network connection, providing a solution for all of those who can't afford a top notch nor a gaming computer.

This project was born with the intention of researching this new form of entertainment, understanding its performance and the technologies that allow it to work properly.

The first main point about it will be understanding WebRTC's technology, which covers all of the complexity and functionality of recording and sending a video of the captured screen from one computer to another. Thanks to this and to the capture of inputs, a streaming web platform that abstracts the user from the fact that he is playing a simulated videogame can be designed.

With this system completely developed, great quantities of simultaneous users can be handled, each one of them using the platform on their own, without noticing any collision between them, thus, creating a genuine videogame streaming platform which provides users, in the most intuitive and effective way, all of the services that a platform like this is expected to provide.

Keywords: *Streaming, WebRTC, leisure, gaming, video.*

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El mundo de los videojuegos es, innegablemente, una de las ramas del entretenimiento más fructíferas en la historia contemporánea de la humanidad. Desde que estos dieron sus primeros pasos allá por los años 50, siempre han buscado reinventarse y alcanzar horizontes nuevos, lo cual los ha llevado a alcanzar su popularidad actual, y es que prácticamente la mitad de los hogares poseen una videoconsola y 3 de 10 planean adquirir una (MarketingCharts, 2021).

Esta popularidad se traduce en un amplio surtido de opciones que permiten a jóvenes y adultos acceder a este medio de entretenimiento, sin embargo, uno de los mayores problemas que este presenta es la propia plataforma en la que el juego es ejecutado. El aumento de precio de las tarjetas gráficas para PCs debido a la compra masiva de estas por *miners* de cripto monedas (Kühne, 2020) sumada a la falta de *stock* derivada de la Covid-19 ha generado una gran traba para compradores estándar ajenos al minado y al mundo de la *blockchain*.

Es por esto que, en tiempos recientes, las plataformas de streaming de videojuegos han hecho su aparición en el mercado, su premisa pretende solucionar este problema recién mencionado, su método de uso consiste en pagar una suscripción (de carácter mensual, generalmente) para acceder a sus servicios, una vez adquirido, la plataforma te permite jugar a cualquier videojuego de su catálogo. Hasta este punto, estas plataformas no tienen ninguna diferencia con otros servicios como *Game Pass* (Microsoft, 2017) o *PlayStation Plus* (Sony, 2010), sin embargo, estas nuevas plataformas presentan una novedad fundamental: en lugar de adquirir el videojuego seleccionado e instalarlo en el PC del cliente, estas plataformas de streaming, y de ahí viene su nombre, ejecutan el juego en uno de sus ordenadores de alta gama, transmitiendo al cliente la imagen del juego ejecutado, de esta manera, un cliente con un ordenador de gama baja puede disfrutar de aquellos videojuegos más exigentes en cuanto a requisitos técnicos simplemente con una buena conexión a internet.

Si bien, como ya se ha mencionado, es un nuevo tipo de plataforma, no por ello existen pocas, en estos últimos años su aparición ha aumentado exponencialmente y se ha pasado de tener una abanico limitado de opciones a un amplio surtido, algunas de las más populares son *Google Stadia* (Google, 2019), especialmente destacada por ser la plataforma de streaming pionera y por el gran punto a favor de estar sincronizada con Youtube por pertenecer ambas plataformas a Google, *Nvidia GeForce Now* (Nvidia, 2020), que destaca por su gran compatibilidad con las tarjetas gráficas de la marca, o *Amazon Luna* (Amazon, 2022), que se trata de la última gran aparición en el mercado y destaca por su coalición con Twitch juntando así los dos mundos del streaming más populares.

Como conclusión, el éxito reciente del formato de las plataformas de streaming de videojuegos está provocando que todas las compañías relacionadas con el mercado tecnológico estén intentando lanzar su propia versión (Keck, 2019), además, al tratarse de una tecnología muy reciente, aún sigue habiendo muchas posibilidades por descubrir y aplicar a este formato.

Es por ello que se presenta “VectorStream”, una plataforma de streaming de videojuegos basada en la tecnología que ofrece Electron para generar aplicaciones de escritorio y en la herramienta WebRTC (utilizada por *Google Meets* o *Discord*, entre otras) que nos permite compartir pantalla entre dispositivos, además de tratarse de un proyecto de código libre, lo cual facilita la búsqueda de otros proyectos de ejemplo.

Gracias a esto se pretende desarrollar una herramienta basada en la escalabilidad y la concurrencia de usuarios, orientada al ámbito de los videojuegos. Se servirá de la famosa arquitectura cliente-servidor para el tratamiento y el envío de datos entre el equipo emisor (fuente) y el receptor (cliente).

Para lograr esto existirán dos aplicaciones, la aplicación de escritorio generada con Electron (como ya se mencionó) y el backend del servidor gestor.

El nombre del proyecto "VectorStream" proviene del concepto de *stream* de datos, como es lógico, mientras que "vector" proviene del concepto de imagen vectorial, la cual mantiene la nitidez y la definición de la misma, logrando que la calidad sea mucho mejor. Usando esta definición como símil entre imagen y vídeo, el nombre de la plataforma "VectorStream" pretende indicar que la calidad de vídeo será nítida como la de una imagen vectorial.

En cuanto a la estructura de esta memoria, en primer lugar, se hablará de los objetivos del proyecto, así como de los diferentes conceptos teóricos relacionados con el mismo (principalmente, conceptos relacionados con el streaming de datos), después de esto se citarán y describirán aquellas metodologías, técnicas y herramientas utilizadas en el proyecto (tanto en su desarrollo como en su documentación). Se presentarán también aquellos aspectos relevantes del desarrollo del sistema y, por último, una sección de resultados, conclusiones y posibles adiciones futuras.

De manera aclarativa, este documento se encuentra completado por los siguientes anexos, estos se pueden consultar para ver detalles a mayor granularidad y también aquellos de caráctermás técnico.

- **Anexo I: Plan del proyecto:** Estimación y planificación temporal del proyecto, análisis del Diagrama de Gantt generado.
- **Anexo II: Especificación de requisitos software:** Elicitación de requisitos del sistema, especificación de objetivos y distribución de casos de uso por paquetes.
- **Anexo III: Análisis de requisitos:** Diagrama de clases del sistema, análisis de los casos de uso definidos previamente mediante el uso de diagramas de secuencia.
- **Anexo IV: Diseño del sistema software:** análisis de los casos de uso en el dominio de la solución mediante diagramas de secuencia, definición de interfaces y del diseño de la interfaz de usuario.
- **Anexo V: Documentación técnica**
- **Anexo VI: Manual de usuario:** Detalle de las vistas de la aplicación y descripción de la funcionalidad que estas ofrecen.

2. ESTADO DEL ARTE

2.1. ELEMENTOS VISUALES

A la hora de realizar un análisis de plataformas similares a la idea a desarrollar, se deben hacer dos preguntas: ¿Hace lo que queremos que haga nuestro proyecto? Y ¿Se ve como queremos que se vea nuestro proyecto?

Es por esto que la búsqueda se dividió con dos objetivos claros en mente: buscar interfaces atractivas para el público relacionado con el sector de los videojuegos y encontrar plataformas útiles desde el punto de vista del desarrollo, que nos proporcionen ejemplos lo suficientemente útiles como para incorporar algo similar en nuestro sistema.

La primera plataforma en ser investigada fue, como es fácil imaginar, Google Stadia, esto se debe a que es la plataforma de juego en streaming más popular en el momento de realizar este proyecto, al tratarse además de una plataforma tan grande, el objetivo al analizar Google Stadia sería tanto el de buscar herramientas como el de buscar interfaces.

El primer elemento que llamó la atención al investigar Stadia fue su manera de presentar los videojuegos de su catálogo, en la **Figura 1**se puede observar una de las Cards que Stadia utiliza para sus videojuegos (Google, 2019).

Figura 1. Detalle de catálogo e iconos de juego de Stadia

Gracias a esto, como se verá más adelante, se tomó la decisión estética de diseñar un catálogo conformado por iconos de videojuegos estilizados, en este caso, el notch presente en el icono de Stadia acabaría derivando en el uso de la propiedad clip path de CSS para conseguir un efecto similar.

En cuanto a funcionalidad, Stadia no proveyó mucha información, como es sabido, las aplicaciones de Google suelen guardar con mucho secretismo su implementación, por lo que, Stadia quedaría en un segundo plano a la hora de tomarla como ejemplo, aunque como estándar a nivel visual fue de gran ayuda como ya se ha explicado.

Una vez investigado WebRTC, se decidió seguir buscando plataformas de videojuegos que ayudasen a la hora de diseñar la interfaz, la siguiente en la lista sería la famosa plataforma Steam (Valve, 2003), como podemos observar en la **Figura 2** se trata de una plataforma con un amplio número de funcionalidades relacionadas con la socialización (Valve, 2003), sería de aquí donde se tomaría la idea de implementar una Navbar.

Figura 2. Barra de navegación de Steam

Como se puede observar, la Navbar divide claramente la zona de navegación (abajo a la izquierda) con la zona de funcionalidades sociales (arriba a la derecha), este concepto será utilizado en la navbar integrada en el proyecto, como se verá más adelante.

Una vez detallados los elementos principales del sistema, se procedió a definir con mayor detalle cómo se diseñarían, para ello se realizaron búsquedas más concretas sobre elementos que se decidieron incluir en el sistema. El primero de estos sería el catálogo y, ya que el sistema a diseñar tiene tantos lazos con el mundo de los videojuegos, se decidió investigar interfaces de videojuegos.

La primera de todas que se investigó fue la interfaz del videojuego League of Legends (Riot, 2009), esta se decidió estudiar debido a que dentro del propio videojuego existe una sección de galería, esta se puede observar en la **Figura 3**.

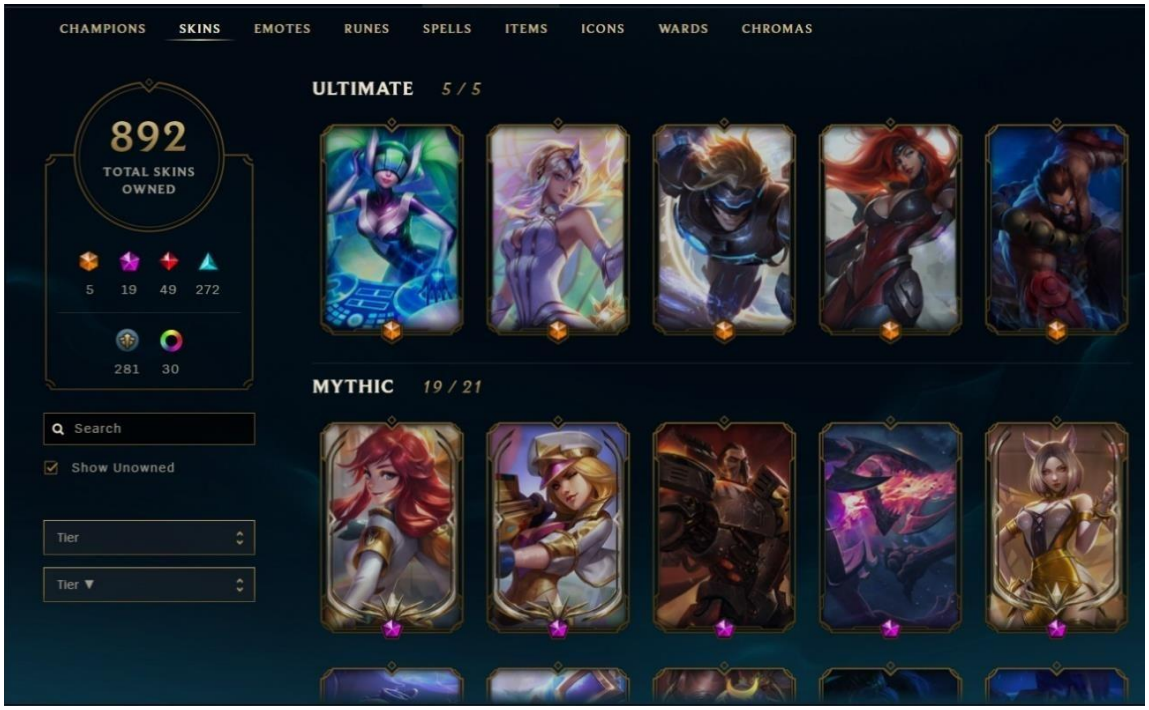


Figura 3. Interfaz de colección de skins en el videojuego League of Legends

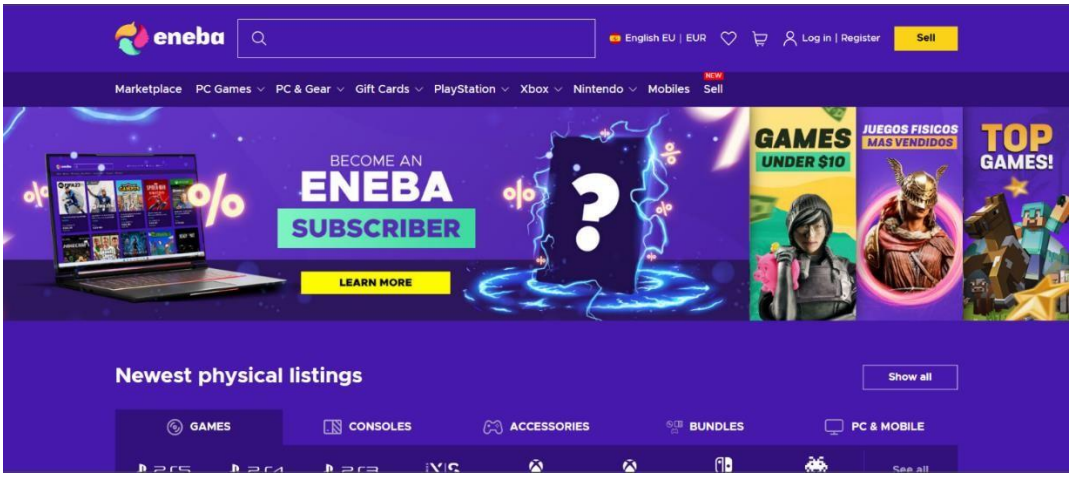
El punto más destacado de la interfaz de colección de skins es la división de las mismas, como se puede ver, la división consiste en la categoría (en este caso, la rareza de la skin) seguida de una serie de tarjetas con la imagen de la skin en cuestión, y, para separar categorías, se vale del uso de un separador. Esta estructura de separación de categorías será utilizada en la versión final como se verá más adelante.

Otros dos puntos que se obtuvieron de esta referencia fueron los dos filtros existentes en el catálogo, uno de ellos se trata de una barra de búsqueda manual y, el otro, se trata de un selector de categorías, ambos serán utilizados también.

Por último, en cuanto a lo que ofrece esta interfaz, se dejaron como tentativas la idea de añadir un contador de videojuegos a modo de estadística para que el usuario sepa cuántos tiene y de qué género con tan solo un vistazo.

El siguiente punto a investigar sería la paleta de colores a utilizar, para esto se investigaron varias webs, las conclusiones sacadas fueron las siguientes:

- El modo principal de la aplicación sería el modo oscuro, este fue elegido ya que a día de hoy está obteniendo mucha popularidad y casi todas las aplicaciones lo incluyen (Ferrer, 2022), además, gracias a frameworks como Bootstrap es mucho más sencillo crear una aplicación con este modo sin mayores esfuerzos.
- Partiendo de esta conclusión, se probó a detallar una paleta de colores para la aplicación, sabiendo que se desarrollaría en modo oscuro, se buscaron aplicaciones con modo oscuro para ver qué colores implementaban.
- primera aplicación buscada fue Eneba, si bien no se trata de un modo oscuro de por sí, la interfaz tiene un fondo monocolor azul/morado que puede ir bien si el resto de elementos son negros o están en modo oscuro. Podemos observar la interfaz de Eneba en la **Figura 4** (Eneba, 2018):



La

Figura 4. Interfaz gráfica de Eneba

Como se verá en la aplicación final, este fondo morado será utilizado para el catálogo, en conjunto con el resto de elementos que estarán en formato oscuro, con esto tendríamos ya el primer color de la paleta.

Basándose en el morado y el negro, que se trataban de los dos únicos colores decididos, se tuvo la idea de utilizar otros dos colores fuertes que pegasen con el negro y el morado, observando logos de aplicaciones similares se observó que el morado, azul y verde eran tres opciones muy populares a la hora de hacer logos, se pueden observar algunos ejemplos en la **Figura 5:**

Figura 5. Logos de aplicaciones de streaming

Con este análisis, se decidió crear una paleta para la aplicación, en la **Figura 6** se puede observar la paleta final utilizada:

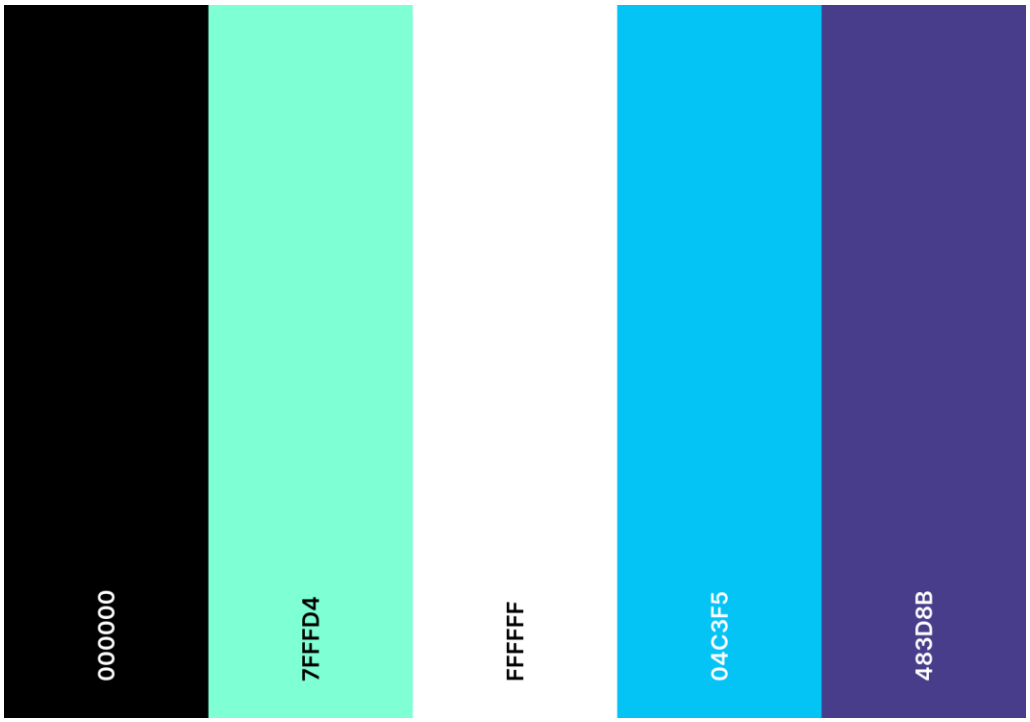


Figura 6. Paleta oficial del proyecto

2.2. ELEMENTOS TÉCNICOS

Habiéndose decidido, en su mayor parte, la interfaz gráfica y su estructura gracias a las referencias investigadas en el apartado anterior, el siguiente paso en esta etapa será el de encontrar qué elementos técnicos (herramientas, hardware, software) utilizados por plataformas similares a la que se va a desarrollar.

Se partirá de saber de antemano que WebRTC será la tecnología a utilizar para retransmitir, por lo que se investigarán otros elementos de la aplicación.

El primer elemento investigado será el hardware, en concreto, controladores (teclado, mandos, ratón) para el usuario, el punto principal a descubrir será comprobar si las diferentes plataformas de streaming proporcionan su propio hardware de control o si prefieren utilizar hardware de otras plataformas estandarizado.

La primera plataforma sometida a esta investigación sería el recientemente estrenado Amazon Luna (Amazon, 2022), el cual proporciona su propio controlador, esto se puede observar en la **Figura 7:**

Figura 7. Mando oficial de Amazon Luna

Navegando plataforma por plataforma se descubrió que absolutamente todas las sometidas a investigación poseían un controlador propio, como es el caso de Amazon Luna, o, las pocas que no, como Nvidia Geforce Now (Nvidia, 2020), aun así, recomendaban controladores a utilizar de marcas asociadas.

Debido a esto, se decidió implementar un sistema propio de control de inputs.

Por último, en esta sección de estado del arte, se ha decidido hablar sobre algunas de las plataformas de streaming más populares a día de hoy, así como los motivos de su éxito y, lo que más interesa en este apartado: analizar en cómo se puede valer de esta información el proyecto a desarrollar.

- En primer lugar, se hablará de la plataforma de streaming por excelencia: Twitch. Si bien, a pesar de ser la plataforma de streaming más popular, no se ha hablado de ella hasta ahora debido a que se trata de una plataforma de streaming social, es decir, los usuarios fuente o "streamers" se graban en directo para una audiencia, es un matiz diferente al proyecto que se debe implementar en este TFG, por lo que solo no es útil hablar de Twitch hasta cierto punto, concretamente, se hablará de la tecnología que implementa a la hora de retransmitir vídeo y audio. Twitch, concretamente, utiliza un sistema de transcoding, del cual hablaremos en detalle más adelante, a grandes rasgos esto significa que la retransmisión cambia de protocolo de envío a lo largo de su proceso de transmisión, concretamente, Twitch se basa en el combo de RTMP y HLS, además de esto, para mantener el chat en vivo, Twitch se sirve del protocolo IRC que, además, permite a los creadores de contenido añadir sus propios bots para los chats. (Twitch, 2015).

Además de esto, cabe destacar que Twitch, en cuanto a cifras, se trata de una plataforma que ha sufrido un enorme boom, por lo que, es un gran ejemplo a la hora de observar una plataforma que ha logrado adaptarse a un problema mayor de escalabilidad.

- La siguiente plataforma a investigar se trata de Google Meet, a diferencia de Twitch, esta plataforma posee un enfoque profesional, orientado a utilizarse en reuniones de diferentes ámbitos, por lo que su principal problema será el de la usabilidad y el de la calidad de la

retransmisión, ya que su público es mucho más diverso que el de Twitch, el cual, suele tratarse de un público joven casi exclusivamente.

Google Meet, por tratarse de un programa creado por Google, utiliza el propio protocolo diseñado por ellos mismos, este protocolo es el ya mencionado WebRTC, uno de los puntos más atractivos es que no necesita enviar el vídeo de manera enrutada por un servidor intermedio, puede enviarlo de peer a peer directamente, lo cual mejora la latencia de la retransmisión, además de esto no incluye plugins adicionales y, además de esto, es compatible con prácticamente todos los navegadores web más populares. (Nguyen, 2020).

2.3. REVISIÓN DEL ESTADO DEL ARTE

Tras observar a nivel técnico y visual todas estas plataformas, se puede concluir diciendo que la gran mayoría de ellas se centran en el modo oscuro y poseen un controlador físico propio. Sin embargo, hay varios factores que, si bien hemos tomado como referencia de varias plataformas, ninguna de ellas los posee integrados de manera simultánea, es por ello que se plantean los siguientes puntos que definirán al proyecto con respecto al resto de plataformas:

- La plataforma funcionará a su vez como plataforma y como una pequeña red social, integrando todos los elementos propios del streaming junto a elementos presentes en varias páginas que funcionan como tienda de videojuegos o launchers (Steam, Epic Games Store, etc), de esta manera el proyecto alcanzaría un alto grado de ampliación futura, pudiendo llegar a ofrecer un sistema de tienda y streaming, similar a Stadia, junto a funciones sociales más complejas como un chat con amigos, partidas multijugador con ellos, etc.
- Como ya se ha comentado, la mayoría de plataformas de streaming ofrecen su propio mando para utilizarlas e, incluso las que no, recomiendan mandos en particular para un mejor control, en este proyecto se ha decidido eliminar esto y ofrecer un control a partir de ratón y teclado, de esta manera ningún usuario se vería privado de utilizar la plataforma por no poder permitirse el mando propio de la misma.

3. OBJETIVOS DEL PROYECTO

3.1. OBJETIVOS TÉCNICOS

El principal objetivo del proyecto es ofrecer a los clientes un sistema de retransmisión en remoto de videojuegos, pudiendo estos jugar al juego deseado de manera online sin preocuparse por las especificaciones de su equipo. Para cumplir este objetivo, y alguna que otra funcionalidad secundaria, se presentaron los siguientes objetivos del proyecto:

- **Gestión de Streaming multimedia:** El sistema debe ser capaz de capturar la pantalla del equipo fuente y transmitirla en tiempo real al equipo cliente, el cual mostrará dicha imagen al usuario. Los pasos a seguir para lograr esto son los siguientes:
    - El equipo fuente crea el socket y establece sus parámetros.
    - El equipo fuente captura su pantalla y establece parámetros de retransmisión como el tamaño de la pantalla o el audio.
    - Mediante el ID de la room recién creada, el equipo fuente crea las manejadoras para los eventos de descubrimiento y respuesta
    - Una vez recibido el evento descubrimiento de vuelta por parte del servidor, se conecta el equipo fuente al servidor y se añade su emisión de vídeo a la lista de vídeos.
    - Si en cualquier momento se recibe alguna petición por parte del usuario, se acepta la petición y se le muestra el stream de vídeo.
    - Se crean las manejadoras para los inputs del cliente, esto no es relevante para la retransmisión en sí, pero lo será para otro de los objetivos del sistema.
- **Gestión de usuarios:** El sistema debe ofrecer a los usuarios una experiencia de usuario cómoda, incluyendo los elementos más comunes como un sistema de registro, login y logout, así como algunos elementos más complejos como un perfil propio o la posibilidad de modificar sus datos personales.
  Para lograr esto se ha decidido crear una base de datos en el servidor del sistema, la cual será una base de datos PostgreSQL, orientada para almacenar los datos relativos alos usuarios, así como las relaciones entre estos.
- **Gestión de catálogo:** El sistema debe ser capaz de listar en la interfaz de usuario los diferentes videojuegos que conforman el catálogo, así como un sistema de búsqueda que incluya un filtro y una barra de búsqueda para facilitar el uso del catálogo al usuario.
- **Gestión de interacción con dispositivos físicos:** Se trata del punto mencionado en el objetivo “gestión de streaming multimedia”, gracias a las manejadoras creadas al final de la comunicación fuente-usuario, el usuario es capaz de enviar su señal de ratón (posición, click izquierdo y derecho, ruleta) y su señal de teclado al equipo fuente. Así es como el sistema logra ocultar al cliente que realmente se encuentra jugando en una máquina externa a la suya
- **Gestión de actividades sociales:** El sistema debe ofrecer al cliente una experiencia social básica, incluyendo una lista de amigos y una lista de usuarios bloqueados, ambas siendo modificables por el cliente y ambas encontrándose almacenadas en la base de datos PostgreSQL del servidor ya mencionada anteriormente.

3.2. OBJETIVOS PERSONALES

Una vez los objetivos técnicos del proyecto se han expuesto, es hora de tratar aquellos objetivos relativos a metas personales.

En primer lugar, el TFG: "plataforma streaming multidispositivo de videojuegos" fue seleccionado por varios factores, pero uno de aquellos que se consideraron determinantes fue el hecho de tratarse de un proyecto relacionado con el ámbito de los videojuegos y, además de esto, un proyecto muy ajeno al conocimiento del programador por aquel entonces, los lenguajes de programación solicitados eran todos de interés y, sin embargo, no había experiencia previa en ellos, por lo que, para lograr obtener estos conocimientos, se decidió escoger este proyecto (cabe aclarar que después de seleccionar este proyecto el programador fue instruido durante su periodo de prácticas de empresa en algunas de las tecnologías que finalmente utilizaría en el diseño del proyecto posteriormente).

Otro de los objetivos principales era el de lograr realizar una plataforma funcional en su totalidad, siguiendo todos los pasos para esto, en concreto, la idea de integrar un servidor y una aplicación web (en este caso de escritorio, pero la idea se mantiene) configurados en la mayor medida de lo posible por el programador era uno de los puntos a lograr y que más se buscaba realizar pues así se obtendría una primera experiencia en un proyecto de estas proporciones.

4. CONCEPTOS TEÓRICOS

En esta sección serán descritos aquellos conceptos teóricos fuertemente asociados con el sistema a desarrollar, no se debe confundir con las herramientas utilizadas en él, pues estas serán descritas y listadas más adelante, lo relativo a este apartado serán conceptos como los estilos o modelos arquitectónicos utilizados o conceptos asociados al envío y recepción de datos.

4.1. SISTEMA DISTRIBUIDO

Este se trata del primer concepto a presentar en esta memoria debido a su alta implicación con el proyecto, se podría considerar al mismo como un gran sistema distribuido, por lo que no tiene sentido continuar explicando otros conceptos más concretos sin explicar este en primer lugar.

Un sistema distribuido se trata de un conjunto de equipos físicos interconectados entre ellos mediante una red de trabajo (*network)*. El objetivo de esta división no es otro que repartir las tareas a realizar para agilizar su finalización. La importancia de su existencia proviene de la, cada vez mayor, cantidad de tareas masivas y altamente complejas que requieren de varios equipos para poder ser gestionadas.

La implementación actual más común para un sistema distribuido es comunicando a los diferentes equipos mediante internet, un diagrama de un pequeño sistema distribuido sin internet puede ser observado en la **Figura 8** (Splunk, 2022):

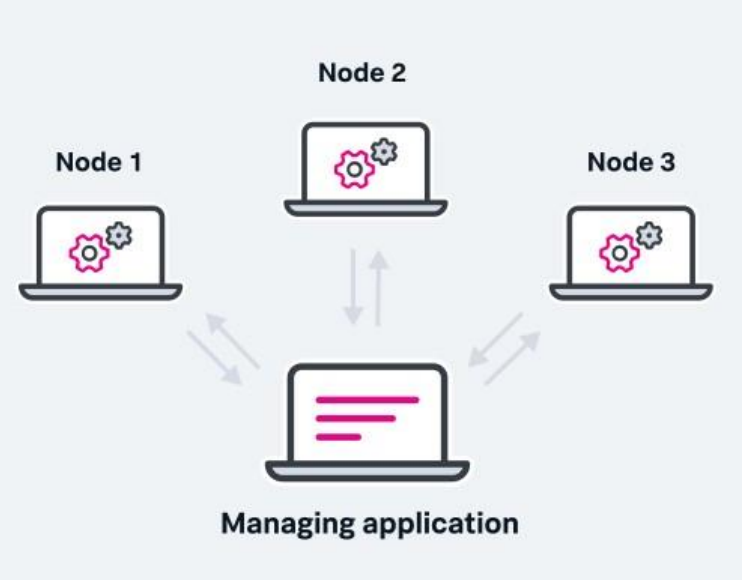


Figura 8. Sistema distribuido básico

Sin embargo, a pesar de haber descrito las maneras más básicas y más populares de implementar un sistema distribuido, existen muchas otras, la que se destacará, por ser la implementada en el sistema actual, es la arquitectura cliente-servidor, como ya explicaremos más adelante.

Algunas de las características que mejor definen a los sistemas distribuidos se pueden encontrar listadas a continuación (Splunk, 2022):

- **Concurrencia:** Los componentes de un sistema distribuido se encuentran en ejecución simultáneamente, no existe la presencia de un reloj global que dicte el ritmo de realización de las tareas, cada nodo opera a su ritmo establecido.

- **Escalabilidad:** Se trata de la característica fundamental de los sistemas distribuidos y el porqué de su extendido uso, tienen una altísima capacidad de cambio a la hora de admitir nuevos nodos para aumentar su tamaño según las tareas a realizar se vayan volviendo más complejas con el paso de los años.
- **Tolerancia a fallos:** En caso de que se produzca una falla en el sistema, será mucho más fácil encontrar el nodo del que proviene el problema y aislarlo para solucionarlo en lo que el resto de nodos opera con normalidad.
- **Transparencia:** Un usuario ajeno al sistema lo verá como si este se tratase de una sola unidad, sin conocer el resto de partes que componen al entorno, haciendo así que los usuarios puedan estar despreocupados de la arquitectura existente en el sistema.
- **Heterogeneidad:** Generalmente, en un sistema distribuido los nodos se encuentran realizando una amplia diversidad de tareas propias, incluso con diferente hardware o software ente ellos, esto permite que estos sistemas tengan una amplia flexibilidad.

Con todo esto, cabe entender que los sistemas distribuidos traen una amplia cantidad de beneficios consigo mismos, algunos de ellos ya han sido mencionados como, por ejemplo, su alta flexibilidad, o su fiabilidad al tratarse de sistemas distribuidos bien construidos. Algún otro beneficio a destacar es su alta eficiencia, pues al dividir tareas entre varios equipos la carga de trabajo de las bases de datos, por ejemplo, puede verse reducida, llegando así a soportar entornos de trabajo masivos.

Sin embargo, no todo son beneficios, ya que contraen algún riesgo al ser utilizados (Splunk, 2022):

- **Seguridad:** Un sistema distribuido es igual de propenso a ser atacado que cualquier otro sistema, sin embargo, al aumentar la superficie del sistema debido al número de nodos que este conlleva, también se aumenta la superficie sobre la que este sistema puede ser atacado.
- **Coste:** Mantener aquellos sistemas distribuidos de mayor complejidad conlleva una carga de mantenimiento enorme, lo cual se traduce también en una cantidad de costes mucho más elevada que en un sistema convencional.
- **El desafío de la sincronización:** Al tratarse de sistemas que no poseen un reloj global, los sistemas distribuidos deben poseer mecanismos de sincronización en aquellos nodos que requieran comunicarse entre ellos, crear redes de este tipo puede resultar desafiante y una limitación a la hora de escalarlos.

4.2. ARQUITECTURA CLIENTE-SERVIDOR

Se trata del estilo arquitectónico que definirá al sistema, se trata de uno de los estilos más conocidos y utilizados, su principal característica es su clara división del sistema en un cliente y un servidor o proveedor. El cliente será el encargado de solicitar servicios y recursos al proveedor y, posteriormente, consumirlos (ReactiveProgramming, 2022).

La conexión entre ambos, en este caso, viene dada por el protocolo SSH, encargado de proporcionar un canal de comunicación fiable.

Otra de sus características fundamentales viene dada por la definición de sistema distribuido, ya que el cliente y el servidor se encuentran distribuidos en diferentes equipos comunicados por mediante una red.

Gracias a este estilo arquitectónico se facilita enormemente la división del sistema, evitando así la centralización del mismo y pudiendo separar los datos (o el modelo del sistema) de la vista del mismo.

Existen varios tipos de arquitecturas cliente-servidor, en función del parámetro seleccionado, en este caso se hablará de los tipos de arquitectura en función del tamaño de los elementos, yaque es la que mejor hablará del sistema realizado, existen dos tipos de arquitectura cliente- servidor siguiendo este criterio:

- ***Fat client (thin server)**:* Se trata de una arquitectura cliente-servidor en la cual el cliente ejecuta el gran grueso de tareas y el servidor queda relegado a un papel secundario, como puede ser el caso de un servidor que funcione únicamente como base de datos.
- ***Fat server (thin client):*** Se trata del caso completamente opuesto al anterior, es en este tipo de arquitectura cliente-servidor en la que nos encontramos con un servidor que gestiona el gran grueso de tareas, proporcionando así más flexibilidad, mientras que el cliente sirve únicamente como una vista para el cliente con un nivel mínimo de funcionalidad. Nuestro sistema caería bajo esta definición de arquitectura cliente-servidor.

(González Reyes, Betty Carvajal, Manrique Neira, Quijije Lucas, & Quijije Toro, 2017)

En la **Figura 9** se puede observar una definición a modo de diagrama de la arquitectura cliente-servidor en su modalidad *fat client* y *fat server* (Arnold, 2007):

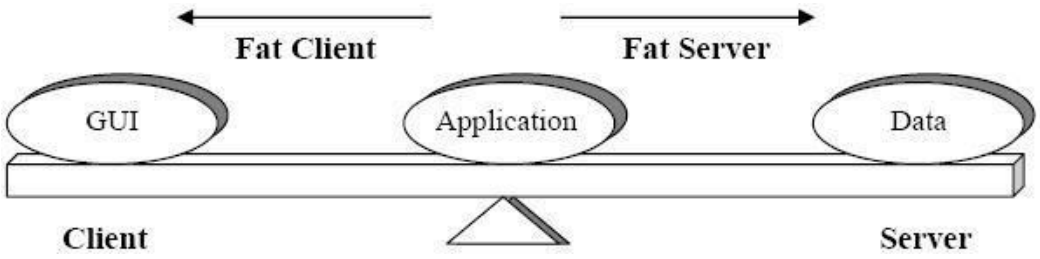


Figura 9. Modelo fat client y fat server

4.3. SERVIDOR WEB

Habiendo definido el tipo de arquitectura del sistema a desarrollar, explicar en detalle qué tipo de servidor se utiliza es el siguiente paso lógico. El servidor utilizado se trata de un servidor web, estos servidores se caracterizan por compartir datos con otros elementos conectados a la web, generalmente, clientes. El servidor web se encarga de responder las peticiones realizadas por el cliente mediante un protocolo web (generalmente HTTP) y además es capaz de comprender URLs. Mediante URLs concretas (), el cliente es capaz de solicitar los datos adecuados y, el servidor, siendo capaz de comprender estas rutas, devuelve al cliente aquello que este solicitó.

De igual manera que con el tipo de arquitectura, existen varios tipos de servidores web, sin embargo, todos ellos se pueden catalogar en dos categorías principales:

- **Servidor web estático:** También conocidos como "pila", son servidores web sin capacidad de modificación de datos, simplemente devuelven los datos al cliente tal y como estos se encuentran alojados en él, sin alterarlos.
- **Servidor web dinámico**: A diferencia de los servidores web estáticos, estos suelen tener software adicional, comúnmente una base de datos, gracias a esto son capaces de modificar sus datos almacenados mediante consultas a la base de datos. El servidor utilizado en el proyecto sería de este tipo al utilizar una base de datos PostgreSQL.

(MDN, 2022)

Al igual que en el punto anterior, se presenta la **Figura 10** a modo aclarativo sobre la funcionalidad de un servidor web dinámico (Microsoft, 2022):

Figura 10. Servidor web dinámico

4.4. SISTEMA GESTOR DE BASES DE DATOS RELACIONAL (SGBDR)

Los SGBDR se tratan de un tipo de Sistema Gestor de Bases de Datos (SGBD), para entender lo que esto significa, se explicará en primer lugar lo que es un SGBD. A grandes rasgos, se tratan de un conjunto de programas y capacidades que permiten a los diferentes nodos que interactúen con la BBDD crear, actualizar, administrar e interactuar con la BBDD (Rouse, 2019).

Un SGBDR se trata de un tipo de SGBD que se caracteriza por su estructura de tabla basada en filas que conecta elementos de datos relacionados y posee funciones que aseguren la seguridad, precisión, integridad y consistencia de los datos (Rouse, 2019). Cabe destacar que la gran mayoría de SGBDR utilizan SQL para acceder a la base de datos.

Otras diferencias que distinguen a un SGBDR de un SGBD son las siguientes (Brush, 2019):

- **Número de usuarios permitidos:** Un SGBD permite un único usuario a la vez, mientras que un SGBDR puede operar con varios, por esto son tan útiles a la hora de diseñar sistemas distribuidos que hagan varios accesos a la BBDD.
- **Escalabilidad**: Un SGBDR puede soportar una cantidad indefinida de datos con su consecuente adición de software y hardware, mientras que un SGBD solo puede soportar pequeñas cantidades de datos.
- **Normalización:** Los SGBDR permiten la normalización de sus tablas, aumentando la cantidad de datos, pero eliminando la redundancia.

En la_ se pueden observar gráficamente las diferencias estructurales entre un SGBDR y un SGBD (STechies, 2022):

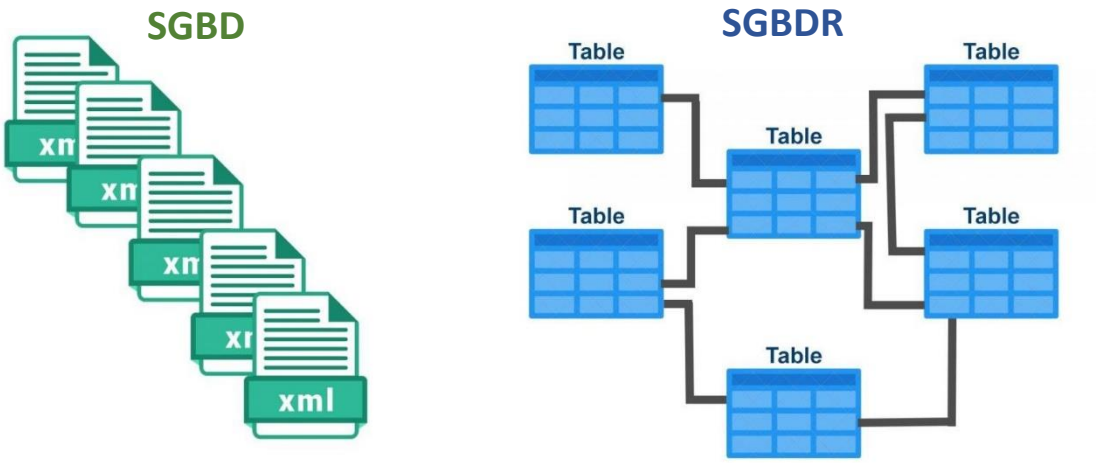


Figura 11. Diferencias estructurales entre un SGBD y un SGBDR

4.5. STREAMING

La definición más simple de *streaming* lo define como la transmisión de un flujo de datos continuo, estable, ordenado y a alta velocidad.

Si bien el streaming de datos en sí es un concepto muy sencillo, cabe destacar que, a bajo nivel, hay muchas opciones que pueden alterar el comportamiento de este sistema de streaming, modificando sus propiedades.

En la **Figura 12** tenemos el modelo de streaming estándar (Jenkov, 2019):

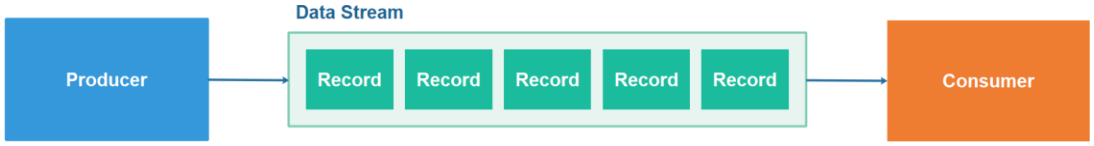


Figura 12. Streaming de datos

Conociendo esto, se procede a explicar muy brevemente los inicios del concepto de streaming:

- La primera aparición del concepto de "streaming" similar a como lo conocemos hoy en día surgió en el año 1920, cuando se patentó un sistema de transmisión y distribución de señales a través de la línea eléctrica, esto se acabaría convirtiendo en el sistema Muzak, el cual proporcionaba de manera continua a sus usuarios sin necesidad de usar radio, que en aquel momento se encontraba en sus primeras etapas. (Wikipedia, 2022).
- No sería ya hasta los 90 que el stream no sufriría su verdadera creación, esta revolución comenzaría con la aparición de las primeras redes wifi y con el aumento de ancho de banda y de equipos a lo largo de los años 2000, también ayudaría el uso incrementado de protocolos web como HTTP o TCP/IP. La primera retransmisión en vivo por internet sería realizada en 1993 por la banda "Severe Tire Damage" (Ruether, 2022) gracias a la tecnología Mbone, que consistía, a grandes rasgos, en una red virtual creada sobre la web para transportar tráfico IP Multicast, de ahí su nombre "Multicast backbone". (Wikipedia, 2022).

Es a partir de los años 2000 cuando este concepto comienza a tomar la forma que tiene hoy en día, esto sucedió gracias a 3 elementos, el primero de ellos fue el más que popular Flash Player (Adobe, 1996) y los protocolos RTMP y RTSP; aún muy populares a día de hoy. Unos años más tarde se anunciaría que los iPhone dejarían de soportar Flash Player y usarían el protocolo HTTP Live Streaming (HLS).

Con la revolución que supuso la llegada al mercado de Youtube, es lógico pensar que la tecnología seleccionada por la plataforma sería la más popular en aquel entonces, sin embargo, Youtube decidió elegir MPEG-DASH, un protocolo recién creado como alternativa a todos los demás protocolos, que estaban asociados a proveedores.

Con el paso de los años, el principal problema a solucionar sería el problema de la latencia, sin embargo, con la mejora de los protocolos y la llegada de la pandemia en 2020, las tecnologías de streaming evolucionaron increíblemente rápido en apenas unos años. A día de hoy, con la muerte de Flash Player en 2020, las plataformas eligen protocolos como Web Real-Time Communications (WebRTC) o HLS de baja latencia (Ruether, 2022).

Conociendo los puntos clave del streaming ahora, indaguemos en detalle sobre estos protocolos mencionados y su funcionalidad

4.5.1. RTMP Y RTSP

En primer lugar, RTMP (Real Time Messaging Protocol), se trata de uno de los protocolos de streaming más longevos que hay, nació inicialmente para establecer una conexión estable entre un servidor de medios y un Flash Player mediante TCP. Actualmente se ha liberado de alguna de esas condiciones y se utiliza para transmitir audio y vídeo entre un codificador y una plataforma de streaming.

En la figura **Figura 13** se puede observar cómo funciona este proceso de envío para un audio (AAC) y vídeo (H.264) comprimidos (Ortiz, 2017):

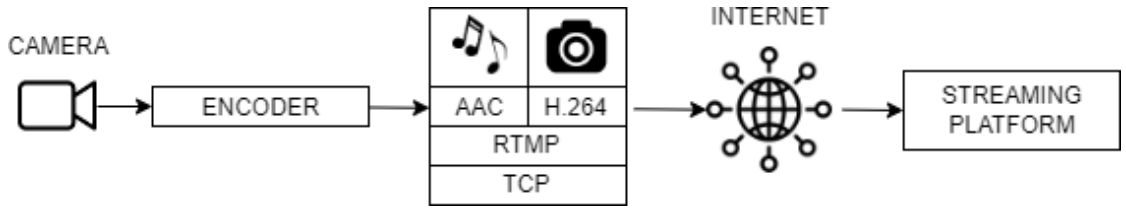


Figura 13. Proceso de retransmisión de audio/vídeo mediante RTMP

RTMP es utilizado principalmente a la hora de recibir retransmisión por parte de los usuarios, sin embargo, no es recomendable usarlo para retransmitir a una audiencia, puesto que no es dinámico, para ese caso se recomendaría usar un protocolo dinámico como HLS que ajusta la calidad de la retransmisión en función de la calidad de la red del cliente.

Por tanto, un buen uso de este protocolo sería utilizarlo para recibir retransmisión, sin embargo, para distribuirla, se usaría HLS, a este proceso de conversión de un protocolo a otro se le conoce como transcoding (Ortiz, 2017).

RTSP (Real Time Streaming Protocol), por otro lado, permite tanto el uso de TCP como de UDP, su uso principal es el de transferir peticiones como “play” o “pause” entre el cliente y la fuente emisora. A día de hoy apenas se utiliza para realizar streaming web, su uso ha quedado relegado a las cámaras IP, como, por ejemplo, las de vigilancia.

4.5.2. HLS

Como ya se ha dicho anteriormente, el protocolo HLS (HTTP Live Streaming) se trata de un protocolo de streaming dinámico, lo cual significa que es óptimo a la hora de necesitar que la retransmisión recibida por el cliente se adapte a las necesidades cambiantes de su red wifi.

Otra de sus enormes ventajas es su alta compatibilidad con casi todos los dispositivos existentes, esto sucede gracias a ser un protocolo de HTTP, el cual se encuentra en todos los dispositivos compatibles con internet.

Su funcionamiento puede proceder de cualquier servidor, ya que simplemente requiere el uso de HTTP, sus fases son las siguientes (Cloudfare, 2022):

1. **Codificación:** Los datos son formateados de manera estandarizada para que cualquier dispositivo pueda reconocerlos, actualmente HLS utiliza codificación H.264 o H.265.
2. **Segmentación:** El vídeo a retransmitir es dividido en varios segmentos (unos 6s por segmento) y crea un índice que le permita conocer el orden secuencial de estos segmentos.
3. **Duplicación:** HLS genera copias en diferentes calidades (480p, 1080p, etc.) de estos segmentos de vídeo, esto es lo que le permite ser dinámico y cambiar la calidad de la retransmisión en cualquier momento

Una vez recibido el archivo por parte del cliente, este se sirve del índice creado para ensamblar el vídeo en el orden correcto.

4.5.3. MPEG-DASH

Habiendo ya explicado el funcionamiento de HLS, entender MPEG-DASH no será muy diferente, ya que posee muchas características en común, en primer lugar, MPEG-DASH también segmenta la retransmisión a enviar para permitir dinamismo en esta, de hecho, las siglas DASH provienen del inglés “Dynamic Adaptative Streaming over HTTP”, observando estas siglas podemos ver otra característica en común, y es que MPEG-DASH también funciona sobre HTTP.

Por ahora, no poseen ninguna diferencia, así que cabría preguntarse qué diferencias existen entre ambos, sin embargo, la diferencia principal entre ellos es la compañía que los creó, MPEG-DASH nació como una alternativa a los sistemas de streaming que existían en torno al años 2010, como, por ejemplo, HLS que fue creado por Apple, es por esto que MPEG-DASH no se distingue tanto de HLS, su principal atractivo era no estar asociado con ningún proveedor (Ruether, 2022).

Sin embargo, a día de hoy existen más diferencias entre ambos, estas se exponen a continuación:

- MPEG-DASH admite cualquier estándar de codificación, sin embargo, HLS solo admite H.264 y H.265.
- Los dispositivos Apple admiten únicamente HLS como protocolo de streaming.
- MPEG-DASH se incorporó con el tiempo como un estándar internacional, sin embargo, HLS no.

5. METODOLOGÍA, TÉCNICAS Y HERRAMIENTAS

En este apartado se detallarán la metodología, técnicas y herramientas utilizadas en el proyecto. En primer lugar, se hablará sobre las metodologías que han regido la estructura del proyecto, lo siguiente serán los patrones y estilos arquitectónicos seguidos, principalmente MVVM y la elicitación de requisitos según el método de Durán y Bernárdez. Por último, se presentarás las herramientas auxiliares, bases de datos y lenguajes de programación utilizados.

5.1. METODOLOGÍA

La metodología principal seguida en el proyecto se trata del Proceso Unificado, que nos servirá para identificar, catalogar y planificar los diferentes casos de uso del sistema de una manera incremental e iterativa.

Para aplicar esta metodología en el proyecto se ha decidido seguir el modelo iterativo que ofrece el Proceso Unificado, dividiendo cada una de las iteraciones en 6 etapas (Modelo de negocio, Requisitos, Análisis, Diseño, Implementación y Pruebas. Se decidió seguir este modelo por varias razones, siendo la principal la magnitud del proyecto y su gran modularidad, seguir un modelo iterativo sería la solución ideal para ir abarcando cada uno de los módulos del proyecto incrementalmente, centrándose en diferentes módulos en función de la iteración en que el desarrollo se encontrase.

Basándose en los aspectos recién mencionados, se presenta una planificación iterativa del proyecto a realizar:

- **Iteración 1**: Se trataría de la iteración inicial, en esta se realizaría la mayor labor de investigación de herramientas y técnicas compatibles con el proyecto, así como una primera elicitación de requisitos, análisis y diseño del sistema, principalmente de aquellos módulos más complejos y que más iteraciones requerirían como puede ser, por ejemplo, el módulo de gestión de retransmisión.
- **Iteración 2**: Se comienza a investigar en profundidad sobre herramientas y plataformas de streaming y captura de inputs, ya que serán los dos puntos más complejos de implementar en el sistema. Se realiza también un primer refinamiento del modelo de dominio y el modelo de análisis, aplicando aquellas funcionalidades descubiertas en la etapa de investigación. Es aquí donde la fase de implementación y pruebas toman un mayor rol pues ya existen artefactos integrados y desplegados en una plataforma inicial.
- **Iteración 3**: Se comienza a investigar sobre aquellas funcionalidades que tienen un papel principal en el sistema, pero no son tan complejas como la retransmisión, se comienza a desplegar el servidor web en el que alojar los datos de usuario y se diseña la funcionalidad del catálogo. El modelo de dominio y de análisis es, de nuevo, refinado y se integran, despliegan y prueban los cambios pertinentes a esta iteración.
- **Iteración 4**: Etapa final del sistema, se realiza una investigación sobre posibles riesgos o fallas en el sistema, así como una búsqueda final de herramientas auxiliares quemejoren la funcionalidad del proyecto. Se refinan por completo los modelos de dominioy análisis y se realiza una última pero exhaustiva etapa de integración, despliegue y pruebas.

Por último, se añaden los hitos pertinentes al final de cada iteración y la planificación quedaría completamente realizada.

El resultado de esto queda ilustrado en las figuras **Figura 14** - **Figura 20** que muestran el diagrama de Gantt del proyecto, realizado mediante la herramienta *Microsoft Project*.

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Proyecto	**130 días**	**mar 01/03/22**	**lun 29/08/22**		
Iteración 1	**45 días**	**mar 01/03/22**	**lun 02/05/22**		
Modelo del negocio	**13 días**	**mar 01/03/22**	**jue 17/03/22**		
Investigar proyectos similares existentes	6 días	mar 01/03/22	mar 08/03/22		Jesús Manuel García Prieto
Investigar sobre diferente software para el desarrollo del sistema	5 días	mié 09/03/22	mar 15/03/22	4	Jesús Manuel García Prieto
Investigar diferentes estilos artísticos y visuales para el desarrollo del sistema	2 días	mié 16/03/22	jue 17/03/22	5	Jesús Manuel García Prieto
Requisitos	**6 días**	**vie 18/03/22**	**vie 25/03/22**		
Definición del modelo inicial de requisitos	6 días	vie 18/03/22	vie 25/03/22	6	Jesús Manuel García Prieto
Análisis	**8 días**	**lun 28/03/22**	**mié 06/04/22**		
Definición inicial de CU Subsistema "Gestión de Usuarios"	2 días	lun 28/03/22	mar 29/03/22	8	Jesús Manuel García Prieto
Definición inicial de CU Subsistema "Gestión de Catálogo"	2 días	mié 30/03/22	jue 31/03/22	10	Jesús Manuel García Prieto
Definición inicial de CU Subsistema "Gestión de Actividades Sociales"	2 días	vie 01/04/22	lun 04/04/22	11	Jesús Manuel García Prieto
Definición inicial de CU Subsistema "Gestión de Retransmisión"	2 días	mar 05/04/22	mié 06/04/22	12	Jesús Manuel García Prieto

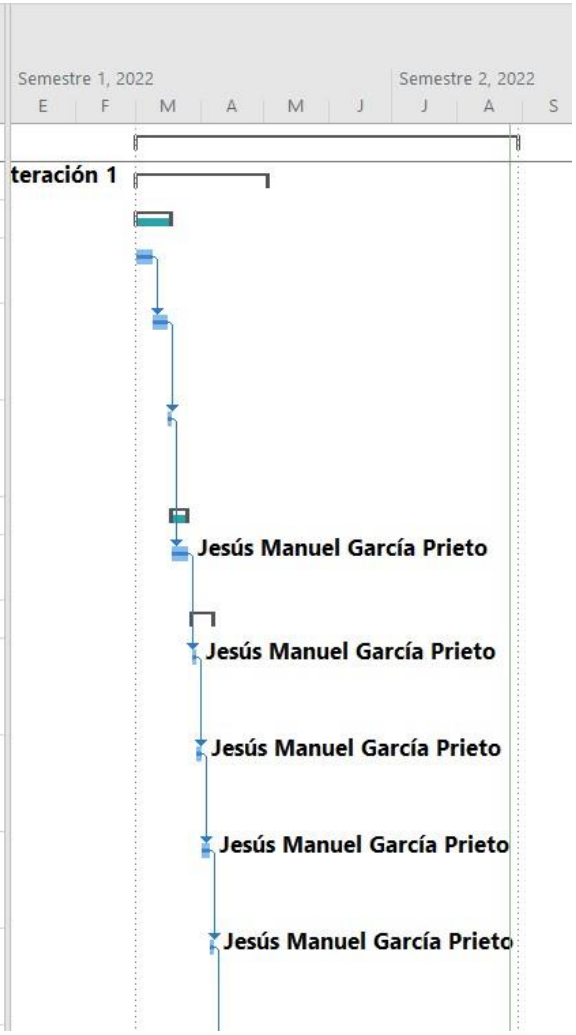


Figura 14. Diagrama de Gantt y Planificación temporal (I)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Diseño	**8 días**	**jue 07/04/22**	**lun 18/04/22**		
Diseño tentativo del Subsistema "Gestión de Usuarios"	2 días	jue 07/04/22	vie 08/04/22	13	Jesús Manuel García Prieto
Diseño tentativo del Subsistema "Gestión de Catálogo"	1 día	lun 11/04/22	lun 11/04/22	15	Jesús Manuel García Prieto
Diseño tentativo del Subsistema "Gestión de Actividades Sociales"	2 días	mar 12/04/22	mié 13/04/22	16	Jesús Manuel García Prieto
Diseño tentativo del Subsistema "Gestión de Retransmisión"	2 días	jue 14/04/22	vie 15/04/22	17	Jesús Manuel García Prieto
Diseño de la arquitectura inicial	1 día	lun 18/04/22	lun 18/04/22	18	Jesús Manuel García Prieto
Implementación	**9 días**	**mar 19/04/22**	**vie 29/04/22**		
Preparación del entorno de desarrollo	1 día	mar 19/04/22	mar 19/04/22	19	Jesús Manuel García Prieto
Despliegue del sistema de envío y recepción de vídeo	4 días	mié 20/04/22	lun 25/04/22	21	Jesús Manuel García Prieto
Implementación de interfaz inicial para el envío y recepción de vídeo	4 días	mar 26/04/22	vie 29/04/22	22	Jesús Manuel García Prieto
Pruebas	**1 día**	**lun 02/05/22**	**lun 02/05/22**		
Pruebas iniciales sobre el sistema de envío y recepción de vídeo	1 día	lun 02/05/22	lun 02/05/22	23	Jesús Manuel García Prieto
Hito 1	0 días	lun 02/05/22	lun 02/05/22	25	Jesús Manuel García Prieto

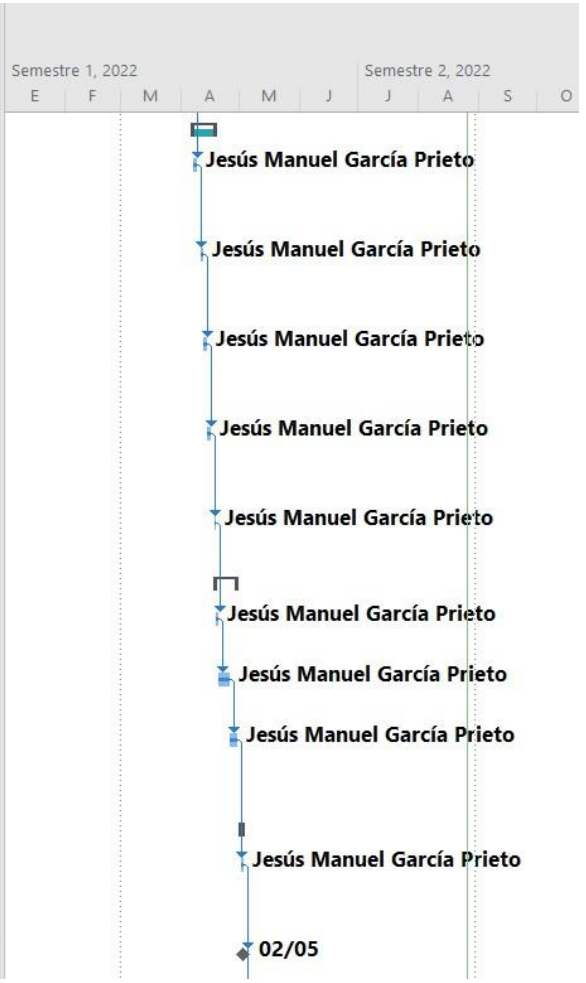


Figura 15. Diagrama de Gantt y Planificación temporal (II)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Iteración 2	**20 días**	**mar 03/05/22**	**lun 30/05/22**		
Modelo del negocio	**5 días**	**mar 03/05/22**	**lun 09/05/22**		
Investigar sobre técnicas de	3 días	mar 03/05/22	jue 05/05/22	25	Jesús Manuel García Prieto
Investigar sobre técnicas de captura y aplicación de	2 días	vie 06/05/22	lun 09/05/22	29	Jesús Manuel García Prieto
Requisitos	**1 día**	**mar 10/05/22**	**mar 10/05/22**		
Refinamiento de requisitos del Subsistema "Gestión de Actividades Sociales"	0,5 días	mar 10/05/22	mar 10/05/22	30	Jesús Manuel García Prieto
Refinamiento de requisitos del Subsistema "Gestión de Retransmisión"	0,5 días	mar 10/05/22	mar 10/05/22	32	Jesús Manuel García Prieto
Análisis	**2 días**	**mié 11/05/22**	**jue 12/05/22**		
Realización de matrices de rastreabilidad	0,5 días	mié 11/05/22	mié 11/05/22	33	Jesús Manuel García Prieto
Refinamiento de análisis del Subsistema "Gestión de Actividades Sociales"	0,5 días	mié 11/05/22	mié 11/05/22	35	Jesús Manuel García Prieto
Refinamiento de análisis del Subsistema "Gestión de Retransmisión"	1 día	jue 12/05/22	jue 12/05/22	36	Jesús Manuel García Prieto
Diseño	**2 días**	**vie 13/05/22**	**lun 16/05/22**		
Refinamiento de diseño del Subsistema "Gestión de Actividades Sociales"	1 día	vie 13/05/22	vie 13/05/22	37	Jesús Manuel García Prieto
Refinamiento de diseño del Subsistema "Gestión de Retransmisión"	1 día	lun 16/05/22	lun 16/05/22	39	Jesús Manuel García Prieto

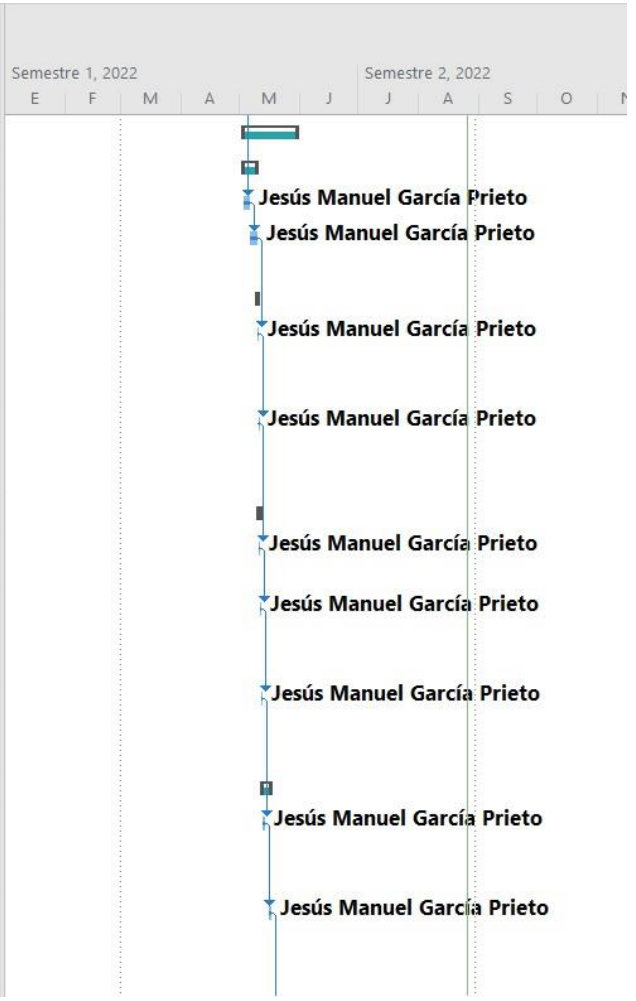


Figura 16. Diagrama de Gantt y Planificación temporal (III)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Implementación	**8 días**	**mar 17/05/22**	**jue 26/05/22**		
Implementación del Subsistema "Gestión de Actividades Sociales"	3 días	mar 17/05/22	jue 19/05/22	40	Jesús Manuel García Prieto
Implementación del Subsistema "Gestión de Retransmisión"	4 días	vie 20/05/22	mié 25/05/22	42	Jesús Manuel García Prieto
Integración y Despliegue de los subsistemas implementados	1 día	jue 26/05/22	jue 26/05/22	43	Jesús Manuel García Prieto
Pruebas	**2 días**	**vie 27/05/22**	**lun 30/05/22**		
Pruebas del Subsistema "Gestión de Actividades Sociales"	0,5 días	vie 27/05/22	vie 27/05/22	44	Jesús Manuel García Prieto
Pruebas del Subsistema "Gestión de Retransmisión"	0,5 días	vie 27/05/22	vie 27/05/22	46	Jesús Manuel García Prieto
Pruebas de Integración	1 día	lun 30/05/22	lun 30/05/22	47	Jesús Manuel García Prieto
Hito 2	0 días	lun 30/05/22	lun 30/05/22	48	Jesús Manuel García Prieto
Iteración 3	**24 días**	**mar 31/05/22**	**vie 01/07/22**		
Modelo del negocio	**7 días**	**mar 31/05/22**	**mié 08/06/22**		
Investigación sobre integración de servidores web	5 días	mar 31/05/22	lun 06/06/22	46	Jesús Manuel García Prieto
Investigación sobre la gestión e implementación de catálogos en plataformas de	2 días	mar 07/06/22	mié 08/06/22	52	Jesús Manuel García Prieto

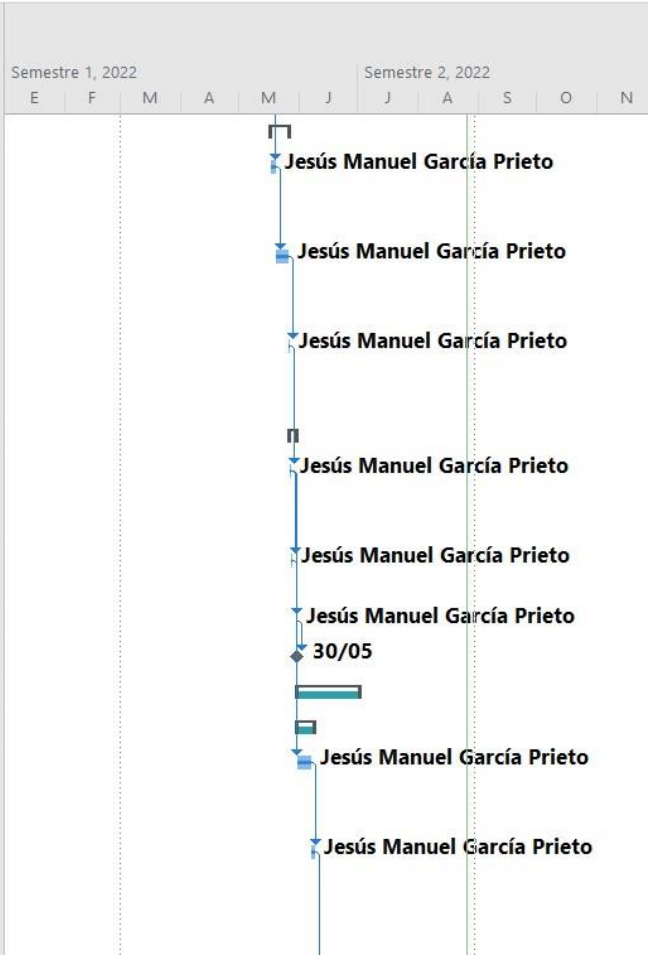


Figura 17. Diagrama de Gantt y Planificación temporal (IV)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Requisitos	**0,5 días**	**jue 09/06/22**	**jue 09/06/22**		
Refinamiento de requisitos del Subsistema "Gestión de Usuarios"	0,25 días	jue 09/06/22	jue 09/06/22	53	Jesús Manuel García Prieto
Refinamiento de requisitos del Subsistema "Gestión de Catálogo"	0,25 días	jue 09/06/22	jue 09/06/22	55	Jesús Manuel García Prieto
Análisis	**1 día**	**jue 09/06/22**	**vie 10/06/22**		
Refinamiento de análisis del Subsistema "Gestión de Usuarios"	0,5 días	jue 09/06/22	jue 09/06/22	56	Jesús Manuel García Prieto
Refinamiento de análisis del Subsistema "Gestión de Catálogo"	0,5 días	vie 10/06/22	vie 10/06/22	58	Jesús Manuel García Prieto
Diseño	**2 días**	**vie 10/06/22**	**mar 14/06/22**		
Refinamiento de diseño del Subsistema "Gestión de Usuarios"	1 día	vie 10/06/22	lun 13/06/22	59	Jesús Manuel García Prieto
Refinamiento de diseño del Subsistema "Gestión de Catálogo"	1 día	lun 13/06/22	mar 14/06/22	61	Jesús Manuel García Prieto
Implementación	**11,5 días**	**mar 14/06/22**	**mié 29/06/22**		
Implementación del Subsistema "Gestión de Usuarios"	6 días	mar 14/06/22	mié 22/06/22	62	Jesús Manuel García Prieto
Implementación del Subsistema "Gestión de Catálogo"	3 días	mié 22/06/22	lun 27/06/22	64	Jesús Manuel García Prieto

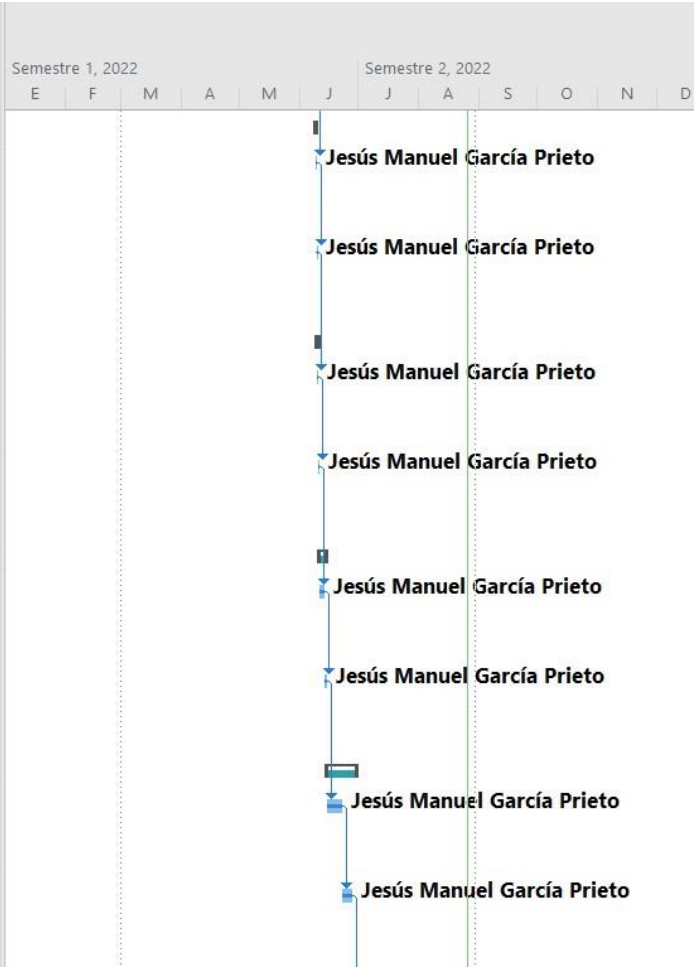


Figura 18. Diagrama de Gantt y Planificación temporal (V)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Integración y Despliegue de los subsistemas implementados	2,5 días	lun 27/06/22	mié 29/06/22	65	Jesús Manuel García Prieto
Pruebas	**2 días**	**jue 30/06/22**	**vie 01/07/22**		
Pruebas del Subsistema "Gestión de Usuarios"	0,5 días	jue 30/06/22	jue 30/06/22	66	Jesús Manuel García Prieto;
Pruebas del Subsistema "Gestión de Catálogo"	0,5 días	jue 30/06/22	jue 30/06/22	68	Jesús Manuel García Prieto
Pruebas de Integración	1 día	vie 01/07/22	vie 01/07/22	69	Jesús Manuel García Prieto
Hito 3	0 días	vie 01/07/22	vie 01/07/22	70	Jesús Manuel García Prieto
Iteración 4	**41 días**	**lun 04/07/22**	**lun 29/08/22**		
Modelo del negocio	**7 días**	**lun 04/07/22**	**mar 12/07/22**		
Revisión de posibles riesgos existentes en el	3 días	lun 04/07/22	mié 06/07/22	70	Jesús Manuel García Prieto
Revisión final de sistemas similares	4 días	jue 07/07/22	mar 12/07/22	74	Jesús Manuel García Prieto
Requisitos	**3 días**	**mié 13/07/22**	**vie 15/07/22**		
Refinamiento final de requisitos en función de los riesgos existentes	3 días	mié 13/07/22	vie 15/07/22	75	Jesús Manuel García Prieto
Análisis	**4 días**	**lun 18/07/22**	**jue 21/07/22**		
Revisión final del modelo de análisis	4 días	lun 18/07/22	jue 21/07/22	77	Jesús Manuel García Prieto
Diseño	**7 días**	**vie 22/07/22**	**lun 01/08/22**		
Revisión final del modelo de diseño	7 días	vie 22/07/22	lun 01/08/22	79	Jesús Manuel García Prieto

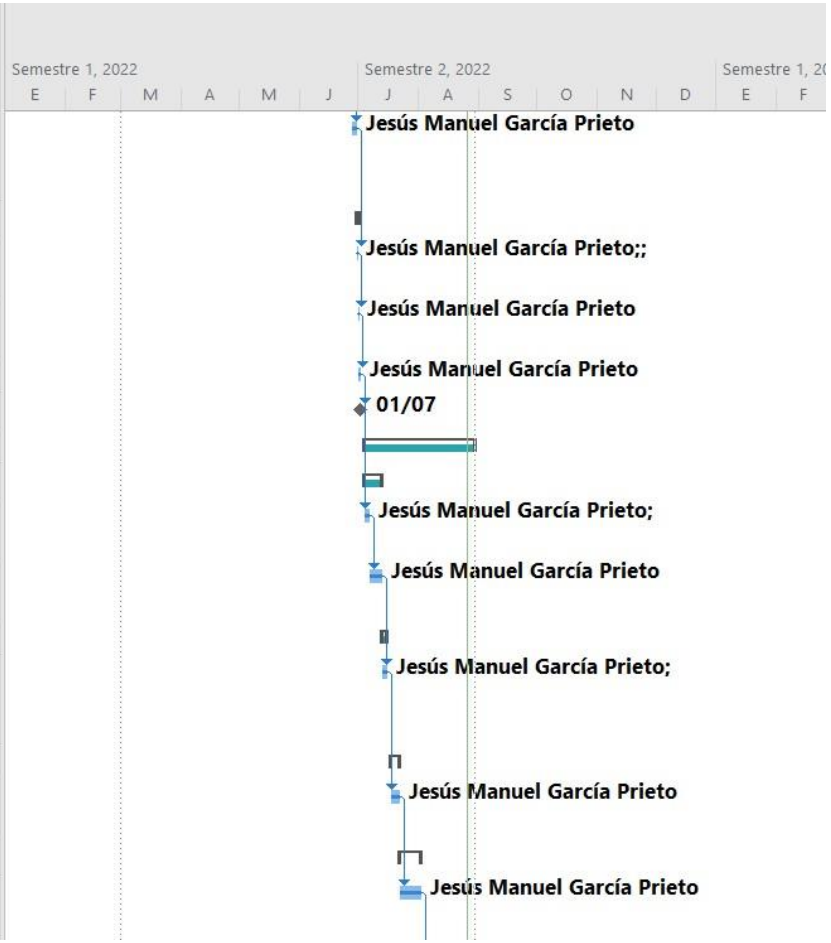


Figura 19. Diagrama de Gantt y Planificación temporal (VI)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Revisión final del modelo de diseño	7 días	vie 22/07/22	lun 01/08/22	79	Jesús Manuel García Prieto
Implementación	**16 días**	**mar 02/08/22**	**mar 23/08/22**		
Integración y despliegue de los cambios realizados al sistema final	16 días	mar 02/08/22	mar 23/08/22	81	Jesús Manuel García Prieto
Pruebas	**4 días**	**mié 24/08/22**	**lun 29/08/22**		
Pruebas de los cambios realizados al sistema final	4 días	mié 24/08/22	lun 29/08/22	83	Jesús Manuel García Prieto
Hito 4	0 días	lun 29/08/22	lun 29/08/22	85	Jesús Manuel García Prieto

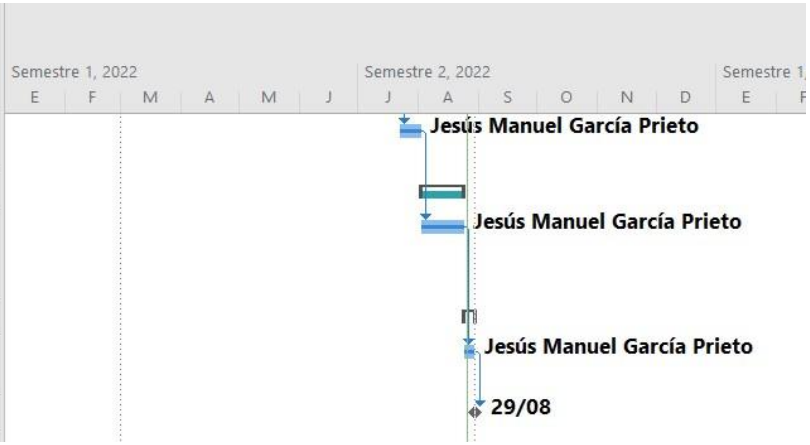


Figura 20. Diagrama de Gantt y Planificación temporal (VII)

5.2. TÉCNICAS

5.2.1. MVVM

Debido a la utilización de Vue3 en el sistema, este se encuentra comprometido con la utilización del patrón MVVM, ya que Vue se especializa en implementar este patrón con facilidad.

Antes de explicar cómo Vue logra implementarlo, se procede a explicar en qué consiste este patrón tan extendido:

Para entender cómo funciona el patrón MVVM vamos a compararlo con su contraparte más popular: el patrón MVC, ambos patrones buscan el lograr dividir la aplicación o el sistema en diferentes capas, para facilitar así su mantenimiento, desarrollo o implementación.

La principal diferencia entre ellos es que el MVVM elimina toda lógica de la Vista, y la Vista-Modelo separa totalmente la Vista del Modelo, a diferencia del patrón MVC en el cual las 3 capas se encuentran interconectadas, como se puede observar en la **Figura 21:** (Zuev, 2020)

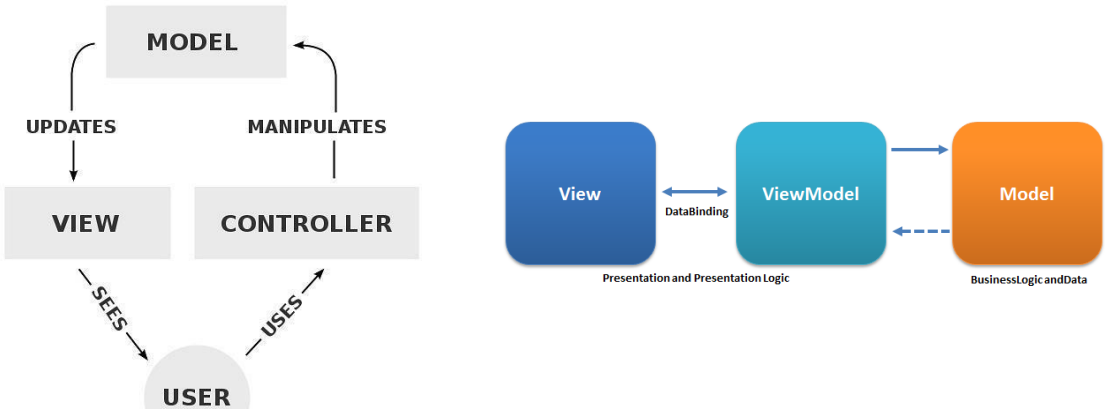


Figura 21. Estructura de los modelos MVC y MVVM

5.2.2. MÉTODO DE DURÁN Y BERNÁRDEZ

Se trata de una metodología desarrollada en el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla, su uso principal es el de recoger y documentar los requisitos de un sistema. (García Peñalvo & García Holgado, 2018)

En la **Figura 22** se pueden observar las diferentes tareas en las que esta metodología recomienda obtener y documentar requisitos (Durán y Bernárdez, 2002):

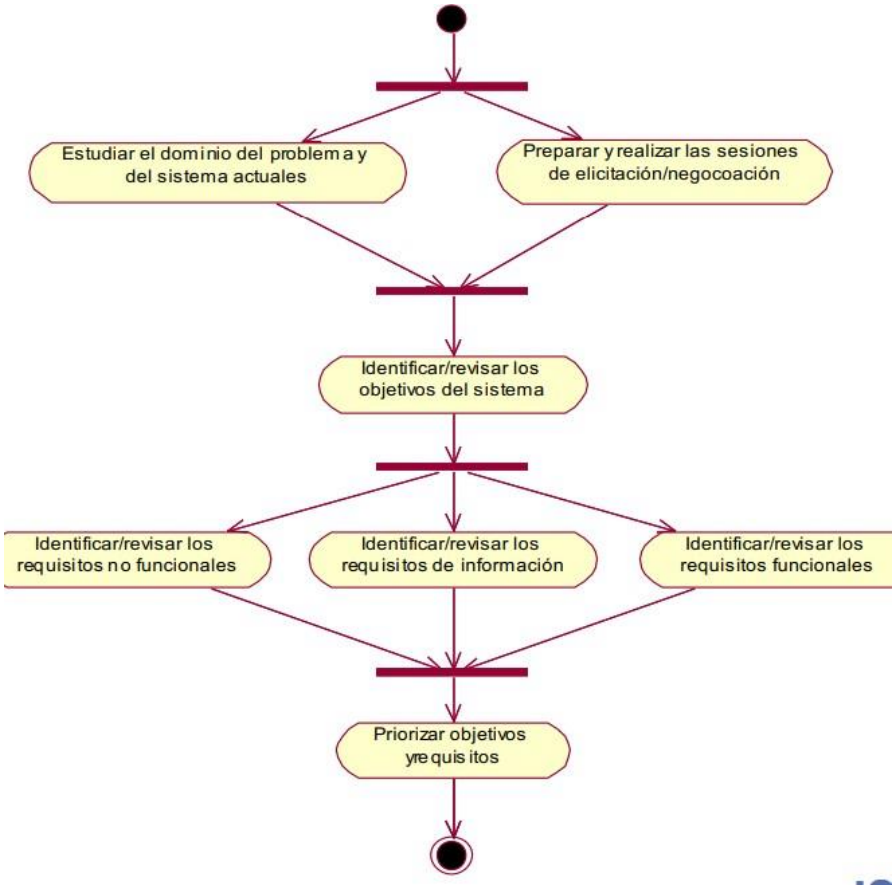


Figura 22. Tareas para la obtención y definición de requisitos

5.3. HERRAMIENTAS

En este apartado se expondrán aquellas herramientas utilizadas durante todo el desarrollo del proyecto, tanto durante su creación como durante su documentación, para ello, se dividirá este apartado en varias subsecciones, explicando en primer lugar las herramientas principales utilizadas en el proyecto, que serán sobre las que más información se exponga, en segundo lugar, la base de datos y lenguajes de programación utilizados, será aquí también donde se expongan aquellas bibliotecas que amplíen o añadan funcionalidad a estos lenguajes y, por último, herramientas auxiliares, que serán, en su mayoría, aquellas utilizadas para realizar la documentación y sus diferentes anexos.

5.3.1. HERRAMIENTAS PRINCIPALES

Si bien estas herramientas se podrían englobar en otros apartados de esta sección como, por ejemplo, "lenguajes de programación", se ha decidido incluirlas en esta sección debido a su gran peso e importancia en el proyecto, a diferencia de otras herramientas, se realizará una explicación exhaustiva sobre estas debido a su magnitud en el sistema.

5.3.1.1. VUE.JS

Vue.js se trata de un framework open-source de JavaScript, orientado al frontend, en concreto, nos permite crear interfaces de usuarios y aplicaciones de una sola página (SPA).

Su gran punto de interés es, además de su sencillez, el haber sabido implementar aquellos elementos imprescindibles presentes en otros frameworks similares, pero eliminando a su vez aquellos que no aporten valor, un gran ejemplo es la eliminación de JQuery, reduciendo considerablemente así la cantidad de código en las aplicaciones que usen Vue.

Algunos otros puntos que caracterizan a Vue son (Barragán, 2021):

- **Reactividad:** Una aplicación realizada con Vue puede reaccionar a cambios realizados en esta de manera dinámica gracias al DOM de Vue, sin necesidad de refrescar la página, esto es logrado gracias a que los modelos del framework son objetos de JavaScript y mejora increíblemente la experiencia de usuario.
- **Componentes:** Se trata de una de las características fundamentales de Vue. Estos componentes son cápsulas rellenas de código reutilizable, por lo general incluyen código en HTML, JavaScript y CSS. Gracias a esto la escalabilidad de los módulos y de la propia aplicación en su conjunto mejora con creces, así como su facilidad a la hora de reemplazar un módulo por otro.
- **Modularidad:** La funcionalidad que Vue ofrece se encuentra dividia entre varias bibliotecas, a diferencia de sus competidores, que ofrecen una gran cantidad de funciones, Vue nos permite seleccionar qué bibliotecas se adaptan mejor a nuestro proyecto, pudiendo añadir más según esto sea necesario. Así Vue logra reducir el tamaño de los proyectos y añadir una capa de sencillez y control al programador.
- **DOM Virtual:** En lugar de actualizar los valores en la vista, Vue crea una vista virtual en paralelo, de esta manera aplica estos cambios de una manera más óptima.
- **Patrón MVVM:** Ya se ha explicado anteriormente el funcionamiento estructural de este patrón, sin embargo, Vue tiene su propia manera de integrarlo en sus proyectos.
  Con todos los elementos explicados a lo largo de este apartado, se procede a presentar cómo divide Vue un proyecto sobre el patrón MVVM (akin-ogundenji, 2015):
    - **El Modelo** en Vue se trata simplemente de los datos en crudo que existan en el sistema o los que se obtengan del servidor. El Modelo no posee ningún tipo de lógica ni comportamiento definidos, más allá de la validación de datos. Además

de esto, no posee ningún tipo de forma de acceso al backend o a alguna API para obtener estos datos, simplemente funciona como un contenedor de datosque serán utilizados por el VM.

- **La Vista** se encarga de renderizar el contenido del Modelo de cara al usuario, sin embargo, la Vista no sabe nada acerca del Modelo y viceversa, la comunicación entre ambas se produce a través de la VM, de la cual, si tiene constancia la Vista, convirtiéndola así en una "Vista activa". Su único fin más alláde mostrar datos al usuario es el de interceptar inputs del usuario y enviarlos alVM.
- **La Vista Modelo** se trata del enlace entre la Vista y el Modelo, toda la lógica requerida para manipular la información contenida en el Modelo se encuentra en la VM y toda la lógica usada por la Vista para tratar o formatear los datos de usuario viene definida por la VM. A diferencia de otros patrones, toda la lógica de negocio del sistema se encuentra acoplada en la VM, en lugar de encontrarse en el Modelo.

En la **Figura 23** (ProgrammerClick, 2020-2022) podemos observar la implementación de este patrón de diseño en Vue.

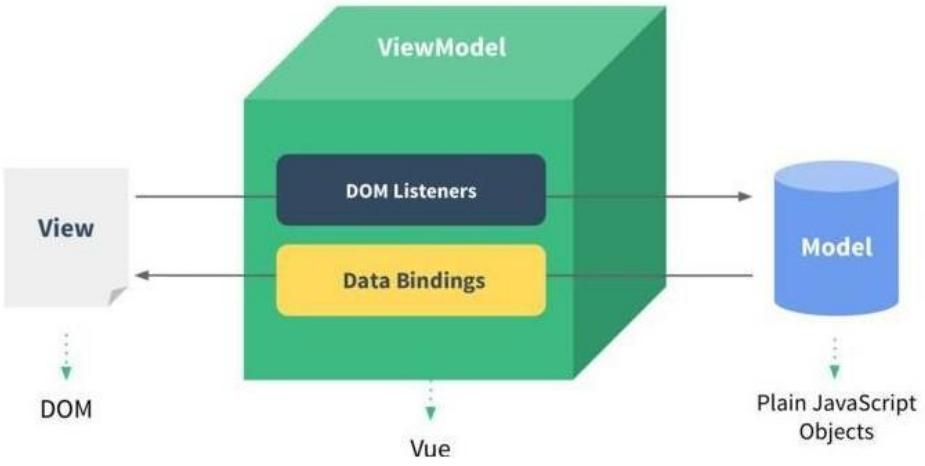


Figura 23. Patrón MVVM en Vue

Como conclusión, se puede afirmar sin duda que Vue.js se trata del framework orientado a frontend más ligero con respecto a su competencia, esto, sumado a su baja curva de aprendizaje, lo convierte en una opción muy atractiva para todo tipo de programadores, bien sea experimentados o no.

5.3.1.2. ELECTRON.JS

Electron se trata de un framework open-source de JavaScript creado sobre Node.js, fue creado por los creadores de GitHub con el objetivo de permitir adaptar aplicaciones creadas sobre HTML y CSS en aplicaciones de escritorio multiplataforma (Garain, 2022).

Dicho de otra forma, Electron nos permite crear aplicaciones de escritorio a partir de aplicaciones que cumplan con los siguientes tres requisitos:

- HTML y CSS usados para la interfaz de usuario.
- JavaScript como lenguaje principal.
- Node.js y npm

Esto permite a desarrolladores de Android o, como es en este caso, Vue, poder generar aplicaciones de escritorio con muy poco esfuerzo adicional.

Su funcionamiento es muy sencillo, y con la explicación de esto se entenderá por qué permite usar Vue, React o Angular a pesar de no ser exactamente HTML y CSS puros. La idea fundamental es que Electron funciona creando dos tipos de procesos (Daniel, 2016):

- El primer proceso es conocido como “main”y funciona sobre Node.js, se trata del proceso principal y consiste en nuestra aplicación en sí misma, este proceso posee varias API que permiten a Electron comunicarse con el SO.
- El segundo proceso es conocido como “renderer” y funciona sobre Chromium con Node.js integrado, este proceso tiene acceso a todos los módulos instalados mediante npm, por lo que es gracias a este proceso por lo que se pueden usar bibliotecas como Vue o React para realizar la interfaz de usuario.

En la **Figura 24** se puede observar la estructura de Electron.js y sus dos tipos de procesos (Daniel, 2016):

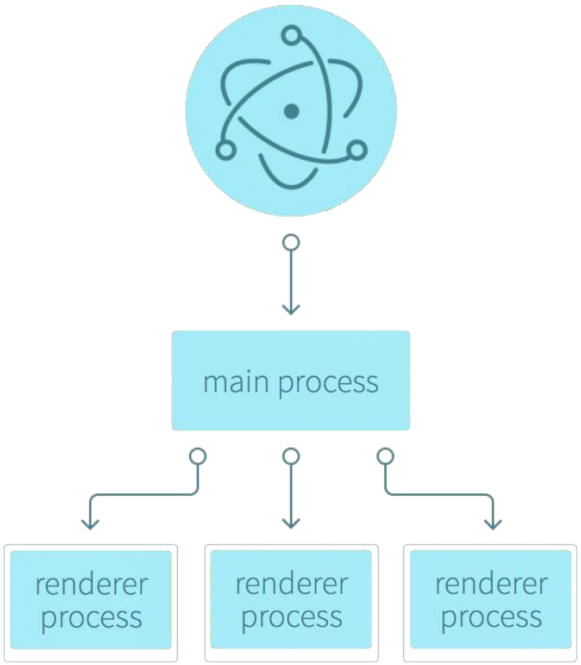


Figura 24. Estructura de Electron.js

5.3.1.3. NODE.JS

Node se trata de un entorno de ejecución open-source para JavaScript, el cual permite a este lenguaje crear sitios web dinámicos muy eficientes, especialmente cuando hay varias conexiones simultáneas (Acibeiro, 2022).

Este lenguaje nació en el año 2009 cuando el lenguaje JavaScript podía ejecutarse únicamente del lado del navegador o del cliente, pero no tenía compatibilidad si se quería ejecutar desde el lado del servidor o fuera del navegador.

Algunas de las ventajas que Node.js presenta son las siguientes:

- **Rendimiento:** El utilizar recursos sin bloqueos y el utilizar menos recursos de lo normal convierte a Node.js en un entorno rápido y eficaz.
- **Escalabilidad:** Al poder procesar tantas conexiones simultáneamente convierte a Node.js en un entorno ideal para aplicaciones que escalan con el tiempo.

Esto lleva a preguntarse qué es npm, npm se trata del gestor de paquetes de Node.js, permitiendo a cualquier proyecto acceder a los miles de proyectos en Node.js que sean compatibles.

En el proyecto a desarrollar se usará en concreto npm para poder tener acceso a todos aquellos proyectos de Node.js que puedan ser de utilidad para el sistema.

5.3.1.4. EXPRESS

Se trata de un framework backend open-source para Node.js, usado principalmente para diseñar aplicaciones web y APIs, este último caso será para el que se use en el proyecto a desarrollar. Su uso más popular es conocido como MEVN (MongoDB, Express, Vue, Node) aunque el tipo de base de datos suele variar como es en nuestro caso.

Algunas de sus funcionalidades más destacadas son (MDN, 2022):

- Establecer ajustes de aplicaciones web como qué puerto usar para conectar.
- Creación de diferentes manejadoras para peticiones HTTP a partir de diferentes rutas (esto será lo que se usará en el proyecto y nos referiremos a ello como endpoint)

5.3.1.5. WEBRTC

WEbRTC (Web Real Time Communicatios) es un proyecto open-source que permite comunicaciones en tiempo real sin plugins adicionales gracias a una API JavaScript (3CX, 2022).

WebRTC admite datos de voz y vídeo enviados entre pares lo cual lo convierte en una solución muy apta para herramientas de video comunicación, llegando a funcionar en todos los navegadores más importantes e incluso en Android y iOS.

Una aplicación de WebRTC generalmente atravesará un flujo de aplicación común. Acceder a los dispositivos multimedia, abrir conexiones de pares, descubrir pares y comenzar a transmitir. En la **Figura 25** se puede observar la estructura de la que este flujo se sirve (Shacklett, 2022):

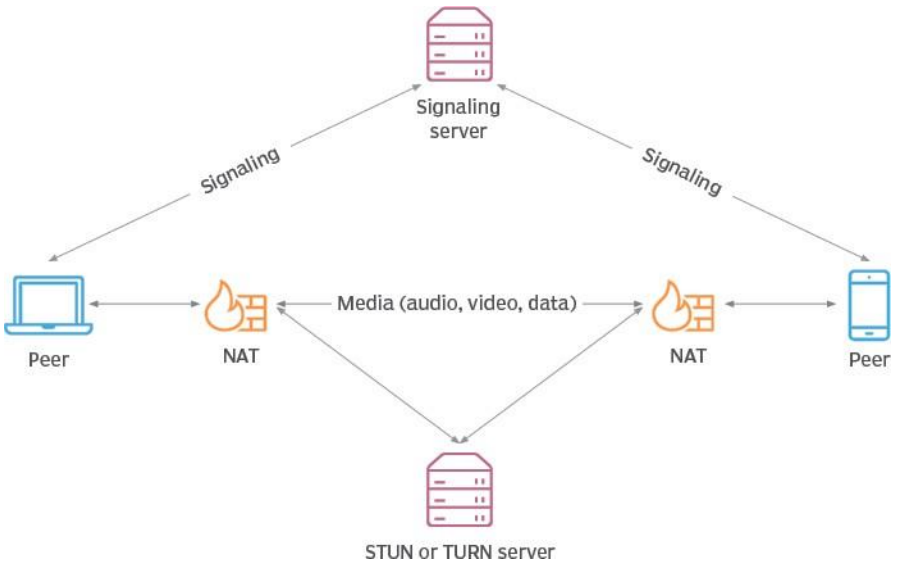


Figura 25. Estructura de una herramienta con WebRTC

Algunos de los proyectos más famosos que se valen de WebRTC son Whatsapp, Discord o Snapchat (WebRTC, 2022)

5.3.2. BASES DE DATOS

A lo largo de esta memoria ya se ha expuesto el concepto de Sistema Generador de Bases de Datos Relacional (SGBDR), esto se debe a que la BBDD utilizada en el proyecto es de este tipo, se trata de PostgreSQL. Se procede a explicar en detalle en qué consiste:

5.3.2.1. POSTGRESQL

PostgreSQL, también conocido como Postgres es un SGBDR, aunque soporta almacenamiento no relacional mediante JSON.

Algunas de las características más relevantes de Postgres son las siguientes (Sharma, 2021):

- Postgres no es solo un SGBD relacional, sino que es objeto-relacional, lo cual quiere decir que es capaz de soportar estructuras de datos complejas y una gran cantidad de tipos de datos, provee capacidad de almacenamiento extensible y es famoso por su integridad de datos.
- Ofrece una gran cantidad de funciones (que siguen en constante crecimiento) que ofrecen facilidades a programadores para crear aplicaciones o a administradores para proteger la seguridad e integridad del sistema.

Esto ofrece a PostgreSQL mucha ventaja frente a otras BBDD con SQL como MySQL, MariaDB o Firebird.

5.3.3. LENGUAJES DE PROGRAMACIÓN

En este apartado se hablará de aquellos lenguajes utilizados en el proyecto, así como de aquellas bibliotecas que amplíen o faciliten su funcionalidad y uso.

5.3.3.1. ROBOT.JS

Se trata de una biblioteca de Node.js orientada a controlar el mouse, teclado y leer la pantalla de un equipo. En el proyecto esta biblioteca ha sido utilizada para forzar los inputs del usuario en la máquina fuente, permitiendo así a este jugar al videojuego ejecutado en el equipo externo

5.3.3.2. HTML

Como ya sabemos, HTML se trata del lenguaje de marcas por excelencia en la web, Vue se basa en él para crear su interfaz, por tanto, HTML, concretamente HTML5, es usado en el proyecto de manera exhaustiva.

5.3.3.3. CSS

Como no puede ser de otra manera, HTML y CSS suelen ir de la mano, en el caso de Vue esto se sigue cumpliendo, por lo que CSS será otro de los pilares fundamentales a la hora de diseñar interfaces gráficas.

No se hablará más de CSS ni HTML por la sencillez de ambos.

5.3.3.4. JAVASCRIPT

JavaScript se trata de un lenguaje de programación nacido inicialmente para interpretar código del lado del cliente en aplicaciones web, aunque con el paso del tiempo se acabó convirtiendo en un lenguaje completo y actualmente se puede utilizar para diseñar aplicaciones en su totalidad, tanto aplicaciones web como backends de otras aplicaciones.

Algunas de las ventajas de JavaScript son las siguientes (Zubikarai, 2021):

- **Eficiencia:** En su versión ejecutada en el lado del servidor, siempre y cuando no requiera recursos externos, JavaScript no se ve retrasado por el backend de la aplicación.
- **Curva de aprendizaje baja:** Al tener una sintaxis inspirada por Java es un lenguaje relativamente sencillo de aprender.
- **Popularidad:** Si ya de por sí se podía encontrar JavaScript por cualquier aplicación web, con la incorporación de Node, ahora se puede encontrar JavaScript en una gran cantidad de backends, con su popularidad en auge constante.
- **Compatibilidad:** Gracias a su soporte de PHP y Perl, JavaScript puede ser utilizado en cualquier aplicación web.
- **Baja carga de servidor:** Al ser un lenguaje client-sided reduce la demanda de las aplicaciones en los servidores, pudiendo erradicar la necesidad de un servidor en aquellas aplicaciones más sencillas.

El proyecto realizado, al estar realizado con Vue, se encuentra programado casi en su totalidad con JavaScript

5.3.3.5. AXIOS

Axios.js se trata de una biblioteca de JavaScript utilizada para realizar peticiones HTTP desde Node.js o peticiones XMLhttpRequest desde el servidor.

Si bien esto ya es algo que realizan varias bibliotecas, Axios ofrece un par de diferencias con respecto a sus competidores (Kollegger, 2018):

- Axios pretende mejorar el proceso del método fetch(), este realiza una petición inicial y nos devuelve un objeto que debe ser, posteriormente, pasado al método json() para ser convertido en datos utilizables. Además de esto, fetch() no siempre detecta errores producidos en esta petición, pues solo devuelve error en situaciones de error muy concretas. Axios pretende solucionar ambas situaciones mediante el método axios.get(), este método, utilizado de igual manera que fetch(), nos devuelve directamente los datos solicitados sin necesidad de utilizar el método json() y, además, en caso de error, activaría el bloque catch() automáticamente.

Como conclusión, Axios no es más que una mejora en la calidad de vida con respecto a los métodos convencionales de petición de datos mediante el protocolo HTTP.

5.3.3.6. CORS

Cors, por su nombre en español "Intercambio de Recursos de Origen Cruzado), se trata de un estándar utilizado en HTTP añadiéndole cabeceras adicionales que permitan que un agente obtenga permisos para poder acceder a los recursos requeridos de un servidor, en un dominio distinto al que este agente pertenece.

Por razones de seguridad las peticiones de tipo XMLhttpRequest o las realizadas mediante fetch() siguen la política de mismo origen (que no permiten realizar estas solicitudes a dominios diferentes al origen), es por esto que se deben añadir las cabeceras CORS para poder acceder a dominios externos (MDN, 2022).

5.3.3.7. SOCKET.IO

Socket.io se trata de una biblioteca que permite la comunicación rápida, bidireccional y basada en eventos entre un cliente y un servidor.

Se basa en el protocolo de TCP WebSocket añadiendo garantías extra como la reconexión automática.

Actualmente las 3 últimas herramientas mencionadas se utilizan en el sistema creado para realizar la comunicación con el servidor: Axios es usado para las peticiones GET por parte del usuario, CORS se utiliza para añadir las cabeceras necesarias para permitir al cliente realizar la petición al servidor (que se encuentra en un dominio diferente) y, socket.io, es usado para establecer una serie de eventos y métodos para realizar la conexión inicial al servidor por parte del cliente y de la fuente (Socket.IO, 2022).

5.3.3.8. VUE.JS

Vue.js fue explicado previamente en su correspondiente apartado, sin embargo, dentro de él fueron utilizadas una serie de bibliotecas y herramientas que amplían su funcionalidad, estas se exponen en este apartado:

5.3.3.8.1. PRIMEVUE

PrimeVue se trata de una biblioteca de interfaz de usuario para Vue, añadiendo más de 90 componentes, plantillas y bloques completos de elementos de interfaz. Compatible actualmente con Vue2 y Vue3.

Algunas de sus mayores ventajas comprenden su gran cantidad de documentación oficial, su compatibilidad con Vue3, la cual, su competencia más cercana como Vuetify aún no posee, su enorme cantidad de componentes y su facilidad para cambiar la aplicación de un tema a otro.

En el proyecto realizado se utiliza, como motivo principal, por ser una de las pocas bibliotecas de interfaz de usuario compatibles con Vue3, algunos de los componentes utilizados son sus Cards y sus Dividers.

5.3.3.8.2. VUE I18N

Vue I18n se trata de un plugin de Vue dedicado a la internacionalización de la aplicación, entre otras cosas, la versión para Vue3 permite localizar una aplicación al máximo, más allá de traducirla, permitiendo localizar elementos más concretos como los números o las fechas y horas.

En el proyecto este plugin fue utilizado a la hora de añadir soporte bilingüe español-inglés.

5.3.3.8.3. JSDOC-VUEJS

Esta biblioteca de Node.js permite adaptar la popular API de documentación para JavaScript JSDoc a Vue, su funcionalidad es exactamente la misma que la de JSDoc y es compatible con él. JSDoc-Vuejs no pretende sustituir el uso de JSDoc, si no que añade un par de etiquetas para facilitar la correcta documentación de un archivo de Vue, como, por ejemplo, etiquetas para documentar los props, data o métodos de Vue.

5.3.3.8.4. BOOTSTRAP VUE

Bootstrap Vue se trata de una biblioteca adaptada a Vue del famoso framework de CSS Bootstrap. Provee una serie de componentes de Vue construidos mediante Bootstrap HTML y decorados con CSS mediante las clases que Bootstrap Vue provee.

A diferencia de Bootstrap en su forma base, Bootstrap Vue proporciona alguna de las características que definen a Vue como ser modular, responsivo y accesible (Khadir, 2022).

5.3.3.8.5. FONTAWESOME

Si bien Fontawesome no es una biblioteca propia de Vue, este tiene algo en común con varias de las herramientas citadas en este apartado, y es que se utiliza para mejorar la interfaz de usuario. Fontawesome, concretamente, se trata de una base de datos de iconos que pueden ser añadidos a la interfaz de diversas aplicaciones, estos iconos vienen en diferentes tamaños y formatos, permitiendo así reemplazar las imágenes por iconos.

5.3.4. HERRAMIENTAS AUXILIARES

En este último apartado sobre herramientas utilizadas en el proyecto se hablará sobre aquellas herramientas auxiliares, se ha considerado como "herramienta auxiliar" a aquellas herramientas que no han tenido un papel en el desarrollo del proyecto per sé, pero que han sido utilizadas bien sea para documentar el proyecto o generar elementos utilizados en el proyecto posteriormente.

5.3.4.1. GIT, GITHUB, GITHUB DESKTOP

Git se trata de un software de control de versiones centrado en ser lo más eficiente y simple posible, esto, sumado a su altísima popularidad hace que sea la opción predilecta a la hora de necesitar un mecanismo de control de versiones o como medio para almacenar o compartir código. Además de esto, se trata de una herramienta de libre uso.

Por otro lado, GitHub se trata de la plataforma web colaborativa que trabaja con Git.

Por último, GitHub Desktop se trata de la aplicación de escritorio de GitHub, la cual, utilizando Git y un IDE como Visual Studio Code es capaz de reconocer los cambios realizados en el código y añadirlos al repositorio de GitHub del proyecto en cuestión, además de esto ofrece los mismos mecanismos de control de versiones de GitHub como el poder descargar repositorios o ramificarlos, entre otros.

5.3.4.2. AMAZON WEB SERVICES

Amazon Web Services (AWS) se trata de una plataforma web que ofrece una serie de servicios entre los que destacan bases de datos, almacenamiento, inteligencia artificial, etc.

Concretamente, para el proyecto a desarrollar, se utilizará AWS para mantener en activo el servidor Express backend y su base de datos PostgreSQL.

5.3.4.3. DBDIAGRAMS.IO

DBDiagrams.io se trata de una herramienta de desarrollo de bases de datos relacionales, se diferencia del resto en que su método de uso implica escribir código en DSL, de esta manera se genera una BBDD relacional rápida e intuitivamente. Además de esto, es compatible con PostgreSQL por lo que exportar estas bases de datos diseñadas resulta increíblemente sencillo.

5.3.4.4. DIAGRAMS.NET, DRAW.IO

Diagrams.net, también conocido como draw.io, es una plataforma web de diseño de diagramas, es especialmente utilizada por tener soporte con UML. Esta herramienta ha sido utilizada para diseñar la totalidad de los diagramas presentes en esta memoria y en los diferentes anexos de la documentación del proyecto.

5.3.4.5. MICROSOFT PROJECT

Microsoft Project es una de las herramientas presentes en el paquete de Office 365 desarrollado por Microsoft. Project, concretamente, se trata de una herramienta de gestión de proyectos.

Para el sistema diseñado, Microsoft Project ha sido utilizada a la hora de crear la planificación temporal del proyecto y el diagrama de Gantt asociado a esta.

5.3.4.6. EZESTIMATE

Se trata de una herramienta CASE en forma de aplicación de escritorio de Windows que permite calcular la estimación temporal y de costes de un sistema a partir de los requisitos del sistema y su complejidad.

5.3.4.7. CLIPPY

Clippy se trata de una herramienta web utilizada para crear paths de CSS, estos se utilizan a modo de máscara en imágenes, permitiendo realizar "cortes" aconvencionales sobre estas. Si bien Clippy no tiene soporte para el path libre, permite el uso de paths poligonales con una interfaz informativa que sirve como soporte a la hora de diseñar el path deseado (Clippy, 2020).

5.3.4.8. COOLORS

Coolors se trata de una herramienta web para la generación de paletas de colores, esta posee todas las funcionalidades que se esperan en una herramienta de este estilo, sin embargo, ofrece una gran cantidad de funcionalidades de calidad de vida, como el poder crear paletas aleatorias pulsando un solo botón o, aún mejor, bloquear un color que te guste y crear una paleta en función a este (Bianzi, 2020).

En el proyecto, Coolors fue utilizada a la hora de desarrollar la paleta que se puede observar en la **Figura 6.**

6. ASPECTOS RELEVANTES DEL DESARROLLO

Durante este apartado se indagará sobre las diferentes partes del desarrollo del proyecto, con especial hincapié en el proceso de desarrollo desde el punto de vista del proceso unificado, algunos de los aspectos que se tratarán son el ciclo de vida del proyecto, elicitación de requisitos, análisis y diseño del sistema y su implementación y pruebas pertinentes.

Siguiendo este proceso incremental dictado por el proceso unificado se plantea la siguiente división:

- En primer lugar, se realizará una explicación del proceso de desarrollo como ya se ha dicho, desde el punto de vista de la ingeniería de software.
- En segundo lugar, se realizará un desglose de la arquitectura del sistema, de manera modular, explicando la toma de decisiones realizada en cada sección, investigaciones previas realizadas, tecnologías utilizadas, etc.
- Por último, se presentarán una serie de aspectos globales del sistema, comenzando por las diferentes vistas presentes en el mismo, seguido de diferentes parámetros de seguridad implementados y, para finalizar, detalles sobre el despliegue del mismo.

Figura 26. Logo del proyecto StreamVector

6.1. MARCO DE TRABAJO

Si bien ya se ha explicado con anterioridad que el marco de trabajo utilizado es el dado por el Proceso Unificado, será en esta sección donde se hable en detalle de las diferentes etapas de este proceso, destacando aquellos elementos más relevantes de cada una, dividiendo esta explicación en las 6 etapas originales del Proceso Unificado (Modelo de Negocio, requisitos, análisis, diseño, implementación y pruebas).

Cabe aclarar que, al tratarse de un modelo incremental, algunas etapas, especialmente aquellas pertenecientes a las primeras iteraciones, serán omitidas en pos de explicar con mayor claridad lo investigado una vez se llegue a la etapa en que esta investigación se refine.

Como bien se sabe a estas alturas, el desarrollo fue dividido en 4 iteraciones, las cuales a su vez se dividen en estas 6 etapas del proceso unificado. Se comenzará a presentar, en primer lugar, la etapa de modelo de negocio, hasta llegar a la última etapa de pruebas, esto sucederá para cada una de las 4 iteraciones.

6.1.1. MODELO DE NEGOCIO

Es en esta etapa donde existe el punto de investigación del proyecto, este proceso de investigación fue dividido entre las 4 iteraciones de tal forma que cada iteración, exceptuando la primera, se centrara en diferentes objetivos del sistema.

Se explica a continuación de forma más detallada cada uno de los modelos de negocio sucedidos en cada iteración del Proceso Unificado:

- **Iteración 1:** El objetivo principal del modelo de negocio durante esta iteración era aclarar todas las dudas presentes en aquel momento sobre el desarrollo del sistema, para esto, se realizó una búsqueda de aplicaciones que integrasen el módulo de retransmisión WebRTC. A partir de esto, se descubrió que WebRTC se encuentra presente en varias tecnologías exitosas como Discord o Google Meets.
  Sabiendo esto se realizó un análisis de riesgos para establecer una visión global del sistema que se pretendía desarrollar y establecer límites en el desarrollo para facilitar la descripción del mismo.
  Por último, hecho esto, se realizó un análisis de los elementos presentes en varias aplicaciones similares, aunque estas no incorporasen WebRTC, como puede ser Google Stadia u otras aplicaciones con catálogos de videojuegos como Steam, gracias a esto se esbozó una primera imagen visual del proyecto, ayudando así a definir qué elementos adicionales estarían presentes en este (catálogo, lista de amigos, perfil, …).
- **Iteración 2:** Con todo esto hecho, es en esta iteración donde se procede a realizar una investigación sobre un módulo de gestión en concreto, se decidió comenzar por aquellos que llevarían más tiempo desarrollar e implementar, ya que serían los que más se beneficiarían de refinamientos posteriores. Estos dos módulos seleccionados fueron el módulo de retransmisión y el módulo de captura de inputs.
  Fue gracias a esta investigación como se descubrieron una serie de bibliotecas de Node.js útiles como Robotjs, que acabaría siendo la biblioteca usada para toda la totalidad de captura de requisitos, en cuanto a la retransmisión, se investigó en profundidad el funcionamiento de la biblioteca Socket.io y del propio WebRTC, para comprender en profundidad sus eventos y métodos de retransmisión de vídeo, respectivamente.
- **Iteración 3:** En esta iteración se comienza a investigar sobre posibles métodos y herramientas para desplegar un servidor web en el sistema, ya que se debía dejar de

utilizar el servidor de prueba de WebRTC y pasar a utilizar un servidor propio para el proyecto que permitiera, además de servir para comunicar cliente y fuente, servir como base de datos. Mediante esta investigación se descubre la plataforma de Amazon Web Services y, con los servicios que esta ofrece y su facilidad de uso, se decidió seguir adelante con ella. Además de esto, se comenzó a investigar en profundidad algún elemento secundario del sistema como el catálogo, observando patrones estructurales y de diseño utilizados por otras plataformas, especialmente Google Stadia.

- **Iteración 4:** En esta última iteración se realizó un análisis de riesgos final que dio como resultado una serie de cambios en el sistema, eliminando aquellos elementos que se consideraron no aptos, y añadiendo elementos nuevos obtenidos de un último análisis como, por ejemplo, permitir cambiar el idioma de la aplicación.

6.1.2. REQUISITOS

En primer lugar, se realizó una propuesta inicial de objetivos del sistema (la cual, se pretendió ser lo más estable posible, pues, un cambio en esta parte podría afectar a todo el desarrollo del sistema) gracias a la primera iteración de modelo de negocio.

A partir de esto, se realizó una primera definición de requisitos mediante la metodología de Durán y Bernárdez, estos recogieron las primeras funcionalidades que se esperaba ver presentes en la aplicación, especialmente las relativas a la retransmisión, ya que, antes esto fuese definido más etapas de refinamiento se podrían aplicar sobre sus requisitos.

La visualización del diagrama de Casos de Uso final se puede observar en la **Figura 27:**

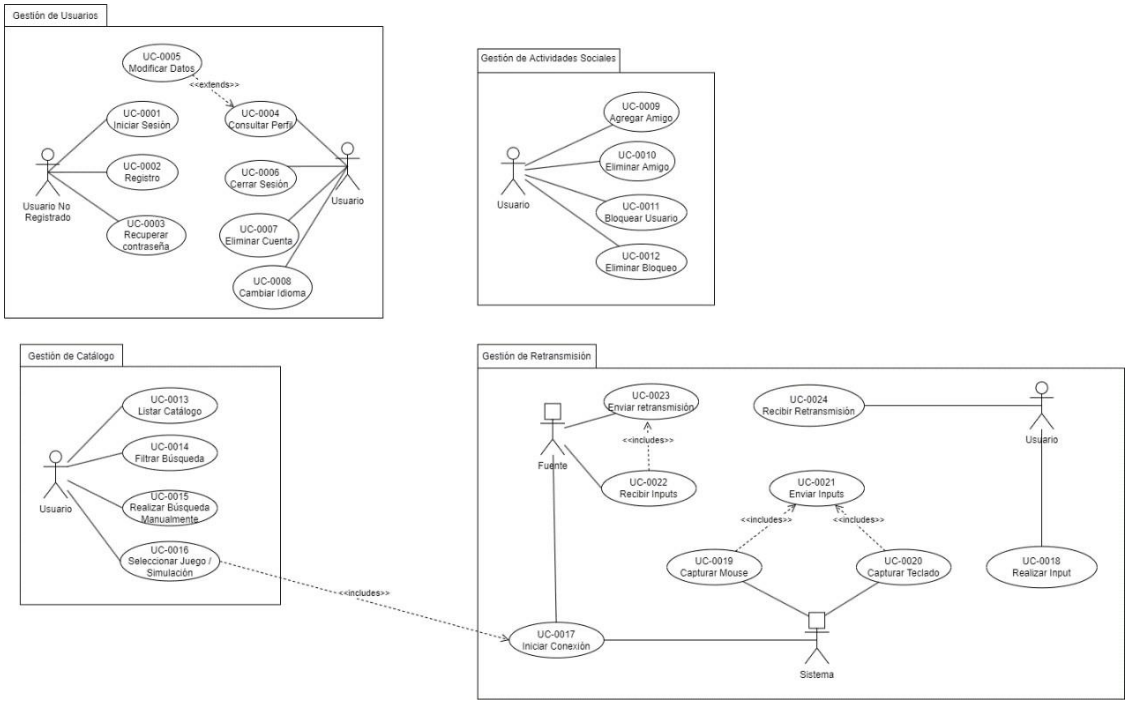


Figura 27. Detalle del diagrama de Casos de Uso

A partir de este diagrama de Casos de Uso se realizó, como ya se ha mencionado, la primera versión de los requisitos del sistema, en concreto, se definieron los requisitos de información (aunque los atributos de los mismos aún no eran fieles a la versión que acabaría siendo la final), los requisitos funcionales, los requisitos no funcionales y las tablas de los actores.

En las tablas **Tabla 1**, **Tabla 2** y **Tabla 3** se pueden observar, listados, los diferentes tipos de requisitos existentes en el sistema, para observarlos en detalle se recomienda ir al "Anexo II: Especificación de Requisitos Software".

Cabe destacar que las últimas iteraciones esta elicitación de requisitos sufrió algunos cambios menores, especialmente los requisitos funcionales. Esto se debe a que, ya que estos se recogieron de forma previa al desarrollo del sistema, surgieron muchos cambios a la hora de implementar el requisito descrito inicialmente. Tras estos cambios se llegó a la versión final de esta etapa.

Tabla 1. Detalle sobre la tabla de requisitos de información

ID	Nombre	Descripción
IRQ-0001	Información de Usuario	El sistema deberá almacenar la información correspondiente a los usuarios del mismo, concretamente:
IRQ-0002	Información de Videojuego	El sistema deberá almacenar la información correspondiente a los videojuegos del mismo, concretamente:

Tabla 2. Detalle de tabla de requisito funcional

ID	Nombre	Descripción
UC-0001	Iniciar Sesión	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite iniciar sesión en el sistema
UC-0002	Registro	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite registrarse en el sistema
UC-0003	Recuperar Contraseña	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite recuperar su contraseña
UC-0004	Consultar Perfil	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite consultar su perfil personal
UC-0005	Modificar Datos	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite modificar sus datos personales

UC-0006	Cerrar Sesión	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite cerrar su sesión
UC-0007	Eliminar Cuenta	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite dar de baja su cuenta
UC-0008	Cambiar Idioma	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite cambiar el idioma de la aplicación*
UC-0009	Agregar Amigo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite agregar a otro usuario como amigo
UC-0010	Eliminar Amigo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite eliminar a otro usuario de su lista de amigos
UC-0011	Bloquear Usuario	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite bloquear a otro usuario
UC-0012	Eliminar Bloqueo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite ver el catálogo
UC-0013	Listar Catálogo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite agregar a otro usuario como amigo
UC-0014	Filtrar Búsqueda	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite eliminar a otro usuario de su lista de amigos
UC-0015	Realizar Búsqueda Manualmente	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite bloquear a otro usuario
UC-0016	Seleccionar Juego/Simulación	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite ver el catálogo
UC-0017	Iniciar Conexión	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba una petición de inicio de conexión
UC-0018	Realizar Input	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el actor ACT-0002 Usuario realice un input
UC-0019	Capturar Mouse	El Sistema deberá comportarse como se indica en el siguiente caso de uso

		cuando el Actor ACT-0005 Sistema reciba un evento de captura de mouse
UC-0020	Capturar Teclado	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba un evento de captura de teclado
UC-0021	Enviar Inputs	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema envíe un input
UC-0022	Recibir Inputs	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0004 Servidor Fuente reciba un input
UC-0023	Enviar Retransmisión	El Sistema deberá comportarse como se indica en el siguiente caso de uso de manera periódica cuando el Actor ACT-0004 Servidor Fuente se conecte con el actor ACT-0002 Usuario
UC-0024	Recibir Retransmisión	El Sistema deberá comportarse como se indica en el siguiente caso de uso de manera periódica cuando el Actor ACT-0002 Usuario se conecte con el actor ACT-0004 Servidor Fuente

Tabla 3. Detalle de tabla de requisito no funcional

ID	Nombre	Descripción
NFR-0001	Portabilidad	El sistema debe poder dar soporte a varios sistemas operativos (Windows, Linux, iOS…)
NFR-0002	Concurrencia	El sistema debe soportar el funcionamiento con varios clientes simultáneamente
NFR-0003	Eficiencia (Inputs)	El sistema deberá registrar e implementar los inputs del cliente en tiempo real, con el menor retardo posible
NFR-0004	Eficiencia (Retransmisión)	El sistema deberá retransmitir el vídeo al cliente con el menor retardo posible y con la mayor calidad de imagen posible
NFR-0005	Escalabilidad	El sistema deberá poder realizar múltiples retransmisiones simultáneamente
NFR-0006	Roles	El sistema deberá poseer una jerarquía de roles de usuario, con un superusuario que gestione las acciones pertinentes.
NFR-0007	Seguridad (Retransmisión)	El sistema deberá asegurar la señal transportada en la retransmisión
NFR-0008	Seguridad (BBDD)	El sistema deberá ofrecer un almacenamiento seguro de los datos sensibles del usuario

NFR-0009	Servidor Web	El sistema poseerá un servidor propio que gestione el backend, así como la retransmisión servidor fuente-cliente
NFR-0010	Usabilidad	El sistema poseerá una interfaz de usuario fácilmente comprensible por el cliente
NFR-0011	Accesibilidad	El sistema poseerá una interfaz de usuario responsive

6.1.3. ANÁLISIS

Al igual que con la elicitación de requisitos, esta etapa comenzó siendo un esbozo de lo que acabaría siendo en la iteración final. En esta primera iteración se decidió diseñar el modelo de dominio para poder utilizarlo como referencia a lo largo de esta primera iteración, y así ampliar lo ya descrito en la etapa de captura de requisitos.

A partir de este modelo de dominio inicial, se presentó una realización de casos de uso orientada al análisis, definiendo así las comunicaciones sucedidas en los diferentes Casos de Uso, por último, se presentó la vista arquitectónica inicial del sistema, aún a alto nivel, utilizando las clases de análisis aparecidas en los diagramas de secuencia.

Se ha decidido incluir el modelo de dominio generado durante esta etapa en la **Figura 28**, además de una tabla asociada en la **Tabla 4** con sus definiciones asociadas. Para ver el resto de la etapa de análisis se recomienda ir al "Anexo III: Análisis de requisitos".

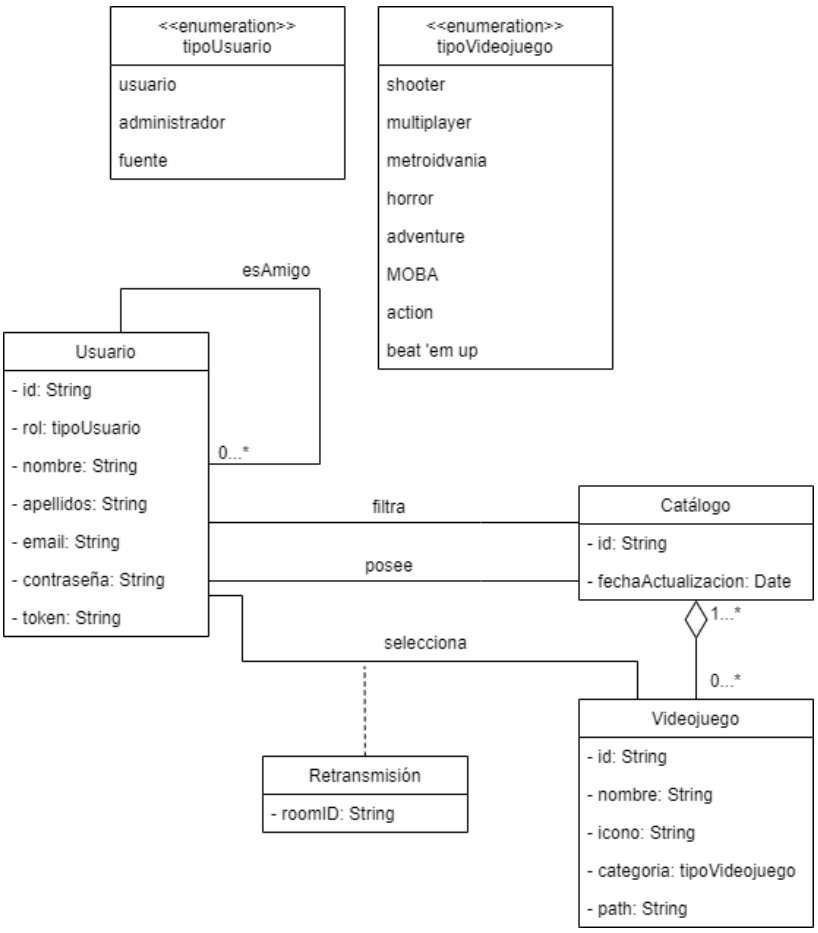


Figura 28. Detalle del modelo de dominio

Tabla 4. Descripción de clases del modelo de dominio

Clase	Descripción
Usuario	Se trata de la clase que representa la sesión de un usuario activo, esta almacena sus respectivos datos de id, rol, nombre, apellidos, email, contraseña y su token único.
Catálogo	Colección de videojuegos, si es accedido por un usuario este puede ser filtrado o permitir realizar una búsqueda sobre él
Videojuego	Clase que representa los datos estáticos de un videojuego en el sistema, esto incluye su nombre, categoría, icono y ruta de ejecución. Posee una relación de composición con el catálogo, pudiendo existir el mismo juego en varios catálogos de usuarios diferentes.
Retransmisión	Se trata de una clase de asociación generada cuando un usuario selecciona un videojuego al que jugar, esta retransmisión posee un token único relativo a esta relación cliente-videojuego, este token será el que identifique a la room a la que se unan los equipos cliente y fuente para retransmitir y recibir vídeo, respectivamente.

6.1.4. DISEÑO

La etapa de diseño del sistema es la primera en verse afectada por la escasez de información en las primeras iteraciones, durante la primera iteración se decidió investigar sobre los patrones utilizados por Vue (MVVM) y por el proyecto en sí (cliente-servidor), sin embargo, hasta que el código y los métodos no fuesen diseñados e implementados, los diagramas de secuencia – diseño no podían ser realizados.

Con el paso de las iteraciones se realizaron los primeros diseños del sistema, como fueron el diseño de la Base de Datos en la iteración 3 y el diseño de la vista de usuario en la Iteración 2 (mediante wireframes), por último, se diseñaron las diferentes interfaces del sistema, un detalle sobre estas puede observarse en la **Figura 32**

En cuanto al diseño arquitectónico, este es lo suficientemente amplio como para ser explicado por su parte en su sección propia más adelante en esta memoria. Para ver más en detalle cualquiera de estos conceptos se recomienda observar el "Anexo IV: Diseño del sistema software".

6.1.5. IMPLEMENTACIÓN

La implementación del sistema ha sido realizada, como ya sabemos, mediante el framework de Vue, que recoge varios lenguajes de programación como JavaScript, HTML y CSS, además de Node y su gestor de paquetes como herramienta auxiliar para gestionar bibliotecas externas.

Se hablará más de la implementación concreta de cada parte del sistema en su respectiva sección, sin embargo, a modo de presentación, se procede a listar los diferentes módulos del sistema que podemos encontrar en la **Figura 29** y una pequeña explicación sobre su funcionalidad.

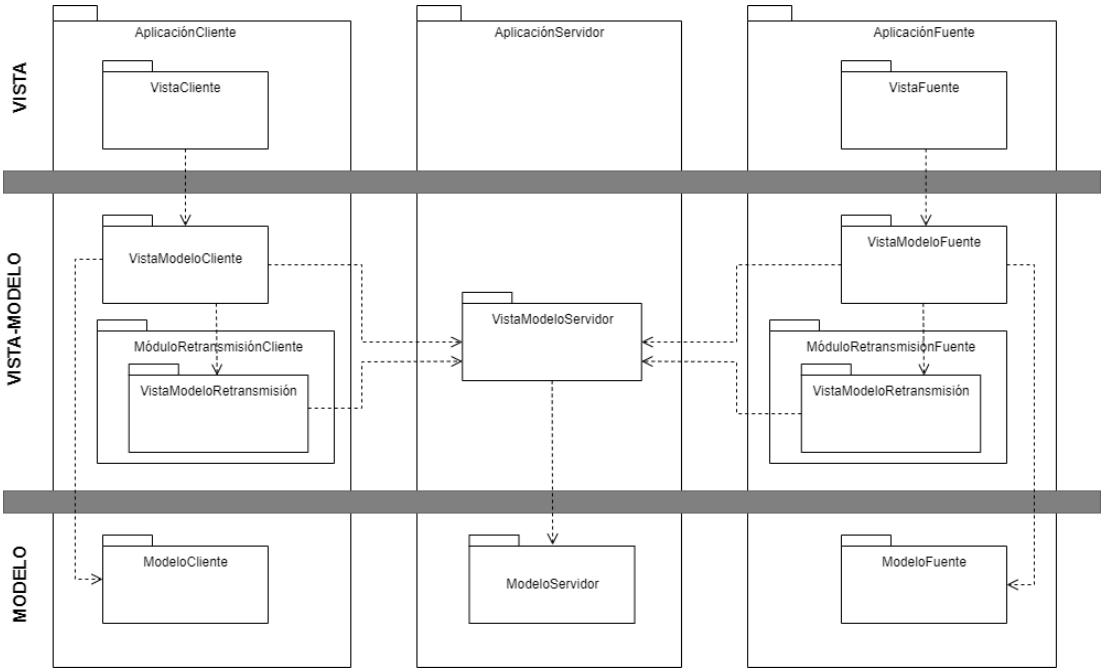


Figura 29. Detalle de la vista arquitéctonica en el diseño

Veamos los diferentes módulos listados:

- Dentro de la Aplicación cliente existen dos secciones claramente distinguidas entre sí, en primer lugar, tenemos el módulo de retransmisión, en el cual se realizan todas las comunicaciones con el servidor web relativas a la retransmisión, será aquí donde usemos WebRTC, entre otras herramientas. El otro módulo se trata de todo lo ajeno a la retransmisión, aquí nos encontramos con elementos más dispares, como el catálogo, el perfil, el login, etc, veremos en futuros apartados estos módulos más en detalle.
- En cuanto al servidor fuente no se detallará mucho, ya que posee los mismos módulos que la aplicación cliente con diferencias menores.
- Por último, el servidor posee dos elementos claramente distinguidos, por un lado, tiene la parte dedicada a responder peticiones relacionadas con la retransmisión y, por otro lado, las dedicadas a responder peticiones de Axios.

6.1.6. PRUEBAS

A lo largo de todo el desarrollo el tipo y número de pruebas fue cambiando drásticamente, adaptándose a las necesidades de cada iteración, se procede a describir cada tipo de pruebas realizado a lo largo del desarrollo del proyecto:

- El primer tipo de pruebas utilizadas son pruebas de funcionamiento sobre cada módulo, asegurándose de que sus acciones se corresponden con los requisitos descritos.
- En segundo lugar, se realizaron pruebas de integración y despliegue, especialmente en la iteración 3 una vez fue desplegado el servidor Express, estas pruebas tienen el objetivo de comprobar la correcta funcionalidad de las peticiones HTTP, esto se realiza por medio de logs.
- Una vez asegurado el correcto funcionamiento del sistema y su correcto despliegue e integración, se propuso realizar una serie de pruebas de carga, iniciando un elevado número de retransmisiones simultáneamente, con varios clientes enviando inputs a su vez.

6.2. ARQUITECTURA

Será este el apartado mencionado durante toda la sección anterior en el que se entrará en detalle sobre todos los diferentes módulos del sistema, esto quiere decir que se hablará en profundidad sobre su funcionalidad, decisiones tomadas en función de investigaciones previas y herramientas utilizadas.

6.2.1. ARQUITECTURA MVVM

Si bien ya se ha hablado extensivamente de la arquitectura MVVM y de su utilización en Vue, no se ha mencionado la decisión de seguir este patrón ni su investigación previa.

La arquitectura MVVM se encuentra integrada de manera nativa en Vue, el cual fue diseñado para ser utilizado con este patrón arquitectónico, esto proporciona unas ventajas a Vue muy características, y es que, al asociar la Vista del patrón MVVM con el DOM de Vue, este permite aislarlo de tal forma que queda relegado a funciones meramente visuales, gracias a esto Vue es capaz de destinar más recursos a su DOM virtual y poder crear una experiencia de usuario dinámica sin necesidad de recargar la página, además de esto, la divisón dentro de un archivo de Vue suele venir dada por 2 claras secciones: la sección “template” asociada al frontend y a eventos producidos por interacciones del usuario con este, la sección “script” destinada a JavaScript y asociada con la Vista Modelo (aquí también se puede encontrar el Modelo en forma de objetos JavaScript o JSON planos). Por tanto, se puede concluir diciendo que Vue proporciona una gran cantidad de comodidades al utilizar este patrón arquitectónico.

En cuanto a la parte no visual del sistema, este patrón se sigue aplicando, separando los datos planos (Modelo) del resto de elementos de gestión (Vista Modelo).

6.2.2. MÓDULO DE RETRANSMISIÓN

Este primer módulo a ser expuesto se trata del módulo más importante y característico del proyecto, se trata del módulo encargado de realizar la conexión, envío y recepción de audio y vídeo entre la fuente-servidor-cliente y viceversa.

La implementación de este módulo se realiza mediante el uso de JavaScript junto a las bibliotecas de Socket.io (para iniciar la conexión) y WebRTC (para enviar/recibir vídeo). La funcionalidad de esto ya fue discutida en apartados anteriores, por lo que se ilustrará el caso de uso “Iniciar Conexión” mediante el diagrama de secuencia – diseño de la **Figura 30**

En cuanto a la captura de inputs, esta se trata de la segunda gran funcionalidad de este módulo, para poder hacer esto se ha valido de los propios eventos de Vue y Electron para la detección de inputs, Vue para capturar inputs de ratón y teclado y Electron para capturar inputs relacionados con la pantalla como la posición del mouse.

Una vez detectado el input con el handler adecuado, este es normalizado (por ejemplo, en el caso de las coordenadas del mouse habría que hallar su nueva posición en función de su posición anterior y el desfase detectado en el momento de llamar al handler) y enviado al equipo fuente, aquí es donde se aplica el input gracias a la biblioteca de Robot.js de la cual ya hemos hablado previamente en esta memoria.

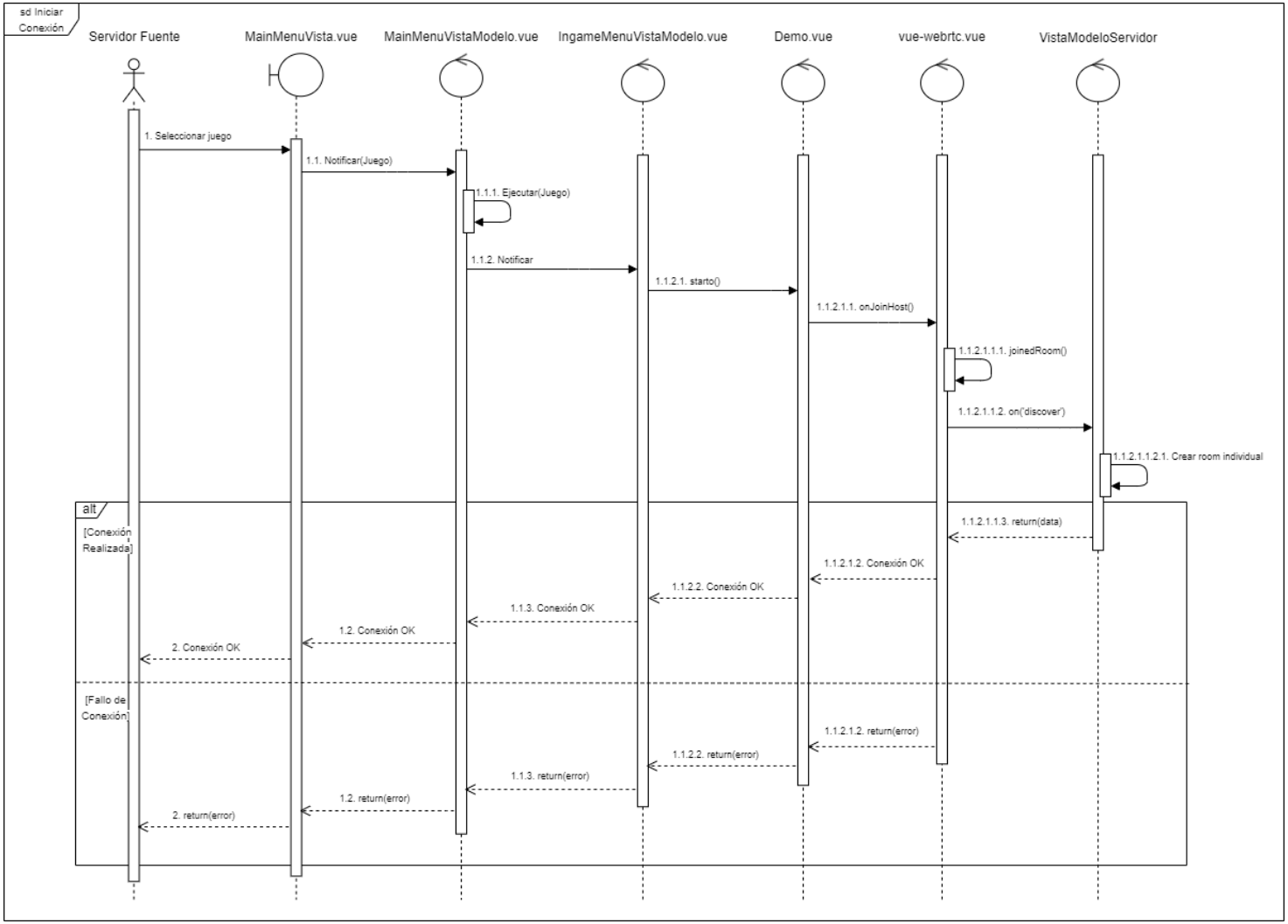


Figura 30. Detalle del diagrama de secuencia - diseño "iniciar conexión"

6.2.3. APLICACIÓN SERVIDOR

La aplicación servidor fue construida sobre Express, como ya hemos explicado en puntos anteriores, además de eso, incluye una base de datos PostgreSQL para almacenar y proteger los datos del usuario.

A estas alturas, se procede a presentar un diagrama que representa, mediante iconos de las herramientas utilizadas, la arquitectura de la conexión entre un equipo y un servidor en este proyecto. Se puede observar en la **Figura 31** (bezkoder, 2022):

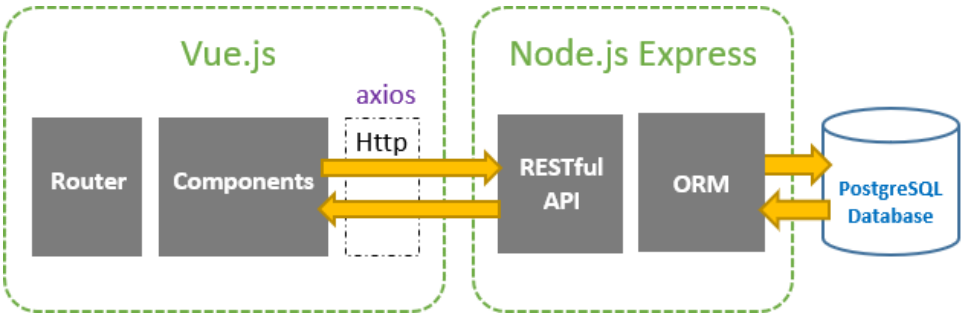


Figura 31. Diagrama arquitectónico de la conexión cliente-servidor en el proyecto

6.2.4. PATRÓN MVVM Y PARÁMETROS DE SEGURIDAD

El patrón MVVM ya ha sido expuesto de manera extensiva a lo largo de esta memoria, sin embargo, debemos aclarar un par de elementos que este aporta a la interfaz de Vue:

- Al lograr separar el Modelo y la Vista Modelo de la Vista se crea una capa de abstracción entre el usuario y el backend del proyecto, esto permite crear experiencias de usuario eficientes y relativamente fáciles de diseñar.
- Otro de los grandes puntos de Vue es su cantidad de eventos añadidos en su interfaz, con ellos se pueden recoger acciones del cliente como pulsar un botón sin necesidad de añadir funcionalidad a la Vista

A la hora de la seguridad nos debemos preguntar si el cliente puede hacer accesos no permitidos a alguna de las interfaces que nuestro servidor Express provee, estas interfaces pueden verse en la Figura 32.

Para asegurarnos de que esto no sucede existen dos mecanismos de seguridad:

- El primero de ellos consiste en el sistema de login, para asegurarnos de que un usuario sin cuenta no pueda acceder al sistema nos valemos de la herramienta bcrypt para validar los datos de usuario.
- Por último, para evitar que un usuario sin cuenta acceda a algún endpoint del servidor Express, se utiliza un sistema de tokens, un token único es asignado a cada usuario al momento de iniciar sesión en el sistema, de este modo solo usuarios con sesiones activas tienen acceso a estos endpoints.

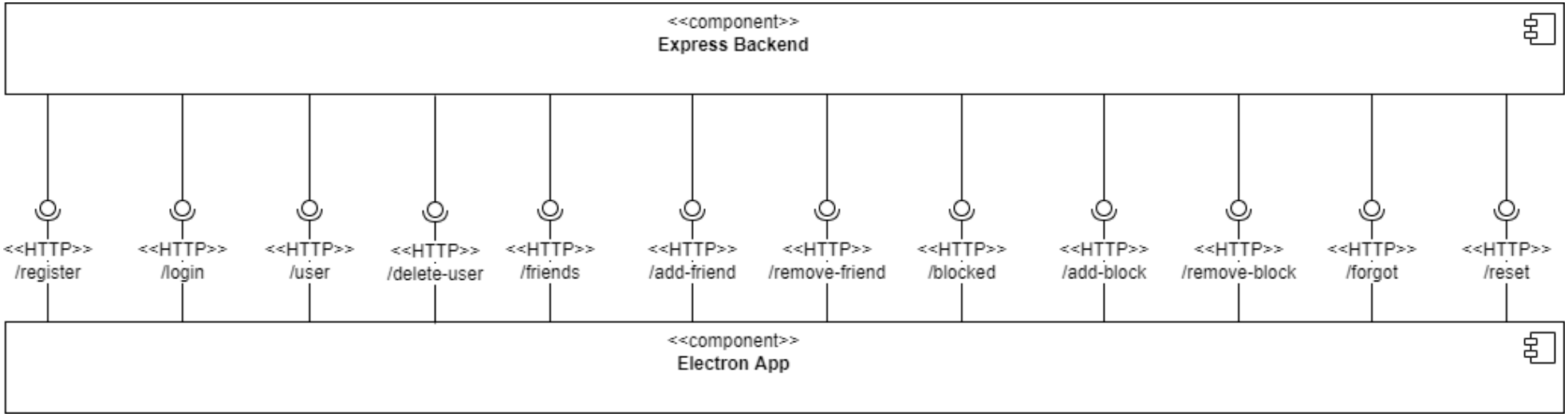


Figura 32. Detalle de las interfaces del sistema

6.3. VISTA

En este apartado se hará un análisis resumido de las diferentes vistas, explicando las herramientas utilizadas y la justificación y planificación detrás de su diseño e implementación.

6.3.3. VISTA DE REGISTRO/LOGIN

Comenzando con la primera vista de todas, se presenta la vista de login/registro, como pequeña aclaración inicial, el orden que se seguirá será el presentado en el "Anexo VI: Manual de usuario", para entender el funcionamiento y el buen uso de la interfaz se recomienda acudir a ese anexo pues, en esta memoria lo que se hará será hablar resumidamente de la vista en cuestión y de las herramientas usadas para su diseño.

A continuación, en la **Figura 33**, se presenta el detalle de la vista de login en concreto

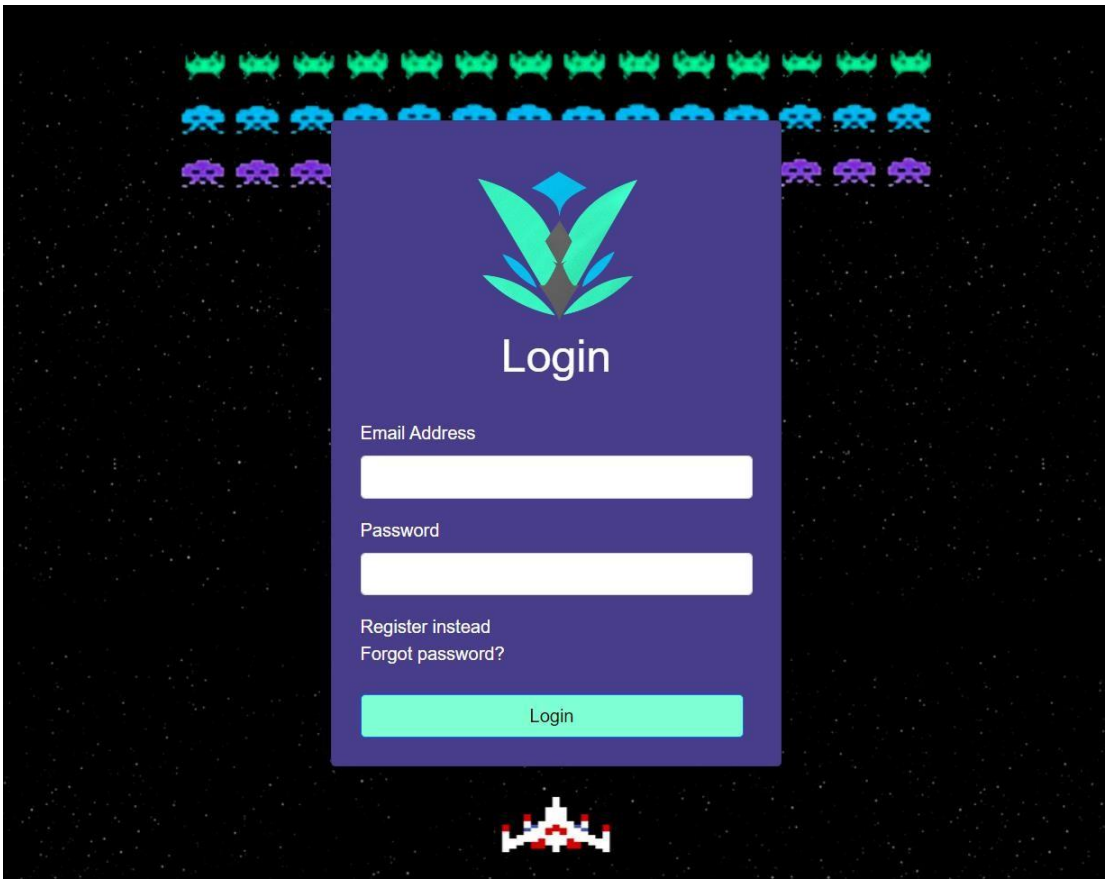


Figura 33. Detalle de la Vista de Login

A rasgos generales, esta vista ofrece al usuario el punto de entrada al sistema, realizando el login (o el registro en caso de no tener cuenta en el sistema), el usuario recibe su token que, sin que el usuario se entere, es el mecanismo que le permitirá utilizar todos los endpoints del sistema al, por ejemplo, visitar su perfil o modificar sus datos.

En cuanto a herramientas, el perfil consiste en un componente Card de PrimeVue, el cual se encuentra, a su vez, compuesto por un formulario y el logo del proyecto. De manera estética, siguiendo la temática retro tan popular en los sistemas relacionados con los videojuegos, se ha decidido añadir un fondo minimalista de una galaxia estrellada con los famosos aliens del videojuego "Space Invaders". A su vez, la nave espacial del protagonista es controlada en función del movimiento del ratón, haciendo así la interfaz un poco más interactiva y original.

6.3.4. VISTA DE CATÁLOGO

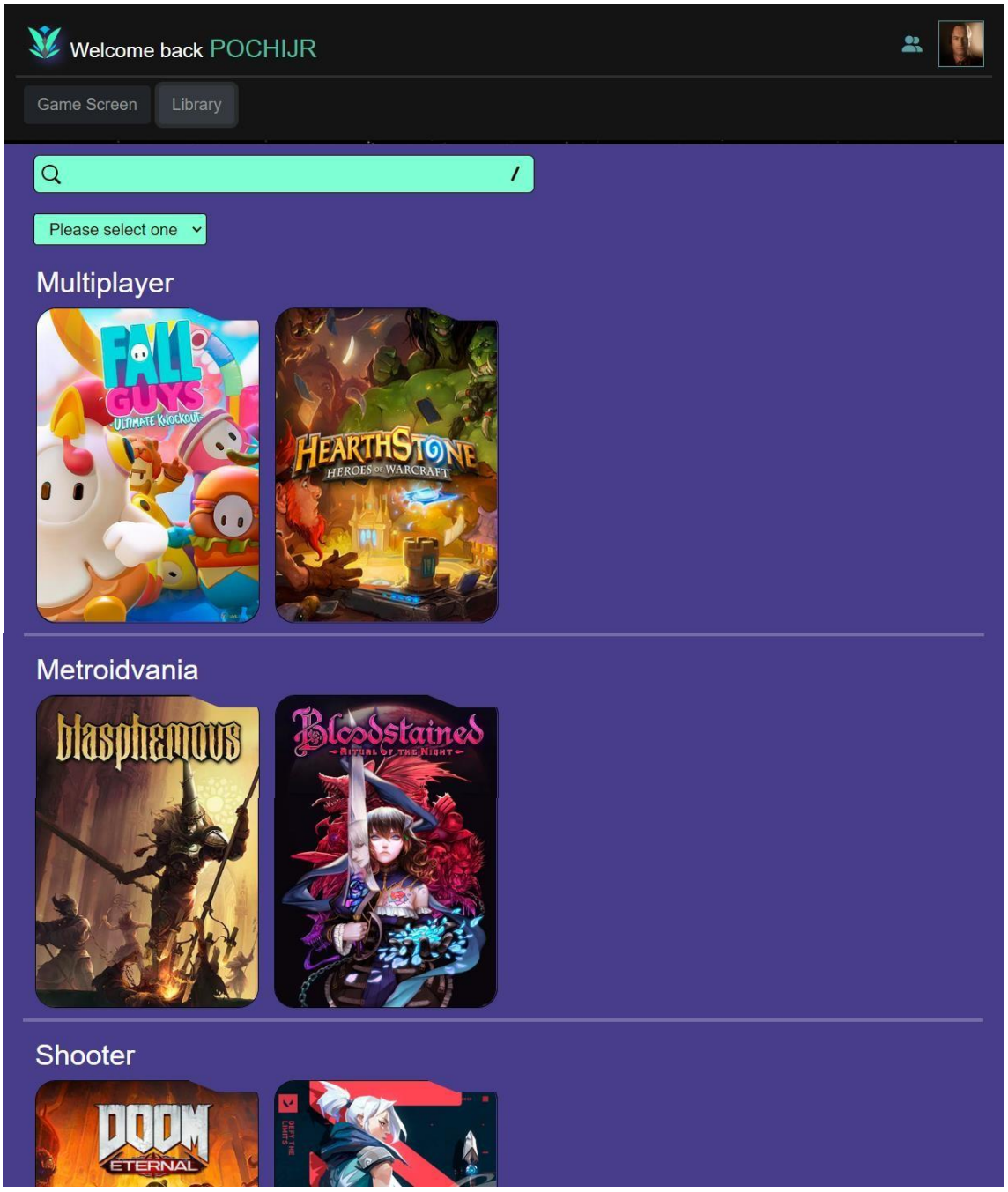


Figura 34. Detalle de la Vista de Catálogo

Siguiendo el orden de esta sección, en primer lugar, se presenta un detalle de la vista a tratar, se puede observar en la **Figura 34**.

El catálogo se trata de la vista más compleja del sistema a nivel visual, en ella tenemos una serie de imágenes listando los diferentes juegos del catálogo, estas se valen de la propiedad clip path de CSS para tener el notch de arriba a la izquierda y se encuentran separadas en función de su categoría, esto se decidió hacer así tras el análisis de plataformas similares realizado previamente en esta memoria.

En cuanto al resto de elementos, el catálogo posee una barra de búsqueda, importada del paquete de Node.js vue-searchbar, esta, tras unos cambios estéticos y de funcionalidad, nos permite filtrar el catálogo de manera dinámica, al igual que el select que se encuentra inmediatamente debajo.

Como pequeña nota, ya que esta es la primera vista en la que se encuentra presente la barra de navegación o navbar, se explicará aquí también:

La navbar se trata de un componente de Bootstrap-vue, gracias a este la propia barra de navegación poseía una serie de atributos de estilos aplicados por defecto, lo cual ayudó a su diseño. En cuanto a su contenido, posee una barra divisora que separa la zona de navegación entre menús de la zona "social", la zona de navegación entre menús se compone de la vista de catálogo y de la vista de retransmisión, mientras que la zona social nos permite acceder a la vista de perfil y a la lista de amigos (la cual se trata, de nuevo, de un paquete de Node.js modificado).

6.3.5. VISTA DE RETRANSMISIÓN

Se trata, como ya se explica en el Anexo VI, de la vista más simple del sistema, ya que simplemente se trata de dos botones que permiten al usuario y a la fuente crear un canal de comunicación entre ambos, si bien se puede realizar esta comunicación mediante la selección de un juego del catálogo, también se ofrece esta funcionalidad como un mecanismo adicional. Orientado más hacia la realización de pruebas, se podría considerar como una funcionalidad deprecada, pero que aporta al proyecto una segunda funcionalidad en cuanto a la retransmisión.

En la **Figura 35** se puede apreciar un detalle de esta vista:

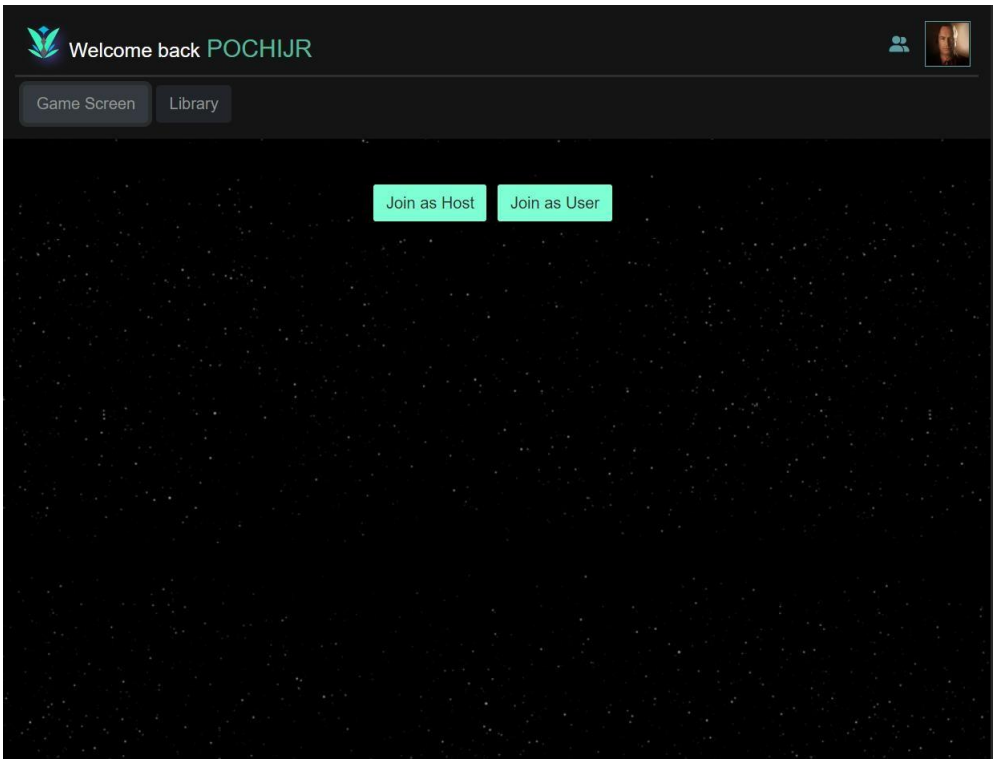


Figura 35. Detalle de la Vista de Retransmisión

6.3.6. VISTA DE PERFIL

Por último, se presenta la vista de perfil, tal y como se puede observar en la **Figura 36:**

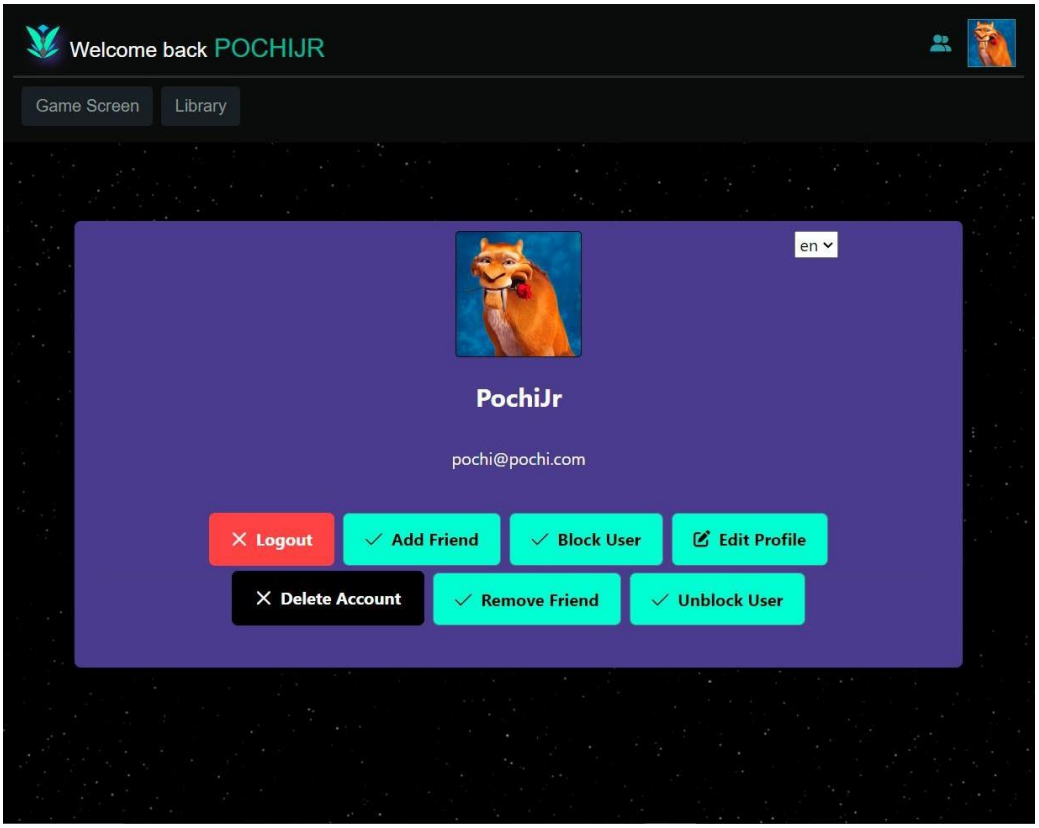


Figura 36. Detalle de la Vista de Perfil

La vista de perfil se trata, al igual que la vista de login/registro, de un componente Card de PrimeVue, este, a su vez, se encuentra compuesto por una serie de botones que aportan cierta funcionalidad al usuario, además de cierto control sobre sus datos.

Como elemento a destacar, el desplegable que se encuentra arriba a la derecha permite a la aplicación cambiar de idioma, esto se logra gracias a Vue I18N como ya se explicó en el apartado de herramientas, más allá de eso, los botones realizan llamadas al servidor Express mediante Axios, estas llamadas van desde solicitar modificar los datos del usuario, eliminar su cuenta o agregar a otro usuario como amigo, entre otros.

6.3.7. OTROS ASPECTOS

Por último, cerrando así la explicación de la vista del sistema, se procede a mencionar algunos de los aspectos que no se han tratado en los puntos anteriores o, por lo menos, no en toda la profundidad que se desearía, en concreto se destaca la usabilidad del proyecto, esto se logra mediante una interfaz sencilla y concisa, algunos de los aspectos que mantienen esta usabilidad son:

- Baja cantidad de botones, al haber pocos botones el usuario no se verá abrumado por los mismos, teniendo además botones con iconos que explican por sí solos su funcionalidad, algún ejemplo de esto es el botón que muestra la lista de amigos del usuario, representado por un icono de un grupo de personas.
- Funciones sencillas, es decir, no existe ninguna funcionalidad en el sistema que fuerce al cliente a pasar por un largo proceso de aprendizaje para utilizarla, todas las funciones ofrecidas por el proyecto son autodescriptivas y, como mucho, pedirán al cliente que escriba un correo, como en el caso del botón de añadir amigos.

7. RESULTADOS Y CONCLUSIONES

En este apartado final de la memoria se hablará sobre aquellos resultados a resaltar con respecto al proyecto, así como todas las conclusiones obtenidas durante el proceso de realización de la aplicación, por último, se tratarán una serie de líneas de trabajo futuras.

7.1. RESULTADOS

En cuanto al apartado de resultados, como se mencionó anteriormente en la sección de pruebas, se realizaron pruebas de carga de usuarios con diferentes retransmisiones en proceso, en la **Tabla 5** se pueden observar algunos de los resultados obtenidos durante estas pruebas.

Como se puede observar, no ha habido ningún tipo de problemática en ninguno de los casos resaltados.

Tabla 5. Resultados

Dato a valorar	Condiciones y resultados
Número de usuarios	Al aumentar el número de usuarios en el sistema se ha comprobado que la calidad de la retransmisión no se vio reducida, así como tampoco se produjo ningún tipode interconexión indeseada ni problemas de concurrencia.
Reducción y aumento de ancho de banda	Al reducir la calidad de la conexión a la red del usuario se podía observar una ligera reducción en la calidad de la retransmisión, sin embargo, esta reducción no fue notable excepto en casos muy drásticos en los que el cliente sufría prácticamente una desconexión a su red wifi. Un aumento en la calidad de la red wifi provocaba un ligero incremento en la calidad de la retransmisión, este era especialmente notable en la tasa de frames.
Tipo de videojuego	Probando diferentes videojuegos (especialmente aquellos con una mayor carga en el procesador y en la tarjeta gráfica) se observó que la retransmisión se veía mínimamente afectada, observando una ligera bajada en la tasa de frames únicamente en aquellos casos en los que el videojuego provocaba una enorme carga en el procesador del equipo fuente.

Otro resultado que se destaca a continuación es la calidad de la retransmisión, gracias al uso de WebRTC se ha logrado una retransmisión estable para la gran mayoría de conexiones wifi, por supuesto, en aquellas de muy mala calidad la retransmisión pierde calidad también, aunque esto al ser dinámico no presenta una mayor problemática para el cliente.

7.2. CONCLUSIONES

7.2.1. CONCLUSIONES TÉCNICAS

En primer lugar, hablando sobre los objetivos del proyecto, todos los objetivos principales y secundarios fueron cumplidos, lo cual es la primera cosa que se busca al hablar de los resultados de un proyecto, por lo que, por esa parte, se podría decir que todo ha ido mejor de lo planeado.

En cuanto a la aplicación, es capaz de contener varios usuarios y fuentes de forma simultánea, manteniendo aun así una enorme calidad de vídeo a tiempo real, he de decir que este objetivo es el que más preocupación generaba, y el uso de WebRTC junto a Robot.js ha sido el gran protagonista de que esto haya sido así.

Algunas de las otras conclusiones destacadas son:

- Haber logrado producir un prototipo funcional de una aplicación de streaming, incluyendo una simulación en tiempo real de retransmisión de vídeo y audio entre dos equipos interconectados por un servidor.
- Crear un sistema de catálogo interactivo con la opción de ser filtrado/buscado en tiempo real.
- Crear un sistema de gestión de usuarios básico que permita a estos registrarse e iniciar sesión en el sistema, así como mantener de manera persistente estos datos, así como proporcionar algún servicio de calidad de vida básico como el poder cambiar de idioma o eliminar la cuenta de forma permanente.
- Siguiendo un poco lo mencionado en el apartado anterior, se ha logrado implementar una interfaz responsive siguiendo las pautas recomendadas por Vue.
- Haber logrado seleccionar herramientas de entre un surtido amplio de estas, eligiendo aquellas que mejor se adaptaban al proyecto y a sus objetivos.

7.2.2. CONCLUSIONES PERSONALES

En cuanto a conclusiones personales, este proyecto, como ya se dijo en los objetivos personales, se trataba de un desafío a nivel personal, y, ahora, en el punto de haberlo finalizado, puedo afirmar con seguridad que se han adquirido muchos conocimientos en ramas que, anteriormente, consideraba demasiado desafiantes o inalcanzables para mí, por destacar algo, he de decir que el uso de un servidor externo y programar sus diferentes interfaces, así como los elementos de conexión al servidor y los métodos de envío de vídeo/audio han sido la parte más interesante y de la que más conocimientos se han adquirido.

Por otro lado, el haber podido aplicar todos los conocimientos aprendidos durante el Grado en Ingeniería Informática, especialmente los de Ingeniería de Software y Gestión de Proyectos, en un proyecto real ha sido una de las otras experiencias a destacar de todo el desarrollo.

Como conclusión final, este proyecto me ha ayudado a comprender y a entender la visión general de un proyecto, a comprender todos los elementos que coexisten dentro del mismo y a distinguir a qué sección del proyecto pertenecen estos y qué posibilidades ofrece cada uno.

7.3. LÍNEAS DE TRABAJO FUTURAS

Si bien este proyecto tiene un alto techo y muchísimas opciones que añadir, me gustaría destacar aquellas que considero mantienen su temática original y mejoran aquellos elementos ya presentes:

- El punto más ambicioso podría ser convertir a la propia plataforma en una gran plataforma de videojuegos, con su propio catálogo y sistema de logros, similar a otras como Steam o Epic Store, sin embargo, esto tan solo es una idea conceptual pues implicaría lanzar la aplicación al mercado.
- Permitir a otros usuarios acceder a tu partida, observando de manera compartida cómo juegas al juego seleccionado.
- Añadir más opciones de personalización al perfil de usuario (fondos, efectos en el icono, etc).
- Añadir soporte global para mando de videoconsola (Xbox, Play Station, Steam Deck, etc).
- Añadir soporte para diferentes calidades visuales en lugar de dejar que WebRTC sea quien considere la calidad de la retransmisión, así como otros parámetros de control durante la retransmisión.
- Añadir elementos extra a la experiencia de usuario: Visitar perfiles ajenos, sistema de chat entre amigos, sistema de logros en cuanto a videojuegos
- En cuanto a interfaz, en el momento en el que salga Vuetify 3, esta podría ser remodelada para añadir un entorno un poco más animado, con componentes un poco más complejos que los que ofrece PrimeVue, aunque esto se trata de algo secundario
- También en cuanto a interfaz, sería ideal crear un nuevo menú principal a modo de página principal de navegación con varios botones que nos permitan viajar a las diferentes vistas, siendo estos animado al poner el cursor encima de ellos, la idea inicial era realizar algo así, pero por cuestiones temporales se desechó la idea de hacer una animación customizada para cada botón, por lo que se optó por una Navbar convencional

8. BIBLIOGRAFÍA

3CX. (2022). *3cx.com*. Obtenido de https://www.3cx.es/voip-sip/que-es-webrtc/

Acibeiro, M. (7 de 2 de 2022). Obtenido de https://www.lucushost.com/blog/que-es-node-js/#Que_es_Nodejs

Adobe. (1996).

akin-ogundenji, M. (29 de 11 de 2015). *medium.com.* Obtenido de https://medium.com/@nohkachi/my-vue-on-mvvm-c3ffb4f0678f#:~:text=MVVM%20stands%20for%20Model%2DView,access%20layer%20of%20the%20application.

Amazon. (2022).

Arnold, R. A. (26 de 12 de 2007). *middlewares.wordpress.com.* Obtenido de https://middlewares.wordpress.com/2007/12/26/fat-servers-or-fat-clients/

Barragán, A. (26 de 11 de 2021). *OpenWebinars.net*. Obtenido de https://openwebinars.net/blog/que-es-vue-js-y-que-lo-diferencia-de-otros-frameworks/

bezkoder. (26 de 4 de 2022). *bezkoder.com*. Obtenido de https://www.bezkoder.com/vue-node-express-postgresql/

Bianzi, F. (2020). *Coolors.co*. Obtenido de https://help.coolors.co/hc/en-us/articles/360010536020-What-is-Coolors-

Brush, K. (2019). *TechTarget.com*. Obtenido de https://www.techtarget.com/searchdatamanagement/definition/RDBMS-relational-database-management-system#:~:text=Some%20examples%20of%20specific%20systems,MySQL%2C%20Microsoft%20SQLServer%20and%20PostgreSQL.

Clippy. (8 de 10 de 2020). *bennetfeely.com*. Obtenido de https://bennettfeely.com/clippy/

Cloudfare. (2022). *cloudfare.com*. Obtenido de https://www.cloudflare.com/es-es/learning/video/what-is-http-live-streaming/

Daniel, S. (2016). *platzi.com*. Obtenido de https://platzi.com/blog/aplicaciones-escritorio-electron-js/

Durán y Bernárdez. (2002).

Eneba. (2018). *eneba.com*. Obtenido de https://www.eneba.com/

FairCom. (29 de 4 de 2022). *docs.faircom.com.* Obtenido de https://docs.faircom.com/doc/knowledgebase/54219.htm

Ferrer, J. (19 de 3 de 2022). *medium.com*. Obtenido de https://uxdesign.cc/why-is-dark-mode-so-captivating-92f2ed4e0dc5

Garain, A. (18 de 7 de 2022). *GeeksForGeeks.com*. Obtenido de https://www.geeksforgeeks.org/introduction-to-electronjs/

García Peñalvo, F. J., & García Holgado, A. (2018). *repositorio.grial.eu.* Obtenido de https://repositorio.grial.eu/bitstream/grial/1150/1/5.%20IngReq.pdf

González Reyes, D., Betty Carvajal, N., Manrique Neira, J., Quijije Lucas, G., & Quijije Toro, K. (2017). *monografias.com*. Obtenido de https://www.monografias.com/docs114/telecomunicaciones-arquitectura-cliente-servidor/telecomunicaciones-arquitectura-cliente-servidor

Google. (2019).

Jenkov, J. (12 de 5 de 2019). *jenkov.com*. Obtenido de https://jenkov.com/tutorials/data-streaming/index.html#data-streaming-comes-in-many-variations

Keck, C. (6 de 11 de 2019). *gizmodo.com*. Obtenido de https://gizmodo.com/why-seemingly-every-company-is-launching-a-gaming-subsc-1835426857

Khadir, S. (19 de 7 de 2022). *stackshare.io*. Obtenido de https://stackshare.io/stackups/bootstrap-vs-bootstrap-vue

Kollegger, E. (9 de 5 de 2018). *medium.com*. Obtenido de https://medium.com/@MinimalGhost/what-is-axios-js-and-why-should-i-care-7eb72b111dc0

Kühne, S. (16 de 7 de 2020). *stehkn*. Obtenido de https://stekhn.github.io/gpu-price-surge-analysis/

MarketingCharts. (28 de 9 de 2021). *Time Spent Playing Video Games Continues to Rise*. Obtenido de MarketingCharts.com: https://www.marketingcharts.com/cross-media-and-traditional/videogames-traditional-and-cross-channel-118663

MDN. (16 de 08 de 2022). *developer.modzilla.com*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Common_questions/What_is_a_web_server

MDN. (4 de 8 de 2022). *developer.modzilla.com*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

MDN. (4 de 8 de 2022). *developer.modzilla.org*. Obtenido de https://developer.mozilla.org/es/docs/Web/HTTP/CORS

Microsoft. (2017).

Microsoft. (9 de 6 de 2022). *docs.microsoft.com*. Obtenido de https://docs.microsoft.com/es-es/dynamics-nav/microsoft-dynamics-nav-web-client-network-architecture

Nguyen, C. (7 de 3 de 2020). *medium.com*. Obtenido de https://medium.com/swlh/webrtc-the-technology-that-powers-google-meet-hangout-facebook-messenger-and-discord-cb926973d786

Nvidia. (2020).

Ortiz, J. (2017). *javierortiz.mx*. Obtenido de https://javierortiz.mx/glosario/protocolos-streaming-video/protocolo-rtmp/

ProgrammerClick. (2020-2022). *programmerclick.com.* Obtenido de https://programmerclick.com/article/27631228428/

ReactiveProgramming. (2022). *reactiveprogramming.io*. Obtenido de https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor

Riot. (2009).

Rouse, M. (10 de 2019). *computerweekly.com*. Obtenido de https://www.computerweekly.com/es/definicion/Sistema-de-gestion-de-bases-de-datos-relacionales-o-RDBMS

Ruether, T. (25 de 4 de 2022). *wowza.com*. Obtenido de https://www.wowza.com/blog/history-of-streaming-media

Shacklett, M. E. (2022). *techtarget.com*. Obtenido de https://www.techtarget.com/searchunifiedcommunications/definition/WebRTC-Web-Real-Time-Communications

Sharma, K. (19 de 1 de 2021). *ubuntu.com*. Obtenido de https://ubuntu.com/blog/what-is-postgresql

Socket.IO. (12 de 6 de 2022). *socket.io*. Obtenido de https://socket.io/docs/v4/

Sony. (2010).

Splunk. (2022). *splunk.com*. Obtenido de https://www.splunk.com/en_us/data-insider/what-are-distributed-systems.html

STechies. (2022). *STechies.com*. Obtenido de https://www.stechies.com/differences-between-dbms-rdbms/

Twitch. (18 de 12 de 2015). *twitch.tv*. Obtenido de https://blog.twitch.tv/en/2015/12/18/twitch-engineering-an-introduction-and-overview-a23917b71a25/

Valve. (2003).

WebRTC. (2022). *webrtc.org*. Obtenido de https://webrtc.org/

Wikipedia. (5 de 8 de 2022). Obtenido de https://en.wikipedia.org/wiki/Mbone

Wikipedia. (6 de 8 de 2022). *Wikipedia.com*. Obtenido de https://en.wikipedia.org/wiki/Muzak

Zubikarai, S. (15 de 3 de 2021). *freecodecamp.org*. Obtenido de https://www.freecodecamp.org/espanol/news/ventajas-y-desventajas-de-javascript/

Zuev, A. (5 de 6 de 2020). *adictosaltrabajo.com.* Obtenido de https://www.adictosaltrabajo.com/2020/06/05/patron-mvvm-en-swiftui/

ANEXO I: Plan del proyecto.

Plataforma streaming multidispositivo de videojuegos

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Agosto de 2022

Autor:

Jesús Manuel García Prieto

Tutores:

Diego J. Valdeolmillos Villaverde

Alfonso González Briones

Francisco Pinto Santos

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El objetivo de este anexo es el establecer una planificación temporal del sistema.

Para lograr este objetivo la planificación del proyecto se regirá bajo el esquema del Proceso Unificado, que nos servirá para identificar, catalogar y planificar los diferentes casos de uso del sistema de una manera incremental e iterativa.

Una vez catalogados y estimada la complejidad de los diferentes casos de uso se procederá a realizar un diagrama de Gantt del sistema para realizar con mayor facilidad la planificación temporal del mismo, siempre siguiendo la estructura y división propias del Proceso Unificado, en concreto se dividirá la planificación en 4 iteraciones con las fases de Modelo de Negocio, Requisitos, Análisis, Diseño, Implementación y Pruebas.

2. ESTIMACIÓN TEMPORAL

Será en esta etapa en la que realicemos la estimación de costes del sistema, para lograr esto usaremos la métrica de los UCP (Use Case Points) y la herramienta EZEstimate que será la encargada de hacer las cuentas necesarias para hallar el esfuerzo total requerido para desarrollar el sistema.

En primer lugar, se listarán aquellos elementos del sistema se deben catalogar para lograr esto:

- **Complejidad de los actores del sistema (UAW)**
- **Complejidad de los Casos de Uso (UUCW)**
- **Factores de complejidad técnica (TCF)**
- **Factores de complejidad del entorno (ECF)**

Se procede ahora a analizar cada uno de estos valores citados.

2.1. COMPLEJIDAD DE LOS ACTORES DEL SISTEMA (UAW)

Para lograr obtener este factor de complejidad se debe listar el número de actores del sistema y su complejidad, para hallar este último valor se seguirá el sistema de pesos mostrado en la **Tabla 6**:

Tabla 6. Tipos de complejidad en actores

Complejidad	Descripción	Peso
Simple	Si el actor es un sistema y la aplicación se comunica con él mediante una API	1
Media	Si el actor es un sistema y la aplicación se comunica con él mediante un protocolo (internet)	2
Compleja	Persona con interfaz gráfica	3

Sabiendo ahora los tipos de complejidad, tan solo queda catalogar cada uno de los actores del sistema tal y como podemos ver en la **Tabla 7**:

Tabla 7. Complejidad de los actores del sistema

Actor	Complejidad	Peso
ACT-0001 Usuario No Registrado	Compleja	1
ACT-0002 Usuario	Compleja	2
ACT-0003 Administrador	Compleja	
ACT-0004 Fuente	Compleja	
ACT-0005 Sistema	Simple	3

2.2. COMPLEJIDAD DE LOS CASOS DE USO(UUCW)

De manera similar al apartado anterior, la estimación del peso de los Casos de Uso viene dada por una plantilla, en este caso se trata del modelo de Karner, que nos permite asignar una complejidad a un Caso de Uso dado su número de transacciones o pasos.

En la **Tabla 8** se puede observar en detalle cómo este método asigna la complejidad a cada tipo de Caso de Uso:

Tabla 8. Tipos de complejidad en Casos de Uso

Complejidad	Descripción
Simple	3 o menos transacciones
Media	De 4 a 7 transacciones
Compleja	Más de 7 transacciones

Cabe destacar que, en este caso, con asignar una complejidad es suficiente, no es necesario asignar un factor de peso, de esto se encarga ya EZEstimate.

Se procede a listar cada uno de los casos de uso del sistema con las diferentes complejidades descritas, así como su número de transacciones a modo de justificación. En la **Tabla 9** se pueden observar los resultados de esta asignación:

Tabla 9. Complejidad de los Casos de Uso del sistema

Caso de Uso	Número de transacciones	Complejidad
UC-0001	6	Media
UC-0002	8	Compleja
UC-0003	6	Media
UC-0004	3	Simple
UC-0005	7	Media
UC-0006	2	Simple
UC-0007	3	Simple
UC-0008	2	Simple
UC-0009	3	Simple
UC-0010	3	Simple
UC-0011	3	Simple
UC-0012	3	Simple
UC-0013	4	Media
UC-0014	4	Media
UC-0015	4	Media
UC-0016	2	Simple
UC-0017	4	Media
UC-0018	2	Simple
UC-0019	3	Simple
UC-0020	3	Simple
UC-0021	1	Simple
UC-0022	3	Simple
UC-0023	2	Simple
UC-0024	2	Simple

2.3. FACTORES DE COMPLEJIDAD TÉCNICA (TCF)

Se tratan de 13 factores relacionados con la complejidad técnica del sistema. Su método de medición difiere ligeramente de los dos apartados anteriores, ya que en este caso el peso de los factores viene dado de manera estática, solo se les debe asignar una complejidad percibida, la cual varía de proyecto a proyecto y oscila entre 0 y 5.

Podemos ver en la **Tabla 10** los 13 tipos de factores de complejidad junto a su peso y su complejidad percibida, además de una pequeña justificación de la misma.

Tabla 10. Complejidad de los Factores de Complejidad Técnica del sistema

Factor de complejidad técnica	Peso	Complejidad percibida	Justificación
Sistema Distribuido	2	4	Posiblemente se trate del factor característico de este proyecto, ya que coexisten varios nodos (cliente-fuente) que se comunican por medio de un servidor
Rendimiento	2	4	De nuevo, otro factor crucial, la retransmisión recibida por el cliente debe llegar con el menor retardo posible para que sus inputs se apliquen con la mayor velocidad posible
Eficiencia del usuario final	1	1	El gran grueso de la aplicación consiste en seleccionar un videojuego y en jugarlo, la aplicación no requiere de mucho esfuerzo parapoder hacer esto
Procesamiento interno complejo	1	3	Únicamente durante el envío de vídeo y de inputs el sistema muestra una gran complejidad, pero fuera de este caso se trata de un sistema simple
Reusabilidad	1	2	El amplio uso de bibliotecas externas limita la facilidad de implementación del sistema en otras plataformas o aplicaciones, sin embargo, ciertos módulos son reutilizables
Facilidad de instalación	0.5	1	El usuario simplemente debe abrir el ejecutable de la aplicación, sin necesidad de hacer nada más posteriormente
Facilidad de uso	0.5	3	Utilizar la aplicación desde el punto de vista del cliente es sencillo, sin embargo, desde el punto de vista del desarrollador se debe mantener el equipo fuente
Portabilidad	2	1	La aplicación está diseñada para funcionar en Windows
Facilidad de cambio	1	3	Debido a la naturaleza modular de Vue es relativamente sencillo aplicar cambios a la aplicación
Concurrencia	1	4	Se debe permitir la coexistencia de varios usuarios e incluso varios equipos fuente
Características especiales de seguridad	1	2	En aquellos aspectos que requieren seguridad (Base de Datos) el servidor aplica ciertos parámetros de seguridad
Acceso directo a terceras partes	1	0	El proyecto no está diseñado para dar acceso a terceras partes
Se requiere entrenamiento especial del usuario	1	0	El usuario no requiere ningún tipo de entrenamiento para utilizar la aplicación

2.4. FACTORES DE ENTORNO (ECF)

De igual manera que ocurrió en el apartado anterior, estos factores se dividen en varios tipos, cada uno de ellos con su propio peso estático predefinido, y, de igual manera también, se les debe atribuir una complejidad percibida. Sin embargo, en este tipo de factores a mayor valor se les asigne, implicarán una menor complejidad en el sistema (ocurrirá al revés para aquellos factores que ponderen negativamente).

Se pueden observar los resultados en la **Tabla 11**:

Tabla 11. Complejidad de los Factores de Entorno del sistema

Factor de entorno	Peso	Complejidad percibida	Justificación
Familiaridad con UML	1.5	4	Debido al alto uso de UML a lo largo del grado, se considera que existe una alta familiaridad con ello
Experiencia con la aplicación	0.5	2	Debido a la falta de experiencia con los lenguajes presentes en el sistema y con los sistemas distribuidos, se considera que existe una baja familiaridad con la aplicación
Experiencia en la orientación a objetos	1	3	Existe cierta familiaridad con la POO gracias al propio grado y a proyectos personales propios
Capacidad de los analistas	0.5	2	Debido al grado existe cierta capacidad de análisis por parte del diseñador
Motivación	1	5	Se trata del proyecto final del desarrollador, por tanto, existen muchas ganas de finalizarlo
Estabilidad de los requisitos	2	4	Siguiendo los diferentes patrones de análisis se han designado unos requisitos lo más estables posibles para el sistema
Trabajadores a tiempo parcial	-1	1	A pesar de existir un único usuario en el equipo de desarrollo, no existieron restricciones temporales debido a realizar el desarrollo en verano
Dificultad del lenguaje de programación	1	2	Ninguno de los lenguajes existentes en el proyecto utilizados por el desarrollador de manera previa, a excepción de durante las prácticas de empresa

2.5. ESTIMACIÓN TEMPORAL (EZEstimate)

El siguiente paso en la estimación temporal será introducir todos estos datos en la herramienta EZEstimate.

En primer lugar, se han introducido los datos relativos a los actores y a los casos de uso, ambos se pueden observar en la **Figura 37:**

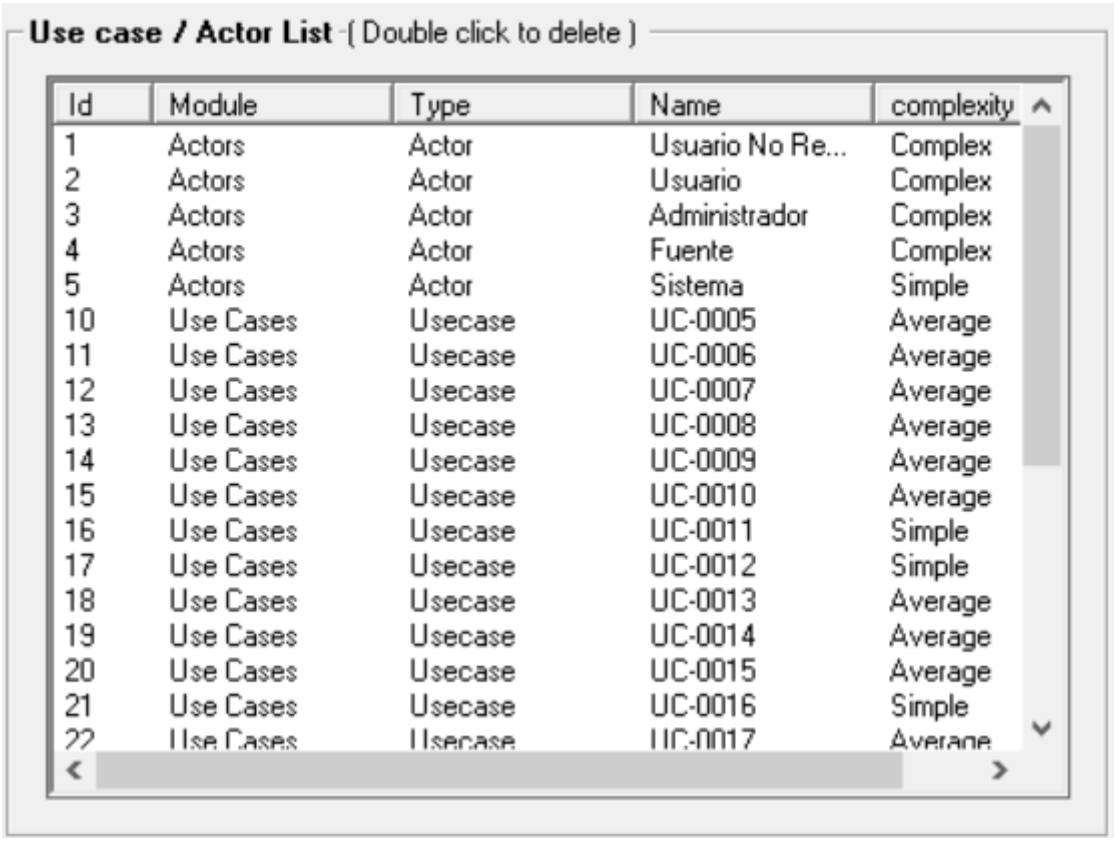


Figura 37. EZEstimate - Actores y Casos de Uso

El siguiente paso consiste en introducir las complejidades percibidas de los factores de entorno y los factores de complejidad técnica. Ambos se pueden observar en las figuras **Figura 38** y **Figura 39**:

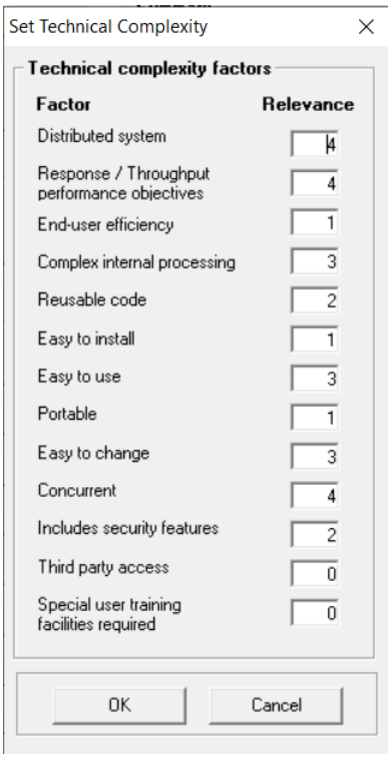


Figura 38. EZEstimate - Factores de complejidad técnica

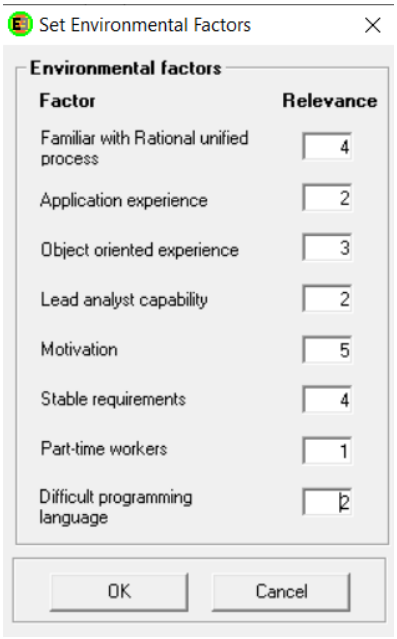


Figura 39. EZEstimate - Factores de entorno

Con todos estos valores y un esfuerzo total estimado de 8, los resultados obtenidos son los presentados en la **Figura 40**:

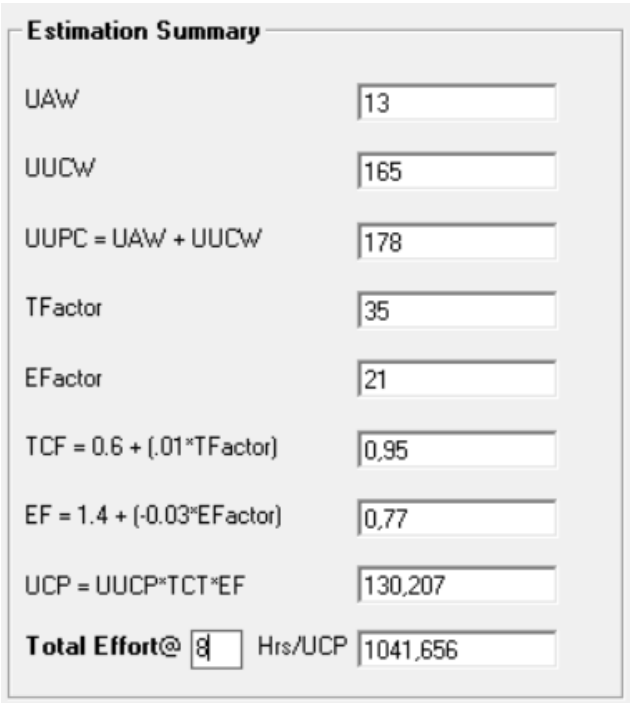


Figura 40. Resultados de la estimación temporal

Como se puede observar, el resultado final es de 1041h de persona que, al tratarse solo de un desarrollador en este proyecto, se reparten exclusivamente sobre él. Estas horas se corresponden con 130 días trabajando 8h al día y 5 días a la semana.

3. PLANIFICACIÓN TEMPORAL

Habiéndose estimado el número de días para completar el desarrollo del proyecto, se procede a realizar la planificación temporal de este, basándonos en este número.

Cabe destacar que, tal y como se ha estipulado en el apartado anterior, se trabajará 8h al día 5 días a la semana, estas restricciones se pueden observar en la **Figura 41**, que nos muestra, a su vez, el calendario de la planificación temporal del proyecto.

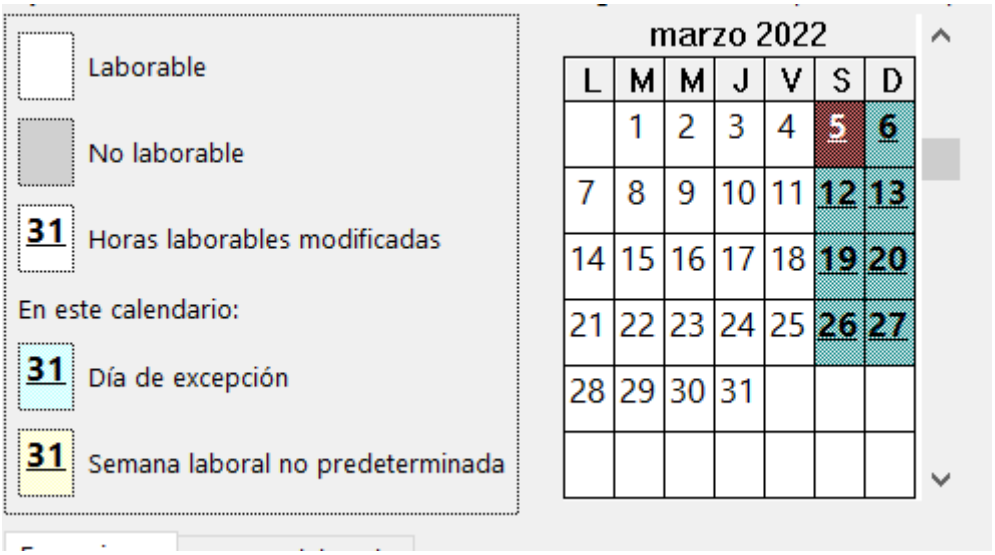


Figura 41. Detalle del calendario de la planificación temporal del proyecto

Como se puede observar, los fines de semana se encuentran remarcados con un color azul, la leyenda nos muestra que esto se debe a que son días de excepción, mientras que, el resto de días de la semana se encuentran remarcados de blanco, es decir, son días laborales.

Con todas las restricciones correctamente indicadas en el sistema, se procede a detallar las diferentes etapas de la planificación temporal.

3.1. DESCRIPCIÓN DE TAREAS

Como ya se explicó con anterioridad, la división de la planificación del sistema se basa plenamente en la disposición original del Proceso Unificado (Modelado de negocio, Requisitos, Análisis, Diseño, Implementación y Pruebas), sabiendo eso, solo falta por aclarar que se ha decidido realizar un total de 4 iteraciones, a diferencia del Proceso Unificado, que propone 8.

En la **Tabla 12** se puede observar un desglose de estas tareas pertenecientes a cada iteración, así como una descripción sobre cada una:

Tabla 12. Descripción de tareas

Nombre			Descripción
0	**Inicio del Proyecto**		Marca temporal que sirve como referencia para identificar el comienzo del proyecto
1	**Iteración 1**		Se realiza una investigación inicial sobre los elementos pertinentes al sistema, así como una definición inicial de las etapas de análisis y diseño
	1.1	**Modelado de Negocio**	
		1.1.1 **Investigar proyectos similares existentes**	Conociendo los elementos de nuestrosistema, se busca investigar aquellos que sean similares o compatibles a sus objetivos, centrándose en aquellos con mayor rendimiento
		1.1.2 **Investigar sobre diferente software para el desarrollo del sistema**	Investigar sobre aquellas técnicas y tecnologías más apropiadas para llevar a cabo la realización de nuestro sistema
		1.1.3 **Investigar diferentes estilos artísticos y visuales para el desarrollo del sistema**	Investigar diferentes sistemas similares en busca de elementos visuales referenciales
	1.2	**Requisitos**	
		1.2.1 **Definición del modelo inicial de requisitos**	Basándose en los conocimientos adquiridos en la etapa anterior, realizar una primera definición de requisitos del sistema, teniendo en cuenta también los objetivos del mismo
	1.3	**Análisis**	
		1.3.1 **Definición inicial de CU Subsistema "Gestión de Usuarios"**	Realizar un primer análisis del subsistema, este será el encargado de gestionar las acciones pertinentes al usuario (modificar datos, login, logout…)
		1.3.2 **Definición inicial de CU Subsistema "Gestión de Catálogo"**	Realizar un primer análisis del subsistema, este será el encargado de gestionar el catálogo y sus elementos (filtrado, búsqueda…)
		1.3.3 **Definición inicial de CU Subsistema "Gestión de Actividades Sociales"**	Realizar un primer análisis del subsistema, este será el encargado de gestionar las acciones sociales (lista de amigos, bloquear usuarios…)
		1.3.4 **Definición inicial de CU Subsistema "Gestión de Retransmisión"**	Realizar un primer análisis del subsistema, este será el encargado de gestionar la retransmisión y los elementos relacionados con esta (enviar vídeo, gestión de inputs…)
	1.4	**Diseño**	

		1.4.1	**Diseño tentativo del Subsistema "Gestión de Usuarios"**	Realizar un primer diseño del subsistema en cuestión, debido a que se trata de una primera iteración del diseño, este irá siendo definido con mayor exactitud en próximas iteraciones
		1.4.2	**Diseño tentativo del Subsistema "Gestión de Catálogo"**	Realizar un primer diseño del subsistema en cuestión, debido a que se trata de una primera iteración del diseño, este irá siendo definido con mayor exactitud en próximas iteraciones
		1.4.3	**Diseño tentativo del Subsistema "Gestión de Actividades Sociales"**	Realizar un primer diseño del subsistema en cuestión, debido a que se trata de una primera iteración del diseño, este irá siendo definido con mayor exactitud en próximas iteraciones
		1.4.4	**Diseño tentativo del Subsistema "Gestión de Retransmisión"**	Realizar un primer diseño del subsistema en cuestión, debido a que se trata de una primera iteración del diseño, este irá siendo definido con mayor exactitud en próximas iteraciones
		1.4.5	**Diseño de la arquitectura inicial**	Realizar un primer diseño de la arquitectura inicial, debido a que se trata de una primera iteración del diseño, esta irá siendo definido con mayor exactitud en próximas iteraciones
	1.5	**Implementación**		
		1.5.1	**Preparación del entorno de desarrollo**	Preparar un primer entorno de desarrollo en el que desplegar el sistema. Ya que se trata de la primera iteración, crear el repositorio pertinente
		1.5.2	**Despliegue del sistema de envío y recepción de vídeo**	Desplegar la plataforma inicial de envío de vídeo entre dos sistemas
		1.5.3	**Implementación de interfaz inicial para el envío y recepción de vídeo**	Desarrollar una interfaz inicial para permitir el envío de vídeo entre dos sistemas
	1.6	**Pruebas**		
		1.6.1	**Pruebas iniciales sobre el sistema de envío y recepción de vídeo**	Realizar las pruebas iniciales sobre el entorno de envío de vídeo entre dos sistemas
2	**Hito 1**			
3	**Iteración 2**			
	3.1	**Modelo de Negocio**		
		3.1.1	**Investigar sobre técnicas de retransmisión de vídeo**	Investigar sobre las diferentes técnicas de envío, transmisión y recepción de vídeo, así como las ventajas y desventajas de las mismas
		3.1.2	**Investigar sobre técnicas de captura y aplicación de inputs**	Investigar sobre las diferentes técnicas de captura y aplicación de inputs, así como las ventajas y desventajas de las mismas
	3.2	**Requisitos**		

	3.2.1	**Refinamiento de requisitos del Subsistema "Gestión de Actividades Sociales"**	Refinamiento de los requisitos iniciales del subsistema “Gestión de Actividades Sociales” en función de las conclusiones obtenidas en la etapa de modelo de negocio
	3.2.2	**Refinamiento de requisitos del Subsistema "Gestión de Retransmisión"**	Refinamiento de los requisitos iniciales del subsistema “Gestión de Retransmisión” en función de las conclusiones obtenidas en la etapa de modelo de negocio
3.3	**Análisis**		
	3.3.1	**Realización de matrices de rastreabilidad**	Realización de las matrices de rastreabilidad del proyecto, en función de los requisitos definidos anteriormente
	3.3.2	**Refinamiento de análisis del Subsistema "Gestión de Actividades Sociales"**	Refinamiento del análisis inicial del subsistema “Gestión de Actividades Sociales” en función de las conclusiones obtenidas en la etapa de modelo de negocio
	3.3.3	**Refinamiento de análisis del Subsistema "Gestión de Retransmisión"**	Refinamiento del análisis inicial del subsistema “Gestión de Retransmisión” en función de las conclusiones obtenidas en la etapa de modelo de negocio
3.4	**Diseño**		
	3.4.1	**Refinamiento de diseño del Subsistema "Gestión de Actividades Sociales"**	Refinamiento del diseño inicial del subsistema “Gestión de Actividades Sociales” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del análisis
	3.4.2	**Refinamiento de diseño del Subsistema "Gestión de Retransmisión"**	Refinamiento del diseño inicial del subsistema “Gestión de Retransmisión” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del análisis
3.5	**Implementación**		
	3.5.1	**Implementación del Subsistema "Gestión de Actividades Sociales"**	Implementación del subsistema “Gestión de Actividades Sociales” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del diseño
	3.5.2	**Implementación del Subsistema "Gestión de Retransmisión"**	Implementación del subsistema “Gestión de Retransmisión” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del diseño
	3.5.3	**Integración y Despliegue de los subsistemas implementados**	Integración y despliegue de los subsistemas implementados previamente
3.6	**Pruebas**		
	3.6.1	**Pruebas del Subsistema "Gestión de Actividades Sociales"**	Realización de pruebas sobre el despliegue del subsistema “Gestión de Actividades Sociales”
	3.6.2	**Pruebas del Subsistema "Gestión de Retransmisión"**	Realización de pruebas sobre el despliegue del subsistema “Gestión de Retransmisión”
	3.6.3	**Pruebas de Integración**	Realización de pruebas para asegurar la correcta integración de los subsistemas

4	**Hito 2**		
5	**Iteración 3**		
	5.1	**Modelo de Negocio**	
		5.1.1	**Investigación sobre integración de servidores web**	Investigar sobre las diferentes tecnologías y arquitecturas sobre las que integrar un servidor web, así como las ventajas y desventajas de las mismas
		5.1.2	**Investigación sobre la gestión e implementación de catálogos en plataformas de videojuegos**	Investigar sobre las diferentes técnicas de guardado y exposición de elementos en catálogos de videojuegos, así como ventajas y desventajas de las mismas
	5.2	**Requisitos**	
		5.2.1	**Refinamiento de requisitos del Subsistema "Gestión de Usuarios"**	Refinamiento de los requisitos iniciales del subsistema “Gestión de Usuarios” en función de las conclusiones obtenidas en la etapa de modelo de negocio
		5.2.2	**Refinamiento de requisitos del Subsistema "Gestión de Catálogo"**	Refinamiento de los requisitos iniciales del subsistema “Gestión de Catálogo” en función de las conclusiones obtenidas en la etapa de modelo de negocio
	5.3	**Análisis**	
		5.3.1	**Refinamiento de análisis del Subsistema "Gestión de Usuarios"**	Refinamiento del análisis inicial del subsistema “Gestión de Usuarios” en función de las conclusiones obtenidas en la etapa de modelo de negocio
		5.3.2	**Refinamiento de análisis del Subsistema "Gestión de Catálogo"**	Refinamiento del análisis inicial del subsistema “Gestión de Catálogo” en función de las conclusiones obtenidas en la etapa de modelo de negocio
	5.4	**Diseño**	
		5.4.1	**Refinamiento de diseño del Subsistema "Gestión de Usuarios"**	Refinamiento del diseño inicial del subsistema “Gestión de Usuarios” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del análisis
		5.4.2	**Refinamiento de diseño del Subsistema "Gestión de Catálogo"**	Refinamiento del diseño inicial del subsistema “Gestión de Catálogo” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del análisis
	5.5	**Implementación**	
		5.5.1	**Implementación del Subsistema "Gestión de Usuarios"**	Implementación del subsistema “Gestión de Usuarios” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del diseño
		5.5.2	**Implementación del Subsistema "Gestión de Catálogo"**	Implementación del subsistema “Gestión de Catálogo” en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del diseño
		5.5.3	**Integración y Despliegue de los subsistemas implementados**	Integración y despliegue de los subsistemas implementados previamente

	5.6	**Pruebas**		
		5.6.1	**Pruebas del Subsistema "Gestión de Usuarios"**	Realización de pruebas sobre el despliegue del subsistema “Gestión de Usuarios”
		5.6.2	**Pruebas del Subsistema "Gestión de Catálogo"**	Realización de pruebas sobre el despliegue del subsistema “Gestión de Catálogo”
		5.6.3	**Pruebas de Integración**	Realización de pruebas para asegurar la correcta integración de los subsistemas
6	**Hito 3**			
7	**Iteración 4**			
	7.1	**Modelo de Negocio**		
		7.1.1	**Revisión de posibles riesgos existentes en el sistema**	Realizar una revisión final de los posibles riesgos existentes en el sistema
		7.1.2	**Revisión final de sistemas similares**	Realizar una última búsqueda de sistemas compatibles con el desarrollado, en busca de posibles adiciones finales o de manera auxiliar a la búsqueda de riesgos
	7.2	**Requisitos**		
		7.2.1	**Refinamiento final de requisitos en función de los riesgos existentes**	Refinamiento final de los requisitos del sistema, modificándolos en función de las conclusiones obtenidas en la etapa de modelo de negocio
	7.3	**Análisis**		
		7.3.1	**Revisión final del modelo de análisis**	Refinamiento final del análisis del sistema, en función de las conclusiones obtenidas en la etapa de modelo de negocio
	7.4	**Diseño**		
		7.4.1	**Revisión final del modelo de diseño**	Refinamiento final del diseño del sistema, en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del análisis
	7.5	**Implementación**		
		7.5.1	**Integración y despliegue de los cambios realizados al sistema final**	Cambios finales en la integración y despliegue del sistema, en función de las conclusiones obtenidas en la etapa de modelo de negocio y del refinamiento del diseño
	7.6	**Pruebas**		
		7.6.1	**Pruebas de los cambios realizados al sistema final**	Realización de pruebas finales sobre el despliegue y la integración del sistema
8	**Hito 4**			

3.2. DIVISIÓN TEMPORAL Y DE RECURSOS

En primer lugar, la división de recursos para este proyecto en concreto, al tratarse de un proyecto realizado por una sola persona, será lineal, es decir, las tareas serán realizadas en el orden designado una tras otra, sin concurrencia entre ellas.

Por otro lado, en cuanto a la división temporal, partiendo de los 130 días estimados, se ha decidido dividir las tareas en 4 iteraciones, como ya se mencionó anteriormente. En la **Tabla 13** se puede observar la división establecida.

Tabla 13. Asignación temporal por iteración

Iteración	Días Asignados
1	45
2	20
3	24
4	41

Como la tabla indica, la primera iteración abarca un gran grueso del proyecto, esto se debe a que el desarrollador no posee una gran cantidad de conocimientos en el área a desarrollar, es por esto que la primera etapa de búsqueda de datos y tecnologías sería una de las más complejas. Ocurre de manera similar con la última iteración, debido a que es en esta donde hay una mayor carga de trabajo asociada a la integración y despliegue y, de nuevo, debido a la baja experiencia del desarrollador en esta rama, esta iteración sería también de las más complejas.

Por último, a modo de desglose aclarativo, se muestra en la **Tabla 14** un detalle de la asignación de días por tarea, así como la asignación de recursos (que, como ya se ha justificado, será del 100% en cada tarea al haber un solo desarrollador).

Tabla 14. Asignación temporal por tarea

ID	Nombre	Asignación temporal	Asignación de recursos
1.1.1	Investigar proyectos similares existentes	6 días	100%
1.1.2	Investigar sobre diferente software para el desarrollo del sistema	5 días	100%
1.1.3	Investigar diferentes estilos artísticos y visuales para el desarrollo del sistema	2 días	100%
1.2.1	Definición del modelo inicial de requisitos	6 días	100%
1.3.1	Definición inicial de CU Subsistema "Gestión de Usuarios"	2 días	100%
1.3.2	Definición inicial de CU Subsistema "Gestión de Catálogo"	2 días	100%
1.3.3	Definición inicial de CU Subsistema "Gestión de Actividades Sociales"	2 días	100%
1.3.4	Definición inicial de CU Subsistema "Gestión de Retransmisión"	2 días	100%
1.4.1	Diseño tentativo del Subsistema "Gestión de Usuarios"	2 días	100%
1.4.2	Diseño tentativo del Subsistema "Gestión de Catálogo"	1 día	100%
1.4.3	Diseño tentativo del Subsistema "Gestión de	2 días	100%

	Actividades Sociales"		
1.4.4	Diseño tentativo del Subsistema "Gestión de Retransmisión"	2 días	100%
1.4.5	Diseño de la arquitectura inicial	1 día	100%
1.5.1	Preparación del entorno de desarrollo	1 día	100%
1.5.2	Despliegue del sistema de envío y recepción de vídeo	4 días	100%
1.5.3	Implementación de interfaz inicial para el envío y recepción de vídeo	4 días	100%
1.6.1	Pruebas iniciales sobre el sistema de envío y recepción de vídeo	1 día	100%
3.1.1	Investigar sobre técnicas de retransmisión de vídeo	3 días	100%
3.1.2	Investigar sobre técnicas de captura y aplicación de inputs	2 días	100%
3.2.1	Refinamiento de requisitos del Subsistema "Gestión de Actividades Sociales"	0,5 días	100%
3.2.2	Refinamiento de requisitos del Subsistema "Gestión de Retransmisión"	0,5 días	100%
3.3.1	Realización de matrices de rastreabilidad	0,5 días	100%
3.3.2	Refinamiento de análisis del Subsistema "Gestión de Actividades Sociales"	0,5 días	100%
3.3.3	Refinamiento de análisis del Subsistema "Gestión de Retransmisión"	1 día	100%
3.4.1	Refinamiento de diseño del Subsistema "Gestión de Actividades Sociales"	1 día	100%
3.4.2	Refinamiento de diseño del Subsistema "Gestión de Retransmisión"	1 día	100%
3.5.1	Implementación del Subsistema "Gestión de Actividades Sociales"	3 días	100%
3.5.2	Implementación del Subsistema "Gestión de Retransmisión"	4 días	100%
3.5.3	Integración y Despliegue de los subsistemas implementados	1 día	100%
3.6.1	Pruebas del Subsistema "Gestión de Actividades Sociales"	0,5 días	100%
3.6.2	Pruebas del Subsistema "Gestión de Retransmisión"	0,5 días	100%
3.6.3	Pruebas de Integración	1 día	100%
5.1.1	Investigación sobre integración de servidores web	5 días	100%
5.1.2	Investigación sobre la gestión e implementación de catálogos en plataformas de videojuegos	2 días	100%
5.2.1	Refinamiento de requisitos del Subsistema "Gestión de Usuarios"	0.25 días	100%
5.2.2	Refinamiento de requisitos del Subsistema "Gestión de Catálogo"	0.25 días	100%
5.3.1	Refinamiento de análisis del Subsistema "Gestión de Usuarios"	0,5 días	100%
5.3.2	Refinamiento de análisis del Subsistema "Gestión de Catálogo"	0,5 días	100%
5.4.1	Refinamiento de diseño del Subsistema "Gestión de Usuarios"	1 día	100%
5.4.2	Refinamiento de diseño del Subsistema "Gestión de Catálogo"	1 día	100%

5.5.1	Implementación del Subsistema "Gestión de Usuarios"	6 días	100%
5.2.2	Implementación del Subsistema "Gestión de Catálogo"	3 días	100%
5.2.3	Integración y Despliegue de los subsistemas implementados	2,5 días	100%
5.6.1	Pruebas del Subsistema "Gestión de Usuarios"	0,5 días	100%
5.6.2	Pruebas del Subsistema "Gestión de Catálogo"	0,5 días	100%
5.6.3	Pruebas de Integración	1 día	100%
7.1.1	Revisión de posibles riesgos existentes en el sistema	3 días	100%
7.1.2	Revisión final de sistemas similares	4 días	100%
7.2.1	Refinamiento final de requisitos en función de los riesgos existentes	3 días	100%
7.3.1	Revisión final del modelo de análisis	4 días	100%
7.4.1	Revisión final del modelo de diseño	7 días	100%
7.5.1	Integración y despliegue de los cambios realizados al sistema final	16 días	100%
7.6.1	Pruebas de los cambios realizados al sistema final	4 días	100%

3.3. RELACIONES ENTRE TAREAS

Como se ha aclarado previamente en este anexo, la planificación temporal diseñada para el proyecto sigue un esquema lineal, es decir, una tarea se realiza sí y solo sí la anterior ha sido finalizada con éxito, es por esto que en la **Tabla 15** podemos encontrar un sistema de prelaciones muy básico, en el cual la tarea anterior precede a la siguiente y así sucesivamente, a diferencia de la primera tarea, que será la única sin relaciones de precedencia.

Tabla 15. Prelaciones

Tarea		**Prelación**	
1.1.1	Investigar proyectos similares existentes	-	-
1.1.2	Investigar sobre diferente software para el desarrollo del sistema	1.1.1	Investigar proyectos similares existentes
1.1.3	Investigar diferentes estilos artísticos y visuales para el desarrollo del sistema	1.1.2	Investigar sobre diferente software para el desarrollo del sistema
1.2.1	Definición del modelo inicial de requisitos	1.1.3	Investigar diferentes estilos artísticos y visuales para el desarrollo del sistema
1.3.1	Definición inicial de CU Subsistema "Gestión de Usuarios"	1.2.1	Definición del modelo inicial de requisitos
1.3.2	Definición inicial de CU Subsistema "Gestión de Catálogo"	1.3.1	Definición inicial de CU Subsistema "Gestión de Usuarios"
1.3.3	Definición inicial de CU Subsistema "Gestión de Actividades Sociales"	1.3.2	Definición inicial de CU Subsistema "Gestión de Catálogo"
1.3.4	Definición inicial de CU Subsistema "Gestión de Retransmisión"	1.3.3	Definición inicial de CU Subsistema "Gestión de Actividades Sociales"
1.4.1	Diseño tentativo del Subsistema "Gestión de Usuarios"	1.3.4	Definición inicial de CU Subsistema "Gestión de Retransmisión"
1.4.2	Diseño tentativo del Subsistema "Gestión de Catálogo"	1.4.1	Diseño tentativo del Subsistema "Gestión de Usuarios"
1.4.3	Diseño tentativo del Subsistema "Gestión de Actividades Sociales"	1.4.2	Diseño tentativo del Subsistema "Gestión de Catálogo"
1.4.4	Diseño tentativo del Subsistema "Gestión de Retransmisión"	1.4.3	Diseño tentativo del Subsistema "Gestión de Actividades Sociales"
1.4.5	Diseño de la arquitectura inicial	1.4.4	Diseño tentativo del Subsistema "Gestión de Retransmisión"
1.5.1	Preparación del entorno de desarrollo	1.4.5	Diseño de la arquitectura inicial
1.5.2	Despliegue del sistema de envío y recepción de vídeo	1.5.1	Preparación del entorno de desarrollo
1.5.3	Implementación de interfaz inicial para el envío y recepción de vídeo	1.5.2	Despliegue del sistema de envío y recepción de vídeo
1.6.1	Pruebas iniciales sobre el sistema de envío y recepción de vídeo	1.5.3	Implementación de interfaz inicial para el envío y recepción de vídeo
3.1.1	Investigar sobre técnicas de retransmisión de vídeo	1.6.1	Pruebas iniciales sobre el sistema de envío y recepción de vídeo
3.1.2	Investigar sobre técnicas de captura y aplicación de inputs	3.1.1	Investigar sobre técnicas de retransmisión de vídeo
3.2.1	Refinamiento de requisitos del Subsistema "Gestión de Actividades Sociales"	3.1.2	Investigar sobre técnicas de captura y aplicación de inputs
3.2.2	Refinamiento de requisitos del Subsistema "Gestión de Retransmisión"	3.2.1	Refinamiento de requisitos del Subsistema "Gestión de Actividades Sociales"
3.3.1	Realización de matrices de rastreabilidad	3.2.2	Refinamiento de requisitos del Subsistema "Gestión de Retransmisión"

3.3.2	Refinamiento de análisis del Subsistema "Gestión de Actividades Sociales"	3.3.1	Realización de matrices de rastreabilidad
3.3.3	Refinamiento de análisis del Subsistema "Gestión de Retransmisión"	3.3.2	Refinamiento de análisis del Subsistema "Gestión de Actividades Sociales"
3.4.1	Refinamiento de diseño del Subsistema "Gestión de Actividades Sociales"	3.3.3	Refinamiento de análisis del Subsistema "Gestión de Retransmisión"
3.4.2	Refinamiento de diseño del Subsistema "Gestión de Retransmisión"	3.4.1	Refinamiento de diseño del Subsistema "Gestión de Actividades Sociales"
3.5.1	Implementación del Subsistema "Gestión de Actividades Sociales"	3.4.2	Refinamiento de diseño del Subsistema "Gestión de Retransmisión"
3.5.2	Implementación del Subsistema "Gestión de Retransmisión"	3.5.1	Implementación del Subsistema "Gestión de Actividades Sociales"
3.5.3	Integración y Despliegue de los subsistemas implementados	3.5.2	Implementación del Subsistema "Gestión de Retransmisión"
3.6.1	Pruebas del Subsistema "Gestión de Actividades Sociales"	3.5.3	Integración y Despliegue de los subsistemas implementados
3.6.2	Pruebas del Subsistema "Gestión de Retransmisión"	3.6.1	Pruebas del Subsistema "Gestión de Actividades Sociales"
3.6.3	Pruebas de Integración	3.6.2	Pruebas del Subsistema "Gestión de Retransmisión"
5.1.1	Investigación sobre integración de servidores web	3.6.3	Pruebas de Integración
5.1.2	Investigación sobre la gestión e implementación de catálogos en plataformas de videojuegos	5.1.1	Investigación sobre integración de servidores web
5.2.1	Refinamiento de requisitos del Subsistema "Gestión de Usuarios"	5.1.2	Investigación sobre la gestión e implementación de catálogos en plataformas de videojuegos
5.2.2	Refinamiento de requisitos del Subsistema "Gestión de Catálogo"	5.2.1	Refinamiento de requisitos del Subsistema "Gestión de Usuarios"
5.3.1	Refinamiento de análisis del Subsistema "Gestión de Usuarios"	5.2.2	Refinamiento de requisitos del Subsistema "Gestión de Catálogo"
5.3.2	Refinamiento de análisis del Subsistema "Gestión de Catálogo"	5.3.1	Refinamiento de análisis del Subsistema "Gestión de Usuarios"
5.4.1	Refinamiento de diseño del Subsistema "Gestión de Usuarios"	5.3.2	Refinamiento de análisis del Subsistema "Gestión de Catálogo"
5.4.2	Refinamiento de diseño del Subsistema "Gestión de Catálogo"	5.4.1	Refinamiento de diseño del Subsistema "Gestión de Usuarios"
5.5.1	Implementación del Subsistema "Gestión de Usuarios"	5.4.2	Refinamiento de diseño del Subsistema "Gestión de Catálogo"
5.2.2	Implementación del Subsistema "Gestión de Catálogo"	5.5.1	Implementación del Subsistema "Gestión de Usuarios"
5.2.3	Integración y Despliegue de los subsistemas implementados	5.2.2	Implementación del Subsistema "Gestión de Catálogo"
5.6.1	Pruebas del Subsistema "Gestión de Usuarios"	5.2.3	Integración y Despliegue de los subsistemas implementados
5.6.2	Pruebas del Subsistema "Gestión de Catálogo"	5.6.1	Pruebas del Subsistema "Gestión de Usuarios"
5.6.3	Pruebas de Integración	5.6.2	Pruebas del Subsistema "Gestión de Catálogo"
7.1.1	Revisión de posibles riesgos existentes en el sistema	5.6.3	Pruebas de Integración
7.1.2	Revisión final de sistemas similares	7.1.1	Revisión de posibles riesgos existentes en el sistema
7.2.1	Refinamiento final de requisitos en función de los riesgos existentes	7.1.2	Revisión final de sistemas similares

7.3.1	Revisión final del modelo de análisis	7.2.1	Refinamiento final de requisitos en función de los riesgos existentes
7.4.1	Revisión final del modelo de diseño	7.3.1	Revisión final del modelo de análisis
7.5.1	Integración y despliegue de los cambios realizados al sistema final	7.4.1	Revisión final del modelo de diseño
7.6.1	Pruebas de los cambios realizados al sistema final	7.5.1	Integración y despliegue de los cambios realizados al sistema final

3.4. DIAGRAMA DE GANTT

Todo lo expuesto a lo largo de este apartado se traduce en el diagrama de Gantt que se puede observar en las siguientes figuras. Este ha sido realizado mediante la herramienta Microsoft Project y nos muestra la asignación temporal, de recursos y relaciones entre tareas de nuestro proyecto.

Debido al tamaño de este diagrama se ha decidido dividir en varias figuras, véanse **Figura 14**, **Figura 15**, **Figura 16**, **Figura 17**, **Figura 18**, **Figura 19** y **Figura 20:**

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Proyecto	**130 días**	**mar 01/03/22**	**lun 29/08/22**		
Iteración 1	**45 días**	**mar 01/03/22**	**lun 02/05/22**		
Modelo del negocio	**13 días**	**mar 01/03/22**	**jue 17/03/22**		
Investigar proyectos similares existentes	6 días	mar 01/03/22	mar 08/03/22		Jesús Manuel García Prieto
Investigar sobre diferente software para el desarrollo del sistema	5 días	mié 09/03/22	mar 15/03/22	4	Jesús Manuel García Prieto
Investigar diferentes estilos artísticos y visuales para el desarrollo del sistema	2 días	mié 16/03/22	jue 17/03/22	5	Jesús Manuel García Prieto
Requisitos	**6 días**	**vie 18/03/22**	**vie 25/03/22**		
Definición del modelo inicial de requisitos	6 días	vie 18/03/22	vie 25/03/22	6	Jesús Manuel García Prieto
Análisis	**8 días**	**lun 28/03/22**	**mié 06/04/22**		
Definición inicial de CU Subsistema "Gestión de Usuarios"	2 días	lun 28/03/22	mar 29/03/22	8	Jesús Manuel García Prieto
Definición inicial de CU Subsistema "Gestión de Catálogo"	2 días	mié 30/03/22	jue 31/03/22	10	Jesús Manuel García Prieto
Definición inicial de CU Subsistema "Gestión de Actividades Sociales"	2 días	vie 01/04/22	lun 04/04/22	11	Jesús Manuel García Prieto
Definición inicial de CU Subsistema "Gestión de Retransmisión"	2 días	mar 05/04/22	mié 06/04/22	12	Jesús Manuel García Prieto

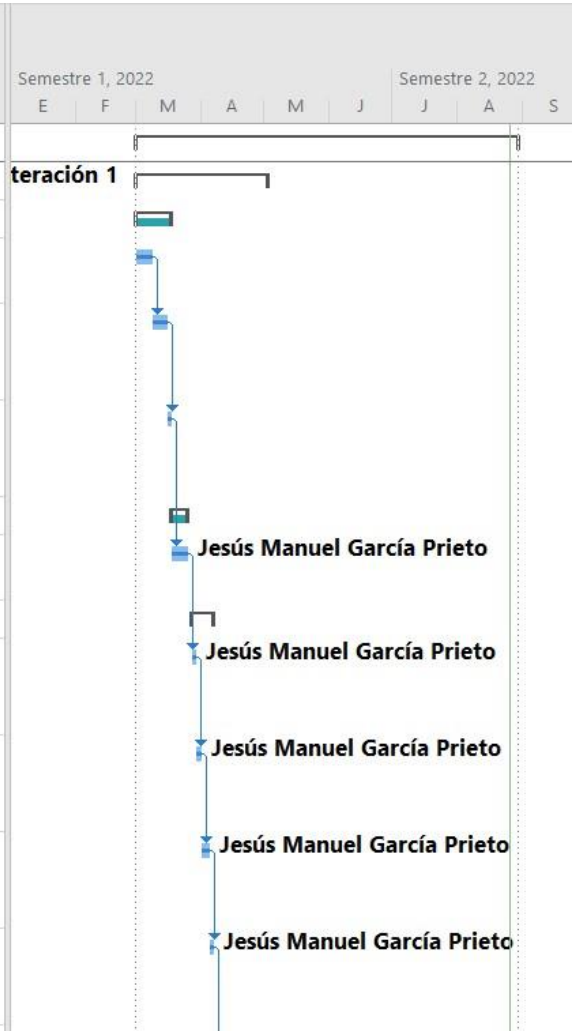


Figura 42. Diagrama de Gantt y Planificación temporal (I)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Diseño	**8 días**	**jue 07/04/22**	**lun 18/04/22**		
Diseño tentativo del Subsistema "Gestión de Usuarios"	2 días	jue 07/04/22	vie 08/04/22	13	Jesús Manuel García Prieto
Diseño tentativo del Subsistema "Gestión de Catálogo"	1 día	lun 11/04/22	lun 11/04/22	15	Jesús Manuel García Prieto
Diseño tentativo del Subsistema "Gestión de Actividades Sociales"	2 días	mar 12/04/22	mié 13/04/22	16	Jesús Manuel García Prieto
Diseño tentativo del Subsistema "Gestión de Retransmisión"	2 días	jue 14/04/22	vie 15/04/22	17	Jesús Manuel García Prieto
Diseño de la arquitectura inicial	1 día	lun 18/04/22	lun 18/04/22	18	Jesús Manuel García Prieto
Implementación	**9 días**	**mar 19/04/22**	**vie 29/04/22**		
Preparación del entorno de desarrollo	1 día	mar 19/04/22	mar 19/04/22	19	Jesús Manuel García Prieto
Despliegue del sistema de envío y recepción de vídeo	4 días	mié 20/04/22	lun 25/04/22	21	Jesús Manuel García Prieto
Implementación de interfaz inicial para el envío y recepción de vídeo	4 días	mar 26/04/22	vie 29/04/22	22	Jesús Manuel García Prieto
Pruebas	**1 día**	**lun 02/05/22**	**lun 02/05/22**		
Pruebas iniciales sobre el sistema de envío y recepción de vídeo	1 día	lun 02/05/22	lun 02/05/22	23	Jesús Manuel García Prieto
Hito 1	0 días	lun 02/05/22	lun 02/05/22	25	Jesús Manuel García Prieto

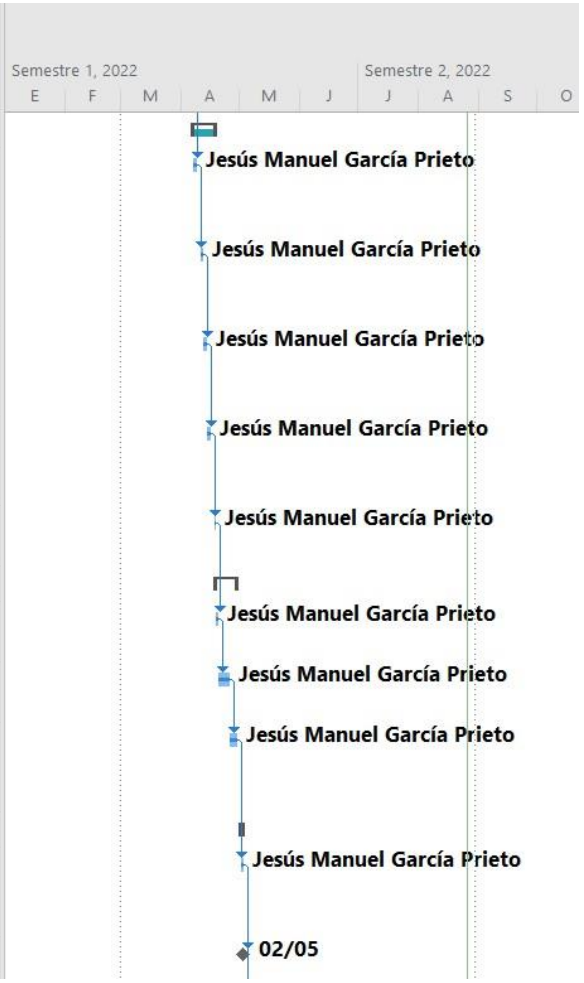


Figura 43. Diagrama de Gantt y Planificación temporal (II)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Iteración 2	**20 días**	**mar 03/05/22**	**lun 30/05/22**		
Modelo del negocio	**5 días**	**mar 03/05/22**	**lun 09/05/22**		
Investigar sobre técnicas de	3 días	mar 03/05/22	jue 05/05/22	25	Jesús Manuel García Prieto
Investigar sobre técnicas de captura y aplicación de	2 días	vie 06/05/22	lun 09/05/22	29	Jesús Manuel García Prieto
Requisitos	**1 día**	**mar 10/05/22**	**mar 10/05/22**		
Refinamiento de requisitos del Subsistema "Gestión de Actividades Sociales"	0,5 días	mar 10/05/22	mar 10/05/22	30	Jesús Manuel García Prieto
Refinamiento de requisitos del Subsistema "Gestión de Retransmisión"	0,5 días	mar 10/05/22	mar 10/05/22	32	Jesús Manuel García Prieto
Análisis	**2 días**	**mié 11/05/22**	**jue 12/05/22**		
Realización de matrices de rastreabilidad	0,5 días	mié 11/05/22	mié 11/05/22	33	Jesús Manuel García Prieto
Refinamiento de análisis del Subsistema "Gestión de Actividades Sociales"	0,5 días	mié 11/05/22	mié 11/05/22	35	Jesús Manuel García Prieto
Refinamiento de análisis del Subsistema "Gestión de Retransmisión"	1 día	jue 12/05/22	jue 12/05/22	36	Jesús Manuel García Prieto
Diseño	**2 días**	**vie 13/05/22**	**lun 16/05/22**		
Refinamiento de diseño del Subsistema "Gestión de Actividades Sociales"	1 día	vie 13/05/22	vie 13/05/22	37	Jesús Manuel García Prieto
Refinamiento de diseño del Subsistema "Gestión de Retransmisión"	1 día	lun 16/05/22	lun 16/05/22	39	Jesús Manuel García Prieto

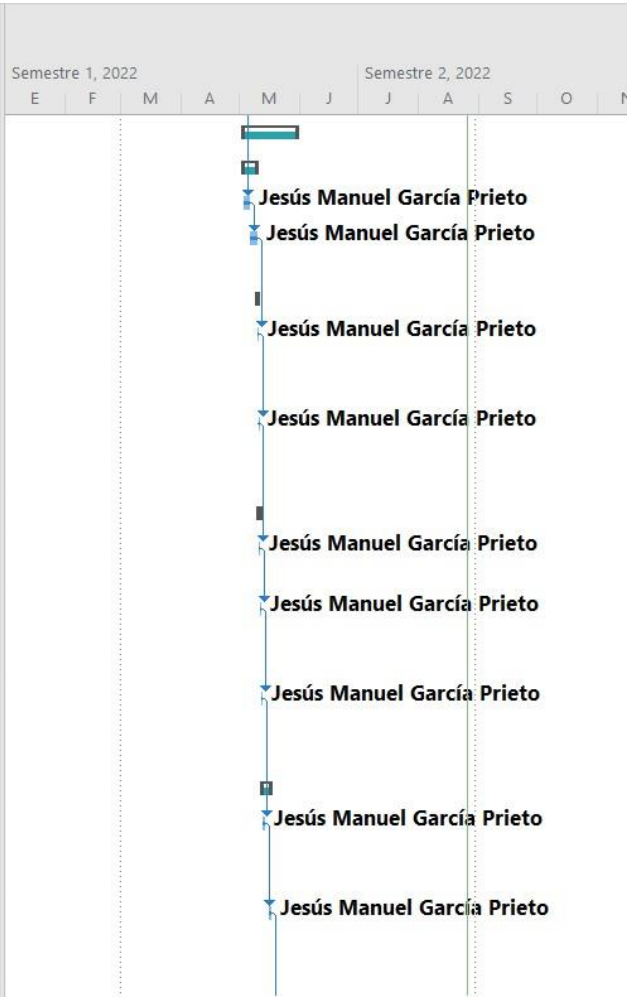


Figura 44. Diagrama de Gantt y Planificación temporal (III)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Implementación	**8 días**	**mar 17/05/22**	**jue 26/05/22**		
Implementación del Subsistema "Gestión de Actividades Sociales"	3 días	mar 17/05/22	jue 19/05/22	40	Jesús Manuel García Prieto
Implementación del Subsistema "Gestión de Retransmisión"	4 días	vie 20/05/22	mié 25/05/22	42	Jesús Manuel García Prieto
Integración y Despliegue de los subsistemas implementados	1 día	jue 26/05/22	jue 26/05/22	43	Jesús Manuel García Prieto
Pruebas	**2 días**	**vie 27/05/22**	**lun 30/05/22**		
Pruebas del Subsistema "Gestión de Actividades Sociales"	0,5 días	vie 27/05/22	vie 27/05/22	44	Jesús Manuel García Prieto
Pruebas del Subsistema "Gestión de Retransmisión"	0,5 días	vie 27/05/22	vie 27/05/22	46	Jesús Manuel García Prieto
Pruebas de Integración	1 día	lun 30/05/22	lun 30/05/22	47	Jesús Manuel García Prieto
Hito 2	0 días	lun 30/05/22	lun 30/05/22	48	Jesús Manuel García Prieto
Iteración 3	**24 días**	**mar 31/05/22**	**vie 01/07/22**		
Modelo del negocio	**7 días**	**mar 31/05/22**	**mié 08/06/22**		
Investigación sobre integración de servidores web	5 días	mar 31/05/22	lun 06/06/22	46	Jesús Manuel García Prieto
Investigación sobre la gestión e implementación de catálogos en plataformas de	2 días	mar 07/06/22	mié 08/06/22	52	Jesús Manuel García Prieto

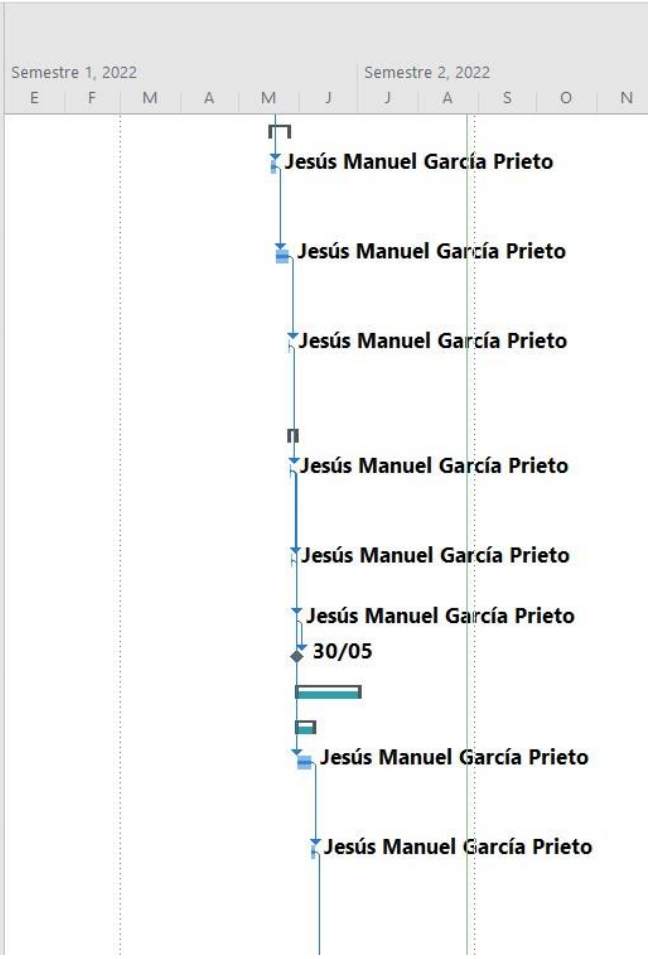


Figura 45. Diagrama de Gantt y Planificación temporal (IV)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Requisitos	**0,5 días**	**jue 09/06/22**	**jue 09/06/22**		
Refinamiento de requisitos del Subsistema "Gestión de Usuarios"	0,25 días	jue 09/06/22	jue 09/06/22	53	Jesús Manuel García Prieto
Refinamiento de requisitos del Subsistema "Gestión de Catálogo"	0,25 días	jue 09/06/22	jue 09/06/22	55	Jesús Manuel García Prieto
Análisis	**1 día**	**jue 09/06/22**	**vie 10/06/22**		
Refinamiento de análisis del Subsistema "Gestión de Usuarios"	0,5 días	jue 09/06/22	jue 09/06/22	56	Jesús Manuel García Prieto
Refinamiento de análisis del Subsistema "Gestión de Catálogo"	0,5 días	vie 10/06/22	vie 10/06/22	58	Jesús Manuel García Prieto
Diseño	**2 días**	**vie 10/06/22**	**mar 14/06/22**		
Refinamiento de diseño del Subsistema "Gestión de Usuarios"	1 día	vie 10/06/22	lun 13/06/22	59	Jesús Manuel García Prieto
Refinamiento de diseño del Subsistema "Gestión de Catálogo"	1 día	lun 13/06/22	mar 14/06/22	61	Jesús Manuel García Prieto
Implementación	**11,5 días**	**mar 14/06/22**	**mié 29/06/22**		
Implementación del Subsistema "Gestión de Usuarios"	6 días	mar 14/06/22	mié 22/06/22	62	Jesús Manuel García Prieto
Implementación del Subsistema "Gestión de Catálogo"	3 días	mié 22/06/22	lun 27/06/22	64	Jesús Manuel García Prieto

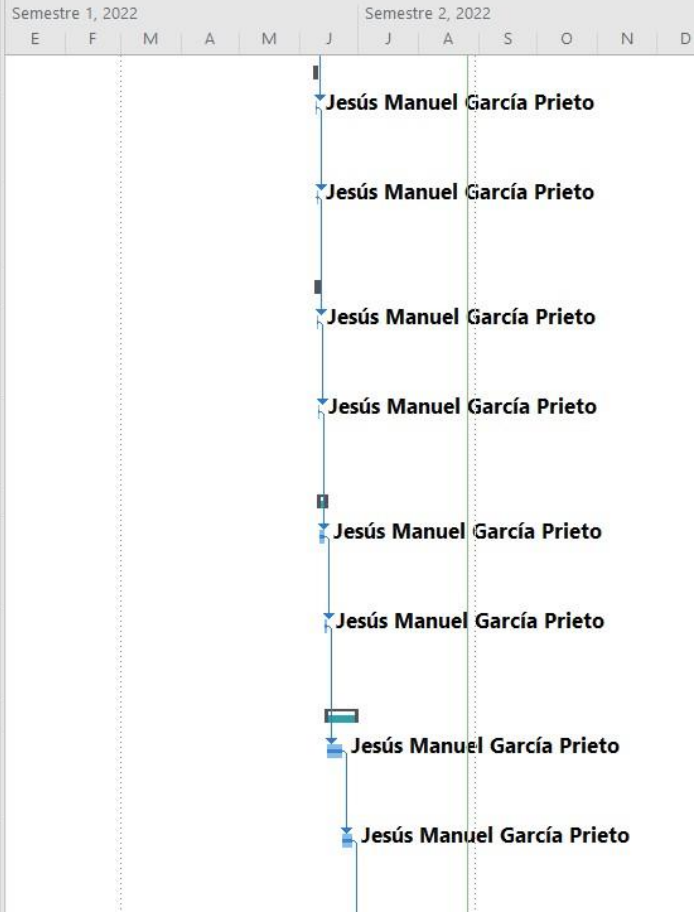


Figura 46. Diagrama de Gantt y Planificación temporal (V)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Integración y Despliegue de los subsistemas implementados	2,5 días	lun 27/06/22	mié 29/06/22	65	Jesús Manuel García Prieto
Pruebas	**2 días**	**jue 30/06/22**	**vie 01/07/22**		
Pruebas del Subsistema "Gestión de Usuarios"	0,5 días	jue 30/06/22	jue 30/06/22	66	Jesús Manuel García Prieto;
Pruebas del Subsistema "Gestión de Catálogo"	0,5 días	jue 30/06/22	jue 30/06/22	68	Jesús Manuel García Prieto
Pruebas de Integración	1 día	vie 01/07/22	vie 01/07/22	69	Jesús Manuel García Prieto
Hito 3	0 días	vie 01/07/22	vie 01/07/22	70	Jesús Manuel García Prieto
Iteración 4	**41 días**	**lun 04/07/22**	**lun 29/08/22**		
Modelo del negocio	**7 días**	**lun 04/07/22**	**mar 12/07/22**		
Revisión de posibles riesgos existentes en el	3 días	lun 04/07/22	mié 06/07/22	70	Jesús Manuel García Prieto
Revisión final de sistemas similares	4 días	jue 07/07/22	mar 12/07/22	74	Jesús Manuel García Prieto
Requisitos	**3 días**	**mié 13/07/22**	**vie 15/07/22**		
Refinamiento final de requisitos en función de los riesgos existentes	3 días	mié 13/07/22	vie 15/07/22	75	Jesús Manuel García Prieto
Análisis	**4 días**	**lun 18/07/22**	**jue 21/07/22**		
Revisión final del modelo de análisis	4 días	lun 18/07/22	jue 21/07/22	77	Jesús Manuel García Prieto
Diseño	**7 días**	**vie 22/07/22**	**lun 01/08/22**		
Revisión final del modelo de diseño	7 días	vie 22/07/22	lun 01/08/22	79	Jesús Manuel García Prieto

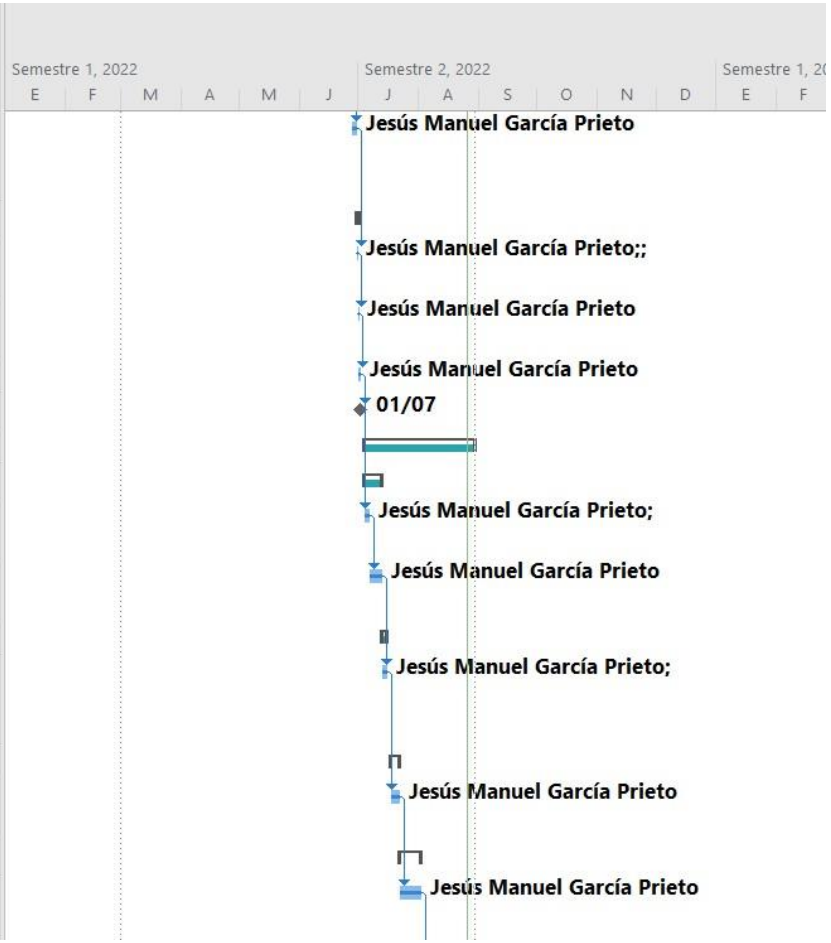


Figura 47. Diagrama de Gantt y Planificación temporal (VI)

Nombre de tarea	Duración	Comienzo	Fin		Nombres de los recursos
Revisión final del modelo de diseño	7 días	vie 22/07/22	lun 01/08/22	79	Jesús Manuel García Prieto
Implementación	**16 días**	**mar 02/08/22**	**mar 23/08/22**		
Integración y despliegue de los cambios realizados al sistema final	16 días	mar 02/08/22	mar 23/08/22	81	Jesús Manuel García Prieto
Pruebas	**4 días**	**mié 24/08/22**	**lun 29/08/22**		
Pruebas de los cambios realizados al sistema final	4 días	mié 24/08/22	lun 29/08/22	83	Jesús Manuel García Prieto
Hito 4	0 días	lun 29/08/22	lun 29/08/22	85	Jesús Manuel García Prieto

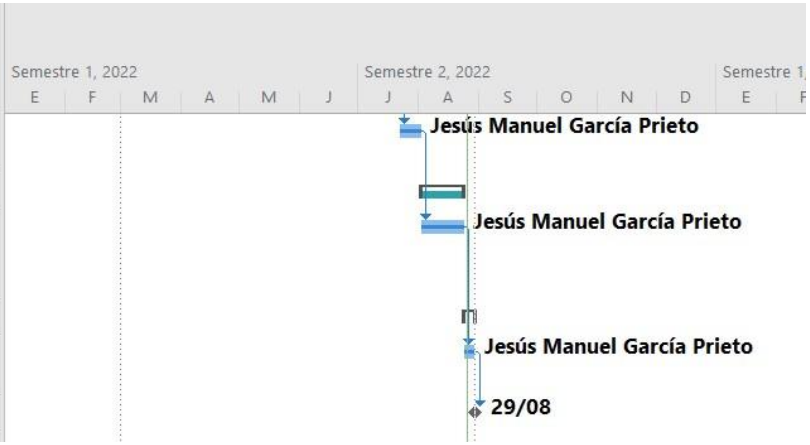


Figura 48. Diagrama de Gantt y Planificación temporal (VII)

4. CONCLUSIONES

Una vez realizadas tanto la estimación como la planificación temporal del proyecto se concluyeque este puede ser realizado en el marco de trabajo estimado, comenzando el proyecto en marzo se garantiza el tener incluso unos días antes de la fecha final de entrega (7 de septiembre)en caso de que se decidiera incluir alguna funcionalidad extra o corregir algún error de última hora.

A modo de repaso se recuerda que el proyecto se estimó en 130 días, si bien se afirmó que se trabajarían 8h diarias, al tratarse de verano el desarrollador tuvo más tiempo algunos días, por lo que, de nuevo, se reafirma la idea de que el proyecto se pudo realizar en el marco temporal establecido.

ANEXO II: Especificación de requisitos software.

Plataforma streaming multidispositivo de videojuegos

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Agosto de 2022

Autor:

Jesús Manuel García Prieto

Tutores:

Diego J. Valdeolmillos Villaverde

Alfonso González Briones

Francisco Pinto Santos

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este documento se recoge todo aquello relacionado con la captura y especificación de requisitos del sistema a presentar.

Como ya sabemos, el sistema busca, como objetivo principal, proveer una plataforma de transmisión de vídeo de una fuente a un cliente, para lograr esto nos valdremos de la metodología de Durán y Bernárdez para la captura y definición de requisitos del sistema, por tanto, la estructura del documento será la siguiente:

- **Definición de participantes**
- **Definición de objetivos del sistema**
- **Captura y especificación de requisitos del sistema**
    - **Diagrama de paquetes**
    - **Definición de actores**
    - **Definición de requisitos funcionales del sistema (Casos de Uso)**
    - **Definición de requisitos no funcionales del sistema**
- **Matriz de rastreabilidad**

2. PARTICIPANTES

Dentro de la misma organización (Universidad de Salamanca) se encuentran los participantes (personas físicas) de este proyecto:

Tabla 1. Organización (Universidad de Salamanca)

Organización	**Universidad de Salamanca**
Dirección	Plaza de los Caídos, s/n, 37008, Salamanca
Teléfono	923294450
Fax	
Comentarios	Ninguno

Tabla 2. Participante (Jesús Manuel García Prieto)

Participante	**Jesús Manuel García Prieto**
Organización	Universidad de Salamanca
Rol	Desarrollador
Es desarrollador	Sí
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 3. Participante (Diego J. Valdeolmillos Villaverde)

Participante	**Diego J. Valdeolmillos Villaverde**
Organización	Universidad de Salamanca
Rol	Tutor
Es desarrollador	No
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 4. Participante (Alfonso González Briones)

Participante	**Alfonso González Briones**
Organización	Universidad de Salamanca
Rol	Tutor
Es desarrollador	No
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 5. Participante (Francisco Pinto Santos)

Participante	**Francisco Pinto Santos**
Organización	Universidad de Salamanca
Rol	Tutor
Es desarrollador	No
Es cliente	No
Es usuario	No
Comentarios	Ninguno

3. OBJETIVOS DEL SISTEMA

A continuación, se procede a listar los diferentes objetivos requeridos para la correcta satisfacción de los requisitos del sistema siguiendo la metodología de Durán y Bernárdez.

Tabla 6. OBJ-0001 Gestión de streaming multimedia

OBJ-0001	**Gestión de Streaming Multimedia**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*El sistema deberá ser capaz de emitir un flujo de imágenes desde la fuente al cliente.*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 7. OBJ-0002 Gestión de usuarios

OBJ-0002	**Gestión de Usuarios**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*El sistema deberá ser capaz de proporcionar a los usuarios un método de registro, login, baja y modificación de cuentas.*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 8. OBJ-0003 Gestión de catálogo

OBJ-0003	**Gestión de Catálogo**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*El sistema deberá ser capaz de mostrar, buscar y filtrar en un catálogo de videojuegos al usuario*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 9. OBJ-0004 Gestión de interacción con dispositivos físicos

OBJ-0004	**Gestión de Interacción con Dispositivos Físicos**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*El sistema deberá ser capaz de capturar inputs de diferentes dispositivos (teclado, mando, ratón…)*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 10. OBJ-0005 Gestión de actividades sociales

OBJ-0005	**Gestión de Actividades Sociales**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*El sistema deberá ser capaz de proporcionar una serie de funciones sociales al usuario (lista de amigos, chat privado…)*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

4. CATÁLOGO DE REQUISITOS DEL SISTEMA

De manera previa a la especificación de cada uno de los requisitos del sistema, se presenta un catálogo ordenado de todos ellos, comenzando por los requisitos de información, posteriormente los requisitos funcionales y, por último, los requisitos no funcionales.

De cada requisito se incluirá su identificador propio y una breve descripción del mismo.

4.1. CATÁLOGO DE REQUISITOS DE INFORMACIÓN

Tabla 11. Catálogo de Requisitos de Información

ID	Nombre	Descripción
IRQ-0001	Información de Usuario	El sistema deberá almacenar la información correspondiente a los usuarios del mismo, concretamente:
IRQ-0002	Información de Videojuego	El sistema deberá almacenar la información correspondiente a los videojuegos del mismo, concretamente:

4.2. CATÁLOGO DE REQUISITOS FUNCIONALES (CASOS DE USO)

Tabla 12. Tabla de Requisitos Funcionales

ID	Nombre	Descripción
UC-0001	Iniciar Sesión	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite iniciar sesión en el sistema
UC-0002	Registro	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite registrarse en el sistema
UC-0003	Recuperar Contraseña	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite recuperar su contraseña
UC-0004	Consultar Perfil	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite consultar su perfil personal
UC-0005	Modificar Datos	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite modificar sus datos personales
UC-0006	Cerrar Sesión	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite cerrar su sesión
UC-0007	Eliminar Cuenta	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite dar de baja su cuenta
UC-0008	Cambiar Idioma (WIP)	
UC-0009	Agregar Amigo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite agregar a otro usuario como amigo
UC-0010	Eliminar Amigo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite eliminar a otro usuario de su lista de amigos
UC-0011	Bloquear Usuario	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite bloquear a otro usuario
UC-0012	Eliminar Bloqueo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite ver el catálogo
UC-0013	Listar Catálogo	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite agregar a otro usuario como amigo

UC-0014	Filtrar Búsqueda	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite eliminar a otro usuario de su lista de amigos
UC-0015	Realizar Búsqueda Manualmente	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite bloquear a otro usuario
UC-0016	Seleccionar Juego/Simulación	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite ver el catálogo
UC-0017	Iniciar Conexión	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba una petición de inicio de conexión
UC-0018	Realizar Input	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el actor ACT-0002 Usuario realice un input
UC-0019	Capturar Mouse	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba un evento de captura de mouse
UC-0020	Capturar Teclado	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba un evento de captura de teclado
UC-0021	Enviar Inputs	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema envíe un input
UC-0022	Recibir Inputs	El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0004 Servidor Fuente reciba un input
UC-0023	Enviar Retransmisión	El Sistema deberá comportarse como se indica en el siguiente caso de uso de manera periódica cuando el Actor ACT-0004 Servidor Fuente se conecte con el actor ACT-0002 Usuario
UC-0024	Recibir Retransmisión	El Sistema deberá comportarse como se indica en el siguiente caso de uso de manera periódica cuando el Actor ACT-0002 Usuario se conecte con el actor ACT-0004 Servidor Fuente

4.3. CATÁLOGO DE REQUISITOS NO FUNCIONALES

Tabla 13. Catálogo de Requisitos no Funcionales

ID	Nombre	Descripción
NFR-0001	Portabilidad	El sistema debe poder dar soporte a varios sistemas operativos (Windows, Linux, iOS…)
NFR-0002	Concurrencia	El sistema debe soportar el funcionamiento con varios clientes simultáneamente
NFR-0003	Eficiencia (Inputs)	El sistema deberá registrar e implementar los inputs del cliente en tiempo real, con el menor retardo posible
NFR-0004	Eficiencia (Retransmisión)	El sistema deberá retransmitir el vídeo al cliente con el menor retardo posible y con la mayor calidad de imagen posible
NFR-0005	Escalabilidad	El sistema deberá poder realizar múltiples retransmisiones simultáneamente
NFR-0006	Roles	El sistema deberá poseer una jerarquía de roles de usuario, con un superusuario que gestione las acciones pertinentes.
NFR-0007	Seguridad (Retransmisión)	El sistema deberá asegurar la señal transportada en la retransmisión
NFR-0008	Seguridad (BBDD)	El sistema deberá ofrecer un almacenamiento seguro de los datos sensibles del usuario
NFR-0009	Servidor Web	El sistema poseerá un servidor propio que gestione el backend, así como la retransmisión servidor fuente-cliente
NFR-0010	Usabilidad	El sistema poseerá una interfaz de usuario fácilmente comprensible por el cliente
NFR-0011	Accesibilidad	El sistema poseerá una interfaz de usuario responsive

5. DEFINICIÓN DE REQUISITOS DEL SISTEMA

5.1. REQUISITOS DE INFORMACIÓN

En este apartado se exponen aquellos elementos persistentes en el sistema, bien sea por su almacenamiento en la BBDD del mismo o de forma estática en el código, de nuevo, siguiendo la metodología de Durán de Bernárdez.

Tabla 14. IRQ-0001 Información de usuario

IRQ-0001	**Información de Usuario**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	Ninguno	
Descripción	El sistema deberá almacenar la información correspondiente a los usuarios del mismo, concretamente:	
Datos específicos	- Nombre - Apellidos - Email - Contraseña - Token	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Tabla 15. IRQ-0002 Información de videojuego

IRQ-0002	**Información de Videojuego**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	Ninguno	
Descripción	El sistema deberá almacenar la información correspondiente a los videojuegos del mismo, concretamente:	
Datos específicos	- Nombre - Icono - Categoría - Ruta	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

5.2. REQUISITOS FUNCIONALES

En este apartado se mostrarán los requisitos funcionales, que indican el comportamiento del sistema ante las diferentes situaciones a las que este se enfrentará, así como los servicios que este proporcionará.

En primer lugar, se presentará la estructura del sistema y de sus CU en forma de diagrama de componentes, por último, se irán listando todos los requisitos capturados, comenzando por los funcionales (Casos de Uso) y terminando con los no funcionales y la matriz de rastreabilidad que muestra las relaciones entre ellos.

5.2.1. DIAGRAMA DE PAQUETES

Como se puede observar en la **Figura 1** , se presenta el diagrama de paquetes del sistema, que nos muestra la división del mismo. Se pueden distinguir 4 paquetes principales: Gestión de usuarios, Gestión de Funciones Sociales, Gestión de retransmisión y Gestión de catálogo.

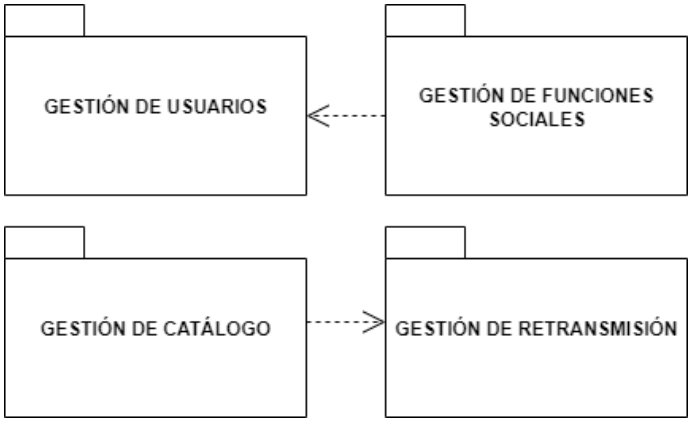


Figura 1. Diagrama de paquetes

5.2.2. DEFINICIÓN DE ACTORES

Podemos identificar a los siguientes actores en nuestro proyecto, junto a su jerarquía:

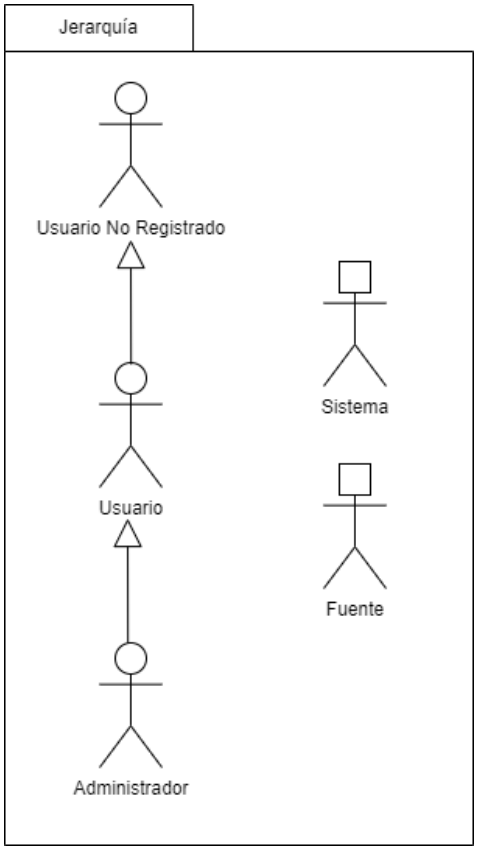


Figura 2. Jerarquía de Actores

A continuación, se presentan las tablas con las definiciones de estos actores:

Tabla 16. ACT-0001 Usuario No Registrado

ACT-0001	**Usuario No Registrado**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*Este actor representa al usuario que aún no ha realizado los procedimientos requeridos para la autenticación en el sistema*
Comentarios	Ninguno

Tabla 17. ACT-0002 Usuario

ACT-0002	**Usuario**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*Este actor representa al usuario que ya ha realizado el registro en el sistema*
Comentarios	Ninguno

Tabla 18. ACT-0003 Administrador

ACT-0003	**Administrador**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*Este actor representa a un superusuario con permisos especiales dedicados a la administración del sistema, entre sus funciones se encuentra la gestión de usuarios y del catálogo*
Comentarios	Ninguno

Tabla 19. ACT-0004 Servidor Fuente

ACT-0004	**Fuente**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*Este actor representa a un objeto hardware, concretamente, al servidor emisor de vídeo al usuario determinado*
Comentarios	Ninguno

Tabla 20. ACT-0005 Sistema

ACT-0005	**Sistema**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Descripción	*Este actor representa una abstracción del propio sistema, su función principal consiste en la detección de eventos como la captura de inputs*
Comentarios	Ninguno

5.2.3. CASOS DE USO DEL SISTEMA

Es finalmente en este apartado donde se exponen las tablas y diagramas de los diferentes Casos de Uso que componen el sistema.

5.2.3.1. GESTIÓN DE USUARIOS

En la **Figura 3** se puede observar el diagrama correspondiente a este apartado, en él se detallan las diferentes acciones relativas a los usuarios, entre ellas se encuentra el acceso a su perfil y las funciones de gestión sobre el mismo.

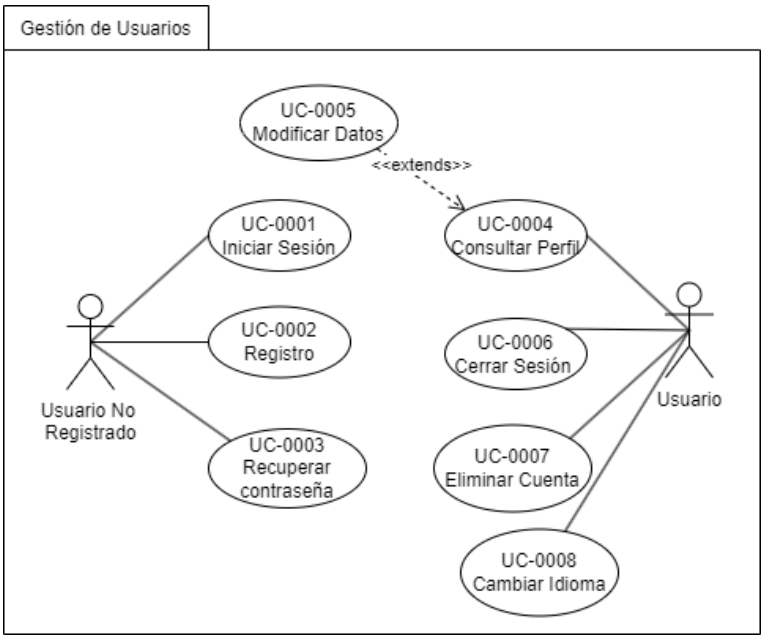


Figura 3. Gestión de Usuarios

Tabla 21. UC-0001 Iniciar Sesión

UC-0001	**Iniciar Sesión**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite iniciar sesión en el sistema*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0001 Usuario No Registrado solicita iniciar sesión en el sistema.*
	2	*El sistema solicita al actor ACT-0001 Usuario No Registrado su correo electrónico y contraseña.*
	3	*El actor ACT-0001 Usuario No Registrado introduce los datos solicitados*
	4	*El sistema valida los datos recibidos*
	5	*El sistema redirige al actor ACT-0001 Usuario Sin Registrar a su página personal*
Postcondición	*El actor ACT-0001 Usuario No Registrado pasa a ser el actor ACT-0002 Usuario*	
Excepciones	**Paso**	**Acción**
	4	*Si los datos validados son incorrectos, el sistema notifica al actor ACT-0001 Usuario No Registrado y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 22. UC-0002 Registro

UC-0002	**Registro**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite registrarse en el sistema*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0001 Usuario No Registrado solicita registrarse en el sistema.*
	2	*El sistema solicita al actor ACT-0001 Usuario No Registrado los datos requeridos para el registro.*
	3	*El actor ACT-0001 Usuario No Registrado introduce los datos solicitados*
	4	*El sistema valida los datos recibidos*
	5	*El sistema registra al usuario en la base de datos*
	6	*El sistema redirige al actor ACT-0001 Usuario Sin Registrar a su página personal*
Postcondición	*El actor ACT-0001 Usuario No Registrado pasa a ser el actor ACT-0002 Usuario*	
Excepciones	**Paso**	**Acción**
	4	*Si los datos validados son incorrectos, el sistema notifica al actor ACT-0001 Usuario No Registrado y el caso de uso queda sin efecto*
	5	*Si se da un error durante el registro en la base de datos, el sistema notifica al actor ACT-0001 Usuario No Registrado y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 23. UC-0003 Recuperar Contraseña

UC-0003	**Recuperar Contraseña**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0001 Usuario No Registrado solicite recuperar su contraseña*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0001 Usuario No Registrado solicita recuperar su contraseña*
	2	*El sistema solicita al actor ACT-0001 Usuario No Registrado su correo electrónico*
	3	*El actor ACT-0001 Usuario No Registrado introduce los datos solicitados*
	4	*El sistema valida los datos recibidos*
	5	*El sistema envía al actor ACT-0001 Usuario No Registrado una nueva contraseña temporal al correo indicado*
Postcondición		
Excepciones	**Paso**	**Acción**
	4	*Si el correo proporcionado por el actor ACT-0001 Usuario No Registrado no se encuentra almacenado en el sistema, este le notifica de vuelta y este caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 24. UC-0004 Consultar Perfil

UC-0004	**Consultar Perfil**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite consultar su perfil personal*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita consultar su perfil personal*
	2	*El sistema redirige al ACT-0002 Usuario a su perfil*
	3	*Si el actor ACT-0002 solicita modificar sus datos se inicia el caso de uso UC-0005 Modificar Datos*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 25. UC-0005 Modificar Datos

UC-0005	**Modificar Datos**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite modificar sus datos personales*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El sistema proporciona al actor ACT-0002 Usuario un formulario de modificación de datos*
	2	*El actor ACT-0002 Usuario modifica los datos deseados*
	3	*El sistema valida los datos recibidos*
	4	*El sistema modifica los datos del actor ACT-0002 Usuario en la base de datos*
	5	*El sistema notifica al actor ACT-0002 sobre la modificación de sus datos*
Postcondición		
Excepciones	**Paso**	**Acción**
	3	*Si los datos validados son incorrectos, el sistema notifica al actor ACT-0001 Usuario No Registrado y el caso de uso queda sin efecto*
	4	*Si se da un error durante el registro en la base de datos, el sistema notifica al actor ACT-0002 Usuario y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 26. UC-0006 Cerrar Sesión

UC-0006	**Cerrar Sesión**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite cerrar su sesión*	
Precondición	*El actor ACT-0002 debe tener una sesión iniciada*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita cerrar su sesión*
	2	*El sistema cierra la sesión del actor ACT-0002 Usuario y le redirige a la pantalla de registro/login*
Postcondición		
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 27. UC-0007 Eliminar Cuenta

UC-0007	**Eliminar Cuenta**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite dar de baja su cuenta*	
Precondición	*El actor ACT-0002 debe tener una sesión iniciada*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita dar de baja su cuenta*
	2	*El sistema elimina los datos del actor ACT-0002 Usuario en la base de datos y le redirige a la pantalla de registro/login*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si se da un error durante la eliminación en la base de datos, el sistema notifica al actor ACT-0002 Usuario y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 28. UC-0008 Cambiar idioma

UC-0008	**Cambiar Idioma**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite cambiar el idioma de la aplicación*	
Precondición	*El actor ACT-0002 debe tener una sesión iniciada*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita cambiar el idioma de su cuenta*
	2	*El sistema actualiza el idioma de la aplicación y muestra al Actor ACT-0002 la versión escogida por este*
	Paso	**Acción**
Postcondición	*Si el actor ACT-0002 Usuario decide denegar su decisión, el sistema es notificado y el caso de uso queda sin efecto*	
Excepciones		
	Paso	**Tiempo máximo**
	-	-
Rendimiento		
	Vital	
Frecuencia esperada	inmediatamente	
Importancia	validado	
Urgencia		
Estado		

5.2.3.2. GESTIÓN DE FUNCIONES SOCIALES

En la **Figura 4** se puede observar el diagrama correspondiente a este apartado, en él se detallan las diferentes acciones sociales que un usuario puede realizar en el sistema, entre estas destacan la lista de amigos o la opción de chatear con estos.

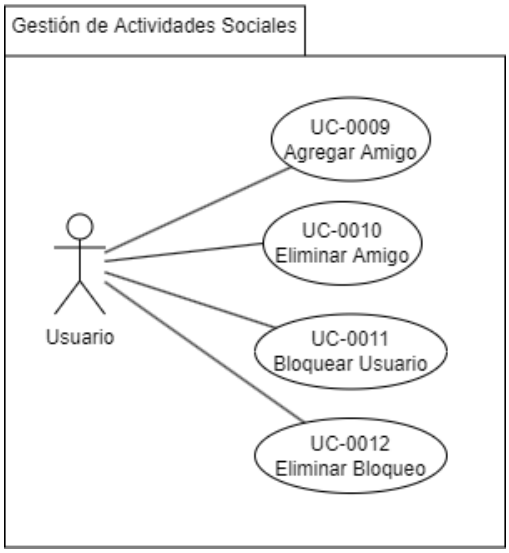


Figura 4. Gestión de Actividades Sociales

Tabla 29. UC-0009 Añadir Amigo

UC-0009	**Agregar Amigo**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS • OBJ-0006 FUNCIONES SOCIALES	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite agregar a otro usuario como amigo*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita agregar a otro usuario como amigo*
	2	*El sistema introduce de manera recíproca a ambos actores ACT-0002 Usuario en sus respectivas listas de amigos*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si se da un error durante la inserción en la base de datos, el sistema notifica a ambos actores ACT-0002 Usuario y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 30. UC-0010 Eliminar Amigo

UC-0010	**Eliminar Amigo**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0006 FUNCIONES SOCIALES	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite eliminar a otro usuario de su lista de amigos*	
Precondición	*El actor ACT-0002 Usuario debe tener al menos un amigo en su lista de amigos*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita eliminar a otro usuario de su lista de amigos*
	2	*El sistema elimina al actor ACT-0002 Usuario seleccionado de la lista de amigos*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si se da un error durante la eliminación en la base de datos, el sistema notifica al actor ACT-0002 Usuario y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 31. UC-0011 Bloquear Usuario

UC-0011	**Bloquear Usuario**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS • OBJ-0006 FUNCIONES SOCIALES	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite bloquear a otro usuario*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita bloquear a otro usuario*
	2	*El sistema introduce al actor ACT-0002 Usuario bloqueado en la lista de usuarios vetados*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si se da un error durante la inserción en la base de datos, el sistema notifica al actor ACT-0002 Usuario y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 32. UC-0012 Eliminar bloqueo

UC-0012	**Eliminar Bloqueo**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0006 FUNCIONES SOCIALES	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite eliminar a otro usuario de su lista de bloqueados*	
Precondición	*El actor ACT-0002 Usuario debe tener al menos un usuario en su lista de bloqueados*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita eliminar a otro usuario de su lista de bloqueados*
	2	*El sistema elimina al actor ACT-0002 Usuario seleccionado de la lista de bloqueados*
Postcondición		
Excepciones	**Paso**	**Acción**
	1	*Si se da un error durante la eliminación en la base de datos, el sistema notifica al actor ACT-0002 Usuario y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

5.2.3.3. GESTIÓN DE CATÁLOGO

En la **Figura 5** se puede observar el diagrama correspondiente a este apartado, en él se detallan las diferentes acciones relacionadas con el catálogo de videojuegos, entre ellas se encuentran los diferentes filtros que existen en él y el propio listado en sí.

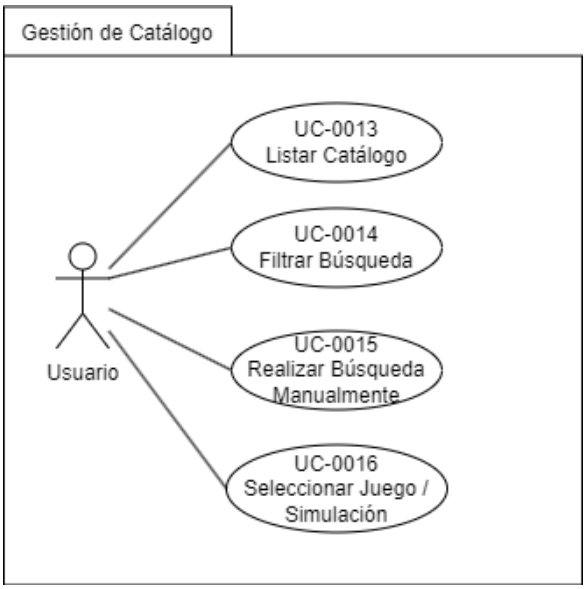


Figura 5. Gestión de Catálogo

Tabla 33. UC-0013 Listar Catálogo

UC-0013	**Listar Catálogo**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0004 VISUALIZACIÓN, BÚSQUEDA Y FILTRADO DE VIDEOJUEGOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite ver el catálogo*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita ver el catálogo*
	2	*El sistema obtiene la lista de videojuegos de la base de datos*
	3	*El sistema lista los videojuegos al actor ACT-0002 Usuario*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si se da un error durante el acceso a la base de datos, el sistema notifica al actor ACT-0002 Usuario y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 34. UC-0014 Filtrar Búsqueda

UC-0014	**Filtrar Búsqueda**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0004 VISUALIZACIÓN, BÚSQUEDA Y FILTRADO DE VIDEOJUEGOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario solicite filtrar el catálogo*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario solicita filtrar el catálogo*
	2	*El sistema reordena la lista de videojuegos en función del filtro seleccionado por el actor ACT-0002 Usuario*
	3	*El sistema lista los videojuegos filtrados al actor ACT-0002 Usuario*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si ningún videojuego cumple el filtro el sistema muestra un catálogo vacío y notifica al actor ACT-0002 Usuario*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 35. UC-0015 Realizar Búsqueda Manualmente

UC-0015	**Realizar Búsqueda Manualmente**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0004 VISUALIZACIÓN, BÚSQUEDA Y FILTRADO DE VIDEOJUEGOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario realice una búsqueda manual*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario realiza una búsqueda manual*
	2	*El sistema compara la búsqueda introducida por el actor ACT-0002 Usuario con la lista de videojuegos*
	3	*El sistema lista los videojuegos coincidentes con la búsqueda manual al actor ACT-0002 Usuario*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si ningún videojuego coincide con la búsqueda manual el sistema muestra un catálogo vacío y notifica al actor ACT-0002 Usuario*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 36. UC-0016 Seleccionar Juego/Simulación

UC-0016	**Seleccionar Juego/Simulación**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0004 VISUALIZACIÓN, BÚSQUEDA Y FILTRADO DE VIDEOJUEGOS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0002 Usuario seleccione un videojuego del catálogo*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario selecciona un videojuego del catálogo*
	2	*Se inicia el caso de uso UC-0017 Iniciar Conexión*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

5.2.3.4. GESTIÓN DE RETRANSMISIÓN

En la **Figura 6** se puede observar el diagrama correspondiente a este apartado, en él se detallan las diferentes acciones relacionadas con la retransmisión de vídeo, así como la captura de inputs.

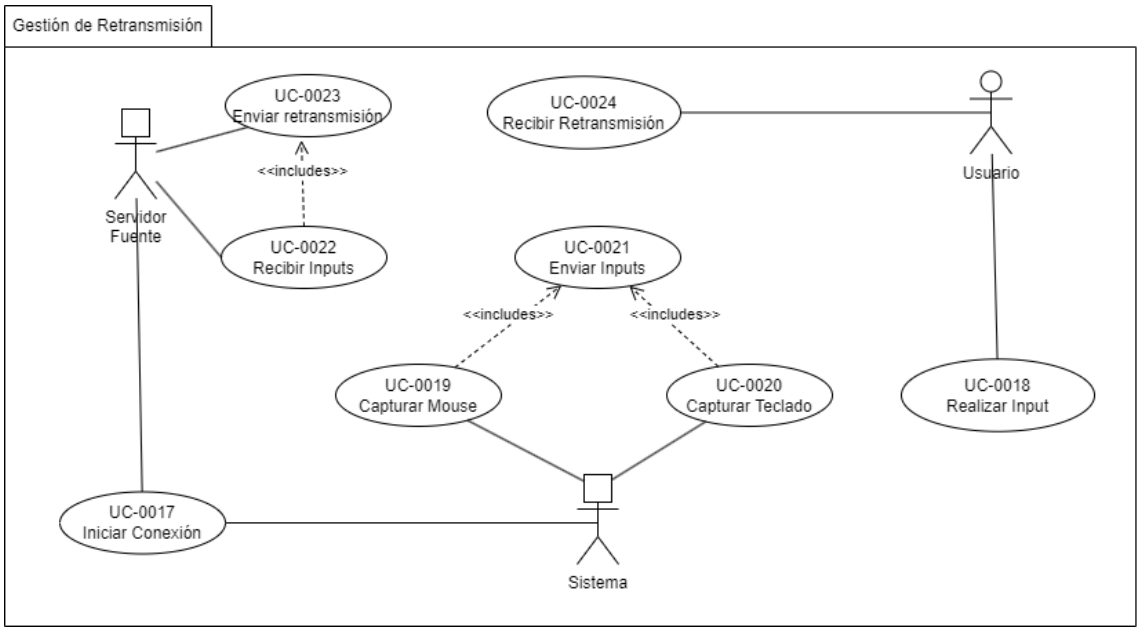


Figura 6. Gestión de retransmisión

Tabla 37. UC-0017 Iniciar Conexión

UC-0017	**Iniciar Conexión**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba una petición de inicio de conexión*	
Precondición	*El actor ACT-0002 Usuario ha seleccionado un videojuego del catálogo*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0005 Sistema recibe una petición de inicio de conexión*
	2	*El actor ACT-0005 Sistema añade al actor ACT-0002 Usuario en una habitación propia con el actor ACT-0004 Servidor Fuente*
	3	*El sistema notifica al actor ACT-0002 Usuario*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	*Si se da un error durante la conexión entre el actor ACT-0002 Usuario y el actor ACT-0004 Servidor Fuente el sistema le notifica y el caso de uso queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 38. UC-0018 Realizar Input

UC-0018	**Realizar Input**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0005 CAPTURA DE INPUTS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el actor ACT-0002 Usuario realice un input*	
Precondición	*El Sistema debe encontrarse en medio de una transmisión*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0002 Usuario realiza un input*
	2	*El sistema recibe la señal del Input y llama al evento correspondiente de captura de inputs*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 39. UC-0019 Capturar Mouse

UC-0019	**Capturar Mouse**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0005 CAPTURA DE INPUTS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba un evento de captura de mouse*	
Precondición	*El sistema se encuentra en medio de una transmisión*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0005 Sistema recibe un evento de captura de mouse*
	2	*El actor ACT-0005 Sistema determina el input recibido (click, movimiento, ruleta…)*
	3	*Se inicia el caso de uso UC-0021 Enviar Inputs*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 40. UC-0020 Capturar Teclado

UC-0020	**Capturar Teclado**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0005 CAPTURA DE INPUTS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema reciba un evento de captura de teclado*	
Precondición	*El sistema se encuentra en medio de una transmisión*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0005 Sistema recibe un evento de captura de teclado*
	2	*El actor ACT-0005 Sistema determina el input recibido (tipo de tecla presionada)*
	3	*Se inicia el caso de uso UC-0021 Enviar Inputs*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 41. UC-0021 Enviar Inputs

UC-0021	**Enviar Inputs**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0005 CAPTURA DE INPUTS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0005 Sistema envíe un input*	
Precondición	*El sistema se encuentra en medio de una transmisión*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0005 Sistema envía el input al actor ACT-0004 Servidor Fuente*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 42. UC-0022 Recibir Inputs

UC-0022	**Recibir Inputs**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0005 CAPTURA DE INPUTS	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso cuando el Actor ACT-0004 Servidor Fuente reciba un input*	
Precondición	*El sistema se encuentra en medio de una transmisión*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0004 Servidor Fuente recibe el input*
	2	*El sistema aplica este input en el actor ACT-0004 Servidor Fuente*
	3	*Se inicia el caso de uso UC-0023 Enviar Retransmisión*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 43. UC-0023 Enviar Retransmisión

UC-0023	**Enviar Retransmisión**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso de manera periódica cuando el Actor ACT-0004 Servidor Fuente se conecte con el actor ACT-0002 Usuario*	
Precondición	*El sistema se encuentra en medio de una transmisión*	
Secuencia normal	**Paso**	**Acción**
	1	*El actor ACT-0004 Servidor Fuente captura su pantalla*
	2	*El sistema empaqueta y transporta la imagen*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

Tabla 44. UC-0024 Recibir Retransmisión

UC-0024	**Recibir Retransmisión**	
Versión	1.0	
Autores	Participante (Jesús Manuel García Prieto)	
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)	
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO	
Descripción	*El Sistema deberá comportarse como se indica en el siguiente caso de uso de manera periódica cuando el Actor ACT-0002 Usuario se conecte con el actor ACT-0004 Servidor Fuente*	
Precondición	*El sistema se encuentra en medio de una transmisión*	
Secuencia normal	**Paso**	**Acción**
	1	*El sistema entrega la imagen del actor ACT-0004 Servidor Fuente al actor ACT-0002 Usuario*
	2	*El sistema muestra la imagen recibida al actor ACT-0002 Usuario*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	inmediatamente	
Estado	validado	

5.2. REQUISITOS NO FUNCIONALES

Es en este apartado donde nos centraremos en aquellas restricciones del sistema y en sus estándares de calidad, así como aquellos factores externos que puedan afectar al mismo. Ya que este proyecto se basa en varias tecnologías web y de sistemas distribuidos, existe una amplia variedad de requisitos no funcionales, como podemos observar a continuación:

Tabla 45. NFR-0001 Portabilidad

NFR-0001	**Portabilidad**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0005 CAPTURA DE INPUTS
Descripción	*El sistema debe poder dar soporte a varios sistemas operativos (Windows, Linux, iOS…)*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 46. NFR-0002 Concurrencia

NFR-0002	**Concurrencia**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS • OBJ-0005 CAPTURA DE INPUTS • OBJ-0006 FUNCIONES SOCIALES
Descripción	*El sistema debe soportar el funcionamiento con varios clientes simultáneamente*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 47. NFR-0004 Eficiencia (Inputs)

NFR-0003	**Eficiencia (Inputs)**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0005 CAPTURA DE INPUTS
Descripción	*El sistema deberá registrar e implementar los inputs del cliente en tiempo real, con el menor retardo posible*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 48. NFR-0005 Eficiencia (Retransmisión

NFR-0004	**Eficiencia (Retransmisión)**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO
Descripción	*El sistema deberá retransmitir el vídeo al cliente con el menor retardo posible y con la mayor calidad de imagen posible*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 49. NFR-0006 Escalabilidad

NFR-0005	**Escalabilidad**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO
Descripción	*El sistema deberá poder realizar múltiples retransmisiones simultáneamente*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 50. NFR-0007 Roles

NFR-0006	**Roles**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS
Descripción	*El sistema deberá poseer una jerarquía de roles de usuario, con un superusuario que gestione las acciones pertinentes.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 51. NFR-0008 Seguridad (Retransmisión

NFR-0007	**Seguridad (Retransmisión)**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO
Descripción	*El sistema deberá asegurar la señal transportada en la retransmisión*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 52. NFR-0009 Seguridad (BBDD)

NFR-0008	**Seguridad (BBDD)**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	Ninguno
Descripción	*El sistema deberá ofrecer un almacenamiento seguro de los datos sensibles del usuario*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 53. NFR-0010 Servidor Web

NFR-0009	**Servidor Web**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0001 EMISIÓN Y RECEPCIÓN DE VÍDEO • OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS • OBJ-0004 VISUALIZACIÓN, BÚSQUEDA Y FILTRADO DE VIDEOJUEGOS • OBJ-0005 CAPTURA DE INPUTS • OBJ-0006 FUNCIONES SOCIALES
Descripción	*El sistema poseerá un servidor propio que gestione el backend, así como la retransmisión servidor fuente-cliente*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 54. NFR-0011 Usabilidad

NFR-0010	**Usabilidad**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS • OBJ-0004 VISUALIZACIÓN, BÚSQUEDA Y FILTRADO DE VIDEOJUEGOS • OBJ-0006 FUNCIONES SOCIALES
Descripción	*El sistema poseerá una interfaz de usuario fácilmente comprensible por el cliente*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 55. NFR-0012 Accesibilidad

NFR-0011	**Accesibilidad**
Versión	1.0
Autores	Participante (Jesús Manuel García Prieto)
Fuentes	Participante (Diego J. Valdeolmillos Villaverde) Participante (Alfonso González Briones) Participante (Francisco Pinto Santos)
Dependencias	• OBJ-0002 REGISTRO, LOGIN, BAJAS Y MODIFICACIÓN DE USUARIOS • OBJ-0003 VISUALIZACIÓN DE PERFIL DE USUARIOS • OBJ-0004 VISUALIZACIÓN, BÚSQUEDA Y FILTRADO DE VIDEOJUEGOS • OBJ-0006 FUNCIONES SOCIALES
Descripción	*El sistema poseerá una interfaz de usuario responsive*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta

6. MATRIZ DE RASTREABILIDAD

Tabla 56. Matriz de Rastreabilidad

	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004	OBJ-0005	OBJ-0006
UC-0001		X				
UC-0002		X				
UC-0003		X				
UC-0004		X	X			
UC-0005		X	X			
UC-0006		X	X			
UC-0007		X	X			
UC-0008		X	X			
UC-0009			X			X
UC-0010		X				X
UC-0011			X			X
UC-0012		X				X
UC-0013				X		X
UC-0014				X		
UC-0015				X		
UC-0016	X			X		
UC-0017	X					
UC-0018	X				X	
UC-0019	X				X	
UC-0020	X				X	
UC-0021	X				X	
UC-0022	X				X	
UC-0023	X				X	
UC-0024	X					
NFR-0001	X				X	
NFR-0002	X	X	X		X	X
NFR-0003	X				X	
NFR-0004	X					
NFR-0005	X					
NFR-0006		X	X			
NFR-0007	X					
NFR-0008						
NFR-0009	X	X	X	X	X	X
NFR-0010		X	X	X		X
NFR-0011		X	X	X		X

7. CONCLUSIONES

Como pequeña conclusión sobre este anexo, se ha recogido el número total de requisitos del sistema, los resultados de esta recolección son los siguientes:

- 2 requisitos de información (IRQ)
- 5 actores (ACT)
- 24 requisitos funcionales (UC)
- 11 requisitos no funcionales (NFR)

Las conclusiones que se pueden sacar de esto son las siguientes:

- La baja cantidad de requisitos de información denota un sistema con baja cantidad de datos persistentes, el diseño de la base de datos será una etapa relativamente sencilla en cuanto a complejidad de la misma, y, considerando que no todos los datos se almacenarán en ella, esto se reduce aún más.
- Existe un alto número de actores, esto se debe al sistema de roles implementado en el sistema
- A pesar de existir una cantidad ligera de Casos de Uso (20-25) la mayoría de estos son complejos, esto se debe a que el gran grueso del proyecto se encuentra en el envío de vídeo e inputs, ambas funcionalidades son altamente complejas de programar, pero apenas proporcionan requisitos para el proyecto.

8. BIBLIOGRAFÍA

García Peñalvo, F. J., Moreno García, M. N., García Holgado, A., & Vázquez Ingelmo, A. (2019-2020). *INGENIERíA DEL SOFTWARE I. "Tema 8 - UML. Modified Modeling Language".* Obtenido de https://moodle2.usal.es/pluginfile.php/996986/mod_resource/content/4/IS_I%20Tema%208%20-%20UML.pdf

IBM. (05 de 03 de 2021). *Modelos de Casos de Uso.* Obtenido de https://www.ibm.com/docs/es/rsar/9.5?topic=approaches-use-case-models

OMG. (Diciembre 2017). *OMG Unified Modeling Language Specification. Version 2.5.1.*

ANEXO III: Análisis de requisitos.

Plataforma streaming multidispositivo de videojuegos

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

Agosto de 2022

Autor:

Jesús Manuel García Prieto

Tutores:

Diego J. Valdeolmillos Villaverde

Alfonso González Briones

Francisco Pinto Santos

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Tras la realización del Anexo II: Especificación de requisitos software, en el cual se recogían y especificaban los objetivos y requisitos del sistema, se pasa a la etapa de análisis, para lograr realizar correctamente esta etapa se seguirá el siguiente modelo:

- **Modelo de dominio:** Primer acercamiento a la definición de elementos del sistema mediante el diagrama de clases del mismo.
- **Realización de casos de uso-análisis:** Pretenden facilitar el entendimiento del envío de mensajes en el sistema en cada Caso de Uso en la etapa de análisis.
- **Clases de análisis:** Nos permiten identificar los diferentes elementos comunicativos del sistema de una manera más concreta mediante el uso de paquetes y clases de análisis.
- **Modelo arquitectónico del análisis:** Acercamiento menos abstracto de la etapa de análisis, sirve como primer paso para la etapa de diseño y como acercamiento al dominio de la solución.

2. MODELO DE DOMINIO

El objetivo principal de este modelo es lograr ser un primer acercamiento a la definición de los elementos del sistema, los cuales servirán más adelante a la hora de desarrollar el diseño del mismo, esto se logra, en primer lugar, al definir las clases conceptuales que aparecen en él.

Para diseñarlo se recogerán las diferentes clases relacionadas con los requisitos recogidos previamente, especialmente, los de información, y se asociarán entre sí. Esto se traduce en el siguiente diagrama de clases:

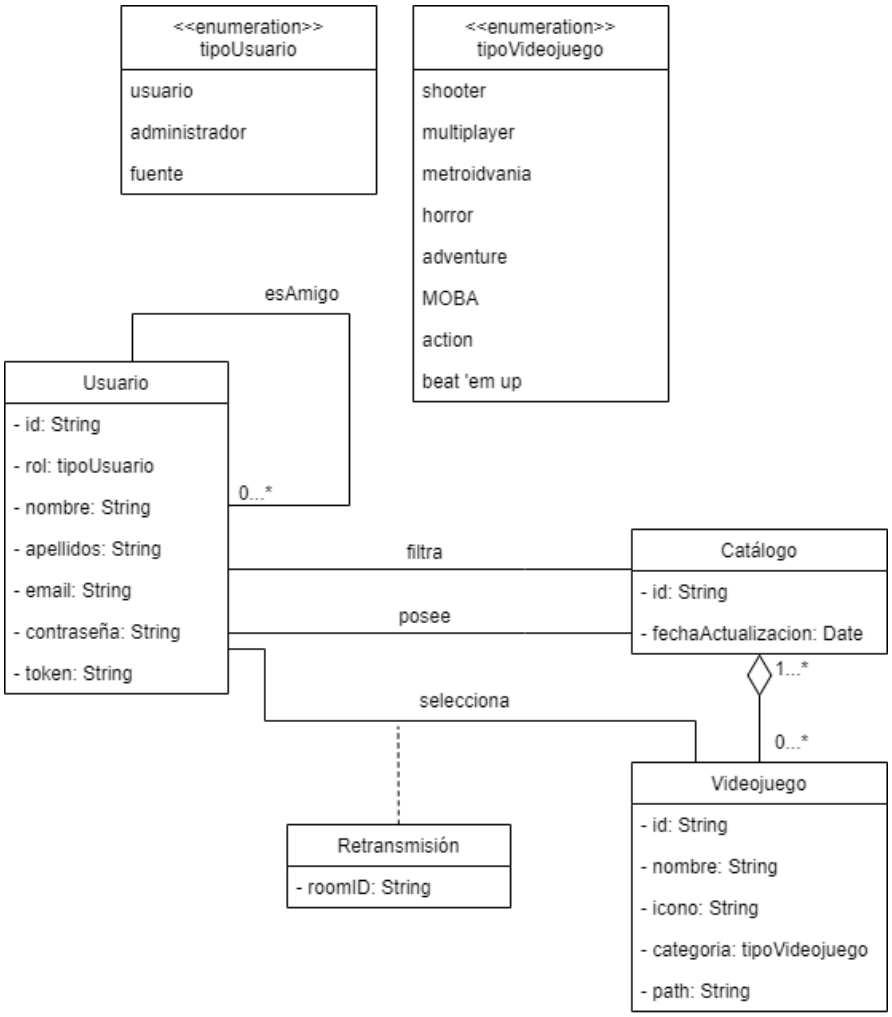


Figura 1. Modelo de Dominio

Se procede a realizar una breve explicación de las diferentes clases conceptuales que aparecen en la **Figura 1**:

- **Usuario:** clase principal del sistema, con sus diferentes roles este puede ser vetado o no de ciertas acciones, entre ellas destacan las relaciones que posee con otros usuarios (chat), o con el catálogo (seleccionar y filtrar videojuegos)
- **Catálogo:** clase que alberga un conjunto de videojuegos y se los muestra al usuario, puede ser filtrado por el mismo
- **Videojuego:** del mismo modo que un chat está conformado por mensajes, un catálogo está conformado de videojuegos, con la diferencia de que esta asociación se trata de una agregación y no de una composición, puesto que un videojuego puede aparecer en varios catálogos diferentes.

- **Retransmisión:** clase asociación dada por la selección de un videojuego por parte del usuario, su dato característico es el número identificador de la *room* en la que se lleva a cabo la retransmisión entre el usuario y el videojuego seleccionado.

3. REALIZACIÓN DE CASOS DE USO-ANÁLISIS

Se llega ya a la etapa de definición de la vista de interacción, debido al tipo de sistema que se va a desarrollar, se ha decidido proseguir en este apartado con los diagramas de secuencia, de entre todas las opciones disponibles, debido a que estos se centran en el intercambio de mensajes entre objetos.

Con el fin de facilitar la comprensión del análisis del sistema a alto nivel, se presentan a continuación estos diagramas ya mencionados.

Se mantendrá el orden y división establecidos en el "Anexo II: Especificación de requisitos software" a la hora de realizar estos diagramas de secuencia.

3.1. DIAGRAMAS DE SECUENCIA (GESTIÓN DE USUARIOS)

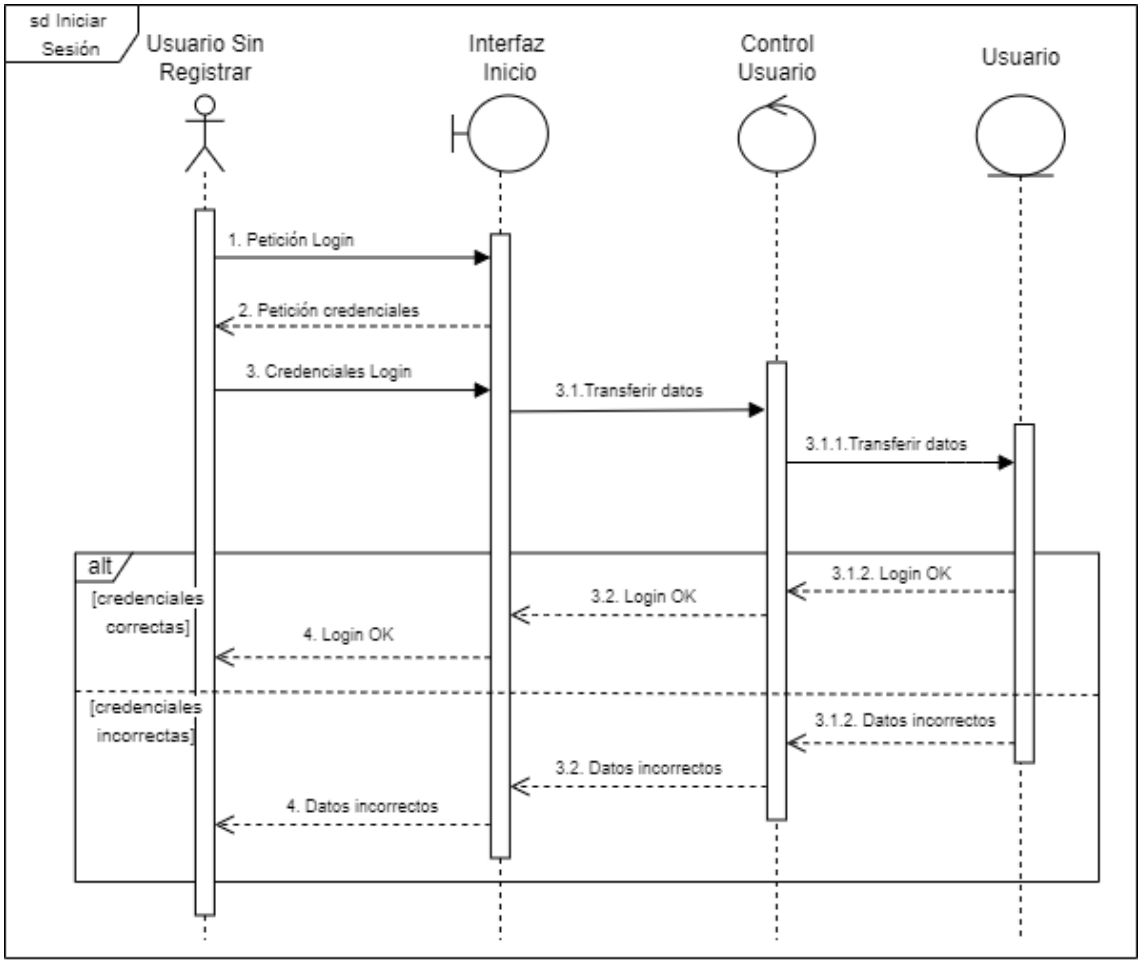


Figura 2. Diagrama de Secuencia – Iniciar Sesión

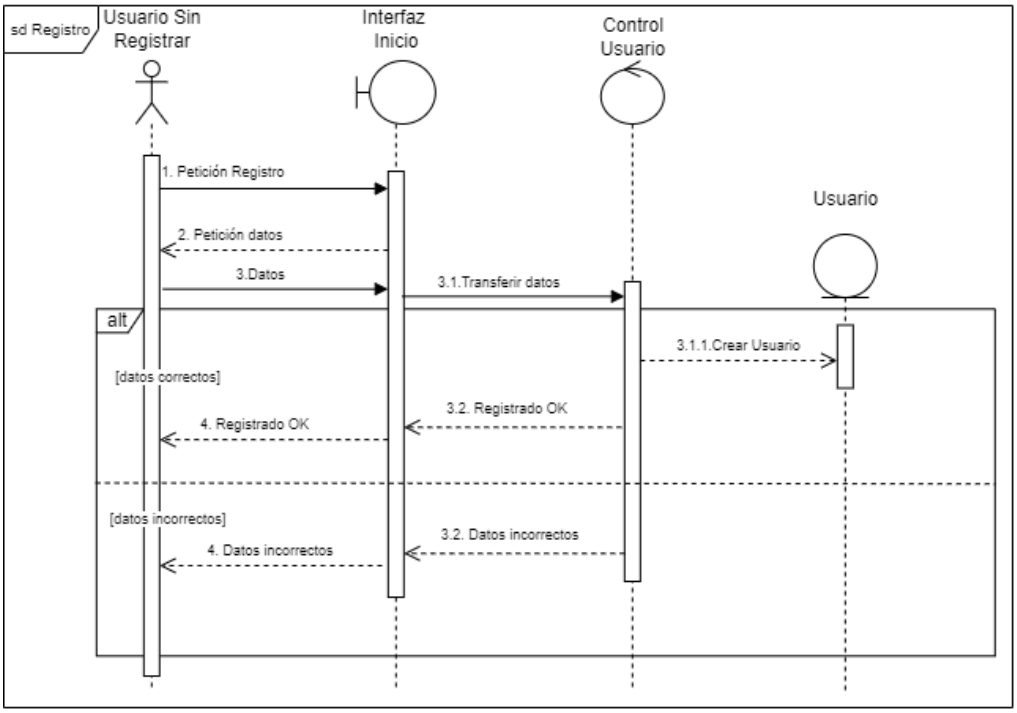


Figura 3. Diagrama de Secuencia – Registro

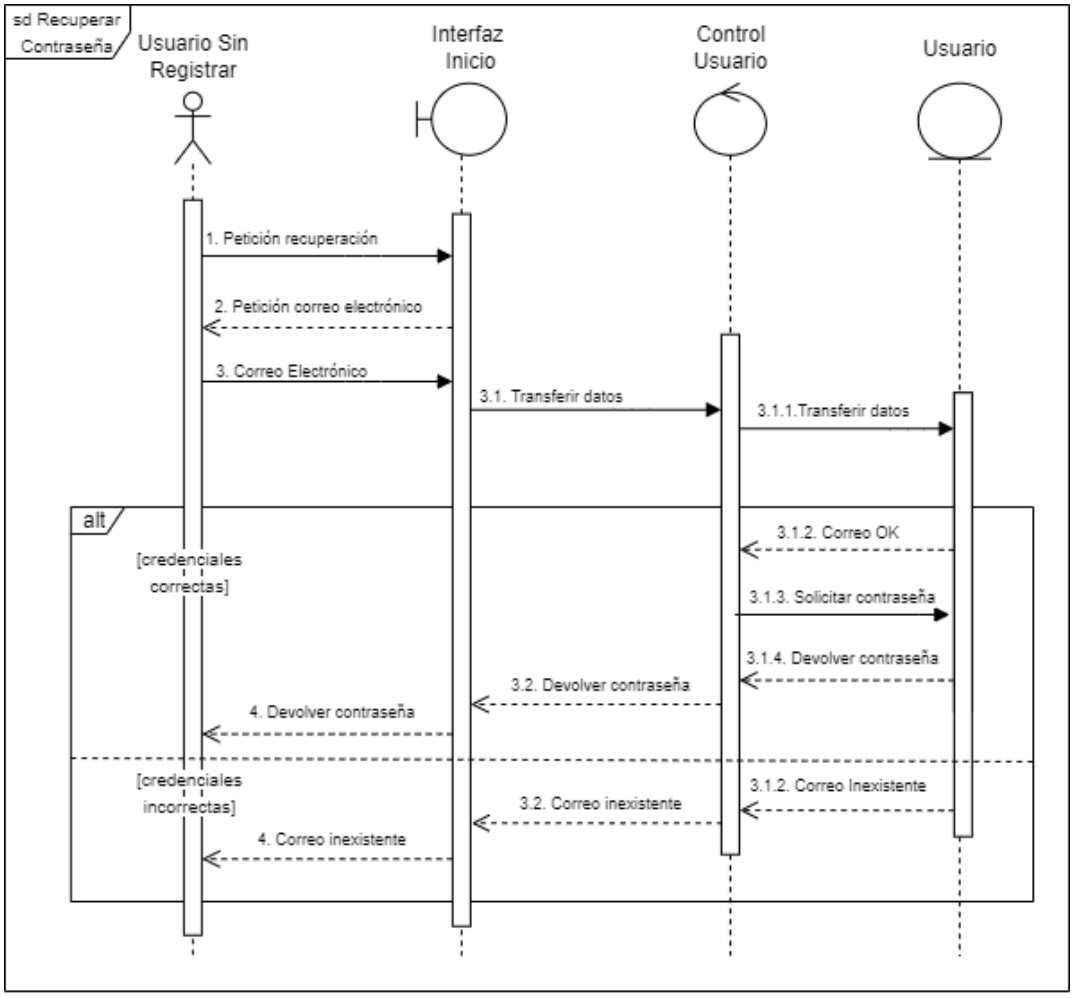


Figura 4. Diagrama de Secuencia – Recuperar Contraseña

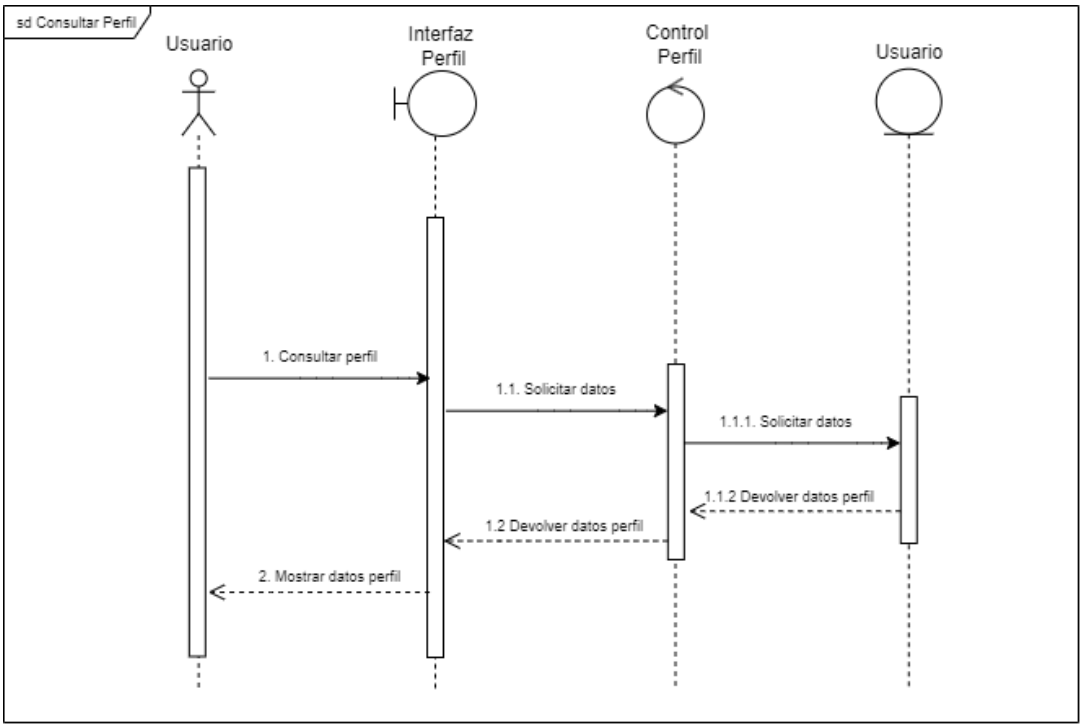


Figura 5. Diagrama de Secuencia – Consultar Perfil

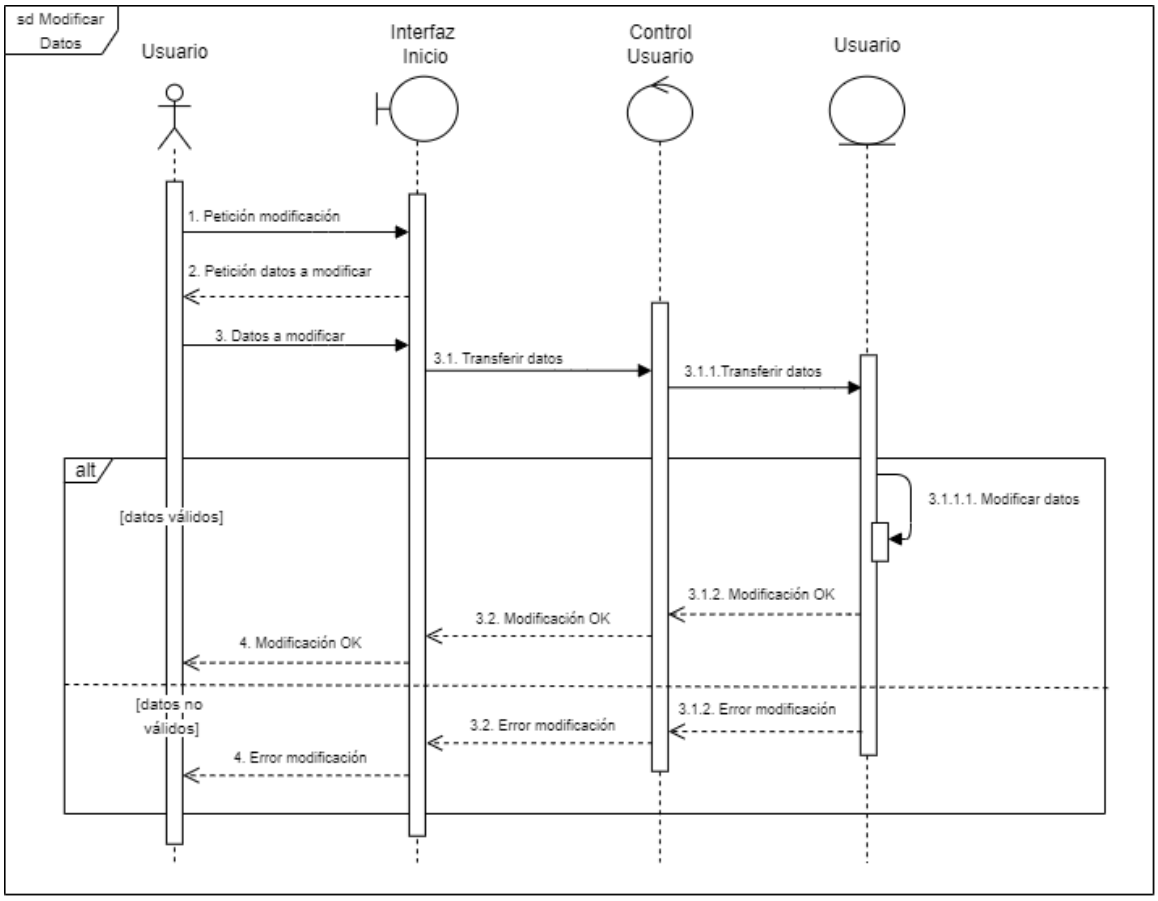


Figura 6. Diagrama de Secuencia – Modificar Datos

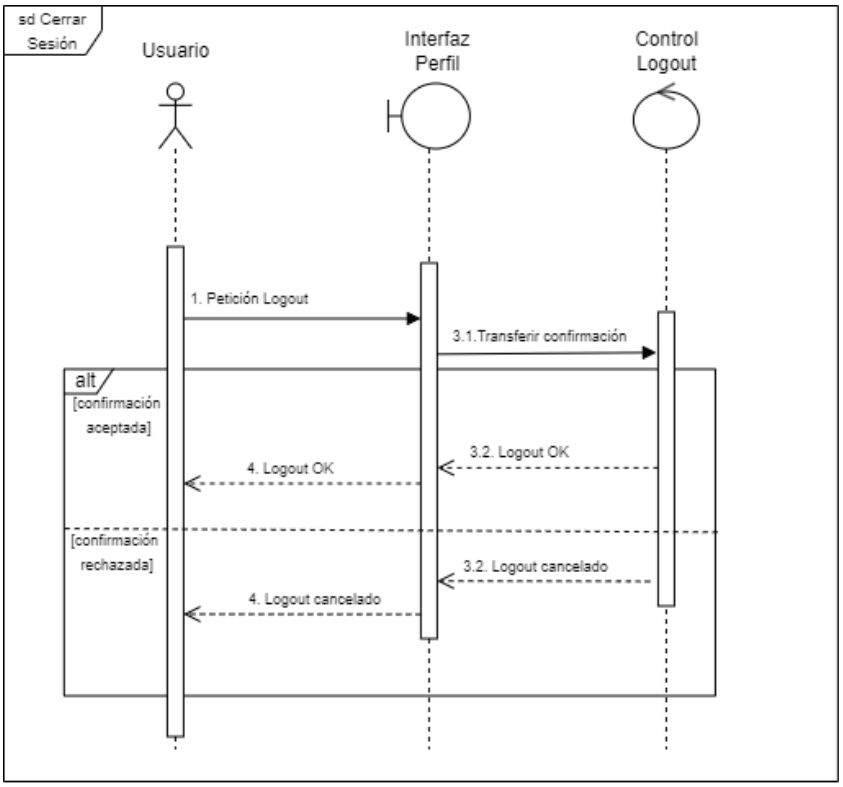


Figura 7. Diagrama de Secuencia – Cerrar Sesión

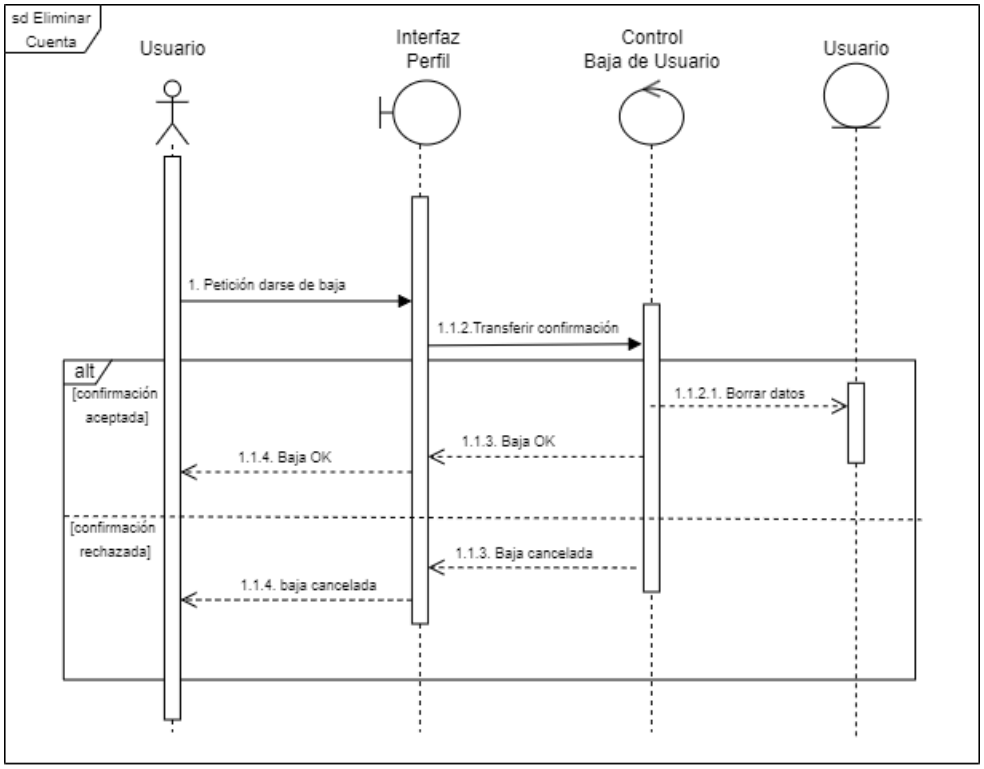


Figura 8. Diagrama de Secuencia – Eliminar Cuenta

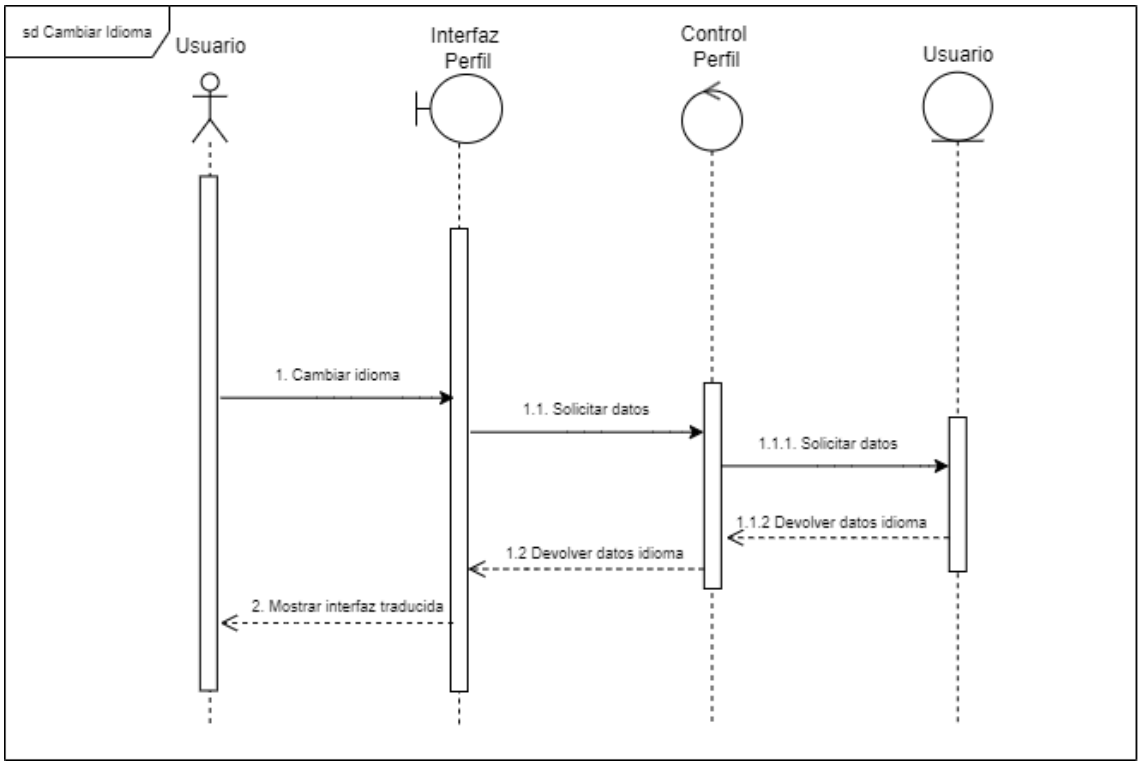


Figura 9. Diagrama de Secuencia - Cambiar Idioma

3.2. DIAGRAMAS DE SECUENCIA (GESTIÓN DE FUNCIONES SOCIALES)

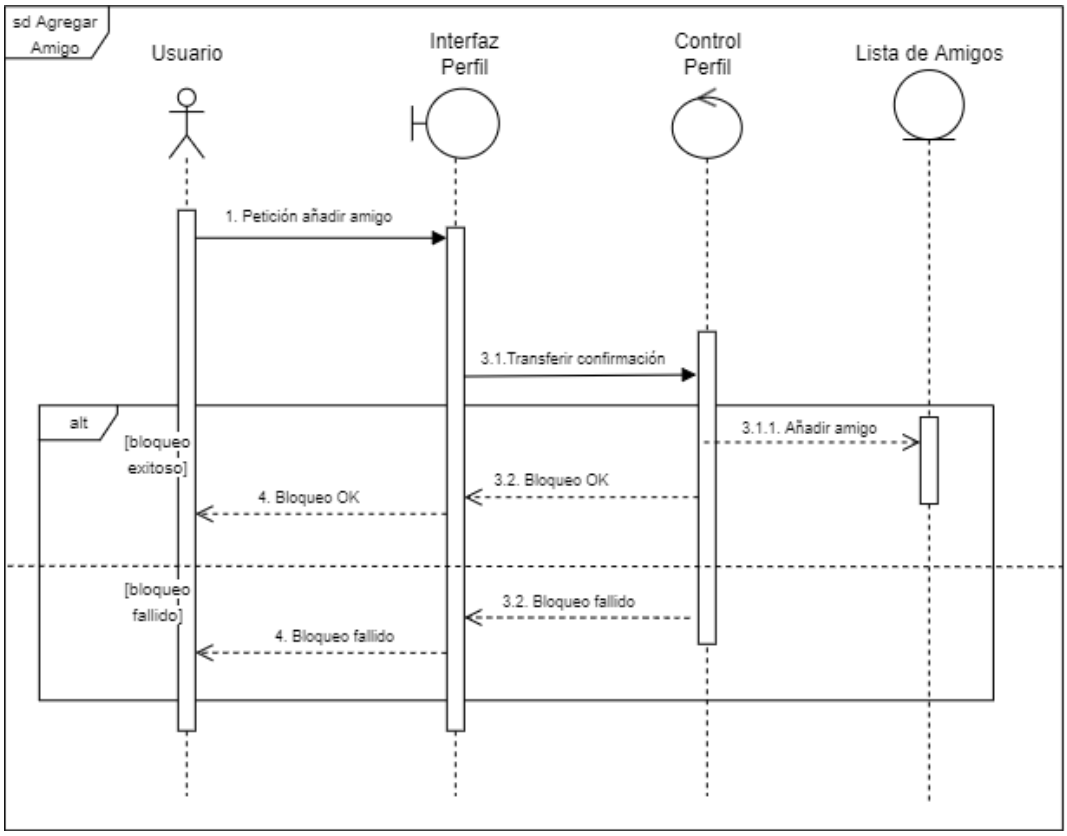


Figura 10. Diagrama de Secuencia – Agregar Amigo

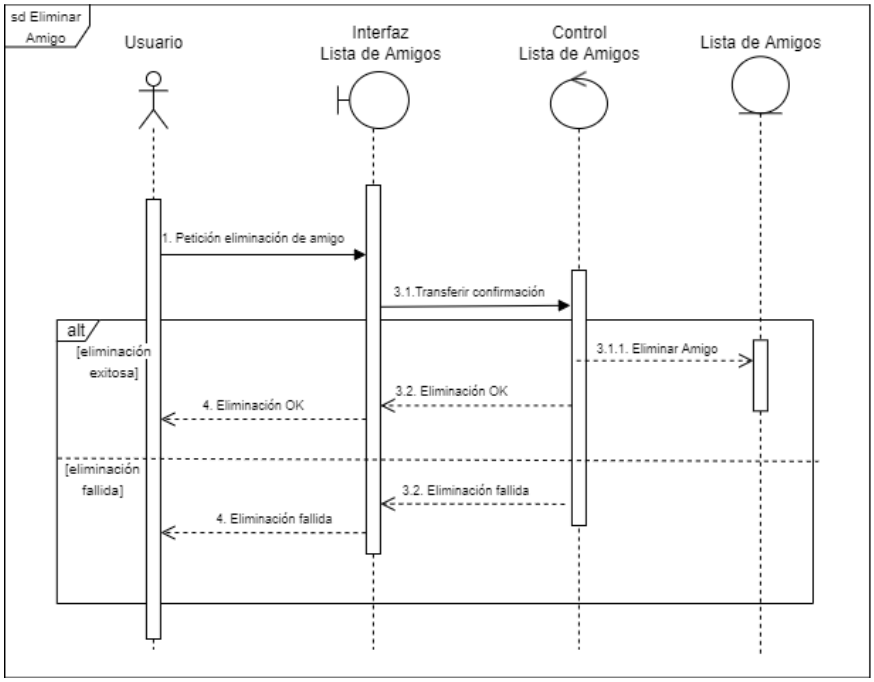


Figura 11. Diagrama de Secuencia – Eliminar Amigo

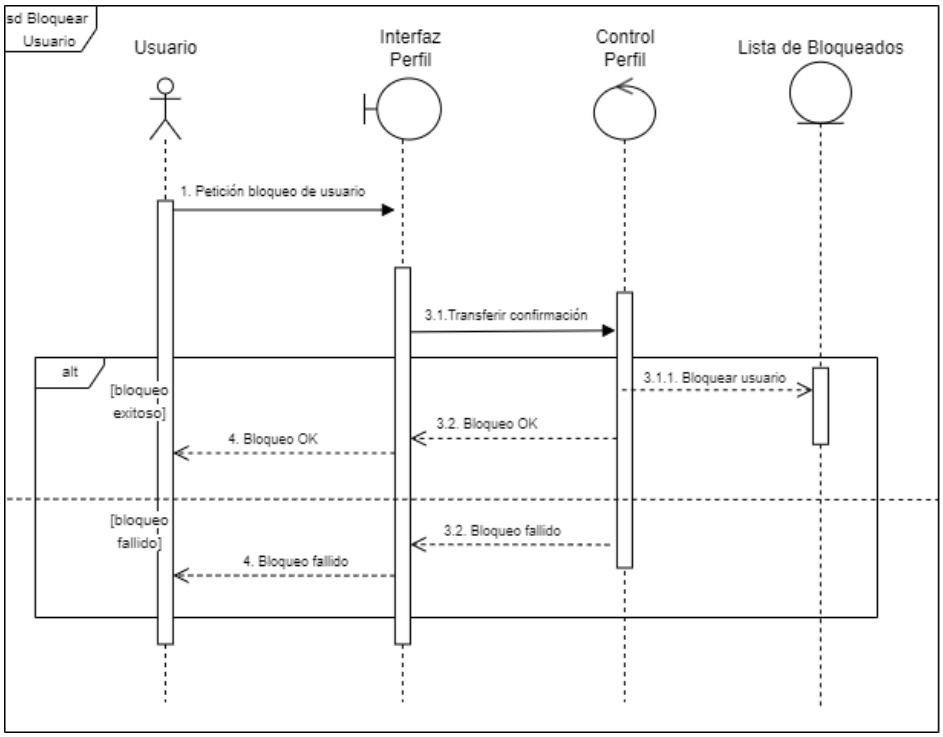


Figura 12. Diagrama de Secuencia – Bloquear Usuario

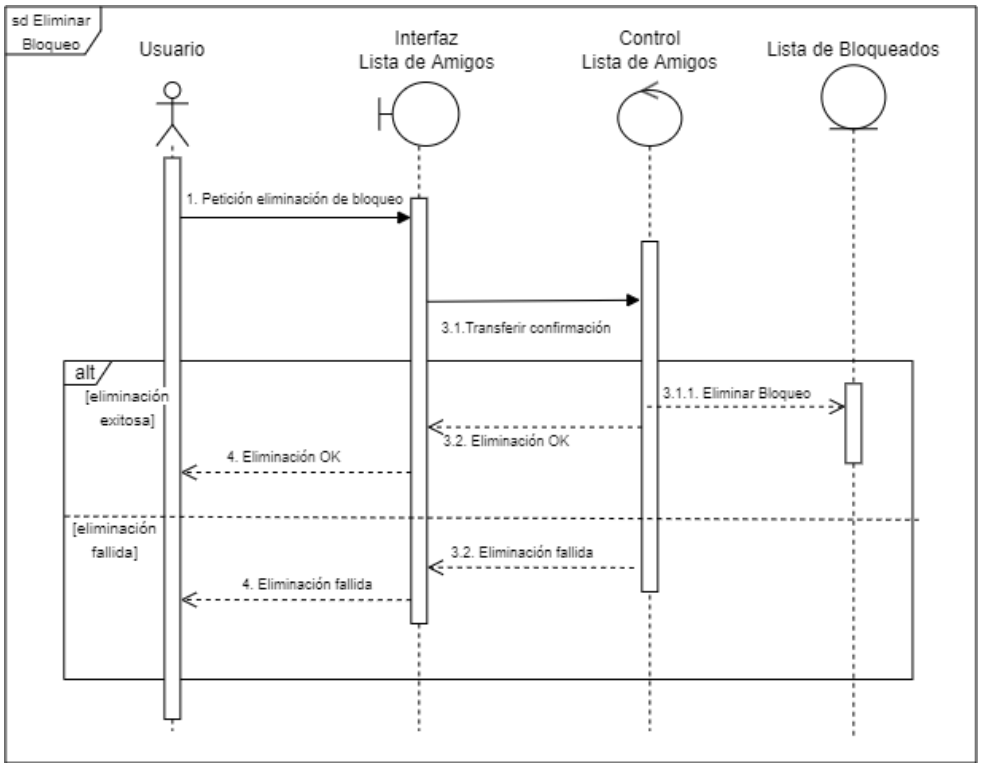


Figura 13. Eliminar Bloqueo

3.3. DIAGRAMAS DE SECUENCIA (GESTIÓN DE CATÁLOGO)

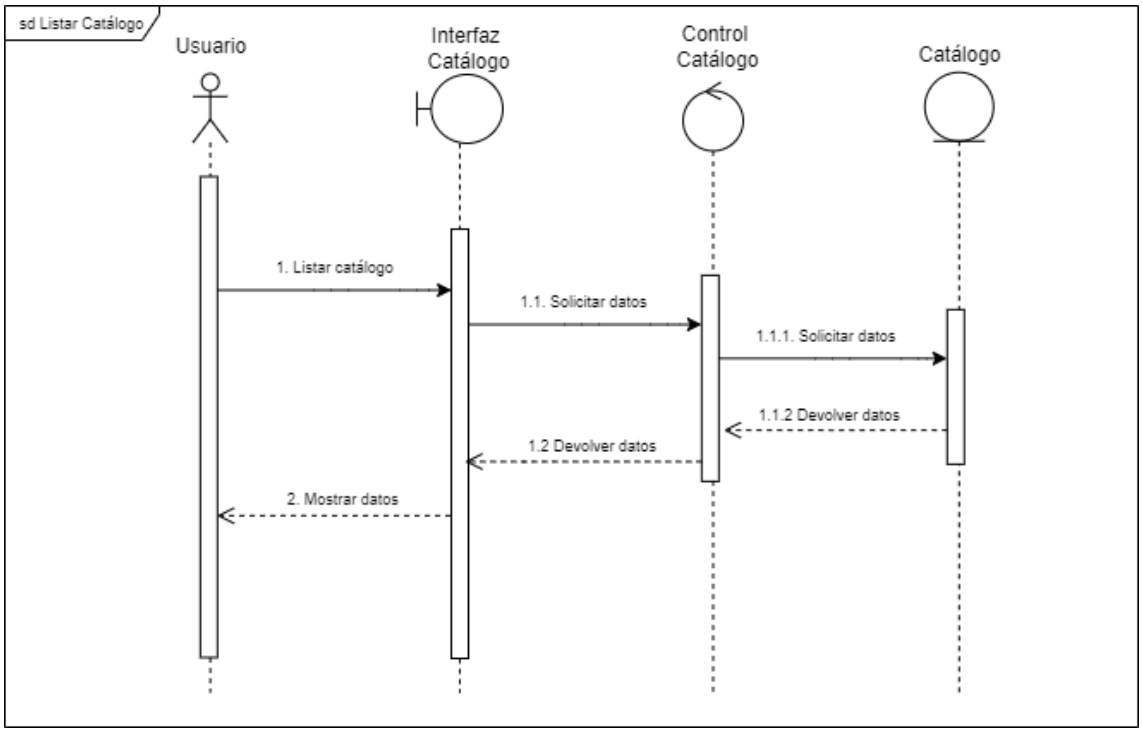


Figura 14. Diagrama de Secuencia – Listar Catálogo

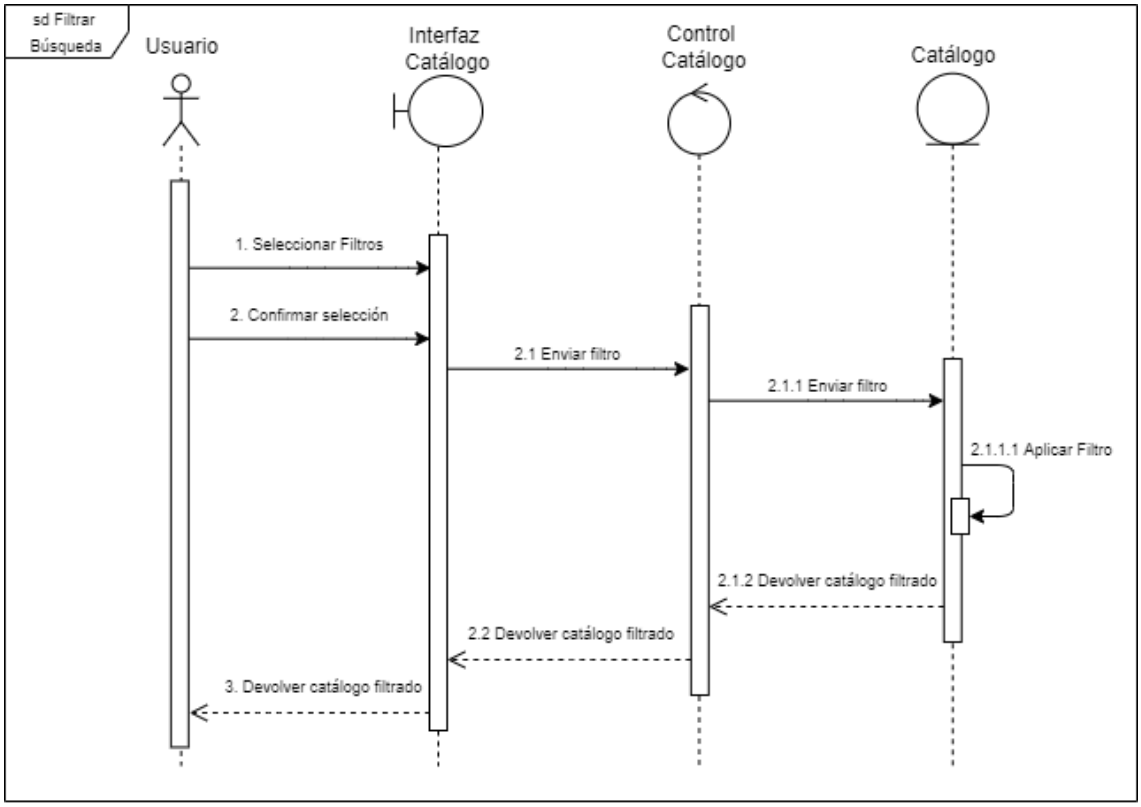


Figura 15. Diagrama de Secuencia – Filtrar Búsqueda

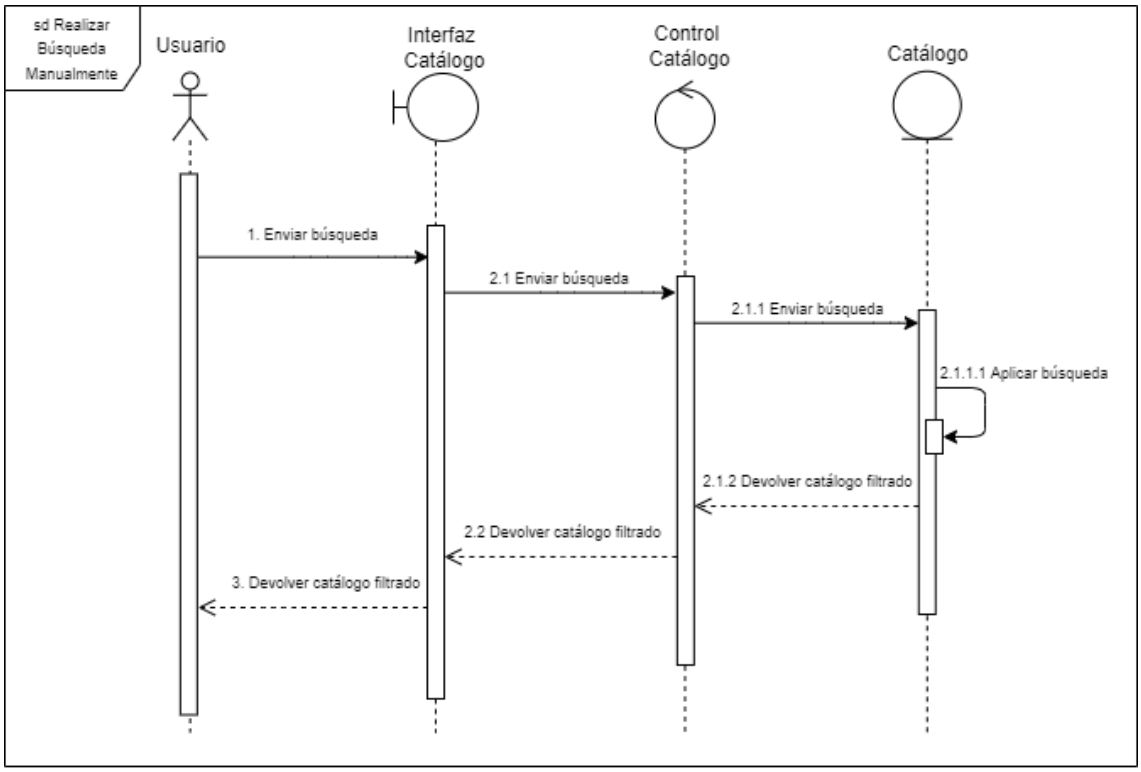


Figura 16. Diagrama de Secuencia – Realizar Búsqueda Manualmente

3.4. DIAGRAMAS DE SECUENCIA (GESTIÓN DE RETRANSMISIÓN)

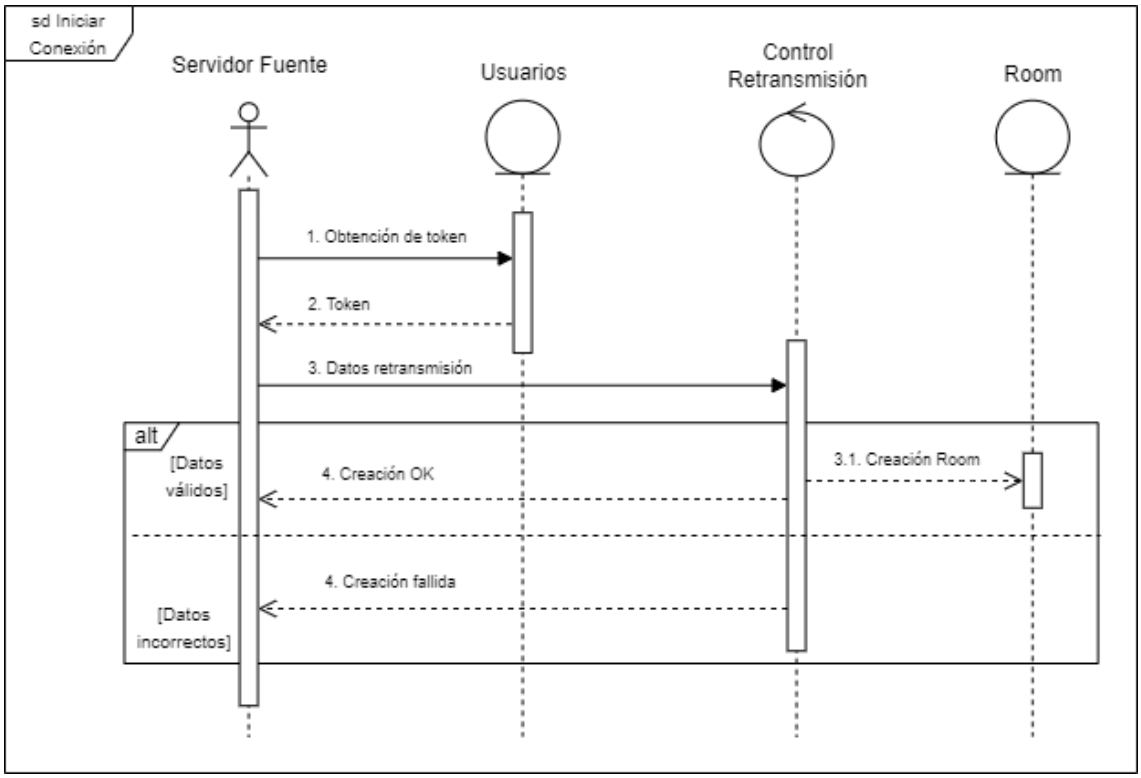


Figura 17. Diagrama de Secuencia – Iniciar Conexión

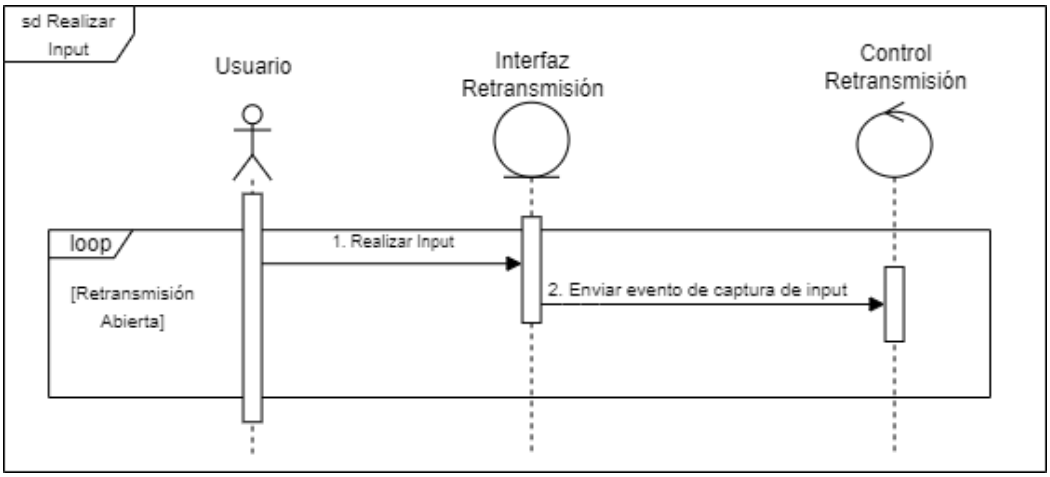


Figura 18. Diagrama de Secuencia – Realizar Input

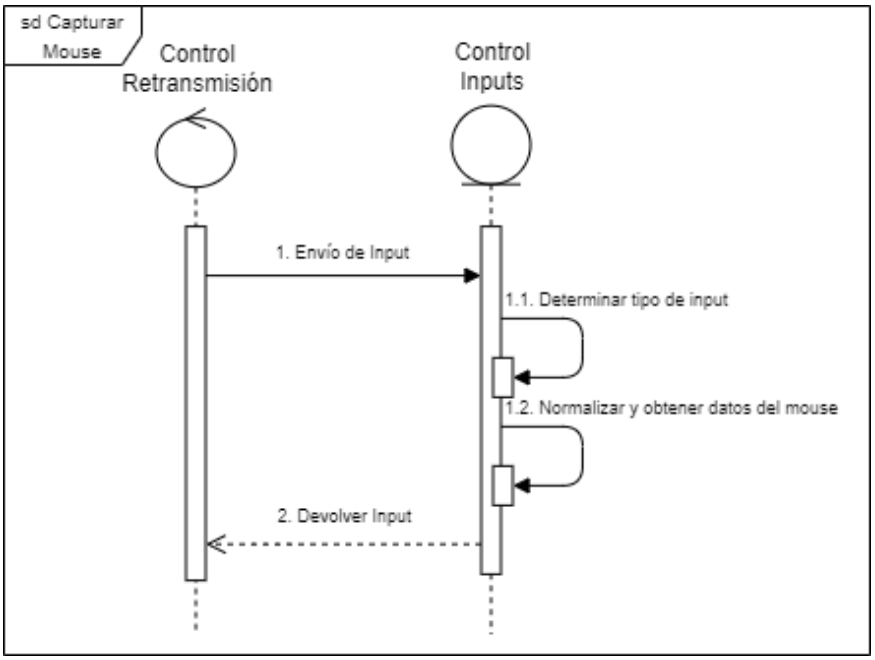


Figura 19. Diagrama de Secuencia – Capturar Mouse

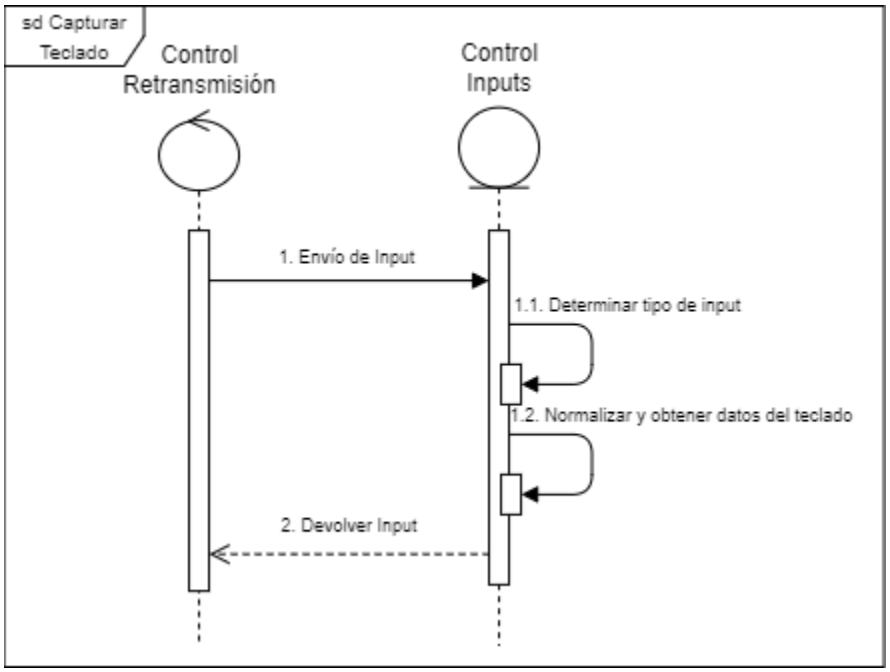


Figura 20. Diagrama de Secuencia – Capturar Teclado

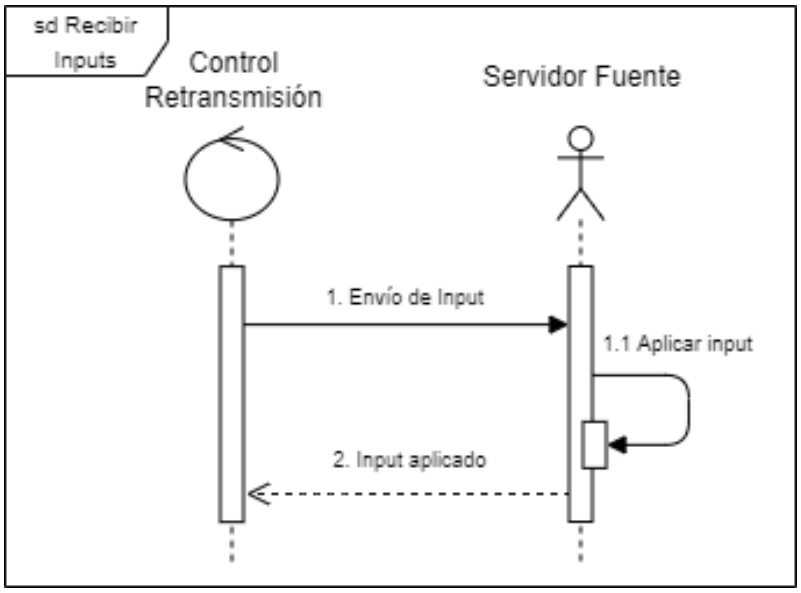


Figura 21. Diagrama de Secuencia – Recibir Inputs

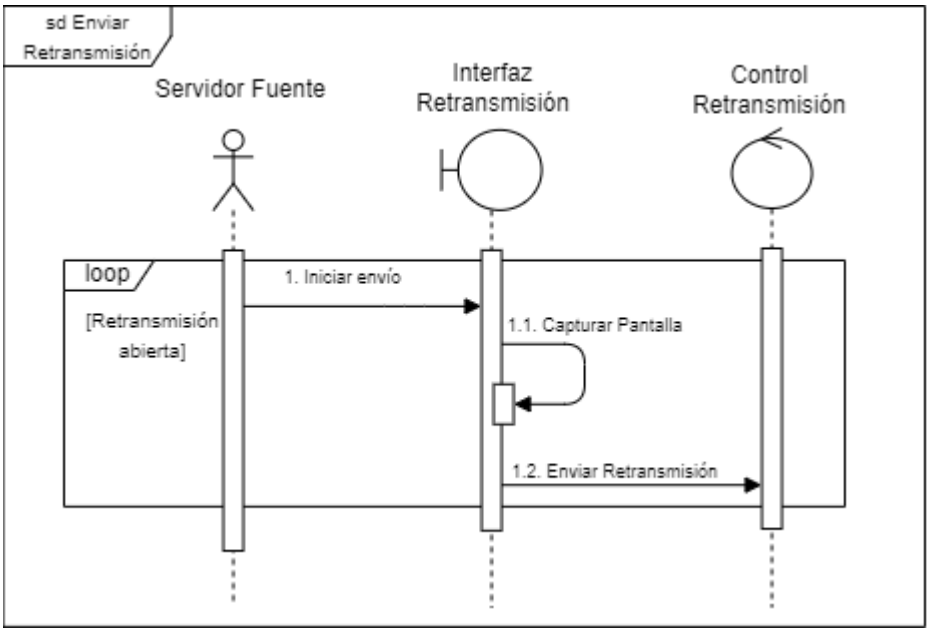


Figura 22. Diagrama de Secuencia – Enviar Retransmisión

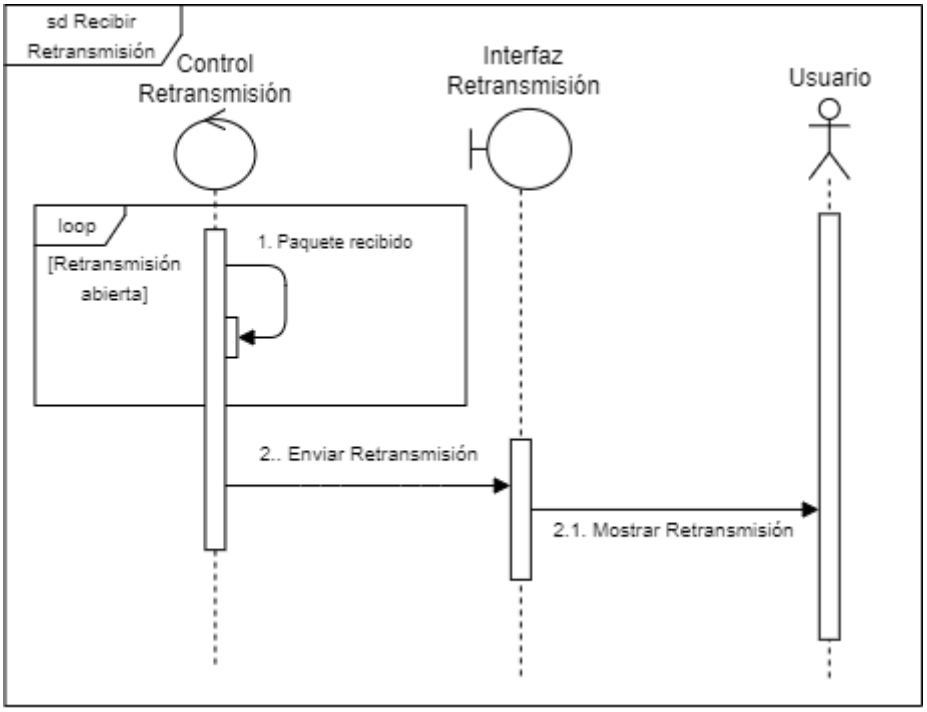


Figura 23. Diagrama de Secuencia – Recibir Retransmisión

4. CLASES DE ANÁLISIS

Mediante una división por paquetes, se exponen a continuación los diagramas de comunicación de las diferentes clases de análisis del sistema que han aparecido en los diagramas de secuencia del apartado anterior.

Las relaciones que se dan entre estos diagramas vienen dadas por la aparición de las mismas clases de análisis entre ellos.

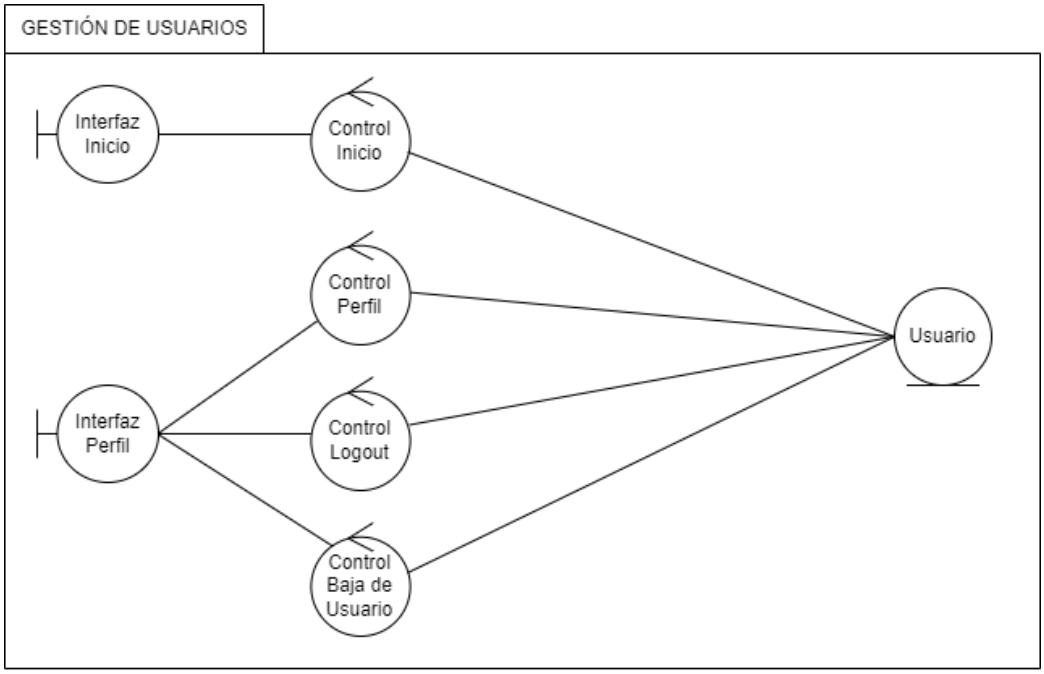


Figura 24. Clases de Análisis – Gestión de Usuarios

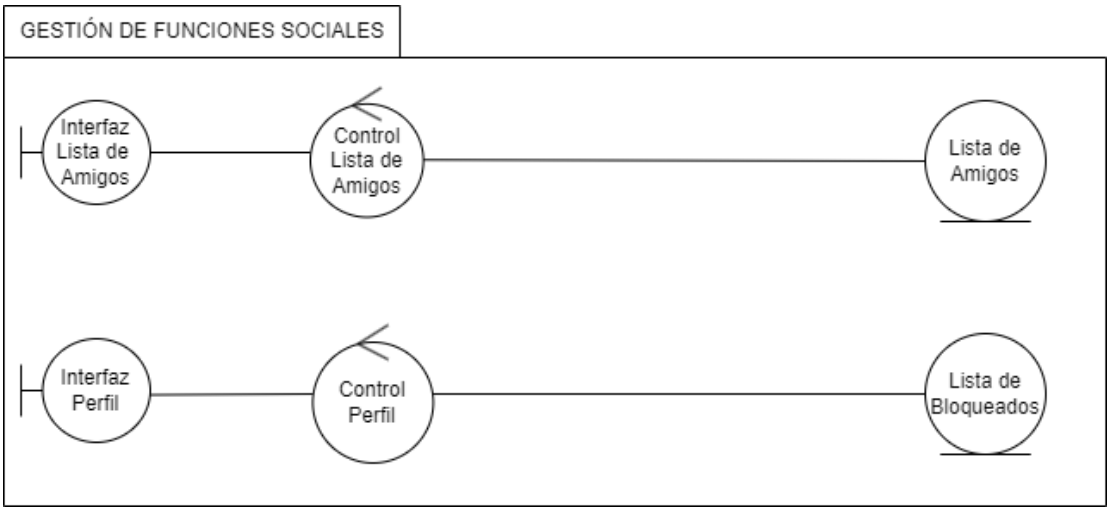


Figura 25. Clases de Análisis – Gestión de Funciones Sociales

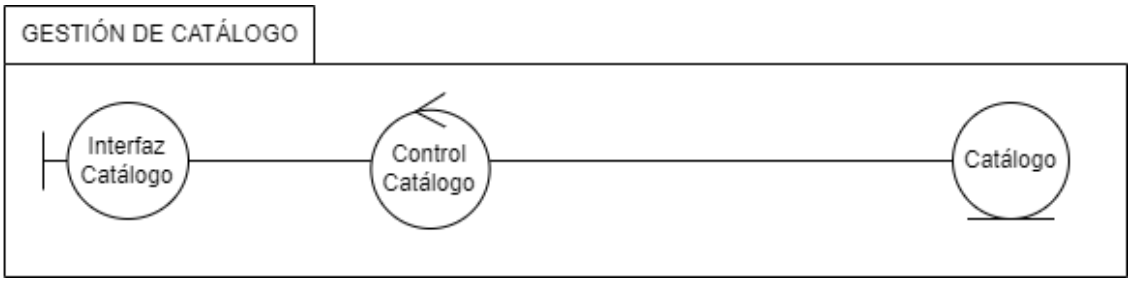


Figura 26. Clases de Análisis – Gestión de Catálogo

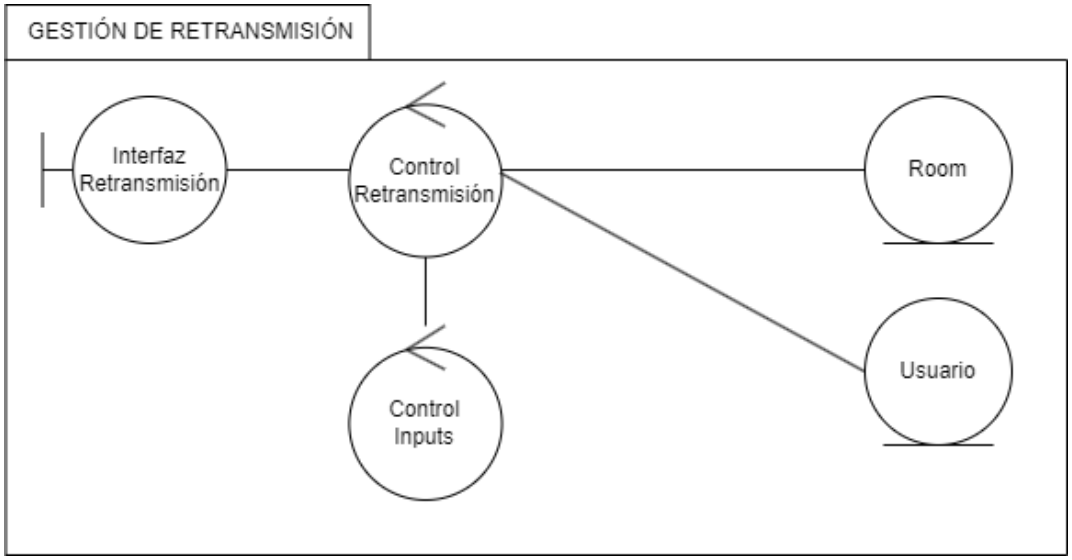


Figura 27. Clases de Análisis – Gestión de Retransmisión

5. MODELO DE ANÁLISIS – VISTA ARQUITECTÓNICA

Con el fin de facilitar la realización del Anexo IV, se presenta a continuación la vista arquitectónica del sistema, en la cual se categorizan las diferentes clases de análisis en las diferentes capas del Modelo Vista Controlador.

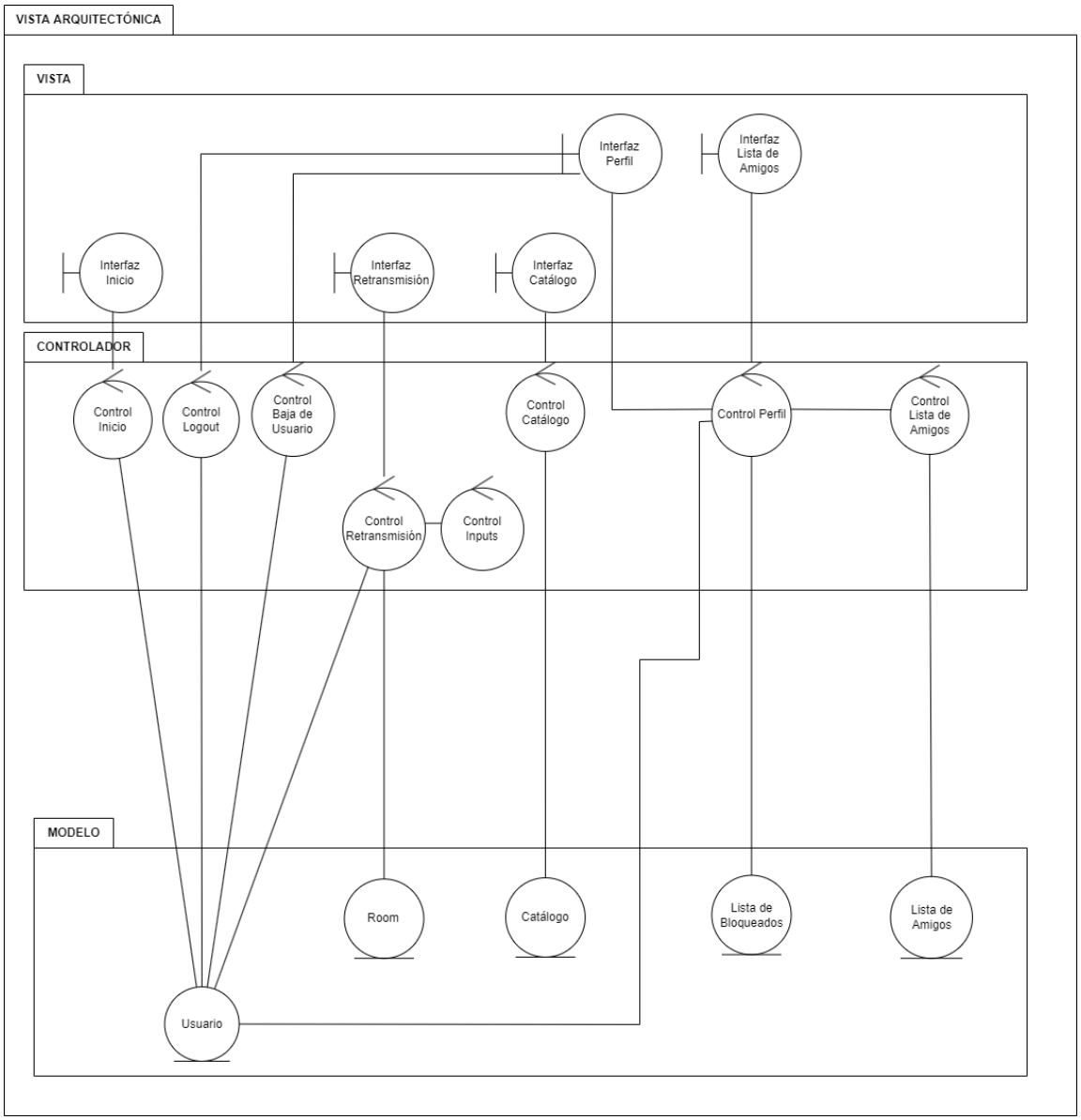


Figura 28. Vista Arquitectónica

6.CONCLUSIONES

Tras el desarrollo de la etapa de análisis se tiene ahora una visión mucho más clara del sistema, esto se traducirá en una mayor facilidad a la hora de comenzar la etapa de diseño en el siguiente anexo.

Las conclusiones obtenidas de este anexo son varias, principalmente: observando el modelo de dominio se puede comprobar lo que ya se afirmaba en el pasado anexo: se trata de un sistema con pocos elementos persistentes en él, estos se limitan a los propios usuarios y a los datos de los videojuegos del catálogo, ni siquiera la retransmisión se puede considerar persistente pues se trata de una clase de análisis generada especialmente para cada dupla cliente-servidor. Además, los diagramas de secuencia proporcionan una primera visión del intercambio de mensajes del sistema, es aquí donde se empieza a ver la complejidad de algunos de estos Requisitos y el motivo por el que el sistema realmente no tiene tantos Casos de Uso. Esto se acentuará aún más en la etapa del dominio de la solución, cuando veamos con exactitud estos mismos diagramas de secuencia adaptados al diseño.

7. BIBLIOGRAFÍA

García Peñalvo, F. J., Moreno García, M. N., García Holgado, A., & Vázquez Ingelmo, A. (2019-2020). *INGENIERíA DEL SOFTWARE I. "Tema 8 - UML. Modified Modeling Language".* Obtenido de https://moodle2.usal.es/pluginfile.php/996986/mod_resource/content/4/IS_I%20Tema%208%20-%20UML.pdf

ANEXO IV: Diseño del Sistema Software.

Plataforma streaming multidispositivo de videojuegos

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Agosto de 2022

Autor:

Jesús Manuel García Prieto

Tutores:

Diego J. Valdeolmillos Villaverde

Alfonso González Briones

Francisco Pinto Santos

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

A diferencia de anexos anteriores, los cuales pertenecían a la fase de análisis en el dominio del problema, el "Anexo IV: Diseño del sistema software" se encuentra en la fase de diseño, la cual se centra en el dominio de la solución. Es por esto que este anexo se encuentra en un punto mucho más cercano a la implementación final del proyecto, debido a esto, los ficheros, métodos, atributos y demás componentes que aparezcan serán los mismos que en el propio sistema o aproximaciones muy cercanas.

La estructura que se seguirá a lo largo de este anexo es la siguiente:

- **Modelo de Diseño:** El cual se divide en las siguientes secciones
    - **Patrones y estilos arquitectónicos:** Se describirán aquellos patrones y estilos utilizados en el desarrollo del proyecto.
    - **Subsistemas de diseño:** Se organizará el sistema en paquetes mostrando las conexiones entre estos.
    - **Clases de diseño:** Se describirá el contenido de cada clase, así como sus relaciones con otras.
    - **Vista arquitectónica:** Vista por capas del modelo MVVM definido en los apartados anteriores.
    - **Realización de Casos de Uso – Diseño:** Se describirá, en el dominio del diseño, los diferentes Casos de Uso mediante el uso de diagramas de secuencia una vez más.
- **Diseño de la Base de Datos:** Se describirá el contenido de la Base de Datos en forma de diagrama, así como las relaciones entre sus elementos.
- **Modelo de Despliegue:** Se mostrarán en forma de componentes las diferentes capas físicas del sistema.

2. MODELO DE DISEÑO

En esta etapa del sistema se hablará sobre los diferentes patrones arquitectónicos utilizados en el mismo, destacando entre ellos, el modelo cliente-servidor que será el que caracterice al sistema mayoritariamente.

2.1. PATRONES Y ESTILOS ARQUITECTÓNICOS

2.2.1. ARQUITECTURA CLIENTE-SERVIDOR

Se trata del estilo arquitectónico que definirá al sistema, se trata de uno de los estilos más conocidos y utilizados, su principal característica es su clara división del sistema en un cliente y un servidor o proveedor. El cliente será el encargado de solicitar servicios y recursos al proveedor y, posteriormente, consumirlos (ReactiveProgramming, 2022).

La conexión entre ambos, en este caso, viene dada por el protocolo SSH, encargado de proporcionar un canal de comunicación fiable.

Otra de sus características fundamentales viene dada por la definición de sistema distribuido, ya que el cliente y el servidor se encuentran distribuidos en diferentes equipos comunicados por mediante una red.

Gracias a este estilo arquitectónico se facilita enormemente la división del sistema, evitando así la centralización del mismo y pudiendo separar los datos (o el modelo del sistema) de la vista del mismo.

Este esquema (adaptado a Vue) se muestra en la **Figura 1:** (FairCom, 2022)

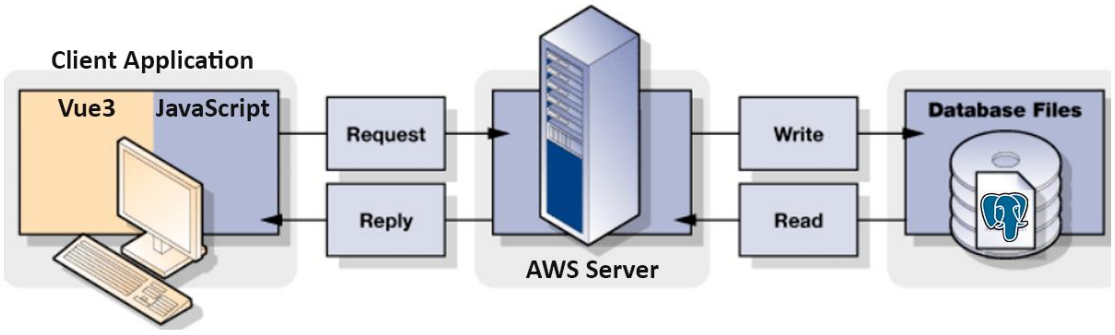


Figura 1. Arquitectura cliente-servidor

2.2.1. PATRÓN MVVM (MODELO-VISTA-VISTA-MODELO)

Debido a la utilización de Vue3 en el sistema, este se encuentra comprometido con la utilización del patrón MVVM, ya que Vue se especializa en implementar este patrón con facilidad.

Antes de explicar cómo Vue logra implementarlo, se procede a explicar en qué consiste este patrón tan extendido:

Para entender cómo funciona el patrón MVVM vamos a compararlo con su contraparte más popular: el patrón MVC, ambos patrones buscan el lograr dividir la aplicación o el sistema en diferentes capas, para facilitar así su mantenimiento, desarrollo o implementación.

La principal diferencia entre ellos es que el MVVM elimina toda lógica de la Vista, y la Vista-Modelo separa totalmente la Vista del Modelo, a diferencia del patrón MVC en el cual las 3 capas se encuentran interconectadas, como se puede observar en la **Figura 2:** (Zuev, 2020)

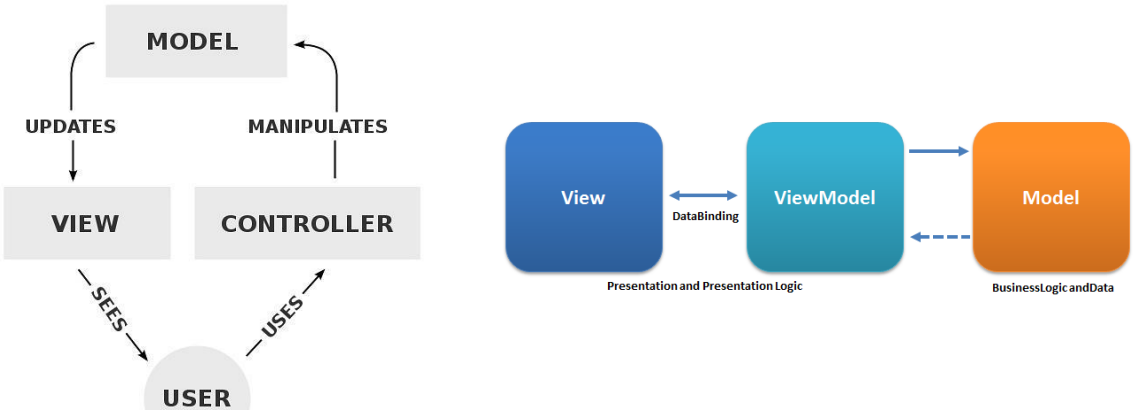


Figura 2. Estructura de los modelos MVC y MVVM

Sabiendo esto, se verá a continuación cómo Vue implementa este patrón (akin-ogundenji, 2015)*:*

- El Modelo en Vue se trata simplemente de los datos en crudo que existan en el sistema o los que se obtengan del servidor. El Modelo no posee ningún tipo de lógica ni comportamiento definidos, más allá de la validación de datos. Además de esto, no posee ningún tipo de forma de acceso al backend o a alguna API para obtener estos datos, simplemente funciona como un contenedor de datos que serán utilizados por el VM.
- La Vista se encarga de renderizar el contenido del Modelo de cara al usuario, sin embargo, la Vista no sabe nada acerca del Modelo y viceversa, la comunicación entre ambas se produce a través de la VM, de la cual, si tiene constancia la Vista, convirtiéndola así en una "Vista activa". Su único fin más allá de mostrar datos al usuario es el de interceptar inputs del usuario y enviarlos al VM.
- La Vista Modelo se trata del enlace entre la Vista y el Modelo, toda la lógica requerida para manipular la información contenida en el Modelo se encuentra en la VM y toda la lógica usada por la Vista para tratar o formatear los datos de usuario viene definida por la VM. A diferencia de otros patrones, toda la lógica de negocio del sistema se encuentra acoplada en la VM, en lugar de encontrarse en el Modelo.

En la **Figura 3** (ProgrammerClick, 2020-2022) podemos observar la implementación de este patrón de diseño en Vue.

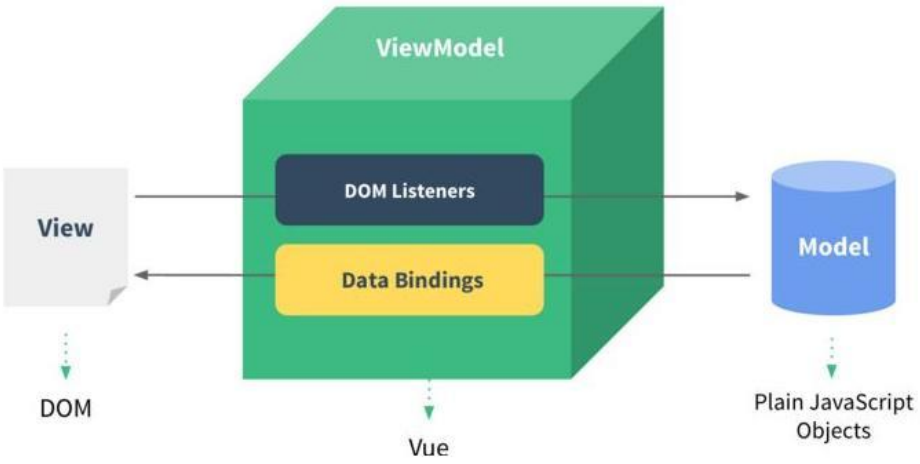


Figura 3. Patrón MVVM en Vue

2.2. SUBSISTEMAS DE DISEÑO

Es en este punto en el que se presentará la división del sistema en una serie de subpaquetes más específicos, concretamente:

- **En el subpaquete AplicaciónCliente** tenemos todo lo relativo al usuario, esto, por supuesto, incluye sus interfaces recogidas en VistaCliente, así como los datos usados en las mismas, recogidos en el ModeloCliente. Como en todo patrón MVVM solo queda por hablar de la VistaModeloCliente, que interconecta estos dos paquetes. A su vez, encontramos el paquete MóduloRetransmisión, que controla toda la lógica relacionada con recibir el vídeo y enviar los inputs al servidor.
- **El subpaquete AplicaciónFuente** posee la misma división que el paquete AplicaciónCliente, con sutiles diferencias a bajo nivel, entre ellas se destaca el hecho de que la Fuente se encarga de enviar imagen de vídeo y de recibir inputs, lo opuesto al cliente. Además, posee pequeños cambios en su Vista.
- Por último, **el subpaquete AplicaciónServidor** recoge toda la lógica implicada en el servidor, como por ejemplo la BBDD PostgreSQL o los handlers de la retransmisión y recepción y envío de inputs.

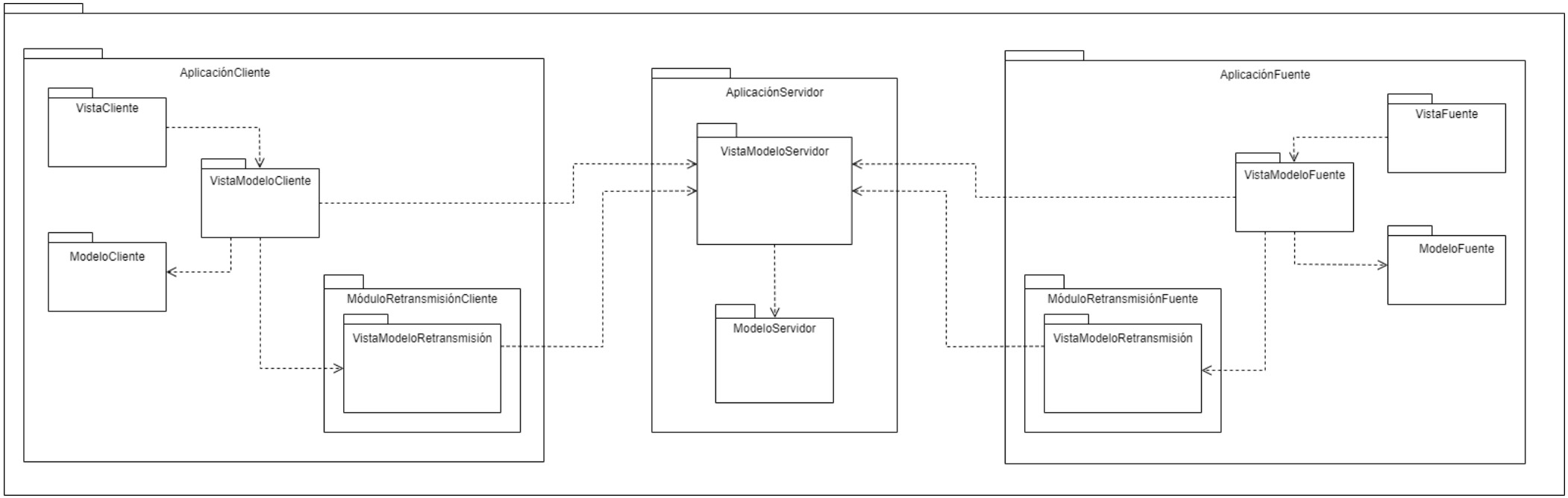


Figura 4. Modelo de dominio

2.3. CLASES DE DISEÑO

En este apartado se especificará el contenido de los diferentes subpaquetes descritos en el apartado anterior en función del patrón MVVM, registrando así las clases y métodos de estos.

Como ya se sabe, el framework de Vue integra JavaScript junto a HTML y CSS, principalmente, es por esto que muchos de los archivos en Vue albergan a estos 3 lenguajes simultáneamente, es por este motivo que en muchos de las clases de diseño veremos al mismo archivo siendo mencionado, sin embargo, serán diferentes secciones del mismo las que serán referenciadas. Por suerte, Vue se encarga ya por su cuenta de dividir estos archivos de manera clara, teniendo la parte de JavaScript dentro de la etiqueta <Script>, entre otras facilidades que nos otorga.

Para lograr distinguir estas secciones dentro de un mismo archivo ". vue" este será referido de la siguiente manera: "nombreArchivo" + "Capa modelo MVVM" + ". vue".

2.3.1. APLICACIÓN CLIENTE

2.3.1.1. VISTA CLIENTE

Como se aclaró previamente, se hará referencia a aquellos archivos (relacionados con el cliente) que posean una parte de Vista dentro del patrón MVVM, a pesar de que dentro de este mismo archivo puedan existir otras partes del patrón MVVM. Esto se puede observar en la **Figura 5:**

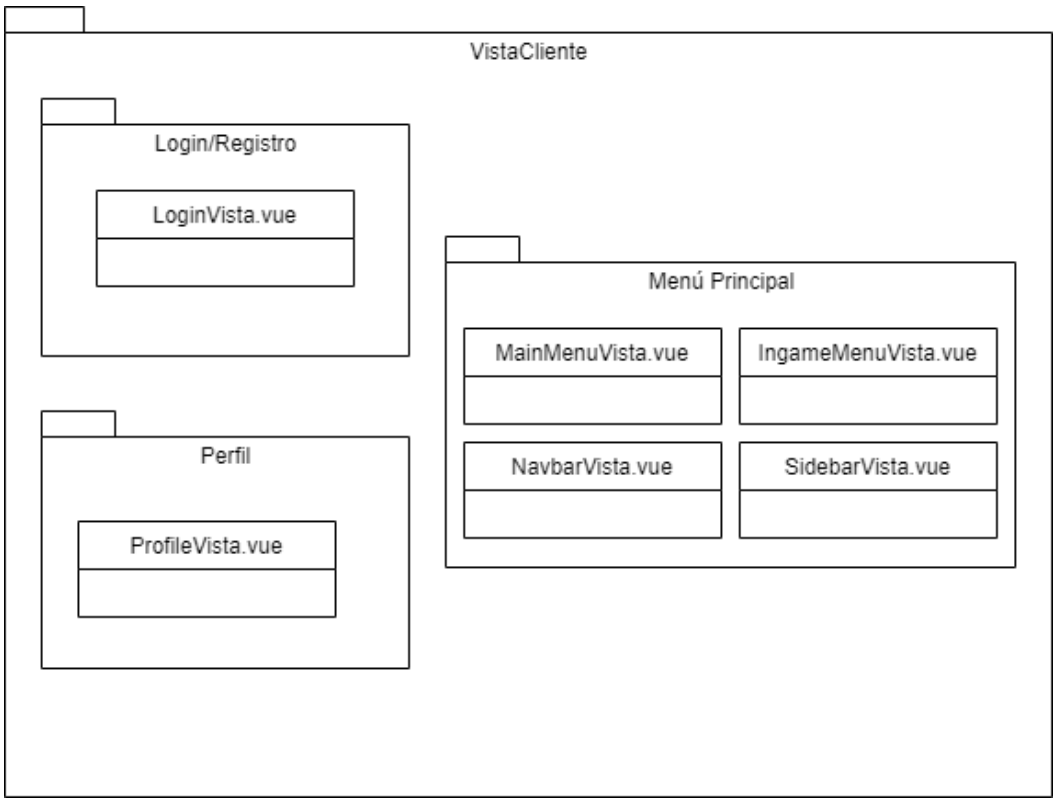


Figura 5. AplicaciónCliente - VistaCliente

2.3.1.2. MODELO CLIENTE

En el diagrama de la **Figura 6** veremos aquellos objetos de JavaScript que existan en el marco del usuario.

Como ya sabemos, los elementos del Modelo en el patrón MVVM no tienen funcionalidad alguna, por lo que serán representados únicamente por su nombre y sus datos.

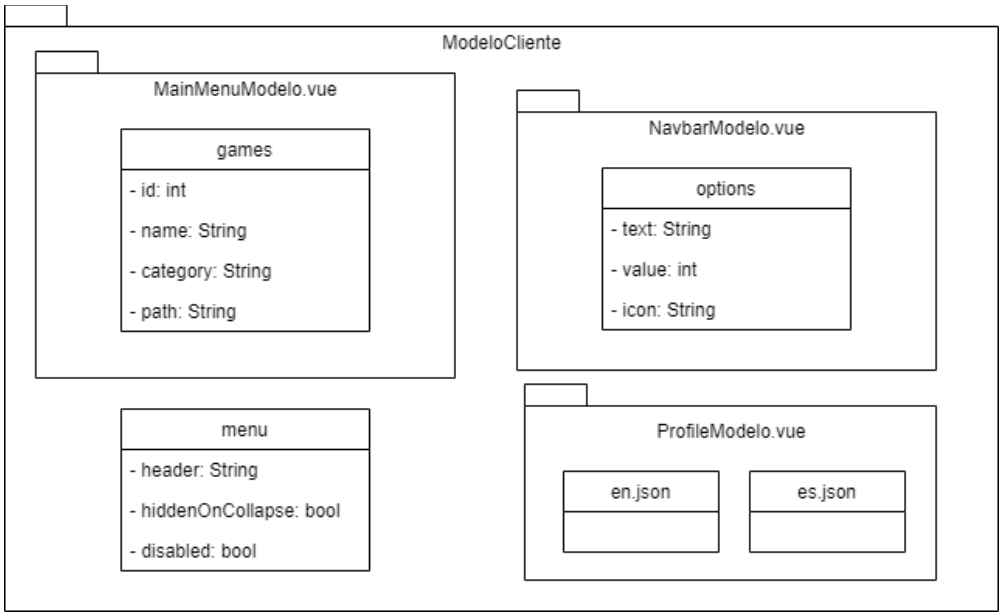


Figura 6. AplicaciónCliente - ModeloCliente

2.3.1.3. VISTA-MODELO CLIENTE

Debido a que muchos de los métodos utilizados en el sistema provienen de eventos emitidos en otros archivos, en lugar de catalogar la VistaModelo en función de en qué archivo se ejecuta la función, se catalogará en cuanto a su funcionalidad. Por ejemplo, cómo podemos observar en la **Figura 7** el método showFriends() es ejecutado en App.vue, sin embargo, este tiene que ver con la barra de navegación superior, por tanto se catalogará en una sección relativa a esta.

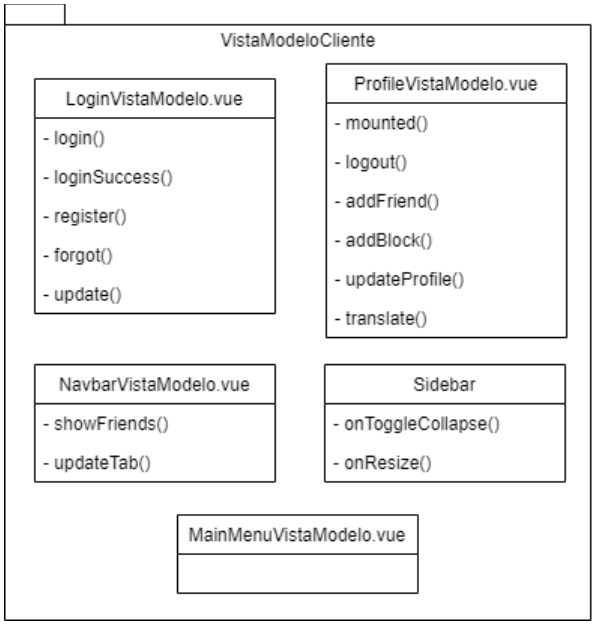


Figura 7. AplicaciónCliente - VistaModeloCliente

2.3.1.4. MÓDULO RETRANSMISIÓN CLIENTE

Tal y como funciona el sistema, este módulo de retransmisión se trata exactamente del mismo módulo que el que se encuentra en la fuente emisora, sin embargo, realiza unas funciones concretas para el cliente y otras para la fuente, es por esto que en el diagrama de la **Figura 8** se incluirán aquellas funcionalidades relacionadas con el cliente y no se incluirán aquellas respectivas a la fuente.

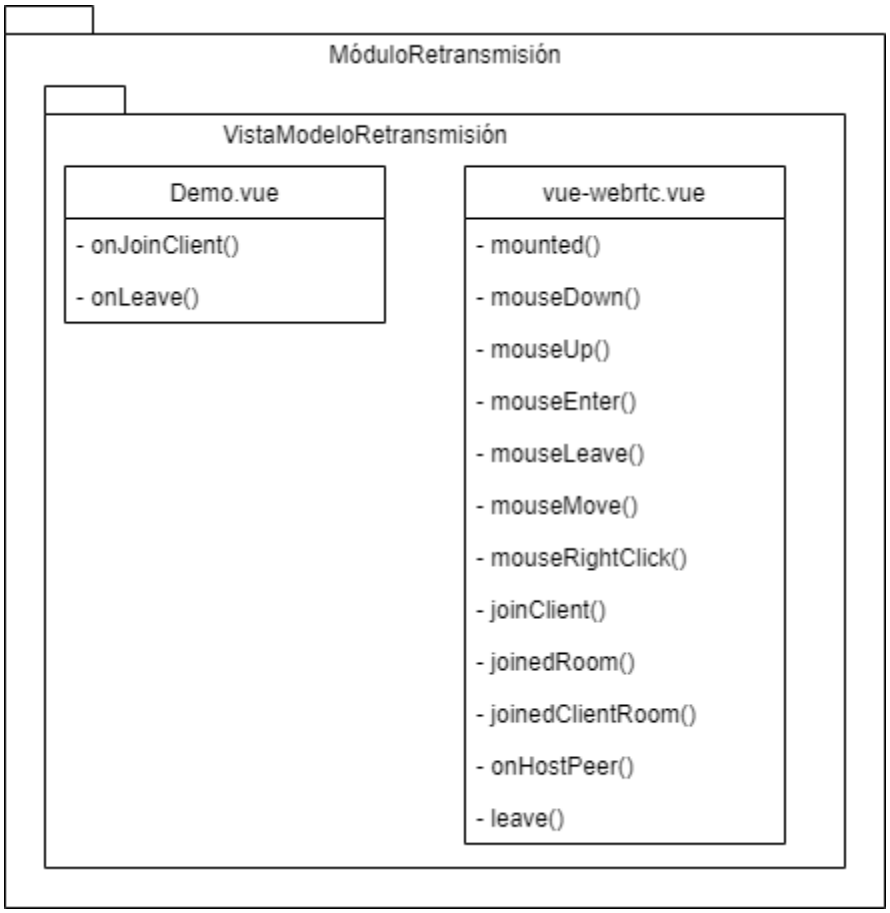


Figura 8. AplicaciónCliente - MóduloRetransmisiónCliente

2.3.2. APLICACIÓN FUENTE

Finalmente llegamos a la aplicación fuente, esta se basa en su totalidad en la aplicación cliente, su funcionalidad varía gracias al uso de cláusulas condicionales y otros diversos métodos, es por esto que sus clases de diseño son las mismas solo que eliminando algunas que se consideran innecesarias, por ejemplo, el perfil, lista de amigos o captura de inputs.

2.3.2.1. VISTA FUENTE

Como se puede observar en la **Figura 9**, este diagrama se compone únicamente de las subclases relativas al Login y al Menú Principal.

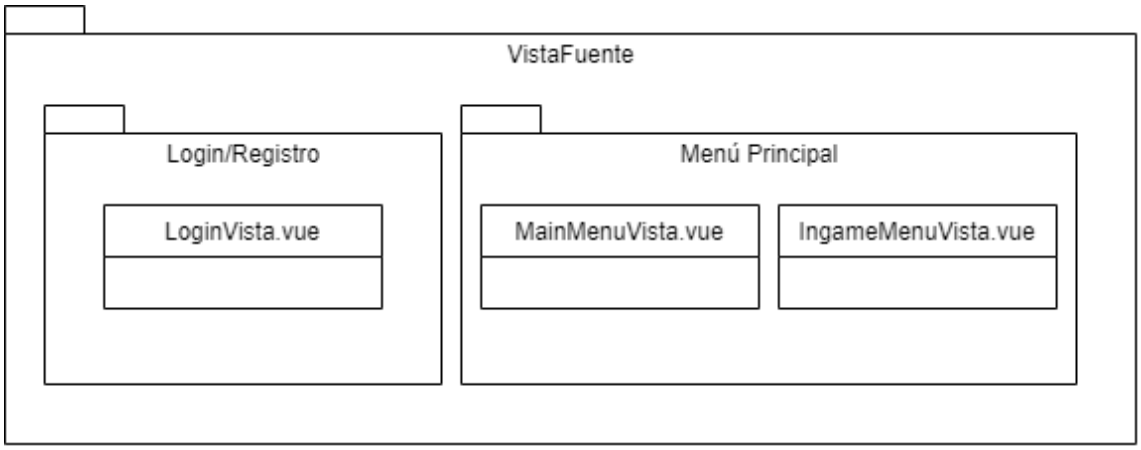


Figura 9. AplicaciónFuente - VistaFuente

2.3.2.2. MODELO FUENTE

De nuevo, la Fuente posee un diagrama reducido comparado con el cliente, esto se puede observar en la Figura 10**:**

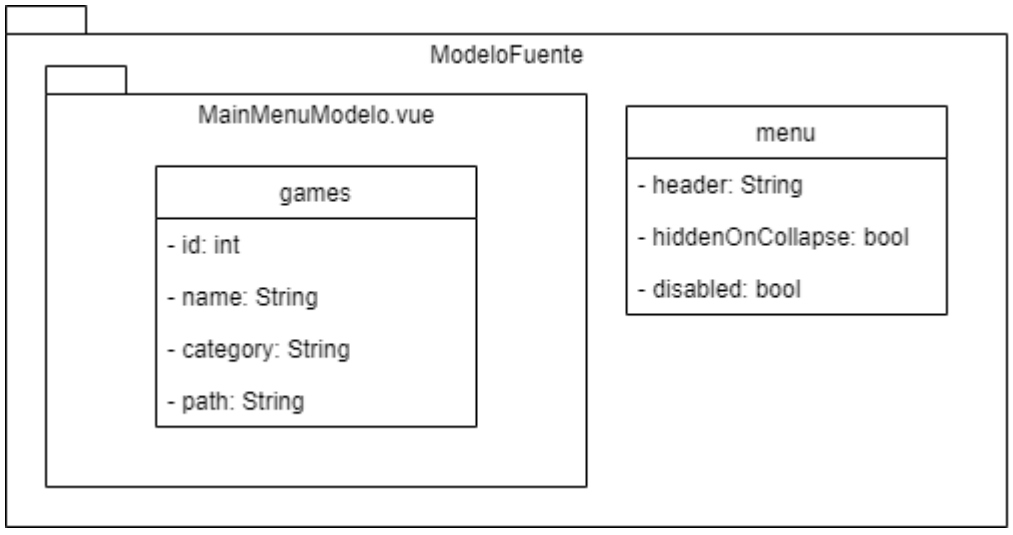


Figura 10. VistaFuente - ModeloFuente

2.3.2.3. VISTA-MODELO FUENTE

Esta subclase se describe en su totalidad a continuación en el diagrama de la Figura 11:

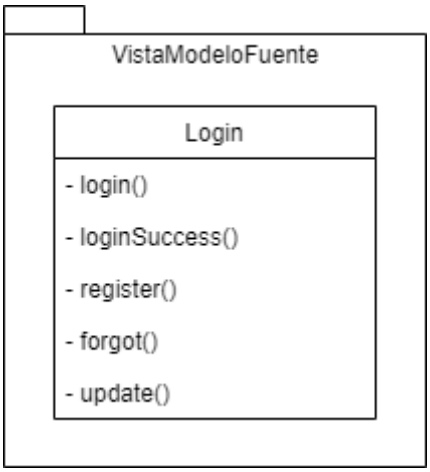


Figura 11. AplicaciónFuente - VistaModeloFuente

2.3.2.3. MÓDULO RETRANSMISIÓN FUENTE

Como ya se mencionó anteriormente, será en este módulo donde se vea la principal diferencia entre la fuente y el cliente. Esta diferencia proviene de la funcionalidad principal de la fuente: emitir, la cual el cliente no realiza, así como la fuente no captura inputs, si no que los recibe y aplica, tal y como muestra el diagrama de la **Figura 12**

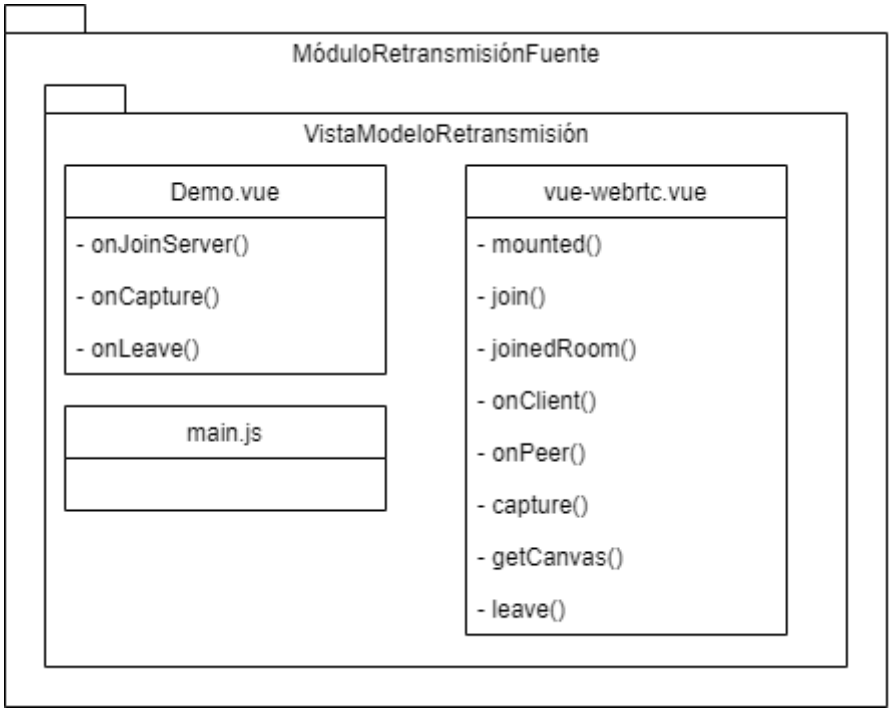


Figura 12. AplicaciónFuente - MóduloRetransmisiónFuente

2.3.3. APLICACIÓN SERVIDOR

El servidor se compone principalmente de una BBDD en PostgreSQL y de un archivo index.js en el cual se encuentran todos los endpoints y handlers, es por esto que en sí no existen apenas métodos, ya que casi la totalidad de estos se trata de cláusulas get y on, es por ello que en el diagrama de clases de diseño sí especificaremos el contenido de estos métodos en pos de facilitar la realización de casos de uso más adelante y de evitar la ambigüedad. Todo esto se puede observar en la **Figura 13:**

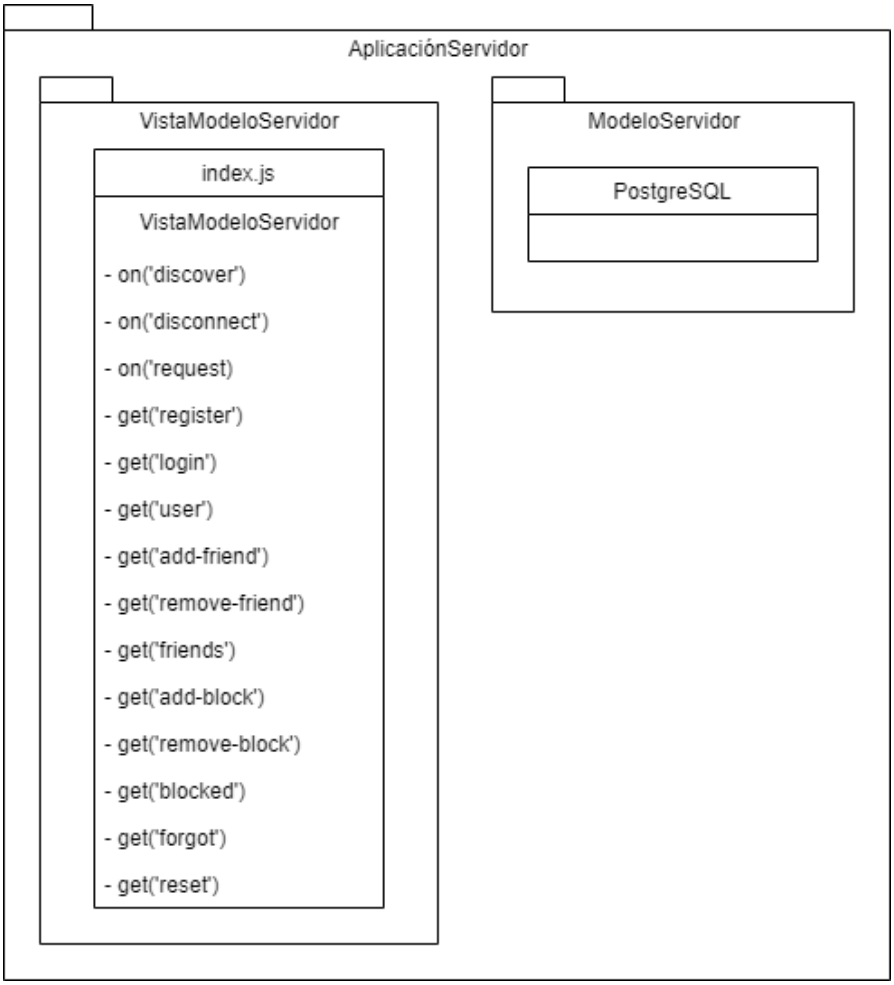


Figura 13. AplicaciónServidor

.

2.4. VISTA ARQUITECTÓNICA

Se procede a presentar la vista arquitectónica del sistema, respetando el patrón MVVM. Gracias a este diagrama se pueden observar con mayor claridad la comunicación y paso de datos existente en el sistema,

Como se puede observar en la **Figura 14**, el sistema ha sido dividido en 3 secciones bien distinguidas entre ellas, cada una de estas representa a cada uno de los estratos del modelo MVVM, siendo, de arriba hacia abajo, Vista, Vista Modelo y Modelo, respectivamente.

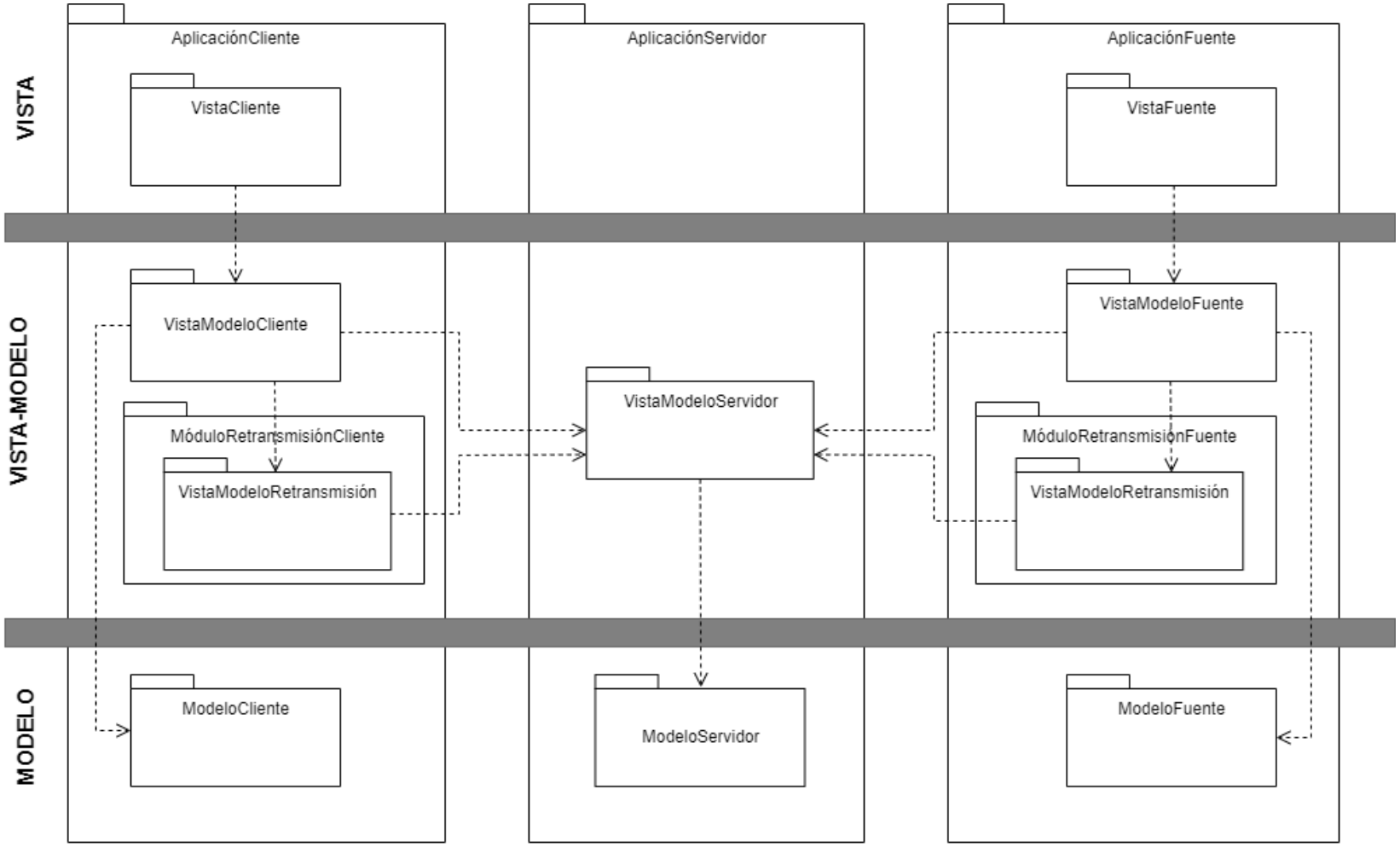


Figura 14. Vista arquitectónica

2.5. REALIZACIÓN DE CASOS DE USO - DISEÑO

Llegados a esta etapa de diseño, se procede a realizar una segunda reinterpretación de nuestros casos de uso, ahora centrada en los objetivos de esta sección, para ello nos valdremos de nuevo de los diagramas de secuencia que recogerán el paso de mensajes dado en las diferentes acciones del sistema.

2.5.1. DIAGRAMAS DE SECUENCIA (GESTIÓN DE USUARIOS)

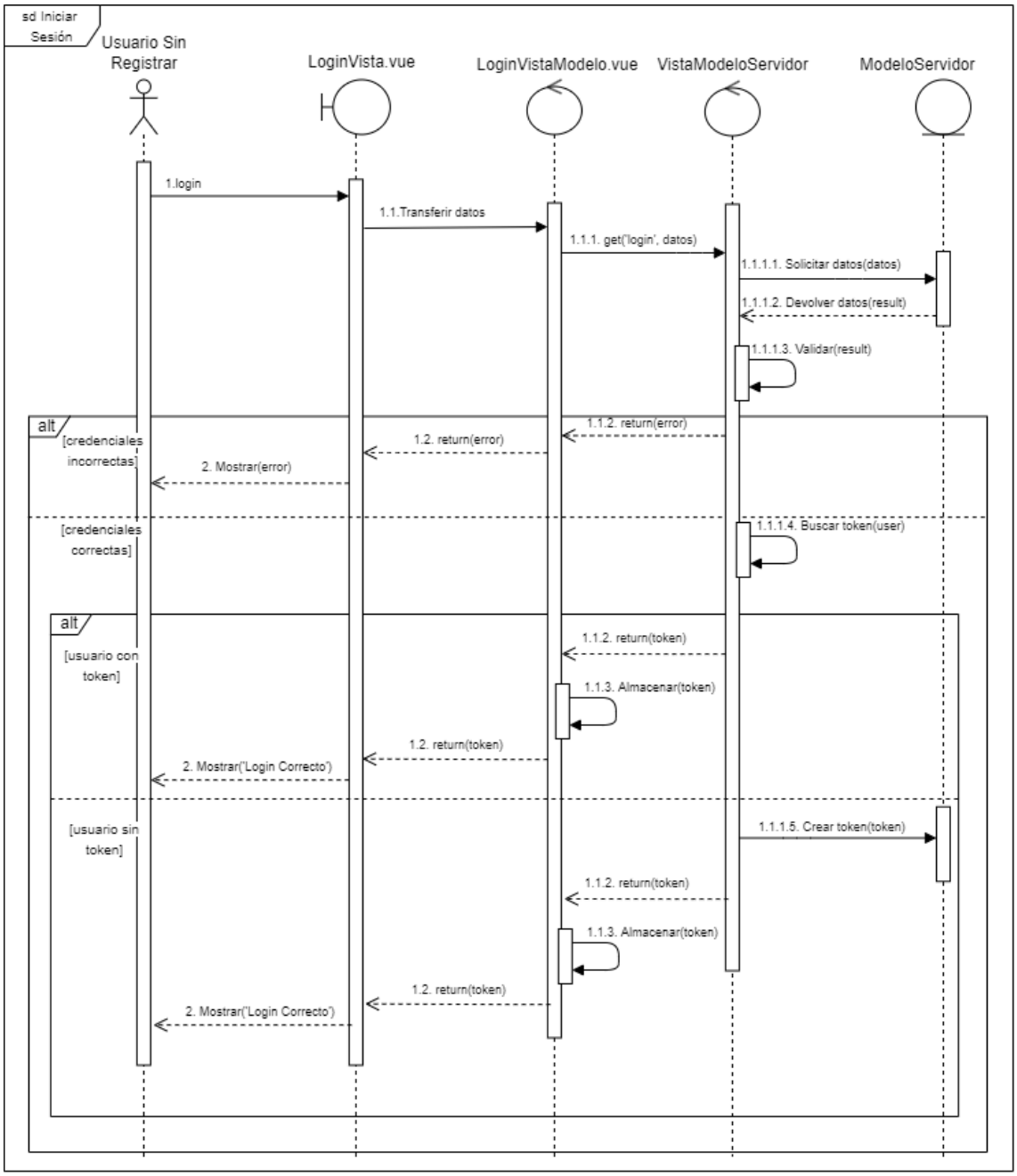


Figura 15. Diagrama de secuencia - Iniciar sesión

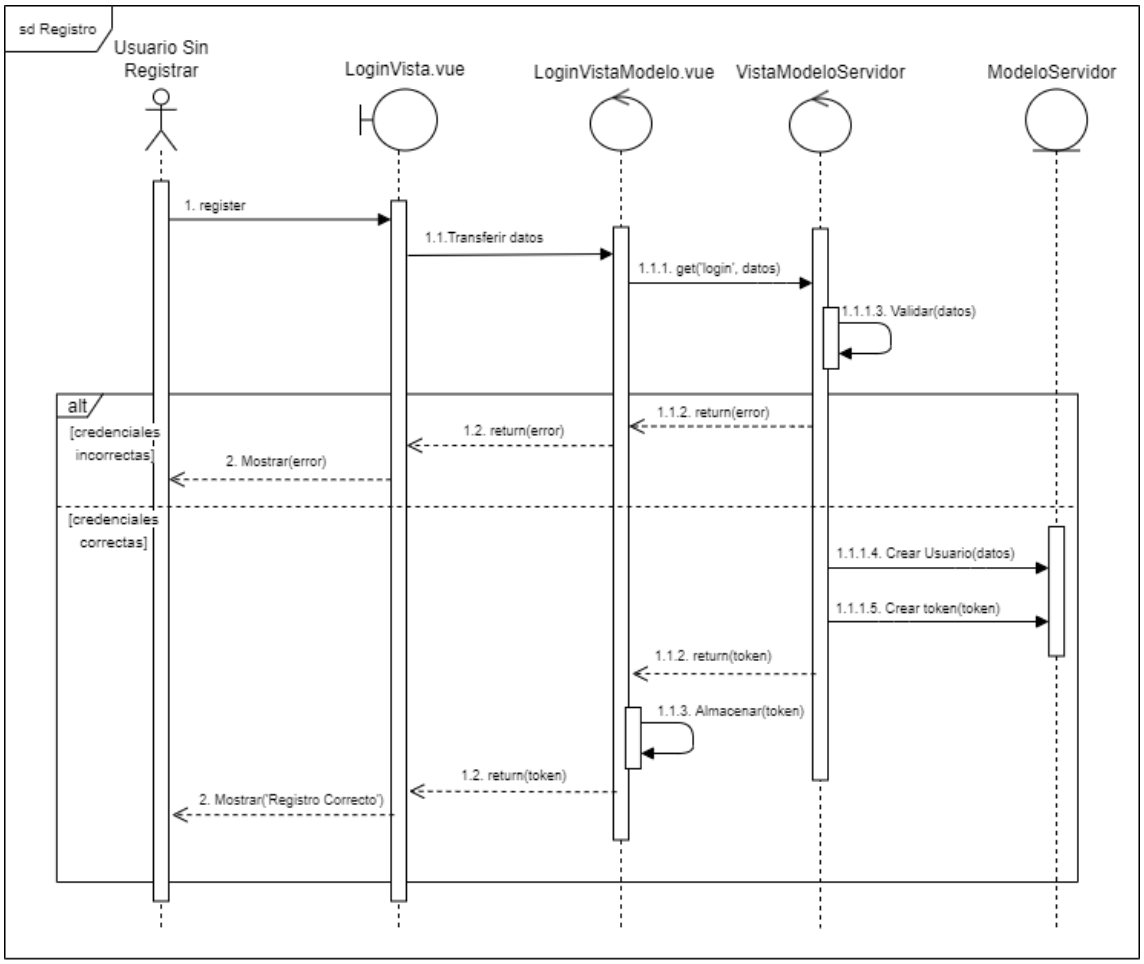


Figura 16. Diagrama de secuencia – Registro

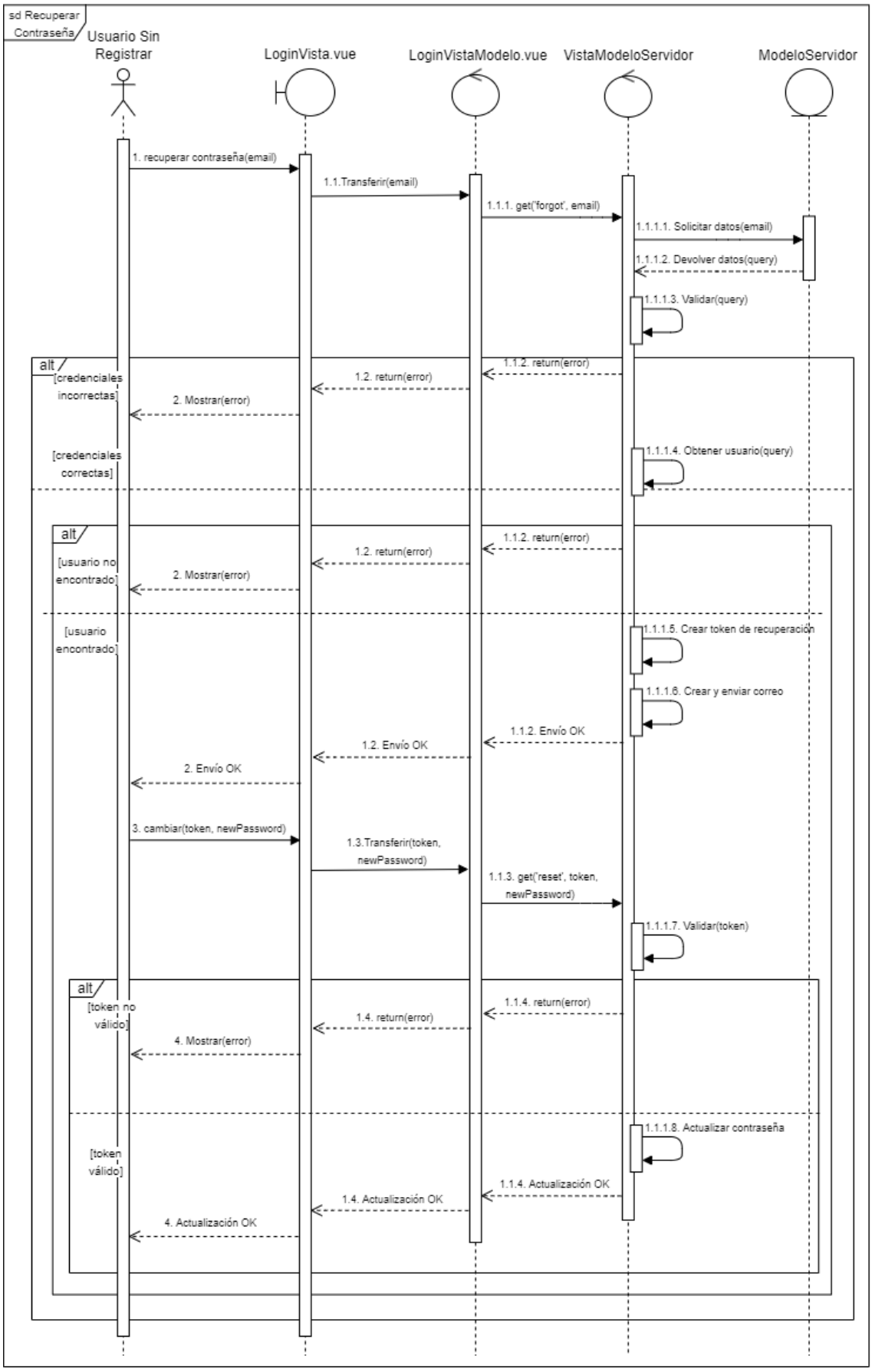


Figura 17. Diagrama de secuencia - Recuperar contraseña

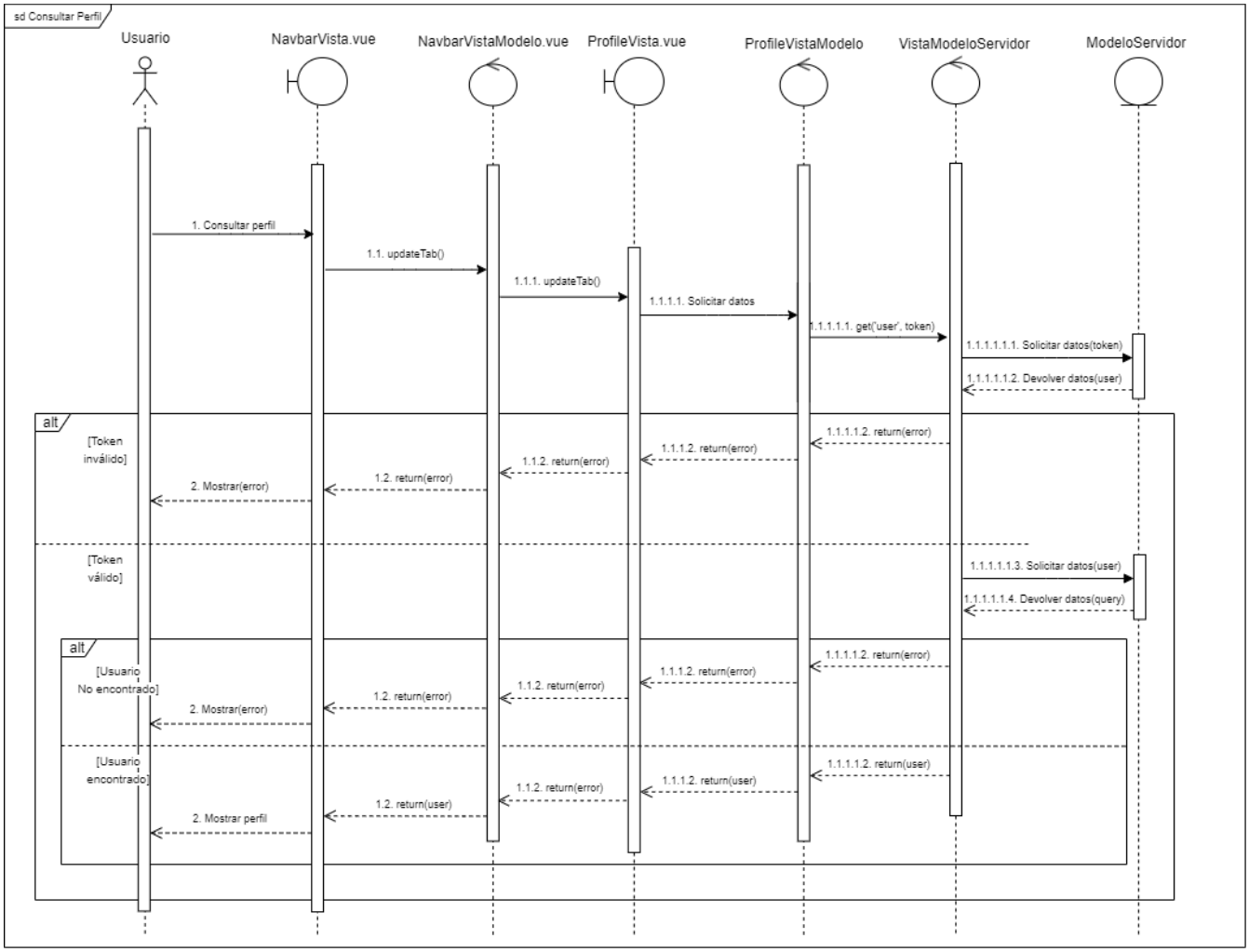


Figura 18. Diagrama de secuencia - Consultar perfil

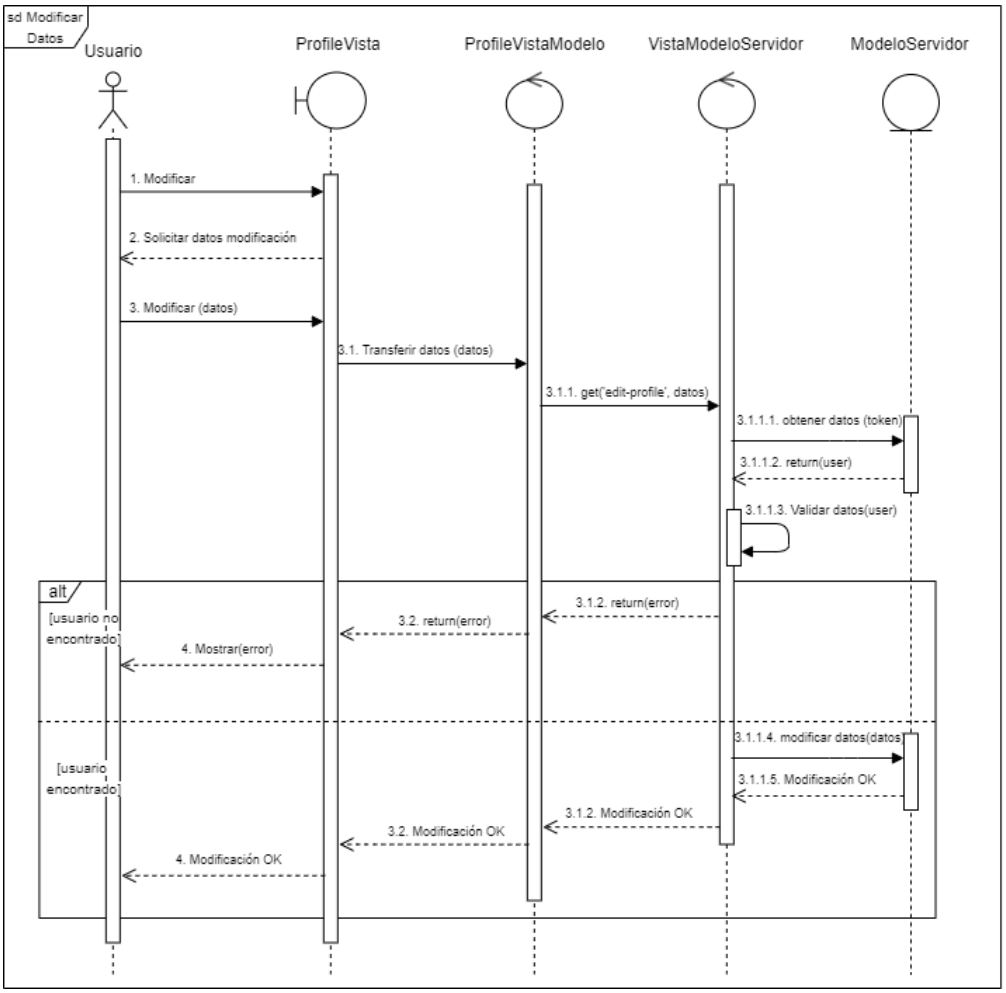


Figura 19. Diagrama de secuencia - Modificar datos

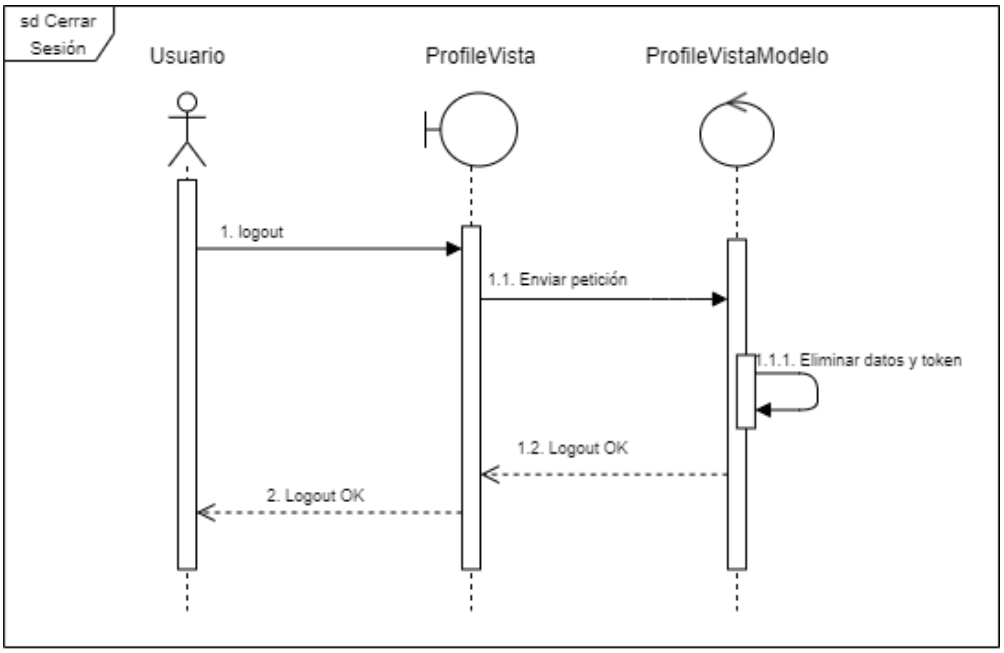


Figura 20. Diagrama de secuencia - Cerrar sesión

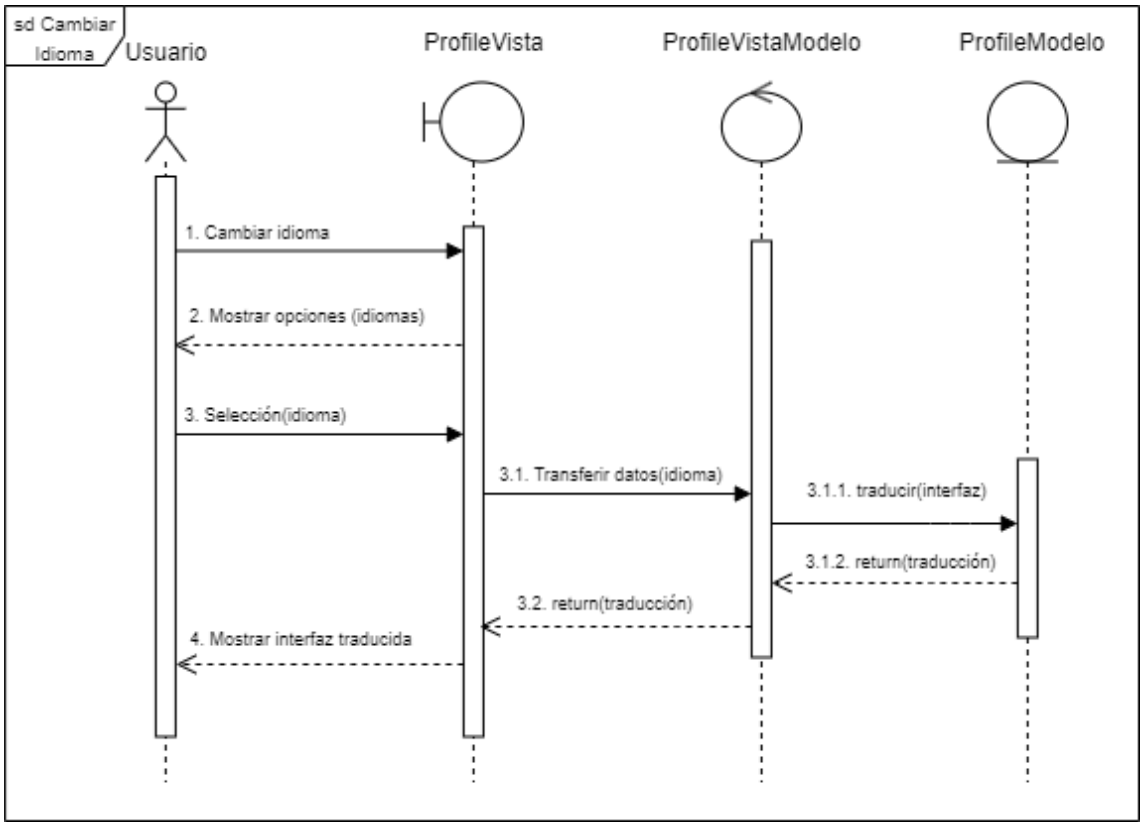


Figura 21. Diagrama de Secuencia - Cambiar Idioma

2.5.2. DIAGRAMAS DE SECUENCIA (GESTIÓN DE FUNCIONES SOCIALES)

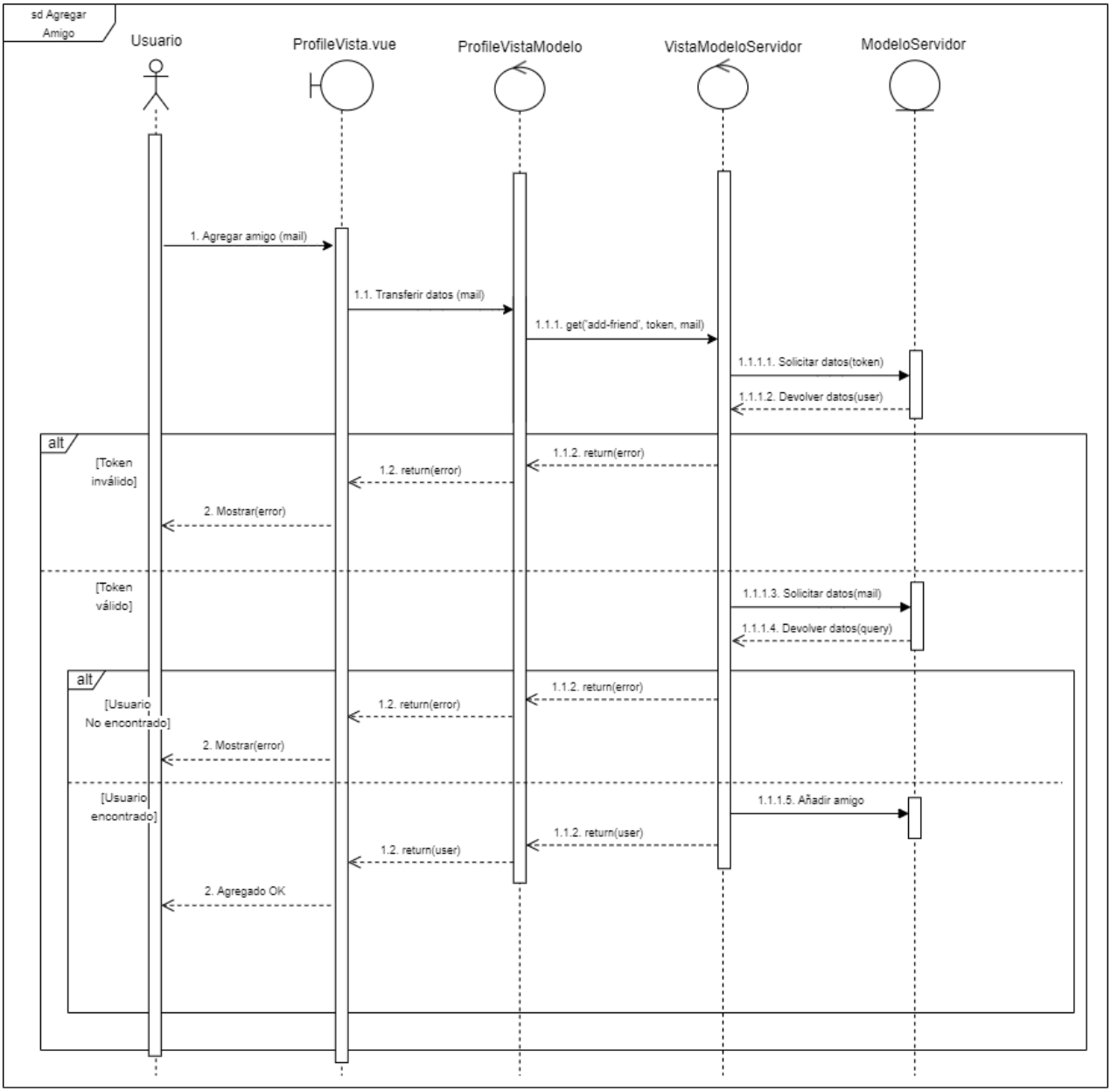


Figura 22. Diagrama de secuencia - Agregar amigo

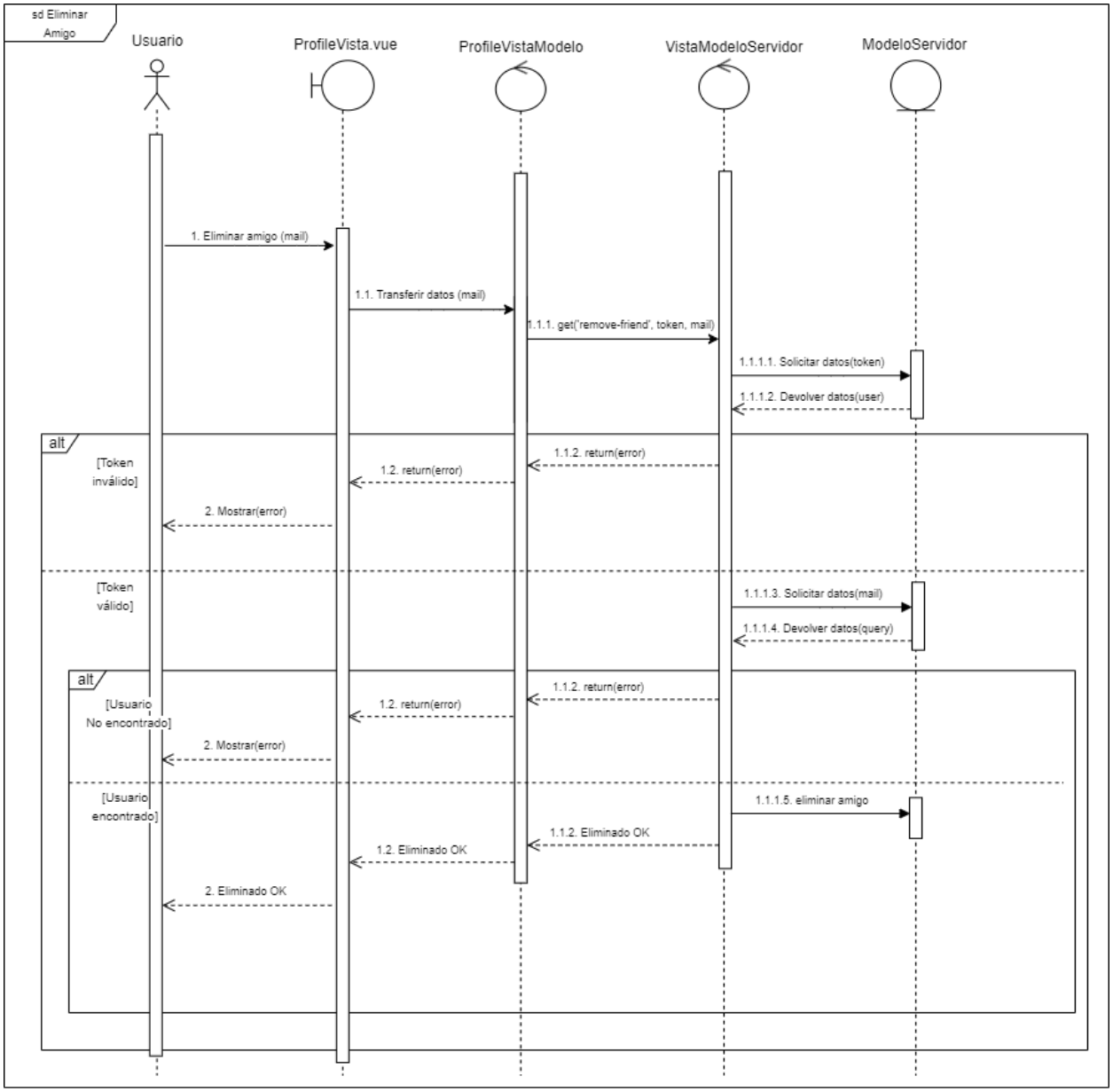


Figura 23. Diagrama de secuencia - Eliminar amigo

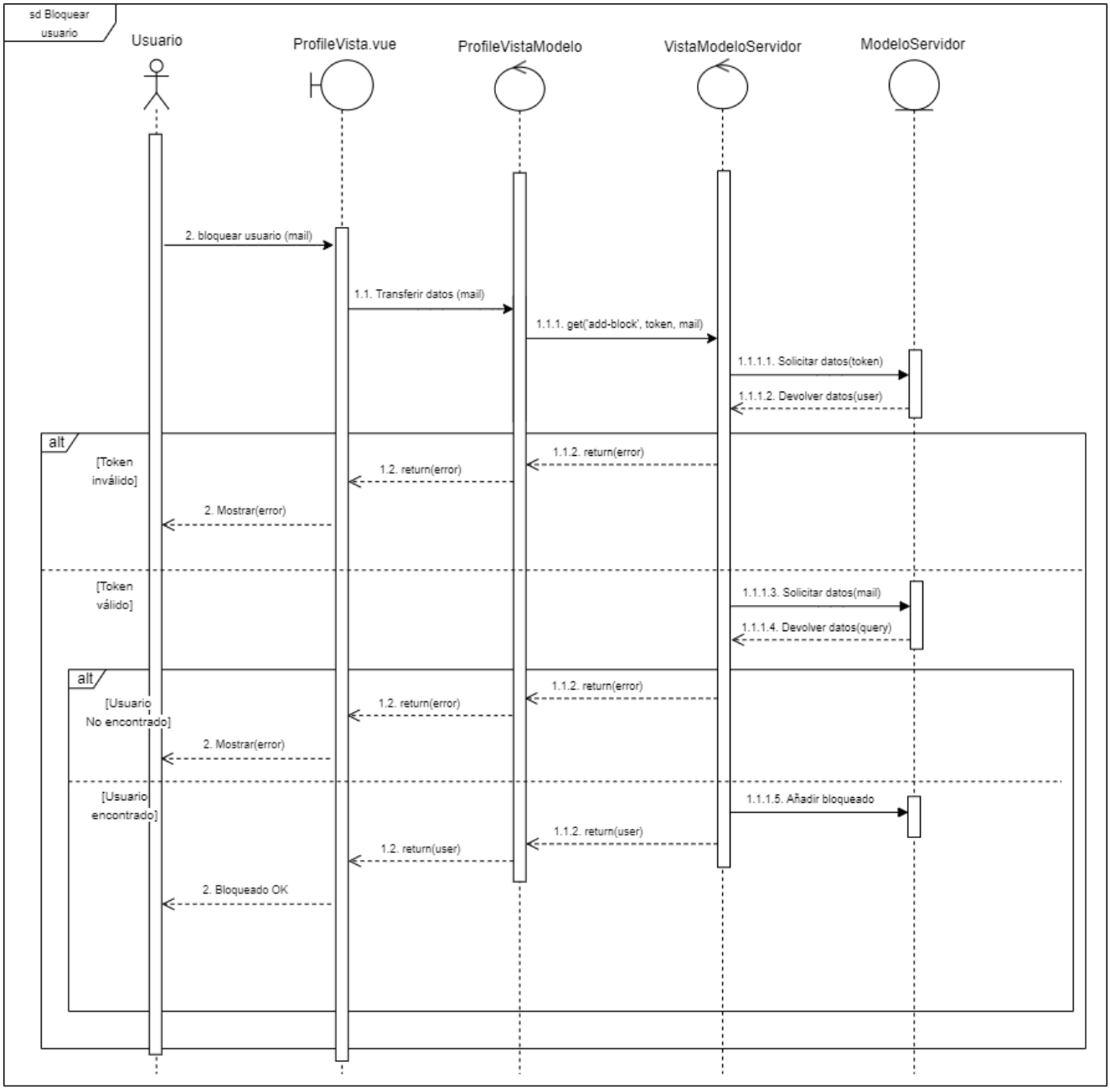


Figura 24. Diagrama de secuencia - Bloquear usuario

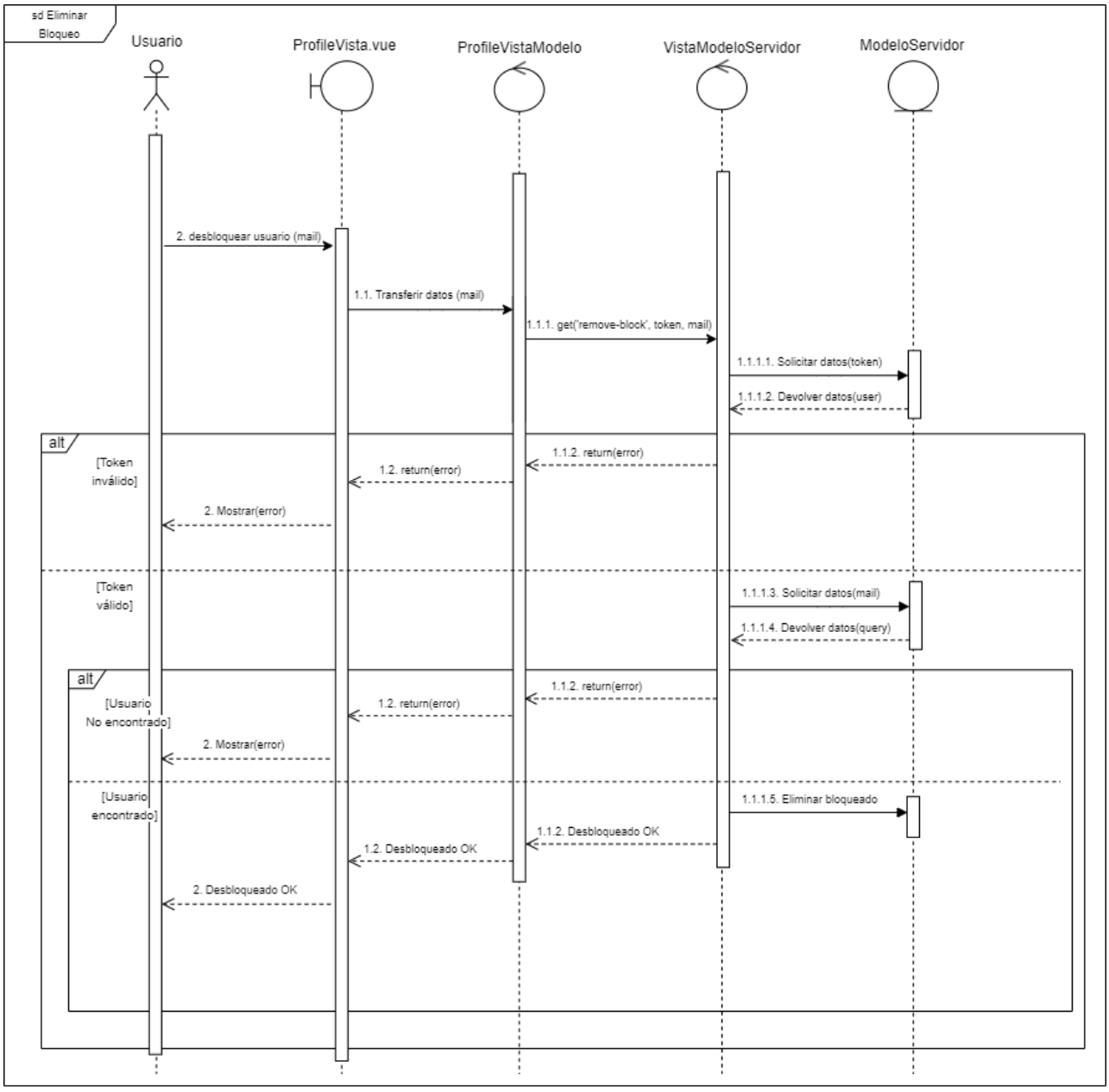


Figura 25. Diagrama de secuencia - Eliminar bloqueo

2.5.3. DIAGRAMAS DE SECUENCIA (GESTIÓN DE CATÁLOGO)

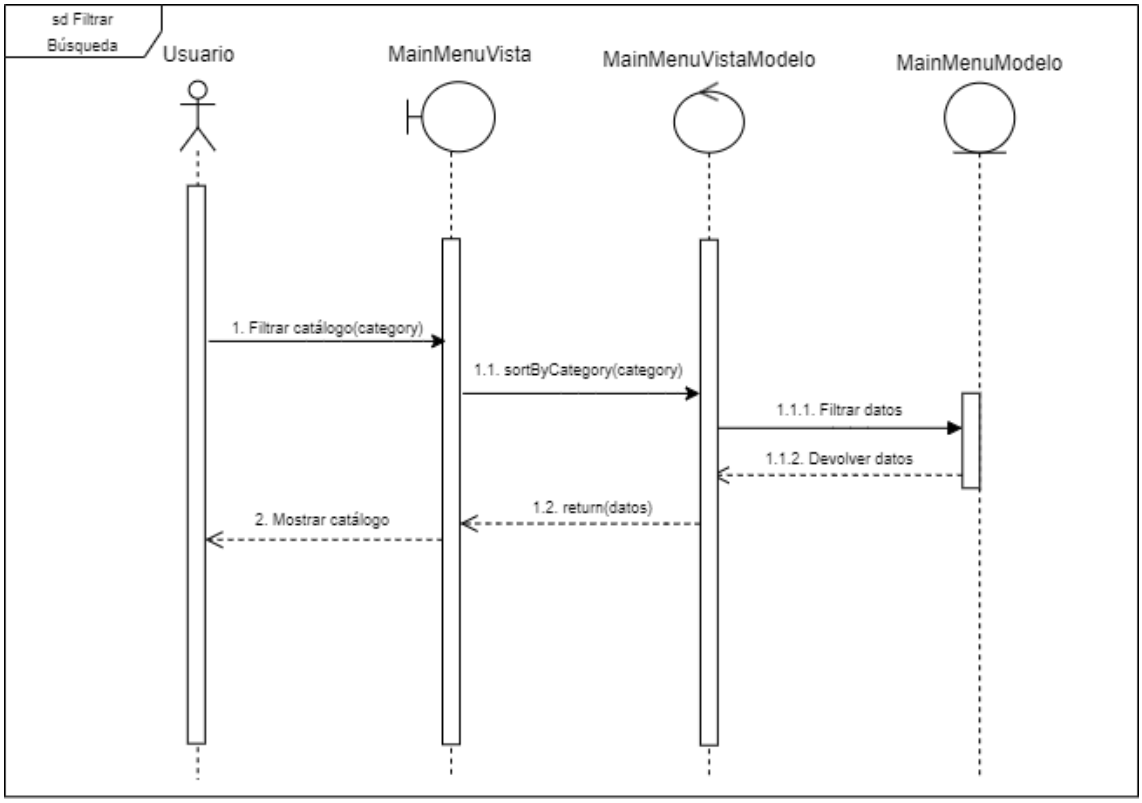


Figura 26. Diagrama de secuencia - Filtrar búsqueda

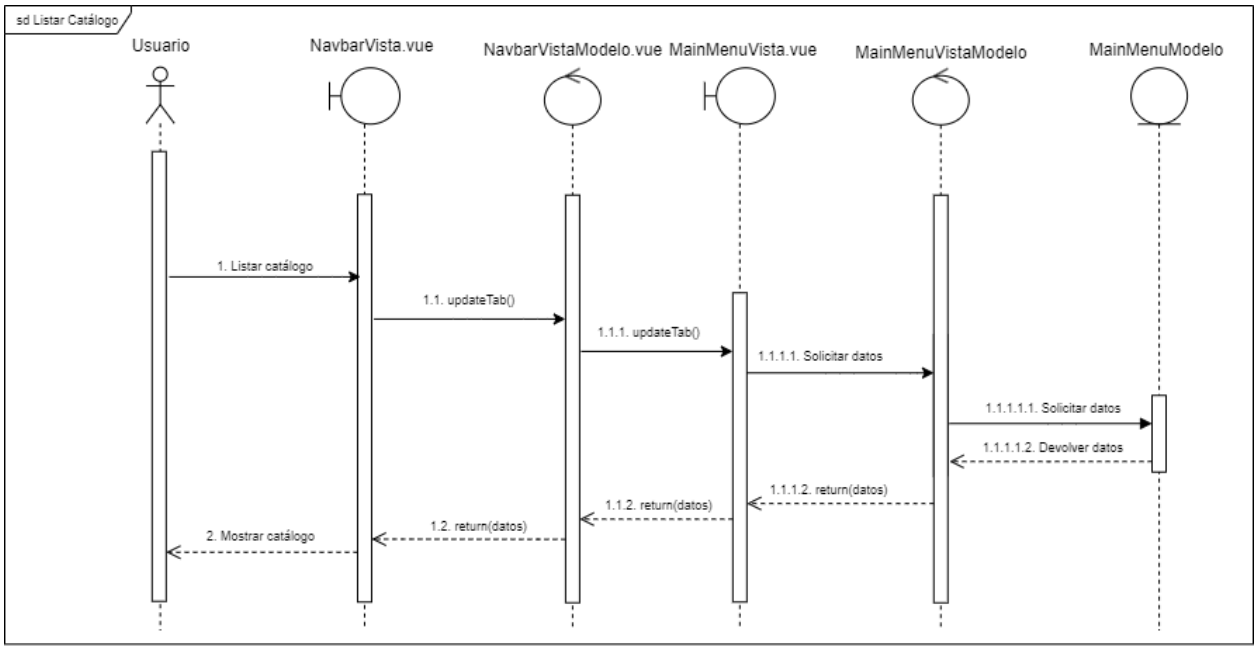


Figura 27. Diagrama de secuencia - Listar catálogo

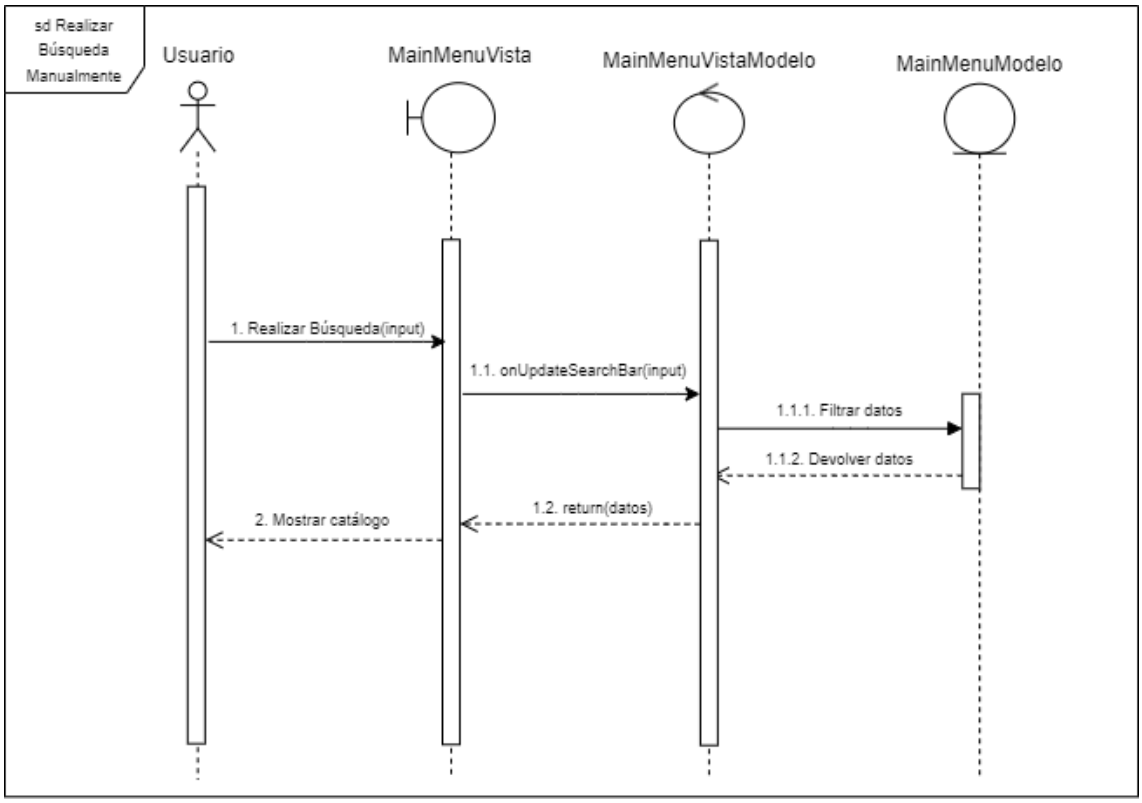


Figura 28. Diagrama de secuencia - Realizar búsqueda manualmente

2.5.4. DIAGRAMAS DE SECUENCIA (GESTIÓN DE RETRANSMISIÓN)

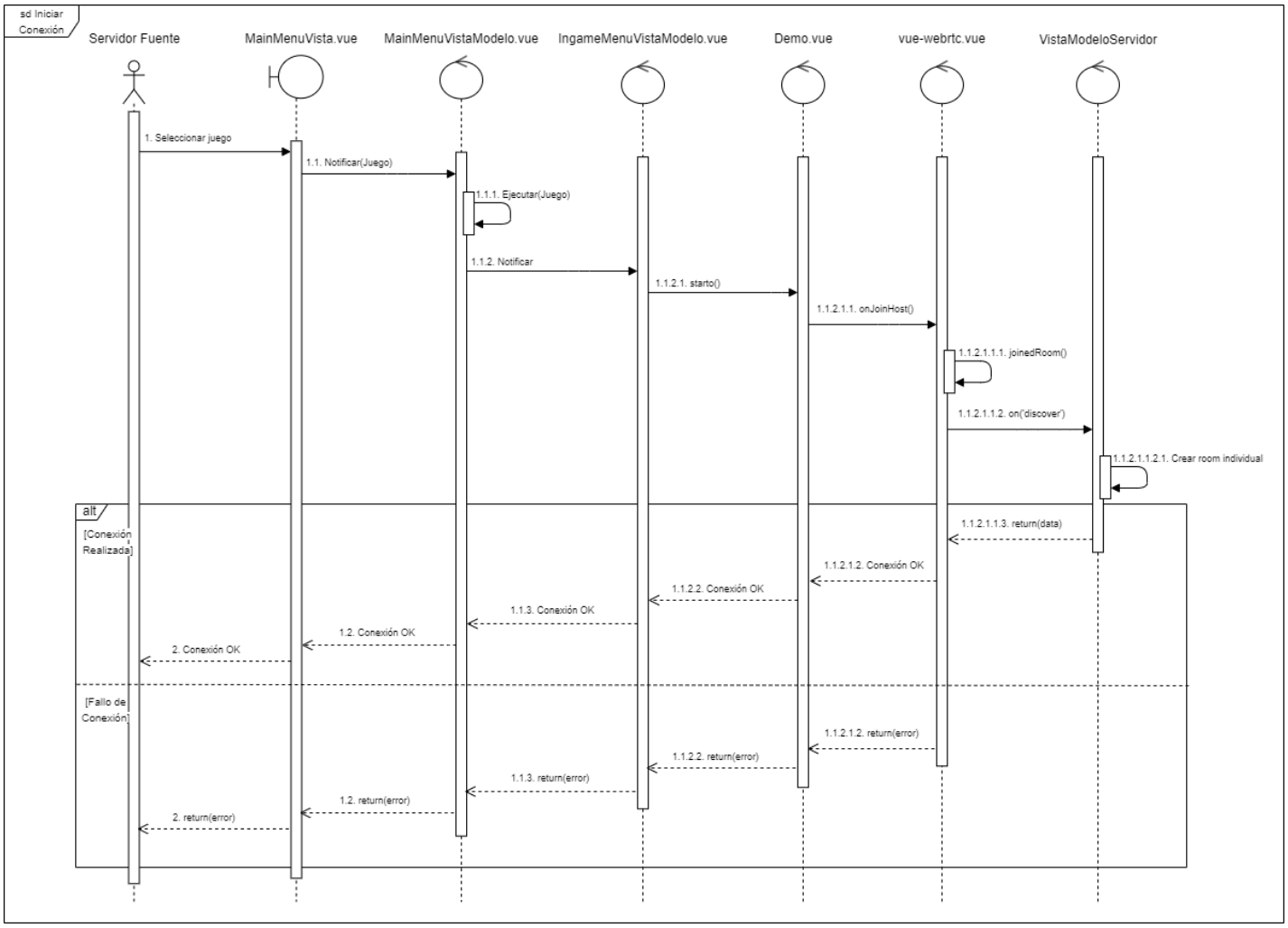


Figura 29. Diagrama de secuencia - Iniciar conexión

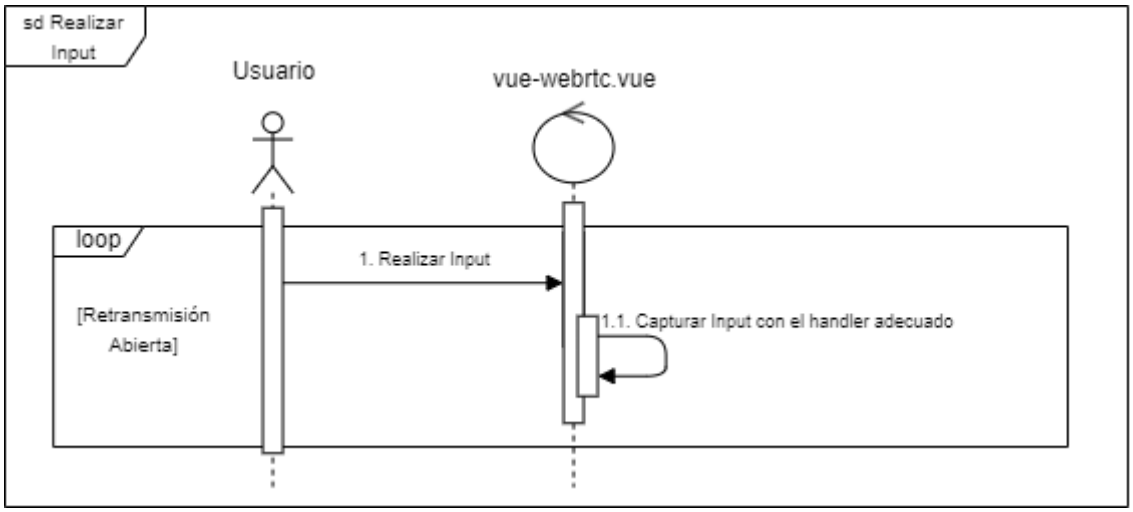


Figura 30. Diagrama de secuencia - Realizar input

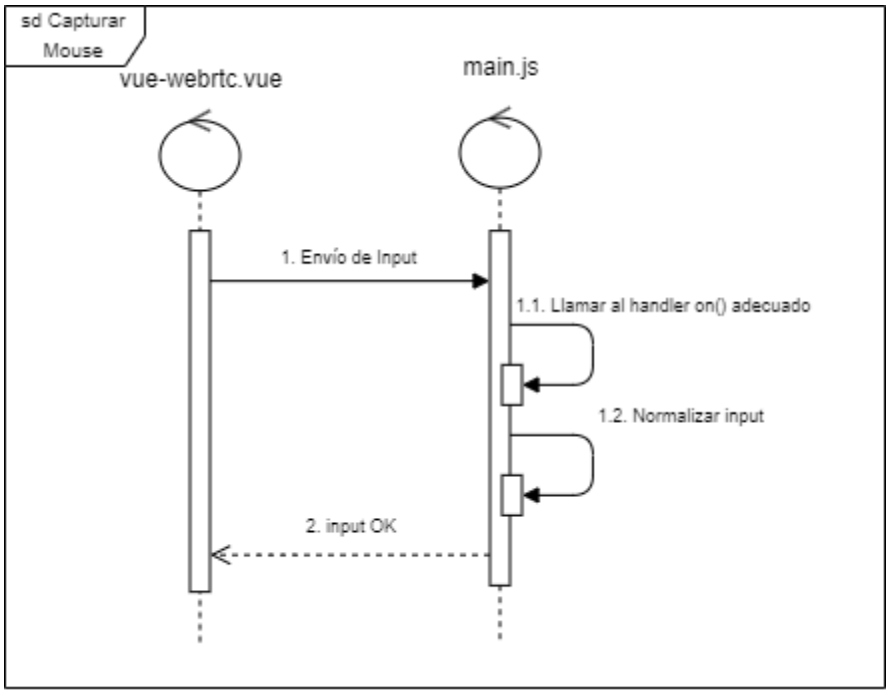


Figura 31. Diagrama de secuencia - Capturar mouse

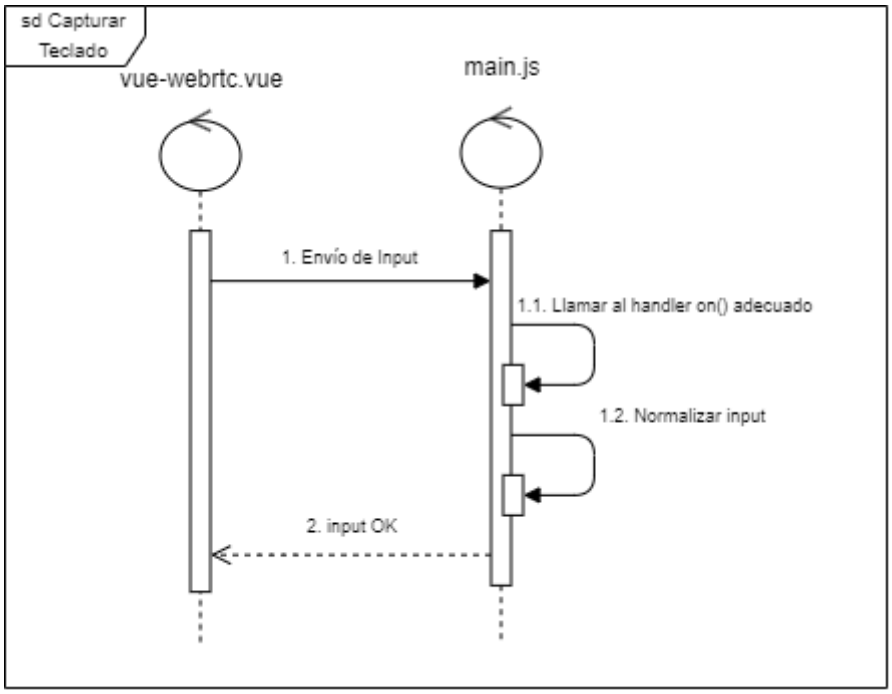


Figura 32. Diagrama de secuencia - Capturar teclado

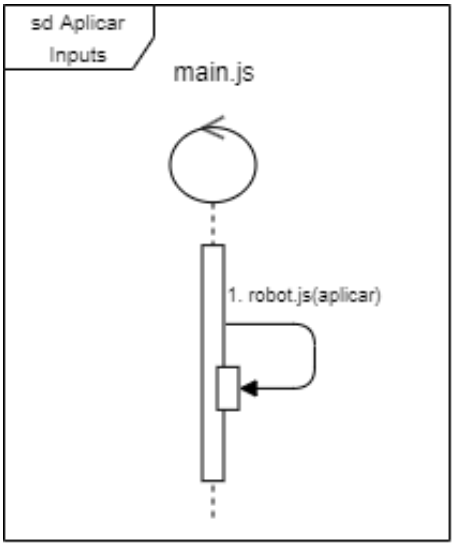


Figura 33. Diagrama de secuencia - Aplicar inputs

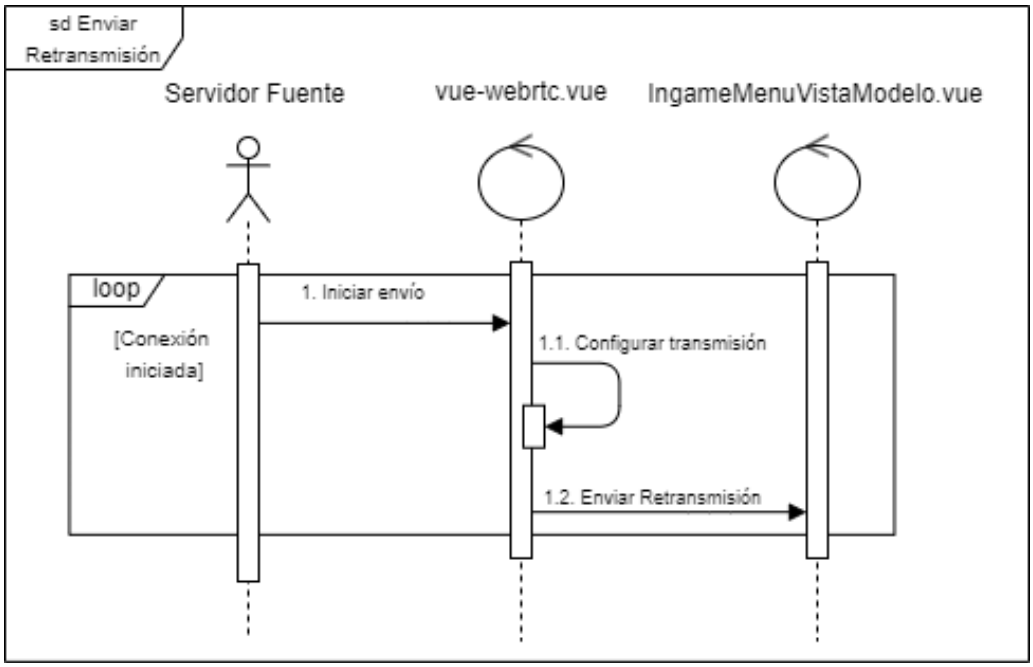


Figura 34. Diagrama de secuencia - Enviar retransmisión

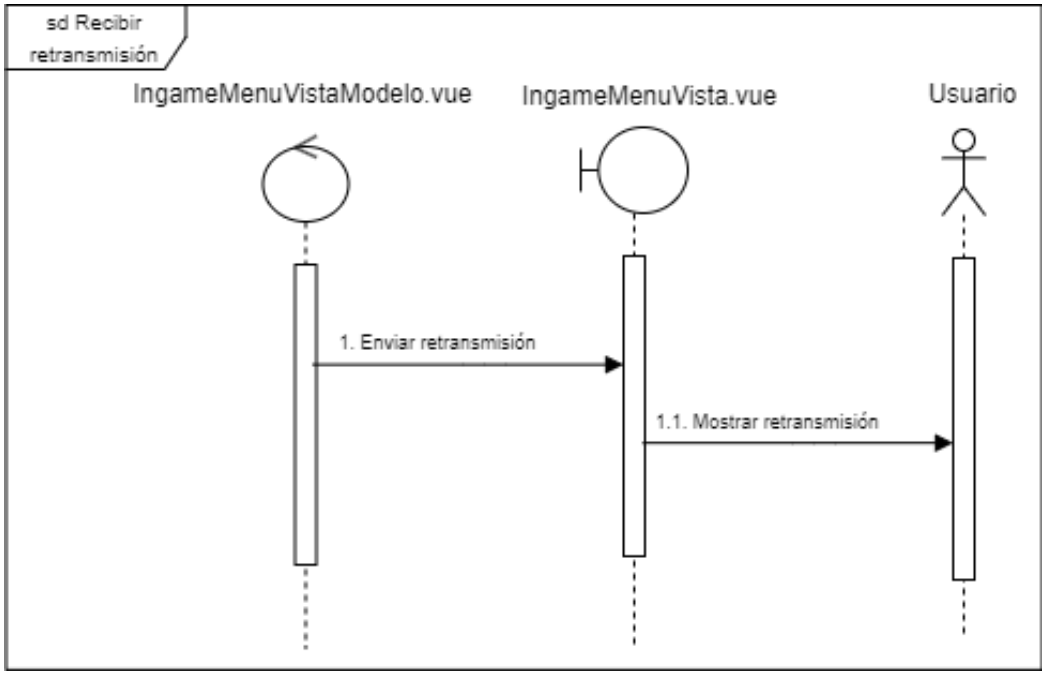


Figura 35. Diagrama de secuencia - Recibir retransmisión

3. DISEÑO DE LA BASE DE DATOS

Como ya se ha mencionado a lo largo de esta documentación, la base de datos del sistema viene dada por PostgreSQL, alojada de manera remota en un servidor de Amazon Web Services.

En la **Figura 36** se puede observar la disposición de la misma.

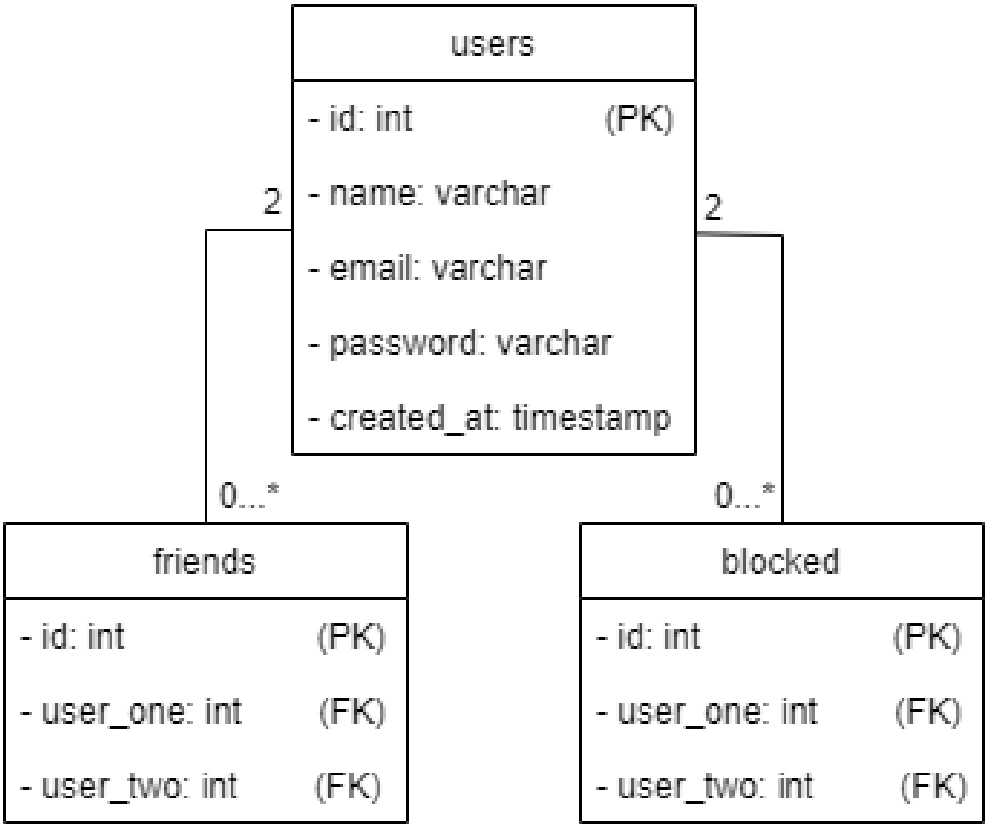


Figura 36. Diseño de la BBDD

4. MODELO DE DESPLIEGUE

En el diagrama de la **Figura 37** se puede observar el modelo de despliegue del sistema, en este, a su vez, podemos encontrar los siguientes nodos que lo conforman:

- **Computadora Cliente:** Representa al equipo físico del usuario, como se puede observar en el diagrama, este se ejecuta en una máquina con Sistema Operativo Windows.
  Este nodo representa las instancias de acceso de todos los tipos de usuario: Usuarios No Registrados, Usuarios y Administradores. Y, como se puede observar, pueden existir varias instancias simultáneas de estos nodos ya que el sistema es concurrente.
- **Computadora Fuente:** De nuevo, se trata de equipo físico, en este caso relativo a la fuente. Este nodo se ejecuta en Windows al igual que el equipo de los usuarios y, al igual que estos permite la existencia de múltiples equipos.
- **Servidor:** Servidor en el que se encuentran almacenados los datos del cliente. Ofrecerá una serie de servicios al cliente y a la fuente por medio de la API. Cabe destacar que solamente existe un único servidor sirviendo a todo el sistema.

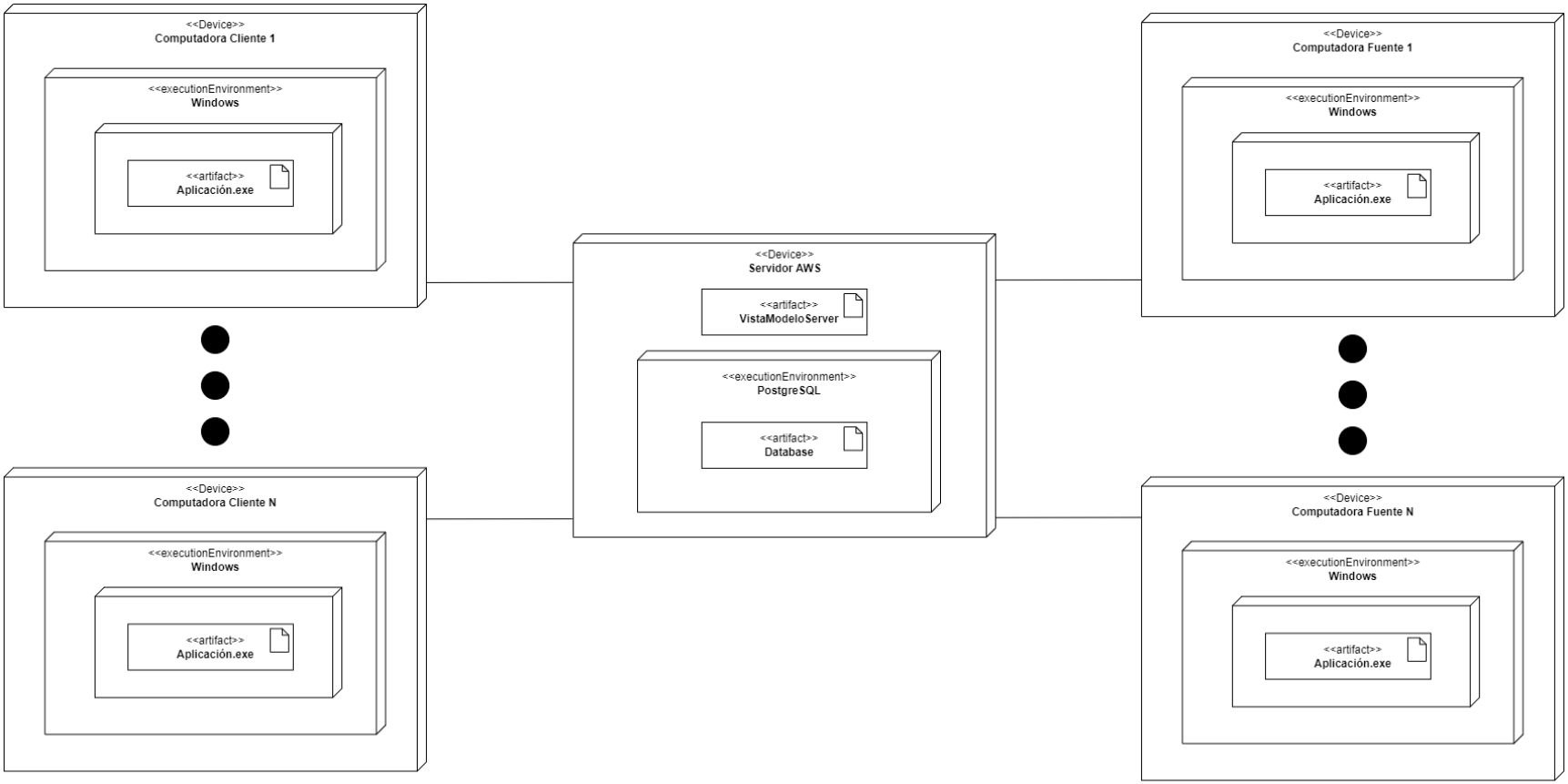


Figura 37. Modelo de despliegue

5. DISEÑO DE LA INTERFAZ

En este apartado se mostrarán y definirán las diferentes interfaces presentes en el sistema y sus componentes. En primer lugar, se presentarán las interfaces y protocolos mediante las cuales son ofrecidas y, posteriormente, mostrar su funcionalidad en detalle.

5.1. INTERFACES OFRECIDAS POR EL SISTEMA

La totalidad de las interfaces presentes en el sistema vienen dadas por el protocolo web HTTPS, concretamente, su variante GET, la división modular de este es bien simple, pues consiste en la instancia de la aplicación ejecutada por Electron y en los accesos que esta realiza a la API Express localizada en un servidor externo.

Para facilitar el entendimiento de esta descripción, se incluye en la **Figura 38** un diagrama de componentes de las interfaces del sistema.

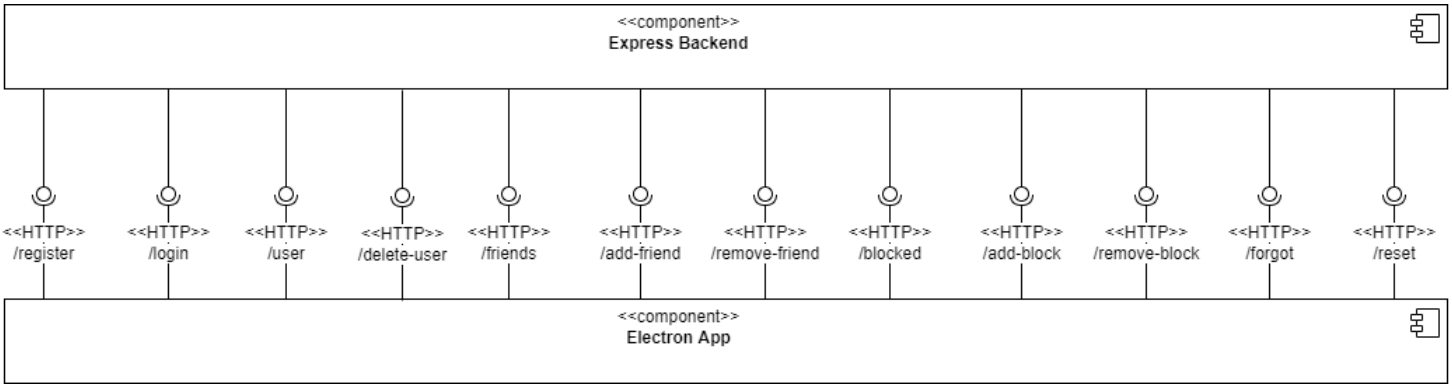


Figura 38. Diagrama de componentes - Interfaces del sistema

5.2. DEFINICIÓN DE INTERFACES

Se procede a describir las diferentes interfaces presentes en el sistema ofrecidas mediante el protocolo HTTPS.

Tabla 1. Endpoint /register

Endpoint	**/register**		
Módulo	*Express Backend*		
Descripción	Realiza una validación de los parámetros de registro introducidos por el usuario y los añade a la BBDD		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	email	String	Email proporcionado por el usuario en el form de registro
	password	String	Contraseña proporcionada por el usuario en el form de registro
	name	String	Nombre proporcionado por el usuario en el form de registro
Retorno	**Nombre**	**Tipo**	**Descripción**
	token	const	Token único que identifica a la sesión del usuario

Tabla 2. Endpoint /login

Endpoint	**/login**		
Módulo	*Express Backend*		
Descripción	Realiza una validación de los parámetros de login introducidos por el usuario y le asigna un token		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	email	String	Email proporcionado por el usuario en el form de registro
	password	String	Contraseña proporcionada por el usuario en el form de registro
Retorno	**Nombre**	**Tipo**	**Descripción**
	token	const	Token único que identifica a la sesión del usuario

Tabla 3. Endpoint /user

Endpoint	**/user**		
Módulo	*Express Backend*		
Descripción	Realiza una petición a la BBDD en busca de los datos del usuario solicitante y se los retorna		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
Retorno	**Nombre**	**Tipo**	**Descripción**
	user	JavaScript Object	Datos del usuario (nombre, mail, …)

Tabla 4. Endpoint /delete-user

Endpoint	**/delete-user**		
Módulo	*Express Backend*		
Descripción	Realiza una petición a la BDD para eliminar los datos del usuario		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
Retorno	**Nombre**	**Tipo**	**Descripción**
	respones	String	Estatus de la operación

Tabla 5. Endpoint /friends

Endpoint	**/friends**		
Módulo	*Express Backend*		
Descripción	Realiza una petición a la BBDD en busca de los amigos del usuario solicitante y se los retorna		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
Retorno	**Nombre**	**Tipo**	**Descripción**
	friends	JavaScript Object	Lista de amigos del solicitante

Tabla 6. Endpoint /add-friend

Endpoint	**/add-friend**		
Módulo	*Express Backend*		
Descripción	El usuario solicita agregar a otro usuario en su lista de amigos		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
	emailToAdd	String	Email del usuario que se agregará a la lista de amigos del solicitante
Retorno	**Nombre**	**Tipo**	**Descripción**
	response	String	Estatus de la operación

Tabla 7. Endpoint /remove-friend

Endpoint	**/remove-friend**		
Módulo	*Express Backend*		
Descripción	El usuario solicita eliminar a otro usuario de su lista de amigos		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
	emailToRemove	String	Email del usuario que se eliminará de la lista de amigos del solicitante
Retorno	**Nombre**	**Tipo**	**Descripción**
	response	String	Estatus de la operación

Tabla 8. Endpoint /blocked

Endpoint	**/blocked**		
Módulo	*Express Backend*		
Descripción	Realiza una petición a la BBDD en busca de los usuarios bloqueados del usuario solicitante y se los retorna		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
Retorno	**Nombre**	**Tipo**	**Descripción**
	blocked	JavaScript Object	Lista de usuarios bloqueados del solicitante

Tabla 9. Endpoint /add-block

Endpoint	**/add-block**		
Módulo	*Express Backend*		
Descripción	El usuario solicita bloquear a otro usuario		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
	emailToAdd	String	Email del usuario que se agregará a la lista de bloqueados del solicitante
Retorno	**Nombre**	**Tipo**	**Descripción**
	response	String	Estatus de la operación

Tabla 10. Endpoint /remove-block

Endpoint	**/remove-block**		
Módulo	*Express Backend*		
Descripción	El usuario solicita eliminar a otro usuario de su lista de bloqueados		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token único del usuario, usado para identificarle rápidamente en el servidor
	emailToRemove	String	Email del usuario que se eliminará de la lista de bloqueados del solicitante
Retorno	**Nombre**	**Tipo**	**Descripción**
	response	String	Estatus de la operación

Tabla 11. Endpoint /forgot

Endpoint	**/forgot**		
Módulo	*Express Backend*		
Descripción	El usuario solicita recuperar su contraseña y el sistema le envía un token de recuperación al correo indicado por el usuario (si este se encuentra en el sistema)		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	email	String	Email provisto por el usuario para recibir el token de recuperación
Retorno	**Nombre**	**Tipo**	**Descripción**
	response	String	Estatus de la operación

Tabla 12. Endpoint /reset

Endpoint	**/reset**		
Módulo	*Express Backend*		
Descripción	El usuario envía el token recibido junto a su contraseña al servidor, el cual se encargará de validar la operación y cambiar su contraseña		
Operación	GET		
Ubicación de los parámetros			
Formato de marshalling			
Parámetros	**Nombre**	**Tipo**	**Descripción**
	token	String	Token de verificación de cambio de contraseña proporcionado por el servidor al usuario
	newPassword	String	Nueva contraseña elegida por el usuario
Retorno	**Nombre**	**Tipo**	**Descripción**
	response	String	Estatus de la operación

5.3. ESPECIFICACIÓN INICIAL DE INTERFAZ GRÁFICA (WIREFRAME)

Será en esta sección donde se vea por primera vez una aproximación inicial de la interfaz del sistema. Para lograr esta representación, se ha valido del uso de wireframes a lo largo de las ilustraciones que se encuentran a continuación.

Las vistas que definen estas ilustraciones son las siguientes:

- Vista de login/registro: Se trata de la vista inicial en la cual el usuario podrá introducir sus credenciales para acceder a su cuenta o crear una de cero.
- Vista de catálogo: Una vez accedido al sistema, el usuario se ve ofrecido un catálogo con los diferentes juegos provistos por el servicio.
- Vista de juego: En principio se encuentra vacía, pero una vez se inicia la retransmisión será aquí donde se acceda a la misma.
- Vista de perfil: Se trata del perfil personal del usuario, en este se muestran diferentes datos sobre el mismo y algunas de las acciones sociales que este puede realizar.

Comenzando a explicar estas vistas tenemos a las **Figura 39** y **Figura 40**, las cuales se conforman de un fondo (indicado mediante el rectángulo tachado) sobre el cual se encuentra un formulario de login o registro, según corresponda, dentro del mismo se encuentran los diferentes inputs a ser rellenados por el cliente, así como el botón de confirmación y el botón para intercambiar una vista por la otra.

Como se puede observar, las diferencias entre ambas son mínimas y varían únicamente en el nombre de la propia vista y en el contenido del formulario.

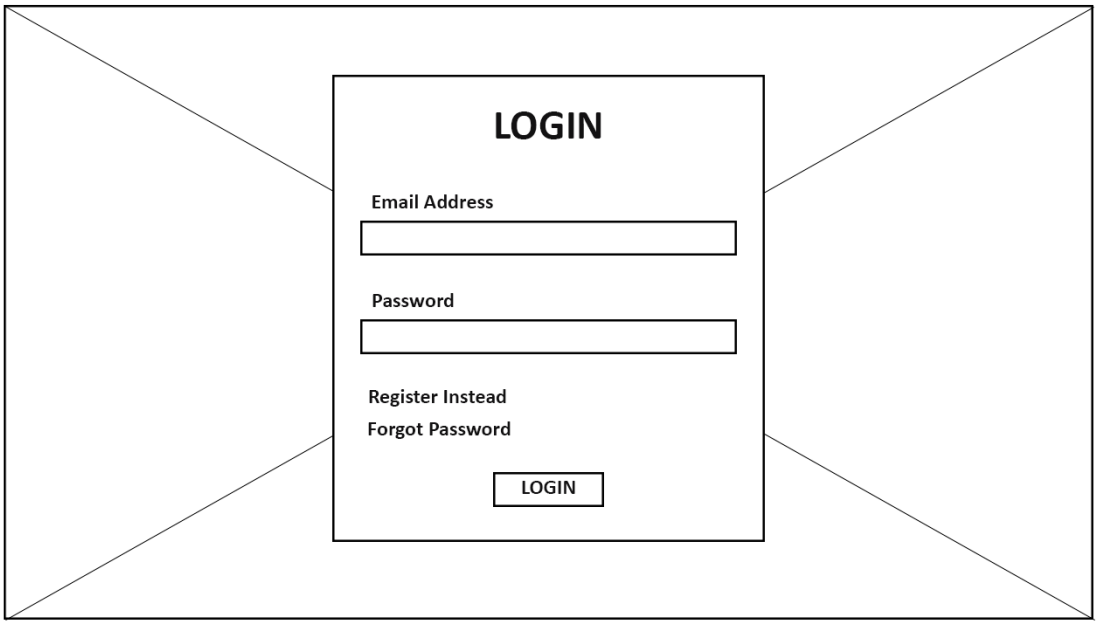


Figura 39. Wireframe - Vista de Login

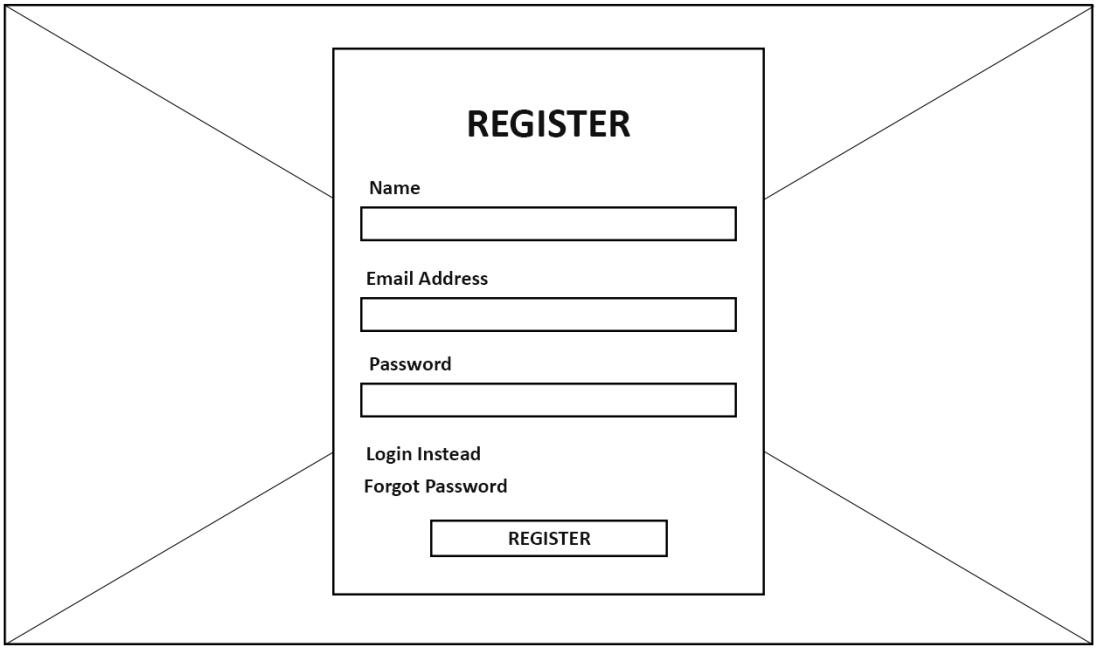


Figura 40. Wireframe - Vista de registro

La siguiente vista a presentar viene dada por la **Figura 41**, esta se trata de una de las dos vistas principales del sistema, ya que ofrece al cliente todos los videojuegos que este puede seleccionar para jugar. Sus características principales son la propia lista de videojuegos, ordenada por el género al que estos pertenecen, así como un filtro de búsqueda manual y un filtro de búsqueda por género.

Cabe destacar también la barra de navegación y la barra lateral, haciendo de menú de navegación y de lista de amigos, respectivamente.

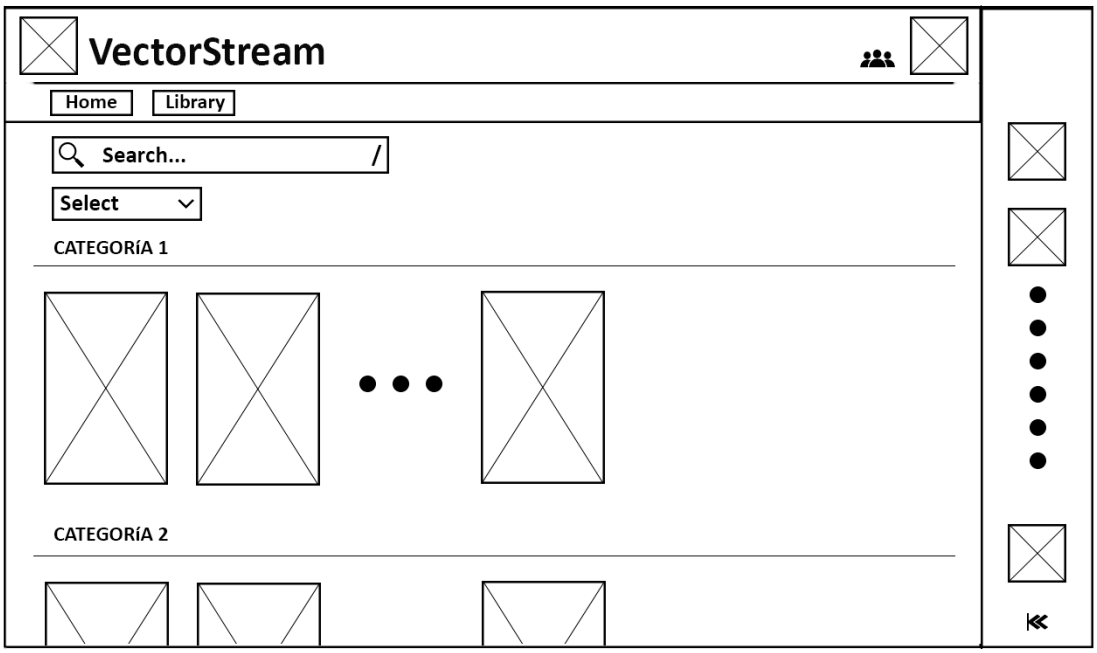


Figura 41. Wireframe - Vista de catálogo

La segunda de las dos principales vistas del sistema se trata de la vista de juego, esta posee dos claros estados: Uno de ellos es el estado pre-retransmisión, en el cual se presenta simplemente como una vista con el botón para unirse a la retransmisión (ya bien sea como host o como cliente) y el estado que se da durante la retransmisión, el cual muestra la imagen recibida y un nuevo botón para finalizar el recibimiento de vídeo. Es por esto que esta vista posee dos figuras: La y la representando los dos estados que se pueden dar.

Por supuesto, mantiene la barra de navegación y la barra lateral de la vista anterior.

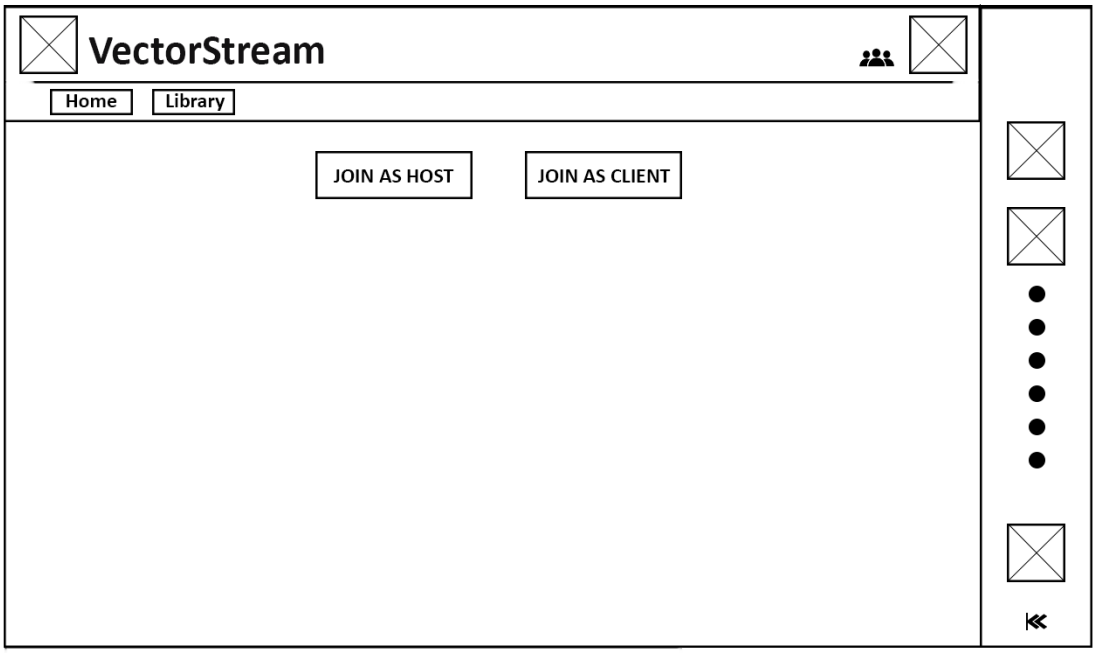


Figura 42. Wireframe - Vista de juego (pre-retransmisión)

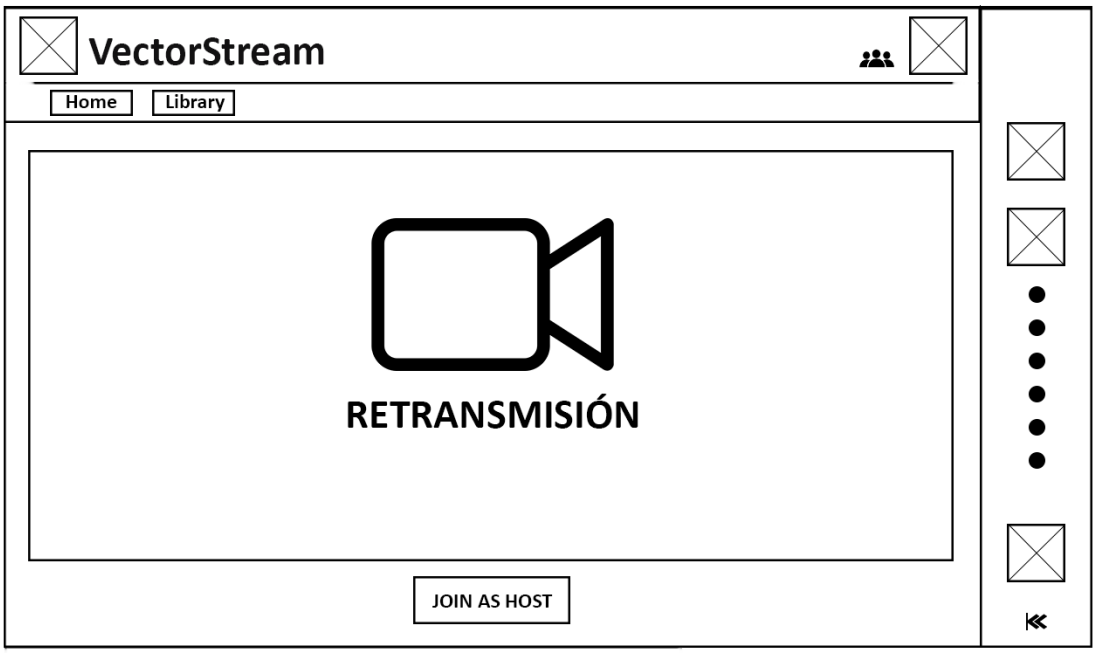


Figura 43. Wireframe - Vista de juego (retransmisión)

Por último, nos encontramos con la vista de perfil, en esta se presentan los diferentes datos del usuario, así como la posibilidad de añadir o bloquear a otros usuarios mediante sus correos electrónicos.

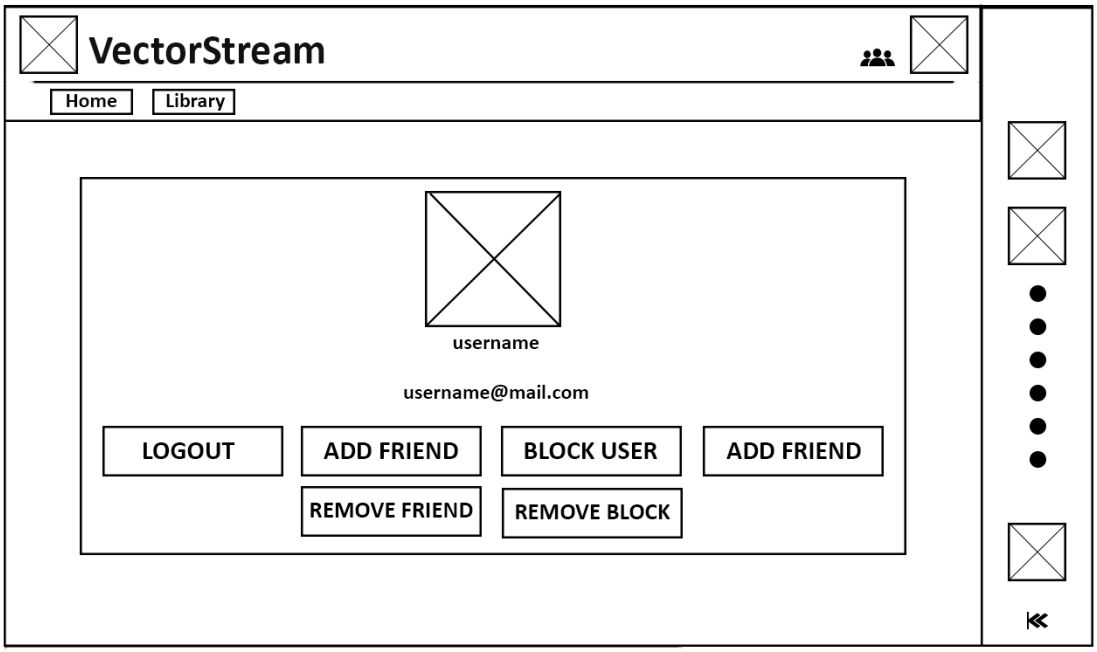


Figura 44. Wireframe - Vista de perfil

6. CONCLUSIONES TRAS EL DESARROLLO

Como el nombre de esta sección sugiere, fue añadida una vez se completó el desarrollo del proyecto, el fin de la misma es aclarar algunos puntos de interés entre el sistema final desarrollado y este anexo:

En cuanto al diseño de la vista, los wireframes fueron respetados al máximo y sirvieron como plantilla a la hora de diseñar los diferentes grids de la vista, esto aceleró la creación del proyecto. Por otro lado, en cuanto al tipo de datos descrito a la hora de definir las interfaces y módulos del sistema, también fueron utilizados como referencia y se encuentran de manera similar implementados en el proyecto.

Otro de los grandes puntos a destacar es la flexibilidad otorgada por el Proceso Unificado, en las primeras iteraciones de este anexo se presentaban algunos diagramas de secuencia – diseño que acabaron difiriendo de su implementación final, gracias al proceso iterativo estos pudieron ser corregidos tomando el proyecto desarrollado como referencia, es por esto también que se puede considerar al proyecto y a este anexo como una relación de retroalimentación: el anexo sirvió como primeras pautas de diseño y estructurales y, el proyecto final desarrollado sirvió como herramienta para pulir y redefinir con exactitud este anexo corrigiendo así incoherencias.

7. BIBLIOGRAFÍA

akin-ogundenji, M. (29 de 11 de 2015). *medium.com.* Obtenido de https://medium.com/@nohkachi/my-vue-on-mvvm-c3ffb4f0678f#:~:text=MVVM%20stands%20for%20Model%2DView,access%20layer%20of%20the%20application.

FairCom. (29 de 4 de 2022). *docs.faircom.com.* Obtenido de https://docs.faircom.com/doc/knowledgebase/54219.htm

ProgrammerClick. (2020-2022). *programmerclick.com.* Obtenido de https://programmerclick.com/article/27631228428/

ReactiveProgramming. (2022). *reactiveprogramming.io*. Obtenido de https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor

Zuev, A. (5 de 6 de 2020). *adictosaltrabajo.com.* Obtenido de https://www.adictosaltrabajo.com/2020/06/05/patron-mvvm-en-swiftui/

ANEXO V: Documentación técnica.

Plataforma streaming multidispositivo de videojuegos

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

Agosto de 2022

Autor:

Jesús Manuel García Prieto

Tutores:

Diego J. Valdeolmillos Villaverde

Alfonso González Briones

Francisco Pinto Santos

Jesús Manuel García Prieto

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este anexo se adjuntará todo lo relativo al código fuente y a la documentación técnica del mismo, comenzando por el código fuente y la estructura de ficheros en la que este se encuentra, seguido de la documentación técnica, presentando los diferentes módulos en los que se ha decidido dividir esta.

En cuanto a las decisiones tomadas, la documentación ha sido generada con JSDoc-VueJS, en el formato estándar, y se ha documentado tanto la aplicación cliente como el servidor backend Express.

2. CÓDIGO FUENTE

El código fuente entregado sigue la estructura original creada por Vue, el código fuente relativo a la aplicación cliente/fuente se encuentra dentro de la carpeta VectorStream/, concretamente:

- **"src"**: Esta carpeta posee el gran grueso del código fuente, en concreto:
  - ***"App.vue"***
  - **"views":** Posee las vistas principales del proyecto (excepto App.vue), estas son:
    - ***"IngameMenu.vue"***
    - ***"Login.vue"***
    - ***"MainMenu.vue"***
    - ***"Profile.vue"***
  - **"components":** Posee los componentes del proyecto, estos son:
    - ***"Demo.vue"***
    - ***"LocaleSwitcher.vue"***
    - ***"Navbar.vue"***
    - ***"vue-webrtc.vue"***
- **"package.json":** Contiene las funciones customizadas para facilitar el uso de electron, se destaca electron:serve que será la encargada de lanzar la aplicación en Electron.
- **"dist":** Generada a partir del comando "npm run electron:build", alberga el ejecutable del proyecto final.
- **"out":** Alberga la documentación técnica de los diferentes módulos del sistema, se hablará más sobre esto en el siguiente apartado.

Además de esto cabe destacar el haberse documentado el código del backend del servidor Express, este código pertenece al archivo index.js, este se encuentra dentro de la carpeta ExpressBackend/.

3. DOCUMENTACIÓN TÉCNICA

La documentación técnica se puede encontrar bajo la carpeta "out", en la **Tabla 1** se puede observar un desglose de los diferentes módulos documentados y sus nombres.

Tabla 1. Módulos documentados

Módulo	Descripción
module-App.html	Documentación generada en HTML con JSDoc-Vuejs para el módulo App
module-Demo	Documentación generada en HTML con JSDoc-Vuejs para el módulo Demo
module-IngameMenu	Documentación generada en HTML con JSDoc-Vuejs para el módulo IngameMenu
module-LocaleSwitcher	Documentación generada en HTML con JSDoc-Vuejs para el módulo LocaleSwitcher
module-Login	Documentación generada en HTML con JSDoc-Vuejs para el módulo Login
module-MainMenu	Documentación generada en HTML con JSDoc-Vuejs para el módulo MainMenu
module-Navbar	Documentación generada en HTML con JSDoc-Vuejs para el módulo Navbar
module-Profile	Documentación generada en HTML con JSDoc-Vuejs para el módulo Profile
module-Vue-webrtc	Documentación generada en HTML con JSDoc-Vuejs para el módulo Vue-webrtc
module-ExpressBackend	Documentación generada en HTML con JSDoc-Vuejs para el módulo Vue-ExpressBackend

Como nota final, se hablará brevemente sobre la documentación generada, desde el módulo se pueden acceder a todos los demás módulos, así como a los diferentes eventos del sistema (con excepción del módulo ExpressBackend, ya que este pertenece a su propia sección por lo que se encuentra apartado del resto). También muestra los diferentes elementos pertenecientes a "data" o "props" y los diferentes métodos y eventos de este módulo en concreto.

ANEXO VI: Manual de Usuario.

Plataforma streaming multidispositivo de videojuegos

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

Agosto de 2022

Autor:

Jesús Manuel García Prieto

Tutores:

Diego J. Valdeolmillos Villaverde

Alfonso González Briones

Francisco Pinto Santos

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

El objetivo principal de este anexo es el de producir y describir un manual para que el usuario base pueda utilizar la aplicación con la mayor comodidad posible, de manera adicional, se incluirá una descripción de la aplicación del equipo fuente para que un usuario administrador pueda utilizarla también.

Comenzando por el manual de la aplicación de usuario, se irán describiendo los diferentes elementos de la misma vista por vista, describiendo también las funcionalidades de los mismos.

1.1. ESTRUCTURA DE LA INTERFAZ GRÁFICA

La interfaz gráfica de la aplicación (tanto la versión cliente como la versión fuente) se divide en las siguientes vistas:

- Vista de Inicio de sesión/Registro: Se trata de la vista inicial de la aplicación, en esta se invita al usuario a iniciar sesión en el sistema o a realizar su primer registro, una vez se inicia sesión/se registra por primera vez, esta pantalla no se vuelve a presentar al usuario a no ser que este realice un logout.
- Vista de retransmisión: Su principal uso será el de mostrar al cliente la pantalla retransmitida desde la fuente.
- Vista de catálogo: Sirve como catálogo de videojuegos, al seleccionar uno se comenzará a retransmitir la imagen del mismo.
- Vista de perfil: Zona personal del usuario donde el mismo puede consultar y modificar sus datos, así como una serie de funciones de gestión sobre su cuenta.

A las 3 últimas vistas se debe sumar la barra de navegación y la lista de amigos, común entre todas ellas.

Se procede a presentar un detalle de cada una de estas vistas con el fin de completar la descripción de las mismas con una imagen visual. Estos detalles se corresponden con las Figuras **Figura 1**, **Figura 2**, **Figura 3** y **Figura 4** que se pueden encontrar a continuación:

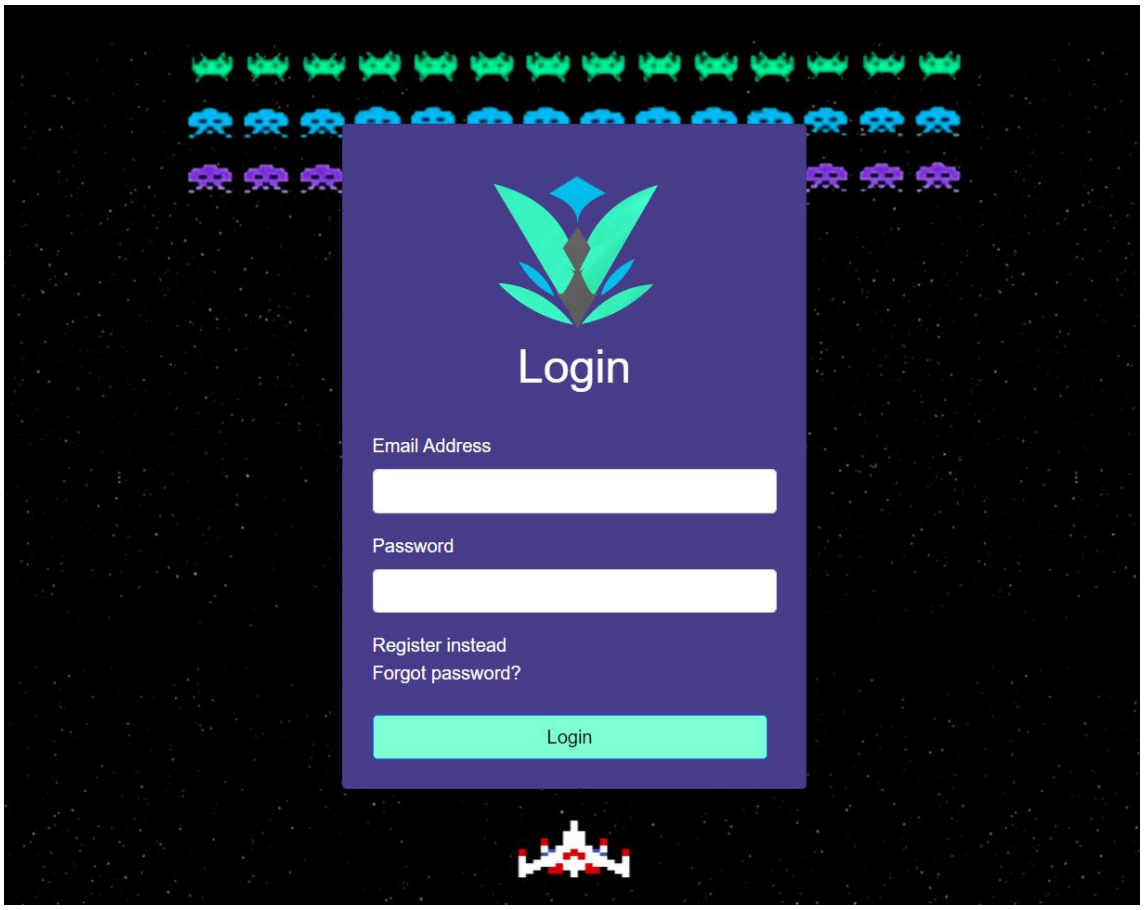


Figura 1. Detalle de la Vista de Login/Registro

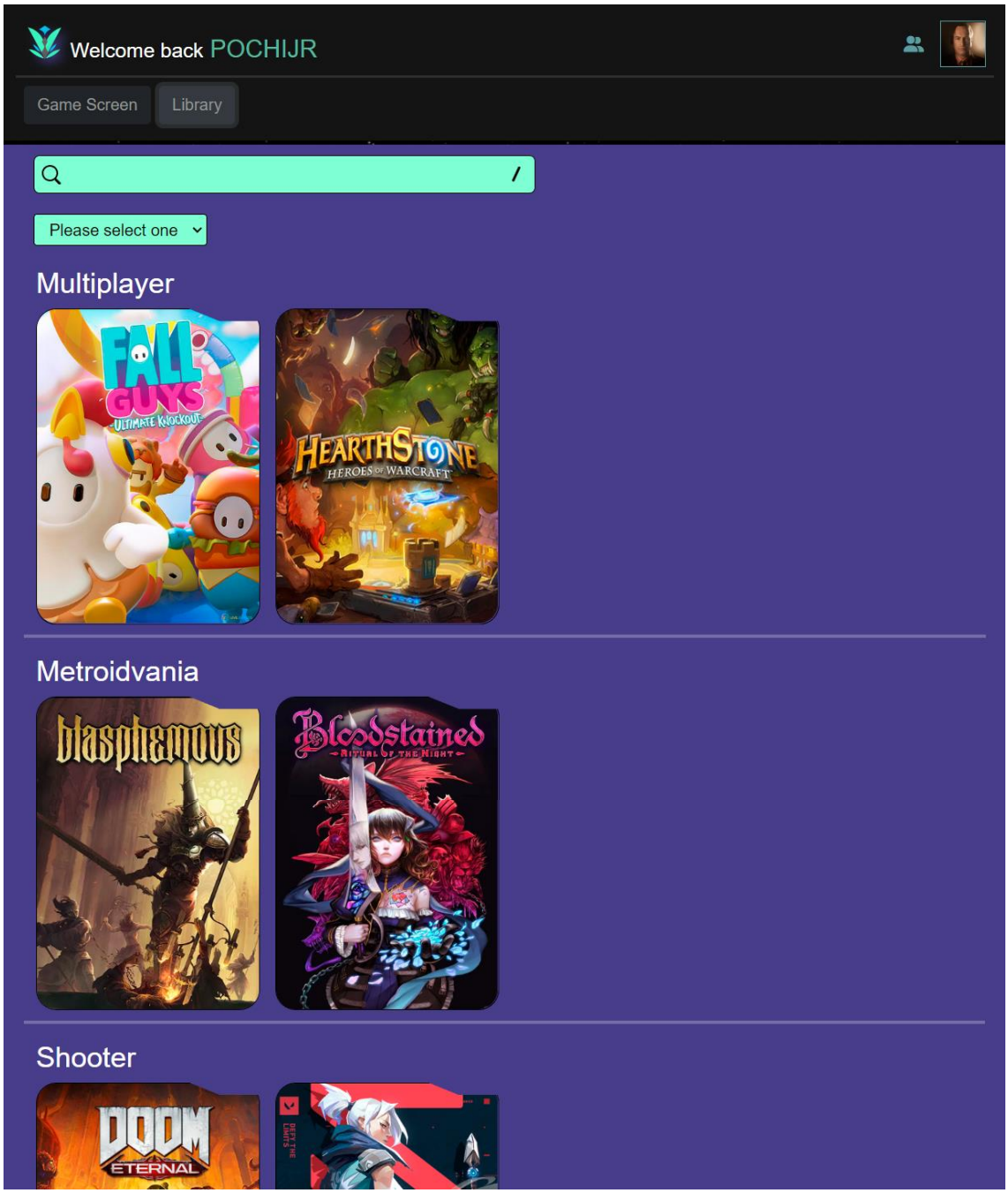


Figura 2. Detalle de la Vista de Catálogo

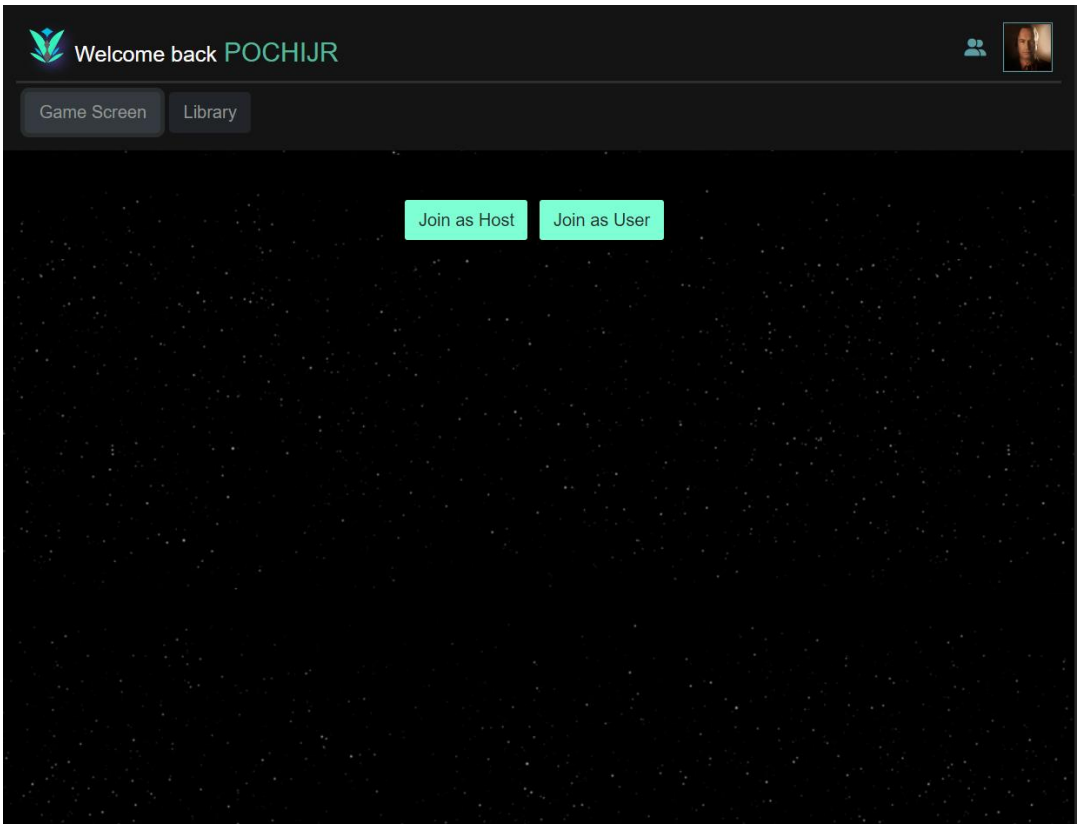


Figura 3. Detalle de la Vista de Retransmisión

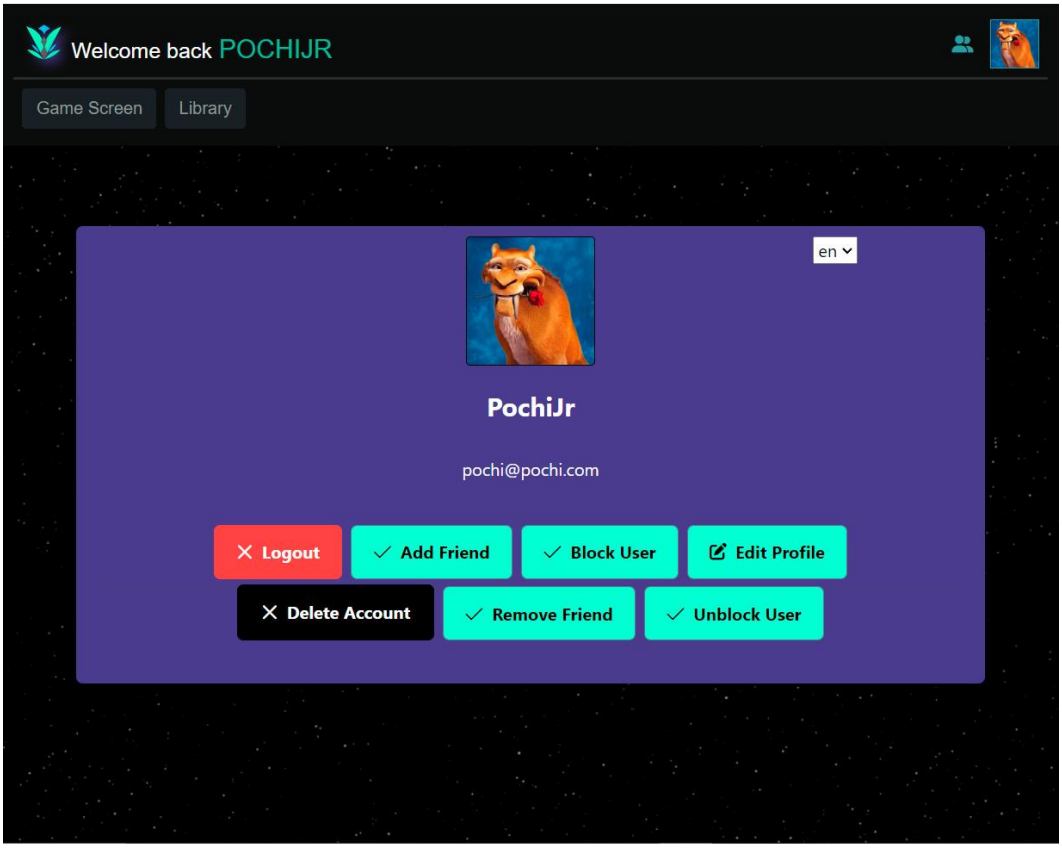


Figura 4. Detalle de la Vista de Perfil

Con el fin de aclarar la navegación entre estas vistas, se procede a incluir el diagrama de la **Figura 5** mostrando las diferentes opciones de navegación entre ellas:

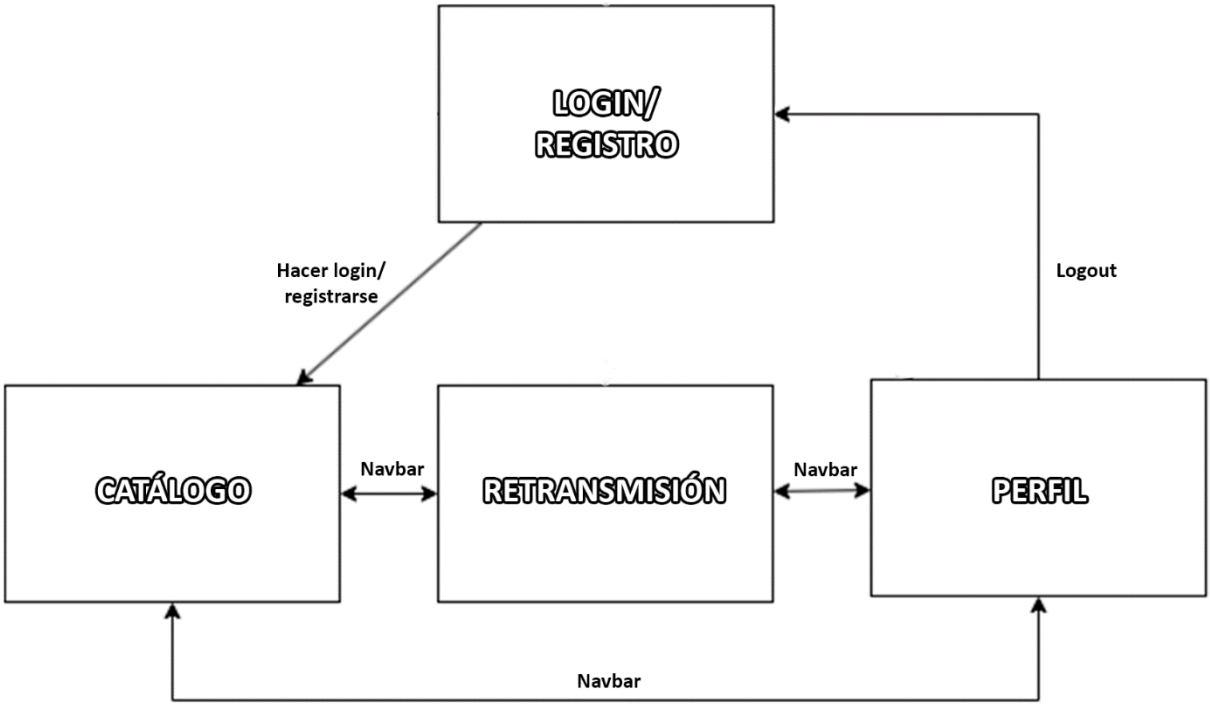


Figura 5. Diagrama de navegación

Como se puede observar, la mayoría de elementos navegacionales se encuentran en la barra de navegación o navbar, mientras que el acceso a la App se realiza en la vista de login/registro y la salida de la misma se realiza en el perfil mediante el botón de logout.

1.2. APLICACIÓN CLIENTE

Es ya en este apartado donde se explicará el uso y funcionalidad de los diferentes elementos de la interfaz gráfica de la aplicación. El orden de explicación será el ya detallado anteriormente, comenzando por la aplicación cliente se comenzará por la vista de login/registro, seguido de la vista de catálogo, la vista de retransmisión y, por último, la vista de perfil. A continuación, se hará lo mismo con la aplicación fuente.

1.2.1. VISTA DE LOGIN/REGISTRO

Para explicar este apartado se seguirá el orden de acción natural de un usuario: registro->login->recuperar contraseña.

1.2.1.1. REGISTRO

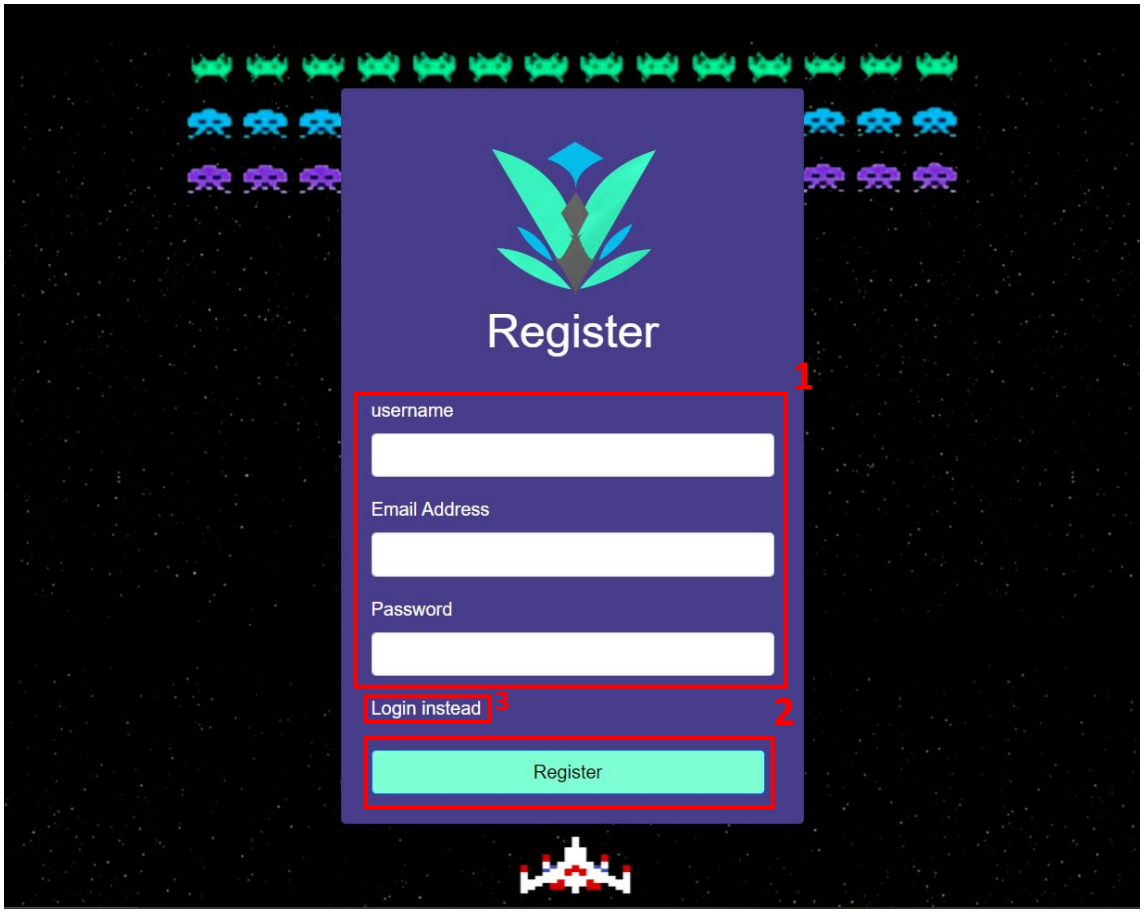


Figura 6. Vista de Login/Registro – Registro

Como se puede observar en la **Figura 6**, se han incluido una serie de recuadros en rojo numerados que servirán para describir con más precisión la vista de registro:

1. En esta sección se encuentran una serie de inputs, los cuales pueden ser rellenados por el usuario para introducir los datos con los que este será añadido al sistema.
2. Una vez el usuario rellena estos inputs, el botón de registro funcionará como envío de estos datos al sistema, el cual, tras unos segundos, dará acceso al usuario si la operación fuese válida.
3. Como último detalle, cabe destacar la posibilidad de navegar a la vista de login si el usuario así lo requiriese, esto sucede al presionar el texto de la sección 3.

1.2.1.2. LOGIN

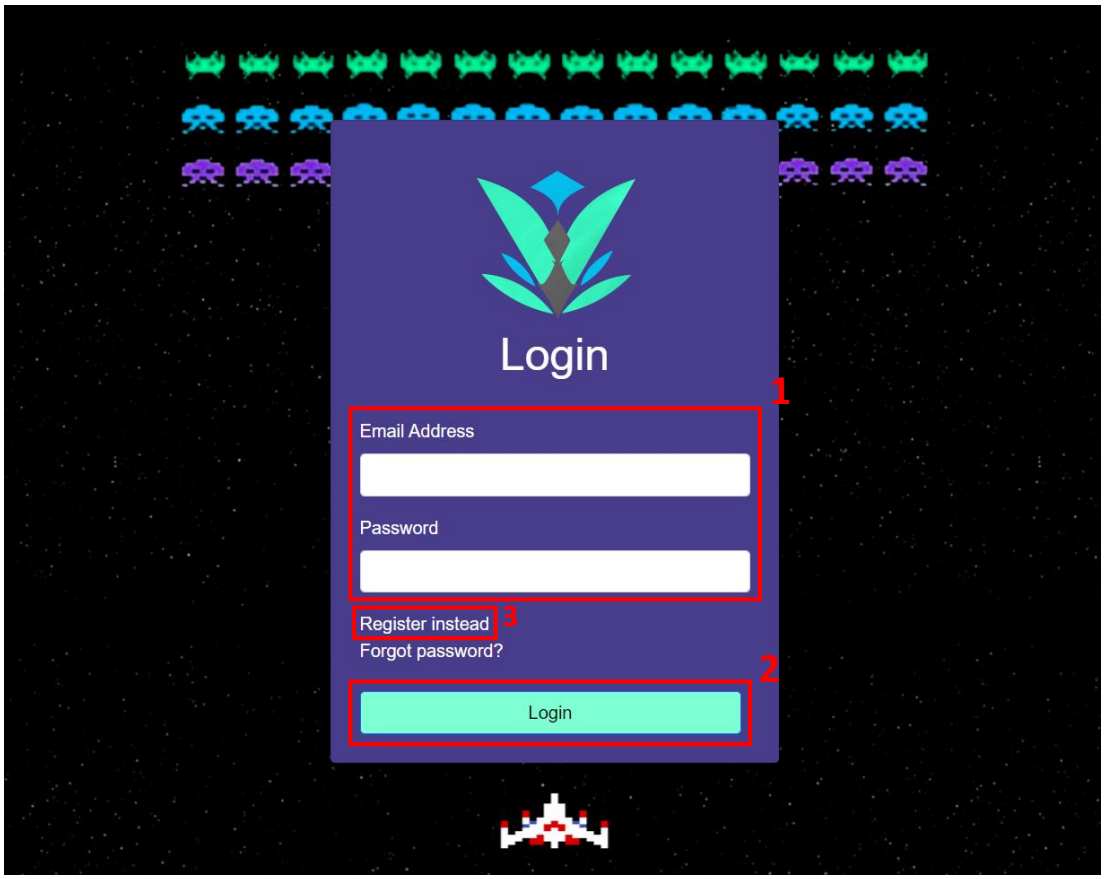


Figura 7. Vista de Login/Registro – Login

De manera muy similar al registro, la **Figura 7** se divide de nuevo en 3 secciones:

1. De nuevo, se ha incluido una sección de inputs en la que el usuario puede introducir sus credenciales, sin embargo, para el login bastará con el email y la contraseña del usuario.
2. Y, de nuevo también, se tiene un botón para confirmar los inputs introducidos y realizar el login, esto también dará acceso al usuario al sistema.
3. Al igual que en el registro, existe un botón para viajar a la vista opuesta, en el caso del login, este sirve para regresar a la vista de registro.

1.2.1.3. RECUPERAR CONTRASEÑA

Para acceder a esta funcionalidad, el usuario debe presionar el botón remarcado en la **Figura 8**, este era la última funcionalidad de la vista de login, sin embargo, pertenece al proceso de recuperación de contraseña, por eso se incluye en este apartado.

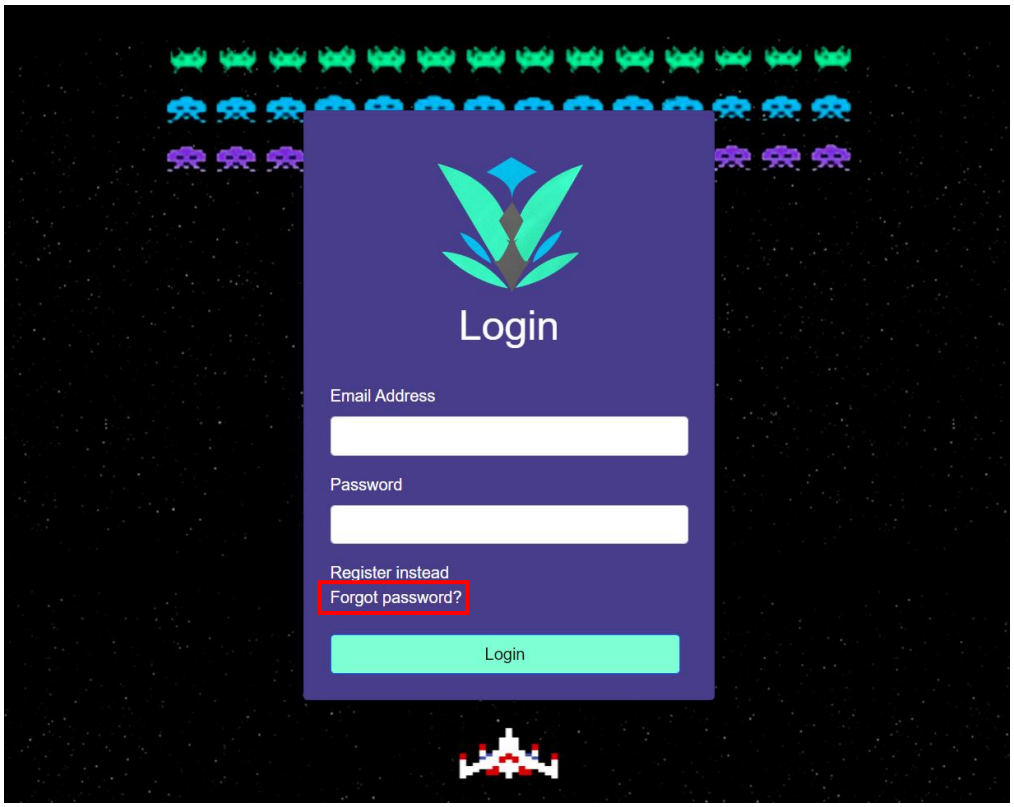


Figura 8. Vista de Login/Registro - Recuperar contraseña

Una vez realizada esta acción, se le presentará al usuario un nuevo input como el que se puede observar en la **Figura 9**, será aquí donde el usuario deba introducir su correo (el cual debe existir en el sistema) y un token será enviado a este.

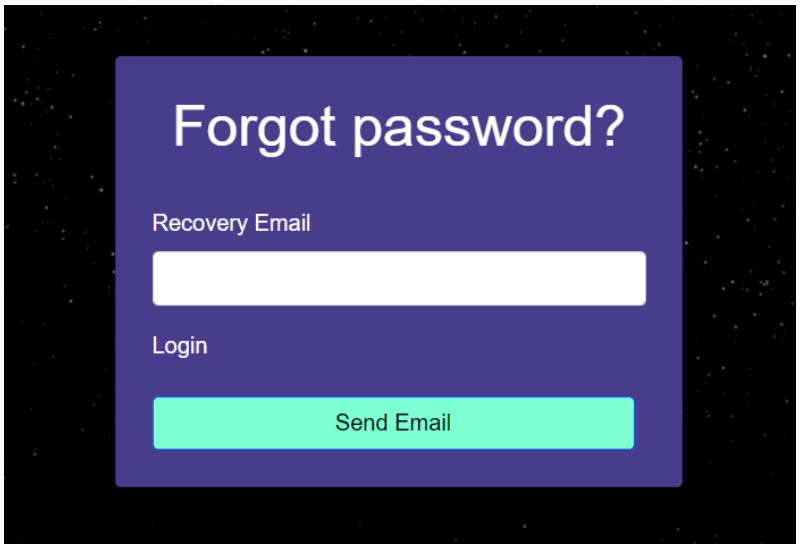


Figura 9. Vista de Login/Registro - Insertar correo

Figura 10. Vista de Login/Registro - Correo con token de recuperación

Una vez recibido este correo, cuyo contenido aparece en la **Figura 10**, el usuario puede regresar de nuevo a la aplicación e introducirlo en el nuevo input que habrá aparecido, tal y como se puede observar en la **Figura 11**, una vez hecho esto el usuario puede escribir su nueva contraseña y proceder a la actualización de la misma. Si el token es correcto la operación se realizará con éxito y el usuario podrá acceder al sistema con normalidad de nuevo.

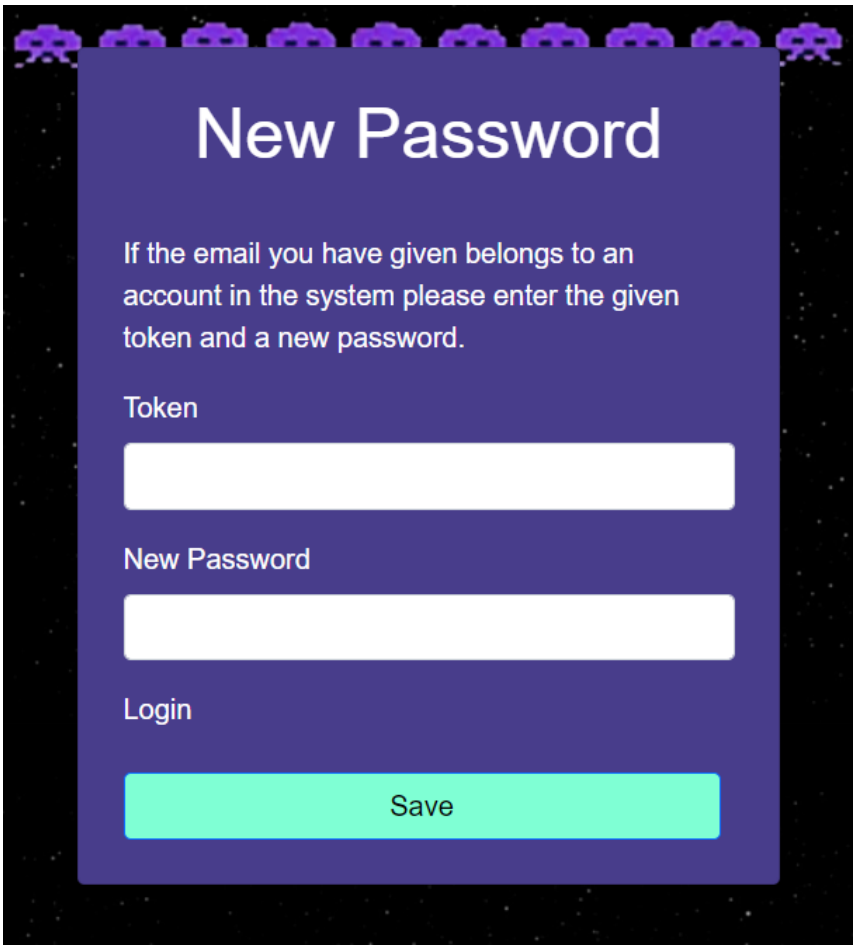


Figura 11. Vista de Login/Registro - Actualizar contraseña

1.2.2. VISTA DE CATÁLOGO

A diferencia del apartado anterior, en este el usuario es libre de realizar una serie de acciones en el orden que este prefiera, es por ello que el orden que se elija en este apartado será el que se puede observar en la **Figura 12** gracias a los recuadros numerados:

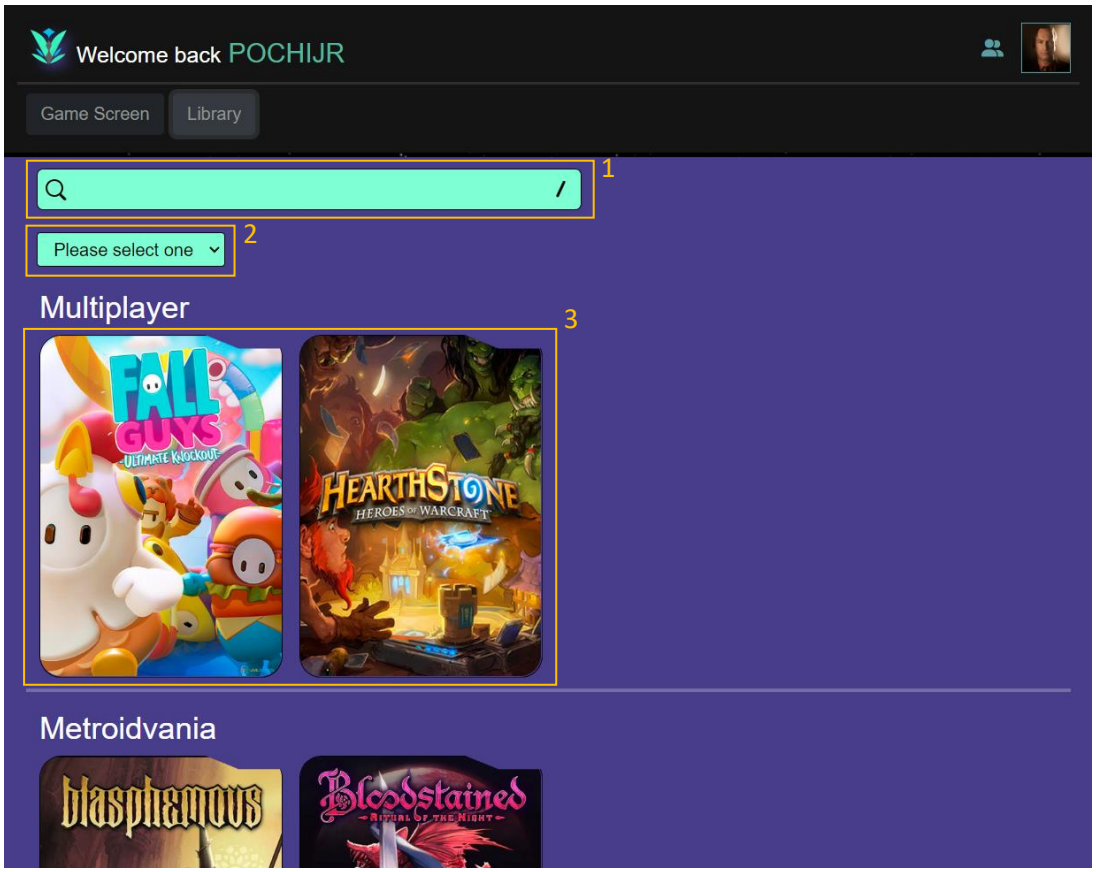


Figura 12. Vista de Catálogo - Numeración de apartados

1.2.2.1. REALIZAR BÚSQUEDA

Dentro de la **Figura 12** se puede observar, remarcada con el número 1, una caja de búsqueda o search box, es aquí donde el cliente puede realizar un filtro del catálogo de manera manual, escribiendo en esta caja el género del videojuego que les interese encontrar.

La búsqueda actualiza al catálogo de manera instantánea, pudiendo escribir únicamente "Shoo" para que aparezca el género "Shooter" como se puede observar en la **Figura 13**

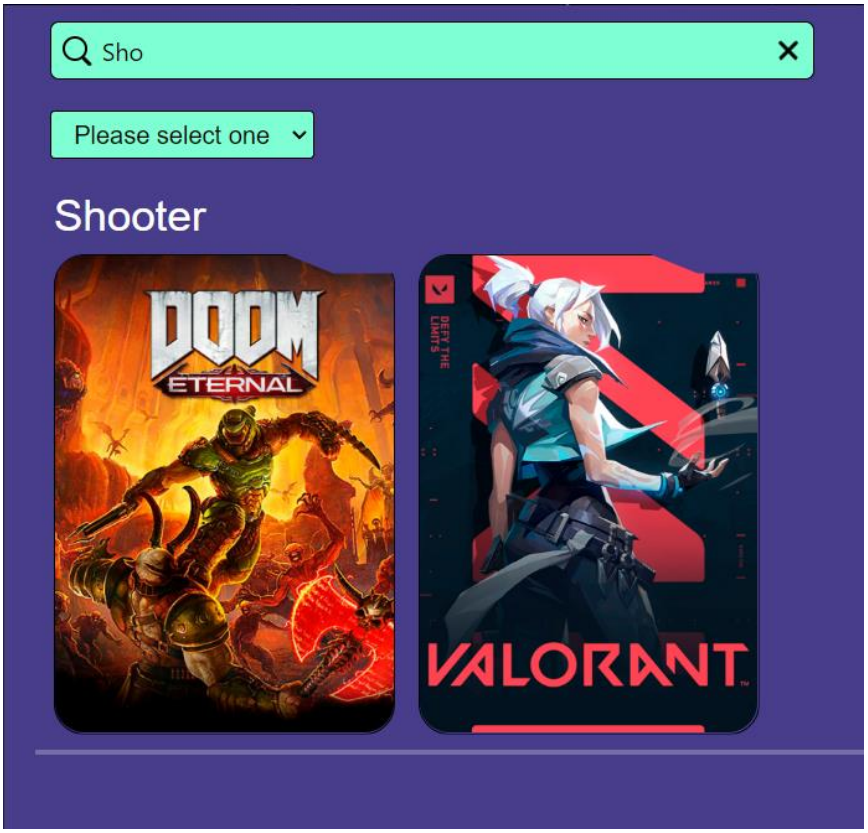


Figura 13. Diagrama de Catálogo – Búsqueda

Como nota final, cabe destacar que pulsando la tecla "barra lateral" se puede acceder directamente a la barra de búsqueda sin necesidad de hacer click sobre esta y, también cabe destacar, la posibilidad de pulsar el icono "X" que aparece en la barra para eliminar el input actual.

1.2.2.2. FILTRAR BÚSQUEDA

De nuevo, en la **Figura 12**, se puede ver el siguiente punto a tratar, en este caso será el filtro del catálogo. A diferencia de la caja de búsqueda, este filtro es estático, es decir, sus opciones derivan directamente del tipo de géneros que haya presentes en el catálogo, por tanto, es una búsqueda mucho más rápida pero menos precisa. Su funcionamiento consiste simplemente en hacer click sobre la misma y se mostrarán las diferentes opciones al usuario, el cual podrá elegir la que prefiera.

En la **Figura 14** podemos ver su funcionamiento

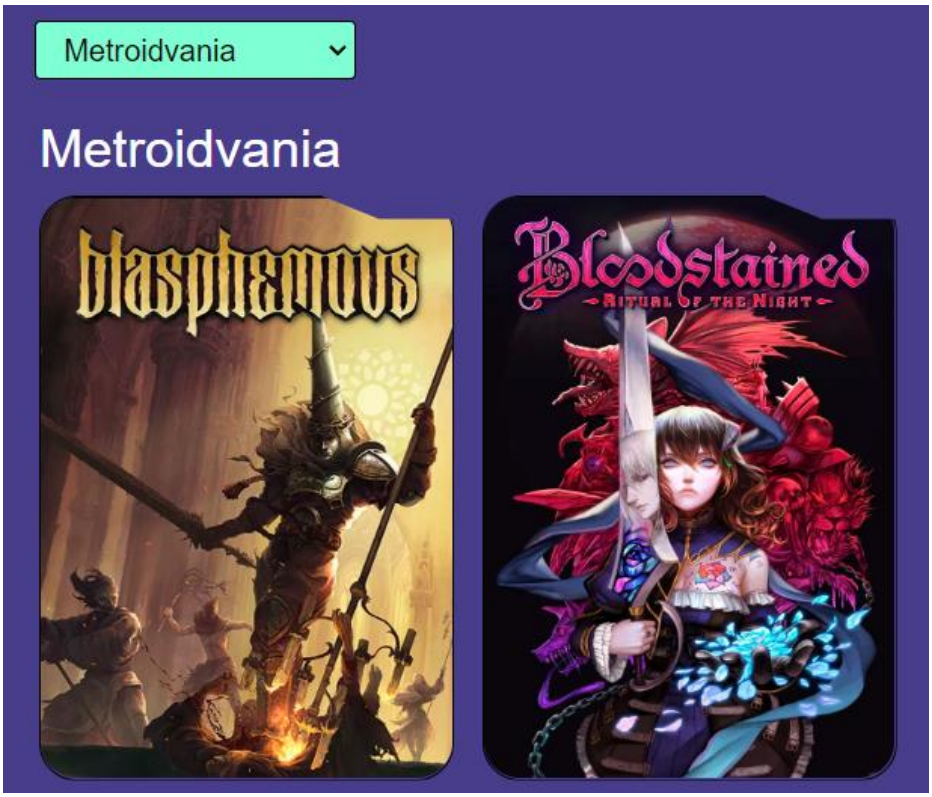


Figura 14. Vista de Catálogo – Filtrar

1.2.2.3. SELECCIONAR VIDEOJUEGO

Se trata de la funcionalidad principal del catálogo, como el propio nombre indica, el usuario puede seleccionar el videojuego que este prefiera para iniciar su retransmisión, sin embargo, esto pertenece a la vista de retransmisión, por lo que se hablará de ello allí.

1.2.3. VISTA DE RETRANSMISIÓN

Partiendo del detalle de la vista de retransmisión visto en la **Figura 3** se puede observar que se trata de la vista con menos elementos de todo el sistema, esto se debe a que esta simplemente cumple el propósito de mostrar al cliente la retransmisión del equipo fuente. Aún así, existen ciertas funcionalidades de las que se hablará a continuación:

1.2.3.1. INICIAR/RECIBIR RETRANSMISIÓN

A pesar de que el funcionamiento principal del proyecto sea el de permitir jugar a un videojuego de manera remota, existe la posibilidad adicional de usarlo para compartir la pantalla de manera remota, de manera similar a como lo realizan aplicaciones como TeamViewer, sin embargo, esto es algo abstraído para un usuario convencional, ya que es la propia fuente la que debe dar lugar a que esto suceda. Para poder lograrlo un equipo debe seleccionar el botón “Join as Host” y otro el botón “Join as Client”, estos se pueden observar remarcados en la **Figura 15.**

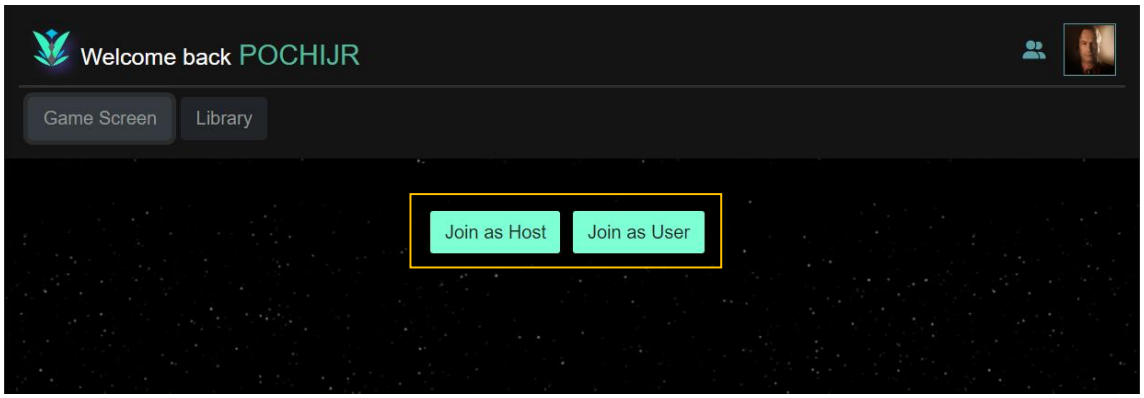


Figura 15. Vista de Retransmisión - Join as Host/Join as Client

Sin embargo, como ya se ha mencionado, este funcionamiento es completamente opcional. El flujo natural del sistema es el ya descrito en el apartado anterior: Seleccionar Juego -> Unirse a retransmisión.

1.2.3.1. FINALIZAR RETRANSMISIÓN

Una vez iniciada una retransmisión, el cliente puede salir de la misma en cualquier momento, el botón que realiza esta acción aparece debajo de la propia retransmisión en el momento en el que el cliente se une a esta, tal y como se observa remarcado en la **Figura 16**:

Figura 16. Vista de Retransmisión - Finalizar retransmisión

1.2.4. VISTA DE PERFIL

La vista de perfil se trata de la que más funcionalidades ofrece, sin embargo, la gran mayoría de ellas se resuelven en pocos pasos, por lo que, a pesar de ofrecer tantas acciones al usuario, se trata de acciones sencillas como añadir a un amigo simplemente introduciendo su correo.

En la **Figura 17** se puede observar el desglose de esta vista en cuanto a sus diferentes funcionalidades.

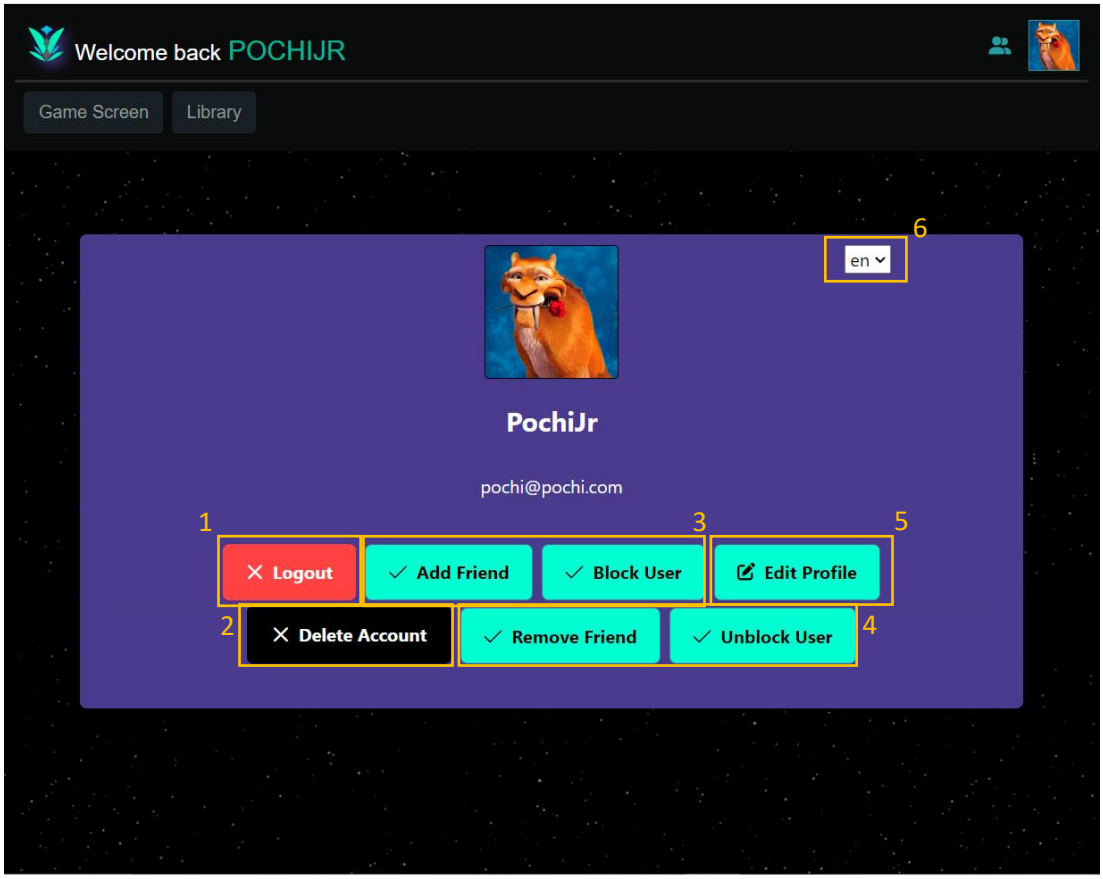


Figura 17. Vista de Perfil - Numeración de apartados

Veamos, según la división designada, los diferentes elementos de la vista de perfil:

1. En primer lugar, en rojo, tenemos el botón de logout, presionarlo hará que el usuario regrese a la pantalla de inicio de sesión con su sesión ahora cerrada.
2. En segundo lugar, en negro, tenemos el botón de eliminar cuenta, este botón, como su nombre indica, eliminará los datos del usuario por completo de la base de datos.
3. En tercer lugar, tenemos a la dupla añadir amigo – bloquear usuario, si bien parecen acciones opuestas, lo cierto es que a la hora de implementar su funcionalidad son idénticas, ya que ambas añaden a un usuario, dado su correo, a una lista, en el primer caso a la lista de amigos y en el segundo caso a la lista de bloqueados. En la **Figura 18** se puede observar la caja de diálogo que aparece al pulsar uno de estos dos botones
4. De igual manera que existe la opción de añadir amigos o bloquear usuarios, existe la opción inversa, el funcionamiento, de cara al usuario, es idéntico al caso anterior.
5. Este botón presenta al usuario la posibilidad de modificar su perfil, en la **Figura 19** se muestra el formulario que se presenta al usuario cuando este presiona este botón.

6. Por último, este cuadro de selección permite al usuario modificar el idioma de su aplicación, actualmente los dos idiomas soportados son el español y el inglés, en la **Figura 20** se muestra el resultado de cambiar el idioma de la aplicación al español.

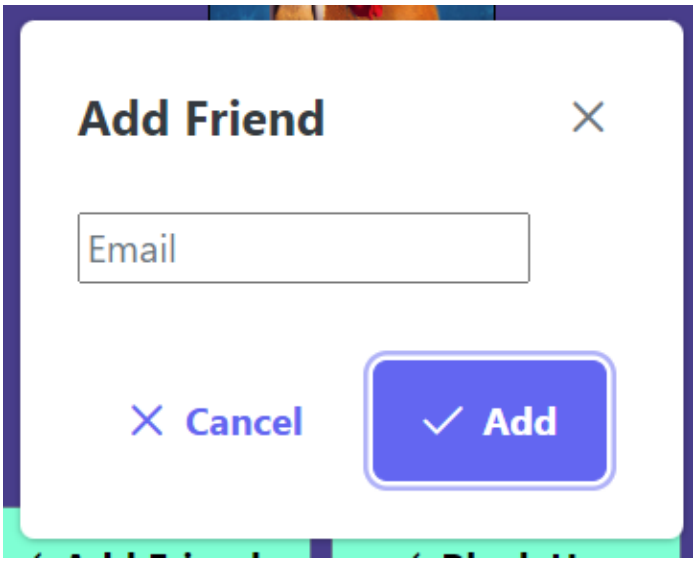


Figura 18. Detalle del cuadro de diálogo presente en varias acciones en la vista de perfil

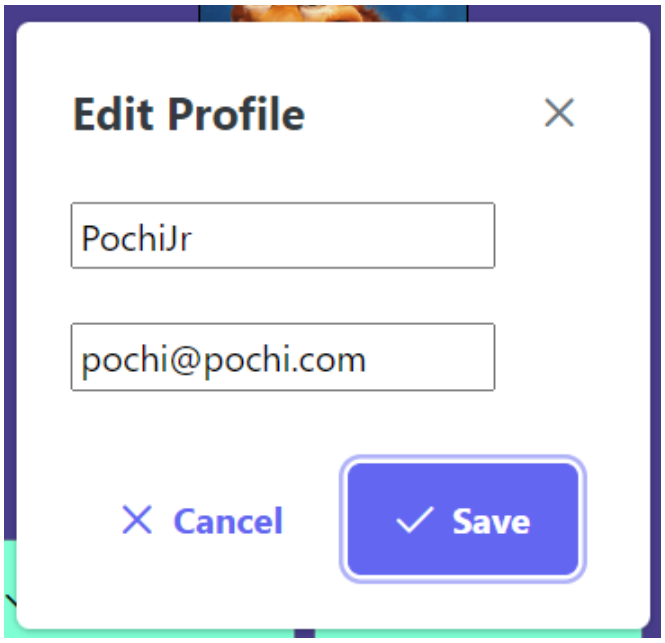


Figura 19. Detalle del cuadro de diálogo de modificación de perfil

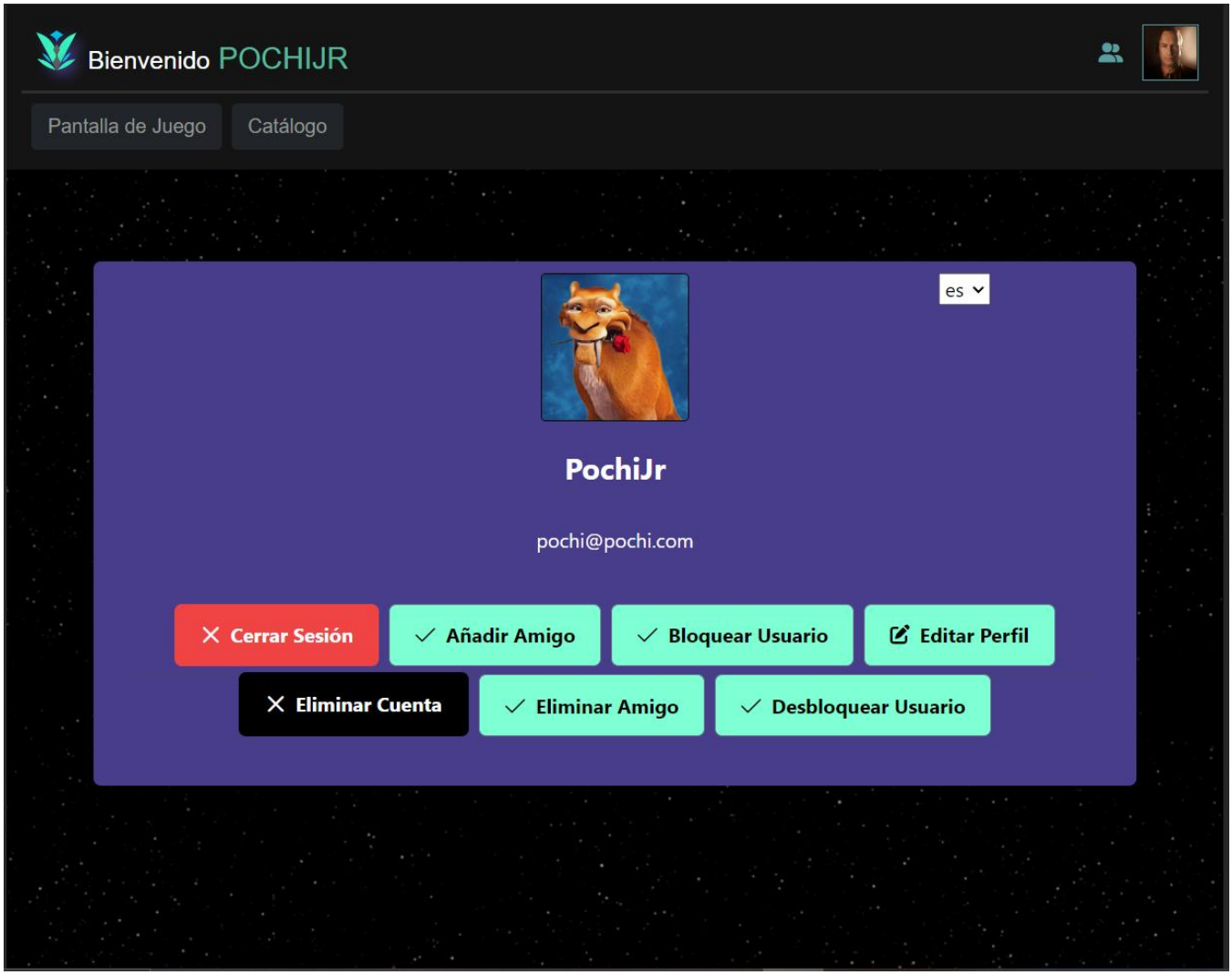


Figura 20. Detalle de la Vista de Perfil en español

1.2.5. BARRA DE NAVEGACIÓN

A pesar de no tratarse de una vista per sé, la barra de navegación es un elemento común a 3 de las 4 vistas del sistema, además de que esta posee varias funcionalidades que deben ser explicadas. En primer lugar, se presentará a modo de resumen un detalle de la barra de navegación segmentada por partes, para así facilitar su posterior explicación. Esto se puede observar en la **Figura 21**:

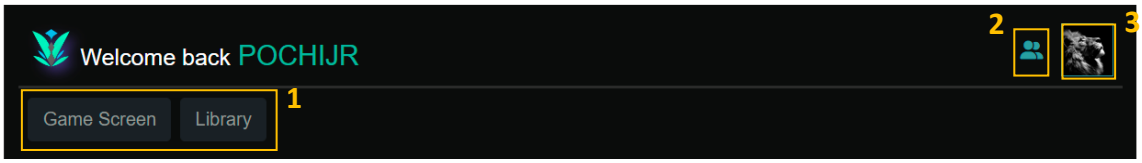


Figura 21. Barra de Navegación

1. En primer lugar, se presentan los botones de navegación, estos son los que permitirán al usuario navegar entre las diferentes vistas del sistema, en este caso, la vista de retransmisión y la vista de catálogo, respectivamente.
2. Este botón con icono de grupo se trata del acceso a la lista de amigos, al presionarlo esta será mostrada al usuario, se describirá en detalle en su respectiva sección.
3. Por último, la foto de perfil del usuario le permite navegar a su perfil propio.

1.2.6. LISTA DE AMIGOS

Se procede a presentar un detalle de la lista de amigos de la misma forma en que se presentó el catálogo. Podemos encontrarlo en la **Figura 22**:

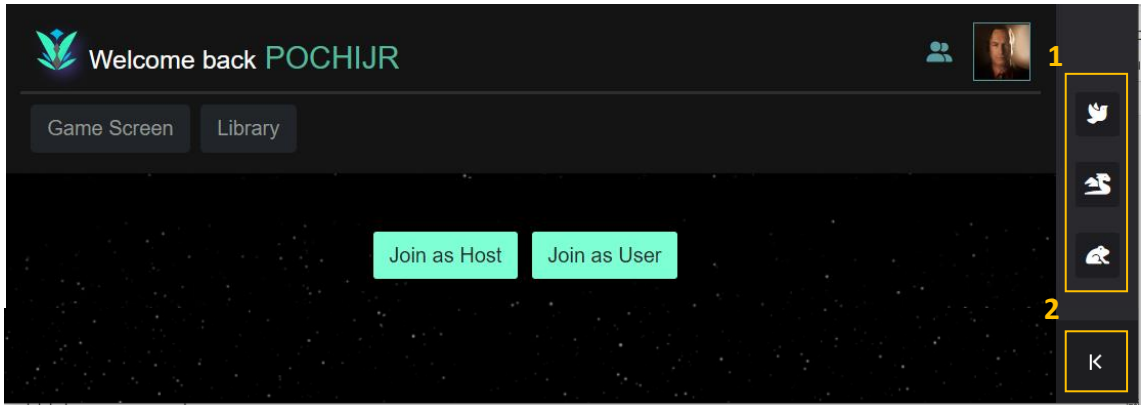


Figura 22. Lista de Amigos (Replegada)

Como se puede observar, la lista de amigos se encuentra presente en el momento en el que se presiona su icono en la barra de navegación, sin embargo, esta se encuentra replegada, se procede a explicar la funcionalidad de la misma en este estado:

1. Estos iconos muestran el número de amigos de la lista, si el usuario quisiera ver sus nombres sin necesidad de desplegar la lista de amigos podría colocar el cursor por encima del icono deseado y este mostraría el nombre del amigo en cuestión, como se puede ver en la **Figura 23**.
2. Este botón permite desplegar la lista, mostrando así los nombres de lo amigos que haya en esta

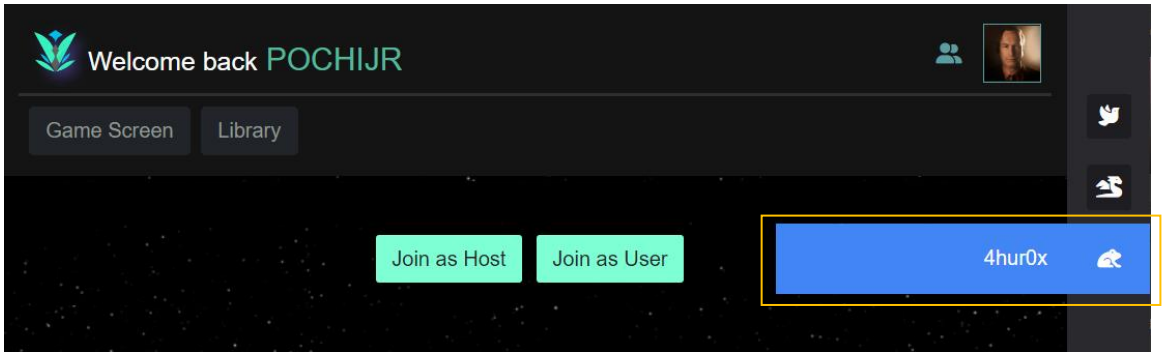


Figura 23. Lista de Amigos - Expandir nombre

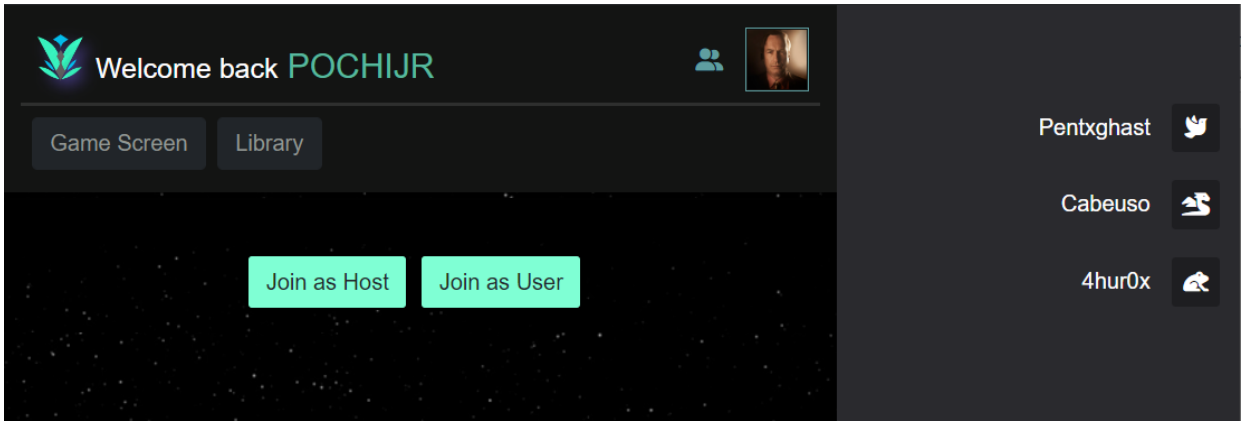


Figura 24. Lista de Amigos (Desplegada)

Por último, se puede observar en la **Figura 24** la versión desplegada de la lista de amigos, su funcionalidad es la misma que su versión replegada, con la única diferencia del botón inferior, que ahora realiza la función inversa.

1.3. APLICACIÓN FUENTE

En este apartado se presentarán aquellos elementos de la interfaz de la versión para equipos fuente del sistema, debido a que la gran mayoría de elementos comparten funcionalidad y uso con los de la interfaz del cliente, en este apartado solo se explicarán aquellos únicos para esta versión.

Como se puede observar en la **Figura 25** la única diferencia fundamental con la aplicación para usuarios es que la aplicación fuente no presenta una barra de navegación:

Figura 25. Detalle de la Vista de Retransmisión en la Aplicación Fuente