ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO WEB ALTERNATIVO DE VIDEOCONFERENCIA Y MENSAJERÍA BASADO EN WEBRTC

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

JORDY ADRIÁN RAMÓN BEDOYA

DIRECTOR: ING. PABLO HIDALGO, MSc.

Quito, marzo 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Jordy Adrián Ramón Bedoya, bajo mi supervisión.

Ing. Pablo Hidalgo, MSc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Jordy Adrián Ramón Bedoya, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Jordy Adrián Ramón Bedoya

DEDICATORIA

Al de arriba, que nunca me ha abandonado y me ha dado fuerzas para seguir luchando, por brindarme sabiduría y paciencia en los momentos más difíciles.

A mi persona, por tantos días llenos de esfuerzos.

Por tantos momentos de renuncia y continuación que me permitieron madurar.

¡¡¡¡Por todo lo que he dejado atrás para lograr este objetivo, y por demostrarme que las metas realmente las pone uno mismo!!!!

A los que me dijeron que no lo lograría, sus palabras me motivaron…gracias a ustedes lo he conseguido.

AGRADECIMIENTO

A Dios por guiarme en el camino.

A ti madre, por no soltarme en ningún momento, por agarrarme la mano mientras caía, por aquellos sermones que me hacen mejor ser humano, por aquellos cocachos que me hacían pisar tierra y por aquellos abrazos y besos en la frente que me hacían recordar que la vida es BONITA…. pero DURA.

A ti Michesito, por enseñarme a estar siempre con una sonrisa mientras enfrentamos los problemas, por mostrarme que el mejor camino para solucionar las cosas es la conversa… por brindarme la pasión de ser amarillo y jugar a la pelota desde el primer día y lo más importante, por enseñarme a orar para acercarme a Dios.

A ti hermano, que eres, has sido, y serás mi mejor ejemplo para seguir, por aquellas conversas que me abrían los ojos, por los contados pero sinceros abrazos que me decían que estabas ahí, por enseñarme que las metas, las pongo yo…lo reconozco… no lo hubiera logrado sin ti. P.D: No te crezcas jaja

A ti Marley, por brindarme tu apoyo cuando más lo necesitaba, por mirarme a los ojos y decirme que no estoy solo, ser mi soporte en mis caídas más fuertes, y por enseñarme que la vida continua.

A mis amigos. Anderson, Carolina, Sebas, Samantha, Elizabeth, por grandes momentos universitarios, por risas que alegraban el día y por las mejores chumas de mi vida.

A usted Ingeniero Hidalgo, por encontrar mil y un peros a mi trabajo de titulación pues eso me ayudaba a mejorar, por enseñarme a defenderme ante cualquier situación y por ser un gran amigo para mí y la familia.

Finalmente, a ti que estas leyendo esto y tengo que agradecerte por tantos momentos compartidos, por mostrarme lo que es vivir y por enseñarme a ser mejor en todo aspecto, sí que supimos vivir……

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

El mundo está en un proceso de transición que transforma a la sociedad industrial en una sociedad digital[1]. En la actualidad las relaciones laborales, comerciales, y educativas, necesitan establecer comunicaciones. El presente trabajo plantea el diseño de un prototipo que trabaja en ambiente Cloud, para desarrollo de videoconferencias mediante el uso de la tecnología WebRTC y un servidor STUN/TURN.

En el primer capítulo se presenta la información teórica acerca de los sistemas de videoconferencia y de las comunicaciones centradas en la web, conceptos claves de la tecnología WebRTC y su arquitectura, así como, las características de Node.Js, HTML 5, Socket.IO y STUN/TURN que se utilizaron para el desarrollo de la aplicación.

El segundo capítulo expone la metodología utilizada para el diseño, codificación e implementación del prototipo. En el diseño se definen necesidades, módulos y componentes del sistema por medio de entrevistas a usuario, en conjunto con diagramas UML y Scrum; en la codificación se elaboran los módulos diseñados anteriormente, con lenguaje de programación JavaScript y, por último, para la implementación del prototipo se hizo uso de las plataformas digitales Heroku, DigitalOcean, y GoDaddy.

El tercer capítulo plasma los resultados y discusión a través de las pruebas realizadas en los diversos escenarios, comprobando el funcionamiento y uso adecuado del sistema.

En el cuarto capítulo constan las conclusiones y recomendaciones que se extrajeron una vez finalizado el proyecto.

Finalmente, en la sección de anexos se encuentran los códigos desarrollados en este trabajo de titulación, las preguntas de la entrevista y el manual de usuario para la aplicación.
PALABRAS CLAVE: WebRTC, STUN/TURN, Videoconferencia, Mensajería, Compartición.

ABSTRACT

The world is in a transition process that transforms the industrial society into a digital society [1]. Nowadays, labor, commercial, and educational relationships need to establish communications. The present work proposes the design of a prototype that works in a Cloud environment, for the development of videoconferences using WebRTC technology and a STUN/TURN server.

The first chapter presents the theoretical information about videoconferencing systems and web-centric communications, key concepts of WebRTC technology and its architecture, as well as the features of Node.Js, HTML 5, Socket.IO and STUN/TURN that were used for the development of the application.

The second chapter presents the methodology used for the design, coding and implementation of the prototype. In the design, needs, modules and components of the system were defined by means of user interviews, together with UML and Scrum diagrams; in the coding, the previously designed modules were developed with JavaScript programming language; and finally, for the implementation of the prototype, the digital platforms Heroku, DigitalOcean, and GoDaddy were used.

The third chapter presents the results and discussion through the tests performed in the various scenarios, verifying the proper functioning and use of the system.

The fourth chapter contains the conclusions and recommendations drawn at the end of the project.

Finally, the annexes section contains the codes developed in this degree work, the interview questions and the user's manual for the application.

KEYWORDS: WebRTC, STUN/TURN, Videoconferencing, Messaging, Sharing.

1 INTRODUCCIÓN

Los sistemas de videoconferencia han ganado un espacio importante dentro de las comunicaciones interpersonales, más aún en situaciones como las que se está atravesando con la actual pandemia del covid-19, la cual ha paralizado las actividades presenciales, incrementando la necesidad del uso de sistemas de videoconferencia, para así lograr comunicaciones digitales en ambientes laborales, personales y educativos.

El presente proyecto tiene como finalidad desarrollar un prototipo alternativo de videoconferencias con el uso de la tecnología WebRTC, proporcionando al usuario una opción para tener comunicación en tiempo real, la cual no tendrá la necesidad de instalar componentes adicionales o realizar compra de membresías para su uso, evitar el uso de espacio de almacenamiento en disco, y ofrecer sesiones que no dispongan de un límite de tiempo.

1.1 OBJETIVOS

Objetivo general

- Desarrollar un prototipo web alternativo de videoconferencia y mensajería basado en WebRTC.

Objetivos específicos

- Analizar la fundamentación teórica necesaria para la realización del proyecto.
- Diseñar los módulos que formarán parte del prototipo.
- Implementar los módulos diseñados.
- Analizar los resultados en base a las pruebas ejecutadas.

1.2 ALCANCE

El proyecto propuesto integra distintos componentes de software para su realización. Se genera un prototipo para videoconferencia, mensajería y compartición de pantalla entre los clientes, haciendo uso de los navegadores Google Chrome, Mozilla Firefox y Opera. El prototipo está conformado por un servidor de señalización alojado en un máquina virtual en la nube, una aplicación web accesible mediante un enlace de Internet, el servicio de Heroku para el alojamiento de la aplicación, el servicio de *hosting* (GoDaddy) con el uso de un dominio seguro que permite el acceso de clientes de diversas redes.

Debido a las limitaciones que tienen las versiones gratuitas de los servicios antes mencionados, se adquiere un dominio y se alquila anualmente una máquina virtual (básica), para la implementación del prototipo.

Se realiza la búsqueda de información acerca de WebRTC, sus características y componentes, para entender su correcto funcionamiento y uso, así como se obtiene información de cuáles son los ambientes donde WebRTC funciona de manera óptima.

El desarrollo de software utiliza como base la metodología ágil Scrum, gracias a la flexibilidad que brinda este sistema para la mejora continua de la aplicación; el prototipo es elaborado de manera incremental e interactiva [9].

Se recaba información del funcionamiento de diversos sistemas de video conferencia, cómo funcionan éstos y qué funciones realizan. Se realizan entrevistas a usuarios de estos servicios, para conocer sus preferencias y aplicarlas al momento del diseño del prototipo.

Se estudia el lenguaje de programación “JavaScript”, el cual es utilizado en conjunto y de manera sencilla con HTML (*HyperText Markup Language)* para el contenido y estructura de páginas web, así como CSS (*Cascading Style Sheets*) para el diseño y visualización de las páginas web; además se usa el entorno de ejecución de JavaScript Node.Js.

La aplicación se desarrolla por módulos donde se distinguen los siguientes: módulo de Conexión, módulo de Presentación, módulo de Mensajería, módulo de Controles y módulo de Señalización que tendrá el servidor STUN/TURN.

Se instala y configura un servidor STUN/TURN que toma la información de los clientes (Dirección IP pública, tipo de NAT *(Network Address Translation*), y el puerto de Internet asociado por el NAT), para configurar la comunicación UDP (*User Datagram Protocol*) entre los clientes y establecer la llamada; de igual manera controla la correcta señalización entre clientes, además de garantizar la comunicación exitosa *peer-to-peer* entre estos. Adicionalmente, no existe la necesidad de desactivar el *firewall* o antivirus de ningún dispositivo.

El servidor es instalado y configurado sobre Linux en la distribución “UBUNTU” en la versión 18.04, está centralizado y se lo ubica en la nube, en una máquina virtual alquilada (Digital-Ocean); una vez realizada la correcta conexión entre los clientes, el servidor habrá cumplido su función.

El escenario de pruebas (Figura 1.1), consta de clientes (mínimo 2 - máximo 4) utilizando los navegadores Google Chrome, Mozilla Firefox y Opera, de una plataforma de servicios en la nube-PaaS (*Platform as a Service*) en la cual se aloja la aplicación de

videoconferencia, así como los servicios web, y de un servidor que aloja los servicios de señalización. Con respecto a la compartición de pantalla solo un usuario podrá compartir pantalla al resto de participantes; no se realizará transferencia de archivos.

Cabe recalcar que el prototipo requiere de una conexión mínima a Internet del usuario de 3 Mbps.

De igual manera en caso de pérdida de conexión a Internet por parte de un usuario, este se verá eliminado de la sesión, y tendrá que volver a acceder al *link* de ésta, ya que no existe moderador alguno en las sesiones.

El prototipo será funcional para los navegadores Chrome (56 o superior), Mozilla Firefox (44 o superior) y Opera (43 o superior).

El proyecto está específicamente orientado a computadoras; la ejecución para celulares no está considerado en este trabajo de titulación.

El proyecto tiene un producto final demostrable.

En la figura 1.1 se presenta el escenario de pruebas.

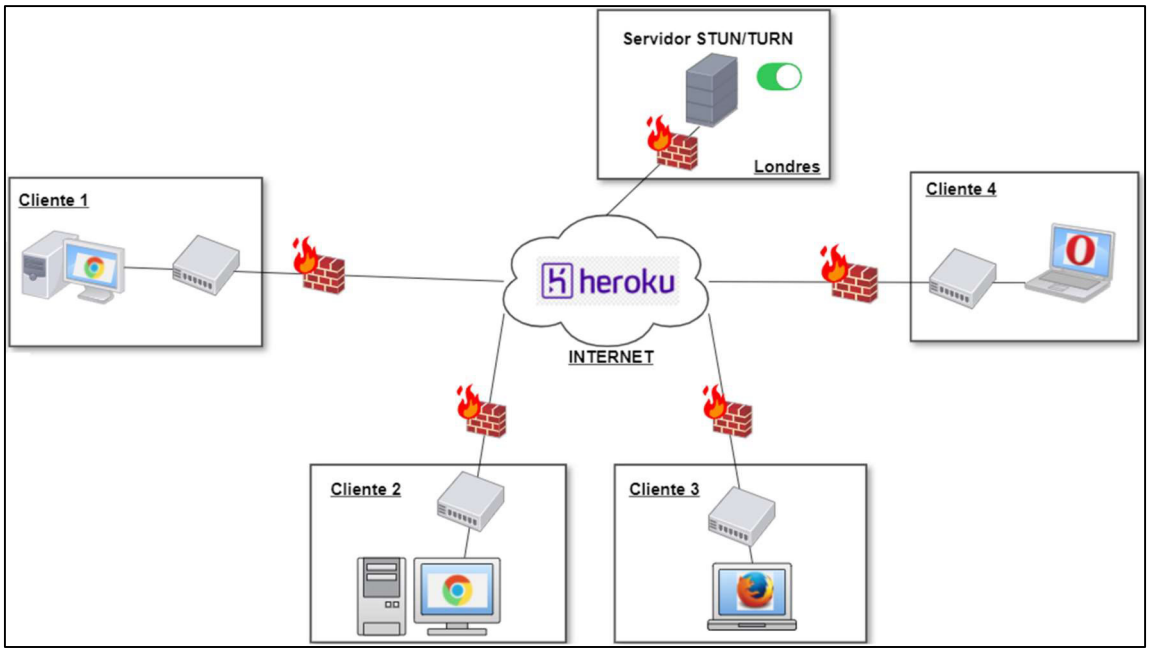


Figura. 1.1 Escenario de pruebas

1.3 MARCO TEÓRICO

1.3.1 SISTEMA DE VIDEOCONFERENCIA

Internet como red de conexión informática permite que las personas intercambien información a través de distintas plataformas web. Por lo cual, es necesario mencionar que los sistemas de videoconferencia son un gran referente para la comunicación en tiempo real, porque estos sistemas también han facilitado que los usuarios superen barreras de distancia para mantener una conversación virtual (no presencial), mediante la transmisión de audio y vídeo en tiempo real. Así, de acuerdo con la necesidad de los participantes, éstos podrán compartir la pantalla para hacer uso de la pizarra electrónica, salas de trabajo, entre otras herramientas. [2].

1.3.1.1 Cloud Computing

El mundo actual ha evolucionado y el concepto de computación no es la excepción; La computación en la nube (*Cloud Computing*) es considerada la nueva tendencia en las tecnologías de la información.

Cloud computing permite acceso remoto a softwares, almacenamiento de archivos y procesamiento de datos por medio de Internet, mostrándose como una alternativa a la ejecución en una computadora personal o servidor local. Para *cloud computing*, no existe la necesidad de instalar aplicaciones localmente en computadoras[3].

La computación en la nube se apoya principalmente de servidores remotos y contenedores de software en la Internet.

- **Contenedores en la Internet**

El diseño y creación de la mayorías de aplicaciones actuales se dan dentro de contenedores, ofreciendo a los desarrolladores flexibilidad al gestionar aplicaciones de gran escala.

Los contenedores son tecnología que se usa para agrupar una aplicación con todos sus archivos necesarios en un entorno de ejecución. Como una sola unidad, el contenedor puede moverse con facilidad y ejecutarse en cualquier sistema operativo en cualquier contexto[4].

1.3.1.2 Tipos de Videoconferencias

- **Basadas en la web**

Las soluciones basadas en la web permiten mantener una comunicación en tiempo real con múltiples usuarios, sin necesidad de instalar componentes adicionales en el dispositivo del usuario. Un gran número de soluciones, también se caracterizan por utilizar la tecnología de WebRTC (Web Real-Time Communications), que mediante la ejecución de navegadores web facilitan el intercambio de texto, datos, audio y vídeo [5].

- **Basadas en Software**

Las soluciones basadas en software son aquellas que para su funcionamiento requieren de archivos instalables. De esta manera, permiten al usuario que mediante el uso de su dispositivo puedan conectarse a una videoconferencia. Entre las soluciones más conocidas están Zoom, Microsoft Teams, Slack, cabe destacar que en la actualidad Zoom es la solución más utilizada a nivel mundial, porque ocupa el primer puesto en el cuadrante de Gartner para soluciones de videoconferencia [6].

1.3.2 COMUNICACIONES CENTRADAS EN LA WEB

1.3.2.1 Navegadores web

Un navegador web es un software que permite al usuario acceder a la búsqueda de información a través de Internet. Así, para que esta búsqueda sea eficiente, los navegadores emplean principalmente los siguientes protocolos: HTTP (*HyperText Transfer Protocol*), HTTPS (*HyperText Transfer Protocol Secure*), FTP (*File Transfer Protocol*), los cuales traducen las páginas web, para finalmente mostrar la información.

Por eso, para que el proceso expuesto se lleve a cabo, las páginas web deben estar escritas en HTML (*HyperText Markup Language*), código que es identificado por una dirección única URL (*Uniform Resource Locator*), que luego es mostrado en la WWW (*World Wide Web*); y así la información es referida al usuario, quien actualmente puede disponer de múltiples navegadores web como:

- *Google Chrome:* navegador web gratuito creado y desarrollado por Google, que cuenta con más de 47 idiomas disponibles y un motor de búsqueda basado en WebKit, V8 y Blink. Es compatible con sistemas operativos como Windows, macOS, Linux, Android y iOS. Según estadísticas hasta abril del 2021, Chrome es el líder de navegadores web de escritorios con el 65.2% de uso a nivel mundial [7].

- *Mozilla Firefox:* navegador web libre y de código abierto, creado por la Corporación Mozilla; cuenta con más de 80 idiomas disponibles y un motor de búsqueda basado en Gecko, Quantum y SpiderMonkey [8]. También es funcional con los sistemas operativos de Windows, macOS, Linux, Android y iOS.

- *Opera:* desarrollado por Opera Software, este navegador es gratuito, además de ser compatible con sistemas operativos como: Windows, macOS y Linux [9]. También cuenta con un motor de búsqueda basado en Blink y Presto, pionero en tecnología de acceso rápido (*SpeedDial*).

1.3.2.2 Aplicación web

Una aplicación web es un programa informático utilizado por múltiples usuarios, quienes haciendo uso de navegadores o *browsers* acceden a Internet, para adquirir respuesta de lo investigado. [10]. Para que estas aplicaciones web sean funcionales hacen uso del protocolo HTTP, mediante el cual se realizan peticiones que son atendidas por el servidor web para responder con el servicio solicitado.

Así, estas aplicaciones constan en 3 partes fundamentales: conexión de red, servidor y cliente; las cuales hacen uso entre otros de *scripts* en el servidor para manejar la información de manera óptima y de *scripts* en el cliente para presentar el servicio a los usuarios. Todo este proceso es necesario porque las aplicaciones web están basadas en el modelo cliente-servidor[11].

De esta manera, también es necesario destacar que las aplicaciones web tienen una funcionalidad de uso comercial y personal, que promueve la productividad en la comunicación a larga distancia, servicios médicos, teletrabajo, motores de búsqueda, entre otros.

Las aplicaciones web comúnmente se modelan mediante el modelo arquitectural por capas, que se menciona a continuación.

- **Arquitectura lógica de capas**

En una arquitectura basada en capas, cada capa trata la información de manera individual, facilitando así, la separación de las tareas que componen una aplicación web con modelo tipo cliente-servidor.

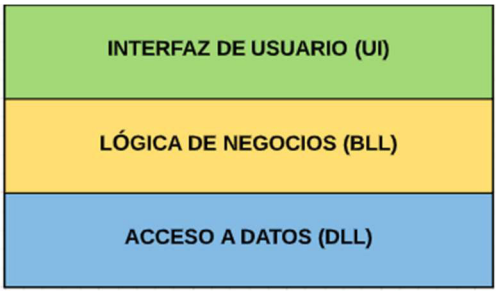


Figura. 1.2 Arquitectura de 3 capas *[12]*

- **Arquitectura de tres capas**

Este es uno de los modelos más utilizado para las aplicaciones web. Por eso, es importante describir la estructura de sus capas que se observa en la Figura 1.2.

- **Capa Interfaz de usuario (UI)**

Se caracteriza por mantener la comunicación con la capa Lógica de Negocios, estableciendo así la interacción entre el usuario y el sistema, con la finalidad de mostrar la información de manera amigable y entendible al consumidor.

- **Capa Lógica de Negocios (BLL)**

La funcionalidad del sistema reside en esta capa, porque es la encargada de gestionar y procesar la información, así como de enviar la respuesta al cliente. Por eso, una vez realizada la petición, este se comunica tanto con la capa de Interfaz de usuario (peticiones), como con la capa de Acceso de Datos (almacenamiento).

- **Capa Acceso a Datos (DLL)**

Esta capa no tiene contacto directo con la interfaz de usuario, pero se encarga de almacenar y contener los datos del cliente, para posteriormente entregar los datos a la capa Lógica de Negocios. En la figura 1.3 se detalla la relación de estas capas.

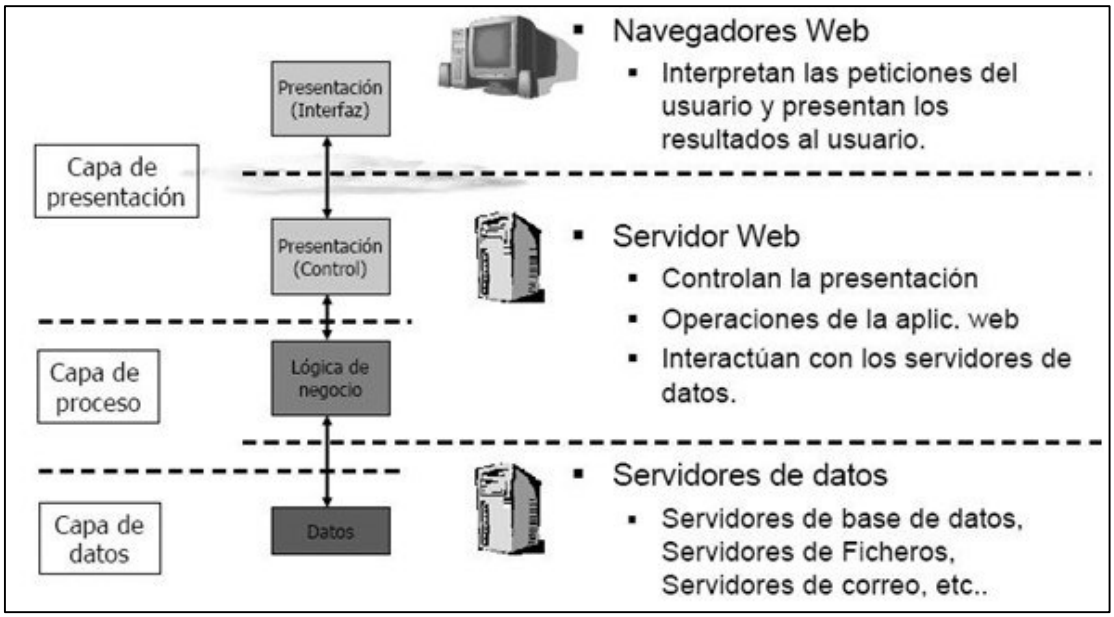


Figura. 1.3 Relación de capas en la arquitectura de 3 capas [10]

1.3.3 TECNOLOGÍA WEBRTC

1.3.3.1 Introducción

WebRTC tiene su nacimiento en mayo del 2010, cuando Google adquiere la compañía "Soluciones IP Globales" también conocida como GIPS, la cual era una empresa encargada de desarrollar códecs de voz y video. Meses después, en el mismo año Google compra a la empresa "On2", que se encargaba de facilitar el desarrollo y uso de aplicaciones de videollamadas o de VoIP[13].

Así, al finalizar el apropiamiento de las empresas, Google, liberó toda la información que pertenecían a GIPS y a On2, cambiando las tecnologías desarrolladas por éstas. Es decir, se trasladó de software propietario, a OpenSource. Para, mayo del 2011, Google en conjunto con IETF (*Internet Engineering Task Force*) y W3C (*World Wide Web Consortium*), añaden una API (Application Programming Interface) de JavaScript, la cual permitió comenzar labores en navegadores web, que dio paso al lanzamiento de una tecnología *opensource* conocida como WebRTC, la cual se enfocó en ser un proyecto basado en navegadores que accedan a la comunicación en tiempo real. Con este propósito, IETF fue el encargado de estandarizar los protocolos de la tecnología, mientras que W3C se encargó de la API del proyecto; su lanzamiento comercial se dio en febrero del 2013.

Una vez realizado el lanzamiento de WebRTC, Google mencionó: "Esta es nuestra contribución inicial para lograr la misión de hacer que el audio y el video estén disponibles

en todos los navegadores, a través de un conjunto uniforme estándar de API. Esta versión inicial proporcionará la funcionalidad que prevemos para WebRTC"[14].

De esta manera, WebRTC se convierte en una tecnología de estándar abierto que admite voz, video y datos genéricos entre pares; los principales componentes de la API de WebRTC son definidas y desarrolladas por W3C en conjunto con IETF [15].

Para proporcionar una comunicación en tiempo real, ésta es funcional en navegadores, a través del uso de interfaces API, [5]. Por ello, entre las características, más relevantes de WebRTC se tiene que, utiliza comunicación sincrónica entre pares[16], diseñado para proveer comunicación, transferencia de archivos, compartición de pantalla y capacidad de control remoto, utilizando APIs sencillos de JavaScript.

1.3.3.2 Arquitectura de WebRTC

El propósito de la arquitectura de WebRTC consiste en elaborar una plataforma de comunicación en tiempo real, que sea sólida y funcione sin complementos extras en diversos navegadores web y múltiples plataformas [17]. La arquitectura de WebRTC se observa en la Figura 1.4, en la cual se exponen los motores de video, motores de voz, códecs, entre otros. (Ver características en la Tabla 1.1).

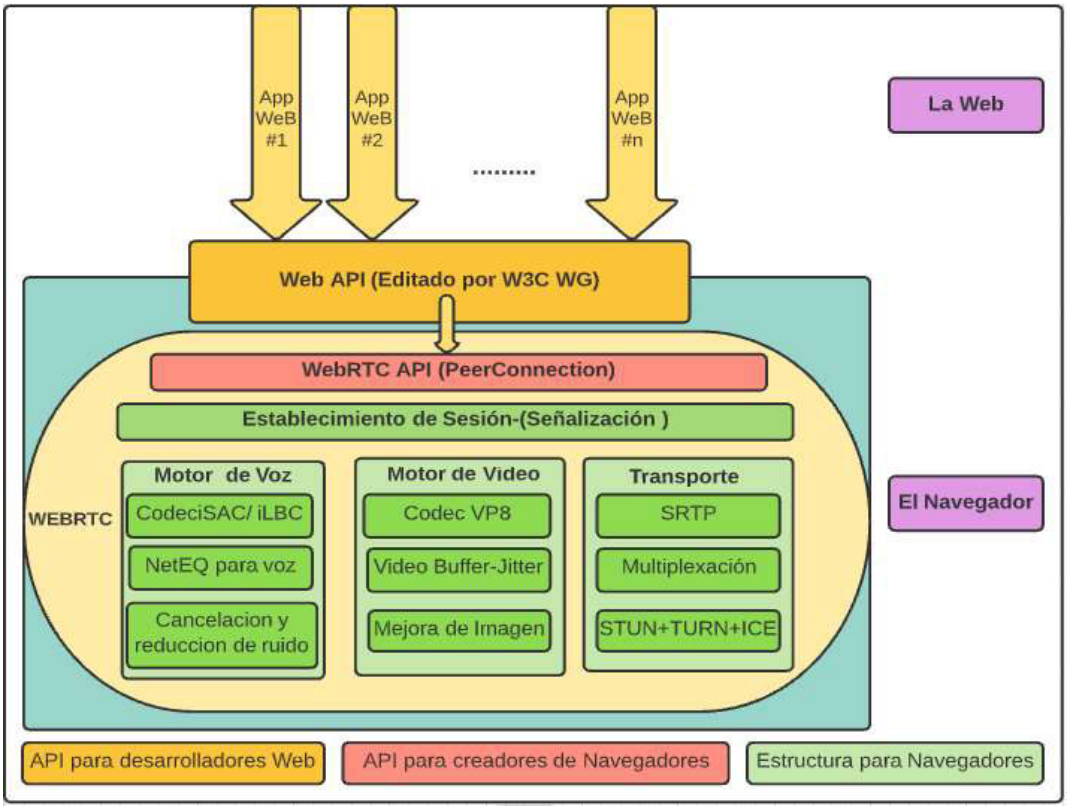


Figura. 1.4 Arquitectura general de WebRTC[17]

Tabla 1.1 Características de la arquitectura general de WebRTC [17]

Elementos	Características
La aplicación web	Permite la comunicación en voz y video, impulsada por la API de WebRTC.
API-WEB	En desarrolladores externos permite desarrollar aplicaciones de tipo videochat.
API nativa WebRTC	Permite a los fabricantes de navegadores implementar fácilmente la propuesta de API web.
Transporte/Sesión	Permite que las llamadas utilicen los mecanismos STUN y TURN para establecer conexiones a través de varios tipos de redes.
Motor de voz	Es un *framework* para el conjunto de medios de audio, desde la tarjeta de sonido a la red.
Motor de video	Es un *framework* para el conjunto de medios de video, desde la cámara a la red y desde la red a la pantalla.

1.3.3.3 Protocolos de WebRTC

Estos protocolos en WebRTC permiten establecer conexiones entre pares para intercambiar información entre la elección de rol (Cliente o servidor), la conexión DTLS *(Datagram Transport Layer Security)* y los números de puertos SCTP (*Secure Control Transmission Protocol*) para desarrollar la multiplexación de asociados, entre otros [18]. En la Figura 1.5 se muestra la estructura general de los protocolos que usa WebRTC.

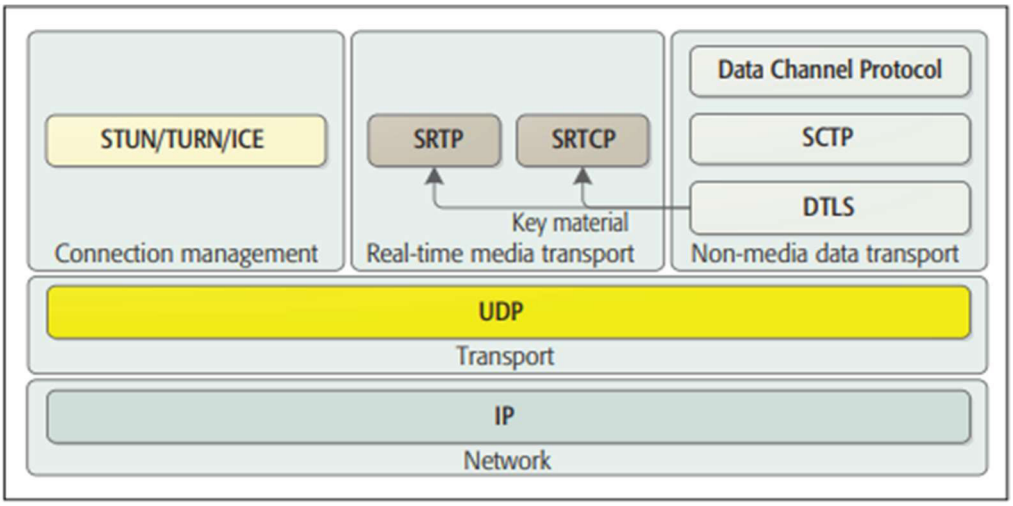


Figura. 1.5 Protocolos de WebRTC [18]

WebRTC al ser una solución de comunicación en tiempo real, requiere de alta velocidad, baja latencia y alto rendimiento, antes que fiabilidad en la comunicación. Por tal motivo en la capa de Transporte, el protocolo UDP (*User Datagram Protocol*) es preferido, antes que el protocolo TCP (*Transmission Control Protocol*); sin embargo, en situaciones especiales el uso de TCP en WebRTC también es posible.

De esta manera, con la selección del protocolo UDP en lugar de TCP en la tecnología WebRTC, la comunicación se vuelve menos segura, y es necesario añadir características que incrementen la seguridad en la comunicación, encriptando toda la información. Para realizar aquel procedimiento, WebRTC opta por los protocolos que se explican en la Tabla 1.2.

Tabla 1.2 Protocolos adicionales para seguridad en WebRTC

Protocolos	Características
DTLS	Proporciona privacidad y seguridad en el protocolo UDP, evita la manipulación y falsificación de datos en la comunicación[19]
SCTP	Protocolo encargado de gestionar las conexiones, tolerancia de errores y control de flujo.[20]
SRTP	Protocolo que otorga confidencialidad, autenticación de mensajes y protección en la respuesta a la comunicación WebRTC.[21] Por otro lado, para la API de WebRTC, se usa el protocolo ICE (*Interactive Connectivity Establishment*), el cual tiene como finalidad conectar un navegador web con otro navegador web, fuera de su propia red local. Los dos protocolos adicionales utilizados por ICE son STUN y TURN [22].
STUN (Session Traversal Utilities for NAT)	Encargado de conocer la dirección IP (*Internet Protocol*) pública de los usuarios, así como de cualquier restricción del *router* que impida una conexión *Peer-to-Peer.*
TURN (Traversal Using Relays around Nat)	Protocolo que ayuda en cruce de direcciones de red para aplicaciones multimedia, así mismo evita las restricciones NAT en los enrutadores y es el encargado de transmitir toda la información de los usuarios para la conexión *peer-to-peer* a través de un servidor TURN.

1.3.3.4 Señalización de WebRTC

Para evitar redundancia y maximizar la compatibilidad con las tecnologías existentes, los métodos y protocolos de señalización no son especificados en los estándares de WebRTC[23]. Sin embargo, en WebRTC la señalización es necesaria, porque indica cómo dos pares deben comportarse para conectarse. Si bien WebRTC no detalla la señalización, puede realizarla de manera simple; para esto hace uso de los siguientes dos protocolos:

- **SDP** (*Session Description Protocol*): encargado de recoger los parámetros para la conexión *peer-to-peer*. Entre los parámetros más importantes se tiene: tipo de medios que se dispone (audio-video) y sus resoluciones, códecs disponibles en los medios e información acerca de la red de los clientes[24].

- **JSEP** (*JavaScript Session Establishment Protocol*): protocolo destinado a imitar la señalización en WebRTC, haciendo uso del formato de SDP. Este protocolo, también es utilizado por el objeto *RTCPeerConnection*, el cual maneja la conexión entre los pares, y contiene información acerca de los servidores STUN/TURN.

1.3.3.5 Servidores STUN /TURN

Para logar la comunicación *peer-to-peer* requerida en WebRTC, es necesario que los usuarios conozcan la dirección IP de los participantes de la comunicación. En una situación ideal, cada punto final de la comunicación WebRTC tiene su propia y única dirección IP, las cuales se conectarían directamente, sin interrupciones [23]. Además, en el mundo real, la conexión puede fallar, debido a que los usuarios están detrás de una NAT (*Network Address Translation*), que dispone de *firewalls* activos, y en muchos casos detrás de programas de protección como antivirus; también se dispone de antispyware, los cuales bloquean ciertos protocolos y puertos. Por tal motivo, WebRTC implementa el protocolo ICE*,* el cual permite que la conexión entre usuarios sea exitosa, y es ahí donde entran en acción los servidores STUN/TURN.

- **STUN**

WebRTC necesita que los usuarios conozcan sus direcciones IP para establecer conexión, tanto en IP pública como privada. Los servidores STUN son la primera opción para obtener una conexión *peer-to-peer.* Por eso, la funcionalidad de STUN es sencilla: un usuario que está detrás de una NAT realiza una petición a un servidor STUN que se encuentra en la nube, este servidor responde con la dirección IP pública y el puerto que la NAT le ha asignado al usuario [25]. Gracias a este método se descubren todas las posibles direcciones, por las cuales se puede establecer conexión. En la actualidad, empresas como

Google, Mozilla entre otros, ofrece el servicio gratuito de servidores con el servicio STUN. La figura 1.6 muestra un esquema del funcionamiento que realiza un servidor STUN.

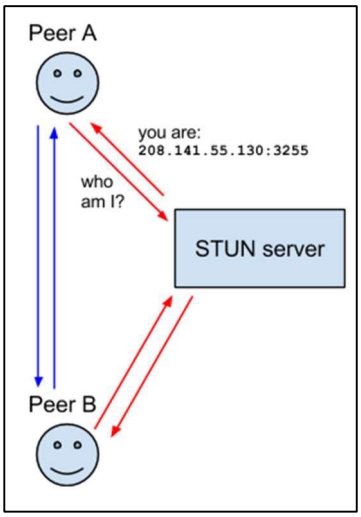


Figura. 1.6 Esquema Servidor STUN [22]

- **TURN**

El servidor TURN entra en acción cuando no es posible conectar los usuarios de manera directa, esto debido a que ciertos usuarios se encuentran detrás de NATs simétricos o *firewalls*. Los servidores TURN son el respaldo para la conexión si los servidores STUN fallan [26]. Por eso, la funcionalidad de TURN es la de retransmitir el tráfico entre los pares, evitando las restricciones que introducen las NAT simétricas [27]. Así, al realizar una retransmisión por el servidor TURN, se agrega latencia a la llamada. Sin embargo, en muchas ocasiones es la única forma de conectarse con el destinatario. La figura 1.7 muestra el esquema de funcionamiento de un servidor TURN.

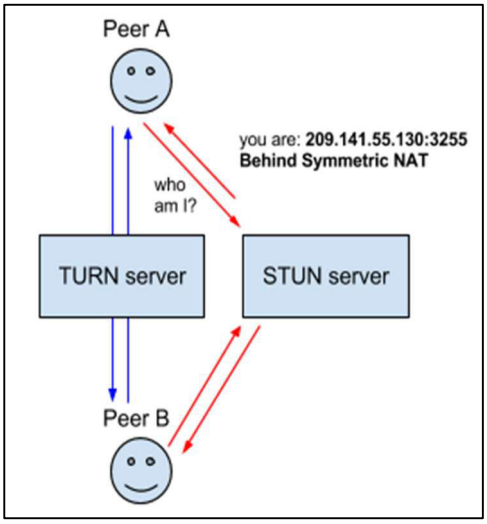


Figura. 1.7 Esquema Servidor TURN [22]

1.3.3.6 WebRTC en Navegadores

WebRTC es un proyecto *opensource,* el cual se encuentra respaldado por Google, Mozilla y Opera; todas las API y protocolos dentro de WebRTC son creados y desarrollados por W3C y el IETF[28].

Las características de WebRTC en los distintos navegadores (Chrome-Mozilla-Opera) no son las mismas. Sin embargo, la interoperabilidad entre ellos es posible mediante la adaptación del navegador que realiza la llamada. De igual manera, tanto Chrome como Mozilla Firefox poseen sus propias interfaces; en la figura 1.8 se muestra las sintaxis utilizadas por estos navegadores.

W3C Standard	Chrome	Firefox
getUserMedia	webkitGetUserMedia	mozGetUserMedia
RTCPeerConnection	webkitRTCPeerConnection	RTCPeerConnection
RTCSessionDescription	RTCSessionDescription	RTCSessionDescription
RTCIceCandidate	RTCIceCandidate	RTCIceCandidate

Figura. 1.8 Sintaxis WebRTC para interoperabilidad [29]

1.3.3.7 Interfaz de Aplicaciones Programadas en WebRTC

Las siguientes APIs son implementadas en WebRTC[30].

- **API MediaStream**

La API MediaStream se divide en dos componentes fundamentales [31].

- **MediaStream:** representa la agrupación de varios objetos *MediaStreamTrack.*
- **MediaStreamTrack:** representa un conjunto de archivos multimedia, como pistas de video y audio, donde cada pista es conocida como *Mediastreamtrack* [32].

La figura 1.9 muestra los dos componentes fundamentales en los que se divide la API de MediaStream.

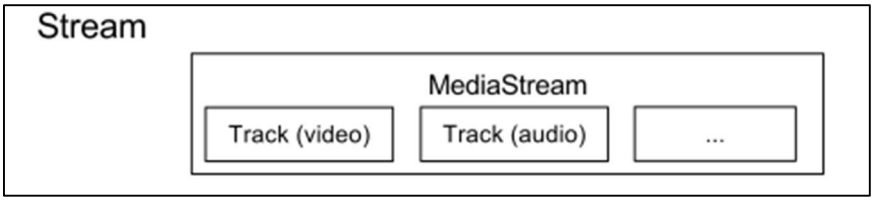


Figura. 1.9 Componentes de la API MediaStream [33]

- **API RTCPeerConnection**

Representa la conexión entre una computadora local y su par remoto [34]. Esta interfaz es útil para hacer una demostración simple de una aplicación funcional de *chat* y video; además provee las características necesarias para monitorizar y cerrar la conexión *peer-to-peer.*

El objeto *RTCPeerConnection* se encarga de establecer conexiones a través de NAT y *firewalls.*

- **API RTCDataChannel**

Es un canal para transmisión bidireccional de datos entre los pares conectados. Se tiene un canal por cada conexión *peer-to-peer* existente.

1.3.4 LENGUAJE JAVASCRIPT

El lenguaje de programación JavaScript fue creado para elaborar páginas web dinámicas, las cuales poseen textos, animaciones, e interacciones con el usuario. Su lenguaje es interpretado, porque no es necesario compilar a las aplicaciones para ser ejecutadas, lo cual facilita mucho su uso al ser funcional directamente sobre el navegador web de preferencia, sin aplicaciones intermedias[35].

Por otro lado, JavaScript se utiliza en la mayoría de los navegadores modernos: Google, Opera, Firefox, entre otros; también está orientado a objetos, aportando versatilidad al uso de éste. La sintaxis de programación es bastante sencilla y similar a otros lenguajes conocidos de programación como: C y Java.

Actualmente, JavaScript también se utiliza en programación con microcontroladores y en servidores. No tiene ninguna relación con el lenguaje de programación Java a pesar de su nombre particular.

1.3.5 HTML5

HTML5 es un nuevo concepto para diseñar sitios web y aplicaciones funcionales, es la combinación de HTML, CSS y JavaScript [36]. Mientras JavaScript otorga la sintaxis necesaria para brindar dinámica a las páginas web a crear, CSS provee de estilos y diseños visuales a la aplicación creada, y HTML es la estructura de los elementos dentro de la aplicación.

1.3.5.1 Organización general de HTML5

HTML5 utiliza los principios de HTLM, para lo cual posee una estructura definida para las aplicaciones [37]. En ese sentido, la organización de elementos, así como de secciones y

etiquetas son consideradas parte de su estructura básica. La Figura 1.10 indica los principales componentes que posee una página web actual.

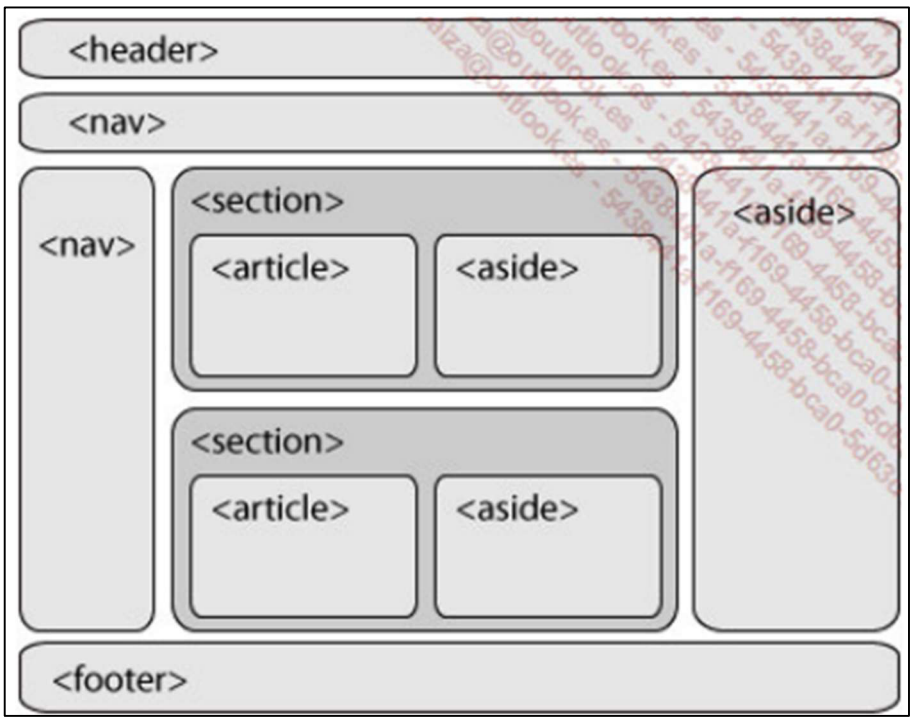


Figura. 1.10 Estructura General de HTML5 [37]

A continuación, se mencionan los elementos de la estructura general de HTML5:

- Elemento *header <Cabecera>:* representa una zona específica para cabeceras, tal como títulos, subtítulos, e información de introducción al usuario. Uso no obligatorio.
- Elemento *nav <Navegación>:* es una barra de navegación para facilitar al usuario la movilidad en el sitio web, puede poseer vínculos de hipertextos.
- Elemento *section<Sección>:* agrupación de elementos del mismo tópico, es la información que se le muestra al usuario. Normalmente acompañada de una cabecera.
- Elemento *article<Artículo>:* es el contenido textual de un sitio web, aquí se encuentra el contenido específico del tópico mencionado en la sección.
- Elemento *aside<Barra lateral>:* contiene información de menor relevancia acerca del contenido de la página, tales como: publicidad, vínculos externos, entre otros.
- Elemento *footer<Pie de página>:* representación de la información final del sitio web. La información legal, condiciones, contactos, entre otros, se especifican en esta zona.

1.3.6 METODOLOGÍA SCRUM

Scrum es una metodología de trabajo ágil direccionada a los productos tecnológicos, se basa en realizar entregas del producto final, mediante el desarrollo del software en ciclos cortos y rápidos comúnmente llamados "*Sprints*" o iteraciones. La colaboración y trabajo cooperativo son fundamentales en SCRUM para obtener el mejor resultado existente en un proyecto [38].

En la Figura 1.11, detalla el modelo de Scrum con los elementos que lo conforman[38].

1.3.6.1 Fases de Scrum

El desarrollo de Scrum consta de 3 fases fundamentales, en conjunto llamadas como "reuniones"[39].

- **Planificación de *Backlog*:** documento que posee los requisitos del producto en orden de importancia.
- **Seguimiento del *sprint* o iteración:** reuniones diarias para conocer el avance de las tareas.
- **Revisión del *sprint* o iteración:** es la verificación de que los objetivos planteados en las iteraciones se han cumplido correctamente, y en caso de ser necesario realizar cambios al *sprint*.

Las fases de Scrum permiten tener un control más adecuado del proyecto de acuerdo con las necesidades del producto final.

1.3.6.2 Elementos de Scrum

Scrum posee utiliza tres elementos para su correcta aplicación[39]:

1. ***Product Backlog*:** es un listado de todas las necesidades, requisitos y funcionalidades en orden de prioridad, las cuales serán colocados en las iteraciones del proyecto.
2. ***Backlog* de la iteración o *Backlog* de *sprint*:** lista de tareas que posee una iteración, son repartidas y asignadas con su tiempo correspondiente a los diversos miembros del grupo de trabajo.
3. **Incremento:** comprueba los requisitos que son completamente funcionales en el producto final.

En base a los resultados obtenidos en el proceso de Scrum, el cliente final puede solicitar realizar cambios y replantear el proyecto en caso de ser necesarios.

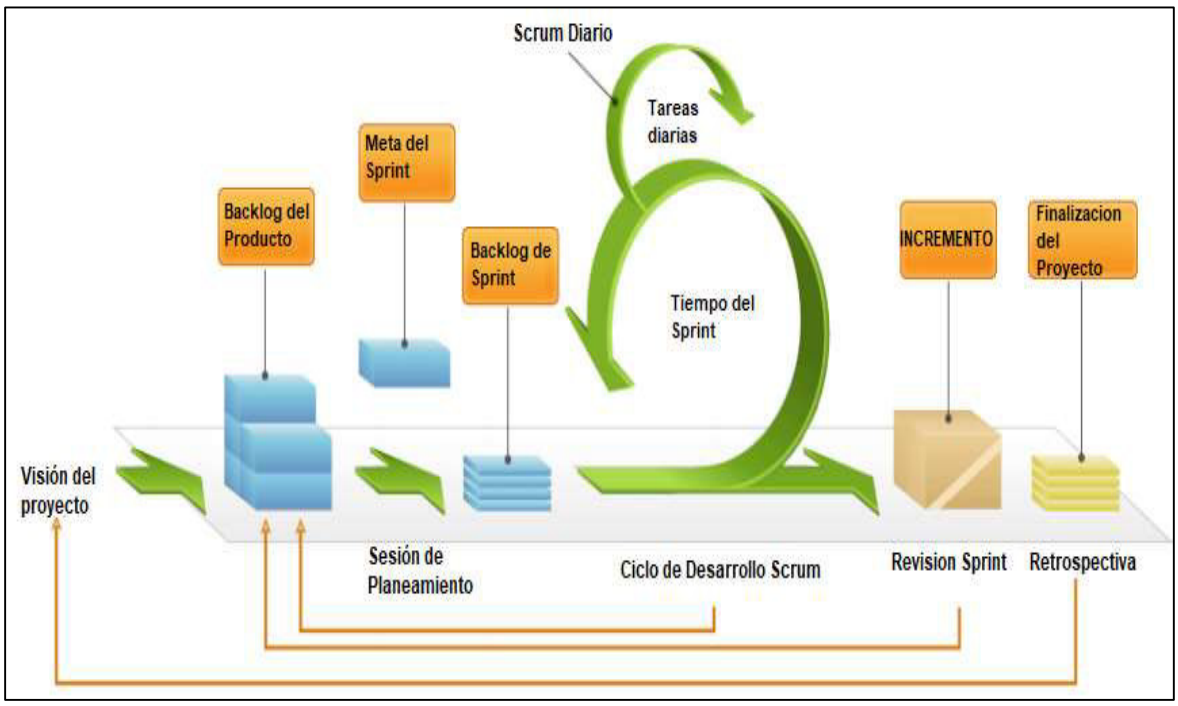


Figura. 1.11 Modelo de Scrum *[40]*

1.3.7 CSS

Hojas de estilo en cascada o CSS, es un lenguaje de programación encargado de la presentación gráfica de un sitio web. Trabaja en compañía de HTML para otorgar estilos visuales a los elementos provistos en la página web, tales como: color, tamaño de fuente, fondo de visualización, bordes de recuadros, tipo de letra, espaciados, entre muchos más[36].

De manera oficial, CSS no es parte de la especificación de HTML5; sin embargo, es considerado un complemento para minimizar la dificultad de HTLM, por eso, brinda a los programadores la oportunidad de satisfacer sus necesidades visuales en los sitios web.

1.3.8 BOOTSTRAP

BootStrap es un *framework* de tipo *front-end* utilizado para facilitar el desarrollo de páginas web mediante su *kit* de herramientas de código abierto. Este permite un *layout* que adapta los elementos del diseño de la pantalla del dispositivo del usuario.[41]

Este *framework, también requiere* el uso de JavaScript y de Popper[1], además de implementar plantillas, elementos y componentes que trabajan en conjunto con HTML y CSS, promoviendo un diseño de aplicaciones muy amigables para el usuario.

1.3.9 NODE.JS

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome, que orienta a eventos asíncronos. Node.js está diseñado para crear aplicaciones *network* escalables[42].

Node.js es de código abierto, multiplataforma, y brinda a los desarrolladores herramientas de lado del servidor, así como permite la elaboración de aplicaciones escritas en JavaScript. Diseñado para mejorar el rendimiento de los programas web en tiempo real.[43].

1.3.9.1 La comunidad y Node.js

Los módulos existentes en Node.js ofrecen un entorno amigable y poderoso para programar; al ser *opensource* la comunidad puede aportar y colaborar para extender las capacidades del entorno Node [44]. Así, para el desarrollo del presente trabajo de titulación se utilizaron los siguientes módulos:

- Módulo NPM (*Node Package Manager*)

Node.js posee su propio gestor de paquetes NPM, que administra los módulos existentes, puede agregar paquetes o dependencias de forma simple.

- Módulo HTTP

Módulo propio de NodeJs. encargado de manejar la conexión HTTP para los clientes, manteniendo solicitudes para un host y puerto determinados.

- Módulo HTTPS

Protocolo HTTP sobre TLS/SSL, módulo encargado de manejar la conexión HTTPS, escuchando y respondiendo solicitudes del cliente de manera segura.

- Módulo Express

Framework desarrollado por la comunidad, de características minimalista (utiliza un mínimo de componentes), permite la elaboración de sitios web, aplicaciones web y APIs [45].

[1] Popper: es una librería de JavaScript que permite el uso de un motor de posicionamiento de información sobre herramientas y cuadros emergentes, se aplica directamente sobre la hoja de estilos de la aplicación.

1.3.10 PEERJS

La API PeerJs envuelve la implementación de WebRTC para los navegadores, proporcionando una conexión *peer-to-peer* completa, configurable y fácil de usar. Además, es una biblioteca que facilita a los desarrolladores a utilizar la tecnología WebRTC, permitiendo la comunicación con mínimo tiempo de retardo, de manera muy sencilla, ágil y versátil; soporta también múltiples canales de datos, así como transmisiones de datos [46].

1.3.10.1 Componentes de PeerJs

A continuación, se detallan las características y componentes más importantes dentro de la API de PeerJs.

- **Configuración**

La configuración de pares es el paso principal para utilizar PeerJs, se debe incluir la librería dentro de la aplicación a desarrollar y crear un *peer* para la comunicación.

- **Conexión de datos**

Conectar y recibir, los pares deben conectarse entre sí para tener comunicación. Estos componentes lo realizan mediante *peer.connect* y *peer.on*.

- **Llamadas a los medios**

PeerJs es multiplataforma y permite la comunicación entre navegadores con la captura de audio-video, envío y recepción de *streams*. Las conexiones *peer-to-peer* requieren de un servidor como intermediario de conexión, que ofrece esta posibilidad mediante el uso de *PeerServerOficial,* el cual es un servidor gratuito y alojado en la nube: cabe recalcar que en este trabajo de titulación se realizó un *PeerServer* propio.

1.3.11 SOCKET.IO

Socket.IO es una biblioteca que permite la comunicación en tiempo real, bidireccional y basada en eventos entre el navegador y servidor [47]. Consta de dos componentes para su funcionalidad: el primero al lado del servidor, el cual es una biblioteca para NODE.JS, y el segundo del lado del cliente, como una biblioteca JS para el navegador del usuario.

1.3.11.1 Características de Socket.IO

Entre las características más importantes de Socket.IO se detallan las siguientes [47]:

- **Confiabilidad**

En caso de no poder realizar conexión mediante el protocolo WebSockets, se realiza un respaldo de *HTTP long-polling*.

- **Reconexión Automática**

En caso de fallar la conexión, se debe configurar la reconexión automática si es posible (No todos los sistemas soportan esta característica de Socket.IO).

- **Almacenamiento en búfer**

Se almacenan eventos ocurridos mientras el Socket no esté conectado. Una vez realizada la reconexión, se restablecen los eventos.

- **Acknowledgments**

Socket.IO proporciona una API clásica de solicitud-respuesta. Se responde solo si existe una solicitud.

- **Transmisión a un grupo de clientes o a todos (*broadcast*)**

Envío de datos a usuarios seleccionados o a todos los clientes ubicados dentro del *socket*.

- **Multiplexación-Espacios de nombres**

Canal de comunicación que permite compartir la lógica de su aplicación en una sola conexión múltiple.

1.3.11.2 Funcionamiento de Socket.IO

Socket.IO usa el protocolo de comunicación WebSocket para establecer una conexión *full-duplex* y de baja latencia entre el servidor y el cliente; elementos tales como proxy, *firewall*, antivirus, etc. no impiden conexiones de este protocolo, por lo cual Socket.IO se vuelve una herramienta fuerte para desarrollar aplicaciones en tiempo real.

Socket.IO no es una implementación de WebSocket, es multiplataforma en navegadores y dispositivos, funciona siempre y cuando el navegador del cliente sea compatible con el protocolo mencionado. En la actualidad el 97% de los navegadores soportan este protocolo de comunicación [48].

2 METODOLOGÍA

Este capítulo establece las funcionalidades que tendrá la aplicación, lo cual permite el desarrollo de cada módulo que la conforma con sus respectivas características; también se detallan las relaciones que los componentes poseen entre sí.

Se tienen dos etapas primordiales para su desarrollo: etapa de diseño del sistema y etapa de implementación. La etapa de Diseño estipula el análisis de requerimientos de la aplicación, su arquitectura, particularidades y necesidades. La etapa de Implementación se divide en tres secciones: la primera es la codificación y establecimiento de software , en la cual se codifica los módulos del sistema y se hace uso de la metodología Scrum, realizando las iteraciones (*Sprints)* necesarias para cumplir con los requisitos estipulados, así como la preparación de software necesario para la ejecución de la aplicación, la segunda es la colocación de la aplicación dentro de un ambiente local, determinando posibles fallas en la ejecución de ésta; y, la tercera referente a la puesta en marcha del prototipo en un ambiente *cloud*.

2.1 ETAPA DE DISEÑO

La etapa de Diseño define: los requerimientos funcionales y no funcionales del sistema en base a entrevistas realizadas a múltiples usuarios y al alcance estipulado, la arquitectura del prototipo, y se modela la estructura del sistema basado en el uso de diagramas UML (Lenguaje Unificado de Modelado).

2.1.1 ENTREVISTAS

Deben ser previamente diseñadas en función al tema de estudio, a la vez de ser planteadas por el entrevistador [49].

2.1.1.1 Preparación de la entrevista

El entrevistador debe hacer sentir cómodo al entrevistado, y para hacerlo deberá conocer el tema a tratar, así como las preguntas que serán aplicadas. Es importante señalar que se debe evitar improvisar.

La entrevista puede ser estructurada o no estructurada. En el presente trabajo de titulación se utilizará el tipo de entrevista estructurada, puesto que esta forma de entrevista permitirá realizar preguntas previamente pensadas y dirigidas a entrevistados particulares, los cuales responden concretamente lo que se les está preguntando.

Entre las ventajas en una entrevista estructurada se tiene que las respuestas obtenidas son fáciles de interpretar, facilitando el análisis comparativo, y el entrevistador no necesita de mucha experiencia para realizar las preguntas.

- **Cronograma de preguntas**

Para la realización de la entrevista se ha realizado un cronograma de 9 preguntas, entrevista que consta en el anexo A.

Los entrevistados serán personas que actualmente se encuentren en la funcionalidad de teletrabajo, estudio virtual y personas afines de tecnología.

2.1.1.2 Resultados de las entrevistas

Doce personas fueron los entrevistados que aportaron en la obtención de datos. En las figuras 2.1 a 2.9 se muestran los gráficos de las respuestas obtenidas y tabuladas de las entrevistas.

La figura 2.1 muestra el rango de edad de los participantes en las entrevistas; se puede observar que la mayoría de los participantes se encuentra entre 21-30 años.

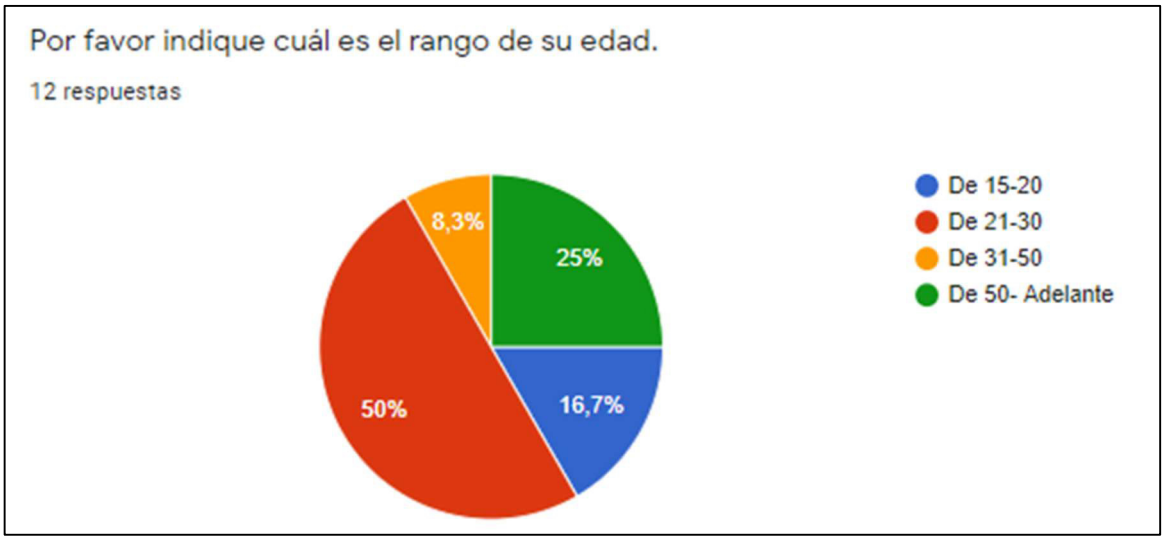


Figura. 2.1 Resultado de las edades de participantes

La figura 2.2 busca conocer si los sistemas de videoconferencia son considerados importantes en la actualidad para los participantes de las entrevistas; el resultado muestra que para todos los participantes los sistemas de videoconferencia son importantes.

Figura. 2.2 Gráfico de importancia de la videoconferencia

La figura 2.3 indica cuáles son los sistemas más utilizados por los participantes. Se puede observar que la plataforma “Zoom” es la más empleada por los entrevistados.

Figura. 2.3 Resultado de los sistemas más utilizados

La figura 2.4 muestra si el participante utiliza algún sistema de videoconferencia en sus actividades diarias. Once de los doce participantes indican que utilizan sistemas de videoconferencia en sus labores cotidianas.

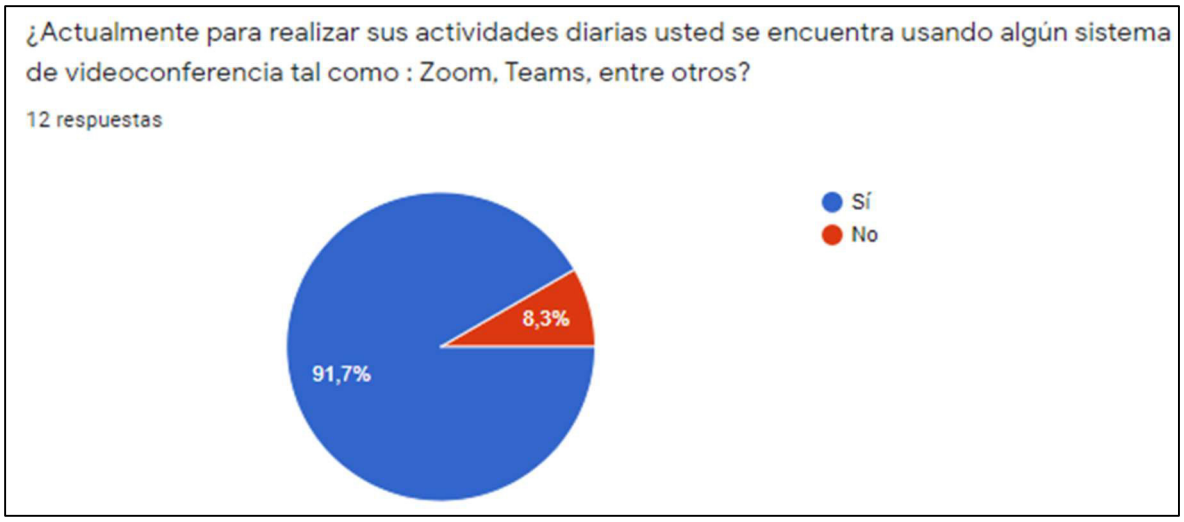


Figura. 2.4 Resultado de uso diario de sistemas

La figura 2.5 indica los tiempos, en el cual el participante hace uso de un sistema de videoconferencia para sus actividades diarias.

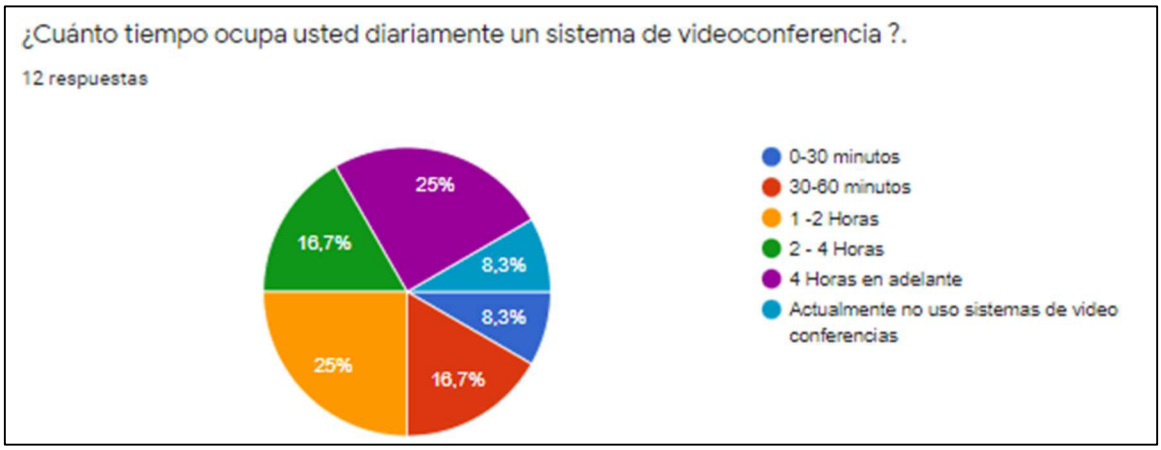


Figura. 2.5 Resultado de tiempo de uso

La figura 2.6 expone cuáles son las características que los participantes consideran de mayor importancia. Gratuidad, compartición de pantalla y tiempo ilimitado de reuniones, son los detalles más valiosos para los participantes.

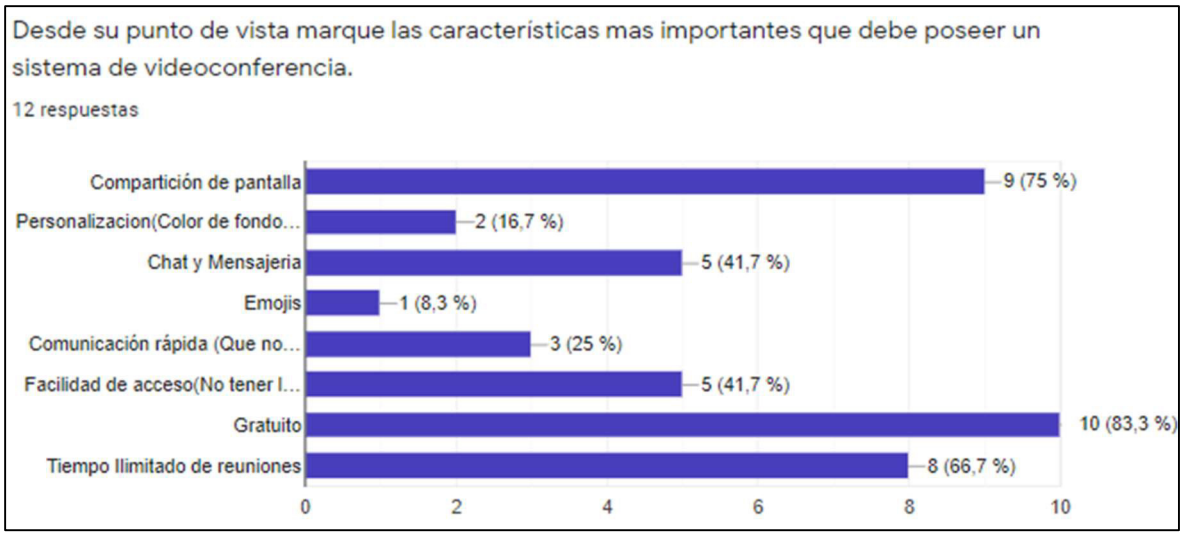


Figura. 2.6 Características más importantes para el usuario

La figura 2.7 indica si un usuario considera incómodo tener un tiempo límite en las reuniones de videoconferencia; 75% de los usuarios respondieron que “Si” les resulta incómodo.

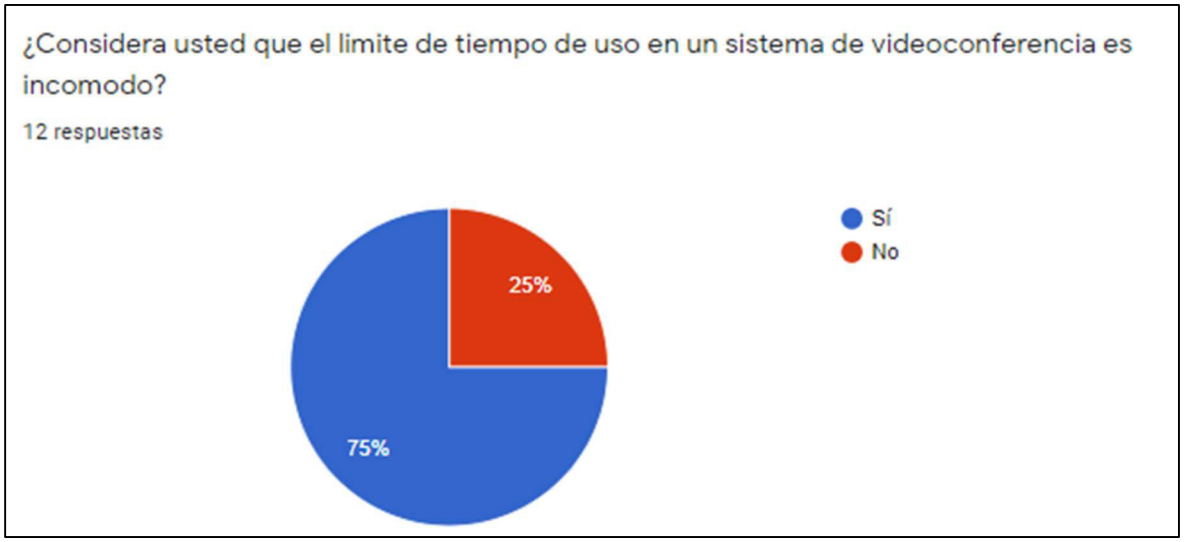


Figura. 2.7 Resultado de Importancia de límite de tiempo

La figura 2.8 es la consulta al participante si desea tener un sistema de videoconferencia sin límite de tiempo en reuniones y que sea gratuito. El 100% de los usuarios respondieron positivamente.

Figura. 2.8 Pregunta de opinión sobre límite de tiempo y gratuidad

La figura 2.9 consulta al participante si desea tener un sistema de videoconferencia con mensajería y *chat*, así como con la opción de compartir pantalla. El 100% de los usuarios respondieron positivamente.

Figura. 2.9 Pregunta de opinión sobre chat y compartición de pantalla

Con los resultados obtenidos en las entrevistas, se puede concluir que la realización de una nueva alternativa de sistema de videoconferencia es viable; así mismo, se concluye las características más importantes para el usuario son la compartición de pantalla, reunión sin límite de tiempo, y gratuidad en el sistema de videoconferencia.

2.1.2 REQUERIMIENTOS DE FUNCIONALIDAD

Esta sección hace distinción entre los requerimientos funcionales, los cuales explican los servicios que proveerá el prototipo, y no funcionales que denotan características generales del prototipo tales como rendimiento, seguridad y disponibilidad.

Los requerimientos funcionales y no funcionales fueron obtenidos a partir de las entrevistas y de ciertas características de las herramientas utilizadas en el desarrollo del trabajo de titulación, así como de características de otros sistemas de videoconferencia.

Se recalca que no todas las características de otros sistemas de videoconferencia pueden ser reflejados en los requerimientos, por el alcance estipulado en el trabajo.

2.1.2.1 Requerimientos Funcionales

Los requerimientos funcionales definidos son los siguientes:

1. La aplicación permitirá la videoconferencia de hasta 4 usuarios simultáneos.
2. La aplicación deberá crear una sala de videoconferencia con su propio identificativo.
3. La aplicación permitirá intercambio de mensajes de texto de hasta 4 usuarios simultáneos.
4. La aplicación notificará a los participantes, por mensaje de texto, el acceso de un nuevo usuario a la videoconferencia.
5. Los participantes no pueden eliminar a otro participante de la reunión.
6. La aplicación permitirá la compartición de pantalla de un usuario con hasta 3 usuarios adicionales.
7. La aplicación permitirá enviar mensajes de texto de máximo 20 caracteres.
8. La aplicación permitirá realizar videoconferencias con enlaces de Internet personalizables.
9. La aplicación permitirá realizar videoconferencias sin límite de duración de llamada.
10. La aplicación deberá ser compatible con los navegadores: Google Chrome, Mozilla Firefox y Opera.
11. La aplicación deberá permitir el reingreso de los participantes a la reunión, en caso de desconexión de éstos, utilizando el *link* de la sala.
12. La aplicación no necesitará de software adicional para su funcionamiento.

13. La aplicación debe ser capaz de operar con una conexión mínima del usuario de 3 Mbps.

2.1.2.2 Requerimientos no funcionales

Los requerimientos no funcionales definidos son los siguientes:

1. Los usuarios deben aplicar las actualizaciones del sistema operativo, así como de los navegadores de forma periódica.
2. En caso de error, la aplicación debe enviar mensajes informativos al usuario.
3. La aplicación por motivos de seguridad hace uso del protocolo HTTPS.
4. La aplicación no necesitará de modificaciones en el dispositivo del usuario (desactivación de *firewall*, antivirus, entre otros.).

2.1.3 ARQUITECTURA DEL PROTOTIPO

La arquitectura del proyecto se basa en el modelo de tres capas, dividiendo tareas en cada una de ellas:

- **Capa de datos:** encargada de manejar correctamente los datos de los usuarios, direcciones IP, puertos, e información adicional para el adecuado funcionamiento de la aplicación. La representación de esta capa está dada por el servidor STUN-TURN.
- **Capa de Lógica de Negocio:** funcionalidad de procesar la información, encargada de manejar las peticiones de los usuarios, así como su respuesta. Esta capa está representada en el servidor web Express perteneciente a Node.js y en el uso de Socket.IO
- **Capa Presentación:** delegada para gestionar las interfaces de usuario y su presentación, así como las interacción del usuario con el sistema. La representación de esta capa se encuentra dada por el navegador utilizado y por servidor web Express perteneciente a Node.js.

En la figura 2.10 se observa la arquitectura del prototipo y las relaciones de las capas.

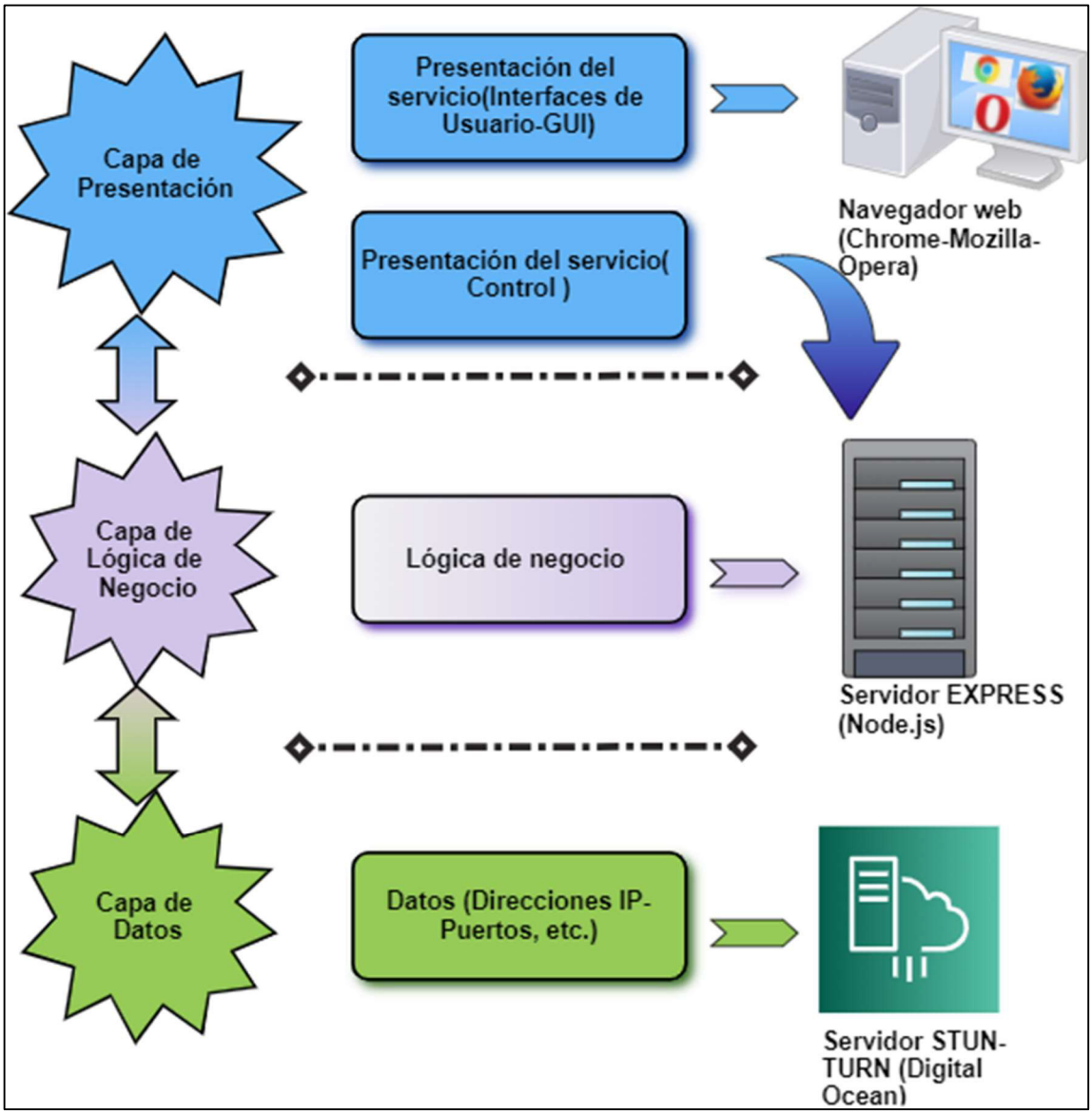


Figura. 2.10 Arquitectura del sistema y relación de capas

2.1.4 DIAGRAMA DE CASOS DE USOS

Los casos de usos son aquellas acciones que los actores efectúan sobre el sistema con la finalidad de obtener un resultado posterior.

El actor principal que interactúa con el sistema es el usuario y basado en los requerimientos funcionales se elabora el diagrama de casos de uso y sus componentes, el cual es mostrado en la figura 2.11.

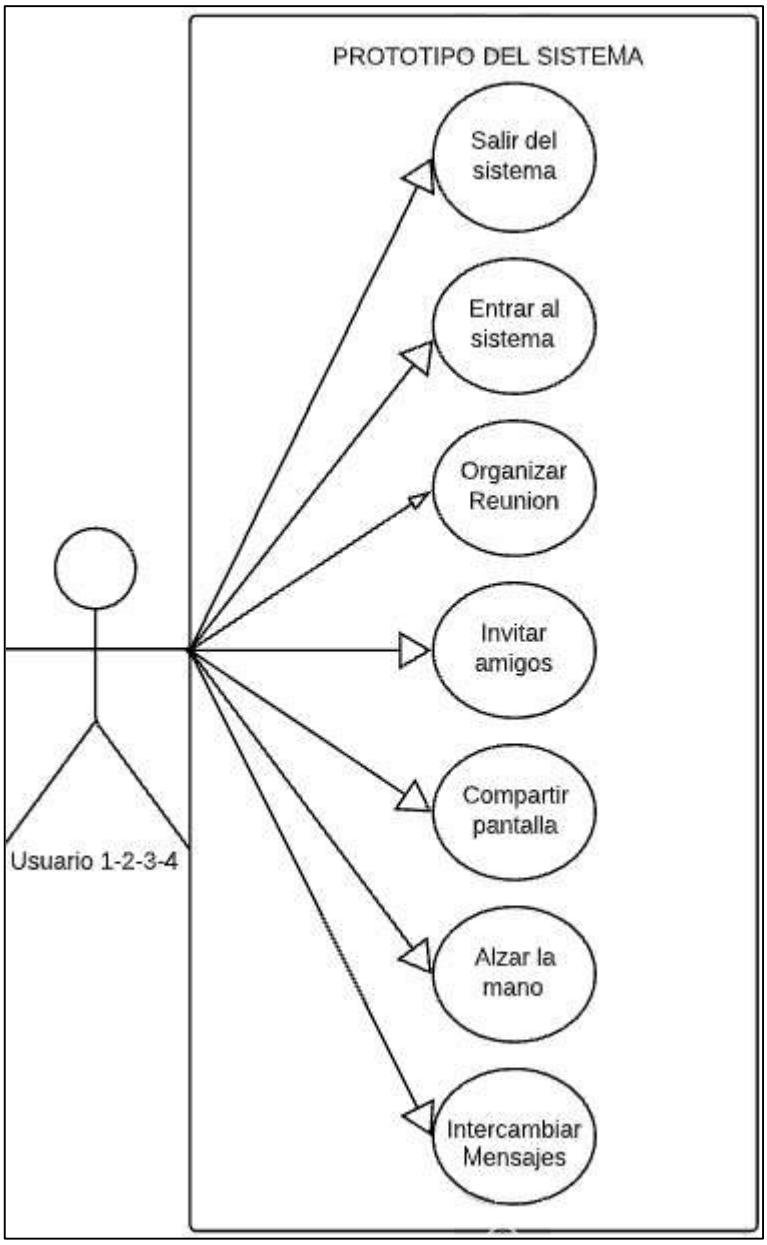


Figura. 2.11 Diagrama de caso de uso del prototipo

2.1.5 MÓDULOS DEL SISTEMA

Basado en los requisitos funcionales y no funcionales, en la arquitectura del sistema, y en los diagramas de estructura del sistema, se han declarado los siguientes módulos para el sistema por su operatividad:

a. **Módulo de Conexión:** permitirá la conexión entre clientes y servidor, para después realizar la conexión *peer-to-peer*.

b. **Módulo de Presentación:** permitirá visualizar los videos de los participantes, así como su modificación en base al número de participantes en la sesión.

c. **Módulo de Mensajería:** permitirá visualizar los mensajes entre los participantes, así como el envío y recepción de mensajes hasta de un máximo 20 caracteres por mensaje, utilizando Socket.IO.

d. **Módulo de Controles:** permitirá el uso de WebRTC para la compartición de pantalla, y características adicionales, tales como activar y desactivar video y/o audio e invitar amigos mediante la copia de enlace en el portapapeles, invitar a usuarios mediante envío de correo electrónico.

e. **Módulo de Señalización:** permitirá el uso del servidor STUN/TURN para conectar usuarios de diferentes redes a través del Internet.

2.1.6 DISEÑO DE PROCESOS E INTERFACES DE USUARIO

Esta sección muestra la serie de pasos a seguir para cada uno de los elementos de los módulos, así como el diseño utilizado en las interfaces de usuario.

Las interfaces de usuario fueron diseñadas para las necesidades antes especificadas, y se ha utilizado la herramienta 'Pencil Project' en su versión *desktop* gratuita.

La progresión de pasos de cada uno de los elementos en los módulos está diseñada con diagramas de actividades, e indican el flujo de trabajo del usuario con el sistema; se hace uso de la herramienta 'Cacco' en su versión *online.*

Cabe recalcar que las interfaces previstas son una representación previa del producto final a desarrollar, estas interfaces se encuentran sujetas a cambios.

2.1.6.1 Módulo de Conexión

Este módulo presentará la página inicial del prototipo mediante la conexión entre navegador y las plataformas GoDaddy, Heroku, y DigitalOcean; la primera impresión del sistema será otorgada por este módulo.

- **Inicio de la aplicación**

Esta interfaz permitirá a los usuarios conocer un poco más del sistema de videoconferencia, su forma de uso y características. Los usuarios no tendrán la necesidad de introducir datos personales para acceder al servicio. Al acceder a la página principal de la aplicación, se estará accediendo a la plataforma GoDaddy, la cual solicitará los *scripts* correspondientes a la plataforma HEROKU, mediante la verificación de su *dyno*. Con la aplicación ya en el navegador, se consultará al servidor STUN/TURN, y si éste se encuentra activo para conectar clientes a la videoconferencia. La figura 2.12 indica la interfaz para el inicio de la aplicación.

Figura. 2.12 Interfaz de inicio de la aplicación

En la figura 2.13 se muestra el diagrama de actividades del Inicio de la aplicación.

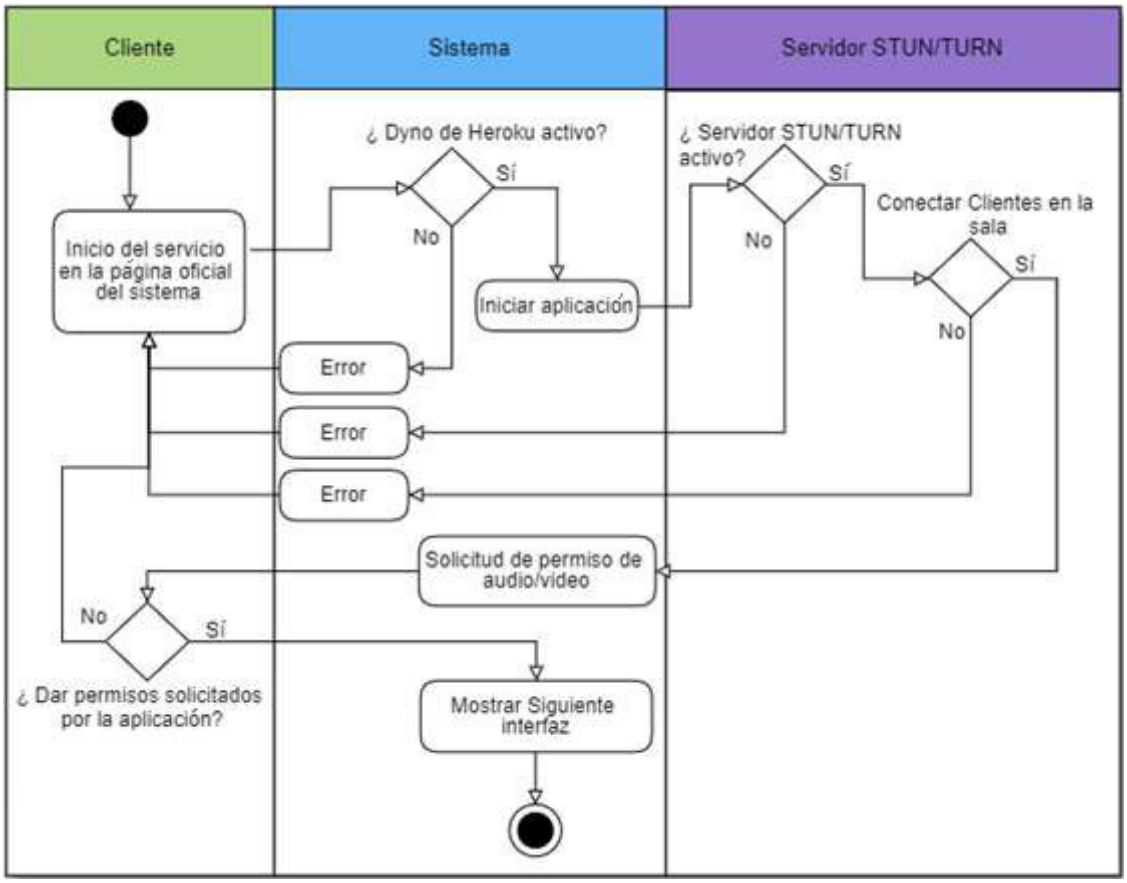


Figura. 2.13 Diagrama de actividades de inicio de la aplicación

2.1.6.2 Módulo de Presentación

Encargado de mostrar al cliente, los videos de todos los participantes de la reunión.

- **Sección de Video**

Interfaz encargada de modificar el tamaño de los videos de los usuarios, acorde al número de clientes activos. El usuario tendrá la facultad de colocar un video específico en pantalla completa mediante el doble *click* al video deseado. La figura 2.14 muestra la interfaz para la sección de video.

Figura. 2.14 Sección de Video

El diagrama de actividades de la Sección de video se muestra en la figura 2.15.

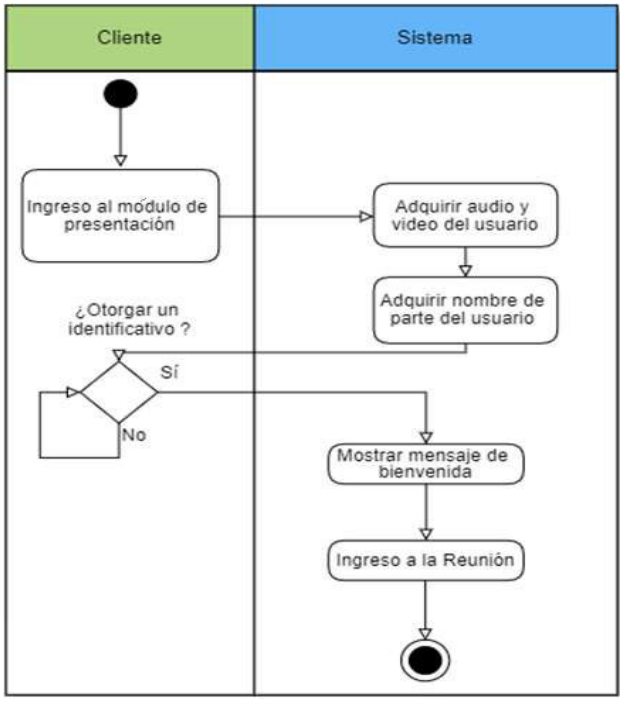


Figura. 2.15 Diagrama de actividades de la sección de video

2.1.6.3 Módulo de Mensajería

- **Panel de mensajes**

La figura 2.16 muestra la interfaz correspondiente al panel de mensajes, en la cual el usuario podrá visualizar los mensajes de texto enviados y recibidos en la reunión, cada uno de ellos con el respectivo nombre del emisor. Con la finalidad de facilitar al usuario la visualización de mensajes de texto se tendrá una barra de desplazamiento vertical que permite mover el texto hacia arriba o abajo.

Figura. 2.16 Panel de mensajes de la aplicación

- **Panel de datos**

En este panel el usuario podrá escribir sus mensajes de texto, para luego ser enviados al servidor, el cual transmitirá el mensaje a cada uno de los participantes de la reunión, con la excepción del emisor.

Se pueden tener mensajes de máximo 20 caracteres, existiendo dos maneras en las que el usuario puede enviar los datos de texto:

1. El usuario decide enviar el mensaje de texto mediante un *click* en un botón (Palomita).
2. El usuario decide presionar la tecla “Enter” para enviar el mensaje de texto.

En este panel se validará que el mensaje enviado no esté vacío, en caso de estarlo no se enviará ningún dato al servidor; de igual manera se validará que se envíe el número de caracteres máximo permitido.

Para el proceso de envío de mensajes se diseñó el diagrama de actividades. La figura 2.17 indica los pasos a seguir para la transmisión de texto.

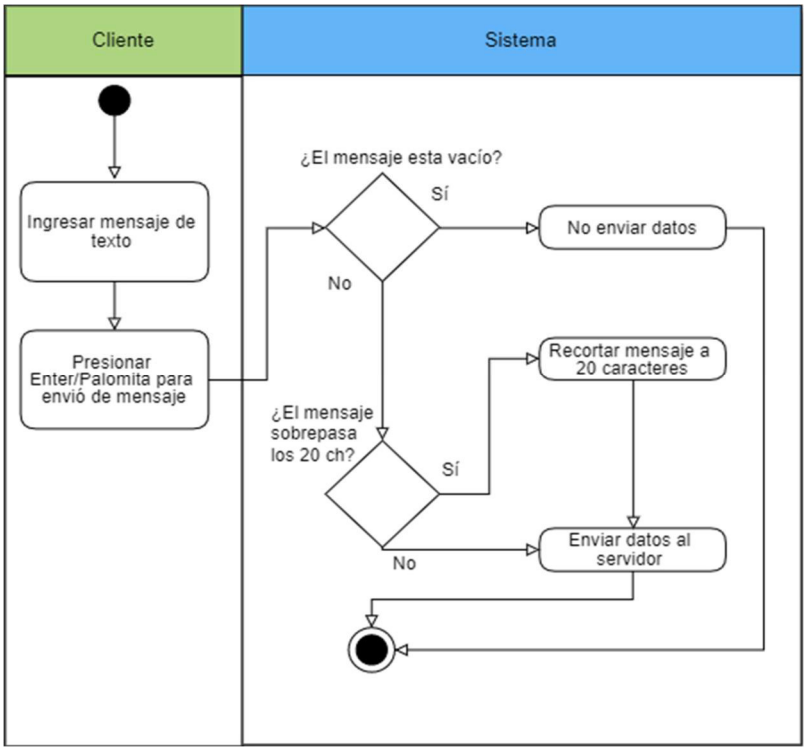


Figura. 2.17 Proceso para la transmisión de mensajes de texto

2.1.6.4 Módulo de Controles

- **Controles multimedia**

La figura 2.18 muestra los controles de multimedia (audio/video) que el usuario dispondrá para su uso; activar-desactivar audio y prender o apagar la cámara son las funciones de estos controles.

Figura. 2.18 Controles de Audio y Video

Cabe recalcar que el usuario podrá mantener simultáneamente su cámara y su audio apagados o prendidos.

- **Controles de reunión**

Estos controles permitirán al usuario tener control acerca de las acciones que puede realizar en la reunión: compartir pantalla, invitar amigos, levantar la mano, mostrar información acerca de quien ha desarrollado el sistema.

En la figura 2.19 se observa el diseño de la interfaz de controles de reunión.

Figura. 2.19 Controles de audio y video

2.1.6.5 Módulo de Señalización

Tiene como finalidad permitir la comunicación entre pares. Se han utilizado las plataformas informáticas de Heroku, Digital Ocean y GoDaddy; los detalles de las plataformas se detallan posteriormente.

Para realizar la comunicación entre usuarios se hace uso de *scripts* programados con lenguaje JavaScript; éstos se ejecutan en el lado del cliente en su navegador y permiten usar las API's de WebRTC para videoconferencia, mensajería, compartición de pantalla. Por otro lado, el servidor web está programado en uno de los *scripts* que se entrega al cliente; la plataforma "Heroku" se encarga de la entrega de los archivos.

Al iniciar la comunicación, previamente, el sistema verificará que exista un servidor de señalización disponible que permita realizar la videoconferencia; en caso de no poseer un

servidor de señalización, el usuario no tendrá la posibilidad de conectarse con otros usuarios.

El servidor de señalización se encuentra ubicado en la nube, en la plataforma Digital Ocean, y tiene tanto el servicio STUN como el servicio TURN, los cuales buscan garantizar la comunicación *peer-to-peer* entre los usuarios. El servidor se encontrará "levantado" en una máquina virtual Ubuntu 18.04.

Cuando el usuario inicia una videoconferencia, el sistema automáticamente creará un sala o canal para compartir la información, esta sala posee su propio identificativo mediante una URI (*Uniform Resource Identifier*), el cual es único y permite que otros usuarios ingresen a la sala; cualquiera de los usuarios puede compartir el enlace de la sala para que ingresen los invitados a la videoconferencia, no hay un moderador en la sesión.

Al momento de recibir el enlace, los usuarios invitados a la sala solo tendrán que ingresar con su navegador, otorgar los permisos correspondientes para audio y video, así como ingresar su nombre para poder acceder a la sala.

En una sala de videoconferencia, cualquiera de los usuarios puede compartir su pantalla con los invitados que se encuentren dentro de la sala. El video del participante que comparte es reemplazado con el video de la compartición, y al momento de terminar su presentación, volverá el video inicial. Los usuarios pueden agrandar el video de la persona que comparte su pantalla, haciendo doble *click* en su imagen referencial.

Se recalca que para la compartición de pantalla no se hace uso de extensiones adicionales, ni de software extra.

La figura 2.20 indica el diagrama de actividades para el proceso de comunicación general, que sigue el funcionamiento de la aplicación para cualquier usuario, suponiendo que el usuario ha otorgado los permisos correspondientes al sistema, y que posea una conexión estable a Internet en su máquina local.

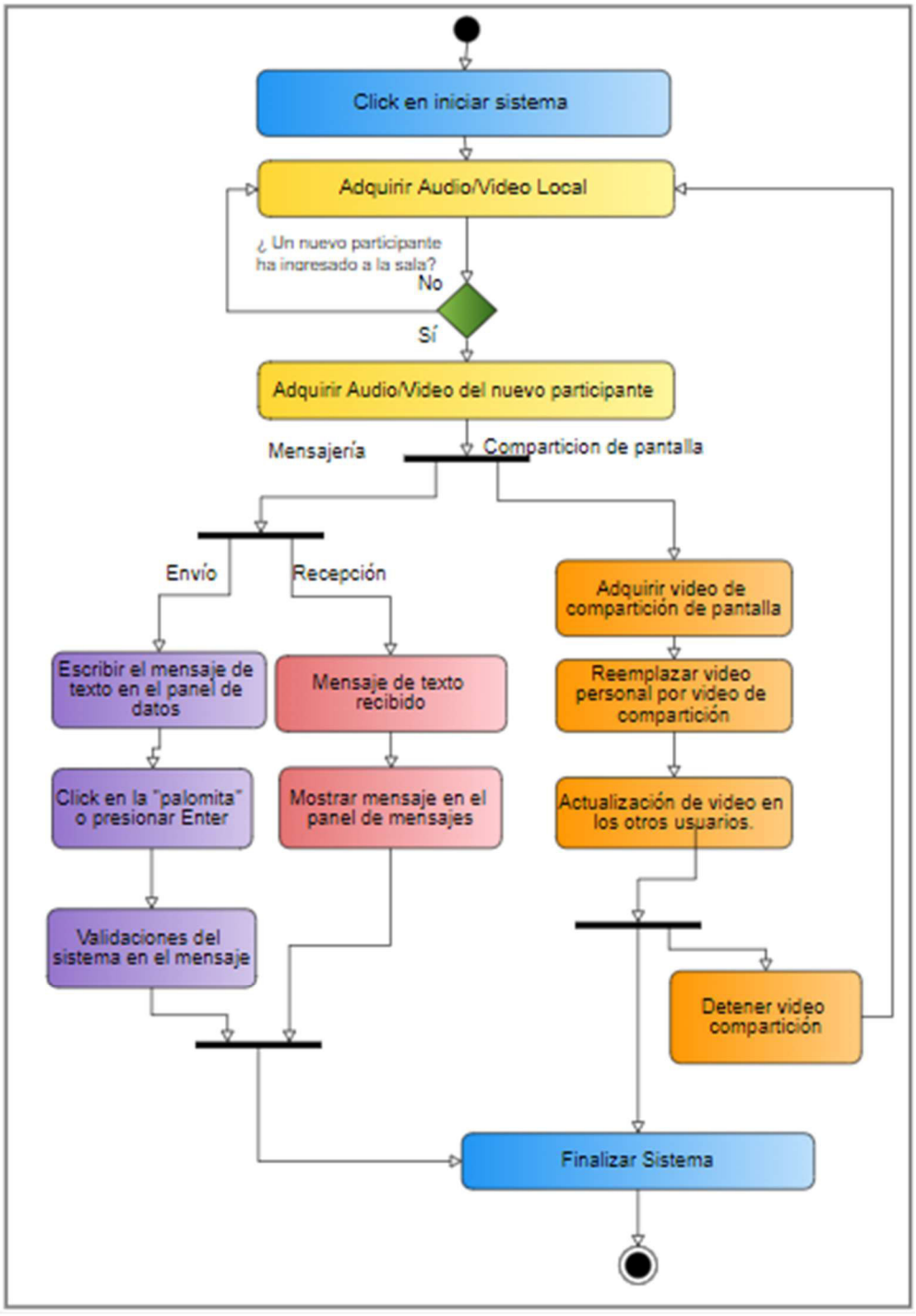


Figura. 2.20 Proceso de comunicación General

2.1.7 DISEÑO EN SCRUM

Con los requerimientos funcionales, no funcionales, módulos definidos y objetivos claros, se hace uso de la metodología Scrum para desarrollar un plan de trabajo, con la finalidad de obtener resultados positivos en el desarrollo de software.

La metodología Scrum permite dividir las tareas en interacciones, las cuales son pequeñas tareas que buscan cumplir una tarea mayor. Scrum otorga flexibilidad y rapidez al momento de ejecutar resultados, también aprueba mantener reuniones constantes para revisar el avance de las iteraciones, así como la retrospectiva adecuada y mejora de las versiones del proyecto en caso de ser necesario.

2.1.7.1 Historias de usuarios

Las historias de usuario permiten describir las funcionalidad que va a poseer el software; pueden evolucionar en el tiempo del proyecto, y se las clasifica basadas en los módulos definidos del prototipo.

Las historias de usuario ayudan a realizar el *Product Backlog* del proyecto.

2.1.7.1.1 Formato de historias de usuario

La tabla 2.1 contiene el formato de historias de usuario, en la que se consideran los siguientes campos:

1. **ID:** es la identificación que distingue a cada historia de usuario.
2. **Título:** título que describe a la historia del usuario.
3. **Prioridad:** importancia que posee una historia de usuario con respecto de otras historias. Se ha clasificado las historias de usuario con los números del 1 al 3, a mayor importancia, mayor número.
4. **Descripción:** es la definición de la historia de usuario, describe sus objetivos.
5. **Pruebas de Aceptación:** son criterios de aceptación que indican que una historia de usuario se ha cumplido con éxito.

Tabla 2.1 Formato de la historia de usuario

ID		**Título:**	
Prioridad			
Descripción:			
Pruebas de aceptación:			

2.1.7.1.2 Desarrollo de Historias de usuario

En esta sección se presentan las historias de usuario desarrolladas para el prototipo, las cuales han sido definidas en base a los requerimientos antes analizados; las historias de usuario se clasifican en base a los módulos especificados con anterioridad.

- **Módulo de Conexión**

En las Tablas 2.2, a 2.4 se describen las historias de usuario correspondientes al Módulo de Conexión.

Tabla 2.2 Historia de Usuario de Página Principal

ID	**HU-01**	**Título:**	**Página principal de la aplicación-Acerca del sistema**
Prioridad	**1**		
Descripción: ▪ Se mostrará la página principal de la aplicación. ▪ Se plasmará información de la aplicación en la página principal.			
Pruebas de aceptación: **La aplicación admite que los clientes puedan visualizar la página de inicio del sistema.** **La página principal de la aplicación muestra información idónea acerca del sistema.**			

Tabla 2.3 Historia de Usuario Conexión Hosting

ID	**HU-02**	**Título:**	**Conexión y configuración del servicio de *hosting* (GoDaddy)**
Prioridad	**3**		
Descripción: ▪ Se realizará la configuración del servicio de *hosting*. ▪ El usuario conseguirá conectarse con éxito al servicio de *hosting*.			
Pruebas de aceptación: **Los clientes podrán conectarse al servicio de *hosting*, con las respectivas especificaciones de la aplicación ya configuradas especialmente para el sistema.**			

Tabla 2.4 Historia de Usuario Conexión Web

ID	HU-03	Título:	Conexión y configuración del servidor web (Heroku)
Prioridad	3		
Descripción: ▪ Se realizará la configuración del servicio Heroku, ubicado en la nube. ▪ El usuario conseguirá recibir los archivos desde la plataforma, para ejecutar el sistema.			
Pruebas de aceptación: **Los clientes podrán conectarse a la plataforma Heroku, la cual le entregará los *scripts* correspondientes para la ejecución del sistema; se creará la sala de videoconferencia para su uso.**			

- **Módulo de Presentación**

En las Tablas 2.5 a 2.7 se muestra el detalle de la historia de usuario correspondiente al Módulo de Presentación.

Tabla 2.5 Historia de Usuario para Presentación de Aplicación

ID	HU-04	Título:	Presentación de la aplicación - Sección Multimedia
Prioridad	3		
Descripción: El usuario podrá acceder a la videoconferencia. La aplicación adquiere los datos de audio y video mediante el uso de WebRTC. El usuario debe otorgar permisos correspondientes para la obtención de multimedia.			
Pruebas de aceptación: **El usuario podrá visualizar su video otorgando los permisos correspondientes; el tamaño de la imagen del usuario se modificará en base al número de participantes en la reunión.**			

Tabla 2.6 Historia de Usuario para Videoconferencia

ID	HU-05	Título:	Presentación de la aplicación - Videoconferencia
Prioridad	2		
Descripción: El usuario podrá realizar videoconferencia con hasta tres participantes extras; los usuarios adicionales necesitan estar en la misma sala.			
Pruebas de aceptación: **La aplicación permite realizar la videoconferencia de manera óptima con hasta un máximo de 4 participantes. En caso de ingresar más usuarios, no se asegura que la aplicación funcione adecuadamente.**			

Tabla 2.7 Historia de Usuario de Compartición de pantalla

ID	HU-06	Título:	Presentación de la aplicación - Compartición de pantalla
Prioridad	3		
Descripción: El usuario podrá compartir su pantalla con hasta tres participantes extras, los usuarios adicionales necesitan estar en la misma sala.			
Pruebas de aceptación: **La aplicación permite a un usuario realizar compartición de pantalla con el resto de los participantes de la videoconferencia.**			

- **Módulo de Mensajería**

En la Tabla 2.8 se detalla la historia correspondiente al módulo de mensajería.

Tabla 2.8 Historia de Usuario de Transferencia de Mensajes

ID	HU-07	Titulo:	Mensajería de la aplicación - Transferencia de Mensajes
Prioridad	3		
Descripción: Los usuarios podrán enviar y recibir mensajes de texto entre los participantes de la reunión, hasta un máximo de 20 caracteres por mensaje.			
Pruebas de aceptación: **La aplicación permite intercambiar mensajes de texto entre todos los participantes que se encuentren en la sala (máximo 4 participantes por sala).**			

- **Módulo de Controles**

En la Tabla 2.9 se detalla la historia correspondiente al Módulo de Controles.

Tabla 2.9 Historia de Usuario Controles

ID	HU-08	Título:	Controles de la aplicación
Prioridad	2		
Descripción: Los usuarios podrán utilizar los controles de la reunión, activar/desactivar audio-video, compartición de pantalla, invitación a la reunión, entre otros.			
Pruebas de aceptación: **La aplicación permite a los usuarios utilizar correctamente los controles de la reunión.**			

- **Módulo de Señalización**

En la Tabla 2.10 se detalla la historia correspondiente al Módulo de Señalización.

Tabla 2.10 Historia de Usuario Instalación servidor STUN/TURN

ID	**HU-09**	**Título:**	**Servidor STUN/TURN**
Prioridad	**3**		
Descripción: La aplicación utilizará los servicios del servidor STUN/TURN para interconectar a los participantes mediante la web.			
Pruebas de aceptación: **La aplicación permite a los usuarios conectarse entre sí accediendo a la aplicación desde lugares remotos, haciendo uso del servidor STUN/TURN.**			

2.1.7.2 Product Backlog

Una vez definidas las historias de usuario, así como los requerimientos de la aplicación, se establece la estructura del *Product Backlog* con sus respectivos componentes para la puesta en funcionamiento del sistema. La tabla 2.11 muestra el *Product Backlog* que se ha utilizado en el progreso del proyecto.

Tabla 2.11 *Product Backlog*

Sprint	Prioridad	Identificativo HU	Título HU
1	3	HU-00	Instalación de IDE de desarrollo y configuración del proyecto
		HU-02	Conexión y configuración del servicio de *hosting* (GoDaddy)
		HU-03	Conexión y configuración del servidor web (Heroku)
2	3	HU-01	Página principal de la aplicación-Acerca del sistema
		HU-04	Presentación de la aplicación -Sección Multimedia
		HU-06	Presentación de la aplicación - Videoconferencia

3	2	HU-05	Compartición de pantalla
	3	HU-07	Mensajería de la aplicación -Transferencia de Mensajes
4	2	HU-08	Controles de la aplicación
5	3	HU-09	Servidor STUN/TURN

El *Product Backlog* para el avance del proyecto consta de cinco *Sprints* para realizar todo lo necesario para su desarrollo. Los *Sprints* se detallan en la codificación del sistema.

2.2 ETAPA DE IMPLEMENTACIÓN DEL PROTOTIPO

Una vez establecido el diseño del software se procede a realizar el establecimiento del software y la codificación de los módulos del sistema, para posteriormente realizar la implementación del prototipo en los ambientes local y *cloud*.

2.2.1 ETAPA DE CODIFICACIÓN

La etapa de Codificación muestra todos los códigos realizados para que el prototipo sea funcional, haciendo uso de los *Sprints* definidos y su planificación.

2.2.1.1 SPRINT 1

2.2.1.1.1 Planificación del Sprint 1

La Tabla 2.12 consta de todas las tareas especificadas en el *Product Backlog* que deben ser realizadas para cumplir con el objetivo del *Sprint 1.*

Tabla 2.12 Planificación *Sprint 1*

ID	Tareas por realizar	Horas
HU-00	Instalar IDE de Visual Studio Code	5
	Instalar Librerías adicionales	8
	Crear Nuevo proyecto Node Express en Visual Studio Code	7
HU-02	Adquirir Dominio Web Privado	3
	Estudiar características del Dominio WEB	2
	Configuración del Dominio WEB	5

HU-03	Adquirir un Dyno (servidor web) para alojamiento de aplicación	2
	Configuración del Dyno Web	3

2.2.1.1.1.1 Instalación y configuración del Entorno de Desarrollo

El proyecto se realizó utilizando el entorno de desarrollo Visual Studio Code en su versión 1.56.2, basado en Electrón en su versión 12.04.

- **Instalación de Node.JS - NPM**

Para usar Node.JS se descarga el software desde su página web: https://nodejs.org/es/ , se instaló y utilizó Node.JS para Windows con su asistente de instalación.

La figura 2.21 muestra las versiones con las cuales se trabajó en el desarrollo del proyecto; para Node.JS se ha utilizado la versión 14.15.4, la cual viene con el sistema de gestión de paquetes NPM en su versión 6.14.11.

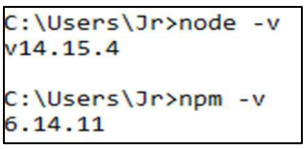


Figura. 2.21 Versiones Node.JS-NPM

- **Inicio de la aplicación web con Node.JS y Express**

En este apartado se indican los pasos a seguir para el inicio de una aplicación web usando Node.JS y Express, así como las configuraciones iniciales que debe contener.

1. Crear una nueva carpeta en el dispositivo local, la cual contendrá todos los archivos que el proyecto necesite.
2. En la barra de herramientas, “abrir” la carpeta antes creada (Figura 2.22) y crear un nuevo archivo con extensión .JS (Figura 2.23)

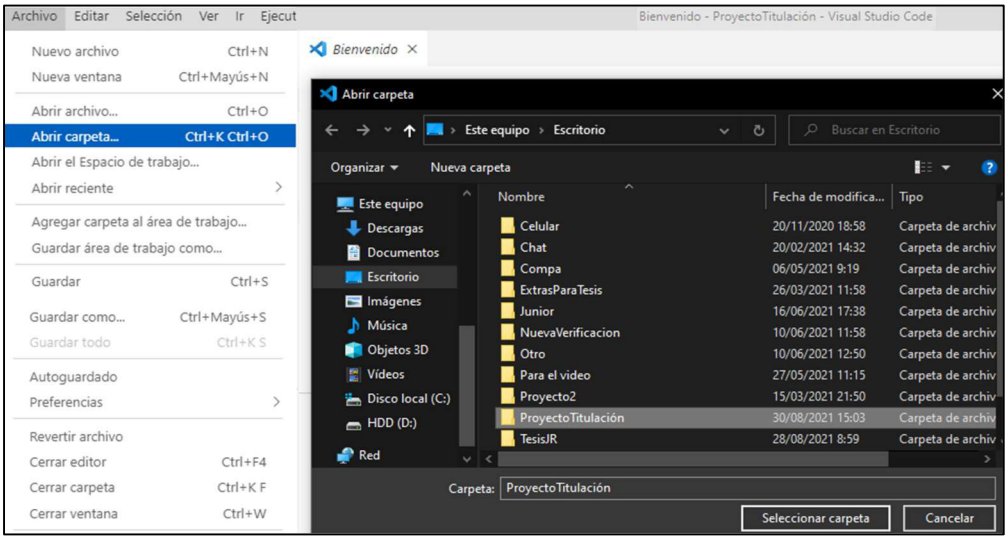


Figura. 2.22 Abrir Carpeta

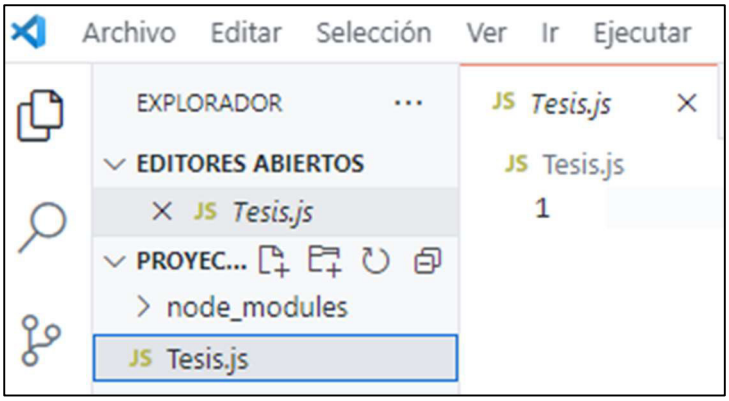


Figura. 2.23 Crear archivo con extensión .js

3. En la barra de herramientas en “Terminal” → “Abrir nuevo terminal”, una vez abierto el terminal se aplica el formato de Node JS, mediante el comando “**npm init**”. El comando solicitará información adicional del autor del proyecto: nombre, descripción, el repositorio GIT (en caso de poseer), entre otros.

4. La figura 2.24 muestra el proyecto ya creado y listo para usar Node JS; librerías adicionales se deben instalar vía consola.

Figura. 2.24 Proyecto Node Js

5. Para la instalación de Express se debe colocar el comando "**npm install express**"; una vez instalado se podrá visualizar la dependencia. La figura 2.25 muestra lo mencionado en este paso. Todas las librerías adicionales han sido incluidas vía consola.

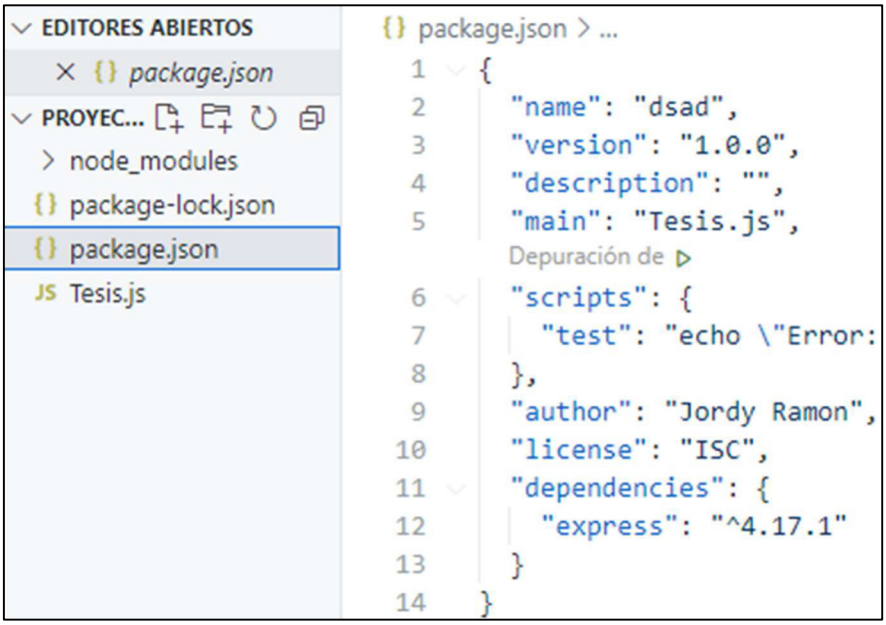


Figura. 2.25 Dependencias del proyecto

- **Servidor Web HTTP**

La configuración del servidor web se encuentra en el *script* principal "server.js", realizada con el módulo "http" y "express". El código 2.1 indica la estructura del código para el servidor.

```
JS server.js > ...
const express= require('express');
const app =express();
const cors = require('cors')
const server = require ('http').Server(app);
const io = require('socket.io')(server)
const {v4: uuidv4 } = require('uuid');
const { ExpressPeerServer } = require('peer');
const peerServer = ExpressPeerServer(server, {
  debug: true
});

server.listen( 3030);
```

Código. 2.1 Configuración del servidor HTTP en “server.js”

En un ambiente controlado el servidor escucha en el puerto 3030, mientras que en un ambiente *cloud* lo hará a través del puerto que le otorgue la plataforma de servicios Heroku, esto gracias al “*process.env.PORT*” que contiene al iniciar la escucha del servidor.

La aplicación usa la función “*getUserMedia*” para adquirir audio y video del participante; esta función debe trabajar sobre conexiones seguras “https”. La conexión segura es dada por la plataforma Heroku en el ambiente *cloud*.

- **UUIDV4 y Motor gráfico**

El desarrollo de la aplicación requiere que los usuarios puedan crear distintas salas de reuniones, para esto se usa un “Identificador único universal” UUIDV4 y se especifica que al iniciar la aplicación se inicie una sala con su propio identificador.

El motor gráfico utilizado para el proyecto es EJS, el cual es un motor compatible con “*Express*”. El código 2.2 muestra la configuración del servidor con UUIDV4 y EJS.

```
app.set('view engine','ejs');

app.use(express.static('public'));


app.get('/',(req, res) =>{

    res.redirect(`/${uuidv4()}`);
})
```

Código. 2.2 Configuración de motor EJS y UUIDV4

Adicionalmente se incluyeron los archivos: “script.js”, “room.js” y “style.css”.

El archivo “script.js” es el *script* principal escrito en JavaScript que contiene las funcionalidades de la aplicación, siendo éste el de mayor importancia. Permite funciones como: compartición de pantalla, uso de botones, manejo de video, configuraciones de acceso STUN-TURN, métodos de comunicación, entre otros.

El archivo “room.js” es el *script* HTML de la aplicación, define la estructura y significado del contenido web que posee el sistema.

El archivo “style.css” es el *script* CSS de la aplicación, contiene todo el diseño gráfico de la aplicación; la presentación del sistema se encuentra en este documento.

2.2.1.1.1.2 Servicio de Hosting (GoDaddy)

Para el desarrollo de este trabajo, es necesario adquirir un dominio web; la página inicial de la aplicación será desarrollada con un plan gratuito que ofrece la empresa. El dominio será intermediario para acceder al servidor STUN/TURN el cual, al encontrarse en la nube, posee su propia IP pública.

- **Adquisición y configuración del dominio web**

La figura 2.26 muestra el dominio adquirido de nombre “turnservejr.com”; el dominio tiene una duración de 12 meses y permite configurar características del DNS (Figura 2.27).

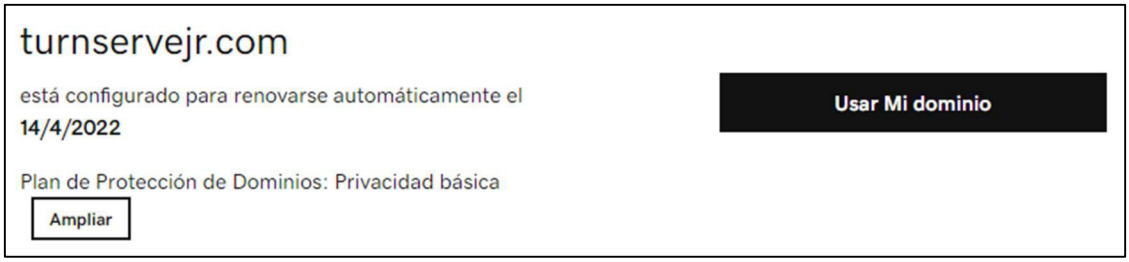


Figura. 2.26 Dominio adquirido turnservejr.com

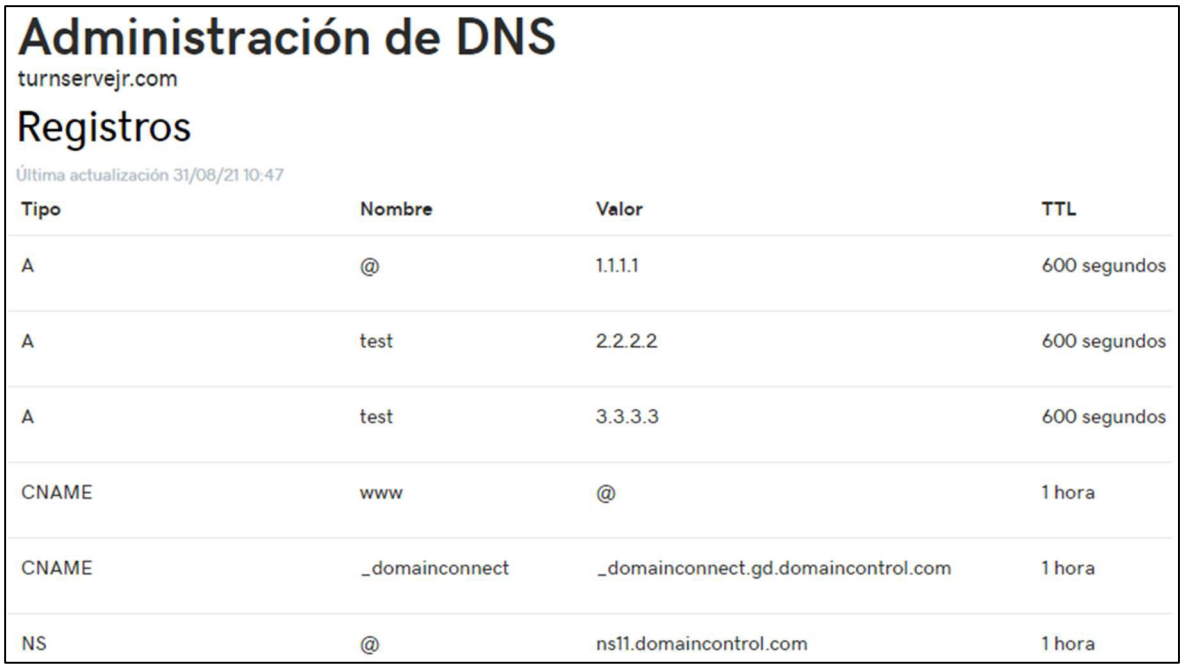


Figura. 2.27 Configuración DNS del dominio

La figura 2.28 indica que el dominio permite configurar la privacidad en cualquier momento.

Figura. 2.28 Configuración de privacidad del dominio

Cuando la aplicación inicie solicitará el servicio del servidor STUN/TURN; al hacerlo lo pedirá a través del dominio el cual a su vez solicitará a la dirección IP pública del servidor; esta configuración se la realiza en el DNS.

2.2.1.1.1.3 Servicio de alojamiento (Heroku)

Esta sección muestra los pasos que fueron seguidos para el correcto manejo de la plataforma de alojamiento Heroku.

- **Adquisición y configuración del servicio de alojamiento Heroku**

Heroku utiliza contenedores para alojar aplicaciones, estos contenedores son conocidos como “dynos” y poseen sus propias características; la figura 2.29 indica los tipos de “*dynos*” que se encuentran en la plataforma. Para el desarrollo del proyecto se ha utilizado un “*dyno*” gratuito, el cual contiene lo necesario para el sistema a desarrollar.

El registro a la plataforma es gratuito.

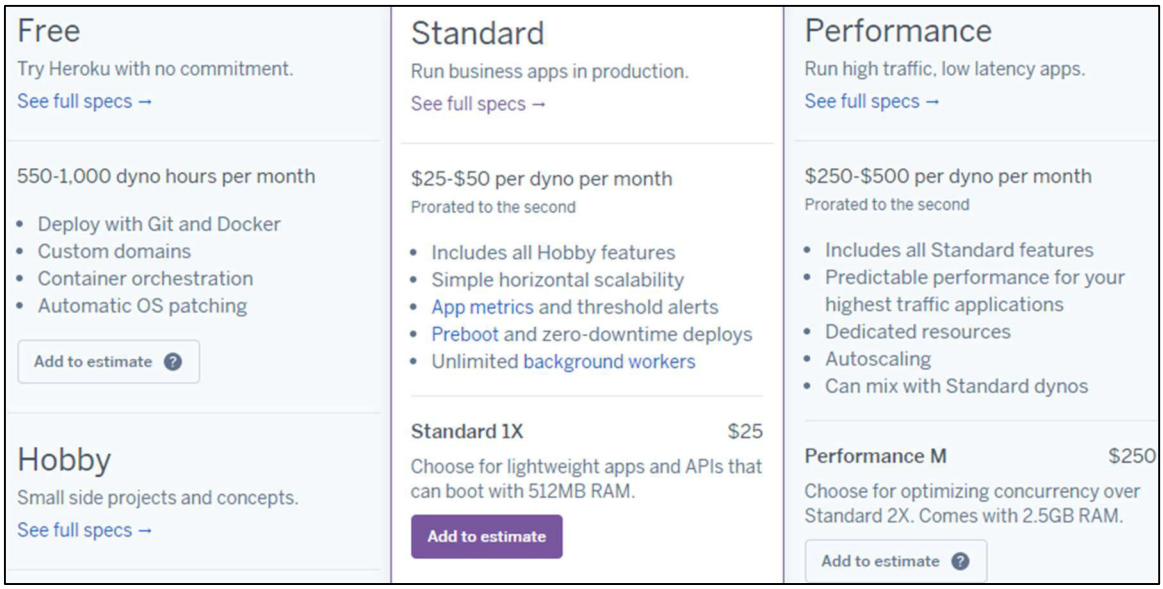


Figura. 2.29 Tipos de *Dynos* en Heroku

Una vez seleccionado el tipo de *dyno* se procede a configurarlo, la figura 2.30 presenta la configuración realizada; un mejor contenedor tendrá mayor número de características. El *framework* utilizado en el contenedor es Node.JS.

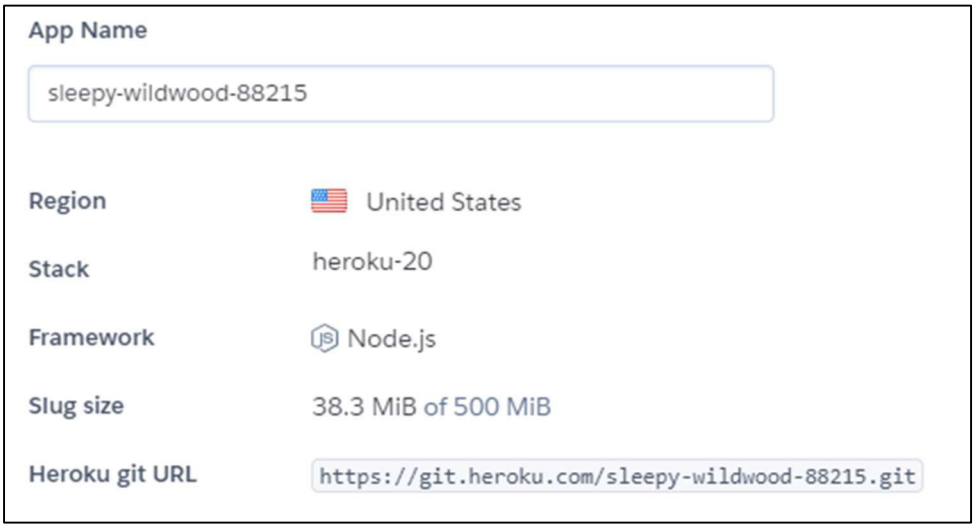


Figura. 2.30 Configuración de *Dyno*

2.2.1.2 SPRINT 2

2.2.1.2.1 Planificación del Sprint 2

La Tabla 2.13 contiene las tareas estipuladas en el *Product Backlog*, y que deben ser realizadas para cumplir con el objetivo del *Sprint 2.*

Tabla 2.13 Planificación *Sprint* 2

ID	Tareas por realizar	Horas
HU-01	Elaborar página principal del sistema	10
HU-04	Creación de vista principal para los usuarios en room.ejs y style.css	10
	Métodos para adquisición de datos multimedia locales en script.js	5
	Métodos para manejo de tamaño de presentación en script.js	5
HU-06	Métodos para manejo de pares en el archivo script.js	10
	Métodos asociados de conexión en el archivo server.js	10

El progreso del *Sprint 2* considera varios aspectos en el desarrollo de la comunicación, en esta iteración se hace uso de diversas funcionalidades de WebRTC, las cuales se ejecutan en el lado del cliente. Múltiples *scripts* se ven relacionados en el avance del *sprint* y también son incluidos en la vista principal de los usuarios.

2.2.1.2.1.1 Página principal de la aplicación

Al adquirir un dominio en GoDaddy, éste ofrece la posibilidad de crear un sitio web gratuito con plantillas básicas; la figura 2.31 indica los tipos de sitios web que dispone la plataforma. Los usuarios podrán acceder al sitio web creado y ésta será la página principal de la aplicación.

Figura. 2.31 Tipos de sitios web en GoDaddy

La figura 2.32 muestra que se ha creado un sitio web público de nombre "http://websaccs.godaddysites.com/" y que se encuentra publicado en la Internet; las personas con acceso a Internet podrán acceder a esta página web.

Figura. 2.32 Sitio Web creado y publicado.

2.2.1.2.1.2 Presentación de la aplicación - Sección Multimedia

- **Vista principal de los usuarios**

La vista principal de los usuarios se ha escrito sobre el archivo "room.ejs", dando una estructura al sistema con lenguaje HTML. El código 2.3 muestra el código desarrollado.

```
<!DOCTYPE html>
<html lang="en" class>
▶ <head>…</head>
▼ <body class wfd-invisible="true">
  ▼ <div class="Principal botonMostrar botonOcultar"> flex
    ▼ <div class="Izquierda"> flex
      ▶ <div class="ParteVideos">…</div> flex
      ▶ <div class="Controles">…</div> flex
      </div>
    ▼ <div class="Derecha"> flex
      ▶ <div class="CabeceraDerecha">…</div>
      ▶ <div class="VentanaChat">…</div>
      ▶ <div class="ContenedorDeMensajes">…</div> flex
      </div>
    </div>
    <script src="script.js" wfd-invisible="true"></script>
  </body>
</html> == $0
```

Código. 2.3 Código de Estructura HTML para la aplicación

El diseño de la vista principal se encuentra escrito en el archivo “style.css” y se relaciona con el archivo “room.ejs” mediante la referencia especificada por el código: “***<link rel="stylesheet" href="style.css">***”, su función es decirle al archivo “room.ejs”, que su hoja de estilo es la referencia “style.css”. El código 2.4 muestra una parte del código utilizado para los estilos, y el código 2.5 indica la relación entre los archivos mencionados.

```
# style.css
public > # style.css > #video-grid
1
2   body{
3       margin: 0;
4       padding: 0%;
5   }
6   #video-grid{
7       display: flex;
8       justify-content: center;
9       flex-wrap: wrap;
10  }
11
12  .Principal{
13      height: 100vh;
14      display: flex;
15  }
16  .Izquierda{
17      flex: 0.8;
18      display: flex;
19      flex-direction: column;
20  }
```

Código. 2.4 Hoja de estilos CSS

```
room.ejs
views > room.ejs > html > body > div.Principal.botonM
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" conter
6       <meta name="viewport" content="width=dev
7       <title>SERVICIO JORDY</title>
8       <link rel="stylesheet" href="style.css">
```

Código. 2.5 Relación entre "Room.ejs" y "Style.css"

- **Métodos de adquisición de datos locales**

Con la estructura y estilo de la aplicación ya definidos, se procede a escribir el código para obtener el contenido multimedia del participante. El código 2.6 muestra la implantación del método para realizar la obtención de datos multimedia.

Para el video local, se usa el método *document.createElement* propio de HTML, creando un objeto de tipo “video”. Los objetos creados pueden ser modificados mediante el método *setAttribute*; es necesario silenciar el video local, caso contrario se escuchará nuestro propio eco al momento de hablar.

Se crea una variable *getUserMedia,* que dispondrá de los métodos WebRTC *navigator.getUserMedia, navigator.mozGetUserMedia* y *navigator.webkitGetUserMedia;* los cuales tienen como función pedir permiso al usuario para usar los dispositivos multimedia solicitados (cámara, micrófono) ; si el usuario no concede permiso, la aplicación no podrá mostrar el video/audio del participante.

Los métodos mencionados funcionan para Google Chrome, Mozilla Firefox y Opera.

```
const mivideo=document.createElement('video');
mivideo.setAttribute("id",itera);
mivideo.setAttribute("width",width);
mivideo.setAttribute("height",height);
mivideo.muted=true;
var getUserMedia = navigator.getUserMedia ||
 navigator.webkitGetUserMedia || navigator.mozGetUserMedia;
navigator.mediaDevices.getUserMedia({
    video: true,
    audio: true
}).then(stream =>{
```

Código. 2.6 Declaraciones para obtención de datos multimedia

Se hace uso del método *document.getElementByID(‘video-grid’)*, para crear una variable *videoGrid,* la cual es encargada de agrupar los ítems de video para un mejor manejo en su presentación; el código 2.7 muestra la creación de este elemento, y cómo se agregan los videos al mismo mediante la función “incluirVideoStream” .

```
const videoGrid=document.getElementById('video-grid')
function incluirVideoStream (video,stream){
    video.srcObject= stream;
    video.addEventListener ('loadedmetadata',()=>{
        video.play();
    })
    videoGrid.appendChild(video);

}
```

Código. 2.7 Creación de ‘video-grid’ y función “IncluirVideoStream”

- **Manejo de tamaño en la aplicación**

La aplicación debe modificarse en base al número de participantes dentro la reunión, de esa manera el espacio de videos se distribuye para una mejor visualización. Se declara la función “relaciónAspecto” la cual está ejecutándose constantemente cada 5 segundos; esta llama a una función secundaria llamada “aspectototal”, su labor es contar los participantes dentro de la reunión y conforme a eso variar el tamaño de todos los objetos tipo “video”.

El código 2.8 indica los métodos utilizados para la función de relación aspecto en la aplicación.

```
484   //FUNCION PARA RELACION DE ASPECTO.
485   function relacionAspecto(){
486     setInterval(aspectototal,5000);
487   }
488   function aspectototal(){
489     var numeroVideo=$('video').length;
490
491       switch(numeroVideo){
492         case 1:
493         //  console.log("Solo hay un participante")
494           $('video').attr("height",540)
495           $('video').attr("width",730)
496
497           break;
498         case 2:
499          // console.log("Solo hay dos participantes")
500           $('video').attr("height",'auto')
501           $('video').attr("width",530)
502           break;
```

Código. 2.8 Métodos para el manejo de tamaño en la aplicación

2.2.1.2.1.3 Videoconferencia

Cuando un usuario decide ingresar a una videoconferencia, se realiza la apertura de una conexión *peer*, y se envía una solicitud al *socket* para pertenecer a la sala de videoconferencia. Con el usuario dentro de la reunión y una vez obtenido su *stream,* se realizan *calls* para enviar y recibir los *streams* correspondientes.

Los archivos importantes para realizar la funcionalidad de videoconferencia en la aplicación son: “server.js”, “script.js” y “room.ejs”. La descripción de los archivos y los métodos utilizados para iniciar la videoconferencia, se mencionan a continuación.

- **Script.js y room.ejs**

El archivo “Script.js” abarca el procedimiento a realizar en el usuario en su ingreso a la reunión; y “room.ejs” contiene el identificativo único de la sala. Se detallan los métodos utilizados:

- *Peer.on*: abre una conexión *peer* con un identificativo denominado “id”.
- *Socket*.emit: Hace un solicitud *emit* de nombre *join-room*, para acceder a la sala correspondiente, enviando el ID de la sala "ROOM-ID”, el cual está como variable constante en el “room.ejs” y enviando el identificativo “id” único del *peer.*

El código 2.9 indica los métodos antes mencionados y detalla el proceso correspondiente al inicio de la conexión de los usuarios.

```
peer.on('open',id =>{
    socket.emit('join-room',ROOM_ID,id);
    console.log("Usted se encuentra dentro de la sala, su id es : "+id);
    identi=id;
    roomIdx=ROOM_ID;
})
```

Código. 2.9 Métodos inicio de conexión en Script.js

- **Server.js**

El archivo dispone de todos los métodos para el manejo del *socket*. El código 2.10 muestra la serie de pasos que realiza el programa cuando un usuario se conecta al servicio.

```
io.on('connection',socket =>{
    socket.on('join-room',(roomId,userId)=>{
    socket.join(roomId,userId);
    socket.broadcast.to(roomId).emit('user-connected',userId);
```

Código. 2.10 Métodos relacionados para la conexión en el socket

Los métodos utilizados se detallan a continuación:

- *io.on* de nombre “*connection*”: detecta las nuevas conexión al *socket* , y escucha en caso de alguna solicitud *emit* provenientes de otros *scripts.*
- *socket.on* de nombre “*join-room*”: respuesta a la solicitud “*join-room*” realizada en la anterior sección por el archivo “script.js” para el ingreso del usuario a la sala de videoconferencia.
- *Socket.broadcast.to(roomId).emit(“user-connected”)*: método encargado de notificar a todos los usuarios pertenecientes a la sala “*roomId*” acerca del ingreso de un nuevo participante.

- **Manejo de *PEERS***

El archivo “script.js” dispone del código para el manejo de la comunicación *peer-to-peer* una vez iniciada la comunicación, así como de los métodos necesarios para una correcta conexión y desconexión de los usuarios en la comunicación *peer.*

A continuación, se esclarece el funcionamiento de los métodos más relevantes para el proceso de la comunicación *peer-to-peer.*

- *Socket.on* de nombre “*user-connected*”: respuesta a la solicitud realizada por el servidor; cuando un nuevo usuario ingresa de la sala, se ejecuta la función “conectarNuevoUsuario” por cada uno de los participantes de la reunión, pasando los parámetros de identificador del usuario (*userId*), y su *stream*. El código 2.11 expone lo implementado en el método.

```
socket.on('user-connected',(userId)=>{
  console.log("conectando nuevo usuario....");
  setTimeout(function ()
      {
      conectarNuevoUsuario(userId,stream);
      },1000)

})
```

Código. 2.11 Método "user-connected" para manejo de *peers*

- La función “conectarNuevoUsuario” recibe los parámetros de “*userId*” y “*stream*” de los participantes ya activos en la reunión; el código 2.12 muestra el proceso de la función.

- El método *peer.call* es el encargado de llamar al nuevo usuario que ingresó a la reunión y enviarle los *streams* de todos los demás participantes; el usuario que ingresó, podrá visualizar a todos los usuarios en la reunión.

- El método *call.on(‘stream’)* recibe el *stream* por parte del usuario que ingresó a la sala, para después colocarlo a la visualización del participante que ya estaba activo.

- El método *call.on(‘close’)* se ejecuta en caso de que se cierre la conexión desde el usuario que responde con su *stream*; su función es cerrar el objeto tipo “video” creado para el mismo.

```
const conectarNuevoUsuario =(userId,stream)=>{
        console.log("El usuario :"+userId+"ha ingresado a la sala");
        let call=peer.call(userId,stream)
        call.on('stream', userVideoStream =>{
            console.log('Recibiendo el stream de: ');
            console.log(userVideoStream.id);
            incluirVideoStream(video1,userVideoStream)
           //peeractual=call.peerConnection;
           peeractual.push(call.peerConnection);
           console.log(peeractual)
          })
        call.on('close',() =>{
            video1.remove();
        })
        peers[userId] = call
}
```

Código. 2.12 Función "conectarNuevoUsuario" en el archivo “script.js”

Los usuarios disponen de métodos *peer* para responder el llamado realizado por otros participantes. El código 2.13 indica el proceso de respuesta de una llamada *peer*.

- El método *peer.on(‘call’)* gestiona el llamado de otros usuarios.
- El método *call.answer* envía como respuesta el *stream* al emisor.

```
peer.on('call', call => {
    peers[call.peer]=call
    call.answer(stream)
    console.log('Envie mi stream');
    const video = document.createElement('video')
    itera++;
    console.log('mitad',itera);
    video.setAttribute("id", itera);
    video.setAttribute("ondblclick","TenerPantallaGigante(this)");
    call.on('stream', userVideoStream => {
        console.log('Ha recibido el stream de:');
        console.log(userVideoStream.id);
    incluirVideoStream(video, userVideoStream)

    peeractual.push(call.peerConnection);

   })
   call.on("close", () => {
    video.remove();
})
 })
```

Código. 2.13 Métodos asociados para respuesta *peer*

2.2.1.3 SPRINT 3

2.2.1.3.1 Planificación del Sprint 3

La Tabla 2.14 contiene las tareas estipuladas en el *Product Backlog*, y que deben ser realizadas para cumplir con el objetivo del *Sprint 3.*

Tabla 2.14 Planificación *Sprint* 3

ID	Tareas por realizar	Horas
HU-05	Métodos referentes para compartición de pantalla	10
	Manipulación de audio/video en la compartición de pantalla	10
HU-07	Métodos referentes al envío y recepción de mensajes en los archivos script.js y server.js	15

2.2.1.3.1.1 Presentación de la aplicación-Compartición de pantalla

- **Métodos para compartición de pantalla**

Una vez iniciada una reunión, se puede compartir la pantalla por parte de un usuario. La función para el botón de compartir pantalla se encuentra en el archivo “room.ejs”, y los métodos encargados de obtener la pantalla del usuario que comparte, se encuentran en el archivo “script.js”. El código 2.14, detalla función del botón para compartir pantalla.

```
<div onclick="SharePantalla()" class="BotonesDeControl">
    <i class="fas fa-photo-video"></i>
    <span>Compartir Pantalla</span>
</div>
```

Código. 2.14 Función de botón "Compartir Pantalla"

El usuario al dar *click* sobre el botón, activa la función *SharePantalla()* la cual es encargada de obtener la pantalla del usuario que comparte. El código 2.15 indica el proceso que realiza la función.

```
//FUNCION COMPARTIR PANTALLA
function SharePantalla(){
navigator.mediaDevices.getDisplayMedia({
  video:{
    cursor: true
  },
  audio:{
    echoCancellation:true,
    noiseSuppression:true
  }
  }).then((stream)=>{
    let videotrack=stream.getVideoTracks()[0];
    videotrack.onended = function(){
      DetenerComparticionPantalla();
    }
    ManejoComparticion(videotrack);
  }).catch((err)=>{
    console.log("No se pudo obtener el stream de comparticion");
  })
}
```

Código. 2.15 Procesos para compartición de pantalla

La función hace uso del método *navigator.mediaDevices.getDisplayMedia()* propio de WebRTC, solicitando al usuario que seleccione y otorgue permiso para realizar la captura de contenido de pantalla o parte de ella (como una ventana). Al ejecutar la función se coloca el atributo *cursor* seteado al valor *true*, lo cual permite que se pueda visualizar el movimiento del ratón en tiempo real, al momento de compartir pantalla.

Los atributos *echoCancellation* y *noiseSuppression* seteados en valores *true*, permiten cancelar el ruido y el eco procedentes de la compartición de pantalla.

Si el usuario accede a compartir su pantalla, se ejecutará la función *ManejoComparticion*, la cual se detalla en la siguiente sección. En caso de no ser posible la compartición de pantalla por cualquier motivo, se mostrará un mensaje de error en consola.

- **Manipulación de compartición y video**

Cuando el usuario accede a compartir pantalla, se pone en marcha la función *ManejoComparticion*, enviando la variable *videotrack.*

La variable *videotrack* selecciona y almacena el *stream* principal del usuario (video y audio), para una vez terminada la compartición, el usuario muestre nuevamente su *stream* original.

La función *ManejoComparticion* toma el video perteneciente a la cámara web para reemplazarlo por el video de la compartición o viceversa, dependiendo de la función que lo convoque (*SharePantalla-DetenerComparticionPantalla*), esto se detalla en el código 2.16. Los demás participantes de la reunión visualizarán el video correspondiente.

```
function ManejoComparticion(videotrack1){
  let videotrack=videotrack1;
  for(let x=0;x<peeractual.length;x++){
    var sender=peeractual[x].getSenders().find(function(s){
      return s.track.kind == videotrack.kind;
    })
    sender.replaceTrack(videotrack);
  }
}
```

Código. 2.16 Función *ManejoComparticion*

El código 2.17 muestra la función para detener la compartición de pantalla.

```
function DetenerComparticionPantalla(){
  console.log("COMPARTICION CANCELADA")
  let videotrack=myVideoStream.getVideoTracks()[0];
  ManejoComparticion(videotrack);
}
```

Código. 2.17 Función *DetenerComparticionPantalla*

2.2.1.3.1.2 Mensajería de la aplicación

La mensajería de la aplicación se basa en recibir y enviar mensajes por parte de los usuarios; los archivos primordiales para realizar esta tarea son los archivos: “script.js” y “server.js”

- **Envío de mensajes**

Los mensajes de los usuarios son enviados al servidor ubicado en el archivo “server.js”, donde las validaciones del mensaje se las realiza en el lado del cliente, antes de ser enviado al servidor. El código 2.18 muestra la función programada que realiza el envío de mensajes en el archivo “script.js”

El envío de mensajes puede ser mediante “Enter” o dando *click* a un botón específico; la lógica de programación se mantiene en ambas situaciones.

```
//FUNCIONALIDAD PARA ENVIAR MENSAJES
$('#MensajeDeChat').keydown(function(e){
    if(e.keyCode === 13){
        var usuariochat = document.getElementById('NombreUsuario').value.trim();
        if (usuariochat === '') {
            Swal.fire('ALTO!!','Coloque su nombre para acceder al chat','info');
            return false;
        }
        if ($('#MensajeDeChat').val() === '')
          return false;
        var datos= $('#MensajeDeChat').val();
        var resultado=datos.slice(0, 20);
        var datosglobales={
            'usuario': usuariochat,
            'mensaje': resultado
        }
        salidaparausuario=usuariochat;
            socket.emit('mensaje',datosglobales);
            $('#NombreUsuario').attr("disabled","disabled")
            $('#MensajeDeChat').val('');
    }
})
```

Código. 2.18 Función para envío de mensajes en el achivo “script.js”

Si el usuario decide enviar un mensaje, el programa ingresa a la función y comienza los métodos correspondientes; tales métodos se mencionan a continuación:

1. *usuario*=== *‘’*: verifica que la persona que envíe el mensaje haya colocado su nombre al iniciar la reunión; si el usuario no proporciona su nombre, se le notificará y no se enviará mensaje alguno.

2. *$(‘#MensajeDeChat’).val*: valida que el usuario no envíe mensajes en blanco a los demás participantes, también verifica que solo se envíen un máximo de 20 caracteres por mensaje.

3. *Socket.emit(‘mensaje’)*: método encargado de enviar el mensaje de texto al servidor, con su respectivo nombre de usuario.

Cuando el mensaje ha sido enviado por parte del usuario, el servidor toma los datos y los reenvía; el código 2.19 muestra el reenvío de mensajes mediante *broadcast* para todos los usuarios del sistema (excepto para quien lo envió).

```
socket.on('mensaje',mensaje =>{
    var d = new Date();
    var uno1=d.getHours();
    var dos2=d.getMinutes();
    var horatotal=uno1+':'+dos2;
    io.to(roomId).emit('MensajeCreado',mensaje,horatotal);
    })
```

Código. 2.19 Método para reenvío de mensajes en el archivo “server.js”- Broadcast

El método *io.to(roomId)* realiza el *broadcast* del mensaje, adicionando la hora actual del sistema.

- **Recepción de mensajes**

Cuando el servidor ha realizado el reenvío del mensaje con éxito, todos los participantes de la videoconferencia podrán visualizarlo en el panel de mensajes. La recepción de mensajes se detalla en el código 2.20.

El método *socket.on (‘MensajeCreado’)*, recibe desde el servidor los parámetros: “msg” y “horario”, los cuales utilizará para componer el mensaje con el texto y la hora actual, posteriormente se muestra al participante el mensaje completo.

```
socket.on('MensajeCreado',(msg,horario) =>{
    console.log('Mensaje desde el servidor:', msg);
    $("ul").append(`<li class="Mensajes"><b>${msg.usuario}</b><br/>${msg.mensaje}
    BotonBajada()
})
```

Código. 2.20 Método para recepción de mensajes en el archivo "script.js"

2.2.1.4 SPRINT 4

2.2.1.4.1 Planificación del Sprint 4

La Tabla 2.15 contiene las tareas estipuladas en el *Product Backlog*, y que deben ser realizadas para cumplir con el objetivo del *Sprint* 4

Tabla 2.15 Planificación *Sprint* 4

ID	Tareas por realizar	Horas
HU-08	Adición de software para controles	10
	Métodos referentes para controles de aplicación	10

2.2.1.4.1.1 Software para controles

Se utilizaron recursos de fuentes externas a las propias de HTML, éstos fueron añadidos mediante código en los *Scripts,* ejecutándose al momento de iniciar la aplicación.

Los recursos utilizados se indican a continuación:

- **Bootstrap**

Bootstrap es introducido en el archivo “room.ejs”, y permitió el uso de una hoja de estilo “css” más amigable para el usuario. El código 2.21 muestra la inclusión de Bootstrap en la aplicación.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
```

Código. 2.21 *Script* "Bootstrap" en el archivo "room.ejs"

- **SweetAlert**

SweetAlert permitió el uso de cuadros emergentes personalizables en JavaScript. El código 2.22 indica la inclusión del recurso “SweetAlert” para la aplicación.

```
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
```

Código. 2.22 *Script* "SweetAlert" en el archivo "room.ejs"

- **Popper**

Popper permitió el uso de un motor de posicionamiento de información, sobre herramientas y cuadros emergentes, y se aplica directamente sobre la hoja de estilos de la aplicación. La inclusión se realizó mediante el *script* obtenido de su página web oficial; el código 2.23 expone lo mencionado, Script "Popper" en el archivo "room.ejs"

```
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js"
```

Código. 2.23 *Script* "Popper" en el archivo "room.ejs"

- **FontAwesome**

FontAwesome permitió el uso de un biblioteca de íconos personalizables y sencillos de implementar. El código 2.24 muestra el *script* utilizado en la aplicación.

```
<script src="https://kit.fontawesome.com/e828ceb8af.js"
```

Código. 2.24 *Script* "FontAwesome" en el archivo "room.ejs"

- **ClipBoard**

ClipBoard permitió copiar texto al portapapeles de manera sencilla, sin marcos ni archivos *flash* adicionales. El código 2.25 indica el *script* utilizado en la aplicación.

```
<script src="https://cdn.jsdelivr.net/clipboard.js/1.5.3/clipboard.min.js"
```

Código. 2.25 *Script* "Clipboard" en el archivo "room.ejs"

Cabe recalcar que los *scripts* añadidos permitieron un mejor manejo de la aplicación en el campo de los controles.

2.2.1.4.1.2 Métodos para controles

Varios controles han sido implementados para el manejo de la aplicación, éstos permiten al usuario utilizar las herramientas del sistema. Los activadores de los controles se encuentran en el archivo “room.ejs”, y las funcionalidades están en el archivo “script.js”.

Los controles implementados se mencionan a continuación:

- **Barra de desplazamiento en el panel de mensajes**

La barra de desplazamiento permite al usuario mover el panel de mensajes para visualizar toda la conversación realizada en la reunión. Se recalca que la aplicación no almacena los mensajes; en caso de desconexión o reconexión de un usuario, este podrá ver la conversación a partir de su punto de ingreso en adelante. El código 2.26 detalla la función para la barra de desplazamiento.

```
//Control-Función Scrollable para panel de mensajes (Barra de desplazamiento)
function BotonBajada(){
    var botonparabajar= $('VentanaChat');
    botonparabajar.scrollTop(botonparabajar.prop("scrollHeight"));
}
```

Código. 2.26 Función para barra de desplazamiento en el archivo "script.js"

El método *scrollTop* permite la adición de una barra de deslizamiento en el panel de mensajes; en caso de no utilizar el método, el valor por defecto es 0.

- **Sonido de la aplicación- Desmutear / Mutear**

La aplicación permite a los participantes silenciar o activar su micrófono durante la reunión; se utiliza un solo botón para realizar ambas funciones. El código 2.27 muestra la implementación en HTML del botón en el archivo “room.ejs”. El código 2.28 indica la función “MuteoDesmuteo” en el archivo “script.js”, la cual se encarga de activar o desactivar el

micrófono del usuario según corresponda; la función se ejecuta una vez que se presiona el control.

```
<div onclick="MuteoDesmuteo()" class="BotonesDeControl botonMuteo">
    <i class="fas fa-microphone-alt"></i>
    <span>Voz</span>
</div>
```

Código. 2.27 Codificación HTML para el control de sonido

```
//Control-Función de sonido
function MuteoDesmuteo(){
    const activo=myVideoStream.getAudioTracks()[0].enabled;
    if (activo){
        myVideoStream.getAudioTracks()[0].enabled=false;
        console.log('VerificacionSonido')
        SeteoBotonMuteo();
    }else{
        SeteoBotonDesmuteo();
        myVideoStream.getAudioTracks()[0].enabled=true;
    }
}
```

Código. 2.28 Función "MuteoDesmuteo" para control de sonido

Los métodos y funciones secundarias utilizados en la función “MuteoDesmuteo” se detallan a continuación:

1. *getAudioTracks*: recoge el audio del usuario, para comprobar si el micrófono está silenciado o no, y llamar a la función correspondiente.
2. *SeteoBotonDesMuteo*: función encargada de activar el micrófono para que el usuario pueda hablar; cambia el icono mostrado, para notificarle al usuario que el micrófono está activado.
3. *SeteoBotonMuteo*: función encargada de desactivar el micrófono para que el usuario sea silenciado; cambia el icono mostrado, para notificarle al usuario que el micrófono está desactivado.

- **Cámara web**

La cámara web permite capturar la imagen del participante, para ser trasmitida a la videoconferencia. El sistema permite activar o desactivar el video del usuario. El código 2.29 expone la codificación HTML en el archivo “room.ejs” para el activador del control. El

código 2.30 indica la función para la activación/desactivación de la cámara web en el archivo “script.js”.

```
<div onclick="CambiosVideo()" class="BotonesDeControl botonVideo">
    <i class="fas fa-camera"></i>
    <span>Apagar Camara</span>
</div>
```

Código. 2.29 Codificación HTML para el control de cámara web

```
function CambiosVideo (){
  const activo=myVideoStream.getVideoTracks()[0].enabled;
  if (activo){
      myVideoStream.getVideoTracks()[0].enabled=false;
      DetenerVideo();
  }else{
      FuncionarVideo();
      myVideoStream.getVideoTracks()[0].enabled=true;
  }
}
```

Código. 2.30 Función "CambiosVideo" para control de sonido

Los métodos y funciones secundarias utilizadas en la función “CambiosVideo” se detallan a continuación:

1. *getVideoTracks*: recoge el video del usuario, para comprobar si está activado o no, y llamar a la función correspondiente.
2. *DetenerVideo:* función encargada de desactivar el video para que el usuario no pueda ser visualizado por los demás miembros de la sala; cambia el icono mostrado, para notificarle al usuario que el video está desactivado
3. *FuncionarVideo*: función encargada de activar el video para que el usuario pueda ser visualizado por los demás miembros de la sala; cambia el icono mostrado, para notificarle al usuario que el video está activado.

- **Salida de la aplicación**

El usuario dispone de un botón de salida, que le permitirá cerrar su conexión con los demás participantes; el activador del botón se encuentra en el archivo “room.ejs” y está indicado en el código 2.31.

La función “Salir” es la encargada de pedir la confirmación de salida al usuario mediante un cuadro emergente; si el usuario acepta, se realiza la desconexión del cliente en el *socket*

y se cierra la ventana de la reunión, caso contrario se mantendrá activa su conexión en la videoconferencia. El código 2.32 detalla la implementación de esta función.

```
<div class="SeccionDeControles">
    <div onclick="Salir()" class="BotonesDeControl">
        <i class="fas fa-door-open"></i>
        <span class="DejarReunion">Abandonar Conferencia</span>
    </div>
</div>
```

Código. 2.31 Codificación HTML para el control de salida

```
//Control-Función de salida
function Salir(){
  var jl=$('#NombreUsuario').val();
  var jl1=$('#MensajeDeChat').val();
  console.log(jl);
if (jl1 ===''){
  salidaparausuario=jl;
}
  socket.emit('salida',salidaparausuario,identi,roomIdx);
  Swal.fire({
  icon:'question',
  title:'Esta seguro de abandonar la sesion?',
  showCancelButton: true,
  confirmButtonText: 'SI!!',
  confirmButtonColor: '#32CD32',
  cancelButtonText: 'Cancelar',
  cancelButtonColor:'#FFA07A',
```

Código. 2.32 Función “Salir” para control de salida

El método *Swal.fire muestra* la ventana emergente personalizada para el usuario; atributos como: color, tipo de texto, título entre otros, pueden ser modificadas por el desarrollador para una interfaz más amigable.

- **Información**

El usuario puede acceder a un apartado de información, donde podrá visualizar datos de la persona que ha desarrollado el sistema. El código 2.33 muestra el activador del botón en el archivo “room.ejs”; el código 2.34 indica la función “información”, la cual se encarga de mostrar un cuadro emergente con los datos antes mencionados.

```
<div onclick="Informacion()" class="BotonesDeControl">
    <i class="fas fa-info-circle"></i>
    <span>Informacion</span>
</div>
```

Código. 2.33 Codificación HTML para el botón de información

```
function Informacion(){
  Swal.fire({
    title:'Aplicacion Desarrollada por: '+'',
    text:'Jordy Adrian Ramon Bedoya',
    showCancelButton: false,
    confirmButtonText: 'OK',
    confirmButtonColor: '#32CD32',});
}
```

Código. 2.34 Función "Informacion" para el botón de información

- **Levantar Mano**

El usuario dispone de un control para “levantar la mano”, al presionar el control se envía un *emoji* de “mano” a toda la sala, para que los participantes visualicen que ese usuario desea tomar la palabra; esto se aprecia en el código 2.35 el cual expone lo antes mencionado.

El código 2.36 muestra el activador HTML en el archivo “room.ejs”.

```
//Control-Función Levantar Mano
function LevantarMano(){
  var usuariochat = document.getElementById('NombreUsuario').value.trim();
  if (usuariochat === '') {
      Swal.fire('ALTO!!','Coloque su nombre para acceder al chat','info');
      return false;
  }
  var datos= "&#9995;";
  var datosglobales={
      'usuario': usuariochat,
      'mensaje': datos
  }
  salidaparausuario=usuariochat;
      socket.emit('mensaje',datosglobales);
      $('#NombreUsuario').attr("disabled","disabled")
      $('#MensajeDeChat').val('');
}
```

Código. 2.35 Función "Información" para el botón de información

```
<div onclick="LevantarMano()" class="BotonesDeControl">
    <i class="far fa-hand-paper"></i>
    <span>LevantarMano</span>
</div>
```

Código. 2.36 Codificación HTML para el botón de “Levantar la mano”

- **Invitar Amigos**

Los usuarios tienen la posibilidad de invitar a más amigos o conocidos a la reunión, para hacerlo el sistema incluye un control que permite la copia del enlace de la reunión, al portapapeles del usuario; esto se detalla en el código 2.37.

El código 2.38 muestra el activador del control, ubicado en el archivo “room.ejs”

```
//Control-Función Invitar Amigos
function InvitarAmigos(){
document.getElementById("enlace").value=window.location.href
var enlace=document.getElementById("enlace")
  Swal.fire({
  title: "Invita a mas amigos!!"+'&#128525;',
  text: "LINK:  "+enlaceAmigos,
  showCancelButton: true,
  confirmButtonText: 'COPIAR!!'+'&#x1f919;',
  confirmButtonColor: '#32CD32',
  cancelButtonText: 'Cancelar'+'&#128528',
  cancelButtonColor:'#FFA07A',
  showLoaderOnConfirm: false,
  allowOutsideClick: true,
  allowEscapeKey: true
    }).then(function(result){
        if(result.isConfirmed){
          enlace.select();
          enlace.setSelectionRange(0,99999);
          document.execCommand('copy');
          Swal.fire({
            title:'LISTO!!'+'&#x1f60e',
            text:'Envia el link a tus amigos para que puedan acceder a la sala!!',
            confirmButtonText: '&#x1f44c',
            confirmButtonColor: '#32CD32'})
      }
    });
  }
```

Código. 2.37 Función "InvitarAmigos" en el archivo “script.js”

```
<div onclick="InvitarAmigos()" class="BotonesDeControl">
    <i class="fas fa-users-cog"></i>
    <span>Invitar Amigos</span>
</div>
```

Código. 2.38 Codificación HTML para el botón de “Invitar Amigos”

- **Envío de mensajes**

El usuario dispone de un botón para enviar mensajes. El código 2.39 indica la codificación HTML activador del botón, ubicado en el archivo “room.ejs”. La funcionalidad del botón es la misma que se realiza el presionar la tecla “enter”.

```
<button onclick="BotonEnvioSMS()" class="BotonEnvioSMS" id="BotonEnvioSMS">
    <i class="far fa-paper-plane"></i>
</button>
```

Código. 2.39 Codificación HTML para el botón “EnvioSMS”

2.2.1.5 SPRINT 5

2.2.1.5.1 Planificación del Sprint 5

La Tabla 2.16 contiene las tareas estipuladas en el *Product Backlog*, y que deben ser realizadas para cumplir con el objetivo del *Sprint* 5.

Tabla 2.16 Planificación *Sprint* 5

ID	Tareas por realizar	Horas
HU-09	Adquisición y configuración de un servidor virtual para soporte de STUN/TURN	6
	Instalación de Servidor STUN/TURN y relación con el sistema	14

2.2.1.5.1.1 Servidor Virtual

Para tener un servidor virtual propio, se usaron los siguientes componentes:

1. Un servidor Ubuntu.
2. Dirección IP pública del servidor.

Se alquiló un servidor virtual en la plataforma Digital Ocean con características básicas para alojar el servicio STUN/TURN. La configuración del servidor virtual seleccionada es: Ubuntu 18.04 versión LTS de 64 bits, 2 GB de RAM, 50 GB de disco duro; se encuentra ubicado en el servidor de Londres de Digital Ocean. La gráfica 2.33 muestra el detalle de la configuración del servidor.

Figura. 2.33 Características del servidor virtual alquilado

La plataforma Digital Ocean ofrece en alquiler servidores virtuales, con su propia dirección IPv4 pública y IPv6 pública (Figura 2.34); esto permite que los usuarios de la aplicación pueden solicitar el servicio a través de Internet.

Figura. 2.34 Dirección IP pública del servidor

El servicio STUN/TURN requiere el servidor con todos sus paquetes actualizados; con el servidor ya activo, se realizó la actualización del *software* del servidor mediante los comandos *apt-get update* (figura 2.35) y *apt-get upgrade* (figura 2.36).

```
root@ubuntu-c-4-8gib-lon1-01:~# apt-get update
Get:1 http://mirrors.digitalocean.com/ubuntu bionic InRelease [242 kB]
Hit:2 https://repos.insights.digitalocean.com/apt/do-agent main InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://mirrors.digitalocean.com/ubuntu bionic-updates InRelease
Hit:5 http://mirrors.digitalocean.com/ubuntu bionic-backports InRelease
Fetched 242 kB in 1s (183 kB/s)
Reading package lists... Done
root@ubuntu-c-4-8gib-lon1-01:~#
```

Figura. 2.35 *apt-get update* para el servidor virtual

```
root@ubuntu-c-4-8gib-lon1-01:~# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  ubuntu-advantage-tools
The following packages will be upgraded:
  base-files libmysqlclient20 motd-news-config python-apt-common py
  ubuntu-release-upgrader-core update-notifier-common
10 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Need to get 1648 kB of archives.
After this operation, 3072 B of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Figura. 2.36 *apt-get upgrade* para el servidor virtual

2.2.1.5.1.2 Servicio STUN/TURN

El servidor STUN/TURN es la parte más importante para el correcto funcionamiento de la aplicación; el servidor es el encargado de permitir la conexión entre usuarios ubicados en distintas redes locales, detrás de *firewalls* o NAT's simétricos.

- **Proyecto Coturn**

El proyecto Coturn es una implementación de código abierto para un servidor STUN/TURN, funcional para WebRTC. La figura 2.37 muestra el comando a ejecutar para instalar Coturn.

```
root@ubuntu-c-4-8gib-lon1-01:~# sudo apt-get install coturn
Reading package lists... Done
Building dependency tree
Reading state information... Done
coturn is already the newest version (4.5.0.7-1ubuntu2.18.04.3).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

Figura. 2.37 Comando para instalar Coturn

Una vez instalado Coturn, se procede a configurar su servicio; lo primero en activar es el servidor *TURN* entrando al archivo de configuración "/etc/default/coturn", y colocando la línea "TURNSERVER_ENABLE=1". La figura 2.38 muestra el proceso realizado.

```
root@ubuntu-c-4-8gib-lon1-01:~# vim /etc/default/coturn

# Uncomment it if you want to have the turnserver running as
# an automatic system service daemon
#
TURNSERVER_ENABLED=1
```

Figura. 2.38 Activación del servidor TURN

Se crea un usuario permanente para el servicio, su función es permitir conectarse a los usuarios de la aplicación; se conectan al servidor mediante credenciales.

Turnadmin es la herramienta que viene incluida dentro de Coturn que permite la creación de usuarios; el comando para hacerlo es "turnadmin -a -u NOMBREUSUARIO -r NOMBREDOMINIO -p CONTRASEÑA". El usuario creado se detalla en la figura 2.39.

```
user=jordyadrianstudentEPN:14as78df95g26sad7
```

Figura. 2.39 Usuario creado para el servidor STUN/TURN

Donde el nombre de usuario es "jordyadrianstudentEPN" y su contraseña es "14as78df95g26sad7", las credenciales son colocadas en el archivo "script.js" para señalar a la aplicación el dominio al que debe apuntar y cuáles son las credenciales para acceder al servicio.

- **Registros o Subdominios para administrar el DNS- (STUN-TURN)**

El dominio utilizado es "turnservejr.com", se encuentra alojado en la plataforma GoDaddy y permite la configuración del DNS.

La aplicación para realizar las conexiones *peer-to-peer,* solicita primero el servicio STUN en caso de no poder hacerlo, solicita el servicio TURN, el cual tiene una mayor probabilidad de éxito. Esto se configuro en los subdominios “stun” y “turn”.

La figura 2.40 detalla la configuración DNS a utilizar; los registros A vinculan al dominio con la dirección IP pública del servidor. Las peticiones realizadas a “turn.turnservejr.com” o “stun.turnservejr.com”, son respondidas por la dirección IP del servidor.

Figura. 2.40 Administración DNS del dominio

La configuración del dominio en el archivo del servidor se detalla más adelante.

- **Certificados SSL para subdominios**

Es necesario tener certificados SSL para los subdominios utilizados. La implementación del servidor y la comunicación WebRTC necesitan del protocolo seguro HTTPS; Cerbot es la herramienta utilizada para la creación de certificados SSL. La figura 2.41 indica los certificados creados para el servidor STUN/TURN.

Cabe recalcar que la tecnología WebRTC no funciona en la Internet si no tiene protocolo seguro HTTPS.

Figura. 2.41 Certificado y llaves para el servidor STUN/TURN

- **Archivo de configuración para servicio STUN/TURN**

En esta sección se detalla la primera parte de las configuraciones, para el servicio STUN/TURN.

1. Para configurar el servicio STUN/TURN es necesario modificar el archivo de configuración “turnserver.conf”; para ingresar se puede utilizar cualquier editor visual (vim, cat, etc.). El comando utilizado fue “vim /etc/turnserver.conf”.
2. El puerto de escucha colocado es 3478 y debe permitir tráfico UDP.
3. El puerto designado para TLS es el 5349.

4. Los mecanismos de autenticación por defecto son: *fingerprint* y *lt-cred-match.*
5. Se añade un mecanismo de autenticación de credenciales a largo plazo mediante un usuario de nombre “jordyadrianstudentEPN” y contraseña “14as78df95g26sad7”.
6. El nombre del servidor y el dato *realm* son seteados con el nombre del dominio que se posee; el dominio utilizado es “turnservejr.com”.
7. Las configuraciones *total-quota* y *stale-nonce* son valores predeterminados por el archivo de configuración.

Las configuraciones mencionadas se detallan en la figura 2.42.

```
# Allow connection on the UDP port 3478
listening-port=3478
# and 5349 for TLS (secure)
tls-listening-port=5349

# Require authentication
fingerprint
lt-cred-mech
user=jordyadrianstudentEPN:14as78df95g26sad7
# Specify the server name and the realm that will be used
# if is your first time configuring, just use the domain as name
server-name=turnservejr.com
realm=turnservejr.com

total-quota=100
stale-nonce=600
```

Figura. 2.42 Primera parte del archivo de configuración “turnserver.conf”

Para la seguridad del servicio, se coloca el certificado con su llave correspondiente, creadas mediante la herramienta Certbot, y se especifica la lista de cifrados permitidos para las conexiones TLS y DTLS. La figura 2.43 muestra la segunda parte del archivo de configuración.

```
# Path to the SSL certificate and private key. In this example we will use
# the letsencrypt generated certificate files.
cert=/etc/letsencrypt/live/turnservejr.com/fullchain.pem
pkey=/etc/letsencrypt/live/turnservejr.com/privkey.pem
#pkey=/usr/local/psa/var/modules/letsencrypt/etc/live/ourcodeworld.com/privkey.pem

# Specify the allowed OpenSSL cipher list for TLS/DTLS connections
cipher-list="ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256

# Specify the process user and group
proc-user=turnserver
proc-group=turnserver
```

Figura. 2.43 Segunda parte del archivo de configuración “turnserver.conf”

Una vez establecido el software para el funcionamiento del prototipo y codificados los módulos, se procede a realizar el despliegue del prototipo en los ambientes local y *cloud*.

2.2.2 DESPLIEGUE LOCAL

La finalidad del despliegue local es conocer cómo se desenvuelve el sistema al ponerse en marcha, con sus respectivos componentes en un ambiente controlado.

- **Servidores de señalización**

En Internet existen múltiples sitios web que ofrecen servidores STUN gratuitos, entre ellos se encuentran: Google, 3cx, nextcloud, entre otros. En el despliegue local, los dispositivos al encontrarse dentro de la misma red usaron servidores STUN provistos por Google para la conexión.

La configuración de los servidores se encuentra en el archivo “script.js”; el código 2.40 muestra los servidores utilizados.

```
const servidores ={
  config: {'iceServers': [
  { url: 'stun:stun.l.google.com:19302' },
  {url: 'stun:stun1.l.google.com:19302'},
  {url: 'stun:stun2.l.google.com:19302'},
```

Código. 2.40 Servidores STUN de Google para el despliegue local

- **Puerto del servidor Express**

El servidor de la aplicación fue seteado con el puerto 3030 para facilidad del despliegue local; el número del puerto puede cambiar si el desarrollador así lo considera. El código 2.41 indica el puerto de escucha del servidor Express en el archivo “server.js”.

```
JS server.js M ●
JS server.js > ...
59  server.listen(3030);
```

Código. 2.41 Puerto escucha del servidor Express para el despliegue local

- **Firewalls y Antivirus**

Para el despliegue local, se desactivaron *firewalls* y antivirus de los dispositivos, para asegurar que las conexiones funcionen adecuadamente. Cabe recalcar que en la configuración *cloud*, este proceso ya no será necesario, puesto que el servidor TURN ya entrará en funcionamiento. La figura 2.44 muestra la desactivación del *firewall* y antivirus de uno de los dispositivos.

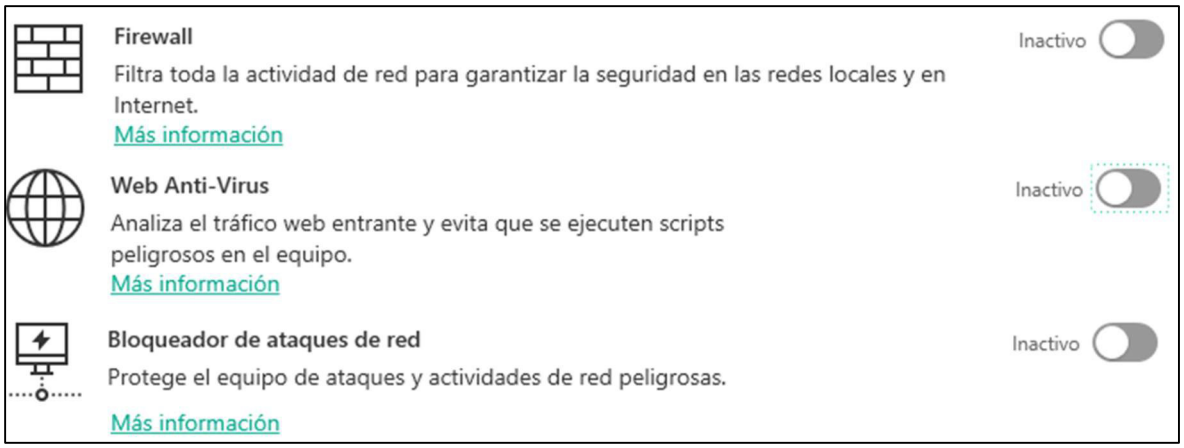


Figura. 2.44 *Firewall* y antivirus desactivados de un dispositivo

2.2.3 DESPLIEGUE CLOUD

Con la verificación del correcto funcionamiento de la aplicación en el ambiente local, se procede a su despliegue en un ambiente *cloud*, para lo cual se usaron los servicios ya configurados de Heroku, GoDaddy y Digital Ocean.

Heroku fue seleccionada por permitir ejecutar la aplicación dentro de contenedores inteligentes, los cuales tienen un entorno de ejecución totalmente administrable. Por otro lado, GoDaddy se escogió por ser una de las empresas más grande en registros de dominios de Internet, posee una política de privacidad completa, así como mecanismos de monitoreo y detección que previenen amenazas a sus servicios, cual otorga seguridad al momento de adquirir el dominio, así como permitir la creación de páginas web gratuitas con plantillas establecidas. Finalmente, Digital Ocean fue seleccionada por permitir alquilar servidores virtual, los cuales son privados ya que la empresa no se involucra en la gestión de sus servidores alquilados, y al poseer múltiples centros de cómputos, ofrece las imágenes de los principales sistemas operativos (Windows, Linux, etc.), junto con sus repositorios de manera local.

El objetivo principal del proyecto es tener una aplicación en ambiente *cloud*, al cual cualquier usuario pueda acceder solo con su conexión a Internet.

2.2.3.1 Gestión del contenedor- Heroku

Una vez creado y configurado el contenedor de Heroku, se procede a realizar la subida de los *scripts* correspondientes a la aplicación, para lo cual, se debió relacionar el proyecto de la máquina local con el contenedor de la plataforma.

Se instala el software “Heroku CLI” para Windows de 64 bits propio de la plataforma, que permite manejar los contenedores directamente desde el terminal de una computadora.

Una vez realizada la instalación se ingresa al proyecto, y en su terminal se ejecuta “heroku login” para acceder a la plataforma.

Con el ingreso realizado, se procede a utilizar los comandos para usar “Heroku CLI”, y relacionar el proyecto con el contenedor; dichos comandos se muestran en la figura 2.45.

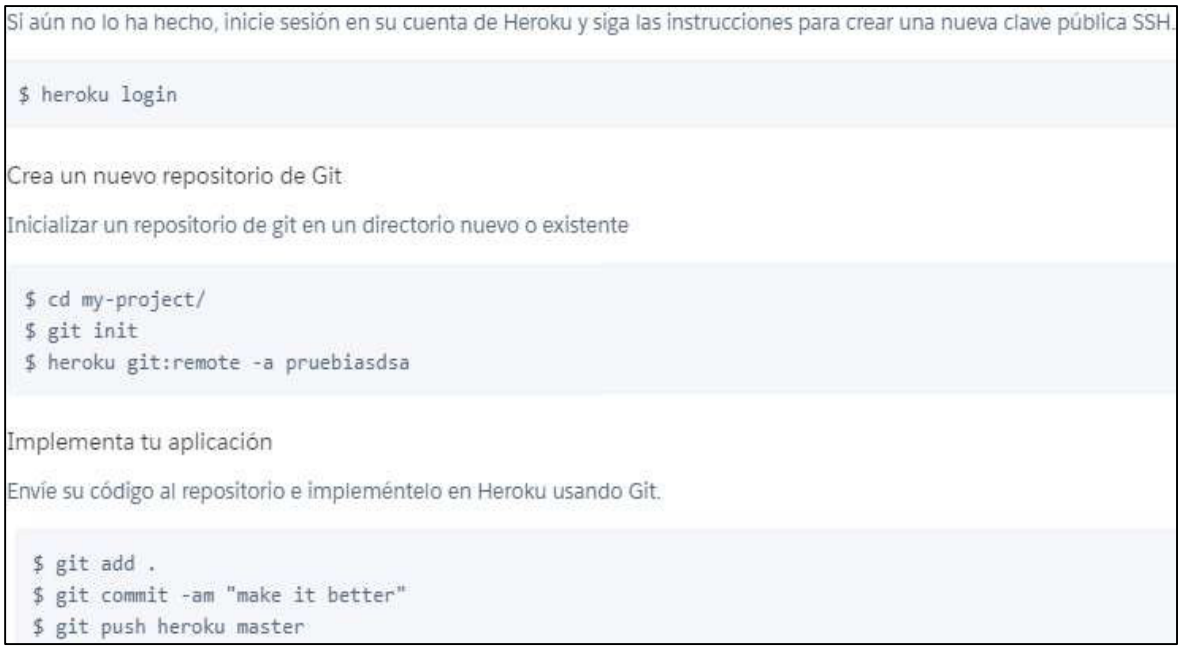


Figura. 2.45 Comandos de manejo para CLI

A través de la consola, el comando “git push heroku master”, permite subir el código al contenedor de Heroku e instalar las dependencias necesarias. La figura 2.46 indica el proceso de subida para la aplicación mediante consola.

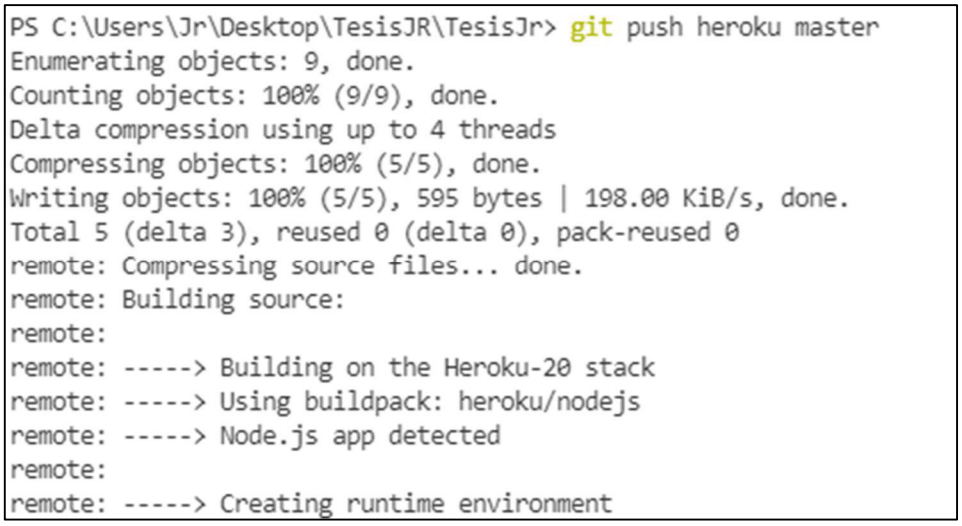


Figura. 2.46 Subida de la aplicación mediante consola

La figura 2.47 expone la verificación de que la aplicación ha sido colocada correctamente en el contenedor, mediante la página web de la plataforma.

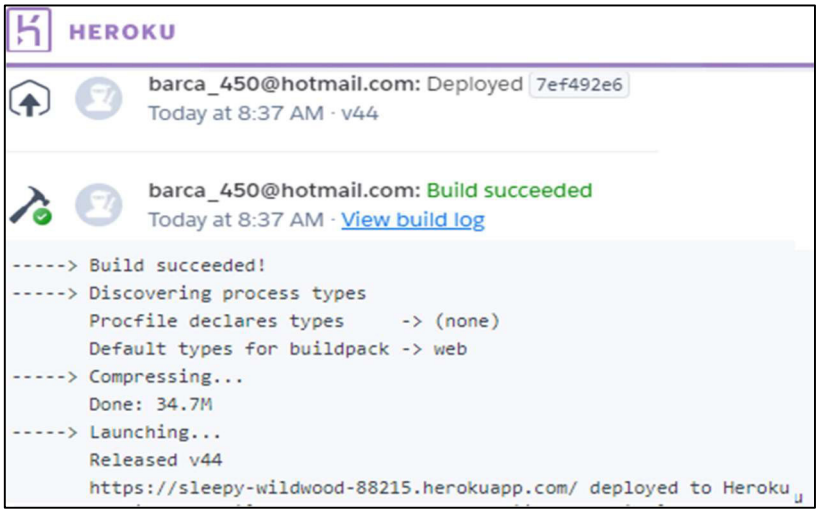


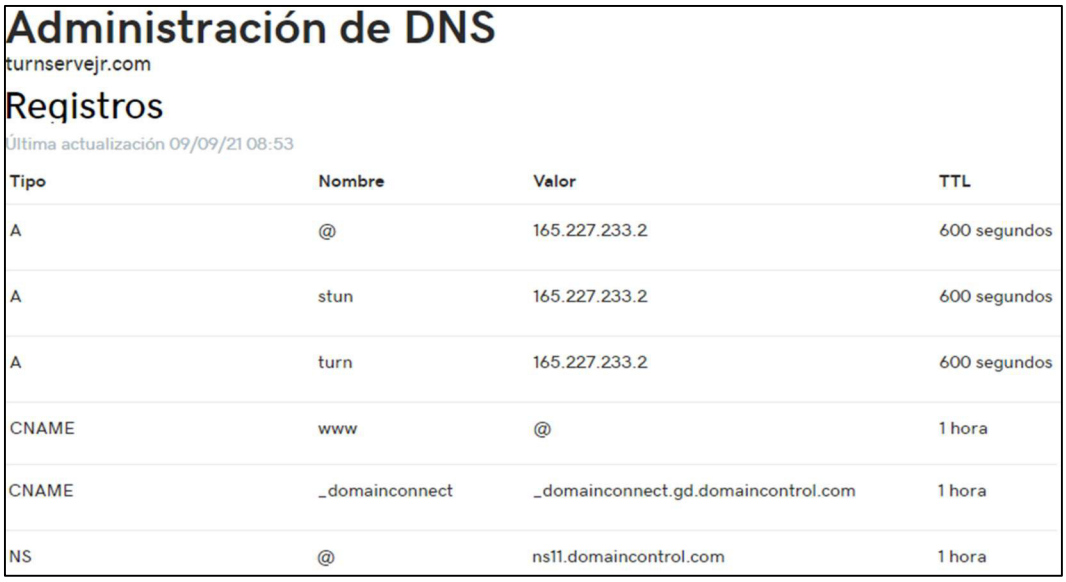
Figura. 2.47 Verificación del despliegue correcto de la aplicación

2.2.3.2 Configuración del sistema

- **Administración DNS-GoDaddy**

Con la aplicación subida a la plataforma Heroku, y la configuración del DNS del dominio ya realizada (Figura 2.48), se modifica el archivo "script.js" colocando los servicios STUN/TURN creados. La figura 2.49 indica la modificación producida.

En el ambiente *cloud* ya es necesario tener un servidor STUN/TURN activo, puesto que se buscará crear conexiones con usuarios fuera de la red local.



Tipo	Nombre	Valor	TTL
A	@	165.227.233.2	600 segundos
A	stun	165.227.233.2	600 segundos
A	turn	165.227.233.2	600 segundos
CNAME	www	@	1 hora
CNAME	_domainconnect	_domainconnect.gd.domaincontrol.com	1 hora
NS	@	ns11.domaincontrol.com	1 hora

Figura. 2.48 Configuración DNS de dominio "turnservejr.com"

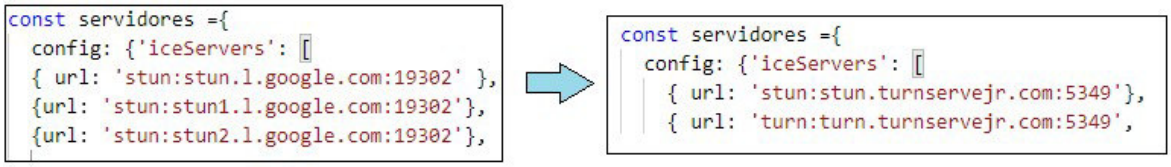


Figura. 2.49 Modificación de los servidores STUN/TURN en el archivo "script.js"

La figura 2.50 muestra las credenciales que lleva subdominio “turn” para poder acceder al servidor STUN/TURN.

Figura. 2.50 Credenciales para acceso del servicio TURN

- **Puerto de la aplicación**

Luego de la gestión del contenedor en Heroku y la configuración de servidores, se realiza la modificación en el puerto de escucha del servidor Express. La figura 2.51 indica el cambio realizado en archivo “server.js” para el puerto de escucha.

Figura. 2.51 Modificación Puerto del servidor Express

La evidencia del despliegue *cloud* con sus respectivas gráficas y uso del prototipo son detalladas en el capítulo siguiente.

3 RESULTADOS Y DISCUSIÓN

Este capítulo contiene las pruebas realizadas al sistema con su respectivo análisis de resultados. La metodología Scrum fue útil en lo realizado; las historias de usuario fueron verificadas una vez terminadas cada una de las tareas estipuladas en el *sprint*.

3.1 PRUEBAS DE FUNCIONAMIENTO

En esta sección se presentan las demostraciones que verifican el adecuado funcionamiento del sistema y cada uno de sus módulos.

Los criterios de aceptación de cada historia de usuario son considerados para cada una de las PF (pruebas de funcionamiento).

Las pruebas de funcionamiento han sido colocadas por secciones, basadas en las tareas realizadas por cada *Sprint*, las mismas que se indicaron en el capítulo 2.

3.1.1 RESULTADOS SPRINT 1

Se los obtiene a partir de las historias de usuario del *sprint* 1, las cuales son especificadas en el *Product Backlog* (Tabla 2.11).

Las historias de usuario se detallan en la tabla 3.1, con su respectivo identificador de la prueba de funcionamiento.

Tabla 3.1 Historia de usuario del *sprint* 1

Identificativo HU	Identificativo PF
HU-02	PF-01
HU-03	PF-02

3.1.1.1 Prueba de funcionamiento para la conexión del servicio de *hosting*-GoDaddy (PF-01)

Esta prueba de funcionamiento corresponde a la historia de usuario 02.

- **Prueba de conexión al servicio de hosting**

Esta prueba permite verificar la correcta conexión al servicio de *hosting*, la cual está relacionada a la conexión de los clientes al servicio mencionado mediante la página web que proporciona la plataforma GoDaddy, la cual se encuentra configurada con las

características de la aplicación. Para esta historia de usuario, el cliente debe poseer acceso a Internet.

En la figura 3.1 se muestra la página web creada en GoDaddy, la misma que se encuentra publicada y puede ser accedida por los clientes.

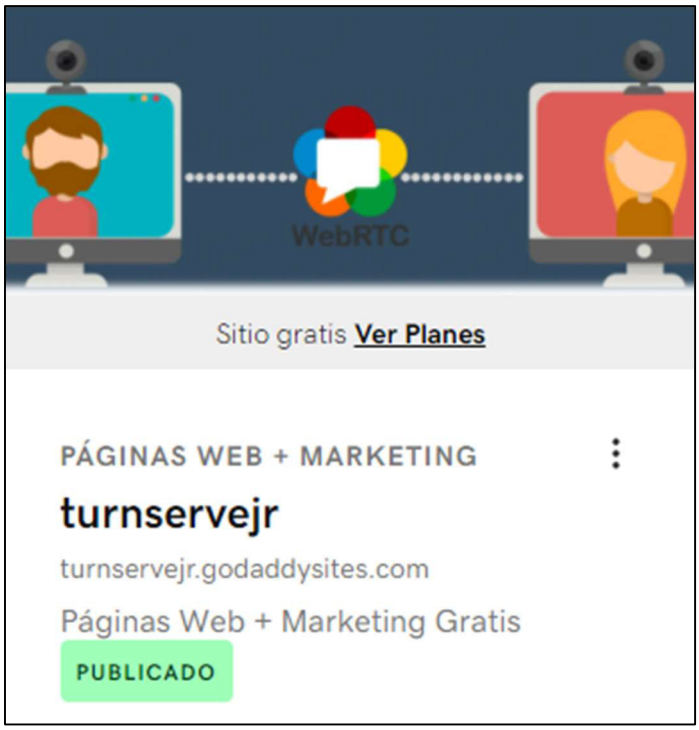


Figura. 3.1 Sitio web otorgado por GoDaddy listo para usar

Cabe recalcar que los clientes no podrán buscar la aplicación en Google, puesto que para aparecer en las opciones de búsqueda de Google se debe ser un sitio con muchas búsquedas de los usuarios, o a su vez, se debe cancelar un pago para ser incluido en el listado de búsqueda. Al ser éste un trabajo de titulación, este pago no fue realizado, por lo cual los clientes deberán acceder al servicio de *hosting* mediante el uso de una URL (Figura 3.2).

Se puede verificar que el sitio web creado, contiene el nombre del dominio adquirido.

Figura. 3.2 Enlace URL para acceso al servicio de *hosting*

Una vez accedido a la URL, el usuario ingresará al servicio de *hosting*, y podrá visualizar la página otorgada por GoDaddy. La figura 3.3 muestra la prueba de conexión del cliente al servicio de *hosting*.

Figura. 3.3 Resultado de la prueba de conexión del cliente al servicio de *hosting*

3.1.1.2 Pruebas de funcionamiento para la conexión del servidor web-Heroku (PF-02)

Estas pruebas de funcionamiento corresponden a la historia de usuario 03.

- **Prueba de adquisición de scripts**

Los usuarios pueden solicitar a la plataforma Heroku los *scripts* correspondientes para ejecutar la aplicación en su computadora y crear una sala de videoconferencias única. Cada petición realizada a Heroku, iniciará una nueva reunión con su identificativo propio.

El usuario inicia la petición, mediante un botón configurado en la página principal, éste permitirá acceder al servicio de Heroku, el cual proporciona los *scripts* necesarios.

La figura 3.4 indica la petición del usuario a la plataforma Heroku, para obtener el servicio del servidor web y cómo éste responde enviando los *scripts* para su ejecución.

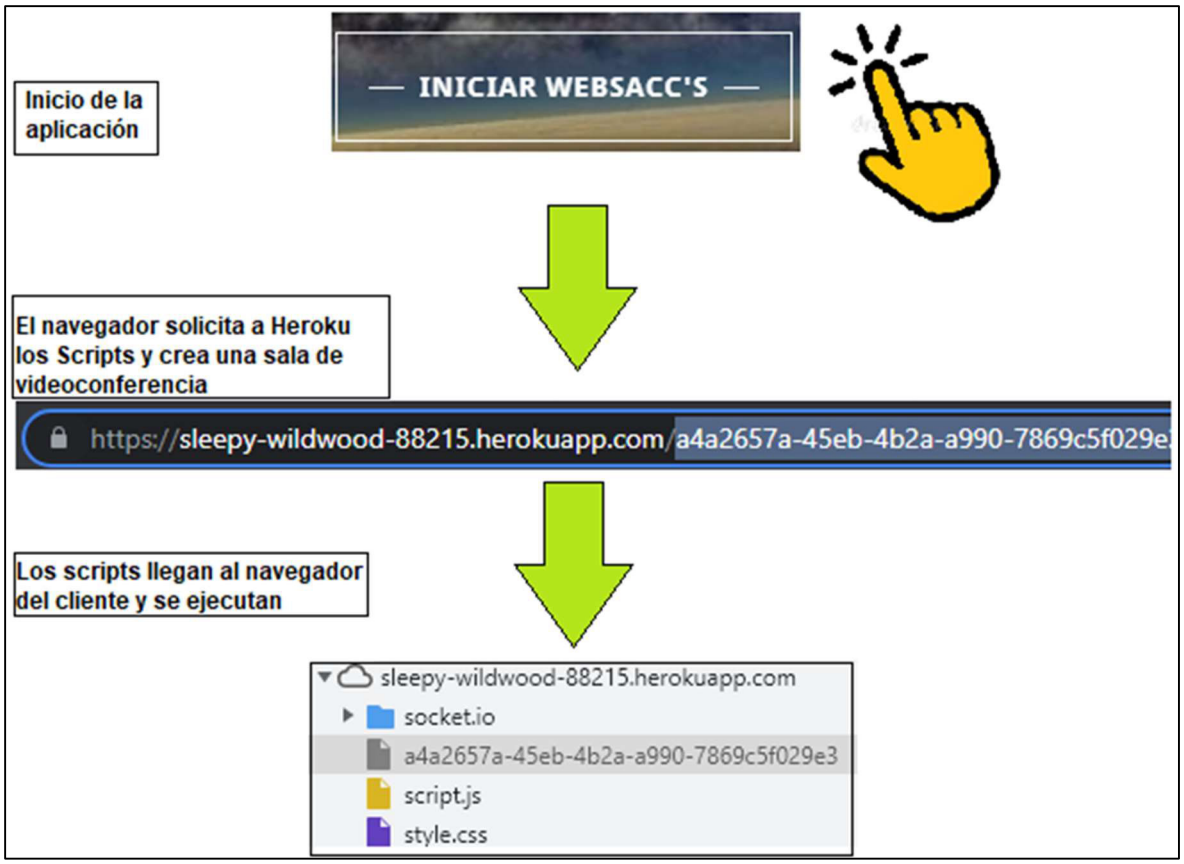


Figura. 3.4 Prueba de adquisición de *scripts* de la plataforma Heroku

El resultado fue el esperado ya que los clientes pueden conectarse correctamente a la plataforma Heroku.

- **Prueba de creación de múltiples salas**

Esta prueba se realiza con el fin de asegurar que, al presionar el botón de inicio de la aplicación, se creen siempre salas únicas con su propio identificativo.

La figura 3.5 expone que cada solicitud a Heroku, crea una nueva sala única.

Figura. 3.5 Prueba de creación de múltiples salas

Se observa que las salas creadas son únicas y poseen su propio identificativo.

3.1.2 RESULTADOS SPRINT 2

Los resultados se obtienen a partir de las pruebas de funcionamiento de las historias de usuario del *Sprint* 2, las cuales son especificadas en el *Product Backlog* (Tabla 2.11).

Las historias de usuario se detallan en la tabla 3.2, con su respectivo identificador de la prueba de funcionamiento.

Tabla 3.2 Historia de usuario del *Sprint* 2

Identificativo HU	Identificativo PF
HU-01	PF-03
HU-04	PF-04
HU-06	PF-05

3.1.2.1 Pruebas de funcionamiento de la página principal de la aplicación (PF-03)

Estas pruebas de funcionamiento corresponden a la historia de usuario 01.

- **Prueba de visualización de la página de inicio del sistema**

La página de inicio fue creada mediante las plantillas gratuitas que ofrece el servicio de *hosting* GoDaddy; el nombre de la página de inicio contiene el nombre de la aplicación “WebSaccs”, y su URL es: http://websaccs.godaddysites.com/.

La prueba corresponde al acceso de un cliente a la página principal de la aplicación; se comprueba que el usuario puede visualizar la página por defecto de ingreso al sistema, así como que el botón permite crear salas de videoconferencia.

La figura 3.6 indica la prueba realizada de ingreso a la página principal de la aplicación. Es importante mencionar que el usuario requiere acceder por la URL de la página, y puede hacerlo mediante el uso del navegador de preferencia; sin embargo, por recomendación técnica se recomienda utilizar los navegadores: Google Chrome, Mozilla Firefox, y Opera, puesto que estos navegadores son los que trabajan con la tecnología WebRTC.

Las pruebas realizadas en los diversos *sprints* hacen uso de los tres navegadores antes mencionados.

Figura. 3.6 Prueba de visualización de la página principal de la aplicación

Se verifica que la página de inicio se la muestra con éxito, siendo su resultado el esperado.

- **Prueba de visualización de la información acerca de la aplicación al usuario**

La página principal debe mostrar información acerca de la aplicación; las pruebas realizadas se muestran a continuación.

La figura 3.7 presenta información acerca de la aplicación, el nombre de la persona que la desarrolló, así como una breve descripción de su uso.

Figura. 3.7 Prueba de visualización de información acerca de la aplicación

La figura 3.8, presenta los servicios que dispone la aplicación tales como videoconferencia, *chat* en vivo, compartición de pantalla, entre los principales que se muestran.

Figura. 3.8 Prueba de visualización de servicios de la aplicación

La figura 3.9 expone información de cómo crear una sala de videoconferencia personalizada con un identificativo propio colocado por el usuario.

Figura. 3.9 Prueba de visualización para realizar salas personalizadas

Se observa que las pruebas fueron exitosas, siendo su resultado el esperado en la historia de usuario 01, cumpliéndose los criterios de aceptación.

3.1.2.2 Pruebas de funcionamiento para la presentación de la aplicación - Sección Multimedia (PF-04)

Estas pruebas de funcionamiento corresponden a la historia de usuario 04.

Se recalca que las pruebas de sección multimedia se las realiza de manera local, en el "localhost"; las pruebas en ambiente *cloud* se detallan posteriormente en las pruebas de la historia de usuario 09-Servidor Stun/Turn.

- **Pruebas de visualización de video y modificación de tamaño del video**

Estas pruebas muestran que el audio y el video del usuario pueden ser obtenidos por la aplicación, y se presenta la imagen de los usuarios que se encuentran dentro de la videoconferencia.

La figura 3.10 (a) muestra la sala de videoconferencia con un solo usuario, dicho usuario ocupa la mayor parte del área de la pantalla. La figura 3.10 (b) muestra la sala de videoconferencia con dos usuarios, la parte del área de la pantalla ya es ocupada por ambos usuarios. La figura 3.10(c) muestra la sala de videoconferencia con 3 usuarios, y cada usuario tiene su propia posición en la pantalla; finalmente la figura 3.10 (d) contiene la sala de videoconferencia con 4 usuarios.

Para esta prueba se evidencia que sus resultados son los esperados, verificándose que los videos de los participantes adquieren su tamaño acorde al número de usuarios en la sala. Los criterios de aceptación de la historia de usuario se han cumplido con éxito, siendo su resultado el esperado.

Figura. 3.10 Prueba de visualización de video y de modificación de tamaño del video

3.1.2.3 Pruebas de funcionamiento presentación de la aplicación - Compartición de pantalla (PF-05)

Estas pruebas de funcionamiento corresponden a la historia de usuario 06.

Se recalca que las pruebas de compartición de pantalla se realizan de manera local, en el "localhost"; las pruebas en ambiente *cloud* se detallarán en las pruebas de la historia de usuario 09-Servidor Stun/Turn.

- **Pruebas de compartición de pantalla con múltiples clientes**

Las siguientes prueban indican la compartición de pantalla realizada por uno de los usuarios, para el resto de los participantes de la videoconferencia. La figura 3.11(a), muestra la compartición de pantalla de un usuario con otro (2 usuarios en total). La figura 3.11(b), refleja la compartición de pantalla de un usuario con dos participantes adicionales

(3 usuarios en total); Finalmente la figura 3.11(c) muestra la prueba de compartición de pantalla de un usuario con 3 usuarios adicionales (4 usuarios en total).

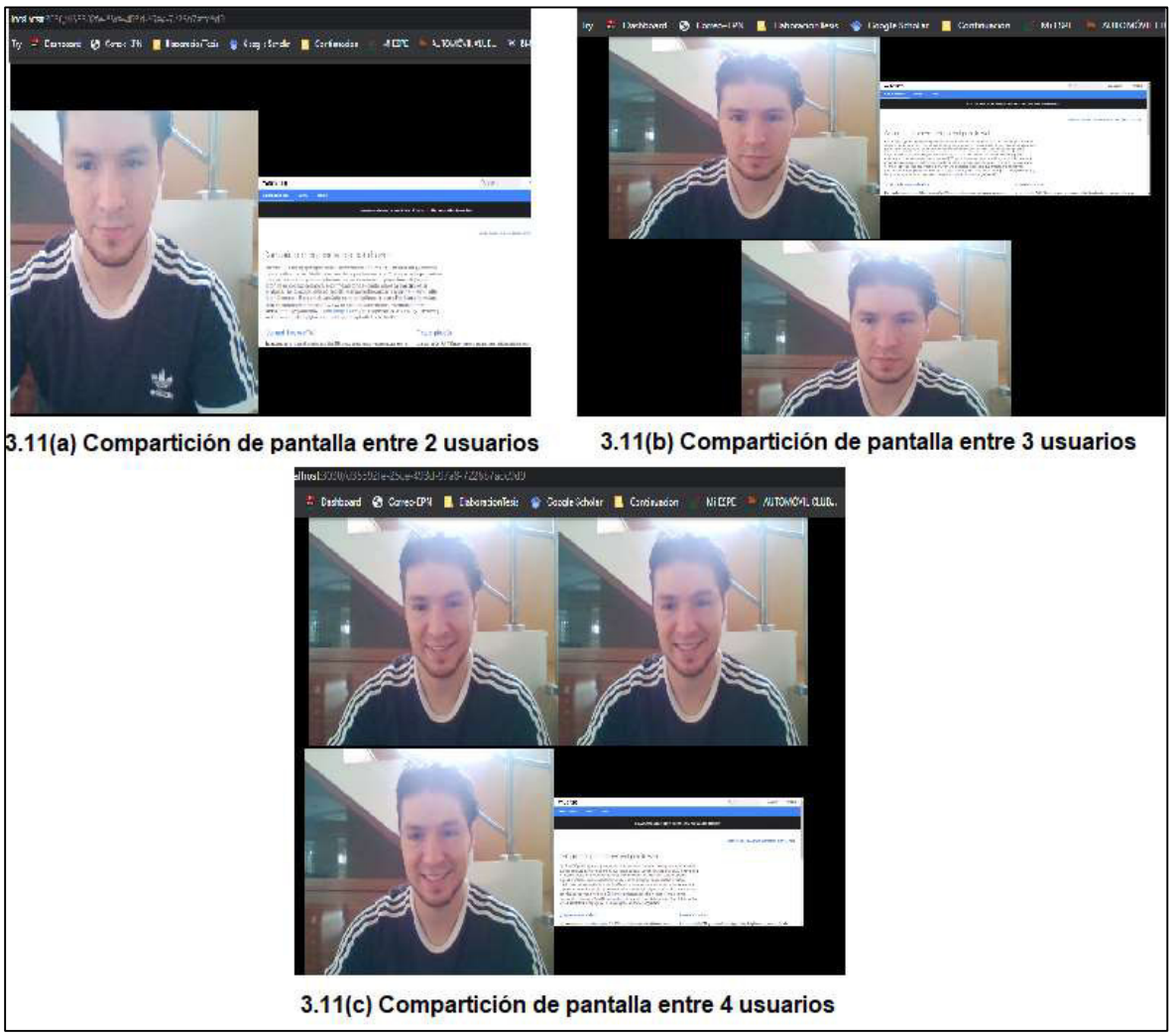


Figura. 3.11 Pruebas de compartición de pantalla en diversos escenarios

Las pruebas determinan que los usuarios pueden visualizar la compartición de pantalla correctamente. Se recalca que, en caso de una desconexión de un miembro, es necesario volver a compartir pantalla, de igual manera si un nuevo usuario ingresa. Los criterios de aceptación de la historia de usuario se cumplieron con éxito, siendo el resultado el esperado.

3.1.3 RESULTADOS SPRINT 3

Estas pruebas de funcionamiento se basan en las historias de usuario del *Sprint* 3, las cuales se especifican en el *Product Backlog* (Tabla 2.11).

Las historias de usuario utilizadas se detallan en la tabla 3.3, con el respectivo identificador para su prueba de funcionamiento.

Tabla 3.3 Historia de usuario del S*print* 3

Identificativo HU	Identificativo PF
HU-05	PF-08
HU-07	PF-09

3.1.3.1 Prueba de funcionamiento para la presentación de la aplicación - Videoconferencia (PF-08)

Esta prueba de funcionamiento corresponde a la historia de usuario 05.

- **Prueba de conexión y presentación de videoconferencia**

La videoconferencia puede abarcar un máximo de 4 usuarios; para esta prueba se realizan conexiones de manera local, en la que todos los usuarios se encuentran en la misma LAN, esto con la finalidad de facilitar la conexión. Cabe recalcar que la conexión necesita ser segura por lo cual utiliza certificados SSL.

La figura 3.12 muestra la prueba realizada, en la que los usuarios se conectaron de manera correcta y la videoconferencia se produjo con normalidad.

El resultado de las pruebas es el esperado en la historia de usuario 05, cumpliéndose los criterios de aceptación.

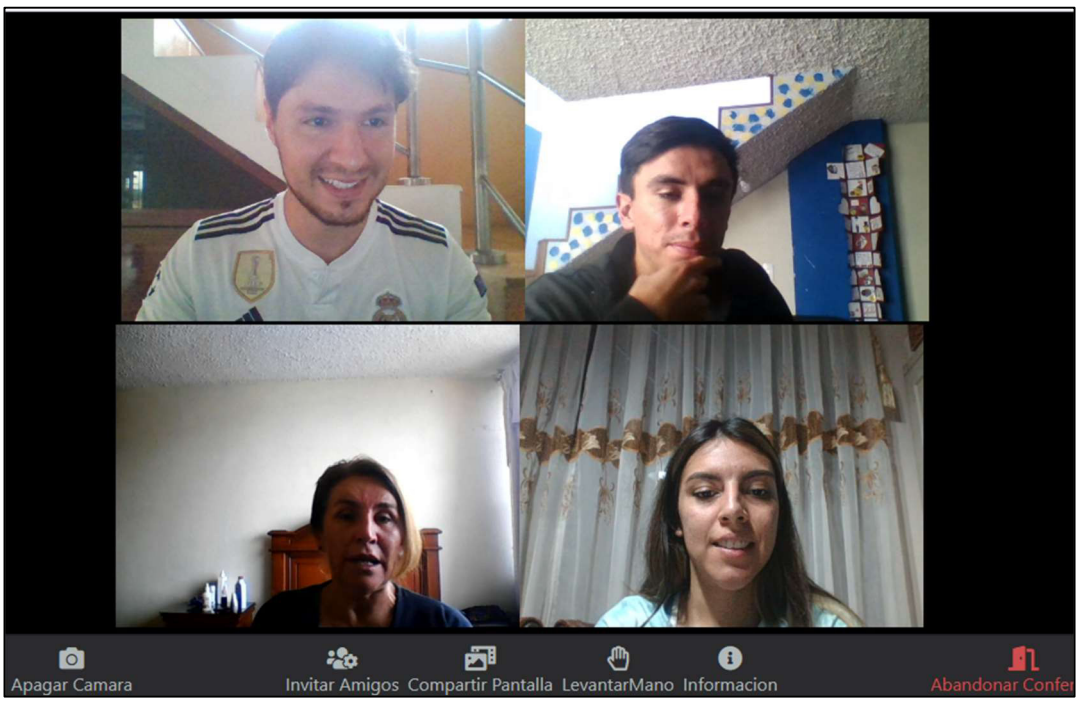


Figura. 3.12 Prueba de videoconferencia

3.1.3.2 Pruebas de funcionamiento Mensajería de la aplicación - Transferencia de Mensajes (PF-09)

Los usuarios pueden enviar y recibir mensajes de texto a través de la aplicación, los cuales se visualizan en la parte derecha de la pantalla; el mensaje está formado por: nombre de la persona que lo envía, hora del dispositivo del usuario y contenido del mensaje.

La figura 3.13 muestra la sección de mensajería, donde se observan 3 secciones: la sección (1) es el panel para el título de la zona, la sección (2) indica los mensajes enviados y recibidos en el *chat*, y, la sección (3) que es la zona de escritura y envío de mensajes, aquí se escribe el contenido del mensaje de texto a ser enviado a los demás participantes.

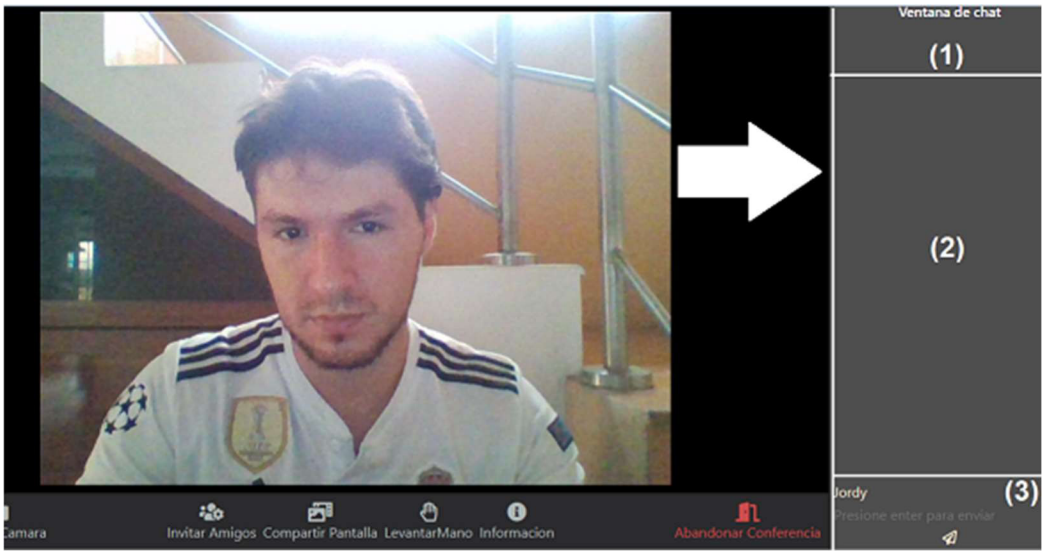


Figura. 3.13 Sección de mensajería y transferencia de mensajes

La figura 3.14, indica los resultados de la prueba de transferencia realizada entre 4 participantes, donde cada uno de ellos enviaron y recibieron los mensajes de texto.

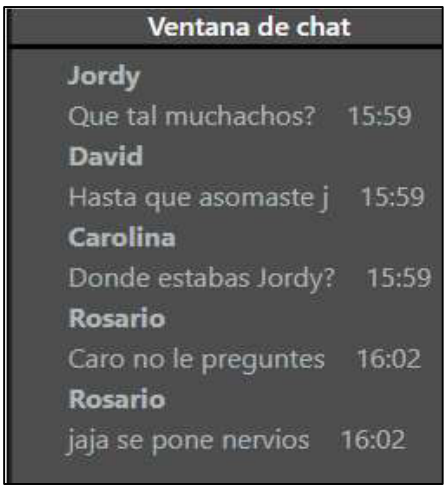


Figura. 3.14 Prueba de transferencia de mensajes

Para esta prueba se evidencia que los resultados son los esperados, verificándose el criterio de aceptación de la historia de usuario 07 en base a sus resultados.

3.1.4 RESULTADOS SPRINT 4

Las pruebas de funcionamiento se basan en las historias de usuario del *Sprint* 4, las cuales se especifican en el *Product Backlog* (Tabla 2.11). Las historias de usuario utilizadas se presentan en la tabla 3.4, con el respectivo identificador para su prueba de funcionamiento.

Tabla 3.4 Historia de usuario del *sprint* 4

Identificativo HU	Identificativo PF
HU-08	PF-10

3.1.4.1 Pruebas de funcionamiento de los controles de la aplicación (PF-10)

Las pruebas de funcionamiento corresponden a verificar el correcto uso de los controles de la aplicación, historia de usuario 8.

- **Prueba de control para el ingreso a la sala**

El usuario debe ingresar su nombre, o algún identificativo para acceder a la videoconferencia. La prueba valida este control al momento de ingresar a la sala; la figura 3.15 indica la solicitud ejecutada por el la aplicación para el ingreso del nombre de usuario, y el mensaje de error que se obtiene en caso de no ingresar la información solicitada.

Figura. 3.15 Prueba de control para el ingreso a la sala

Si el usuario no ingresa algún identificativo, no podrá acceder a la sala.

La figura 3.16 expone el acceso correcto del usuario a la sala, una vez ingresado su identificativo.

Figura. 3.16 Acceso correcto del usuario a la sala

Se observa que la prueba tiene éxito ya que el control se ejecuta de la manera esperada.

- **Prueba de control de audio**

El usuario tiene la facultad de activar o no su micrófono; en caso de activarlo, los demás miembros de la sala podrán escucharlo, en tanto que, si el micrófono se encuentra desactivado, el usuario no podrá ser escuchado.

La figura 3.17, expone la prueba de control de audio para el micrófono desactivado; se puede apreciar que la computadora no está reproduciendo ningún sonido, puesto que el micrófono no recepta el audio del usuario.

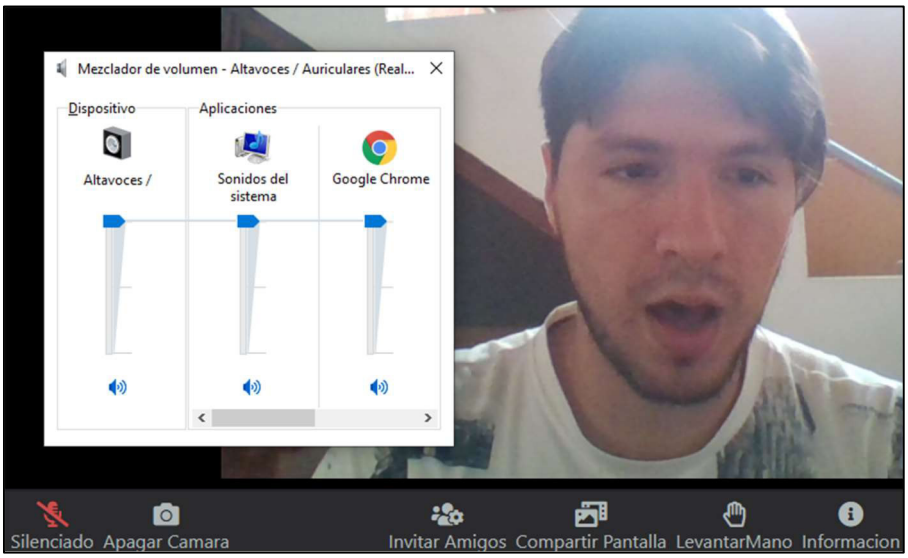


Figura. 3.17 Prueba de micrófono desactivado

En la figura 3.18 se observa la prueba de control de audio para el micrófono activado, pudiéndose apreciar que la computadora está reproduciendo la voz del usuario.

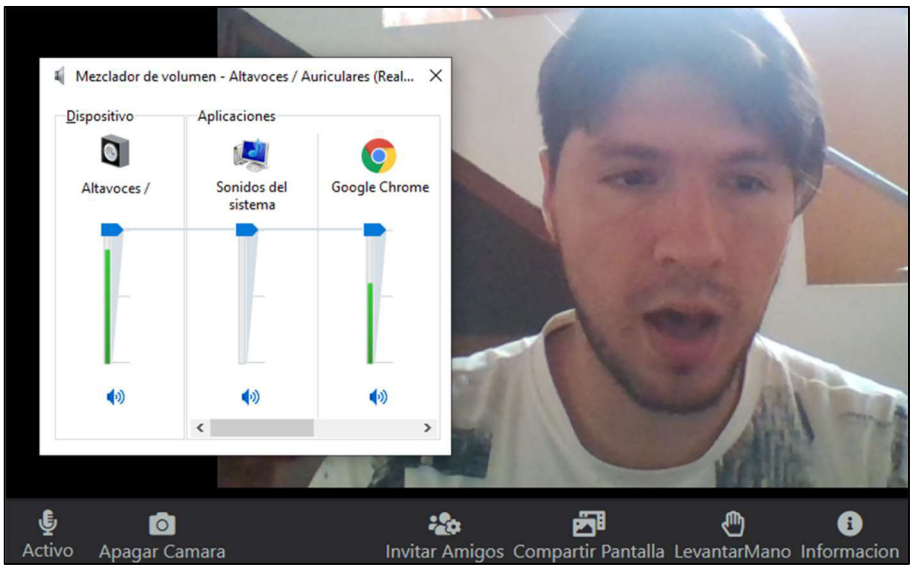


Figura. 3.18 Prueba de micrófono activado

El resultado de estas pruebas es el esperado, ya que el control funciona correctamente.

- **Prueba de control de video**

Al igual que en el caso anterior el usuario tiene la potestad de elegir la activación o desactivación del video. La figura 3.19 muestra la prueba para el control de activación del video.

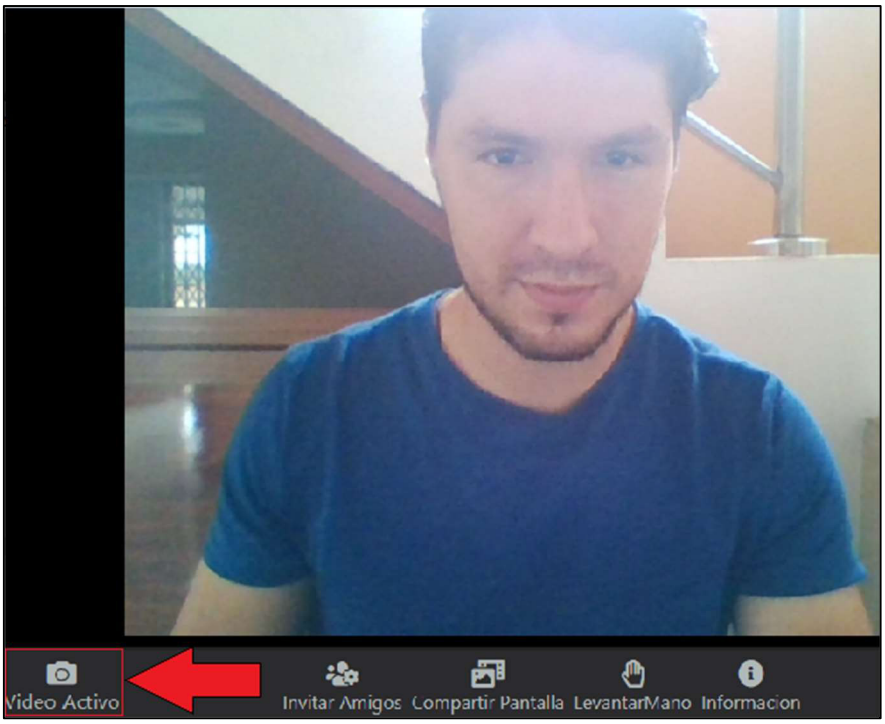


Figura. 3.19 Prueba de control de video – Activo

En la figura 3.20 se observa la prueba de control que desactiva el video.

Figura. 3.20 Prueba de control de video – Desactivado

Las pruebas evidencian que el resultado es el esperado y que el control funciona correctamente.

- **Prueba de control para ‘Invitar amigos’**

Este control permite al usuario ‘invitar más amigos’ mediante la copia del enlace URL de la reunión al portapapeles del usuario, para luego ser enviado a través del medio que el usuario desee: WhatsApp, Facebook, correo electrónico, entre otros. La figura 3.21 detalla el proceso del control y su uso correspondiente.

Figura. 3.21 Prueba de control para ‘Invitar Amigos’

- **Prueba de control para la compartición de pantalla**

El usuario puede compartir su pantalla, para lo cual dispone de un botón que permite seleccionar el contenido a compartir: ventanas, pestañas del navegador o pantalla completa.

La prueba de compartición de pantalla se la realiza en 3 etapas. La figura 3.22 presenta la acción del botón y las opciones que se despliegan para realizar la compartición; la figura 3.23 indica la selección de uno de los contenidos a ser compartidos; y, finalmente en la figura 3.24 se indica la detención de la compartición por parte del usuario.

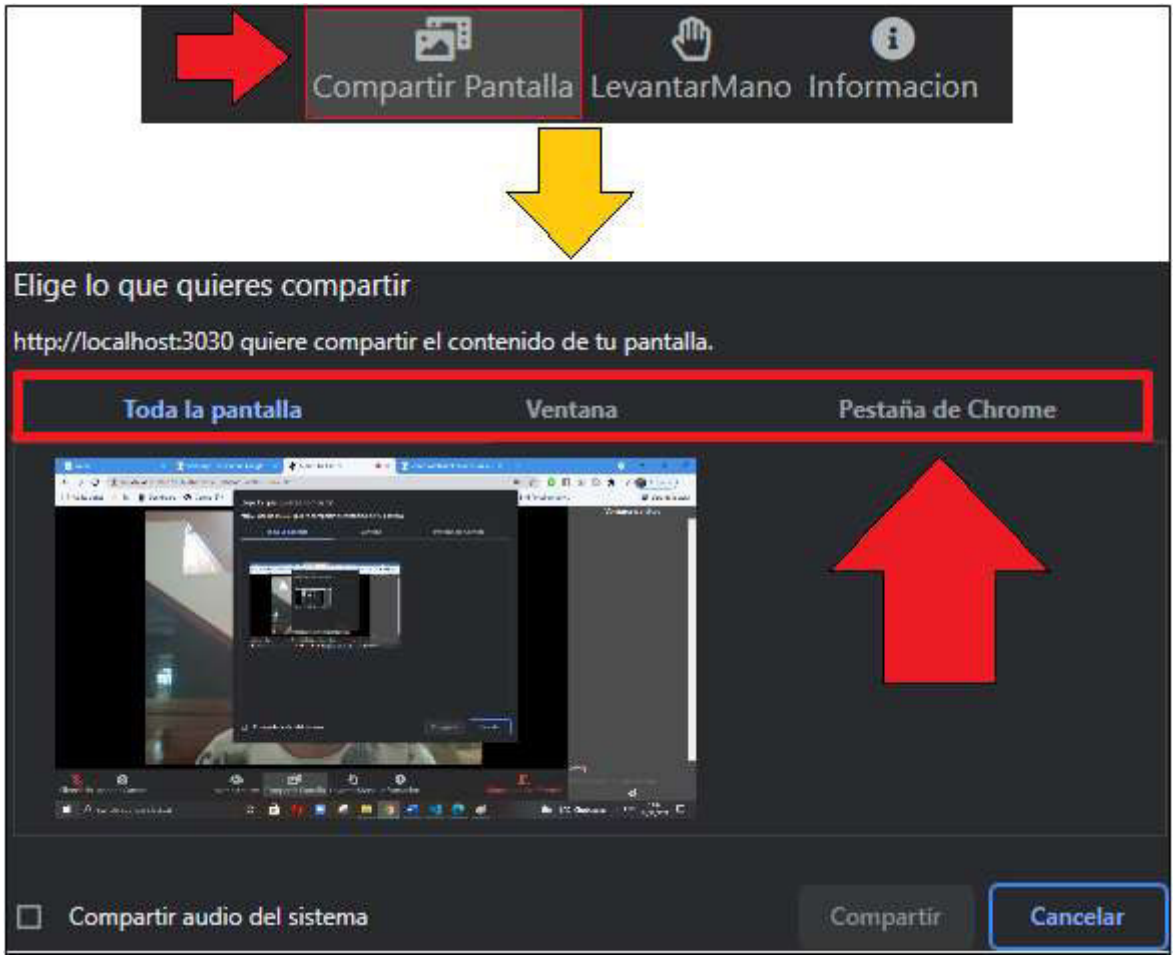


Figura. 3.22 Prueba de control de opciones en la compartición de pantalla

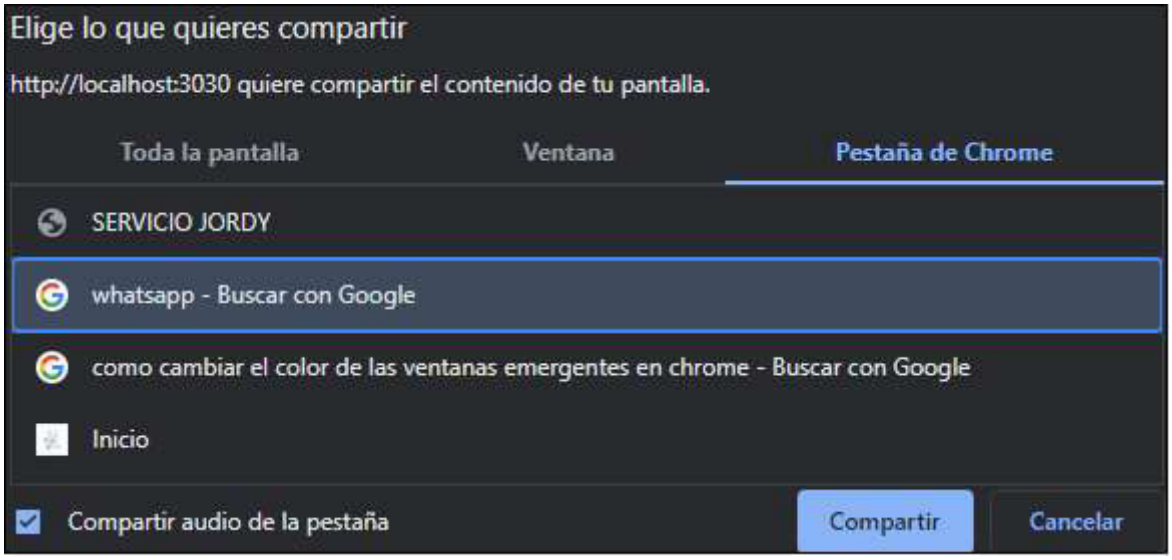


Figura. 3.23 Prueba de control de compartición de pantalla

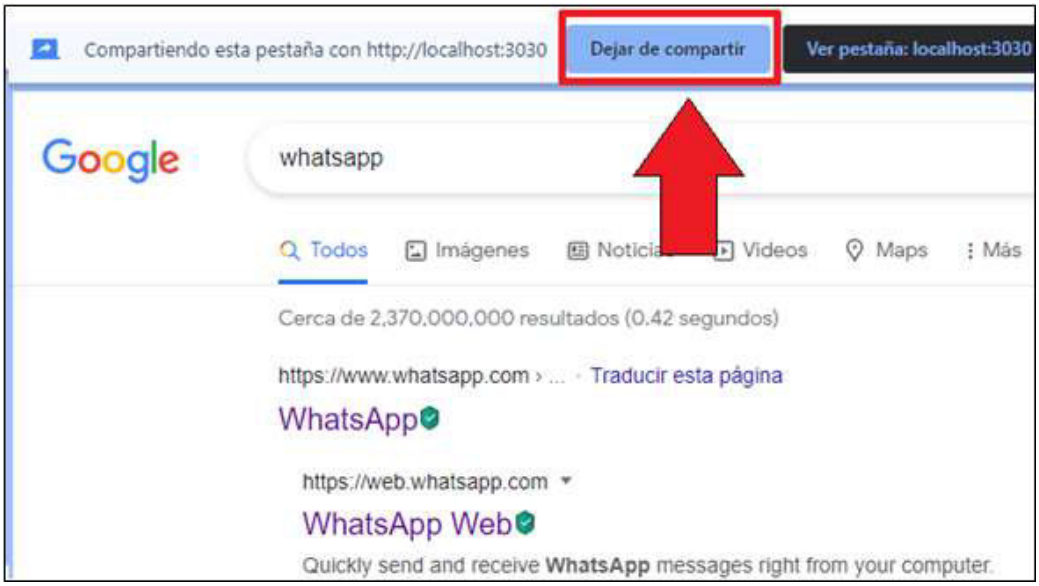


Figura. 3.24 Prueba de control de la desactivación de compartición de pantalla

La prueba evidencia el correcto funcionamiento del control para la compartición de pantalla.

- **Prueba de control para salir de la conferencia**

El usuario al momento de retirarse de la conferencia debe hacer uso del botón correspondiente, con lo que la aplicación cierra sus conexiones *peer* y realiza la salida del *socket* de comunicación. La figura 3.25 muestra el proceso de salida que realiza el usuario al momento de abandonar la sala.

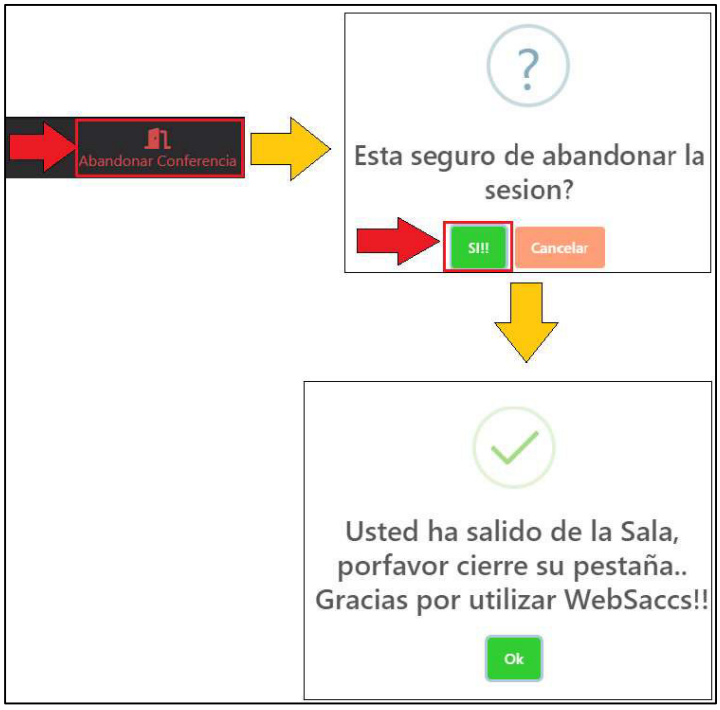


Figura. 3.25 Prueba de control para salida de conferencia

Cabe recalcar, cuando un usuario se desconecta o abandona la sala, todos los participantes restantes de la reunión, pueden visualizar un mensaje de notificación en la ventana de *chat* acerca de la salida del usuario que se retira (ver figura 3.26).

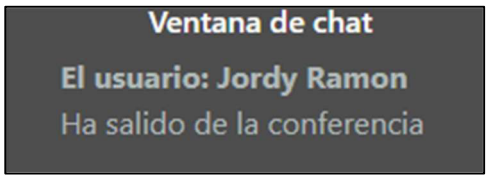


Figura. 3.26 Notificación del servidor por la salida de un usuario

Se verifica que los resultados de las pruebas de los controles han sido satisfactorios, ya que se cumplen los criterios de aceptación de la historia de usuario correspondiente.

3.1.5 RESULTADOS SPRINT 5

Sus pruebas de funcionamiento se basan en las historias de usuario del *Sprint* 5, las cuales son especificadas en el *Product Backlog* (Tabla 2.11).

Las historias de usuario utilizadas se detallan en la tabla 3.5, con el respectivo identificador para su prueba de funcionamiento.

Tabla 3.5 Historia de usuario del *sprint* 5

Identificativo HU	Identificativo PF
HU-09	PF-11

3.1.5.1 Prueba de funcionamiento del servidor STUN/TURN

La prueba realizada, fue el uso de la aplicación en un ambiente *cloud*, permitiendo a usuarios de distintas locaciones, conectarse a una sala de videoconferencia creada. Para la prueba, se conectaron 3 personas adicionales, las cuales se encontraban geográficamente en las ciudades de: Quito, Ambato y Santo Domingo de los Colorados.

Para hacer uso del servidor STUN/TURN, éste, debe estar encendido, lo cual se presenta en la figura 3.27.

Figura. 3.27 Servidor STUN/TURN activo

WebRTC permite verificar el funcionamiento de un servidor STUN/TURN mediante una herramienta creada por ellos denominada “WebRTC samples Trickle ICE”.

La figura 3.28 indica la comprobación del servidor STUN mediante el uso de la herramienta.

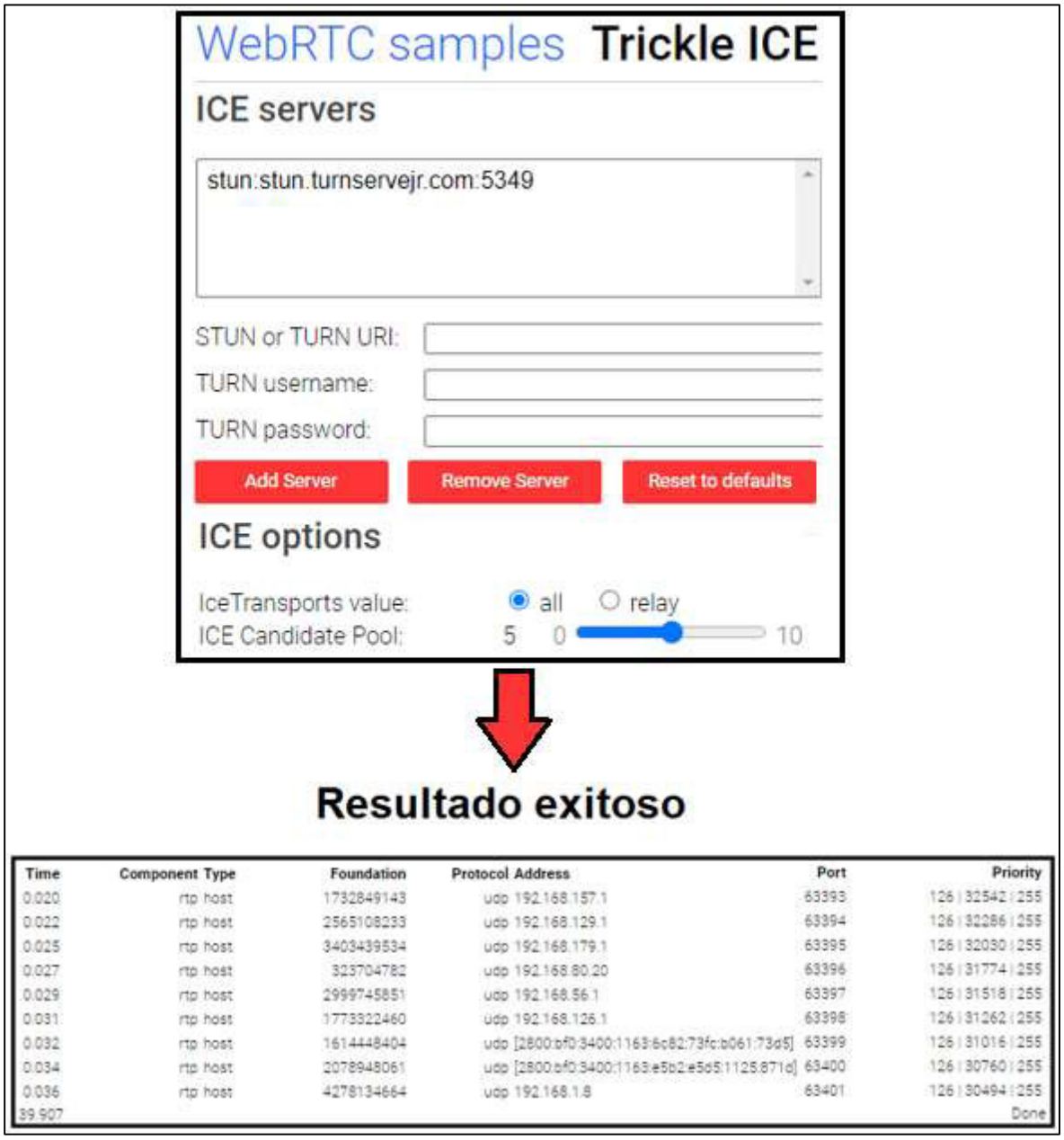


Figura. 3.28 Comprobación del servicio STUN en el servidor

Para verificar al servicio TURN es necesario introducir las credenciales de acceso al servidor.

La figura 3.29 indica la comprobación del servicio TURN, mediante el uso de la herramienta.

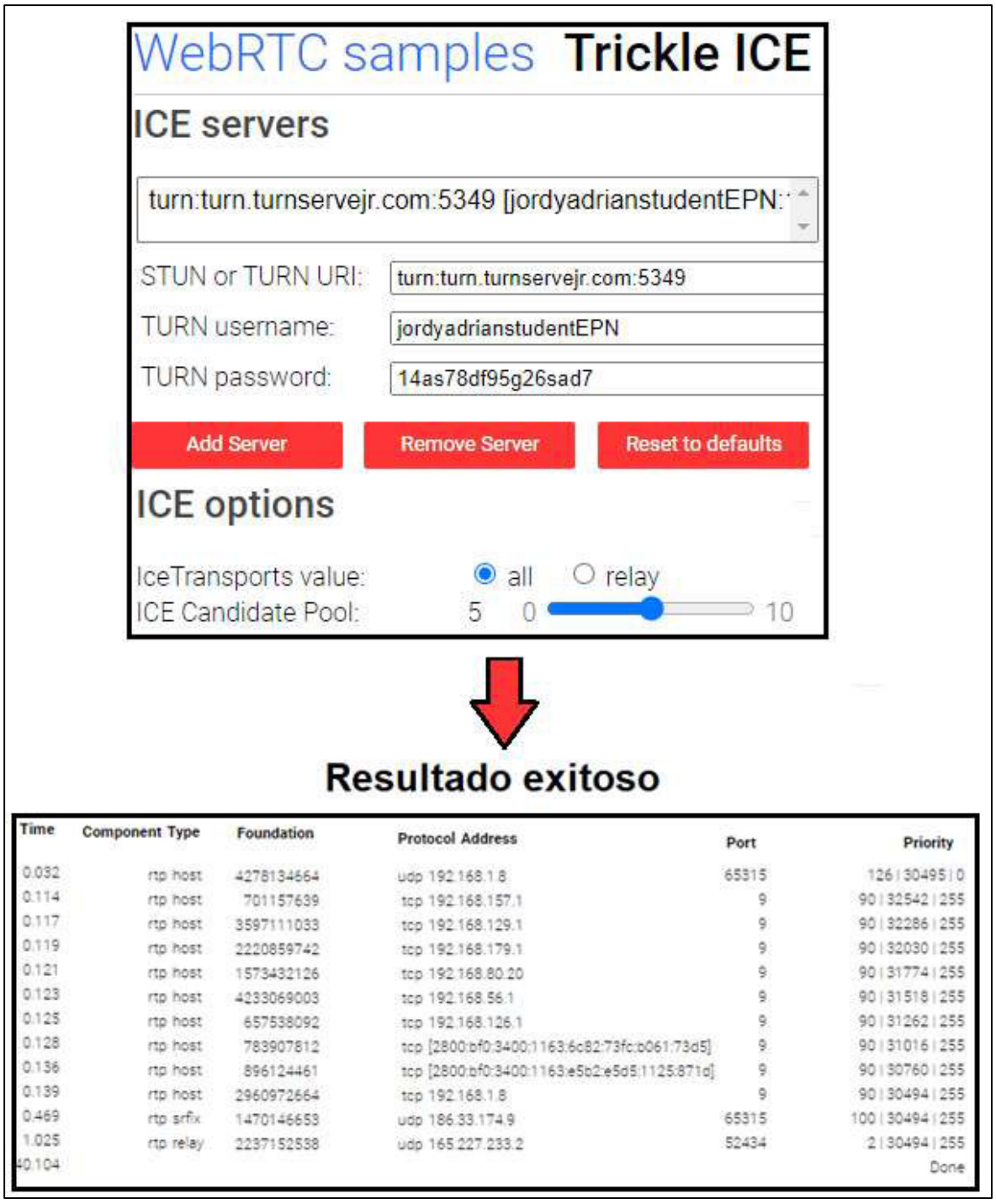


Figura. 3.29 Comprobación del servicio TURN en el servidor

Cuando el servidor se encuentra activado el usuario puede crear las reuniones de manera efectiva, asegurando la conexión entre clientes. La figura 3.30 muestra la creación de una sala en Internet.

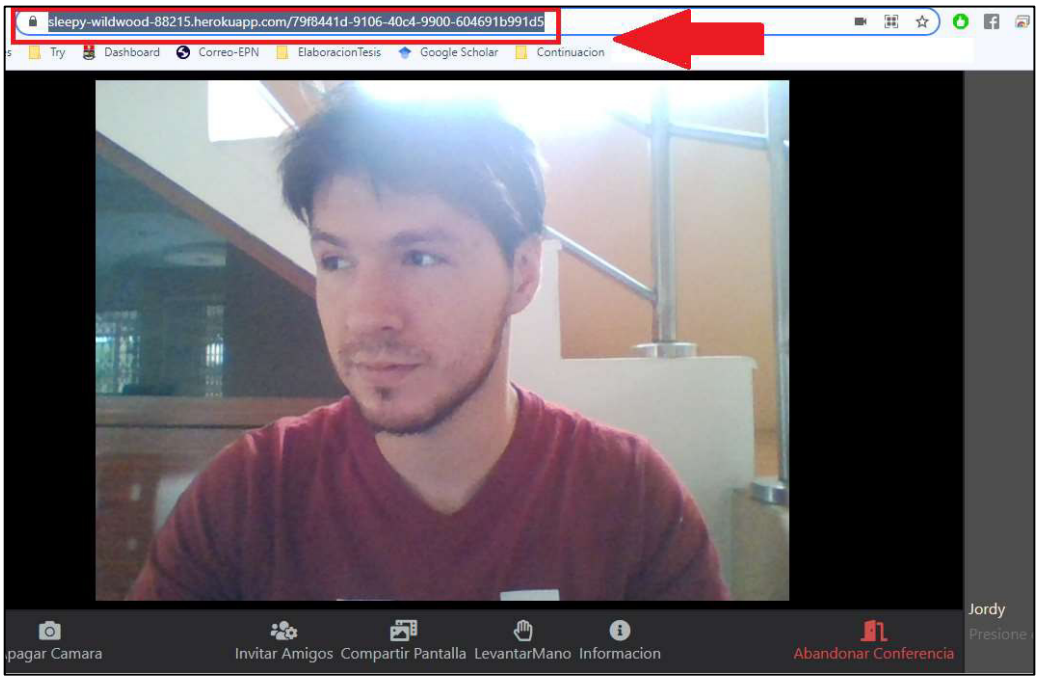


Figura. 3.30 Creación de sala en Internet

Con la sala creada, solo es necesario compartir el enlace; para hacerlo, se utiliza el control "invitar amigos" para copiar el *link* de la reunión y enviarlo a las demás personas. La figura 3.31 muestra la conexión de 4 usuarios que se encuentran en diferentes LANs.

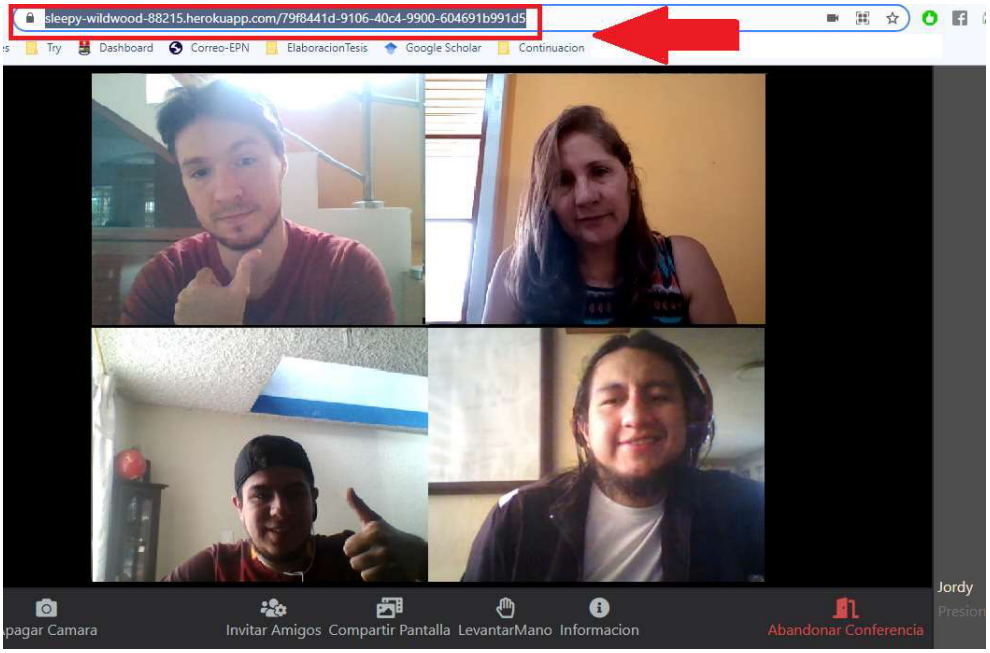


Figura. 3.31 Prueba de la aplicación haciendo uso del servidor STUN/TURN

Estas pruebas permiten evidenciar que el servidor y sus servicios STUN/TURN funcionan correctamente, ya que se cumplen los criterios de aceptación de la historia de usuario correspondiente.

3.1.6 PRUEBAS ADICIONALES

Las pruebas adicionales están fuera de las pruebas de aceptación, estas pruebas se realizaron para poner conocer como el aplicativo opera bajos condiciones no controladas, los resultados obtenidos son solo datos informativos.

- **Prueba de consumo de ancho de banda**

La aplicación consume un ancho de banda no determinado; sin embargo, según WebRTC cada conexión *peer-to-peer* generada, ocupa un ancho de banda (computacional) teórico de 200 Kbps. Por esta razón se desea obtener el consumo de ancho de banda para diferente número de usuarios conectados a la conferencia.

Para esta pruebas se utiliza la herramienta “AppNetworkCounter”, la cual es de licencia gratuita y permite medir distintos parámetros de las aplicaciones en el sistema, tales como cantidad de bytes enviados y recibidos, cantidad de paquetes enviados y recibidos, cantidad de bytes IPv4 enviados / recibidos, entre otros [50].

La figura 3.32 muestra que el consumo de ancho de banda (AB) del aplicativo, dentro del navegador Chome es de 0.24 Mbps, el cual se lo obtiene ejecutando la aplicación “websaccs”, con una sola conexión *peer* (2 usuarios).

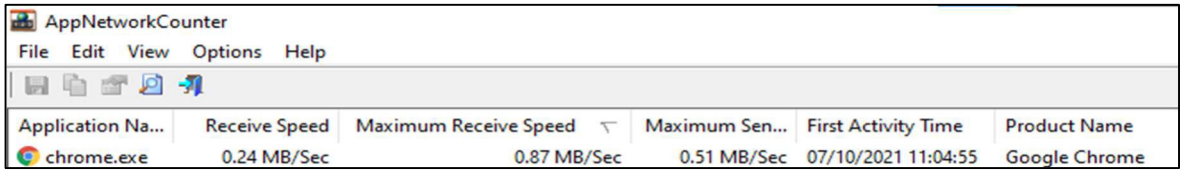


Figura. 3.32 Consumo de AB con una conexión *peer*

La figura 3.33 muestra que el consumo de AB del navegador Chome es de 0.43 Mbps, cuando se ejecuta la aplicación “websaccs”, con dos conexiones *peer* (3 usuarios).

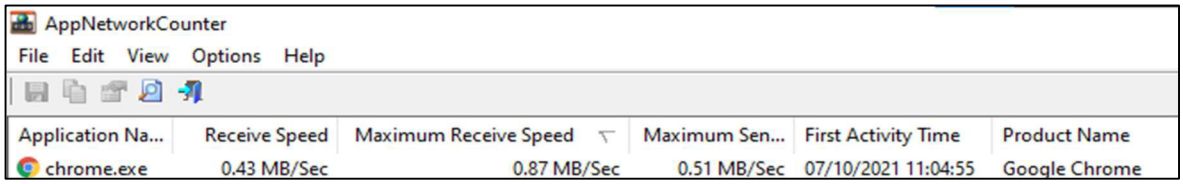


Figura. 3.33 Consumo de AB con dos conexiones *peer*

La figura 3.34 muestra que el consumo de AB del navegador Chome es de 0.73 Mbps, cuando solo se ejecuta la aplicación “websaccs” con tres conexiones *peer* (4 usuarios).

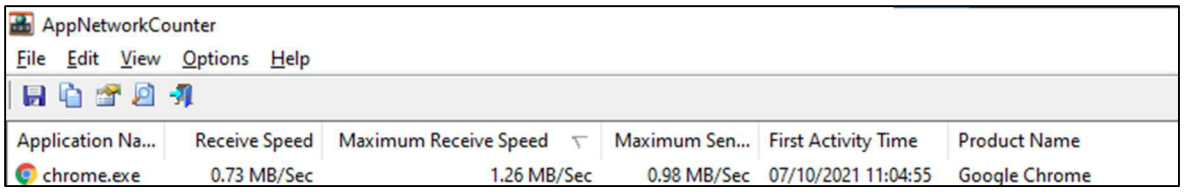


Figura. 3.34 Consumo de AB con tres conexiones *peer*

La aplicación fue diseñada para 4 usuarios simultáneos; por motivos de prueba, se añadió un quinto usuario para visualizar el consumo del AB.

La figura 3.35 muestra que el consumo de AB del navegador Chome es de 1.03 Mbps, cuando se ejecuta la aplicación “websaccs” con 4 conexiones *peer* (5 usuarios).

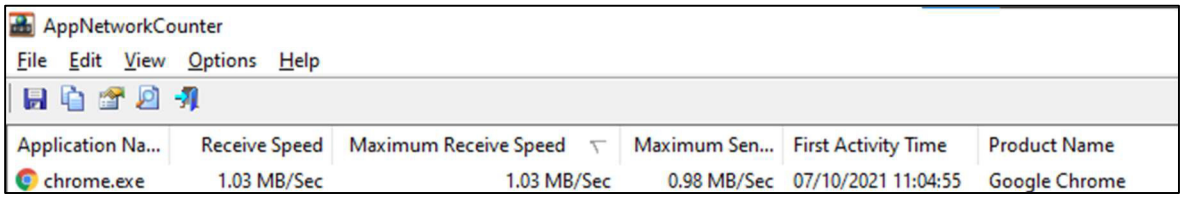


Figura. 3.35 Consumo de AB con cuatro conexiones peer

Se recalca que los resultados de consumo de AB pueden variar de acuerdo con el numero de conexión, o a las funciones que el usuario este utilizando dentro de la aplicación, tal como compartir pantalla con contenido dinámico o estático.

- **Prueba de número de usuarios y dispositivos**

Esta prueba permite la inclusión de más usuarios a la reunión, sobrepasando el valor máximo de 4 usuarios por reunión; los usuarios añadidos se conectan por celular y computadora. La figura 3.36 indica una sala de videoconferencia con 5 usuarios.

La conexión mediante celular no está especificada para este trabajo de titulación, sin embargo, fue considerada para esta prueba.

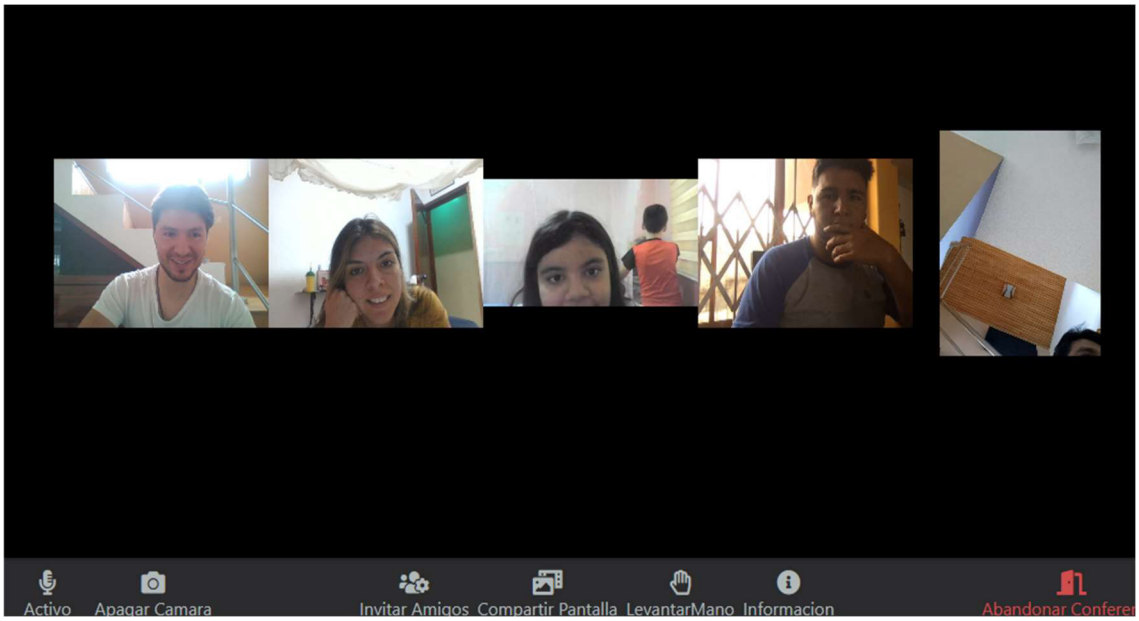


Figura. 3.36 Videoconferencia con 5 usuarios simultáneos

Los resultados muestran que si bien es cierto se permite visualizar los participantes de la videoconferencia, la pantalla asignada a cada participante se ve afectada (reducida) por la inclusión de este quinto usuario.

- **Prueba de tiempo de respuesta de la aplicación**

Esta prueba permite verificar el tiempo de respuesta de la aplicación al momento de ejecutar las funcionalidades de creación de sala y acceso de usuarios, para el ingreso de nuevos usuarios a la videoconferencia.

Para la prueba de creación de sala, se abrieron 4 salas distintas desde la página de inicio de la aplicación, obteniendo así los valores de tiempo de respuesta.

Se hizo uso del reloj interno de la computadora, se realizaron varios intentos de creación y se obtuvieron los tiempos respectivos; con los resultados se pudo obtener el rango de tiempo buscado.

La figura 3.37 indica que el tiempo de respuesta para la creación de una nueva sala, el cual oscila en el rango de 10 a 12 segundos.

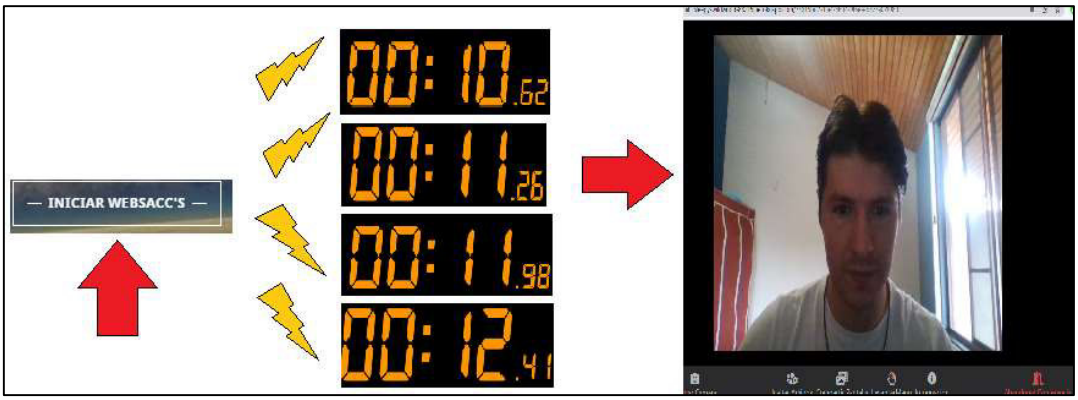


Figura. 3.37 Prueba de tiempo de respuesta para creación de sala

Para la prueba de acceso de usuario, se enviaron enlaces de reuniones ya creadas a varios participantes, para que éstos accedan a la sala.

Para esta prueba se hizo uso del reloj interno de la computadora, se realizaron varios intentos en los cuales se tomó el tiempo de respuesta para el acceso de un usuario a sala.

La figura 3.38 indica que el tiempo de respuesta, oscila en el rango de 4 a 5 segundos.

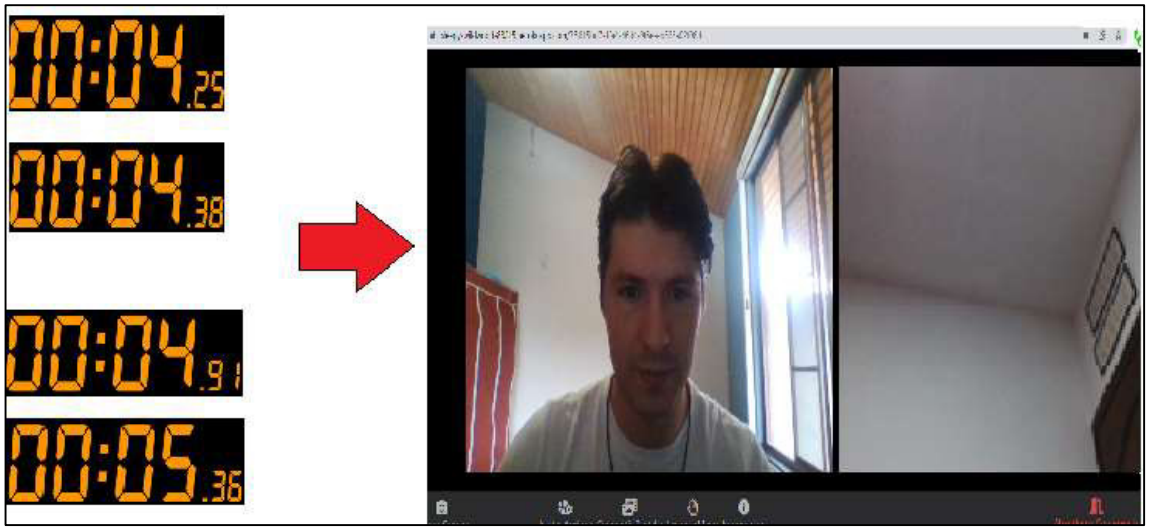


Figura. 3.38 Prueba de tiempo de respuesta para acceso a la sala

3.1.7 ANÁLISIS DE RESULTADOS

Los resultados obtenidos en las pruebas realizadas fueron positivos, el prototipo funciona de la manera esperada y se cumplen todos los criterios de aceptación establecidos en las historias de usuario. Los requerimientos de esta alternativa de sistema de videoconferencia se efectuaron como se lo esperaba.

La tabla 3.6 muestra el cumplimiento de las historias de usuarios y el tiempo que abarcó realizar cada una de ellas.

Tabla 3.6 Cumplimiento de las historias de usuario

Identificativo HU	Título HU	Criterio cumplido	Tiempo [Horas]
HU-00	Instalación de IDE de desarrollo y configuración del proyecto	Si	20
HU-02	Conexión y configuración del servicio de hosting (GoDaddy)	Si	10
HU-03	Conexión y configuración del servidor web (Heroku)	Si	5
HU-01	Página principal de la aplicación-Acerca del sistema	Si	10
HU-04	Presentación de la aplicación -Sección Multimedia	Si	20
HU-06	Presentación de la aplicación - Videoconferencia	Si	20
HU-05	Presentación de la aplicación -Compartición de pantalla	Si	20
HU-07	Mensajería de la aplicación -Transferencia de Mensajes	Si	15
HU-08	Controles de la aplicación	Si	20
HU-09	Servidor STUN/TURN	Si	20
		TOTAL	**160 horas**

La alternativa desarrollada permite realizar videoconferencias, compartición de pantalla, haciendo uso de las API's de WebRTC. Su uso es factible para redes externas, permitiendo así, conectar usuarios de diversas locaciones geográficas a través de la Internet.

Se realizaron correcciones después de la etapa de pruebas, enfocadas a mejorar la experiencia de usuario, añadiendo detalles en los diversos componentes de la aplicación.

En el Módulo de Presentación, se mejoraron detalles en el aspecto de visualización, los contenedores de los videos adoptan un mejor tamaño en su presentación.

En el Módulo de Controles, se añadieron nuevas características tales como: enviar enlace de la reunión por correo electrónico, y nuevas notificaciones para el usuario.

Al ser comunicaciones *peer-to-peer* la rapidez en la transferencia de datos es mayor, lo cual otorga calidad a las videoconferencias realizadas con WebRTC.

3.1.7.1 Situaciones encontradas en la aplicación

Los tiempos de respuesta para crear por primera vez una reunión en la aplicación, están en un rango de 10 a 12 segundos, esto dado que el servidor tiene que enviar los *scripts* que el usuario va a ejecutar en su computadora. Por otro lado, el ingreso de los usuarios a la reunión es de 4 a 5 segundos, ya que la aplicación necesita solicitar permisos al usuario para acceder a su audio y video.

La resolución y calidad de la imagen de los participantes en la videoconferencia, está dada por las resoluciones que permite su *hardware* (monitor-cámara), así como de la calidad de conexión a internet.

El de audio en la videoconferencia se escucha con claridad. Esto se logra ya que WebRTC dispone en su motor de voz de: AEC (cancelación de eco), ANS (reducción automática de ruido), y AGC (control automático de ganancia).

El ancho de banda que cada usuario ocupe en la aplicación será determinado por la cantidad de usuarios dentro de la sala y de las funciones que se hagan uso dentro de la aplicación; a mayor cantidad de usuarios, mayores conexiones *peer-to-peer,* mayor ancho de banda consumido o viceversa. De igual manera, si un usuario está compartiendo contenido, el ancho de banda se verá afectado si el contenido es, estático o dinámico. En WebRTC, el comportamiento para el AB se puede considerar lineal, ya que por pruebas se apreció que cada usuario ocupa alrededor de 0.5 Mbps de consumo.

3.1.7.2 Problemas encontrados durante las pruebas

Los siguientes son problemas encontrados durante las diversas pruebas realizadas a la aplicación, dichos problemas fueron corregidos en la versión final del prototipo.

- Al momento de acceder por primera vez a una reunión, el usuario puede presentar fallas en su solicitud de audio - video, esto se debe a que la conexión *peer-to-peer* no se ha realizado con éxito, por lo cual el usuario deberá refrescar su página para acceder nuevamente a la reunión.
- En la desconexión de un usuario de la reunión, la imagen de éste puede permanecer en la videoconferencia por unos segundos. La razón es que el cierre de la conexión *peer-to-peer* puede tomar un tiempo; sin embargo, una vez que se haya concluido con éxito la salida, la imagen desaparecerá. Se recalca que este inconveniente es aleatorio.

- Si un usuario coloca a otro participante en “pantalla completa”, los controles del video del participante se activan para el primer usuario, permitiendo así al usuario pausar o silenciar al otro participante, en su sesión; esto se debe a que contenedor de video que aloja a los usuarios, son propios de HTML; sin embargo, se solucionó este inconveniente estableciendo configuraciones especificas en los contenedores.
- Las invitaciones por *email* tienen un limitado número de uso, 200 correos mensuales, y al momento de llegar al correo del invitado, deben ser buscados en la carpeta de *spam* o “no deseados”.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- La información teórica levantada en el capítulo 1 fue fundamental para conocer los principales aspectos de un sistema de videoconferencia, así como su uso y características, además de las herramientas utilizadas para la implementación posterior del prototipo.

- El diseño realizado en el presente plan de titulación permitió tener una idea organizada de los diferentes pasos a seguir al momento de realizar la implementación del prototipo.

- El diseño del sistema de videoconferencia fue realizado de manera modular, donde cada módulo posee funciones únicas, las cuales en conjunto permiten tener una aplicación operativamente funcional. Para esta labor se utilizó la metodología ágil Scrum la cual permite organizar mediante *Sprints* las tareas a realizar.

- La implementación del sistema de videoconferencia diseñado fue realizada mediante el uso de las herramientas definidas previamente en el capítulo 1, las cuales facilitaron al desarrollo del trabajo de titulación aportando entre otros con diagramas, interfaces, y así lograr tener una aplicación funcionalmente operativa.

- En el capítulo de resultados se realizaron diversas pruebas para evaluar la funcionalidad del diseño y su implementación, logrando así, tener una retroalimentación general de la aplicación y con esto obtener ideas precisas para mejorar la aplicación en caso de ser necesario.

- Los resultados obtenidos de la prueba de usuarios simultáneos en la aplicación, permitió concluir que la aplicación funciona adecuadamente hasta un máximo de 4 usuarios, sin embargo, la aplicación tiene una flexibilidad de hasta 6 usuarios, existiendo la posibilidad que presente inconvenientes.

- Uno de los factores críticos al momento de utilizar la interfaz gráfica de la aplicación, es la adaptabilidad de la imagen a la pantalla del usuario, ya que cada dispositivo posee su propia característica en tamaño de pantalla; haciendo uso de *Bootstrap* se ayudó a mejorar la presentación visual, solucionando la criticidad antes mencionada.

- Los resultados en el tiempo de respuesta de la aplicación para crear una nueva sala o conectar un usuario, evidencian valores menores a 15 segundos para su correcta

ejecución, esto debido a que el usuario tenía el mínimo de ancho de banda recomendado de 3 Mbps, lo cual cumple exitosamente con el diseño realizado.

- Para la aplicación de la tecnología WebRTC se utilizó la biblioteca PeerJS, permitiendo desarrollar la aplicación web mediante una API simplificada para conexiones punto a punto, esto ayudó a reducir en gran cantidad las líneas de códigos implementadas y otorgó mayor libertad de la codificación rígida que contiene WebRTC en general.

- Finalmente, y gracias a las pruebas realizadas en el capítulo 4, se puede concluir que los objetivos planteados en el presente plan de titulación se han cumplido a cabalidad, teniendo así un prototipo viable como una posible solución alternativa, la cual es funcional para un sistema web de videoconferencia y mensajería.

4.2 RECOMENDACIONES

- Es recomendable poseer los conceptos claros que serán utilizados para el diseño de la aplicación; con esto se tendrá total conocimiento en la toma de decisiones para el desarrollo de la aplicación.

- Puesto que la aplicación fue diseñada para un máximo de 4 usuarios, se recomienda para trabajos futuros acerca de este tema, obtener un servidor STUN/TURN de mayor capacidad, lo cual permitirá tener mayor capacidad de usuarios en las sesiones.

- Utilizar los navegadores web con sus versiones actualizadas, en Chrome (56 o superior), Mozilla Firefox (44 o superior) y Opera (43 o superior); caso contrario la aplicación puede no funcionar como se lo espera.

- Emplear el navegador de Google Chrome para el uso de la aplicación, puesto que Google es el motor primario de WebRTC, proveyendo soporte en la API de JavaScript y una integración total de sus componentes, antes que los demás navegadores (Mozilla Firefox-Opera), así mismo en las pruebas realizadas Chrome, presento un mejor desenvolvimiento (Funciones y métodos) el momento de ejecutar la aplicación.

- Un inconveniente encontrado en el desarrollo del prototipo es la elaboración de la interfaz gráfica con HTML5 y CSS, puesto que éstos tienen configuraciones preestablecidas difíciles de modificar. Se recomienda, practicar a profundidad estos lenguajes de programación.

- En trabajos futuros acerca de este tema, es recomendable poseer conocimientos avanzados de lenguaje JavaScript (Multimedia-Conexión con sockets), para así personalizar las API(Cámara-Audio) existentes en WebRTC, permitiendo ejecutar funciones más específicas y elaboradas.
- El presente proyecto no estipula el funcionamiento de la aplicación en dispositivos celulares, por lo cual se recomienda para futuros trabajos acerca de este tema, usar el *framework "*React Native", el cual permite desarrollar aplicaciones nativas para Android o IOS con el lenguaje JavaScript.
- WebRTC promete ser una tecnología innovadora con grandes virtudes; por lo cual es de suma importancia continuar las investigaciones de esta índole, puesto que a futuro permitirá mejorar la experiencia como usuario de sistemas de videoconferencia.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] E. G. Domenech, “LA VIDEOCONFERENCIA: UN RETO Y UNA NECESIDAD ACTUAL,” p. 7, Mar. 2018.

[2] A. Medina Chacon, “La videoconferencia: conceptualización, elementos y uso educativo,” vol. 1, no. 1, pp. 2–6, 2003. Accessed: Jun. 29, 2021. [Online]. Available: https://dialnet.unirioja.es/descarga/articulo/6871638.pdf

[3] M. N. Sadiku, S. M. Musa, and O. D. Momoh, “Cloud computing: opportunities and challenges,” *IEEE Potentials*, vol. 33, no. 1, pp. 34–36, 2014.

[4] “¿Qué son los contenedores? ¿Cómo funcionan con Docker?” https://www.hpe.com/es/es/what-is/containers.html (accessed Mar. 16, 2022).

[5] WebRTC, “WebRTC,” *webrtc.org*, 2021. https://webrtc.org/ (accessed Jun. 29, 2021).

[6] Gartner, “Gartner Reprint,” 2020. https://www.gartner.com/doc/reprints?id=1-2468658S&ct=200917&st=sb (accessed Jun. 29, 2021).

[7] W3Counter, “W3Counter: Global Web Stats - May 2021,” 2021. https://www.w3counter.com/globalstats.php?year=2021&month=5 (accessed Jun. 29, 2021).

[8] “SpiderMonkey: The Mozilla JavaScript runtime - Mozilla | MDN,” Jul. 21, 2020. https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey (accessed Jun. 29, 2021).

[9] C. Opera, “Navegador de Opera para ordenadores | Tu compañero perfecto en la red | Opera,” 2021. https://www.opera.com/es/computer/opera (accessed Jun. 29, 2021).

[10] Programación Web, “Arquitectura de las aplicaciones Web,” 2018. https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/ (accessed Jun. 29, 2021).

[11] E. Marini, “El modelo cliente/servidor,” *Obtenido Httpswww Linuxito Comdocsel-Modelo-Cliente-Serv. Pdf*, 2012.

[12] S. Smith, “Arquitecturas de aplicaciones web comunes|Microsoft Docs,” Dec. 01, 2020. https://docs.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures (accessed Jun. 28, 2021).

[13] W. Google, “What is WebRTC?,” *FAQ*. https://webrtc.googlesource.com/src/+/refs/heads/main/docs/faq.md (accessed Jul. 01, 2021).

[14] H. Alvestrand, “Publicación de Google del código fuente de WebRTC,” May 31, 2011. https://lists.w3.org/Archives/Public/public-webrtc/2011May/0022.html (accessed Jul. 01, 2021).

[15] W3C-IETF, “Web Real-Time Communications (WebRTC) transforms the communications.Wide Web Consortium (W3C) Recommendation and multiple Internet Engineering Task Force (IETF) standards.” https://www.w3.org/2021/01/pressrelease-webrtc-rec.html.en (accessed Jul. 01, 2021).

[16] W. Elleuch, “Models for multimedia conference between browsers based on WebRTC,” *Wirel. Mob. Comput.*, vol. ., no. ., p. 279,284, Oct. 2013.

[17] WebRTC, “Arquitectura | WebRTC.” https://webrtc.github.io/webrtc-org/architecture/ (accessed Jul. 01, 2021).

[18] F. Weinrank, M. Becke, J. Flohr, E. Rathgeb, I. Rungeler, and M. Tuxen, “WebRTC Data Channels,” *IEEE Commun. Stand. Mag.*, vol. 1, no. 2, pp. 28–35, 2017, doi: 10.1109/MCOMSTD.2017.1700007.

[19] E. Rescorla, “Datagram Transport Layer Security Version 1.2.” Jan. 2012. Accessed: Jul. 05, 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6347.txt

[20] IBM, “IBM Docs,” Apr. 28, 2021. https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-

south.containers.appdomain.cloud/docs/es/aix/7.2?topic=protocol-stream-control-transmission (accessed Jul. 05, 2021).
[21] M. Baugher, “The Secure Real-time Transport Protocol (SRTP),” *Network Working Group*, Mazo 2014. https://www.rfc-editor.org/rfc/rfc3711.txt (accessed Jul. 05, 2021).
[22] MDN, “WebRTC protocols - Referencia de la API Web | MDN.” https://developer.mozilla.org/es/docs/Web/API/WebRTC_API/Protocols (accessed Jul. 05, 2021).
[23] S. Dutton, “WebRTC in the real world: STUN, TURN and signaling,” *Google Nov*, 2013.
[24] B. Jansen, “Performance analysis of WebRTC-based video conferencing,” 2016.
[25] IETF, “STUN-rfc5389.” https://datatracker.ietf.org/doc/html/rfc5389 (accessed Jul. 07, 2021).
[26] “WebRTC Signaling Servers – STUN vs TURN,” *WebRTC.ventures*, Dec. 28, 2020. https://webrtc.ventures/2020/12/webrtc-signaling-stun-vs-turn/ (accessed Jul. 07, 2021).
[27] “Servidor TURN,” *WebRTC*. https://webrtc.org/getting-started/turn-server?hl=es (accessed Jul. 07, 2021).
[28] WebRTC, “WebRTC project owned by Google.” https://webrtc.googlesource.com/src/+/refs/heads/main/docs/faq.md (accessed Jul. 05, 2021).
[29] WebRTC, “Interop Notes-WebRTC.” https://webrtc.github.io/webrtc-org/web-apis/interop/ (accessed Jul. 05, 2021).
[30] S. Dutton, “Get Started with WebRTC - HTML5 Rocks,” *HTML5 Rocks - A resource for open web HTML5 developers*. https://www.html5rocks.com/en/tutorials/webrtc/basics/ (accessed Jul. 05, 2021).
[31] W3C, “Media Capture and Streams.” https://w3c.github.io/mediacapture-main/#stream-api (accessed Jul. 05, 2021).
[32] MDN, “MediaStream - Web APIs | MDN.” https://developer.mozilla.org/en-US/docs/Web/API/MediaStream (accessed Jul. 05, 2021).
[33] WebRTC, “WebRTC Native APIs | WebRTC.” https://webrtc.github.io/webrtc-org/native-code/native-apis/ (accessed Jul. 05, 2021).
[34] MDN, “RTCPeerConnection - Web APIs | MDN.” https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection (accessed Jul. 05, 2021).
[35] J. E. Pérez, “Introduccion a JavaScript.” 2019.
[36] J. D. Gauchat, *El gran libro de HTML5, CSS3 y Javascript*, Primera Edicion. España: Marcombo, 2012.
[37] C. Aubry, *HTML5 y CSS3-Revolucione el diseño de sus sitios web*. Ediciones ENI, 2012.
[38] “Qué es SCRUM,” *Proyectos Ágiles*, Aug. 04, 2008. https://proyectosagiles.org/que-es-scrum/ (accessed Aug. 11, 2021).
[39] M. Trigas Gallegos, “Metodología Scrum.” Accessed: Jun. 12, 2021. [Online]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf
[40] “Scrum Methodology for Digital Product Development,” *4Geeks Blog*, Nov. 11, 2019. https://blog.4geeks.io/scrum-for-digital-product-development/ (accessed Aug. 11, 2021).
[41] M. O. contributors Jacob Thornton, and Bootstrap, “BootStrap Introduction.” https://getbootstrap.com/docs/5.0/getting-started/introduction/ (accessed Jul. 15, 2021).
[42] Node.js, “Oficial Page NodeJS,” *Node.js*. https://nodejs.org/es/about/ (accessed Jul. 15, 2021).

[43] MDN, “Introducción a Express/Node - Aprende sobre desarrollo web | MDN.” https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction (accessed Jul. 15, 2021).
[44] NodeOS, “Útiles herramientas para el entorno de programacion Node.js.” https://blog.aulaformativa.com/utiles-herramientas-para-el-entorno-de-programacion-node-js/ (accessed Jul. 15, 2021).
[45] Express, “Express 5.x - Referencia de API.” http://expressjs.com/en/5x/api.html (accessed Jul. 15, 2021).
[46] PeerJS, “PeerJS: simple peer-to-peer con WebRTC.” https://peerjs.com/ (accessed Jul. 15, 2021).
[47] D. Arrachequesne, “Socket.IO-Introduction,” *Socket.IO*, Jul. 16, 2021. https://socket.io/docs/v4/index.html (accessed Jul. 16, 2021).
[48] “‘websocket’ | Can I use... Support tables for HTML5, CSS3, etc.” https://caniuse.com/?search=websocket (accessed Jul. 16, 2021).
[49] D. J. Quispe Parí and G. Sánchez Mamani, “Encuestas y entrevistas en investigación científica,” *Rev. Actual. Clínica Investiga*, p. 490, 2011.
[50] “Monitor network usage / bandwidth of every application on Windows,” *NirSoft*. https://www.nirsoft.net/utils/app_network_counter.html (accessed Oct. 07, 2021).

ANEXOS

ANEXO A: Preguntas de la entrevista para obtención de datos.

ANEXO B: Manual de Uso de la aplicación.

ANEXO C: Scripts para el servidor y cliente.

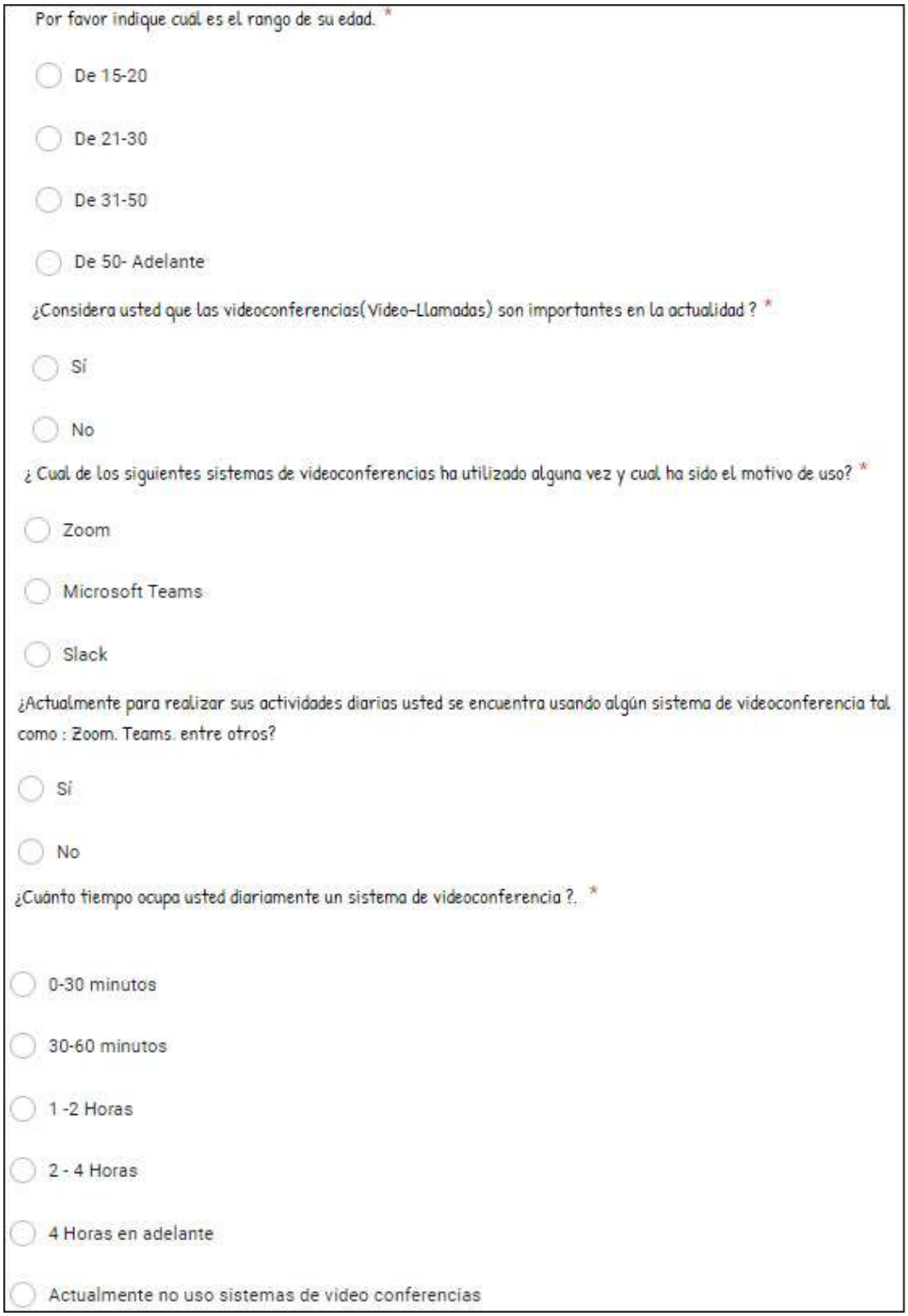
ANEXO D: Código fuente de la página web.

Nota: Dada su extensión, los anexos C y D se encuentran disponibles en el siguiente enlace de GoogleDrive:

https://docs.google.com/document/d/12CEJPcrbqT7j2iuDx2f0YiHVFSBZ6W2r/edit?usp=sharing&ouid=100104831262442156129&rtpof=true&sd=true

ANEXO A

PREGUNTAS DE LA ENTREVISTA PARA OBTENCIÓN DE DATOS



Por favor indique cuál es el rango de su edad. *

- De 15-20
- De 21-30
- De 31-50
- De 50- Adelante

¿Considera usted que las videoconferencias(Video-Llamadas) son importantes en la actualidad ? *

- Sí
- No

¿ Cual de los siguientes sistemas de videoconferencias ha utilizado alguna vez y cual ha sido el motivo de uso? *

- Zoom
- Microsoft Teams
- Slack

¿Actualmente para realizar sus actividades diarias usted se encuentra usando algún sistema de videoconferencia tal como : Zoom. Teams. entre otros?

- Sí
- No

¿Cuánto tiempo ocupa usted diariamente un sistema de videoconferencia ?. *

- 0-30 minutos
- 30-60 minutos
- 1 -2 Horas
- 2 - 4 Horas
- 4 Horas en adelante
- Actualmente no uso sistemas de video conferencias

Figura A. 1 Preguntas de la entrevista- Parte 1/2

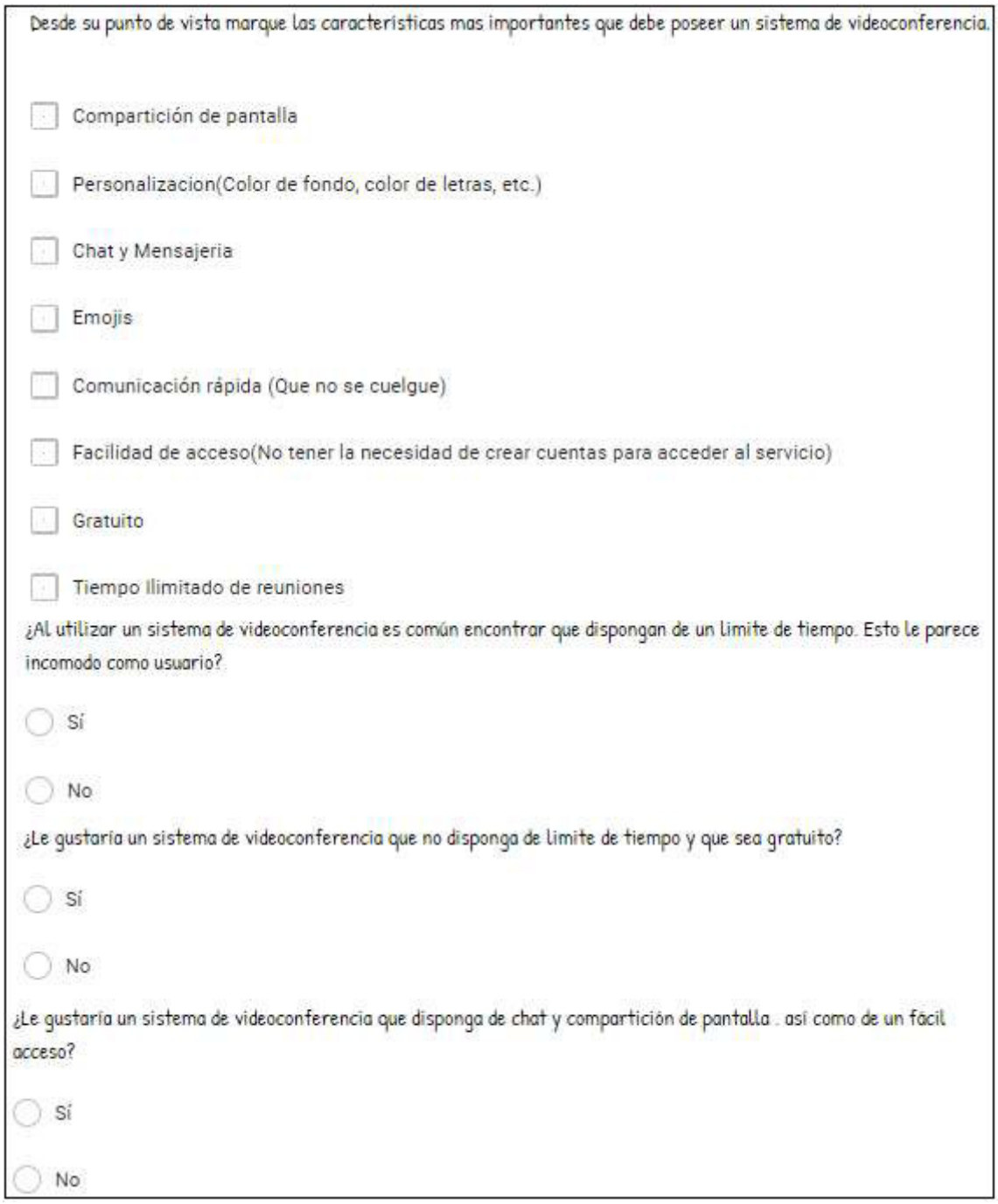


Figura A. 1 Preguntas de la entrevista- Parte 2/2

ANEXO B

MANUAL DE USO DE LA APLICACIÓN

El presente proyecto desarrollo un prototipo alternativo de videoconferencias con el uso de la tecnología WebRTC, proporcionando al usuario una opción para comunicación en tiempo real a través de Internet.

Se recomienda utilizar los navegadores Google Chrome, Mozilla Firefox y Opera en sus versiones más actuales.

El siguiente es el manual de usuario para el uso de la aplicación, dirigido a cualquier persona que desee hacer uso de este.

B.1 Presentación

- **Página de inicio**

Para acceder a la página principal de la aplicación es necesario ingresar al siguiente enlace:

https://websaccs.godaddysites.com/inicio

Una vez dentro el usuario observara la página principal, tal como se lo muestra a continuación en la figura B.1.

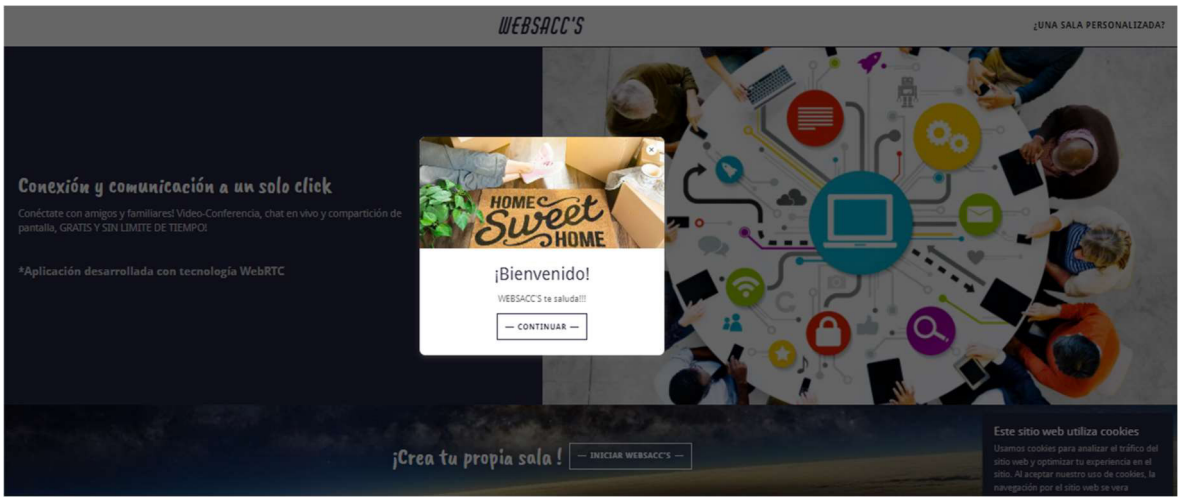


Figura B. 1 Página principal de la aplicación

- **Creación de una nueva reunión**

Una vez dentro de la página de inicio el usuarios podrá crear una nueva videoconferencia mediante un clic en la opción “Iniciar WebSacc’s”, lo cual se expone en la Figura B.2.

Figura B. 2 Opción para crear una videoconferencia

Al hacerlo se abrirá una nueva ventana, y en ella se podrá visualizar la sala recién creada, figura B.3.

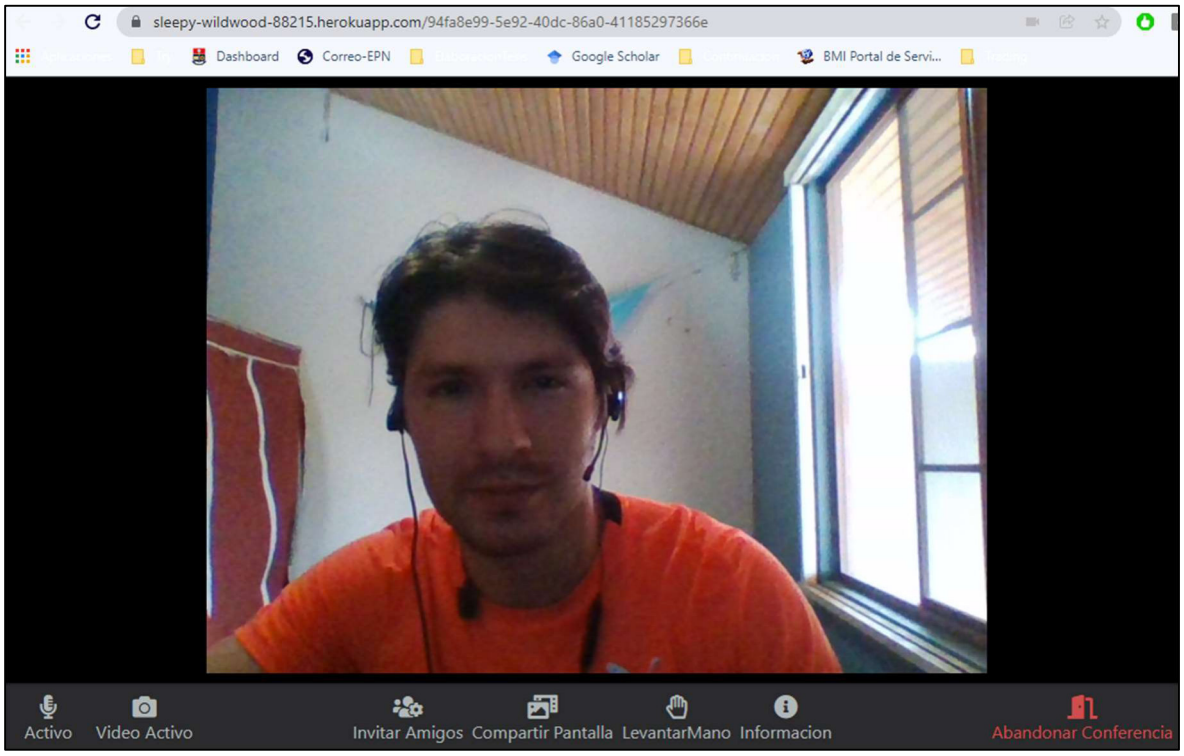


Figura B. 3 Sala creada exitosamente

Se recalca que el usuario debe ingresar un identificativo antes de ingresar a la sala, caso contrario no podrá acceder, de igual manera es muy importante otorgar los permisos de multimedia que la aplicación solicite, en caso de no hacerlo, el usuario no podrá visualizar a ningún participante de la reunión.

- **Creación de una sala personalizada**

Una vez creada una nueva sala, el usuario tiene la posibilidad de personalizar su link de ingreso, cambiando el parámetro especificado en la figura B.4.

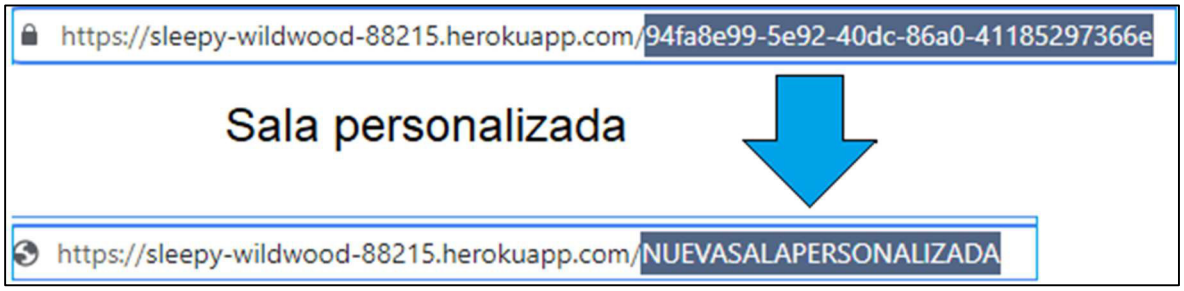


Figura B. 4 Parámetro modificar para una sala personalizada

Con los parámetros ya modificados, deberá refrescar la página, y se accederá a una nueva sala personalizada, esto se muestra en la figura B.5.

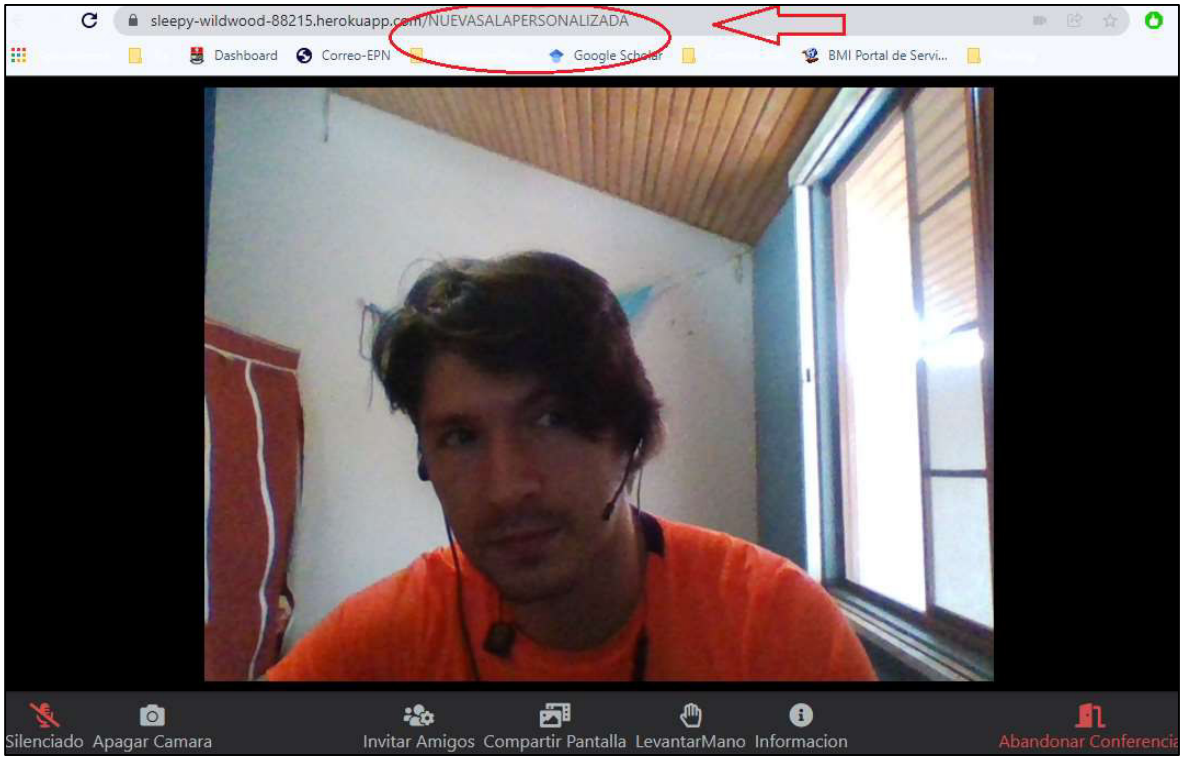


Figura B. 5 Sala personalizada creada exitosamente

- **Audio y video dentro de la aplicación**

Dentro de la sala el usuario tiene la posibilidad de activar-desactivar su video y audio, dando clic sobre los respectivos botones.

La figura B.6, indica como se visualizan los botones cuando el audio y video están activados.

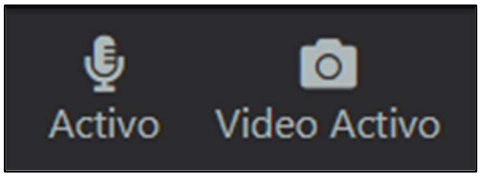


Figura B. 6 Botones activos de audio y video

La figura B.7, indica como se visualizan los botones cuando el audio y video están desactivados.

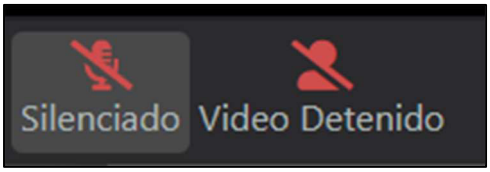


Figura B. 7 Botones desactivados de audio y video

- **Mensajería**

Dentro de una sala, el usuario puede enviar mensajes de texto a través de la ventana de chat. Figura B.8.

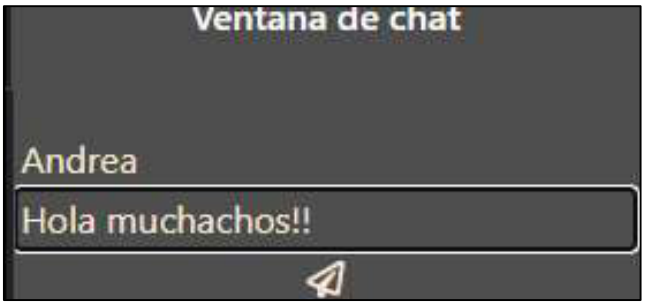


Figura B. 8 Ventana de chat

Él envió del mensaje se realiza mediante la tecla “Enter”, o presionando sobre la “palomita en el apartado de mensajería, con esto, el mensaje es enviado a todos los participantes de la reunión. Figura B.9.

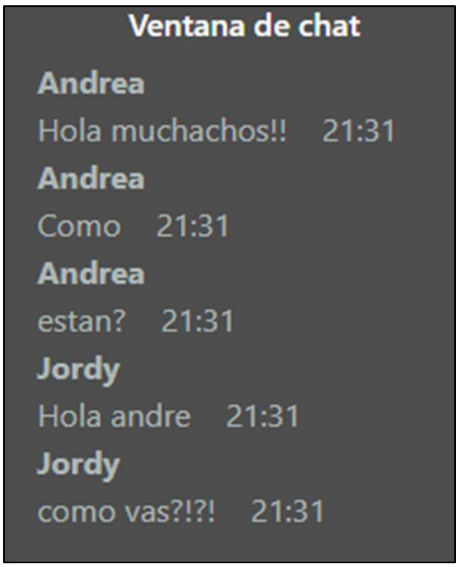


Figura B. 9 Envió de mensajes exitoso

- **Compartición de pantalla**

Para compartir pantalla se selecciona el icono correspondiente, y se mostrara las opciones para compartir pantalla (Compartir solo ventanas-Compartir toda la pantalla- Compartir solo una carpeta, etc.), visualizar figura B.10.

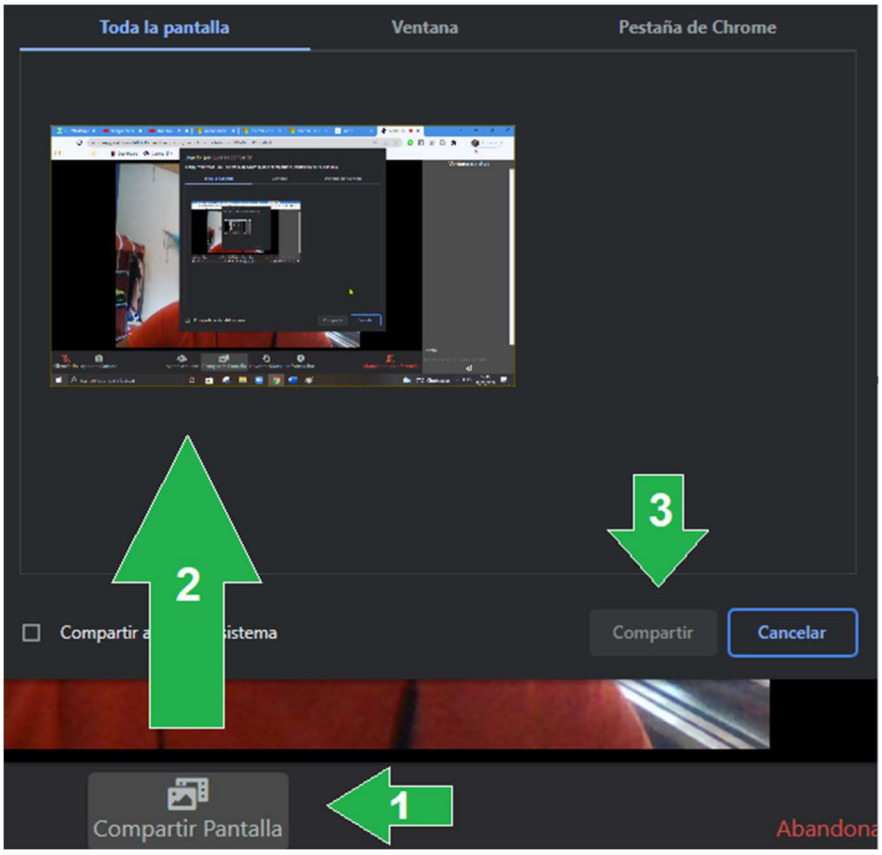


Figura B. 10 Proceso para compartición de pantalla

Una vez que el usuario haya seleccionado lo que desea compartir, esta se compartirá con los usuarios que estén dentro de la videoconferencia, visualizar figura B.11.

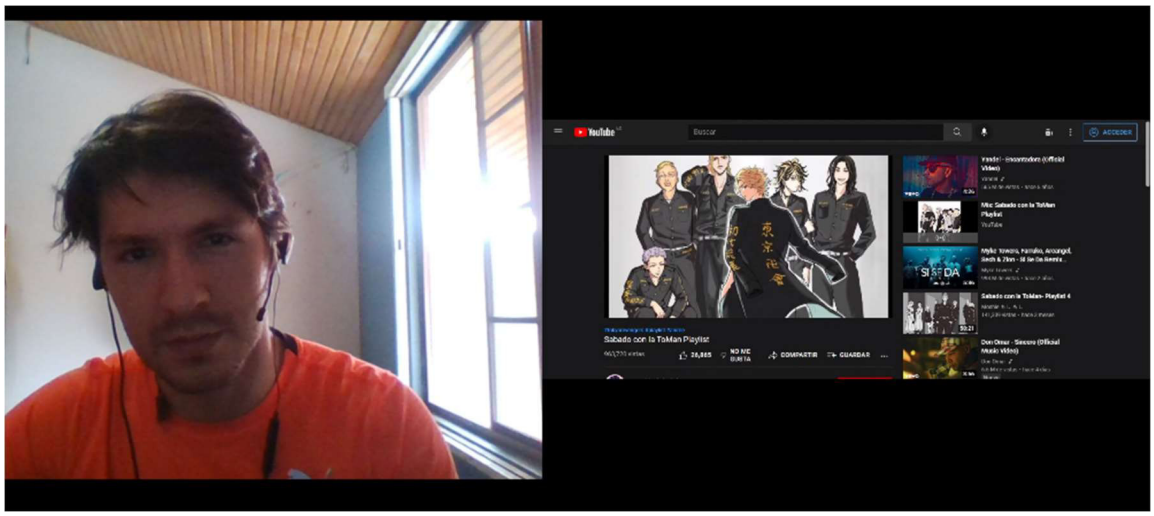


Figura B. 11 Visualización exitosa de la compartición de pantalla

- **Levantar la mano**

Como usuario puede solicitar la palabra para hablar, seleccionando el botón “Levantar la mano”, se enviara un emoji de “pedir la palabra” al chat de los participantes, de esta manera quien está hablando entiende que hay una persona que desea hablar. Figura B.12.

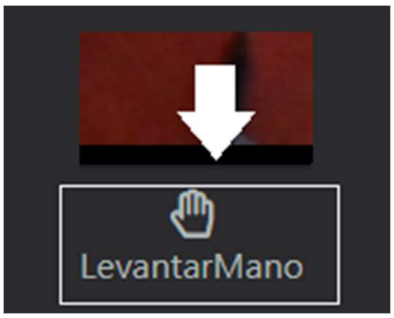


Figura B. 12 Botón para "Levantar la mano"

- **Salida y desconexión**

Para salir de la aplicación el usuario debe seleccionar la opción “abandonar conferencia”, permitiendo una desconexión apropiada de la aplicación. Figura B.13.

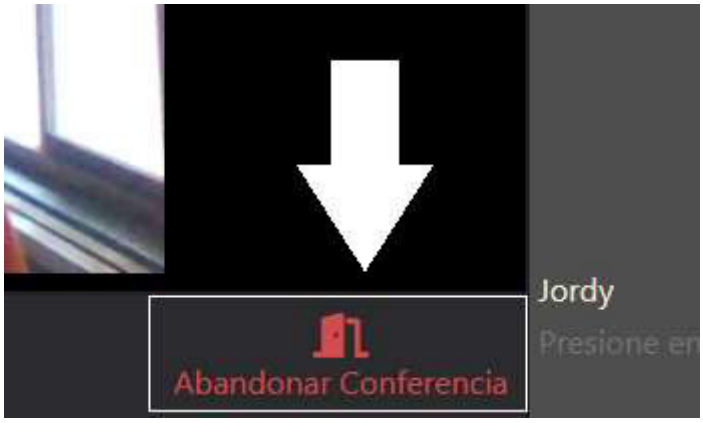


Figura B. 13 Botón para "Abandonar conferencia"

- **Recomendaciones**

En caso de la no visualización de un usuario en la videoconferencia, se deberá refrescar la página con el respectivo link de la reunión. (No olvidar otorgar permisos de multimedia a la aplicación).

Si no se visualiza la compartición de un usuario, pídale a dicho usuario que vuelva a compartir el contenido. Si esto se mantiene pídale al usuario que comparte que refresque su sala.

De preferencia utilizar el navegador Google Chrome puesto que este mantiene actualizada su tecnología WebRTC.

ANEXO C

SCRIPTS PARA EL SERVIDOR Y CLIENTE

Dada su extensión, el anexo C se encuentra disponible en el siguiente enlace de GoogleDrive:

https://docs.google.com/document/d/12CEJPcrbqT7j2iuDx2f0YiHVFSBZ6W2r/edit?usp=sharing&ouid=100104831262442156129&rtpof=true&sd=true

ANEXO D

CÓDIGO FUENTE DE LA PÁGINA WEB

Dada su extensión, el anexo D se encuentra disponible en el siguiente enlace de GoogleDrive:

https://docs.google.com/document/d/12CEJPcrbqT7j2iuDx2f0YiHVFSBZ6W2r/edit?usp=sharing&ouid=100104831262442156129&rtpof=true&sd=true

ORDEN DE EMPASTADO