ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA APLICACIÓN WEB ALTERNATIVA PARA VIDEOCONFERENCIA Y COMPARTICIÓN DE PANTALLA CON EL USO DE WEBRTC

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

MICHELLE ALEJANDRA PERALBO VELASCO

michelle.peralbo@epn.edu.ec

DIRECTOR: ING. ANDRÉS FERNANDO REYES CASTRO, MSc.

andres.reyes@epn.edu.ec

CODIRECTOR: ING. ANA MARÍA ZAMBRANO VIZUETE, PhD.

ana.zambrano@epn.edu.ec

Quito, marzo 2019

AVAL

Certificamos que el presente trabajo fue desarrollado por Michelle Alejandra Peralbo Velasco, bajo nuestra supervisión.

_____

Ing. Andrés Reyes, MSc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

_____

Ing. Ana Zambrano, PhD.
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Michelle Alejandra Peralbo Velasco, 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 cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Michelle Alejandra Peralbo Velasco

DEDICATORIA

A mi madre Mercedes, mi fuente infinita de fortaleza, por ser mi pilar y no permitir que me derrumbe en los momentos difíciles. Por su amor y su paciencia.

A mi hermano Jeremy, por ser mi más grande amigo, por todos los momentos en los que me dio su apoyo, confió en mí y me alentó a hacer mi mejor esfuerzo.

A mi padre Freddy, por las enseñanzas y valores que en algún momento supo impartirme para hacer de mí una persona de bien.

AGRADECIMIENTO

“Tantas veces me mataron

tantas veces me morí,

sin embargo, estoy aquí resucitando.

Gracias doy a la desgracia y a la mano con puñal

porque me mató tan mal y seguí cantando”

María Elena Walsh

A cada uno de mis familiares, por sus consejos, su bondad y apoyo, que hicieron esta etapa de mi vida más llevadera.

A mis amigos, que siempre me brindaron su apoyo incondicional y que han sido parte importante de mi carrera universitaria y de mi vida en general.

A mis profesores, que supieron guiarme a lo largo de este camino y me enseñaron a reconocer en la dificultad, las oportunidades, de manera especial agradezco al Ing. Andrés Reyes por ser mi guía en este proceso y al Ing. Pablo Hidalgo por sus consejos profesionales y su aliento constante para lograr mis objetivos.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

En la actualidad las comunicaciones interactivas en tiempo real mediante videoconferencia se han tornado fundamentales para las empresas, el comercio electrónico y la educación. Por ello se ha desarrollado un prototipo de aplicación web que permite la comunicación mediante videoconferencia y que añade la característica de compartición de pantalla mediante el uso de la tecnología WebRTC, utilizando para su señalización Socket.IO. El presente documento consta de cuatro capítulos:

El primer capítulo entrega una introducción de las comunicaciones basadas en web, conceptos relevantes de videoconferencia y de la tecnología WebRTC, arquitectura, APIs y arquitecturas de videoconferencia soportadas. Posteriormente se describen las características relevantes a la implementación de este prototipo: Node.js, MySQL, HTML5, y Socket.IO para la señalización, además de características relevantes de la metodología Scrum.

En el segundo capítulo se presenta la metodología empleada para el desarrollo de este proyecto técnico, dividida en tres fases: diseño, codificación e implementación. La fase de diseño consta del establecimiento de requerimientos, módulos, roles y diseño de los componentes del sistema, la fase de codificación corresponde al desarrollo del sistema aplicando la metodología Scrum dividido en cuatro *sprints* y la fase de implementación se refiere a la implementación del prototipo en el ambiente Google Cloud.

En el tercer capítulo se presentan los resultados de las pruebas de funcionamiento aplicadas al sistema en cada uno de los *sprints* y posteriormente se realiza un análisis de los resultados obtenidos.

El cuarto capítulo contiene conclusiones y recomendaciones que se obtuvieron luego de la realización de este proyecto técnico.

PALABRAS CLAVE: WebRTC, videoconferencia, compartición de pantalla, JavaScript

ABSTRACT

Nowadays, real-time interactive communications through videoconferencing have become fundamental for companies, e-commerce and education. Therefore, a web application prototype has been developed that allows communication by videoconference and includes the screen sharing feature using WebRTC technology and Socket.IO signaling. This document has four chapters:

The first chapter introduces web-based communications, relevant concepts of videoconferencing and WebRTC technology, architecture, APIs and supported videoconferencing architectures. Subsequently, the characteristics relevant to the implementation of this prototype are described: Node.js, MySQL, HTML5, and Socket.IO for signaling, as well as the characteristics of Scrum methodology.

The second chapter presents the methodology used for the development of this technical project, divided into three phases: design, coding and implementation. The design phase consists of the establishment of requirements, modules, roles and design of the components of the system, the coding phase corresponds to the development of the system applying Scrum methodology divided into four sprints and the implementation phase refers to the prototype implementation in the Google Cloud environment.

In the third chapter, the results of the performance tests applied to the system in each of the sprints are presented and then an analysis of the results obtained is made.

The fourth chapter contains conclusions and recommendations that were obtained after the completion of this technical project.

KEYWORDS: WebRTC, video conferencing, screen sharing, JavaScript

1. INTRODUCCIÓN

En la actualidad la videoconferencia ha ganado atención de manera muy rápida y se ha convertido en una forma popular de comunicación; debido a la necesidad creciente de comunicaciones interactivas en tiempo real y su uso como herramienta colaborativa en las empresas ha logrado que esta se masifique rápidamente.

Debido a esta necesidad de comunicación, en la actualidad existen varias soluciones que brindan opciones avanzadas de videoconferencia, entre dos o más usuarios; entre las cuales tenemos aquellas orientadas a hardware, a software y orientadas a la web.

WebRTC o Web Real-Time Communications [1], surge como una tecnología de navegador, que permite realizar comunicaciones en tiempo real de manera fácil y accesible. WebRTC agrega funcionalidades adicionales al navegador web y permite que los navegadores realicen una videoconferencia *peer to peer* sin la necesidad de instalar complementos o aplicaciones adicionales [2].

1.1 Objetivos

El objetivo general de este estudio técnico es:

- Desarrollar una aplicación web alternativa para videoconferencia y compartición de pantalla mediante el uso de WebRTC.

Los objetivos específicos de este estudio técnico son:

- Describir de manera general las características más relevantes de la arquitectura WebRTC, señalización IP y metodología Scrum.
- Diseñar los módulos que contendrá la aplicación en base a los diagramas de clases y relacional.
- Desarrollar los módulos del sistema.
- Verificar el correcto funcionamiento de la aplicación desarrollada.

1.2 Alcance

En términos generales se propone el diseño e implementación de una aplicación web para videoconferencia, compartición de pantalla y transferencia de mensajes entre los participantes, mediante el uso exclusivo del navegador Google Chrome y el conjunto de APIs JavaScript que posee WebRTC.

Para el desarrollo de software se utiliza la metodología ágil Scrum, debido a que se desarrolla la aplicación de manera iterativa e incremental, y por la flexibilidad que brinda la metodología para la mejora continua del sistema.

La aplicación cuenta con dos perfiles de usuario: Administrador y Usuario, que tienen funciones compartidas y algunas otras diferentes dependiendo del rol.

Los lenguajes de programación a usar son HTML5 y JavaScript usando librerías como jQuery; además se usa el entorno de ejecución de JavaScript Node.js. Esta aplicación es desarrollada por módulos donde se distinguen los siguientes: Autenticación, Administración de Usuarios, Módulo de Comunicación y Módulo de Ayuda que documenta la implementación de un sistema de videoconferencia utilizando servicios web con la tecnología WebRTC.

Se diseña una base de datos utilizando un diagrama relacional y el gestor será MySQL, y se elabora el correspondiente diagrama de clases y diagramas de secuencia para modelar procesos.

El escenario de pruebas (Figura 1.1), consta de cuatro clientes utilizando el navegador Google Chrome, y de un host servidor que aloja los servicios web, de señalización y la base de datos. Con respecto a la compartición de pantalla solo uno de los participantes comparte la pantalla al resto. Las restricciones del prototipo son que no se puede realizar cambio de organizador de la videoconferencia, no se realiza transferencia de archivos y mientras se efectúa la compartición de pantalla por alguno de los participantes la función de video de dicho participante se desactiva.

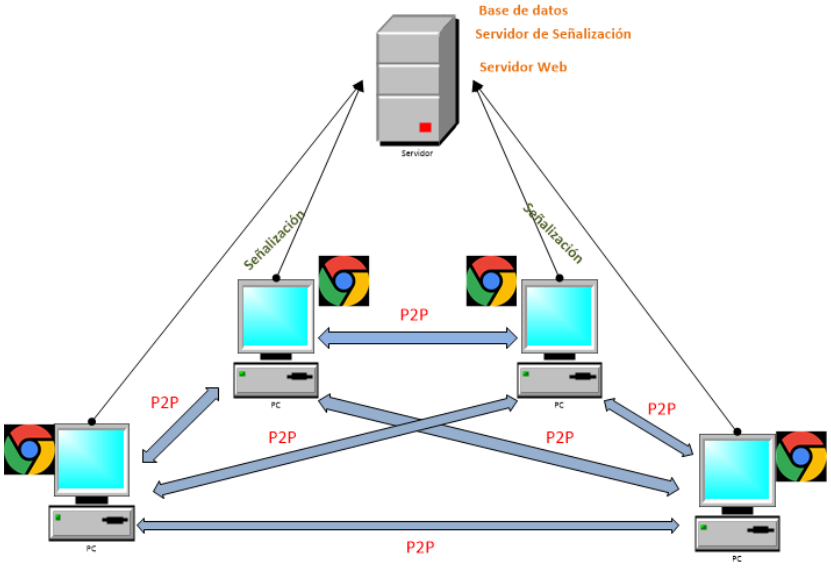

Figura 1.1 Escenario de pruebas

1.3 Marco Teórico

1.3.1 Comunicaciones Basadas En Web

1.3.1.1 Arquitectura Web

La arquitectura web se centra en las tecnologías y principios básicos que sustentan la Web, incluidos los URI[1] y el protocolo HTTP [3].

1.3.1.2 Aplicación Web [4]

Una aplicación web es un programa informático que utiliza navegadores web y la tecnología web para realizar distintas tareas a través de la red. La aplicación web es proporcionada por un servidor web y utilizada por usuarios que se conectan desde cualquier ubicación a través de navegadores, por tanto, tiene una arquitectura tipo cliente-servidor.

Se usan una combinación de *scripts* de lado del servidor para manejar el almacenamiento y recuperación de la información; y *scripts* de lado del cliente para presentar la información a los usuarios.

Las aplicaciones de este tipo requieren un servidor web que distribuye páginas de información formateada a los clientes que las solicitan, y realizan las tareas solicitadas por los mismos, y de una base de datos para almacenamiento de la información.

Las aplicaciones web, pueden proveer cualquier tipo de funcionalidad que las empresas u organizaciones puedan requerir para hacer que estas sean más productivas, por ejemplo, aplicaciones web como catálogos de productos, motores de búsqueda, correo y aquellas que permiten o simplifican la comunicación entre los colaboradores.

Las aplicaciones web dinámicas, permiten a los usuarios interactuar con sus datos para obtener la información que necesitan, y también son útiles en la automatización de procesos.

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

- **Arquitectura Lógica por Capas**

En un modelo de capas, cada capa[2] representa un elemento que procesa o trata información.

[1] URI: *Uniform Resource Identifier*, es una cadena de caracteres utilizada para identificar un recurso, que permite la interacción con representaciones de este a través de la red.
[2] Capa: manera de administrar la complejidad de la aplicación dividiendo la aplicación según los responsabilidades o intereses, ayuda a mantener organizado un código base que crece para que los desarrolladores puedan encontrar fácilmente donde se implementa una funcionalidad determinada

La programación por capas permite la separación de las partes que componen un sistema de software o una arquitectura tipo cliente-servidor.

Actualmente las arquitecturas de aplicaciones web comunes vienen dadas por el modelo de tres capas (Figura 1.2).

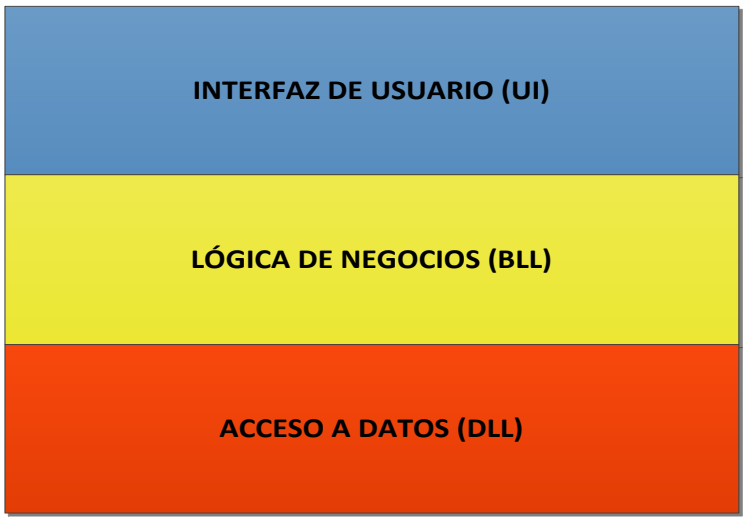

Figura 1.2 Capas de aplicación típicas [5]

- **Arquitectura de tres capas**

Como se muestra en la Figura 1.2, las capas componentes de una aplicación web son:

- *Interfaz de usuario (UI):* también conocida como capa de presentación, presenta el sistema al usuario y captura su información para ser enviada al servidor. Esta capa se comunica exclusivamente con la capa de negocio.
- *Capa lógica de negocios (BLL):* Residen las funciones que se ejecutarán, se reciben peticiones de usuario, se las procesa y se envían las respuestas. Esta capa tiene comunicación tanto con la capa de presentación como con la de acceso a datos.
- *Capa de acceso a datos (DAL):* Almacena los datos del sistema y de los usuarios. Se comunica con la capa de negocio para devolver datos solicitados.

La Figura 1.3 muestra la interacción entre las capas mencionadas anteriormente.

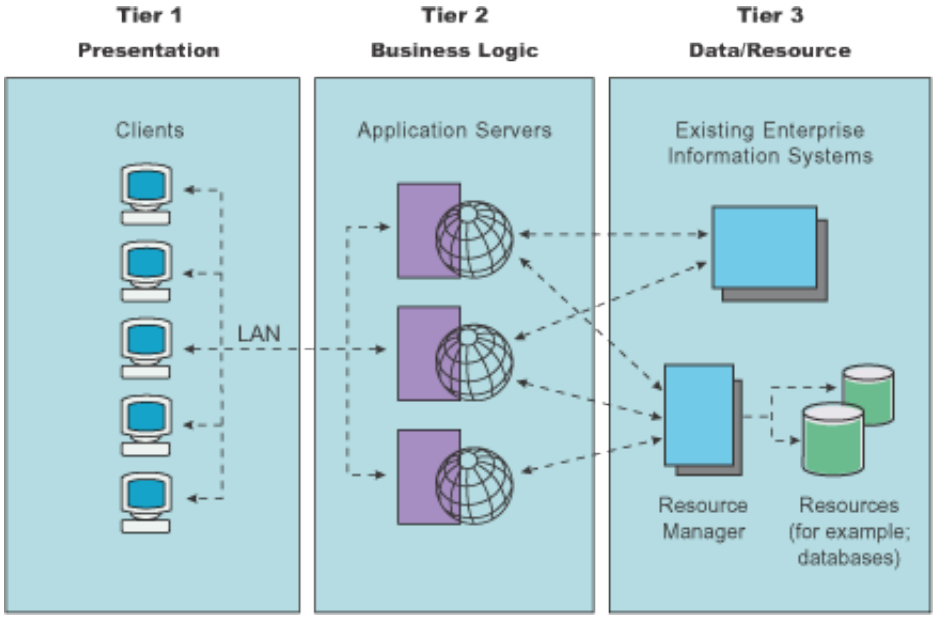

Figura 1.3 Interacción de capas en la arquitectura de tres capas [6]

1.3.1.3 Navegador Web

Es una aplicación de software que permite a los usuarios de Internet acceder a la información de la *Word Wide Web*, para búsqueda de información, o acceso a servicios.

Cada página web, imagen o video individual se identifica con una URL[3] distinta, lo que permite a los navegadores recuperar los sitios y mostrarlos en el dispositivo del usuario.

Los navegadores traducen páginas y sitios web entregados mediante el protocolo HTTP (*Hypertext Transfer Protocol*) en contenido entendible por los usuarios, también son capaces de mostrar contenidos de otros protocolos como HTTPS (*Hypertext Transfer Protocol Secure*), FTP (*File Transfer Protocol*), manejo de correo electrónico y archivos. La mayoría de los navegadores también permiten añadir *plugins*[4] externos para mostrar contenido activo en las páginas como video, audio y contenido Flash.

En la actualidad existen variedad de navegadores web disponibles, con diferentes funciones, apariencia y diseñados para ejecutarse en diferentes sistemas operativos. Entre los más destacados están:

- *Google Chrome:* Navegador web gratuito, desarrollado por Google, compatible con los sistemas operativos Microsoft Windows, Linux, macOS, iOS y Android. En el año 2018 StatCounter (empresa de estadísticas global) considera que Google

[3] URL: *Uniform Resource Locator*, es una referencia a un recurso web que especifica su ubicación en la red informática y un mecanismo para recuperarla.
[4] *Plugin*: software de complemento instalado en un programa para mejorar sus capacidades.

Chrome tiene alrededor del 66% de uso como navegador web de escritorio en todo el mundo [7]. Google Chrome basa su funcionamiento en los motores Blink[5], Webkit[6] y V8[7].

- *Mozilla Firefox:* Es un navegador web gratuito y de código abierto, desarrollado por Mozilla Foundation y su filial Mozilla Corporation. Se encuentra disponible para los sistemas operativos Windows, MacOS, Linux y BSD. Su motor de diseño es Gecko[8][8].

- *Safari:* Navegador web desarrollado por Apple. Está basado en el motor Webkit, se encuentra disponible para los sistemas operativos macOS, iOS [9].

- *Microsoft Edge:* Navegador web desarrollado por Microsoft e incluido en Windows 10, Windows 10 Mobile y Xbox One [10], reemplazando a Internet Explorer como el navegador por defecto en los dispositivos Microsoft.

1.3.2 Videoconferencia

Videoconferencia es una comunicación en línea de personas situadas en lugares distantes entre sí, donde la comunicación es simultánea y bidireccional; los participantes podrán visualizar video y escuchar a los otros participantes en tiempo real. A un nivel más avanzado se añaden otras características, como la compartición de contenido mientras se realiza la llamada [11].

1.3.2.1 Soluciones de Videoconferencia

- **Orientadas a Hardware**

Las soluciones orientadas a hardware o sistemas de videoconferencia dedicados son aquellas que incluyen un grupo de tecnologías asociadas, y posee los componentes necesarios empaquetados en un equipo físico que contendrá una cámara de video [11].

- **Orientadas a Software**

Las soluciones orientadas a software son aquellas que corresponden a programas instalables, que permiten a los ordenadores ser utilizados como dispositivos de videoconferencia. La solución más conocida es Skype [12], que permite la ejecución de

[5] Blink: Es un motor renderizado, desarrollado por Google, mejora la velocidad al cargar el contenido DOM y reduce el tiempo de parada.
[6] Webkit: Plataforma para aplicaciones que funciona como base en algunos navegadores web.
[7] V8: motor de código abierto para JavaScript creado por Google.
[8] Gecko: nombre del motor de presentación desarrollado por la Fundación Mozilla, función de Gecko es leer el contenido de la web, tanto HTML, CSS, XUL, como JavaScript, y presentarlo en pantalla o imprimirlo.

telefonía por Internet y video, incluye transmisión de mensajes de texto e intercambio de archivos.

- **Orientadas a La Web**

Las soluciones orientadas a la web son aquellas que permiten la comunicación entre dos o más usuarios y permiten el intercambio de video, audio y datos en tiempo real, a través de un navegador web. Las soluciones de este tipo pueden manejar protocolos propietarios como Hangouts[9] [13], o manejar arquitecturas de tipo WebRTC para realizar la comunicación en tiempo real.

1.3.3 WebRTC

1.3.3.1 Introducción

WebRTC es un proyecto libre y de código abierto que permite a los navegadores y aplicaciones móviles, realizar comunicaciones en tiempo real, mediante la utilización de APIs[10] JavaScript simples [14].

Persigue el objetivo de que se realicen aplicaciones ricas y de alta calidad para el navegador, dispositivos móviles y dispositivos IoT y permitir que todos se comuniquen a través de un conjunto unificado de protocolos.

WebRTC no requiere descargas o instalación de software adicional. Este proyecto está respaldado por Google, Mozilla y Opera. Está diseñado para proveer comunicaciones de audio y video, transferencia de archivos, compartición de pantalla y capacidades de control vía remota [15].

La manera de señalización de las comunicaciones no está definida en WebRTC, debido a que de esta manera se evita la redundancia y se maximiza la compatibilidad con las tecnologías establecidas; diferentes aplicaciones pueden preferir utilizar diferentes protocolos para señalizar la llamada.

Los estándares subyacentes a WebRTC están siendo desarrollados por: W3C (*World Wide Web Consortium*) y la IETF (*Internet Engineering Task Force*).

[9] Hangouts: Aplicación de mensajería multiplataforma desarrollada por Google, permite mantener conversaciones entre dos o más usuarios.
[10] API: *Application Programming Interface*, conjunto de subrutinas, funciones y procedimientos para ser utilizados como capa de abstracción por otro software.

1.3.3.2 Desarrollo e Historia [16]

WebRTC se introdujo por primera vez y fue declarado como *open-source* en mayo del 2011 por la compañía Google, pero su lanzamiento comercial fue realizado a finales del 2012 en el navegador Google Chrome.

El aparecimiento de WebRTC se da después de que en 2010 Google adquiriera On2[11], compañía que había desarrollado el códec de video VP8 mismo que fue presentado al mundo con el nombre de proyecto WebM[12], y cuya idea de lanzamiento fue reemplazar al códec H.264[13] para reducir el costo de patentes.

En el mismo año Google adquiere GIPS[14], compañía dedicada al desarrollo de *frameworks* de medios, que facilitan el desarrollo de aplicaciones tipo VoIP y de videollamadas.

Luego de estas adquisiciones la empresa tomó los activos de GIPS y On2 y los hizo *open-source*, incluyendo todos los códecs de voz y video, hasta ese momento propietarios, y adicional agregaron una API de JavaScript que serviría de capa de integración para los navegadores web. Desde ese momento WebRTC pasó a ser impulsado como estándar en IETF y W3C.

WebRTC apareció como un *framework* abierto para la web que permite comunicaciones en tiempo real en el navegador, que incluyen los componentes básicos para las comunicaciones en la web: red, audio y video; estos componentes se pueden acceder a través de una API de JavaScript [17].

1.3.3.3 Arquitectura WebRTC [15]

La arquitectura general de WebRTC se detalla en la Figura 1.4, está compuesta por códecs, motores de medios y capa de transporte.

[11] On2: conocida como The Duck Corporation, compañía especializada en desarrollar tecnologías de compresión de video.
[12] WebM: formato multimedia abierto y libre, desarrollado por Google compuesto por el códec de vídeo VP8 y el códec de audio Vorbis.
[13] H.264: Norma que define un códec de video de alta compresión, desarrollada conjuntamente por el ITU-T *Video Coding Experts Group* (VCEG) y el ISO/IEC *Moving Picture Experts Group* (MPEG).
[14] GIPS: empresa que proporcionaba motores de medios de voz y vídeo licenciados sobre IP para programadores de software y fabricantes de infraestructura

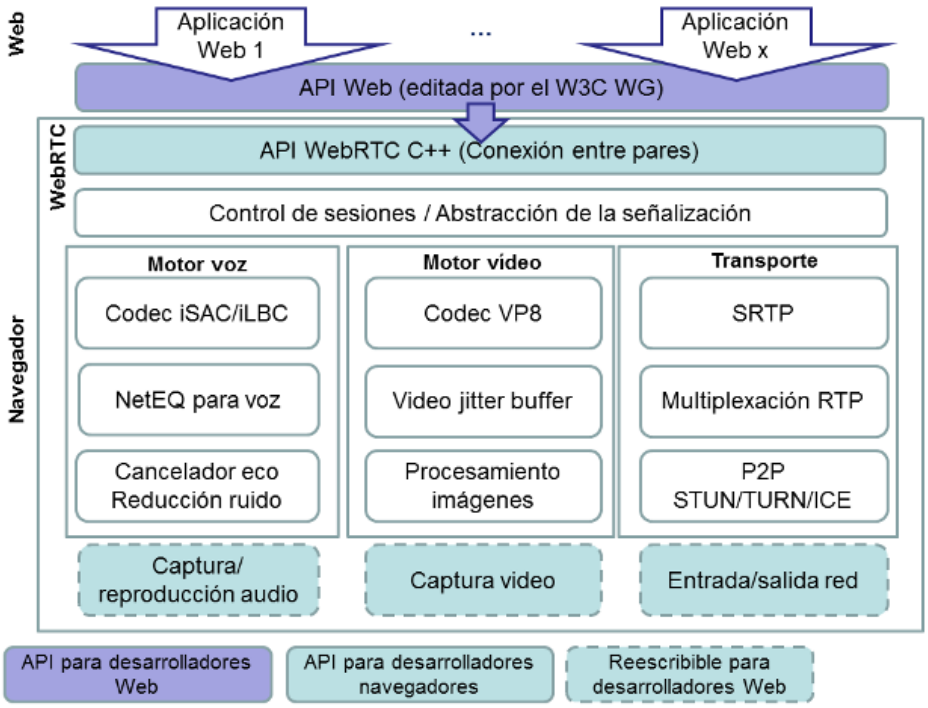

Figura 1.4 Arquitectura WebRTC [15]

A continuación, se detallan los componentes más importantes de la arquitectura:

- **Web API:** Una API para ser utilizada por desarrolladores externos de aplicaciones web, similares a videochat.
- **WebRTC Native C ++ API:** Una capa API que permite a los fabricantes de navegadores implementar fácilmente la propuesta de la API web.
- **Gestión de sesiones:** Una capa de sesión abstraída, que permite la configuración de llamadas y la capa de administración. Esto deja la decisión de implementación del protocolo de señalización al desarrollador de la aplicación.
- **Transporte/Sesión:** Una pila de red para RTP, el protocolo de tiempo real.
- **Motor de voz:** Es un *framework* para la cadena de medios de audio, desde la tarjeta de sonido a la red.
- **Motor de Video:** Es un *framework* para la cadena de medios de video, de la cámara a la red y de la red a la pantalla.

1.3.3.4 Stack de Protocolos [2]

En la Figura 1.5 se muestra la distribución general del stack de protocolos de WebRTC, en la parte superior izquierda se notan los protocolos para medios y en la parte superior

derecha aquellos para transferencia de datos, ambos correspondientes a la capa Aplicación del modelo ISO/OSI.

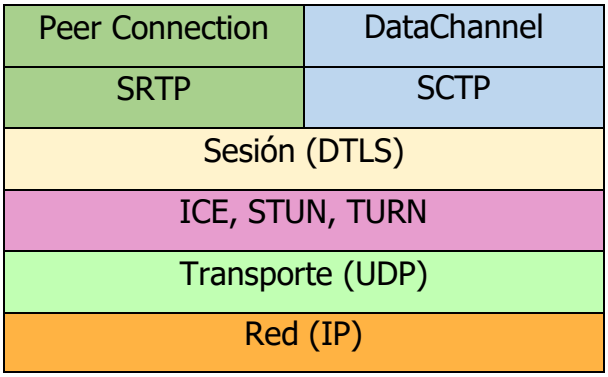

Figura 1.5 Stack de protocolos de WebRTC [2]

El protocolo de transporte utilizado en WebRTC es UDP (*User Datagram Protocol*), debido a que la transmisión de audio, video y datos entre navegadores requiere más la velocidad ante la confiabilidad que ofrece TCP (*Transmission Control Protocol*), aunque el uso de TCP también es posible.

Desde que WebRTC adopta UDP en la parte de transporte, se añaden protocolos que brinden confiabilidad, encripción y control de flujo. WebRTC requiere que todos los datos que se transfieren estén encriptados.

- *DTLS (Datagram Transport Layer Security)*: Protocolo encargado de agregar seguridad a UDP, evitando la falsificación de mensajes y manipulación de mensajes en UDP.
- *SRTP (Secure Real-time Transport Protocol):* Es el protocolo usado por WebRTC para la entrega de medios, se usa para transferir las transmisiones de audio y de video entre los clientes.
- *SCTP (Stream Control Transmission Protocol):* Agrega funciones de multiplexación, control de flujo y control de congestión a UDP [18].

1.3.3.5 Arquitecturas de Videoconferencia

En general, hay tres modelos principales de implementación de una videoconferencia utilizando WebRTC [19]:

- **MCU (*Multipoint Control Unit*):** la unidad de control multipunto (Figura 1.6), converge todas las secuencias de video y audio en una sola transmisión donde todos los clientes están conectados a un solo puerto.

En la MCU, el servidor recibe todas las transmisiones de los clientes, luego decodifica y codifica de nuevo en una sola transmisión y envía de vuelta a los clientes respectivos.

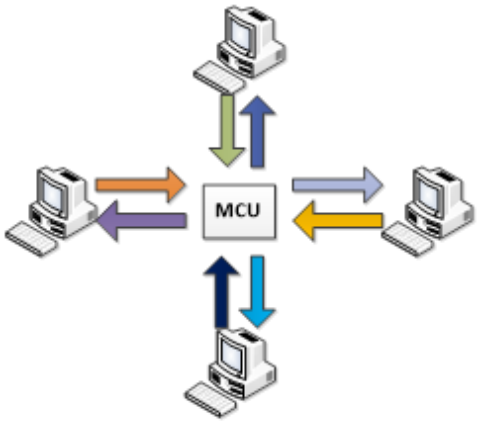

Figura 1.6 Arquitectura MCU

- **SFU (*Selective Forwarding Unit*):** en SFU todos los clientes envían diferentes flujos de video con diferentes códecs y tasas de bits. El servidor reenvía los flujos a cada uno de los participantes de la videoconferencia. (Figura 1.7)

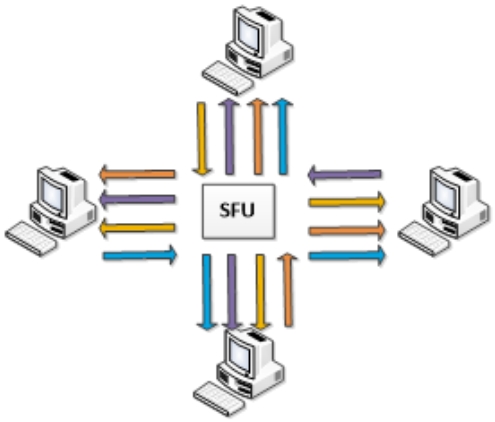

Figura 1.7 Arquitectura SFU

- **MESH:** la arquitectura para videoconferencia tipo mesh es ampliamente usada por aplicaciones WebRTC. En esta arquitectura cada cliente individual se conecta a otro cliente e intercambia los datos y los flujos de medios directamente entre los navegadores sin la participación del servidor (Figura 1.8). Basado en su ancho de banda podría escalar hasta diez participantes remotos.

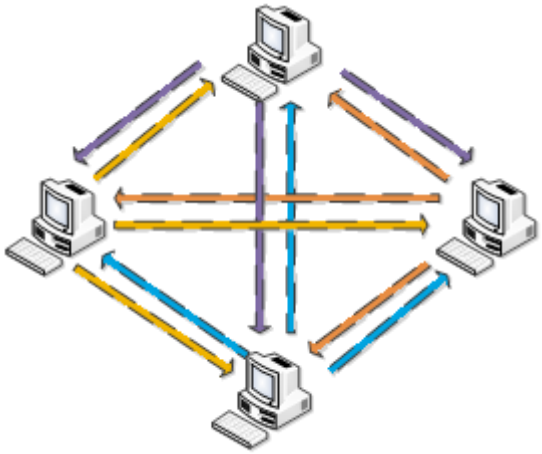

Figura 1.8 Arquitectura tipo MESH

1.3.3.6 Señalización en WebRTC

Dado que WebRTC no define la señalización de las comunicaciones a continuación se explica su función dentro de la comunicación.

El servidor de señalización coordina todas las comunicaciones e intercambia la información necesaria para establecer una conexión entre pares [2]. El servidor de señalización es de tipo centralizado, no *peer to peer*.

Se usa también para intercambiar otro tipo de información como: mensajes de control para iniciar llamadas, desconectar o recibir llamadas, mensajes de error, datos de red, metadatos de medios etc.

WebRTC utiliza el protocolo SDP (*Session Description Protocol*) para describir los parámetros de la conexión *peer to peer* [2], algunos de los parámetros que describe son: tipos de medios ofrecidos (video o audio), las resoluciones de estos, los códecs de medios, capacidades de decodificación, información de conexión de red entre otros.

El protocolo JSEP[15] es el encargado de abstraer todos los parámetros descritos en un objeto RTCPeerConnection[16].

1.3.3.7 WebRTC y Navegadores [20]

La iniciativa de WebRTC se encuentra soportada actualmente por Google Chrome, Mozilla Firefox y Opera, tanto en sus versiones de escritorio como para Android.

[15] JSEP: *JavaScript Session Establishment Protocol*, supone un modelo en el que una aplicación JavaScript se ejecuta dentro de un tiempo de ejecución que contiene API de WebRTC.
[16] RTCPeerConnection: representa una conexión WebRTC entre un computador local y un par remoto.

Cabe mencionar que no todos los navegadores tienen las mismas características de WebRTC al mismo tiempo.

Las diferencias entre los navegadores Google Chrome y Mozilla Firefox con respecto a la sintaxis de los métodos utilizados se muestran en la Figura 1.9.

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

Figura 1.9 APIs WebRTC compatibles con los navegadores Chrome y Firefox [20]

1.3.3.8 APIs WebRTC

Las APIs nativas para la comunicación en WebRTC definidas por W3C son:

- **MediaStream**

Describe un flujo de datos de audio o video, los métodos para trabajar con ellos, las limitaciones de este tipo de datos, las respuestas de error y los eventos asociados al proceso [21]. Los dos componentes principales en MediaStream API son las interfaces [22]:

- **MediaStreamTrack:** esta interfaz representa un flujo de contenido de los medios, un flujo representa gran cantidad de tracks que pueden ser de audio o de video.
- **MediaStream:** se usa para agrupar varios objetos MediaStreamTrack en una unidad que se puede grabar o representar en un elemento multimedia.

En la Figura 1.10 se muestra un diagrama de bloques que representa el API MediaStream.

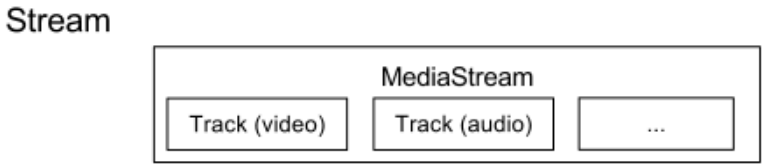

Figura 1.10 Diagrama de bloques de MediaStream [23]

- **PeerConnection**

RTCPeerConnection representa una conexión WebRTC entre la computadora local y el par remoto [24]. Provee métodos para conectarse al par remoto, mantener y monitorear la conexión además de cerrarla si es necesario.

El constructor del objeto RTCPeerConnection toma un objeto de configuración que contiene las direcciones de los servidores que ayudan a establecer sesiones a través de NAT (*Network Address Translation*) y *firewalls*.

Se utiliza el protocolo ICE (*Internet Communications Engine*) junto con los servidores STUN (*Session Traversal Utilities for NAT*) y TURN (*Traversal Using Relay NAT*) para poder atravesar NAT y *firewalls*. ICE permite encontrar el mejor camino entre los pares, utilizando una conexión directa entre ellos o indirecta en el caso de que los clientes necesiten traducción de direcciones IP (NAT).

En el caso de NAT asimétrico ICE utilizará un servidor STUN, que permitirá que los clientes descubran su dirección IP pública y el tipo de NAT que está detrás y se utilizará esta información para el establecimiento de conexión de medios. Usualmente el servidor STUN solo se usa durante la configuración de la conexión, luego de ello los medios fluirán directamente entre los clientes.

En el caso de que el servidor STUN no pueda establecer la conexión, ICE activará TURN, usualmente utilizado para NAT de tipo simétrico, permitiendo la conexión de medios mediante la retransmisión de medios entre los pares WebRTC.

Todas las soluciones WebRTC, deben estar preparadas para admitir los servicios STUN y TURN y sus requisitos de procesamiento.

En la Figura 1.11 se muestra un diagrama de bloques de los componentes principales del API PeerConnection.

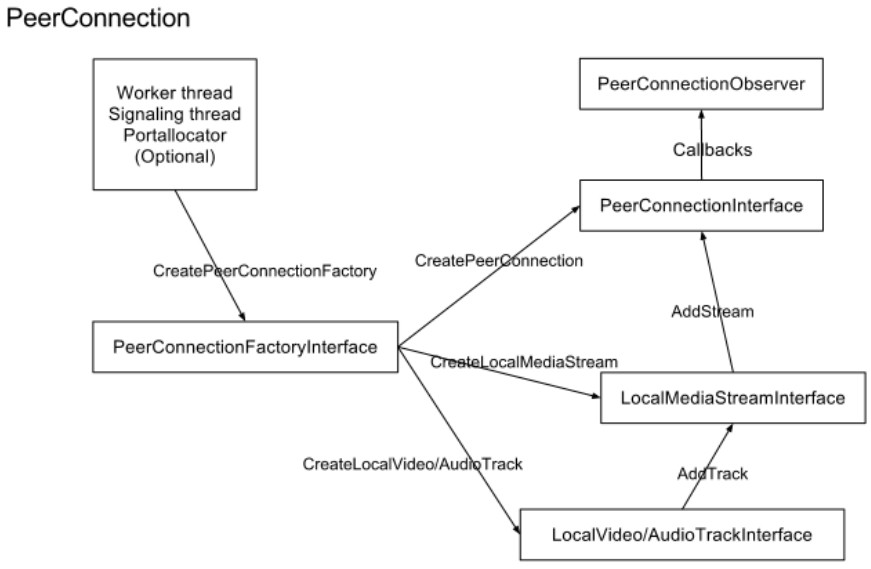

Figura 1.11 Diagrama de bloques de API PeerConnection [23]

- **DataChannel**

Es una característica que permite abrir un canal entre pares y se usa para transferir datos de manera bidireccional *peer to peer.* Cada canal de datos está asociado a un RTCPeerConnection.

1.3.4 HTML5 [25]

HTML5 es una combinación mejorada de JavaScript, HTML y CSS[17]; HTML provee los elementos estructurales, CSS se encarga de la parte visual de la estructura, y JavaScript tiene todo el poder necesario para proveer dinamismo, y en el caso de WebRTC provee las APIs necesarias para la construcción de aplicaciones web completamente funcionales.

1.3.4.1 Estructura Global HTML

Los documentos HTML se encuentran estrictamente organizados, declarando cada sección con etiquetas específicas. En la Figura 1.12 se muestra la organización de la mayoría de los sitios web actuales, considerando la estructura básica de HTML5, y posteriormente se describen brevemente cada una de las etiquetas.

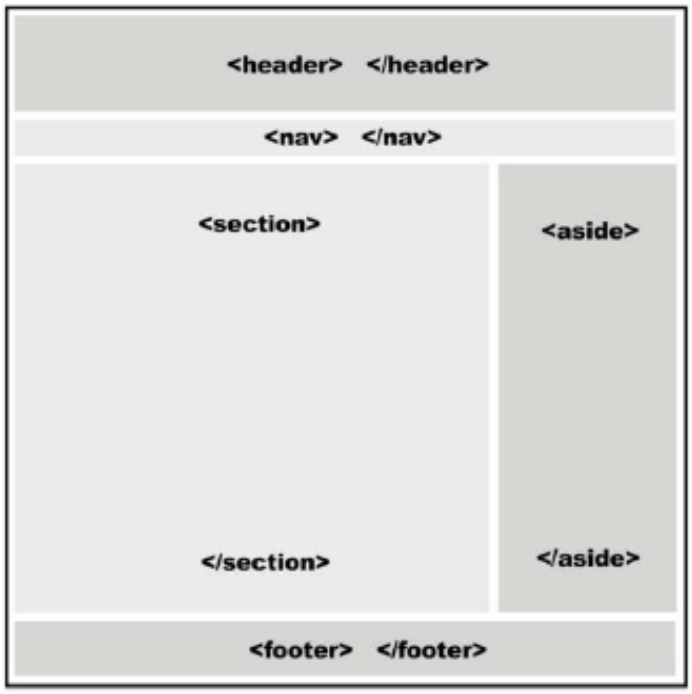

Figura 1.12 Representación visual de un diseño utilizando elementos HTML5 [25]

- *<header> (Cabecera):* Brinda información introductoria. Usualmente se coloca el título, subtítulos y descripción breve del sitio.

[17] CSS: *Cascading Style Sheets*, describe la presentación de documentos HTML.

- *<nav> (Barra de navegación):* Donde se suele ofrecer menús, o lista de enlaces para facilitar a los usuarios la navegación a través del sitio.
- *<section> (Sección de información principal):* Se ubica la información más importante que puede ser representada en diferentes formas, en varios bloques, o columnas.
- *<aside> (Barra Lateral):* Usualmente ubicada al lado de la información principal, brinda datos relacionados a la información principal, pero de menor importancia.
- *<footer> (Pie):* Se usa para cerrar el diseño del documento.

1.3.5 CSS

CSS es el lenguaje para describir la presentación de las páginas web, incluido los colores, el diseño y las fuentes [26]. CSS permite la separación del contenido del documento de la presentación de este.

CSS no es parte de la especificación de HTML5, pero trabaja en conjunto con HTML para proveer estilos visuales a los elementos del documento.

1.3.6 JavaScript [27]

JavaScript es el lenguaje de programación de la Web. La mayoría de los sitios web modernos, los navegadores, consolas de video juegos, tabletas y *smartphones* incluyen intérpretes de JavaScript.

JavaScript es un lenguaje de programación interpretado, no tipificado, dinámico y de alto nivel, adecuado para estilos de programación funcionales y orientados a objetos. Deriva su sintaxis de Java, sus funciones de primera clase de Scheme[18] y su herencia basada en prototipos de Self[19].

1.3.7 Bootstrap [28]

Bootstrap es un *framework* desarrollado y liberado por Twitter cuyo objetivo principal es facilitar el diseño de sitios y aplicaciones web; Bootstrap es un conjunto de herramientas de código abierto, que incluye varias plantillas de elementos de diseño basado en HTML, CSS y JavaScript.

[18] Scheme: lenguaje de programación interpretado, expresivo y soporta varios paradigmas con semántica sencilla.
[19] Self: lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems, basado en idea de prototipos en lugar de clases.

1.3.8 APIs HTML5 JavaScript [29]

Las APIs HTML5 usando el lenguaje JavaScript se han construido para permitir a los desarrolladores crear funcionalidades complejas de manera simple, proporcionando una sintaxis más fácil.

Tabla 1.1 APIs de navegador

API	Descripción	Ejemplo
De manipulación de documentos	Permite manipulación de documentos cargados en el navegador.	DOM[20] permite manipular HTML y CSS
De obtención de datos del servidor	Permite actualizar pequeñas secciones de una página web, sin volver a cargar toda la página.	XMLHttpRequest Fetch API
De manipulación de gráficos	Permiten actualizar mediante programación los datos de pixeles contenidos en un elemento HTML.	Canvas WebGL
De Audio y Video	Manipulación de multimedia, creación de controles personalizados para reproducción de audio y video	HTMLMediaElement Web Audio API WebRTC
De dispositivos	Permite manipular y recuperar datos de hardware de dispositivos modernos.	Geolocation
De almacenamiento	Para almacenamiento de datos en el lado del cliente.	Web Storage API IndexedDB API

[20] DOM: *Document Object Model*, interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML.

JavaScript del lado del cliente, tiene muchas API disponibles; que no son parte del lenguaje en sí, sino que están construidas sobre el lenguaje JavaScript central. Se dividen en dos categorías:

a) **APIs de Navegador:** Son aquellas integradas en el navegador web, pueden exponer datos de este y del entorno informático circundante y se podría utilizar para realizar funcionalidades complejas útiles. Las APIs de Navegador más conocidas se encuentran en la Tabla 1.1.

b) **APIs de terceros:** No están integradas en el navegador de manera predeterminada, y generalmente se debe obtener su código de algún lugar de la Web. Las APIs de terceros más comunes se muestran en la Tabla 1.2

Tabla 1.2 APIs de terceros más comunes

API	Descripción
Twitter API	Permite mostrar los últimos tweets en el sitio web
Google Maps API	Manipular mapas en el sitio web
Facebook Suite APIs	Permite usar varias partes del ecosistema de Facebook.
YouTube API	Permite incrustar videos de YouTube en el sitio.

1.3.9 Node.js

1.3.9.1 Definición [30]

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Usa un modelo de operaciones entrada/salida sin bloqueo y orientado a eventos asíncronos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

Node.js puede generar contenido dinámico en la página, realizar operaciones como: crear, abrir, leer, escribir, eliminar y cerrar archivos en el servidor, además puede realizar operaciones en la base de datos.

1.3.9.2 NPM

NPM es un gestor de paquetes, que sirve para administrar los módulos de Node.js, además de distribuir paquetes y agregar dependencias de manera sencilla.

1.3.9.3 Módulos de Node.js

Los módulos de Node.js pueden ser considerados como las librerías usadas en JavaScript, que sirven para mantener cierto grupo de funciones separadas del código principal.

A continuación, se explican los módulos y *frameworks* de Node.js relevantes para la elaboración de este Trabajo de Titulación:

- *Express:* es el *framework* más popular de Node, que proporciona mecanismos para:
  - Escritura de manejadores de peticiones con diferentes verbos HTTP (GET, POST, PUT, DELETE) en diferentes rutas (*paths*).
  - Se puede integrar con motores de renderización de vistas (Pug/Jade) para generar respuestas basadas en la introducción de datos en plantillas.
- *https:* proporciona una forma de hacer que Node.js transfiera datos a través del protocolo HTTP TLS / SSL, que es el protocolo HTTP seguro. Los métodos de este módulo pueden ser utilizados para crear un servidor de tipo HTTPS que escuchará en los puertos del servidor y brinda una respuesta al cliente.
- *bcrypt:* módulo que permite cifrar los datos mediante la función de *hashing* de *passwords* denominada Bcrypt.
- *passport:* es un *middleware* de autenticación compatible con *Express*, que autenticará solicitudes a través de complementos conocidos como estrategias[21]. Antes de autenticar las solicitudes, se deben configurar las estrategias.
- *fs:* el módulo del sistema de archivos permite trabajar con el sistema de archivos de la computadora.
- *mysql:* este módulo será útil para acceder a una base de datos MySQL, mediante un controlador de base de datos MySQL.

1.3.10 Socket.IO [31]

Socket.IO es una librería de JavaScript para aplicaciones en tiempo real, permite la comunicación bidireccional basada en eventos. Funciona en todas las plataformas, navegadores o dispositivos, centrándose por igual en la fiabilidad y la velocidad.

[21] Estrategias: mecanismos de autenticación que reconocen que cada aplicación tiene requisitos de autenticación únicos, pueden variar desde la verificación de las credenciales de nombre de usuario y contraseña, autenticaciones delegadas por OAuth o autenticaciones federadas.

Socket.IO requiere: un servidor Node.js y una librería cliente JavaScript para el navegador. En el lado del servidor Socket.IO será incluido como un módulo de Node.js mediante npm y utilizado en el servidor de señalización.

1.3.10.1 APIs Socket.IO

A continuación, se describen las APIs principales de Socket.IO tanto para el cliente como para el servidor.

- **Servidor**

Antes de utilizar las APIs del lado del servidor, se debe incluir el módulo mediante require('socket.io'). A continuación, se mencionan las APIs principales:

- *Server :* Crea un nuevo servidor.
- *Socket:* Es la clase para interactuar con los clientes del navegador.
- *Client:* Representa una conexión de transporte entrante. El cliente puede asociarse con sockects multiplexados.

- **Cliente**

Los eventos más importantes del API Socket.IO del lado del cliente se mencionan a continuación:

- *Connect:* Se lanza cuando se produce la conexión con el servidor.
- *Error:* cuando se produce un error de conexión.
- *Reconnect:* lanzado después de una reconexión exitosa.
- *Disconnect:* lanzado después de una desconexión exitosa.

1.3.11 MySQL [32]

Es un sistema de administración de bases de datos SQL de código abierto, desarrollado, distribuido y respaldado por Oracle Corporation.

Las bases de datos MySQL son relacionales y almacenan los datos en tablas separadas. Las estructuras de la base de datos están organizadas en archivos físicos optimizados para la velocidad, además se ofrece un entorno de programación flexible dado su modelo lógico con objetos como bases de datos, tablas, vistas, filas y columnas.

El software de base de datos MySQL es un sistema cliente-servidor que consiste en un servidor SQL multiproceso que admite diferentes *back-ends*, varios programas de cliente y

bibliotecas, herramientas administrativas y una amplia gama de interfaces de programación de aplicaciones.

1.3.12 Scrum [33]

Scrum es un marco de trabajo para el desarrollo de software basado en los métodos ágiles, cuyo objetivo es el control activo sobre el estado actual del software. Scrum da prioridad a los individuos y las interacciones sobre los procesos y las tareas.

Es adecuado para entornos de desarrollo caracterizados por tener:

- Incertidumbre con respecto a que se plantee el objetivo a alcanzar sin proporcionar un plan detallado del producto.
- Equipos que se organizan por si solos, y que reúnan las características de autonomía, autosuperación para mejorar las soluciones iniciales y auto-enriquecimiento para complementar las soluciones.
- Control moderado, referente a que se establecerán controles suficientes, pero no tan estrictos para no impedir la creatividad de los miembros del equipo.
- Transmisión de conocimiento entre los miembros del equipo.

1.3.12.1 Roles

Como en todo proceso de desarrollo de software existen roles, los cuales determinan comportamientos y actividades importantes del proyecto, en Scrum se definen cinco grupos:

- **Propietario del producto:** persona que determina las prioridades del proyecto, que guiará al equipo Scrum hacia el cumplimiento de los objetivos.
- **Scrum Manager:** se encarga de gestionar y facilitar la ejecución del proyecto, es quien debe asegurarse del cumplimiento de la metodología y atender asuntos externos al proyecto.
- **Equipo Scrum:** encargados de construir el producto, conformado por los desarrolladores.
- **Interesados:** también llamados *stakeholders* son las personas interesadas en promover el proyecto.
- **Usuarios:** aquellos que realizarán las pruebas lógicas a la aplicación y verificarán si cumple con las expectativas.

1.3.12.2 Eventos

- ***Sprint*:** es la base de Scrum, su duración máxima aproximada es de 30 días, donde se efectúa el desarrollo de software y realizan las reuniones. El *sprint* está compuesto por la planificación del *sprint*, reuniones diarias, revisión del *sprint* y retrospectiva del *sprint*.
- **Planificación del *sprint*:** en esta reunión el propietario del producto explica a todo el equipo las prioridades y a partir de ellas se elabora la pila del *sprint*.
- **Reuniones diarias:** son reuniones de entre 15 y 30 minutos de duración entre el Scrum Manager y el equipo.
- **Revisión del *sprint*:** reunión informativa donde se presenta el incremento, se plantean sugerencias de cambio y se anuncia el *sprint* siguiente.
- **Retrospectiva del *sprint*:** se realiza después de cada *sprint* y se recolectan los comentarios mismo, para mejorar los procesos.

1.3.12.3 Artefactos

Son los modelos de información generados durante el proceso de desarrollo, Scrum cuenta con tres:

- **Pila del producto (*Product Backlog*):** es una lista ordenada y priorizada de las tareas que componen el proyecto siendo el propietario del producto el que decide sobre ella.
- **Pila del Sprint (*Sprint Backlog*):** son los requisitos o ítems seleccionados de la pila del producto para su ejecución en un *sprint*.
- **Incremento:** es una parte del producto desarrollado en un *sprint*, que puede ser utilizado; contiene las pruebas y documentación.

En la Figura 1.13 se puede observar el proceso de la metodología Scrum.

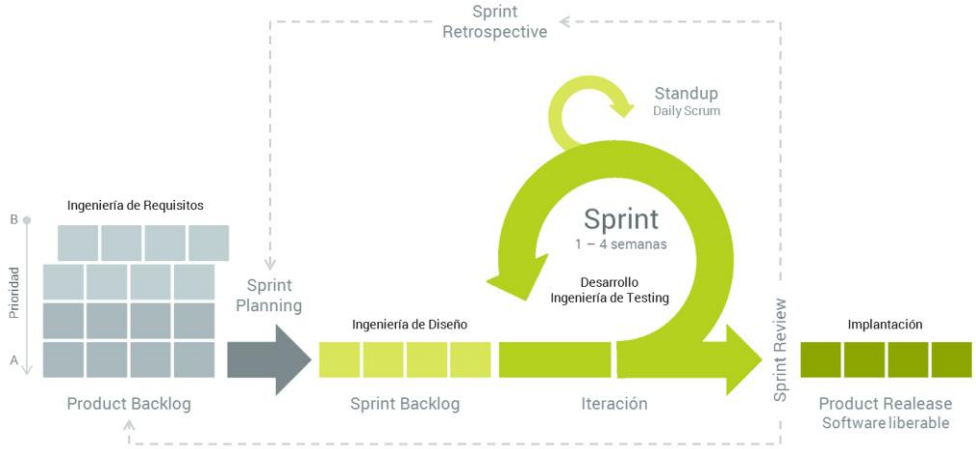

Figura 1.13 Proceso de la metodología Scrum [34]

1.3.13 Trabajos Relacionados

La adopción de WebRTC actualmente está influenciada por los avances tecnológicos que incluyen el soporte de los navegadores y los productos comerciales. En cuanto a WebRTC en el campo educativo se puede referir que uno de los cambios más innovadores dentro del sistema educativo es el uso de video para traspaso del conocimiento, que refuerza los métodos de enseñanza a distancia y se ofrece a estudiantes y profesores la capacidad de conectarse y comunicarse a través de chat de video directamente a través de su navegador.

Existen varias aportaciones de proyectos experimentales del uso de WebRTC orientados al campo educativo como, por ejemplo: “Diseño e implementación de un sistema de enseñanza de video en línea basado en WebRTC” [35] para el sistema operativo Android, este sistema de enseñanza ofrece una plataforma abierta de aprendizaje amigable y basada en estándares que permite la colaboración de enseñanza en audio y video en cualquier momento con los participantes en cualquier ubicación. Este proyecto fue desarrollado con la idea de impulsar el desarrollo de la educación en red, además de promover la popularidad de la educación continua.

La usabilidad y accesibilidad de estos servicios educativos determinará la adopción en uso, y los sistemas de enseñanza de video en línea basados en WebRTC tienen un espacio de desarrollo y condiciones de aplicación muy amplios.

Al ser un sistema orientado a la educación y cuyos usuarios serán profesores y estudiantes, el sistema se compuso de cuatro módulos (Figura 1.14), de los cuales el módulo Online Classroom, es aquel que realiza un proceso de enseñanza de audio y video en tiempo real entre docentes y estudiantes mediante el uso de WebRTC.

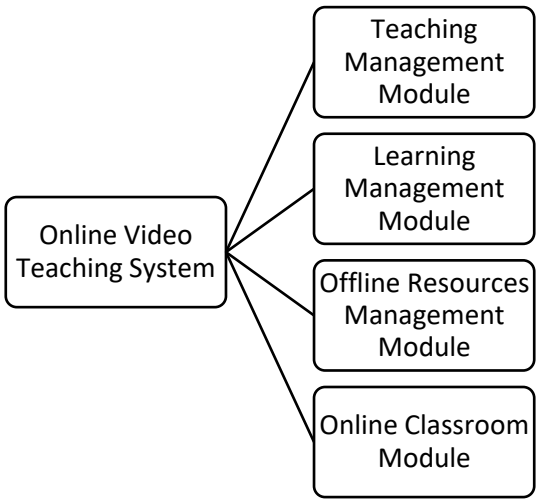

Figura 1.14 Módulos del sistema desarrollado por [35]

Por otra parte, existen proyectos mucho más ambiciosos en los cuales el uso de WebRTC esta inmiscuido de manera parcial para las comunicaciones en tiempo real, tal es el caso del proyecto: "*Development of Multimedia System for Supporting Education in Rural Areas*" [36], desarrollado en Indonesia; en el cual se planteaba la creación de un sistema para mejorar la calidad de la educación denominado Virtual Classbox 4.1.

Este sistema fue desarrollado para mejorar la calidad de educación en áreas rurales con tecnología de videoconferencia, dado que en las zonas rurales los procesos educativos enfrentan problemas debido a los recursos de enseñanza limitados.

Virtual Classbox 4.1 es una herramienta de conferencia multimedia para apoyar el proceso de aprendizaje a distancia. Se desarrolló utilizando WebRTC y Skype [36]. WebRTC se utiliza como base VoIP en comunicaciones multimedia en tiempo real, usando JavaScript y HTML. Por lo tanto, un navegador web se puede utilizar para comunicarse, por lo que puede ser operado fácilmente.

La configuración del sistema de Virtual Classbox se puede ver en la Figura 1.15. El dispositivo Virtual Classbox obtiene tres entradas; el micrófono proporciona la entrada de sonido, la cámara de mano o la webcam proporcionan la entrada de video, el teclado-mouse se usará para dar comandos al Dispositivo de la Caja de Clase Virtual.

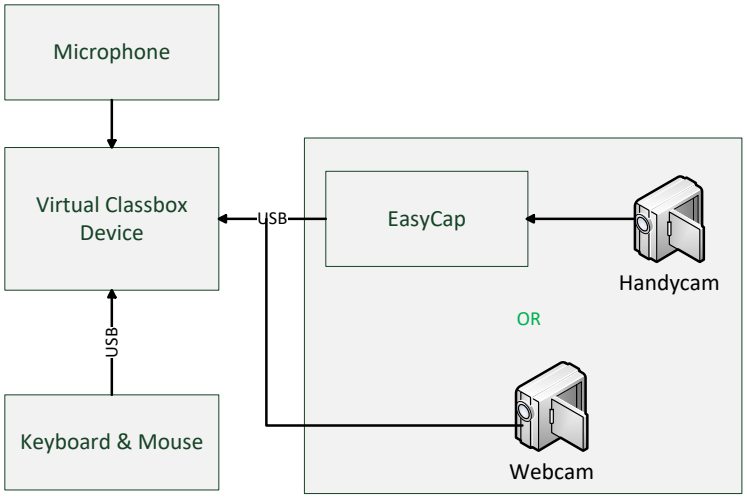

Figura 1.15 Configuración del sistema Virtual Classbox [36]

2. METODOLOGÍA

En este capítulo se detallan tres procesos fundamentales: el diseño del sistema, la codificación de los módulos y la implementación del prototipo. La fase de diseño comprende la arquitectura del prototipo, análisis de requerimientos y modelado del sistema mediante diagramas UML. La fase de codificación, que se llevó a cabo siguiendo los lineamientos de la metodología Scrum, detalla los *sprints* correspondientes para el cumplimiento de los requerimientos. La fase de implementación corresponde a la puesta en marcha del prototipo fuera del entorno local, para ser sometido a pruebas posteriormente.

2.1 Diseño del Prototipo

En esta sección se define la arquitectura del prototipo en base al modelo de tres capas, se definen los requerimientos funcionales y no funcionales de acuerdo con el alcance definido, se modelan la base de datos y la estructura del sistema mediante diagramas UML.

2.1.1 Arquitectura

La aplicación será diseñada con un modelo de tres capas, donde la carga se divide en tres partes que son:

- *Capa de acceso a datos:* es la capa encargada del manejo de la persistencia de la información y roles de los usuarios, así como información complementaria de la aplicación. Provee el acceso a la base de datos que tiene la información, roles de usuario e información de ayuda.
- *Capa de negocio:* se encargará de la lógica de negocio, es decir que recibirá las peticiones de los clientes y devolverá las respuestas a los mismos en función de sus necesidades, esta capa se encuentra reflejada en los métodos del servidor web Node.js.
- *Capa de presentación:* manejará la lógica de presentación, presenta las interfaces de usuario para la interacción usuario-sistema; esta capa se encuentra dividida entre el servidor Web Node.js y el navegador que presenta las vistas al usuario.
- *Base de datos de usuarios, roles e información*: tendrá toda la información de los usuarios y roles registrados en la aplicación y la información a mostrar en el módulo de ayuda.

En la Figura 2.1 se presenta la arquitectura del sistema y la interacción entre sus capas.

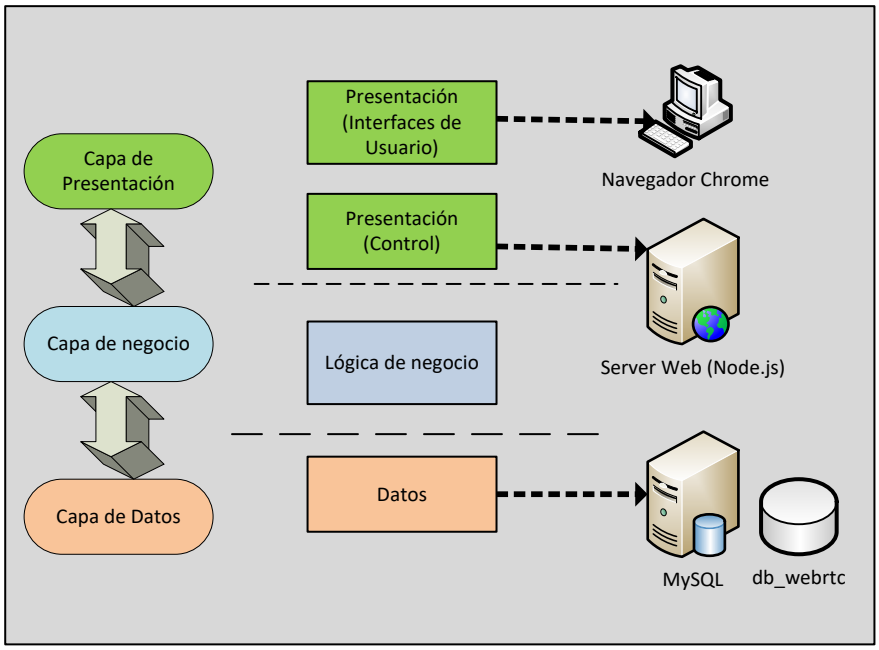

Figura 2.1 Diagrama arquitectural del sistema e interacción entre capas

2.1.2 Análisis de requerimientos

De acuerdo con el alcance de este Trabajo de Titulación los requerimientos generales propuestos para el sistema que se desarrolló son los siguientes:

- Permitir la realización de videoconferencia en el navegador entre 4 participantes.
- Permitir compartición de pantalla por parte del organizador de la videoconferencia.
- Permitir intercambio de mensajes entre los participantes.
- Permitir creación, modificación, lectura y eliminación de usuarios.
- Permitir asignación y eliminación de roles de usuario.
- Contar con un módulo de ayuda que explique el proceso de desarrollo de la aplicación.
- Contar con un rol de administrador del sistema para operaciones CRUD de usuarios.

2.1.3 Especificación de requerimientos

En esta sección se especifican requerimientos funcionales que describen los servicios que prestará el sistema, y no funcionales que se refieren a propiedades del sistema como seguridad y disponibilidad.

2.1.3.1 Requerimientos funcionales

Se han definido los siguientes requerimientos funcionales:

- La aplicación permite al administrador crear, leer, modificar y eliminar usuarios.

- La aplicación permite al administrador y al usuario modificar su información personal y contraseña.
- El administrador podrá asignar o eliminar un rol ya sea administrador o usuario.
- La aplicación permite a los usuarios registrarse por sí mismos proporcionando cierta información.
- La aplicación permite a los usuarios y administradores iniciar sesión, utilizando su correo electrónico y contraseña.
- La aplicación permitirá a los usuarios realizar videoconferencia hasta con tres participantes más que estén registrados en la aplicación.
- La aplicación permitirá al usuario organizador de la videoconferencia compartir su pantalla con hasta tres usuarios registrados.
- La aplicación permitirá a un usuario intercambiar mensajes con hasta tres usuarios registrados.
- La aplicación proporcionará información acerca de la implementación y uso del sistema.
- La aplicación proporcionará un manual de uso para los usuarios.

2.1.3.2 Requerimientos no funcionales

Se han definido los siguientes requerimientos no funcionales:

- Para ingresar a la aplicación se utilizarán contraseñas.
- Las contraseñas no pueden estar en texto visible.
- Las contraseñas se almacenarán encriptadas en la base de datos.
- No se pueden registrar usuarios o administradores duplicados.
- La aplicación debe soportar hasta cuatro usuarios autenticados en el módulo de comunicación.

2.1.4 Diagrama de Casos de Uso

Después del análisis de los requerimientos funcionales se elaboró el diagrama de casos de uso que se muestra en la Figura 2.2.

Los actores que interactúan con el sistema son: usuario y administrador. Los casos de uso definidos son las acciones que cada uno de los actores realizará en el sistema para conseguir algún resultado específico.

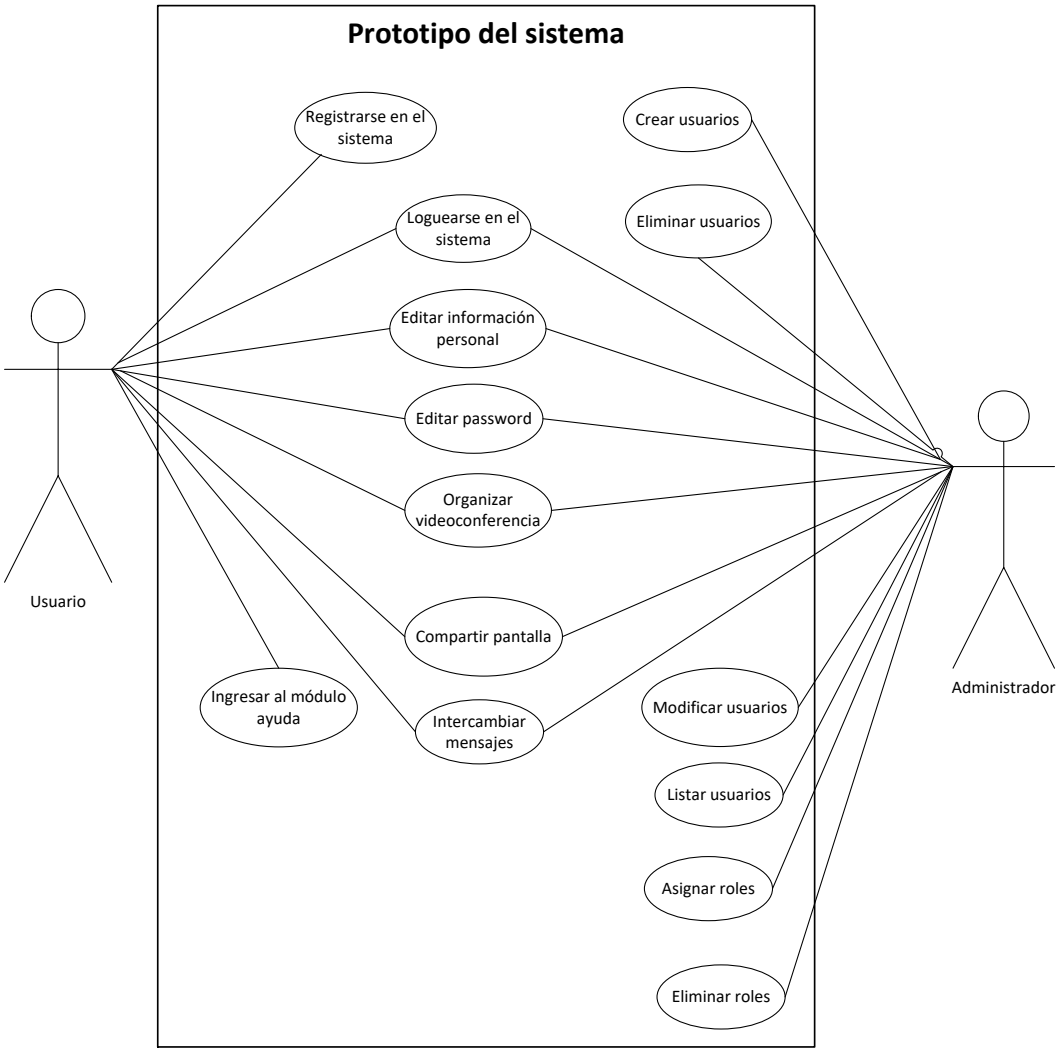

Figura 2.2 Diagrama de casos de uso

2.1.5 Roles y Permisos

Después del análisis del diagrama de casos de uso se han definido los siguientes roles y permisos para los actores del sistema:

2.1.5.1 Roles

Se definen dos roles que interactúan con la aplicación web:

- **Administrador:** actor encargado de gestionar la aplicación web, realiza acciones CRUD de los usuarios, gestiona los roles de éstos, tiene acceso a la comunicación en tiempo real.
- **Usuario:** actor que accede a la aplicación, luego de registrarse o de ser registrado por un administrador y que utiliza las funcionalidades de videoconferencia, compartición de mensajes y de pantalla, además accede a la información de ayuda.

2.1.5.2 Permisos

De acuerdo con los roles planteados se definen dos tipos de permisos:

- **Administrador:** tiene todos los permisos en el sistema, es capaz de realizar todas las acciones CRUD tanto de información de usuarios como de roles, en el resto de los usuarios y administradores, y tiene acceso al módulo de comunicación.
- **Usuario:** los permisos de usuario se limitan a actualización de datos personales y contraseña. Puede acceder a la información de ayuda y al módulo de comunicación.

Un administrador puede asignar rol de administrador a un usuario normal, o a su vez rol de usuario a un administrador. Por lo cual existirán usuarios de la aplicación con dos roles, y con acceso a todas las funciones.

2.1.6 Diagrama Entidad-Relación

En base a los requerimientos de usuario establecidos se elaboró el diagrama Entidad-Relación que se muestra en la Figura 2.3. Este diagrama sirvió para el modelado la base de datos que se utilizó para el almacenamiento de información de usuarios, roles e información de ayuda. El diagrama fue generado en MySQL Workbench[22].

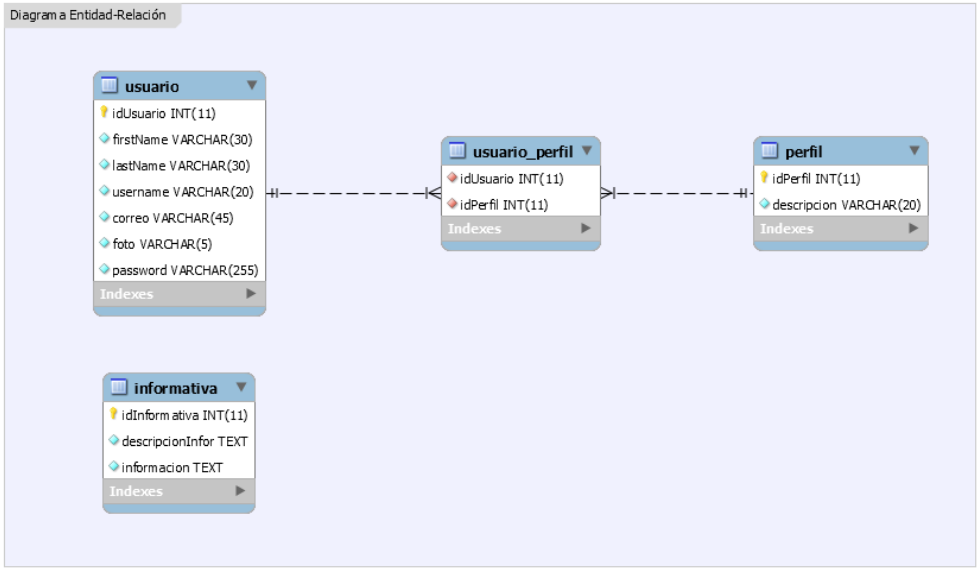

Figura 2.3 Diagrama Entidad-Relación

La tabla `usuario` se encarga de almacenar información de los usuarios en el sistema, cuenta con un `idUsuario`, un identificador único y auto numerado; otros campos tienen la propiedad de ser únicos, y son: `correo` y `username`, para evitar duplicidad de usuarios.

[22] MySQL Workbench: herramienta visual de diseño, administración, gestión y mantenimiento de bases de datos para el sistema de base de datos MySQL.

La tabla `perfil` cuenta con un identificador único denominado `idPerfil`, y el campo descripción que distinguirá entre administrador y usuario.

La tabla `usuario_perfil` describe la relación de muchos a muchos entre la tabla `usuario` y la tabla `perfil`. Finalmente la tabla `informativa` almacena la información de ayuda, el *script* y archivo para la subida de información se encuentra en el ANEXO D.

2.1.7 Diagrama de Clases Servidor Web

De lado del servidor se ofrecen métodos que sirven para reflejar los procedimientos detallados en los casos de uso y también los métodos correspondientes a la interacción con la base de datos.

Cabe mencionar que JavaScript es un lenguaje orientado a objetos basado en prototipos, por lo cual no se notarán clases explícitas, el diagrama que se muestra en la Figura 2.4 detalla la estructura del sistema, mostrando los atributos, operaciones y relaciones entre los objetos.

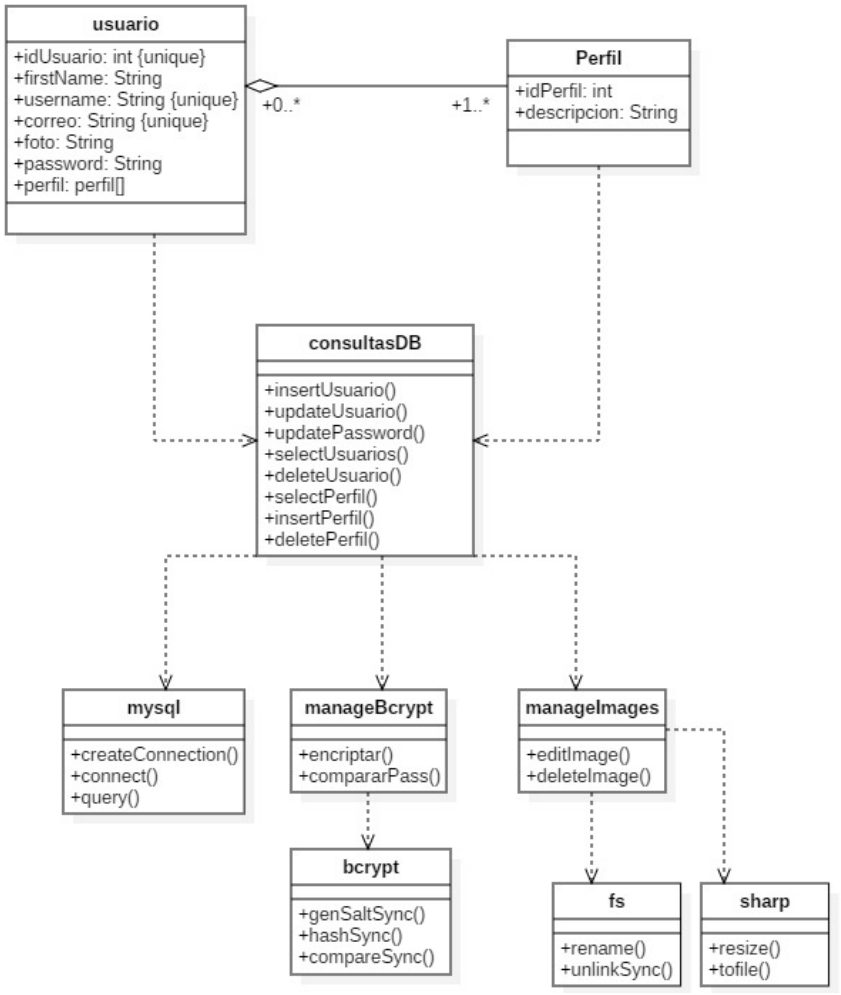

Figura 2.4 Diagrama de clases del sistema

2.1.8 Módulos del Sistema

En base a los requerimientos expuestos, la distribución de roles y los diagramas de estructura del sistema se han definido los siguientes módulos del sistema por su funcionalidad:

a. **Módulo autenticación**: permite validar credenciales de los usuarios para el ingreso a la aplicación y permite el registro de usuarios.

b. **Módulo de administración de usuarios y roles***:* permite crear, editar, eliminar y ver a los usuarios registrados en la aplicación y ver, añadir y eliminar roles de los usuarios registrados.

c. **Módulo de ayuda:** contará con un manual de uso de la aplicación y documentará aspectos importantes de este sistema de videoconferencia basado en la tecnología WebRTC.

d. **Módulo de comunicación:** Permite el uso de WebRTC para realizar videoconferencia, compartición de pantalla y transferencia de mensajes entre los participantes.

2.1.9 Vistas de Usuario y Diseño de Procesos

A continuación, se presenta el diseño de las interfaces de usuario y el modelado de procesos para cada uno de los componentes de los módulos.

Los procesos se modelaron mediante diagramas de actividades, utilizando la herramienta StarUML[23], y muestran los procesos como un flujo de trabajo entre los usuarios y el sistema.

Las vistas que se detallan han sido diseñadas en base a los requerimientos definidos anteriormente, se ha utilizado la herramienta online moqups[24] en su versión gratuita.

Es importante aclarar que las interfaces finales variaron con respecto a las presentadas en esta fase por motivos de presentación de la aplicación.

2.1.9.1 Módulo de Autenticación

- **Inicio de Sesión**

Esta vista permite a los usuarios autenticarse en la aplicación ingresando sus credenciales: correo electrónico y contraseña. Se pueden autenticar los usuarios registrados en la aplicación independiente de su rol, se valida que los campos se encuentren llenos, que el

[23] StarUML: Herramienta de software para la creación de diseños y diagramas UML.
[24] moqcups: Aplicación web simplificada e intuitiva que sirve para crear y colaborar en *wireframes*, maquetas, diagramas y prototipos.

correo electrónico sea válido y se notifica al usuario en caso de infringir alguna de estas reglas. La Figura 2.5 muestra el diseño de la vista de inicio de sesión.

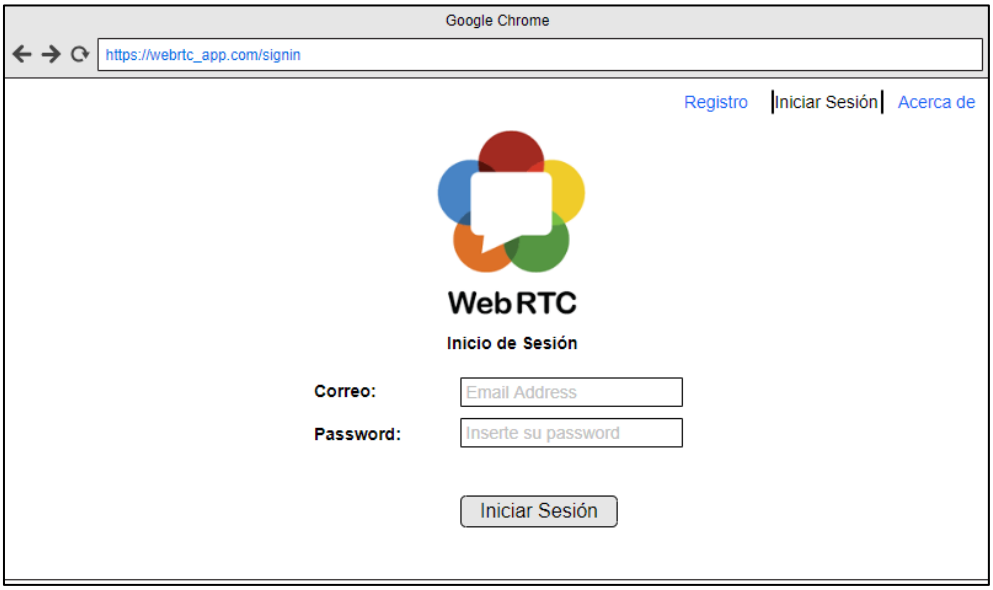

Figura 2.5 Vista de Inicio de Sesión

El diagrama de actividades asociado con la autenticación se muestra en la Figura 2.6.

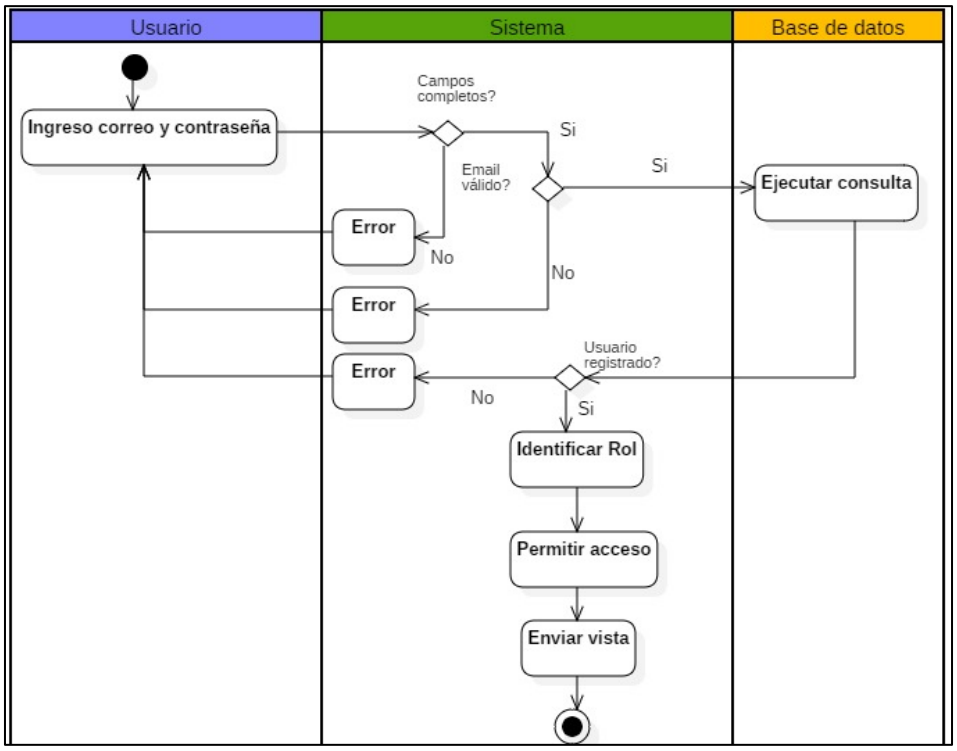

Figura 2.6 Proceso de autenticación

2.1.9.2 Módulo de Administración de Usuarios y Roles

- **Panel de Usuarios**

En la Figura 2.7 se muestra la vista correspondiente al panel para la administración de usuarios, a la cual tendrán acceso usuarios con rol Administrador; en esta vista se lista a todos los usuarios registrados en la aplicación con la información asociada a los mismos, y botones de acción para realizar operaciones de edición y eliminación.

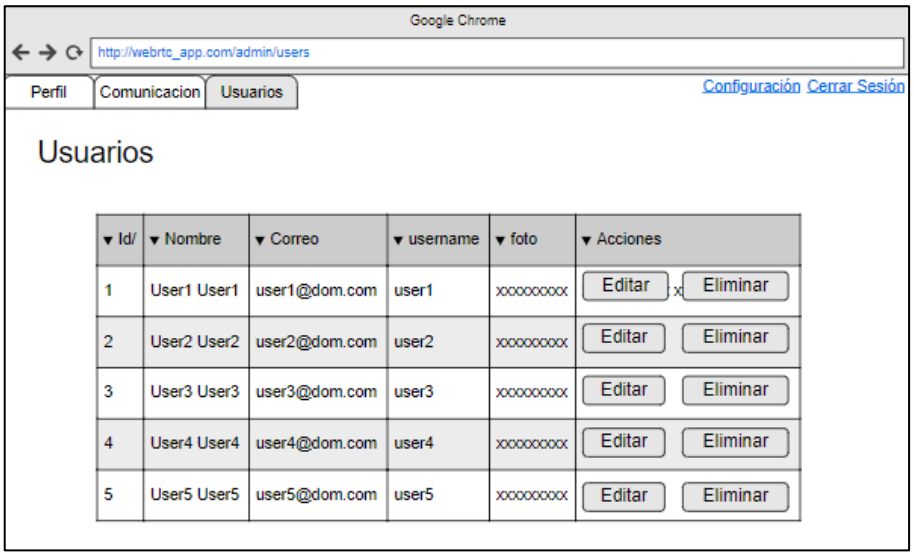

Figura 2.7 Vista del panel de administración de usuarios

- **Creación de Usuario - Registro**

Existen dos maneras de crear usuarios:

1. Un usuario decide registrarse dando clic en el botón ‘Registrarse’ de la página de inicio; en este caso se le asignará el rol usuario automáticamente, la vista asociada se muestra en la Figura 2.8.

2. Un administrador que ha iniciado sesión crea un nuevo usuario desde el panel de administración de usuarios; el administrador que crea un usuario tiene la potestad de asignar el rol o los roles; la vista para creación de usuarios desde un administrador se muestra en la Figura 2.9.

En estas vistas se validará que los campos estén llenos, que se ingrese un correo válido, se pide confirmación de contraseña y se evita la duplicidad de los usuarios por los campos correo y *username* que han sido declarados como únicos en la base de datos.

Figura 2.8 Vista para registro de usuarios

Figura 2.9 Vista para creación de usuarios desde un administrador

Para el proceso de registro o creación de usuarios se ha diseñado el diagrama de actividades que se muestra en la Figura 2.10, que detalla el proceso de creación de usuarios.

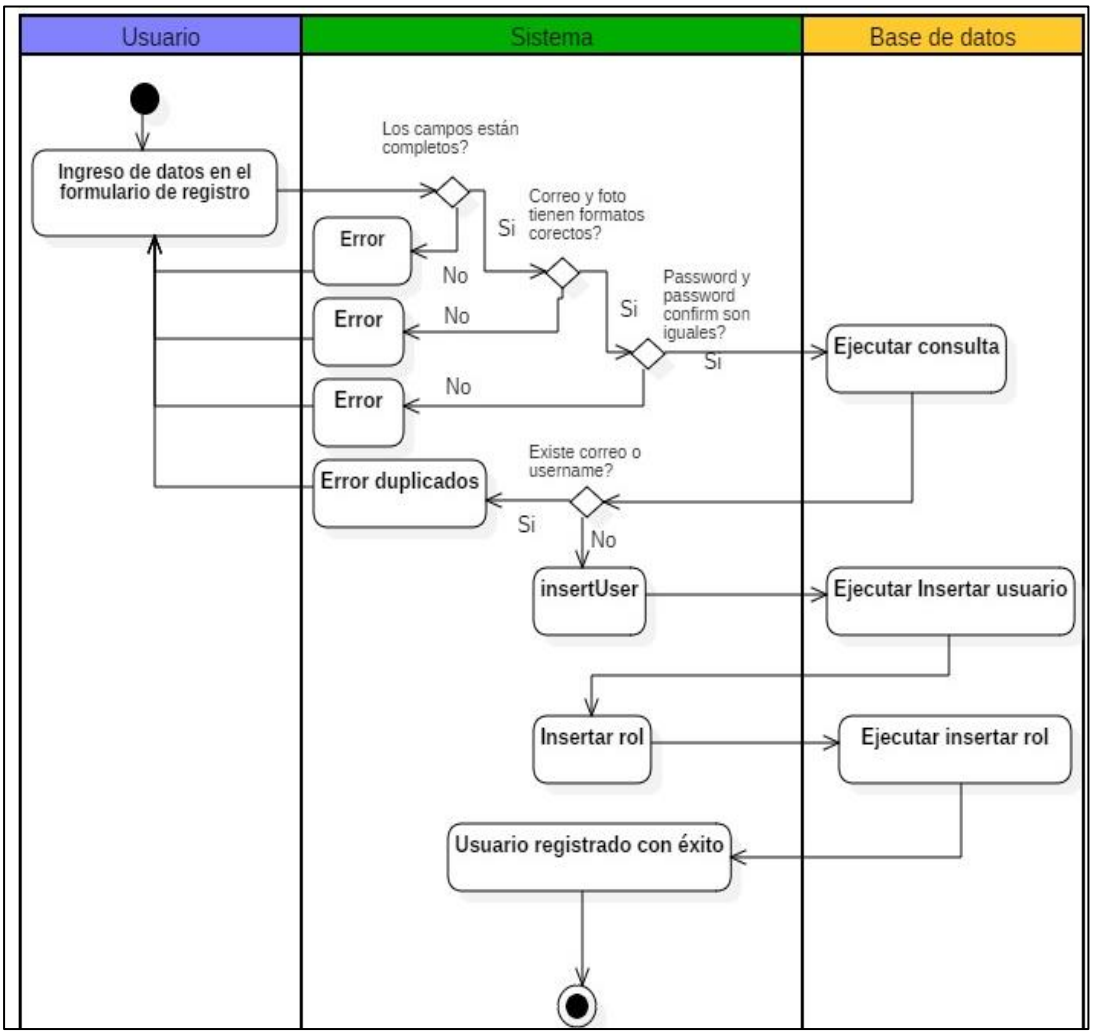

Figura 2.10 Proceso de creación de un usuario

- **Editar Configuración de Usuario**

Con relación a la edición de la información de un usuario se puede mostrar dos vistas: la primera vista para editar información como: nombre, apellido, foto y *username* y se muestra en la Figura 2.11 y la segunda vista será destinada al cambio de contraseña y se muestra en la Figura 2.12.

En la vista de cambio de contraseña, se requiere confirmación de la contraseña nueva.

Los administradores pueden editar únicamente la información personal de otros usuarios, no su contraseña, y accederán a la edición desde el panel de administración de usuarios.

Figura 2.11 Vista para edición de datos personales

Figura 2.12 Vista para cambio de contraseña

En la Figura 2.13 se detalla el proceso de edición de un usuario mediante el diagrama de actividades, en la edición de información de usuario no se puede editar el correo con el que se ha registrado.

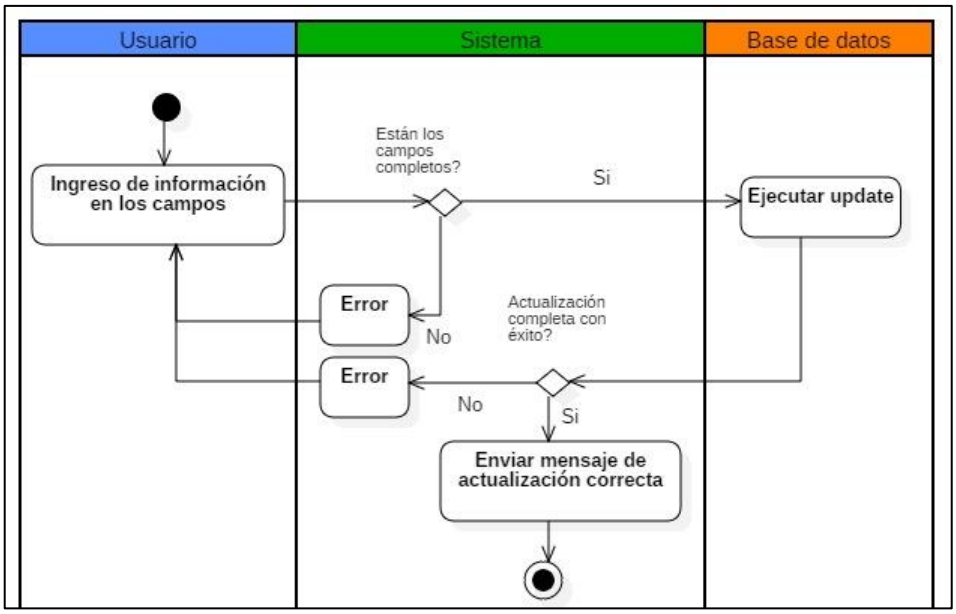

Figura 2.13 Proceso para edición de información de usuario

La Figura 2.14 detalla el proceso de edición de contraseña mediante el correspondiente diagrama de actividades.

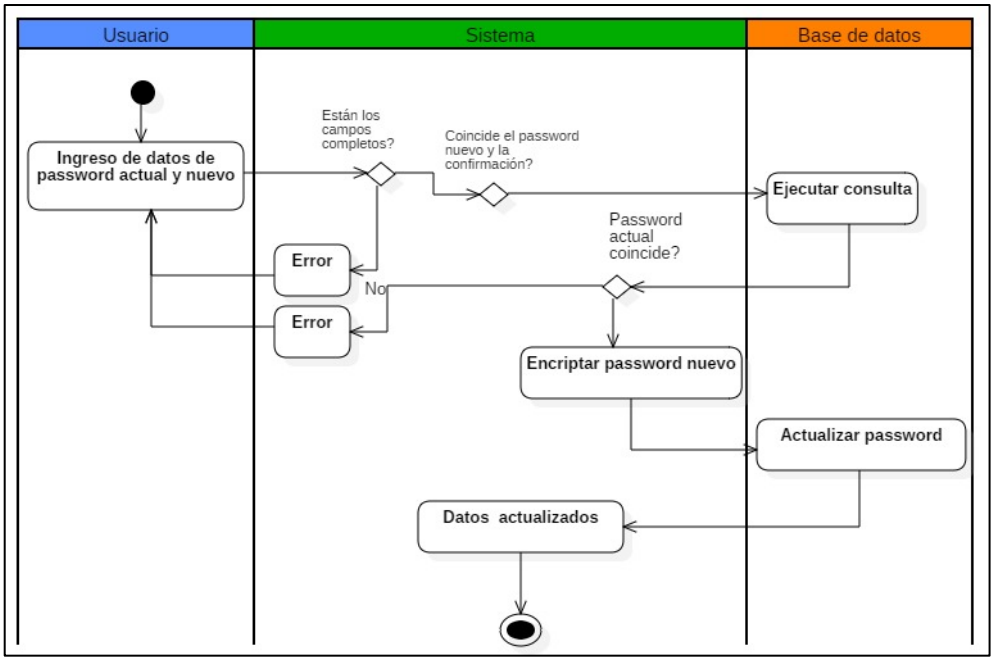

Figura 2.14 Proceso para cambiar contraseña

- **Eliminar Usuario**

La eliminación de usuarios se lleva a cabo desde el panel de administración de usuarios; solo los usuarios tipo administrador pueden eliminar usuarios. Al eliminar un usuario en la

base de datos, también se borrarán los registros asociados a este en la tabla '*usuario_perfil'* debido a que se declaró a la tabla '*usuario*' con la propiedad *'DELETE ON CASCADE'*.

En la Figura 2.15 se muestra la vista del panel de usuarios y la opción confirmación de eliminación luego de presionar el botón 'Eliminar'.

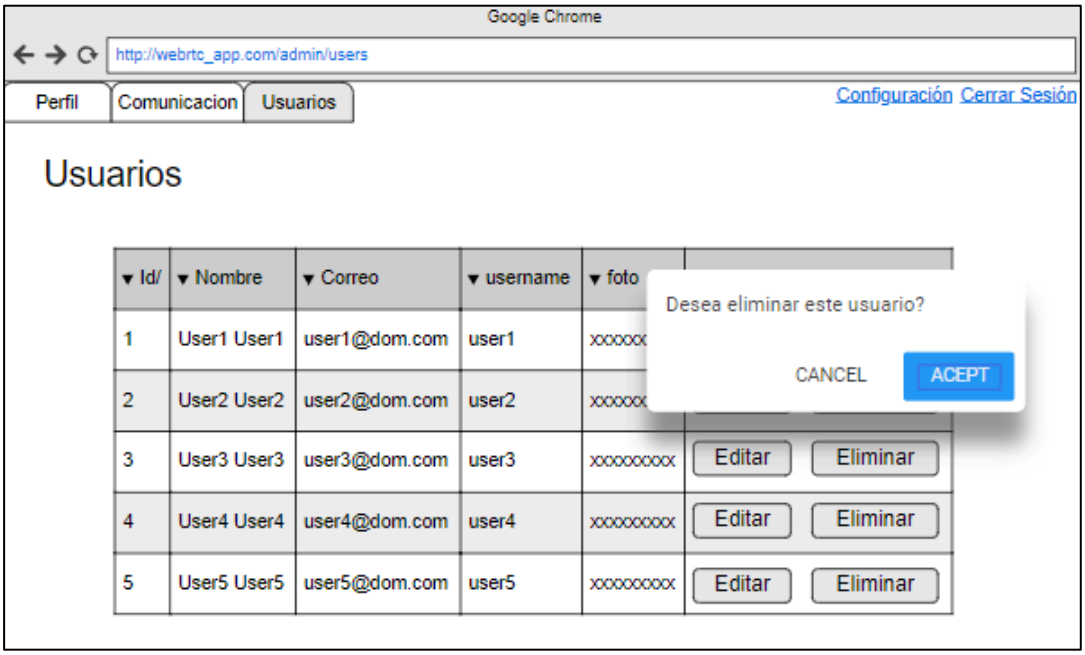

Figura 2.15 Vista para eliminación de un usuario

El proceso asociado a la eliminación de un usuario se detalla en el diagrama de actividades de la Figura 2.16.

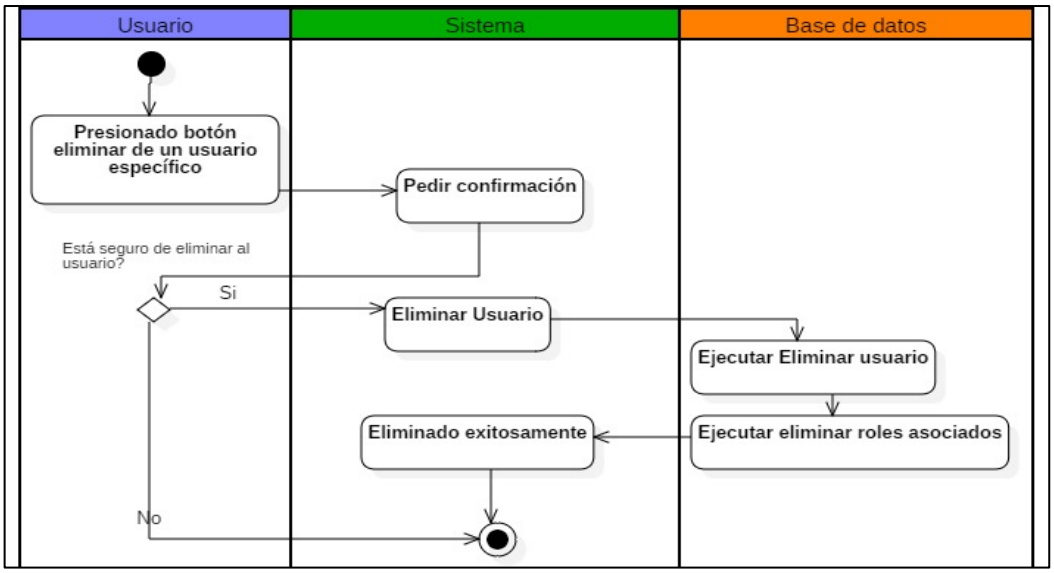

Figura 2.16 Proceso para eliminar un usuario

- **Panel de Administración de Roles**

La vista del panel de administración de roles se muestra en la Figura 2.17. Esta vista estará disponible para usuarios con rol administrador, en ella se lista a los usuarios registrados en la aplicación, los roles asignados al momento y botones de acción para eliminar rol en caso de tener más de uno y de añadir en el caso de tener un solo rol asignado.

Figura 2.17 Vista del panel de administración de roles

- **Añadir Rol**

Para añadir un rol de usuario se accede al panel de administración de roles, solo los usuarios tipo administrador pueden añadir roles. Para añadir un rol, debe existir confirmación por parte del usuario que desea añadir el rol.

En la Figura 2.18 se muestra la vista del panel de administración de roles y la opción de confirmación para añadir un rol.

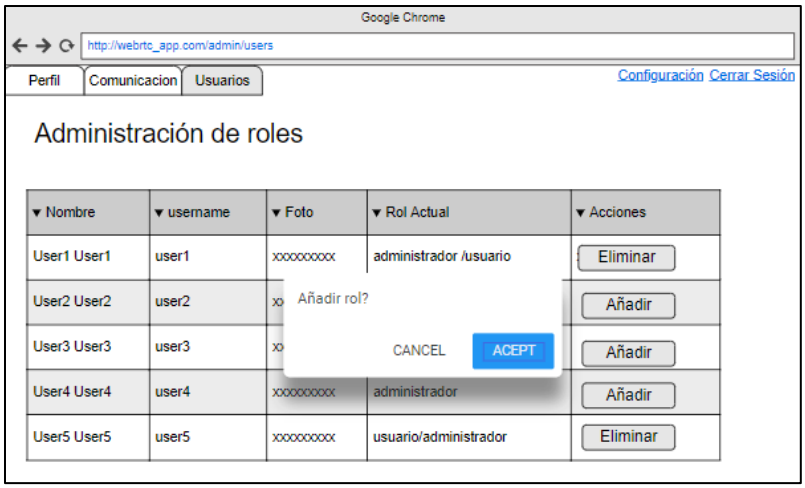

Figura 2.18 Vista para agregar un rol

El proceso para añadir un nuevo rol de usuario se detalla en el diagrama de actividades de la Figura 2.19.

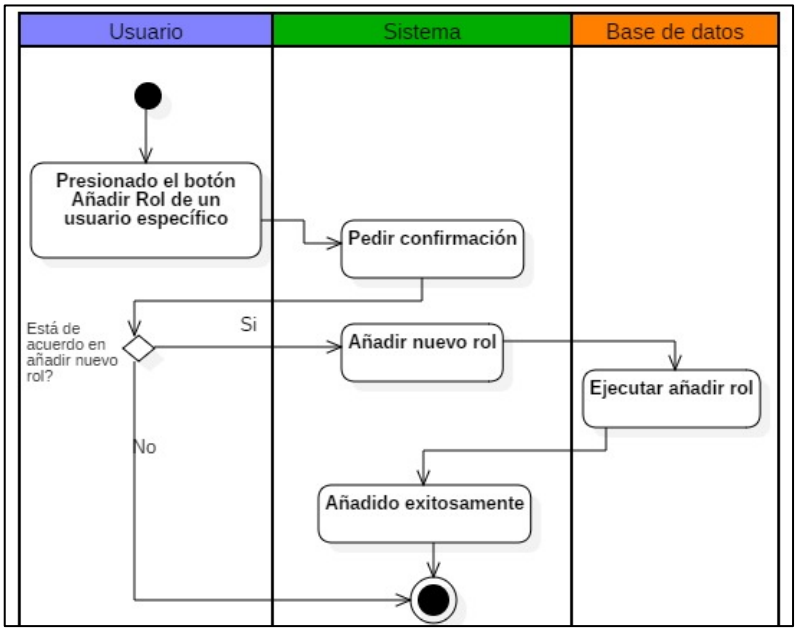

Figura 2.19 Proceso para eliminación de un rol

- **Eliminar Rol**

La vista correspondiente a la eliminación de un rol se detalla en la Figura 2.20. La acción de eliminar rol solo la pueden llevar a cabo administradores. Se accede a esta vista luego de dar clic en el botón 'Eliminar' en el panel de administración de roles.

Si un usuario posee dos roles sólo se puede eliminar uno. No se pueden eliminar todos los roles de un usuario.

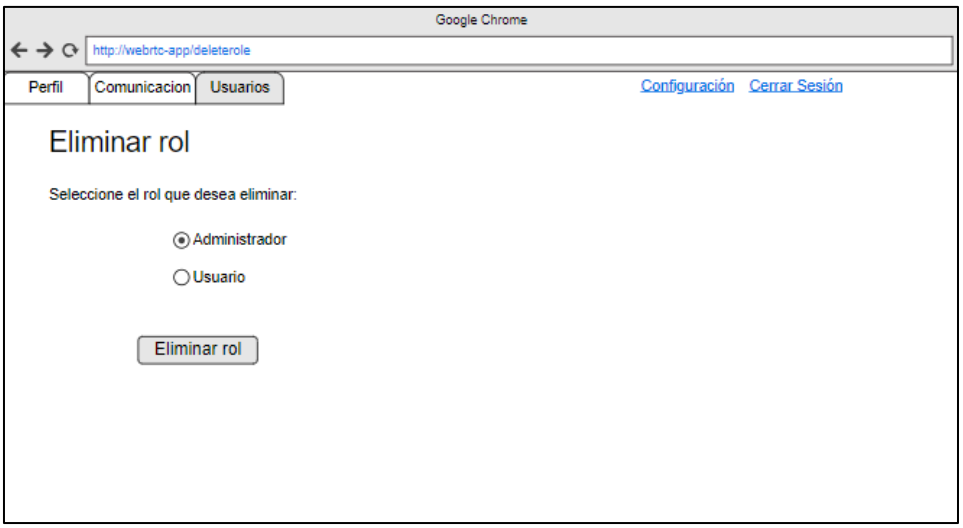

Figura 2.20 Vista para eliminación de un rol

En la Figura 2.21 se detalla el proceso de eliminación de un rol de usuario.

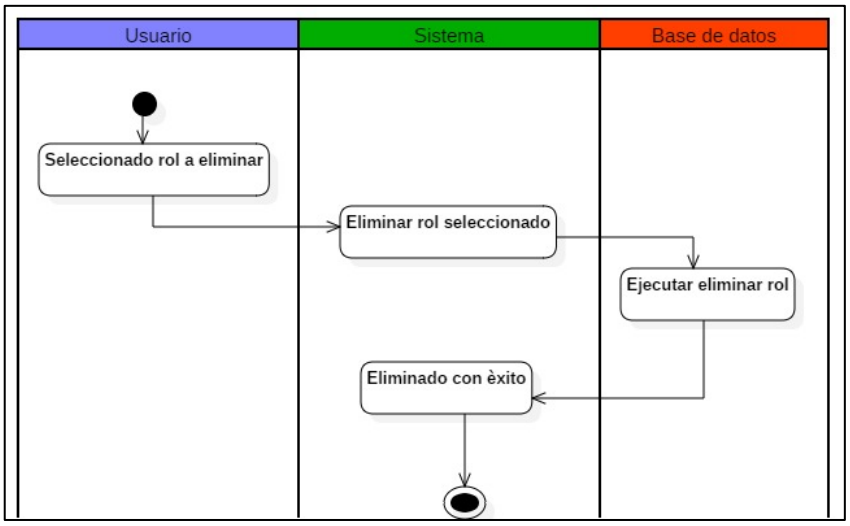

Figura 2.21 Proceso para la eliminación de un rol

2.1.9.3 Módulo de Ayuda

El módulo de ayuda será accesible para los usuarios de la aplicación que cuenten con el rol de usuario. Este módulo se divide en cuatro partes secundarias, que muestran información relevante del desarrollo de la aplicación y su uso.

Las partes del módulo ayuda son: diagrama del prototipo, características de aplicación (videoconferencia, compartición de pantalla y chat), creación de cuenta, roles de usuario, y manual de uso. Las cuatro vistas del módulo información se detallan en la Figura 2.22.

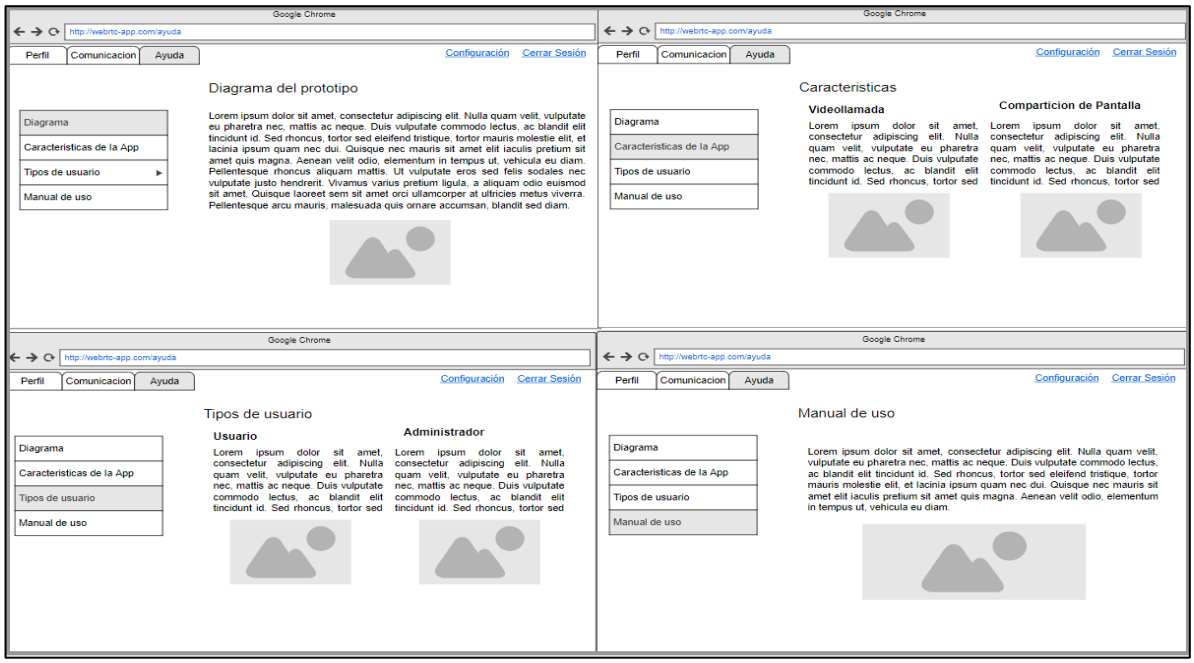

Figura 2.22 Vistas del módulo de ayuda

El proceso de obtención de las vistas para el módulo de ayuda se muestra en la Figura 2.23, se muestra un solo proceso dado que la obtención es similar.

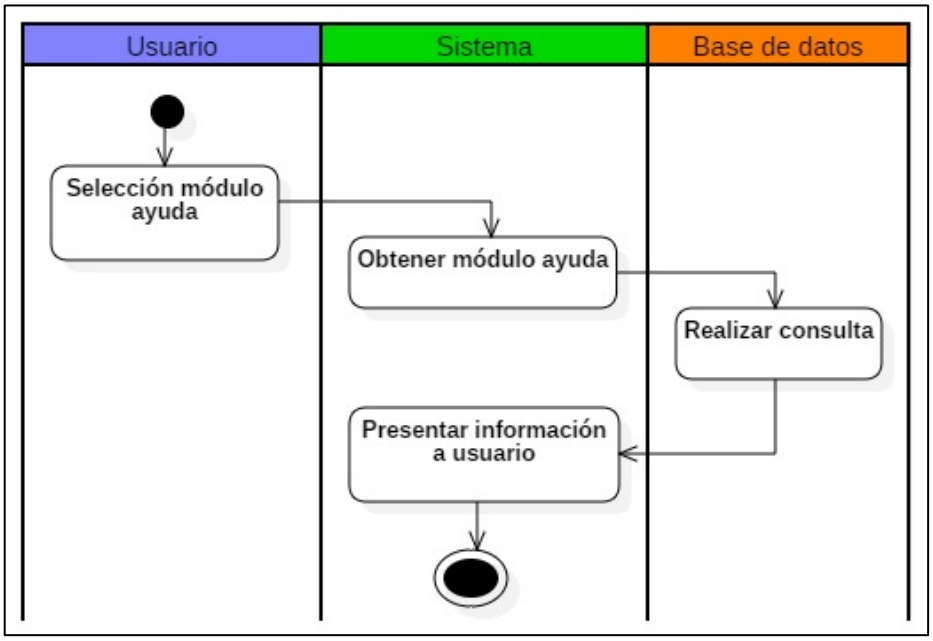

Figura 2.23 Proceso de obtención de ayuda para el usuario

2.1.9.4 Módulo de Comunicación

La vista del módulo de comunicación se presenta en la Figura 2.24.

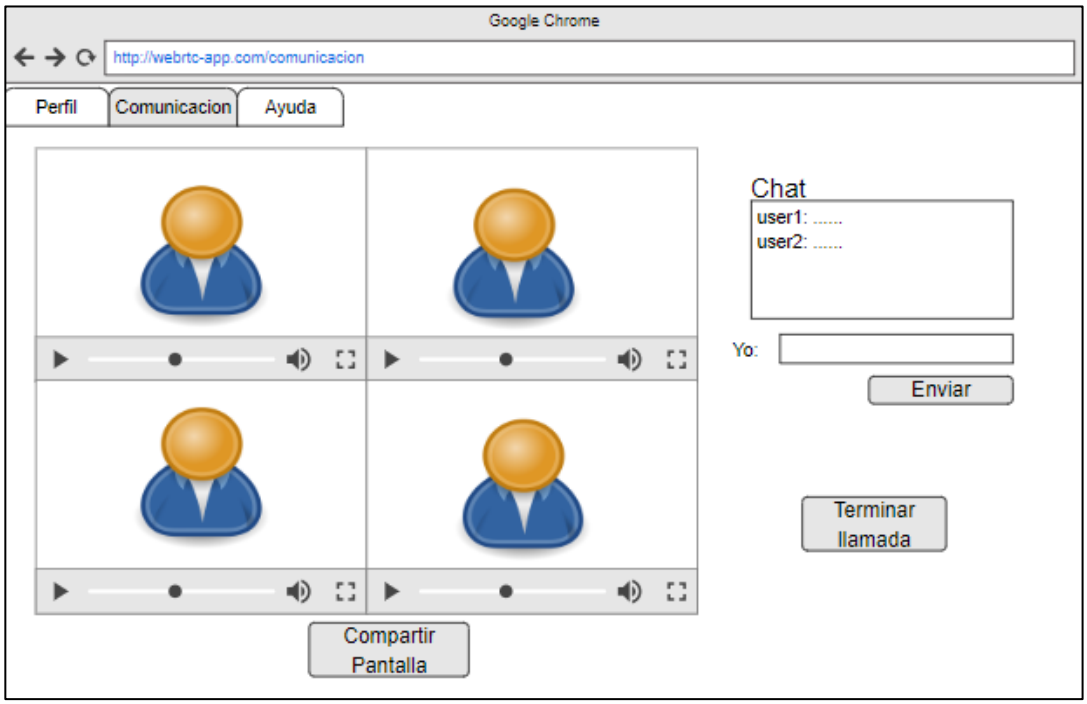

Figura 2.24 Vista del panel de comunicación

Los *scripts* asociados a la comunicación en tiempo real con el uso de las APIs de WebRTC para la videoconferencia, compartición de pantalla y chat se ejecutan en el lado del cliente, en el navegador, el servidor web únicamente sirve estos archivos.

Antes de iniciar la comunicación, se controla que exista un servidor de señalización disponible para realizar la videoconferencia. Si no se tiene un servidor de señalización no se le permite al usuario iniciar la comunicación.

Cuando un usuario decide iniciar videoconferencia se crea un cuarto o canal para compartir información, al cual otros usuarios podrán acceder mediante la URI designada para ese cuarto. Si un usuario decide ser organizador de videoconferencia deberá compartir el enlace del cuarto con los usuarios que desee se unan a esta videoconferencia.

Los usuarios que reciban el enlace del organizador podrán por su parte aceptar unirse a la misma o simplemente ignorar la invitación y navegar normalmente por la aplicación.

El organizador de la videoconferencia es el único que podrá utilizar la opción de compartir la pantalla con los invitados que se unan a la videoconferencia, y lo realizará mediante un proceso propio de WebRTC llamado renegociación[25]. El organizador decide alternar entre la compartición de su pantalla y la compartición de video vía cámara web. Los invitados a la videoconferencia o participantes secundarios únicamente podrán compartir video y mensajes con todos los participantes de la videoconferencia.

Para la compartición de pantalla se debe utilizar una extensión de Google Chrome para capturar el contenido de la pantalla, además permite determinar si esta extensión está instalada o no para poder notificar al usuario y proporcionarle el enlace para descargarla.

En la Figura 2.25 se presenta el diagrama de actividades que describe de manera general el funcionamiento de la comunicación para el organizador de la videoconferencia, suponiendo que es un usuario registrado, que ha iniciado sesión, ingresó al módulo de comunicación y tuvo disponible el servidor de señalización.

La Figura 2.26 muestra el proceso general de comunicación para los usuarios que se unirán a partir del enlace enviado por un organizador, suponiendo que el usuario es un usuario registrado y que ha iniciado sesión.

[25] Renegociación: Se refiere a usar conexiones entre pares creadas previamente para añadir o modificar *streams* mediante la renegociación de las descripciones de las sesiones (Oferta/Respuesta SDP).

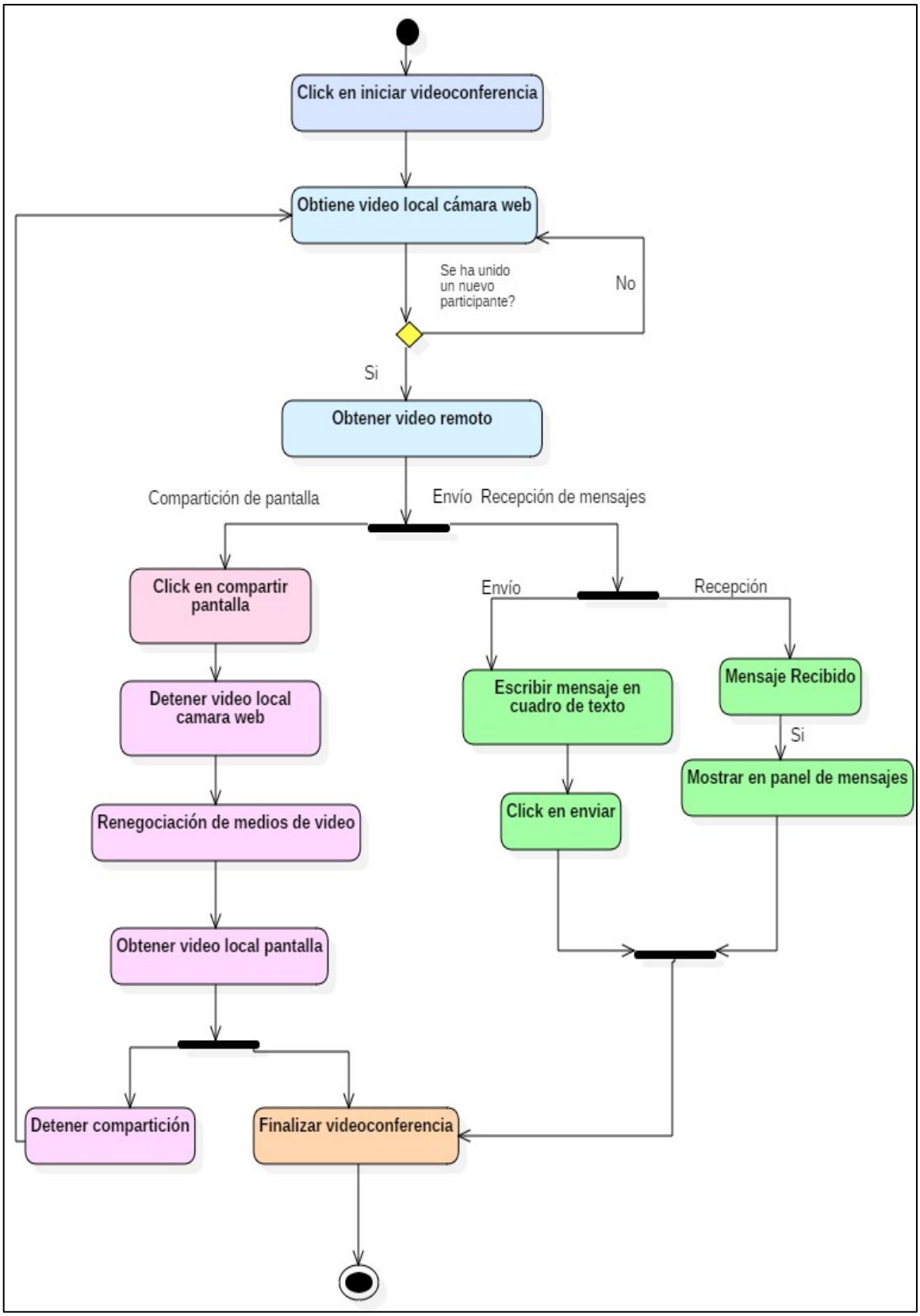

Figura 2.25 Proceso de comunicación - Organizador

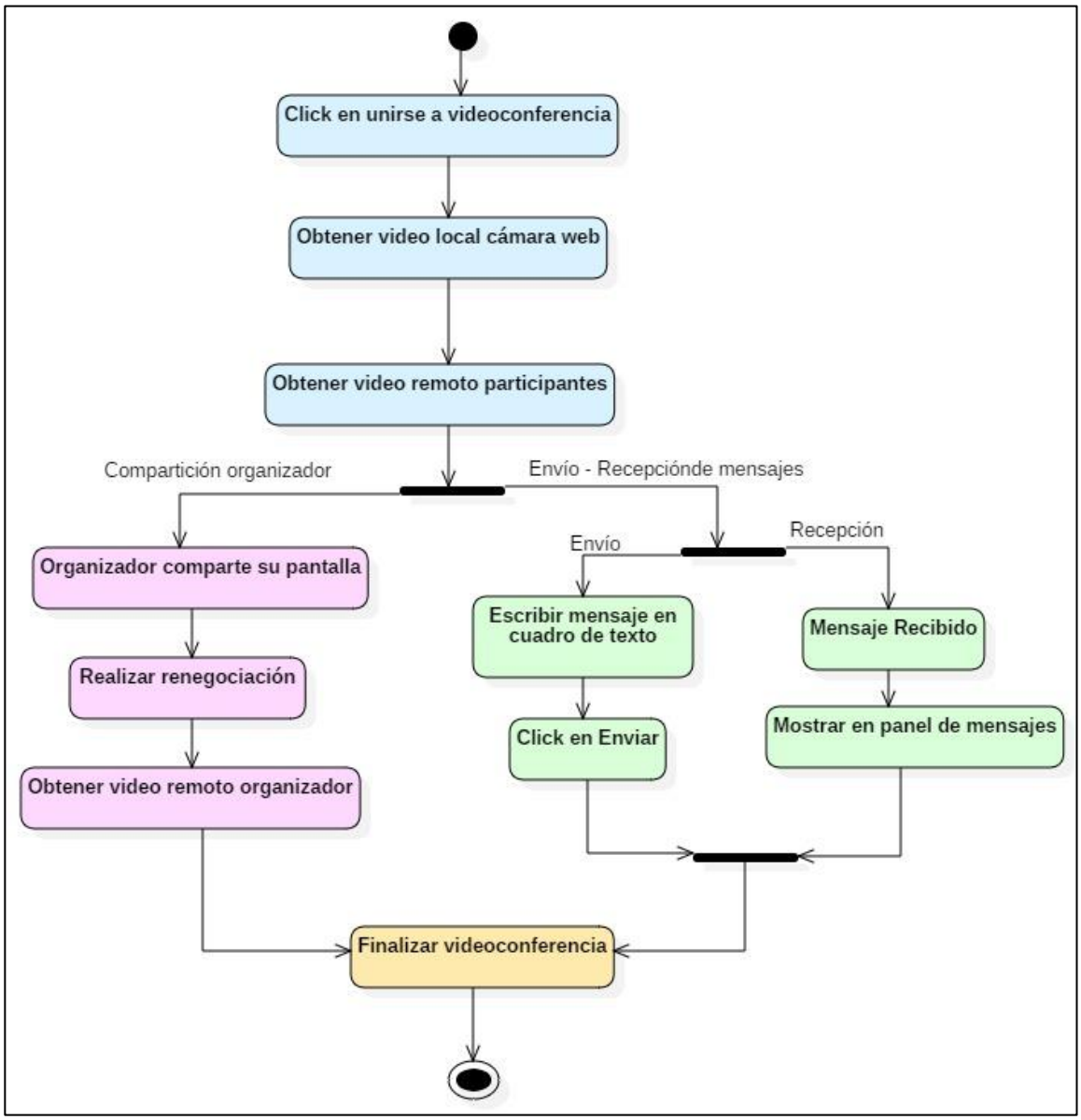

Figura 2.26 Proceso de comunicación – Usuario secundario

Para el módulo de comunicación también se definieron diagramas de secuencia que detallaron de mejor manera la interacción entre pares y el proceso de renegociación para la compartición de pantalla involucrando terminología de las APIs de WebRTC.

En la Figura 2.27 se muestra el diagrama de secuencia que involucra una comunicación entre dos pares, un *peer* organizador y un *peer* invitado y su relación con el servidor de señalización.

En la Figura 2.28 se muestra el diagrama de secuencia de la renegociación entre dos pares, al momento que un organizador decide compartir pantalla con sus invitados.

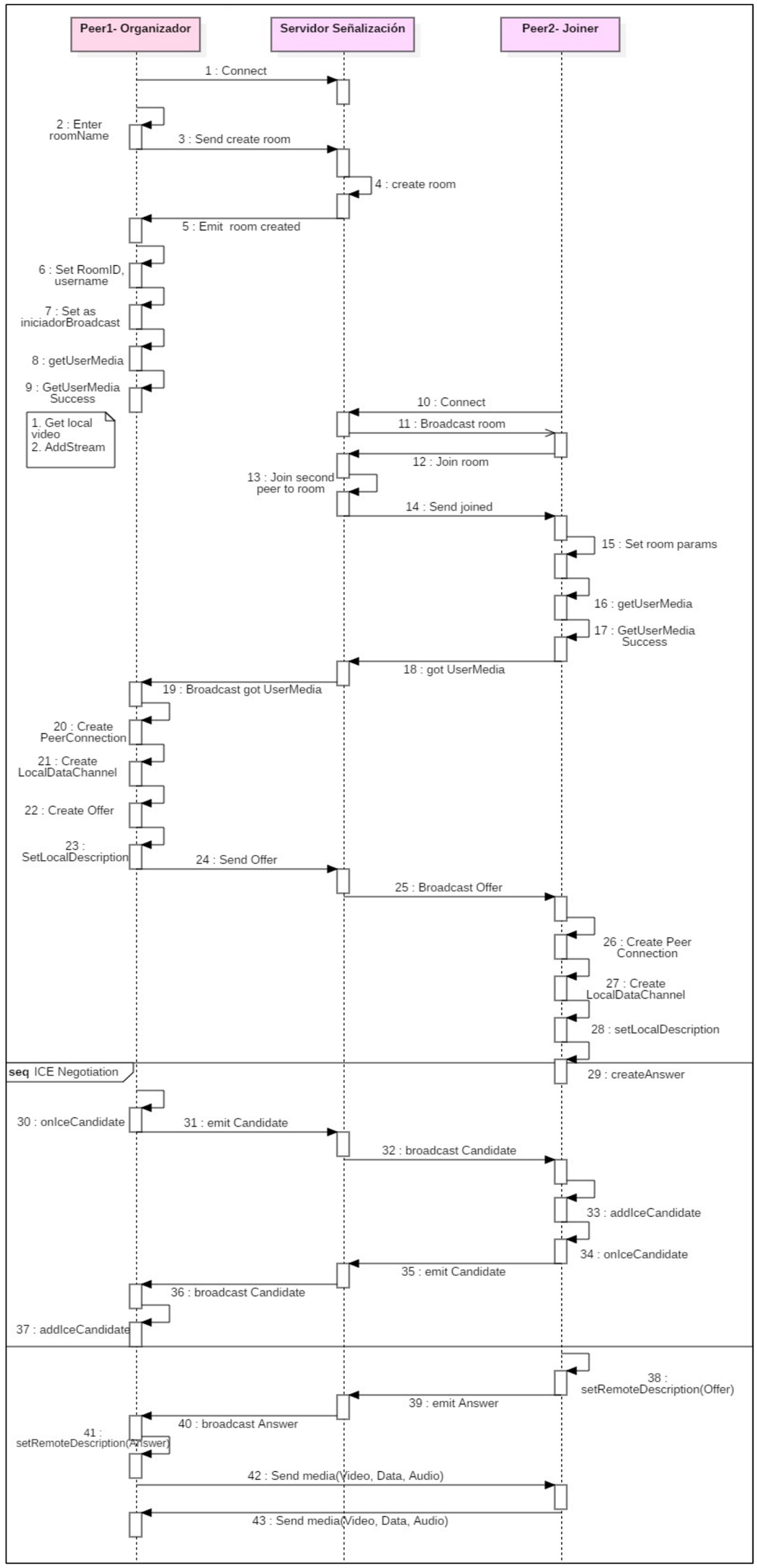

Figura 2.27 Diagrama de secuencia asociada a la comunicación entre dos pares

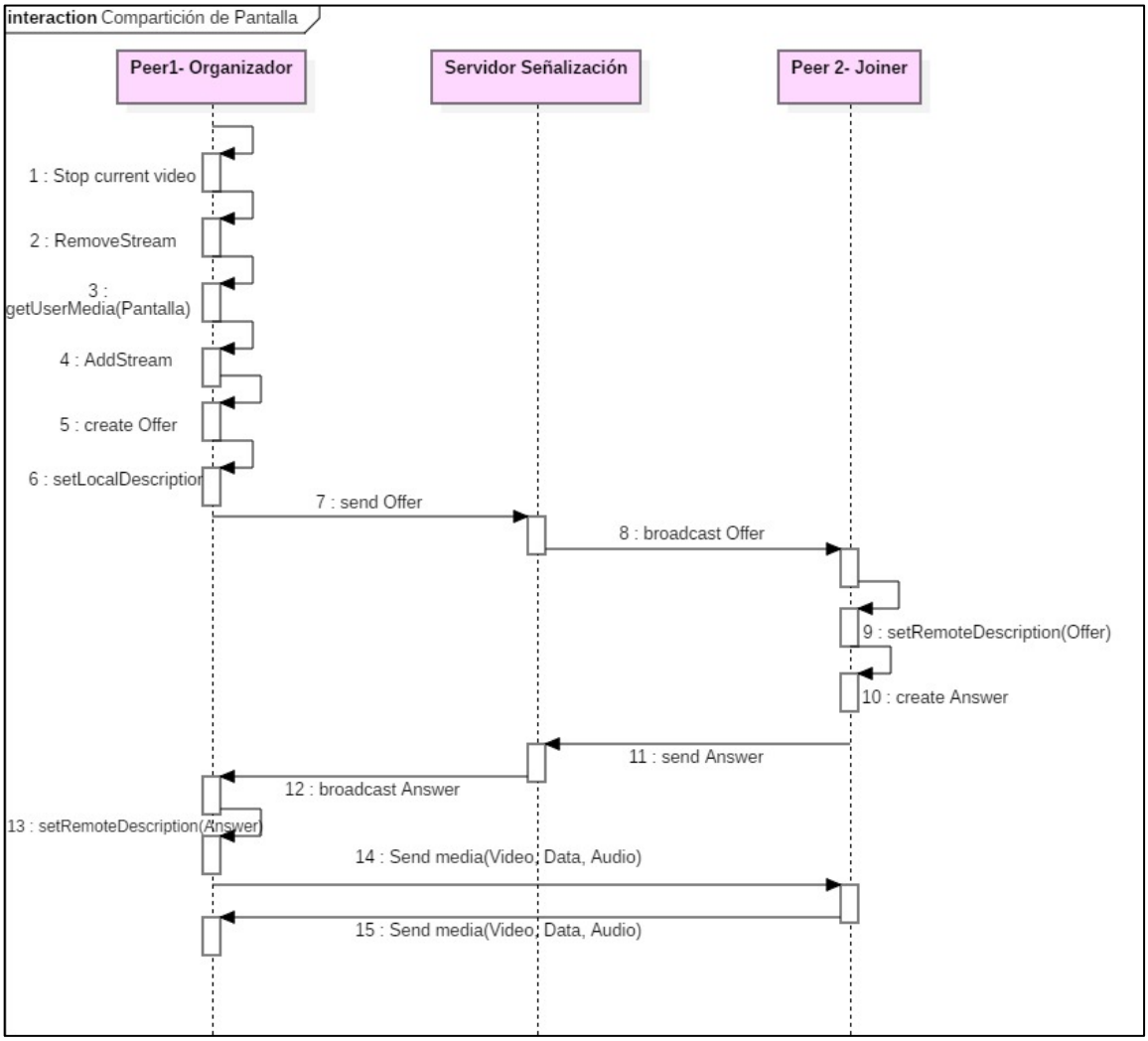

Figura 2.28 Diagrama de secuencia: renegociación para compartir la pantalla

2.1.10 Historias de Usuario

Las historias de usuario son utilizadas para la descripción de requisitos de software, están clasificadas de acuerdo con los módulos definidos y fueron útiles para la definición del *product backlog*.

2.1.10.1 Formato

El formato de las historias de usuario (Tabla 2.1) contiene los siguientes campos:

- *ID: identificación* única de la historia de usuario.
- *Rol:* indica el usuario al que estará dirigida la funcionalidad.
- *Nombre de la historia de usuario:* título descriptivo de la historia de usuario
- *Prioridad:* nivel de importancia de la historia de usuario, se han clasificado en: Alta, Media y Baja.
- *Descripción:* se describe los objetivos de la historia de usuario.

- *Criterios de aceptación:* indica la manera en que el usuario confirma que se ha cumplido el objetivo de la historia de usuario.

Tabla 2.1 Formato de historia de usuario

ID		**Rol**		**Prioridad**	
Nombre HU:					
Descripción:					
Criterios de Aceptación:					

2.1.10.2 Detalle de Historias de Usuario

Se presentan las historias de usuario definidas según los requerimientos analizados, están clasificadas según los módulos que se definieron en la sección de diseño.

- **Módulo de Administración de Usuarios y Roles:**

En las Tabla 2.2, Tabla **2.3**, Tabla **2.4** y Tabla **2.5** se detallan las historias de usuario correspondientes al módulo de Administración de usuarios y de roles en la aplicación.

Tabla 2.2 HU de Administración usuarios

ID	**HU-01**	**Rol**	**Administrador**	**Prioridad**	**Alta**
Nombre HU: Administración de usuarios					
Descripción: El administrador podrá realizar las acciones de: - Crear usuarios con roles: administrador, usuario, ambos - Modificar información de usuarios - Leer información de usuarios - Eliminar usuarios					
Criterios de Aceptación: **La aplicación permite al administrador realizar un CRUD de usuarios.**					

Tabla 2.3 HU Registro de usuarios

ID	HU-02	Rol	Usuario	Prioridad	Alta
Nombre HU: Registro de usuarios					
Descripción: Un usuario puede registrarse en la aplicación mediante un formulario de registro que contendrá: - Nombre - Apellido - Username - Correo - Foto - Contraseña de ingreso a la aplicación Se asignará el rol usuario a quien se registre por este medio					
Criterios de Aceptación: **La aplicación permite a un usuario nuevo registrarse en la aplicación con rol usuario.**					

Tabla 2.4 HU Edición de información personal y contraseña

ID	HU-03	Rol	Usuario/Administrador	Prioridad	Media
Nombre HU: Editar información personal y contraseña					
Descripción: Un usuario puede editar su información personal, excepto correo y cambiar su contraseña de ingreso a la aplicación					
Criterios de Aceptación: **La aplicación permite a un usuario editar su información personal y cambiar su contraseña.**					

Tabla 2.5 HU Administración de roles

ID	HU-04	Rol	Administrador	Prioridad	Media
Nombre HU: Administración de roles					
Descripción: El administrador puede: - Visualizar roles - Añadir rol - Eliminar rol					
Criterios de Aceptación: **La aplicación permite a un administrador añadir o eliminar un rol a un usuario específico desde el panel de administración de roles.**					

- **Módulo de Autenticación**

Las Tabla 2.6 y

Tabla 2.7 corresponden a las historias de usuario definidas para el Módulo de Autenticación.

Tabla 2.6 HU Autenticación en la aplicación

ID	HU-05	Rol	Administrador/ Usuario	Prioridad	Alta
Nombre HU: Autenticación en la aplicación					
Descripción: Administradores y usuarios deben ingresar a la aplicación mediante un login; para la autenticación se requiere: - Correo - Contraseña					
Criterios de Aceptación: **Tanto administradores como usuarios pueden acceder a la aplicación después de autenticarse.**					

Tabla 2.7 HU Mostrar perfil de usuario

ID	HU-06	Rol	Usuario/Administrador	Prioridad	Media
Nombre HU: Mostrar perfil de usuario					
Descripción: Después de haberse autenticado el usuario/administrador es redirigido a su perfil, que contiene su información: - Nombres - Correo - Roles - Username - Foto					
Criterios de Aceptación: **La aplicación permite a los usuarios visualizar su perfil luego de autenticarse.**					

- **Módulo de Comunicación**

Las Tabla 2.8, Tabla 2.9 y Tabla 2.10 corresponden a las historias de usuario del Módulo de Comunicación.

Tabla 2.8 HU Videoconferencia

ID	HU-07	Rol	Usuario/ Administrador	Prioridad	Alta
Nombre HU: Videoconferencia					
Descripción: El usuario podrá realizar videoconferencia con hasta tres participantes adicionales, registrados en la aplicación.					
Criterios de Aceptación: **La aplicación permite realizar videoconferencia hasta con cuatro participantes**					

Tabla 2.9 HU Compartición de pantalla

ID	HU-08	Rol	Usuario/Administrador	Prioridad	Alta
Nombre HU: Compartición de pantalla					
Descripción: El usuario organizador de la videoconferencia podrá compartir su pantalla con el resto de los participantes (máximo cuatro participantes por videoconferencia)					
Criterios de Aceptación: **La aplicación permite a un usuario organizador de la videoconferencia compartir su pantalla con el resto de los participantes.**					

Tabla 2.10 HU Transferencia de mensajes

ID	HU-09	Rol	Usuario/Administrador	Prioridad	Alta
Nombre HU: Transferencia de mensajes					
Descripción: Los usuarios podrán intercambiar mensajes de texto con el resto de las participantes					
Criterios de Aceptación: **La aplicación permite a los usuarios intercambiar mensajes con el resto de las participantes.**					

- **Módulo de Ayuda**

En la Tabla 2.11 se muestra la historia de usuario correspondiente a el módulo de ayuda de la aplicación.

Tabla 2.11 HU Información de desarrollo y manual de usuario

ID	HU-10	Rol	Usuario	Prioridad	Alta
Nombre HU: Información de desarrollo de la aplicación y manual de usuario					
Descripción: El usuario podrá ingresar al módulo informativo, donde encontrará documentado características relevantes de esta aplicación de videoconferencia utilizando WebRTC además de un pequeño manual de uso.					
Criterios de Aceptación: **La aplicación permite a los usuarios ingresar al módulo de ayuda y visualizar la información de las características de la aplicación y un manual de usuario.**					

- **Historias de Usuario Complementarias**

La historia de usuario complementaria corresponde a la página de inicio de la aplicación y la página *Acerca de* y se muestra en la Tabla 2.12.

Tabla 2.12 HU *Home Page* y *Acerca de*

ID	HU-11	Rol	Usuario/Administrador	Prioridad	Baja
Nombre HU: Home Page/ Acerca De					
Descripción: Se mostrará una página de inicio de la aplicación. Se visualizará información acerca de la aplicación.					
Criterios de Aceptación: **La aplicación permite a los usuarios observar una página de inicio.** **La aplicación muestra información de la aplicación en la pestaña 'Acerca de'.**					

2.1.11 Product Backlog

Después de haber definido los requerimientos se procedió a listar los ítems del *product backlog* que se requieren para la implementación del sistema, utilizando las historias de usuario definidas y cumpliendo de esta manera los lineamientos de la metodología Scrum.

En la Tabla 2.13 se detalla el *product backlog* utilizado para el desarrollo de la aplicación.

Tabla 2.13 *Product backlog*

ID	Prioridad	Historia de Usuario	Título historia de usuario
1	**Alta**	HU-00	Creación Base de datos, instalación de IDE de desarrollo.
		HU-01	Administración de usuarios
		HU-02	Registro de usuarios en la aplicación
		HU-05	Autenticación en la aplicación
2	**Alta**	HU-07	Videoconferencia
		HU-08	Compartición de pantalla
		HU-09	Transferencia de mensajes
3	**Alta**	HU-10	Información de desarrollo de la aplicación y manual de usuario
	Media	HU-04	Administración de roles
4	**Media**	HU-03	Edición de información personal y contraseña
		HU-06	Mostrar perfil de usuario
	Baja	HU-11	Home Page/ Acerca De

Tomando en cuenta el *product backlog* definido, se determinó que se requieren cuatro *sprints* para desarrollar el sistema. En la siguiente sección se detallan las tareas realizadas en cada uno de los *sprints*.

2.2 Codificación del Prototipo

En esta sección se detallan la planificación y el desarrollo de cada uno de los *sprints* definidos.

2.2.1 Sprint 1

2.2.1.1 Sprint Planning

En la Tabla 2.14 se presentan los ítems del *product backlog* seleccionados para cumplir con el objetivo del *sprint* 1.

Tabla 2.14 *Sprint backlog 1*

	Historia de usuario	Tareas	Esfuerzo	Horas
1	Creación Base de datos, instalación de IDE de desarrollo.	Instalar IDE de desarrollo WebStorm.	Medio	3
		Instalar MySQL y MySQL WorkBench.	Medio	2
		Construir base de datos con sus tablas, atributos y relaciones.	Alto	3
		Instalar Node.js y npm.	Medio	2
		Crear un nuevo proyecto Node Express en WebStorm.	Alto	5
2	Administración de Usuarios	Crear nuevo usuario (vista y métodos).	Alto	5
		Ver a los usuarios registrados en la aplicación (vista y métodos).	Alto	5
		Editar información de un usuario(vista y métodos).	Alto	5
		Eliminar un usuario (vista y métodos).	Alto	5
3	Registro de usuarios en la aplicación	Registrar a un usuario en la aplicación(vista y métodos).	Alto	5
4	Autenticación en la aplicación	Crear vista para inicio de sesión.	Alto	5
		Crear métodos necesarios para autenticarse en la aplicación.	Alto	5

2.2.1.2 Desarrollo del Sprint 1

- **Base de Datos**

Se utilizó MySQL 8.0 como motor de la base de datos, y MySQL Workbench para generar y administrar la base de datos de manera más visual.

A continuación, se muestra un ejemplo de la creación de una de las tablas de la base de datos, el resto de ellas se crearon de manera similar; todos los *scripts* referentes a la creación de la base de datos, tablas, atributos y algunos valores iniciales requeridos se encuentran en el ANEXO B**.**

El Código 2.1 muestra el *script* para la creación de la tabla `usuario`, mediante la sentencia `CREATE TABLE`, como atributos de esta tabla se han definido la columna `idUsuario` como clave primaria, entero auto incremental y no nulo, las columnas `firstName`, `lastName`, `username`, `correo`, `foto` y `password` como columnas de tipo *varchar* de diferentes tamaños para su almacenamiento, adicionalmente las columnas `correo`, `username` e `idUsuario` han sido definidos como únicas para evitar duplicidad en los usuarios que se registren.

```
1.  CREATE TABLE IF NOT EXISTS `db_webrtc`.`usuario` (
2.    `idUsuario` INT NOT NULL AUTO_INCREMENT,
3.    `firstName` VARCHAR(30) NOT NULL,
4.    `lastName` VARCHAR(30) NOT NULL,
5.    `username` VARCHAR(20) NOT NULL,
6.    `correo` VARCHAR(45) NOT NULL,
7.    `foto` VARCHAR(5) NOT NULL,
8.    `password` VARCHAR(255) NOT NULL,
9.    PRIMARY KEY (`idUsuario`),
10.   UNIQUE INDEX `Username_UNIQUE` (`username` ASC),
11.   UNIQUE INDEX `Correo_UNIQUE` (`correo` ASC),
12.   UNIQUE INDEX `idUsuario_UNIQUE` (`idUsuario` ASC))
```

Código 2.1 *Script* para la creación de la tabla `'usuario'`

- **Preparación del Entorno de Desarrollo**

El IDE instalado y utilizado para el desarrollo de los módulos del sistema es WebStorm 2017.3.

- **Instalación de Node.js – npm**

La descarga de Node.js se la realiza directamente desde su página web: https://nodejs.org/es/download/, para el desarrollo de esta aplicación se descargó Node.js para Windows y se usó el asistente de instalación.

La versión de Node.js instalada para el desarrollo de la aplicación es la versión 8.11.1 y la versión del gestor de paquetes npm es la 5.6.0. como se muestra en la Figura 2.29.

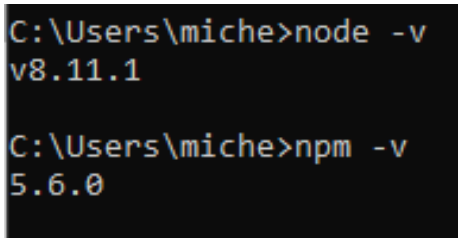

Figura 2.29 Versiones de Node.js y npm instaladas

- **Creación de aplicación web usando Node.js y Express**

En esta sección se explica los pasos seguidos para crear una aplicación web usando Node.js con *Express* y las configuraciones iniciales para el desarrollo de la aplicación.

Para crear un proyecto usando Node.js y *Express* se siguen los siguientes pasos (Figura 2.30):

1. En la barra de herramientas seleccionar: *File → New → Project*
2. Seleccionar: Node.js Express App. Presionar: *Create*.

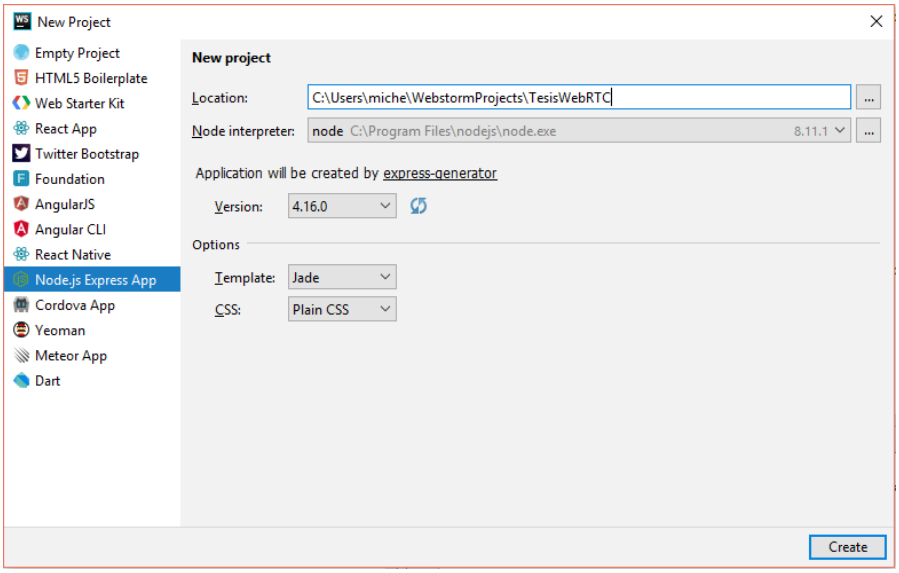

Figura 2.30 Creación de un proyecto en WebStorm

En el nuevo proyecto, se genera el archivo 'app.js', que es el archivo de configuración principal de *Express* en cual se deberán incluir algunos módulos para el correcto funcionamiento de la aplicación, estos se irán detallando en el desarrollo de esta sección de acuerdo con las partes desarrolladas.

- **Servidor Web HTTPS**

La configuración inicial de un servidor web HTTPS se localiza en el archivo 'app.js' y se la realiza mediante el módulo 'https' en conjunto con '*express*', en el Código 2.2 se muestra

la configuración para el funcionamiento del servidor web HTTPS. Al ser un servidor seguro requiere certificado y la llave privada como se constata en la línea 9 de Código 2.2 en la creación del servidor.

Estos certificados fueron generados con openSSL versión 1.0.2g en Ubuntu 16.04.

```
1.  var express = require('express');
2.  var https=require('https');
3.  var app = express();
4.  var puerto=1418;
5.  var opcionesCert={
6.      key: fs.readFileSync("./certificados/localhost.key"),
7.      cert: fs.readFileSync("./certificados/localhost.crt"),
8.  };
9.  var httpsServer =https.createServer(opcionesCert,app);
10. httpsServer.listen(puerto);
```

Código 2.2 Configuración del servidor https en 'app.js'

En el desarrollo de esta aplicación se requiere de forma obligatoria el uso de un servidor HTTPS por motivos de seguridad del navegador Chrome, especialmente para el uso de la función *getUserMedia* de la API de WebRTC la cual se encarga de obtener el video local de la cámara web o de la pantalla.

- **Direccionamiento**

En *Express*, direccionamiento hace referencia a la definición de puntos finales de aplicación (URI) y cómo responden a las solicitudes de cliente [32]. Para el direccionamiento se utilizó la clase *express.Router* para crear manejadores de rutas modulares, por lo cual se añadió la configuración del Código 2.3 al archivo 'app.js'.

```
1.  var indexRouter = require('./routes/routes');
2.  app.use('/', indexRouter);
```

Código 2.3 Configuración de direccionamiento en 'app.js'

La línea uno de Código 2.3 requiere al archivo 'routes.js' que contendrá las rutas definidas y que responderán al cliente que las solicite.

- **Motor de plantillas**

Existen motores de plantillas compatibles con *Express*, para el desarrollo de esta aplicación se utilizó Jade[26], esta configuración se añadió en 'app.js' y se puede visualizar en el Código 2.4.

[26] Jade: Motor de plantillas usado en el lado del servidor en Node.js.

```
1.  app.set('views', path.join(__dirname, 'views'));
2.  app.set('view engine', 'jade');
```

Código 2.4 Configuración del motor de plantillas Jade en ‘app.js’

Finalmente se añadieron los archivos ‘AdminController.js’, ‘UserController.js’ y ‘HomeController.js’.

El archivo ‘AdminController.js’ contiene todos los métodos exclusivos para los usuarios con rol administrador que soliciten acceso a rutas específicas de administrador como por ejemplo el panel de usuarios o el panel de administración de roles.

El archivo ‘UserController.js’ contiene métodos para acceso al módulo de ayuda, módulo de comunicación, edición de perfil y contraseña personal, además se encuentran todas las funciones para registro, autenticación y ‘*acerca de*’ la aplicación.

- **Administración de usuarios**

Dentro de la administración de usuarios se requería implementar tanto: interfaces de usuario como métodos que permitan responder al usuario de manera adecuada a sus peticiones.

En esta sección se explica de manera resumida la implementación de los métodos y vistas para las funcionalidades de: creación, listado, edición y eliminación de usuarios.

- **Creación de usuarios**

Para la creación de usuarios se crearon los métodos *getNewUserView* y *postNewUser* ubicados en el archivo ‘AdminController.js’.

getNewUsersView está encargado de enviar el formulario de ingreso de un nuevo usuario a el administrador que lo requiera, la implementación de este método se muestra en el Código 2.5.

Para este formulario se utilizaron elementos *input*: tipo texto para el ingreso de nombre, apellido y *username*, tipo *email* para el correo electrónico, tipo *password* para la contraseña y confirmación de la misma, tipo *file* para cargar una imagen, elementos de tipo *checkbox* para seleccionar los roles que se asignarán al nuevo usuario y un botón tipo ‘*submit*’ para guardar el nuevo usuario. Todos los campos de este formulario tienen la propiedad ‘*required’* para validar que los campos estén llenos antes de enviar la petición POST mediante el botón.

```
getNewUserView: function (req, res, next) {
        res.render('admin/create', {
            isAuthenticated: req.isAuthenticated(),
            user: req.user,
            error: req.flash('error')
        })
    },
```

Código 2.5 Método *getNewUserView* para obtener el formulario de creación de usuario

postNewUser es el método que se ejecuta cuando un usuario envía una petición POST al servidor para guardar un nuevo usuario. Este método procesa los datos ingresados en el formulario, encripta la contraseña utilizando el módulo Bcrypt de Node.js (Código 2.6), y guarda los campos correspondientes al usuario y los roles asignados en diferentes variables, para finalmente llamar a los métodos *insertUsuario* e *insertUsuarioPerfil* para el ingreso de los valores en la base de datos.

```
var pass = funciones.manageBcrypt.encriptar(req.body.password);
```

Código 2.6 Uso de función de encripción para la contraseña

Los métodos *insertUsuario* e *insertUsuarioPerfil* se encuentran declarados en el archivo 'consultasDB.js' , que contiene todos los métodos asociados a transacciones realizadas con la base de datos a lo largo de todo el proyecto, esta colección de métodos requiere al módulo *mysql* de Node.js para la conexión con la base de datos creada anteriormente. En el Código 2.7 se muestra como ejemplo el método de conexión a la base de datos y el método *insertUsuario* , el contenido completo del archivo 'consultasDB.js' se encuentra en el ANEXO C**.**

```
var mysql = require('mysql');
var config = require('.././database/config_db');
function connection() {
    var db = mysql.createConnection(config);
    db.connect();
    return db;
}
module.exports = {
insertUsuario: function (user, callback) {
        var db = connection();
        db.query('INSERT INTO usuario SET ?', user, function (err, rows, fields){
            if (err) {
                callback(err, {error: err})
            } else {
                db.end();
                callback(null, {
                    'insertID': rows.insertId
                });}
        })
    }}
```

Código 2.7 Método *connection* e *insertUsuario* con el uso del módulo mysl

Finalmente, el método *postNewUser* redirige al panel de usuarios si la transacción en la base de datos fue exitosa, de presentarse el caso de intento de ingreso de un usuario duplicado se redirige a la página de creación y se muestra un mensaje de error.

Para el envío de mensajes de éxito o de error a lo largo del desarrollo del proyecto se utilizó el módulo '*flash*' de Node.js cuya configuración en el archivo 'app.js' se muestra en el Código 2.8.

```
1. var flash=require('connect-flash');
2. app.use(flash());
```

Código 2.8 Configuración del módulo 'flash' en 'app.js'

Para el direccionamiento se declararon las rutas asociadas a la creación de un usuario en el archivo 'routes.js', y se muestra en el Código 2.9.

```
1. router.get('/admin/create', AuthMiddleware.isLogged, Controllers.AdminController.
   getNewUserView);
2. router.post('/admin/create', AuthMiddleware.isLogged, Controllers.AdminController
   .postNewUser);
```

Código 2.9 Rutas asociadas a la creación de un usuario

- **Panel de usuarios**

Para listar los usuarios o mostrar la información de los usuarios en el panel de administración de usuarios se ha utilizado el método *getUsersPanel* definido en 'AdminController.js' y que se muestra en el Código 2.10.

```
1.  getUsersPanel: function (req, res, next) {
2.          consultas.selectUsuarios(req.user.idUsuario, function (err, data) {
3.              if (data.rows) {
4.                  res.render('admin/usuarios', {
5.                      consulta: data.rows,
6.                      isAuthenticated: req.isAuthenticated(),
7.                      user: req.user,
8.                      message: req.flash('info')
9.                  });
10.             }
11.         });
12.     },
```

Código 2.10 Método *getUsersPanel*

Este método devuelve al usuario la vista del panel de usuarios, el método *selectUsuario* fue declarado en el archivo 'consultasDB.js' y realiza un SELECT de todos los usuarios registrados en la aplicación excepto aquel que se encuentra consultando.

La vista del panel de usuarios contiene un elemento *table* con columnas para el 'ID de usuario', 'Nombre', 'Apellido', '*Username*', 'Correo', 'Foto', y 'Acciones'. Las acciones se muestran como botones para cada uno de los usuarios listados, los botones son: 'Editar' y 'Eliminar'. En el Código 2.11 se encuentra parte de la implementación de la vista de panel de usuarios, el código completo se encuentra en el ANEXO C.

```
1.  table(class='table table-striped table-bordered table-responsive' )
2.      tr
3.          th ID
4.          th Nombre
5.          th Apellido
6.          th Username
7.          th Correo
8.          th Foto
9.          th Acciones
10.     each item in consulta
11.         tr
12.             td #{item.idUsuario}
13.             td #{item.firstName}
14.             td #{item.lastName}
15.             td #{item.username}
16.             td #{item.correo}
17.             td
18.                 img( class='img-
    rounded' src='/images/#{item.idUsuario}.#{item.foto}' stylestyle= style="max-
    width:70px; margin-top: -7px;")
19.             td
20.                 a(href="/admin/#{item.idUsuario}/edit" class='btn btn-
    warning btn-sm') Editar
21.                 form(action="/admin/eliminar/#{item.idUsuario}?_method=DELETE" me
    thod='post')
22.                     input(type='submit' value="Eliminar" class='btn btn-
    danger btn-sm')
```

Código 2.11 Extracto del archivo usuarios.jade

Dentro del archivo de rutas 'routes.js' se añadió la ruta para la petición GET del panel de usuarios.

- **Editar usuario**

Para acceder a la vista de edición de usuario se lo realiza desde el panel de administración de usuarios, presionando el botón 'Editar', en la petición GET se envía el *idUsuario*, para saber que usuario se editará.

Los métodos asociados son: *getEditView* que envía el formulario de edición al administrador, este formulario contiene elementos *input* de tipo texto para ingreso de nombre, apellido y *username*, y de tipo *file* para subir una imagen, se valida todos que todos los campos estén llenos antes de enviar la petición PUT al servidor mediante el botón 'Guardar Cambios'.

editUser es el método que se encarga de guardar los datos ingresados en el formulario de manera temporal, en una variable usuario y hace uso del método *updateUsuario* que es el método que ejecuta la sentencia UPDATE en la base de datos, de ser exitosa la transacción se redirige el panel de usuarios.

Dentro del archivo de rutas 'routes.js', se añadió la ruta GET y PUT para edición de usuarios como se muestra en el Código 2.12.

```
1. router.get('/admin/:idUsuario/edit', AuthMiddleware.isLogged, Controllers.AdminCo
   ntroller.getEditView);
2. router.put('/admin/:idUsuario/edit', AuthMiddleware.isLogged, Controllers.AdminCo
   ntroller.editUser);
```

Código 2.12 Rutas asociadas a la edicion de un usuario por parte de un administrador

Los formularios HTML no permiten el uso de los verbos PUT y DELETE de HTTP, por lo cual se añade en el servidor el módulo 'method_override' que permite especificar estos verbos no admitidos por HTML mediante campos ocultos en los formularios para que no existan problemas por este tipo de solicitudes enviadas al servidor. La configuración de este módulo en 'app.js' se muestra en el Código 2.13 y su uso en el formulario de edición de usuario se muestra en el Código 2.14.

```
1. var methodOverride=require('method-override');
2. app.use(methodOverride("_method"));
```

Código 2.13 Inclusión del módulo method-override en 'app.js'

```
1. form(action='/admin/#{id.idUsuario}/edit?_method=PUT' method='post' autocomplete=
   'off' enctype='multipart/form-data')
```

Código 2.14 Uso de method-override en el formulario de edición de un usuario

'method-override' se utiliza en todas las peticiones de tipo PUT y DELETE en el presente proyecto.

- **Eliminar usuarios**

Para eliminar un usuario se debe presionar el botón eliminar correspondiente a un usuario específico en el panel de administración de usuarios. Las propiedades del botón eliminar se muestran en el Código 2.15.

```
1. form(action="/admin/eliminar/#{item.idUsuario}?_method=DELETE" method='post')
2.     input(type='submit' value="Eliminar" class='btn btn-danger btn-sm')
```

Código 2.15 Propiedades del botón 'Eliminar'

deleteUser es el método asociado a la eliminación de un usuario y se muestra en el Código 2.16.

```
1.  deleteUser: function (req, res, next) {
2.         consultas.deleteUsuario(req.params.idUsuario, function (err, data) {
3.             if (data.msg) {
4.                 req.flash('info', data.msg);
5.                 res.redirect('/admin/usuarios');
6.             } })}
```

Código 2.16 Método deleteUser declarado en 'AdminController.js'

La ruta asociada a la eliminación se encuentra en el Código 2.17 añadido en el archivo 'routes.js'.

```
1.  router.delete('/admin/eliminar/:idUsuario', AuthMiddleware.isLogged, Controllers.
    AdminController.deleteUser);
```

Código 2.17 Ruta para eliminación de un usuario

- **Registro de usuarios en la aplicación**

Un usuario puede registrarse a sí mismo en la aplicación y se le asigna el rol de usuario automáticamente. Para este tipo de registro se han incluido los métodos: *getSignUpView* y *postSignUp* en el archivo 'routes.js'.

getSignUpView responde las solicitudes de registro de los clientes y devuelve la vista con el formulario para el registro, el formulario es similar al de 'Creación de usuarios' de la sección 2.2.3.2, con excepción de la asignación de roles.

postSignUp es el encargado de manejar la petición POST del usuario registrándose, este método es similar al método *postNewUser* de la sección de 2.2.3.2 y utiliza los métodos *insertUsuario* e *insertUsuarioPerfil* para las transacciones en la base de datos. Si el registro fue exitoso este método redirige al usuario a la página de inicio de sesión, caso contrario redirige a la página de registro y muestra un error.

Las rutas declaradas en 'routes.js' asociadas al registro de un usuario por sí mismo se detallan en el Código 2.18.

```
1.  router.get('/users/registro', Controllers.UserController.getSignUpView);
2.  router.post('/users/registro', Controllers.UserController.postSignUp);
```

Código 2.18 Rutas asociadas al registro de un usuario

- **Autenticación en la aplicación**

Para la obtención de la vista de inicio de sesión se desarrolló el método *getSignInView* ubicado en el archivo 'UserController.js' y detallado en el Código 2.19.

La vista de inicio de sesión consta de elementos *input* tipo *email* para el correo electrónico, y tipo *password* para la contraseña del usuario. Este formulario valida que los campos estén llenos y que el correo electrónico ingresado sea válido.

```
1.  getSignInView: function (req, res, next) {
2.          return res.render('users/signin', {
3.              message: req.flash('info'), authmessage: req.flash('authmessage')
4.          });
5.      },
```

Código 2.19 Método *getSignInView* para obtener vista de inicio de sesión

La ruta incluida para obtención de la vista de inicio de sesión se muestra en el Código 2.20.

```
1.  router.get('/users/signin', Controllers.UserController.getSignInView);
```

Código 2.20 Ruta asociada a la obtención de vista de inicio de sesión

Para el proceso de autenticación se utilizó el módulo Passport de Node.js utilizando el método '*authenticate*' como se muestra en el Código 2.21,

```
1.  router.post('/users/signin', passport.authenticate('local', {
2.      successRedirect: '/users/panel/', //si la autenticacion es exitosa
3.      failureRedirect: '/users/signin', //si falla la autenticación
4.      failureFlash: true //para mostrar un mensaje de la falla en la autenticacion

5.  }));
```

Código 2.21 Uso del método *authenticate* para autenticación

Para el uso de Passport se incluyó en el archivo de configuración 'app.js' el Código 2.22.

```
1.  var passport=require('passport');
2.  require('./passport/passport')(passport);
3.  app.use(passport.initialize());
4.  app.use(passport.session());
```

Código 2.22 Configuración de Passport en 'app.js'

Para el desarrollo de este proyecto se utilizó Passport con una estrategia local, es decir se utiliza métodos propios para la autenticación. Para este proceso primero se toma el correo electrónico ingresado por el usuario y se comprueba su existencia en la base de datos, si no hay resultados se redirige a la página de inicio de sesión y muestra un error. Si se

encuentra el *email* en la base de datos se devuelve todo el usuario y perfil o perfiles asociados a este usuario y utilizando la función del módulo Bcrypt se compara la contraseña ingresada con la contraseña almacenada en la base de datos.

De ser correcto, dentro de la variable *req.user* se guarda toda la información asociada al usuario que inició la sesión y se redirige al perfil de usuario. De ser incorrecto redirige a la página de inicio de sesión y muestra un error. En el Código 2.23 se muestra la parte de la comparación de contraseñas con Bcrypt. El código completo se encuentra en el ANEXO C.

```
1.  if (funciones.manageBcrypt.compararPass(password, user.password)) {
2.       return done(null, {
3.                            idUsuario: user.idUsuario,
4.                            firstName: user.firstName,
5.                            lastName: user.lastName,
6.                            username: user.username,
7.                            correo: user.correo,
8.                            foto: user.foto,
9.                            descripcion: user.perfil
10.                          });
11.      }
```

Código 2.23 Uso de método *compararPass* para autenticación

2.2.2 Sprint 2

2.2.2.1 Sprint Planning

En la Tabla 2.15 se presentan los ítems del *product backlog* seleccionados para cumplir con el objetivo del *sprint* 2.

Tabla 2.15 *Sprint backlog 2*

	Requerimiento	Tareas	Esfuerzo	Horas
5	Videoconferencia	Creación de vista principal para el usuario: comunicación.jade.	Alto	10
		Métodos asociados al archivo principal.js.	Alto	10
		Métodos asociados a connection.js.	Alto	10
		Métodos asociados a WebRTC.js.	Alto	10
6	Compartición de pantalla	Métodos asociados para compartición de pantalla.	Alto	10
		Métodos asociados para renegociación.	Alto	10
7	Transferencia de mensajes	Métodos asociados a la transferencia de mensajes.	Alto	5

2.2.2.2 Desarrollo del Sprint 2

Para el desarrollo del *sprint* 2 se tuvieron varias consideraciones ya que se trató del desarrollo de todos los componentes del módulo de comunicación.

El módulo de comunicación realiza las funcionalidades de WebRTC en el lado del cliente por lo cual existen varios *scripts* asociados al proceso de comunicación y que se incluyen en la vista principal que envía el servidor web.

Existen varios ejemplos desarrollados para comunicación en tiempo real con WebRTC los cuales se encuentran disponibles en la plataforma GitHub. Para el desarrollo de este módulo se ha tomado en cuenta algunos componentes del repositorio 'WebRTC Experiments' [33], los cuales fueron adaptados y modificados para los fines de funcionalidad requeridos por el proyecto. Uno de estos componentes corresponde al servidor de señalización denominado 'Socket.io over Node.js' [34], disponible para instalar vía npm o para ser descargado, modificado y usado, la versión de la librería Socket.IO utilizada es la 0.9.

El uso y modificación de los componentes descritos anteriormente no infringe ninguna ley de propiedad intelectual.

Otras librerías de JavaScript que se han utilizado para brindar funcionalidad al módulo de comunicación se irán mencionando en las siguientes secciones.

Los archivos principales para el funcionamiento del módulo de comunicación son:

- comunicacion.jade (GET del módulo de comunicación por parte del usuario)
- principal.js
- connection.js
- WebRTC.js

Las librerías utilizadas para compartición de pantalla:

- getScreenId.js [35]
- DetectRTC.js [36]
- adapter-latest.js [37]

- **Videoconferencia**

Los archivos relevantes para cumplir con la funcionalidad de videoconferencia en la aplicación son: 'comunicación.jade', 'principal.js', 'connection.js' y 'WebRTC.js'. A

continuación, se explica el contenido de los archivos y los métodos más importantes asociados a la función de videoconferencia.

- **comunicación.jade**

Este archivo contiene la estructura principal HTML del módulo de comunicación y detalla la ubicación de los elementos en la página principal. Se cuenta con un elemento de tipo *section* llamado 'principal', donde se ubican varias tablas que se mencionan a continuación:

- La tabla denominada 'conferencia' será la primera en mostrarse al usuario al ingresar al módulo y contendrá un cuadro de texto para el ingreso de nombre del cuarto de comunicación y un botón para iniciar la videoconferencia.
- La tabla denominada 'listaCuartos' se mostrará a los usuarios que intentan unirse a un cuarto establecido, mostrando el nombre del cuarto, quién organizó la llamada y un botón para unirse a la videoconferencia.
- La tabla denominada 'tablaMensajes' es la asociada a el envío y recepción de mensajes y se detallará en la sección de transferencia de mensajes.

Dentro de otra sección denominada 'video' se ubicó un *div* denominado 'contenedorLocal' destinado a albergar el video local del usuario, ya sea por cámara web o por compartición y se declaró un *div* 'contenedorRemotos' que contiene los elementos de video de los demás participantes. También se añadieron botones de acción para: compartir pantalla, detener compartición y detener videoconferencia.

Los *scripts* que utiliza esta vista se muestran en el Código 2.24. El código completo se muestra en el ANEXO C**.**

```
1.  link(rel='stylesheet', href='//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css')
2.  script(src="https://code.jquery.com/jquery-3.3.1.slim.min.js")
3.  script(src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js")
4.  link(rel='stylesheet', href='/stylesheets/style.css')
5.  script(src="/javascripts/socket.io.js")
6.  script(src="https://webrtc.github.io/adapter/adapter-latest.js")
7.  script(src="/javascripts/DetectRTC.js")
8.  script(src="/javascripts/getScreenId.js")
9.  script(src="/javascripts/WebRTC.js")
10. script(src="/javascripts/connection.js")
11. script(src="/javascripts/principal.js")
```

Código 2.24 *Scripts* requeridos para el módulo de comunicación

- **principal.js**

El archivo ‘principal.js’ contiene varios métodos importantes para señalización y manejo de acciones de los botones que se encuentran en la vista ‘comunicación.jade’.

A continuación, se explica la funcionalidad de algunos de los métodos más importantes para el proceso de videoconferencia.

- *senalizacionSockets*: recibe como parámetro de entrada la configuración del canal y configura los eventos de la librería ‘socket.io’ para la conexión y transmisión de mensajes, es el encargado de verificar la conexión con el servidor de señalización al ingresar al módulo de comunicación, el Código 2.25 muestra la implementación de este método.

```
1.  senalizacionSockets: function (configCanal) {
2.          var serverSenalizador = 'https://192.168.1.88:4444/';
3.          configCanal.channel = configCanal.channel || location.href.replace(/\/|:|#
    |%|\.|\[|\]/g, '');
4.          var sender = Math.round(Math.random() * 999999999) + 999999999;
5.          io.connect(serverSenalizador).emit('new-channel', {
6.              channel: configCanal.channel, sender: sender});
7.          var socket = io.connect(serverSenalizador + configCanal.channel);
8.          socket.on('error', function (err) {
9.              $('#alertaConexionServer').show();
10.             $('.principal').toggle();
11.         });
12.         socket.channel = configCanal.channel;
13.         socket.on('connect', function () {
14.             if (configCanal.callback) configCanal.callback(socket);
15.         });
16.         socket.send = function (mensaje) {
17.             socket.emit('message', {
18.                 sender: sender, data: mensaje})
19.         };
20.         socket.on('message', configCanal.onmessage);
21.     }
```

Código 2.25 Método *senalizacionSockets* declarado en ‘principal.js’

- *cuartoEncontrado:* recibe como parámetro de entrada la configuración del cuarto de comunicación y será el encargado de mostrar a un usuario que se une a una videoconferencia el botón para unirse.

- *anadirVideoRemoto*: recibe como parámetro de entrada el video proveniente del par remoto. Hace uso del método *manejarVideo* que se encarga ajustar el tamaño del video para adaptarlo a la ventana y añadirlo al contenedor de videos remotos. El Código 2.26 muestra el método *anadirVideoRemoto* y la invocación del método de ajuste del elemento de video.

```
anadirVideoRemoto: function (media) {
        contadorVideo++;
        manejarVideo(contadorVideo, media);
    },
```

Código 2.26 *Método anadirVideoRemoto*

Se utilizó JQuery para el manejo de los botones de inicio y fin de videoconferencia, en el Código 2.27 se muestra la implementación del manejador del evento clic en el botón de inicio de videoconferencia.

```
$(document).ready(function () {
    $("#inicio-videoconferencia").click(function () {
        if ($('#nombre-cuarto').val()) {
            var username = usernameV.value;
            captureUserMedia(function () {
                conexion.crearCuarto({
                    userName: username,
                    roomName: document.getElementById('nombre-cuarto').value
                });
            });
            document.getElementById('txtlink').value = window.location;
            $('#modalURL').modal('show');
            $('#alertaPortapeles').hide();
            ocultarElementos();
        } else {
            alert('Ingrese un nombre de cuarto!!');
        }
    });
});
```

Código 2.27 Manejador del botón inicio-videoconferencia

Dentro del archivo ‘principal.js’ se declara la función *captureUserMedia()* utilizada para obtener el video local del usuario.

- **connection.js**

Contiene los métodos asociados a comunicación entre *sockets* y controla los mensajes que se envían como señalización. También declara a los pares WebRTC y procesa los resultados de los métodos de la API WebRTC que se ejecutan en el archivo ‘WebRTC.js’.

El archivo ‘connection.js’ retorna algunos métodos que son utilizados por ‘principal.js’ para la creación del cuarto para videoconferencia y para unirse a un cuarto establecido.

En el Código 2.28 se detalla la implementación de los métodos de creación y unión a un cuarto.

```
 crearCuarto: function (roomParams) {
             self.roomName = roomParams.roomName;
             self.roomID = uniqueToken();
             self.userName = roomParams.userName;
             iniciadorBroadcast = true;
             isGetNewRoom = false;
             iniciarBroadcast();
         },
unirseCuarto: function (roomParams) {
    self.roomID = roomParams.roomID;
    self.userName = roomParams.userName;
    isGetNewRoom = false;
    self.joinedARoom = true;
    self.broadcasterid = roomParams.joinUser;

    abrirSubSocket({
       channel: self.userID
            });
    socketDefault && socketDefault.send({
         participant: true,
         userID: self.userID,
         joinUser: roomParams.joinUser
            });
        },
```

Código 2.28 Métodos *crearCuarto* y *unirseCuarto*

Dentro del archivo ‘connection.js’ se declaró una variable denominada *peerConfig* que hace uso de los métodos declarados en ‘principal.js’.

El método *iniciarPeer* es el encargado de crear un nuevo par WebRTC usando los métodos alojados en ‘WebRTC.js’ la implementación de este método se encuentra en el Código 2.29.

```
function iniciarPeer(ofertaSDP) {
             if (!ofertaSDP) {
                 peerConfig.onOfferSDP = enviarSDP;
             } else {
                 peerConfig.offerSDP = ofertaSDP;
                 peerConfig.onAnswerSDP = enviarSDP;
             }
             peer = WebRTC(peerConfig);
         }
```

Código 2.29 Método *iniciarPeer* para inicio de comunicación WebRTC

- **WebRTC.js**

Contiene la utilización de los métodos propios de la API de WebRTC, la configuración de ciertos parámetros para crear un canal de comunicación y el manejo de ciertos eventos propios de WebRTC.

Se declaran las variables *PeerConnection*, *SessionDescription* e *IceCandidate* y se definen los servidores ICE públicos de Google en caso de ser necesarios.

En el Código 2.30 se observa la declaración de las variables antes mencionadas, la creación de una nueva conexión y un canal SCTP para la transmisión de datos.

```
1.   var w = window,
2.          PeerConnection = w.webkitRTCPeerConnection,
3.          SessionDescription = w.RTCSessionDescription,
4.          IceCandidate = w.RTCIceCandidate;
5.
6.  var iceServers = [];
7.  iceServers.push({
8.          url: 'stun:stun.l.google.com:19302'
9.      });
10.
11. var peer = new PeerConnection(iceServers, optional);
12. var dataChannelDict = {};
13. channel = peer.createDataChannel('sctp-channel', dataChannelDict);
```

Código 2.30 Declaración de un nuevo canal 'WebRTC.js'

Para la creación de ofertas y respuestas SDP se usan los métodos *createOffer* y *createAnswer* que usan métodos propios del API de WebRTC que sirven para crear ofertas y respuestas SDP y para guardar las descripciones locales y remotas de los pares. Estos métodos se muestran en el Código 2.31.

```
1.  function createOffer() {
2.          if (!options.onOfferSDP) return;
3.
4.          peer.createOffer(function (sessionDescription) {
5.              peer.setLocalDescription(sessionDescription);
6.              options.onOfferSDP(sessionDescription);
7.      }
8.
9.
10. function createAnswer() {
11.     if (!options.onAnswerSDP) return;
12.         peer.setRemoteDescription(new SessionDescription(options.offerSDP), onSdp
    Success, onSdpError);
13.         peer.createAnswer(function (sessionDescription) {
14.             peer.setLocalDescription(sessionDescription);
15.             options.onAnswerSDP(sessionDescription);
16.         }, onSdpError, constraints);
17.     }
```

Código 2.31 Declaración de métodos *createOffer* y *createAnswer*

El código completo del archivo 'WebRTC.js' se encuentra en el ANEXO C.

- **Compartición de Pantalla**

La compartición de pantalla se dará una vez iniciada la videoconferencia y solo por parte del usuario que inició la videoconferencia. El manejador del botón Compartir pantalla se encuentra en el archivo 'principal.js' y se detalla en el Código 2.32.

```
$(document).ready(function () {
    $("#compartirPantalla").click(function () {
        captureUserMediaScreen(function () { });
        botonesComparticion('compartir');
    })
});
```

Código 2.32 Manejador del botón compartir pantalla

La función *CaptureUserMediaScreen()* es la función encargada de obtener la pantalla del organizador, utilizando la librería getScreenId.js [35].

- ***getScreenId*:** permite obtener un parámetro denominado *'chrome-media-source-id*' que se utiliza para capturar pantallas, *getScreenId* interactúa con la extensión del navegador llamada *Screen Capturing* [38], para obtener el id antes mencionado. getScreenId.js se conecta con la extensión de Chrome usando un *iframe* interno para enviarse mensajes post entre sí. Por su parte la extensión del navegador se encarga del uso de *Desktop Capture API*[27] para la obtención de la pantalla.

Luego de obtener el video de captura de pantalla, se lo ubica dentro del contenedor de video local, reemplazando al elemento de video perteneciente a la cámara web.

En el archivo 'connection.js' una función importante para la compartición de pantalla es:

- *share*: función utilizada para detener el broadcast del video procedente de la cámara web y de enviar un mensaje de cambio de *stream* a los demás participantes para iniciar la renegociación, además retira el video local del canal WebRTC como se muestra en el Código 2.33.

```
peer.peer.removeStream(configCanal.attachStream);
```

Código 2.33 Uso de *removeStream* para remover video del canal WebRTC

Después del intercambio de estos mensajes de señalización y del retiro del *stream* de video proveniente de la cámara web del canal, el organizador que decidió compartir la pantalla, agrega el nuevo *stream* de video al canal e inicia la renegociación creando una oferta SDP como se muestra en el Código 2.34.

```
peer.peer.addStream(configCanal.attachStream);
                peer.peer.onnegotiationneeded = function () {
                    peer.peer.createOffer().then(function (offer) {
                        return peer.peer.setLocalDescription(offer);
 }) }) }
```

Código 2.34 Inicio del proceso de renegociación entre pares

[27] *Desktop Capture API*: API de captura de escritorio que se puede usar para capturar contenido de la pantalla, ventanas individuales o pestañas.

Por su parte los pares remotos crean una respuesta SDP que se muestra en el Código 2.35, y se utilizan los mismos métodos usados para el video de cámara web para colocar el *stream* remoto en los diferentes contenedores de video remoto.

```
peer.peer.setRemoteDescription(respuesta.sdpR);
                peer.peer.createAnswer().then(function (answer) {
                    return peer.peer.setLocalDescription(answer);
                }).then(function () {
                    socket.send({
                        userID: self.userID,
                        sdpA: peer.peer.localDescription
                    })
                }).catch();
```

Código 2.35 Creación de respuesta SDP de un par secundario en la renegociación

Para la compartición de pantalla también se incluye el uso de DetectRTC.js [36], que antes de iniciar la videoconferencia notificará al usuario si la extensión necesaria está instalada.

- **Transferencia de mensajes**

Para la transferencia de mensajes se usa el mismo canal creado para la videoconferencia, la tabla asociada al envío de mensajes llamada *tablaMensajes* contiene un *input* tipo texto para ingresar el mensaje y un botón 'Enviar' denominado *postMessage*.

La tabla asociada para mostrar los mensajes recibidos se denomina *outputMensajes*. Estas tablas se harán visibles una vez que se inicie la videoconferencia.

Los métodos asociados a la transferencia de mensajes en el archivo 'principal.js' son:

- *Envío de mensajes*: el manejador del botón 'Enviar mensaje' se muestra en el Código 2.36. En el archivo 'connection.js' el método asociado al envío de mensajes se denomina 'enviar' y envía los mensajes a través del canal RTCDataChannel.

```
$(document).ready(function () {
    $("#inputMensaje").change(function () {
        conexion.enviar(this.value);
        var tr = document.createElement('tr');
        tr.innerHTML =
            '<td style="width:40%;"> Tu : </td>' +
            '<td>' + inputMensaje.value + '</td>';
        outputMensajes.insertBefore(tr, outputMensajes.firstChild);
        inputMensaje.value = '';
    })
});
```

Código 2.36 Manejador de evento para envío de mensajes

- *Recepción de mensajes*: el método involucrado en la recepción de mensajes se denomina *mensajeEnCanal y* se encarga de ubicar en la tabla *outputMensajes* algún mensaje recibido, su implementación se detalla en el Código 2.37.

```
1.  mensajeEnCanal: function (data) {
2.          if (!outputMensajes) return;
3.          var tr = document.createElement('tr');
4.          tr.innerHTML =
5.              '<td style="width:40%;">' + data.sender + ' : ' + '</td>' +
6.              '<td>' + data.message + '</td>';
7.          outputMensajes.insertBefore(tr, outputMensajes.firstChild);
8.      },
```

Código 2.37 Método *mensajeEnCanal* para mensajes recibidos

2.2.3 Sprint 3

2.2.3.1 Sprint Planning

En la Tabla 2.16 se presentan los ítems del *product backlog* seleccionados para cumplir con el objetivo del *sprint* 3.

Tabla 2.16 *Sprint backlog 3*

	Requerimiento	Tareas	Esfuerzo	Horas
8	Información de desarrollo de la aplicación y manual de usuario	Información: Diagrama del prototipo (vista y métodos)	Alto	5
		Información: Características de aplicación(Vista y métodos)	Alto	5
		Información: Creación de cuenta y roles de usuario(vista y métodos)	Alto	5
		Información: Manual de uso	Alto	10
9	Administración de roles	Ver a los usuarios registrados y sus roles asignados (vista y métodos)	Medio	10
		Añadir un rol (métodos)	Medio	10
		Eliminar un rol (vista y métodos)	Medio	5

2.2.3.2 Desarrollo del Sprint 3

- **Información de desarrollo de la aplicación**

Para el desarrollo del módulo de ayuda o de información se declararon métodos para el manejo de peticiones GET que realizarán los usuarios. Para una mejor organización se dividió al módulo informativo en cuatro partes que son:

- Diagrama del prototipo
- Características de la aplicación

- Creación de cuenta y roles de usuario
- Manual de uso

La implementación de los métodos para cada parte es similar por lo que a continuación se detalla la implementación de 'Diagrama del prototipo', el código asociado al resto de partes se detalla completo en el ANEXO C.

- **Diagrama del prototipo**

Para mostrar información acerca del diagrama del prototipo se utilizó el método *getDiagram* que envía al usuario la vista y la información asociada al diagrama, en el Código 2.38 se muestra la implementación de este método.

```
1.  getDiagram: function (req, res, next) {
2.          consultas.selectInformacion('diagram explanation', function (err, data) {
3.              if (err) throw err;
4.
5.              res.render('users/ayuda/diagrama', {
6.                  isAuthenticated: req.isAuthenticated(),
7.                  user: req.user,
8.                  informacion: data.informacion
9.              });
10.         });
11. }
```

Código 2.38 Implementación del método *getDiagram* en 'UserController.js'

En la línea dos del Código 2.38 se muestra el uso del método *selectInformacion* declarado en 'consultasDB.js' este método realiza un SELECT de la información solicitada en la base de datos y cuyo resultado se mostrará en la vista que se envía al usuario, *selectInformacion* usado por todos los métodos asociados a la obtención de información de ayuda.

La vista enviada al usuario se compone principalmente de elementos tipo *p* para mostrar el texto informativo y de tipo *img* para mostrar imágenes.

La tabla 'informativa' almacena la información correspondiente a este módulo en la base de datos.

- **Administración de roles**

- **Panel de Administración de Roles**

Para listar los usuarios de la aplicación y los roles asociados a cada uno de ellos se ha utilizado el método *getRolesPanel* definido en 'AdminController.js' y que se muestra en el Código 2.39.

```
1.  getRolesPanel: function (req, res, next) {
2.          consultas.selectAllRoles(req.user.idUsuario, function (err, data) {
3.              if (data.consulta) {
4.                  res.render('admin/roles', {
5.                      isAuthenticated: req.isAuthenticated(),
6.                      user: req.user,
7.                      consulta: data.consulta,
8.                      message: req.flash('info')
9.                  })
10.             }
11.         })
12.     }
```

Código 2.39 Método *getRolesPanel*

Este método devuelve al usuario la vista del panel de administración de roles, en la línea dos del Código 2.39 se utiliza el método *selectAllRoles* que fue declarado en el archivo 'consultasDB.js' y realiza un SELECT de todos los usuarios registrados más los roles de estos.

La vista de panel de administración de roles contiene principalmente un elemento *table* con columnas para el nombre, *username*, foto, rol actual y acciones. En la columna de acciones se toma en cuenta si: un usuario tiene un solo rol se muestra el botón 'Añadir', si el usuario cuenta con dos roles se muestra el botón 'Eliminar'.

- **Añadir un Rol**

Para añadir un rol se ha utilizado el método *addRole*, declarado en el archivo 'AdminController.js' y que se muestra en el Código 2.40.

```
1.  addRole: function (req, res, next) {
2.          var User = {};
3.          User.idUsuario = req.params.idUsuario;
4.
5.          if ((req.body['adminCh' + req.params.idUsuario] === 'on') || (req.body['us
    erCh' + req.params.idUsuario] === 'on')) {
6.              consultas.insertRol(User, function (err, data) {
7.                  if (data.msg) {
8.                      req.flash('info', data.msg);
9.                      res.redirect('/admin/roles');
10.                 }
11.             })
12.         }
13.     }
```

Código 2.40 Método *addRole* declarado en 'AdminController.js'

Para añadir un rol, se lo hará desde el panel de administración de roles, el rol que se desea añadir será seleccionado en la columna roles, y se presionará el botón 'Añadir'. La ruta asociada a la acción de añadir un rol de usuario se muestra en el Código 2.41.

```
1.  router.post('/admin/roles/:idUsuario', AuthMiddleware.isLogged, Controllers.Admin
    Controller.addRole);
```

Código 2.41 Ruta asociada para añadir un rol declarada en ‘routes.js’

- **Eliminar un rol**

Para eliminar un rol se implementaron los métodos *getDeleteView* y *deleteRol*, declarados en ‘AdminController.js’.

getDeleteView, es el encargado de responder las peticiones GET de un usuario que presionó el botón ‘Eliminar’ en el panel de administración de roles y devuelve la vista asociada a la eliminación. Los elementos principales de esta vista corresponden a un formulario que muestra los roles actuales , *checkboxes* que permiten al usuario escoger el rol a eliminar, y un botón que envia la petición de DELETE.

deleteRol es el método que se ejecuta cuando el usuario haga una petición de tipo DELETE para eliminar un rol, parte de la implementación de este método se muestra en el Código 2.42.

```
1.  deleteRole: function (req, res, next) {
2.          consultas.deleteRol(rolSeleccionado, req.params.idUsuario, function (err,
                data) { if (data.msg) {
3.                req.flash('info', data.msg);
4.                res.redirect('/admin/roles');
5.              }
6.          })
7.      }
```

Código 2.42 Método *deleteRole* para eliminación de un rol de usuario

El método *deleteRol* presente en la línea dos del Código 2.42 ha sido declarado en ‘consultasDB.js’ y ejecuta la sentencia DELETE en la base de datos.

Las rutas asociadas al proceso de eliminación se muestran en el Código 2.43 y se encuentran declaradas en el archivo ‘routes.js’.

```
1.  router.get('/admin/roles/:idUsuario/delete', AuthMiddleware.isLogged,Controllers.
    AdminController.getDeleteView);
2.  router.delete('/admin/roles/delete/:idUsuario',AuthMiddleware.isLogged, Controlle
    rs.AdminController.deleteRole);
```

Código 2.43 Rutas asociadas a la eliminación de un rol de usuario

2.2.4 Sprint 4

2.2.4.1 Sprint Planning

En la Tabla 2.17 se presentan los ítems del *product backlog* seleccionados para cumplir con el objetivo del *sprint* 4.

Tabla 2.17 *Sprint backlog 4*

	Requerimiento	Tareas	Esfuerzo	Horas
10	Edición de información personal y contraseña	Editar información de usuario (vista y métodos).	Medio	10
		Editar contraseña de usuarios(vista y métodos).	Medio	10
11	Mostrar perfil de usuario	Vista y métodos para perfil de usuario.	Medio	5
12	Home Page/ Acerca De	Vista para página home.	Bajo	5
		Vista para página acerca de	Bajo	5

2.2.4.2 Desarrollo del Sprint 4

- **Edición de información personal y contraseña**

  - **Edición de información personal**

Para la edición de la información personal de un usuario registrado, se asocian dos métodos : *getUserConfigurationView* y *updateUserConfiguration* declarados en el archivo 'UserController.js' y que son similares a los métodos correspondientes a edición de un usuario por parte de un administrador presentado en la sección 2.2.3.2: Desarrollo del Sprint 1.

- *getUserConfigurationView:* envía el formulario de edición al usuario, se valida que los campos estén llenos y en un formato correcto, no se puede actualizar el correo electrónico.

- *updateUserConfiguration:* es el método que ejecuta la sentencia UPDATE en la base de datos, de ser exitosa la transacción se redirige al usuario a su perfil.

Las rutas asociadas a la edición de datos personales se muestran en el Código 2.44 y están declaradas en el archivo 'routes.js'.

```
1. router.get('/users/configuracion', AuthMiddleware.isLogged, Controllers.UserContr
   oller.getUserConfigurationView);
2. router.put('/users/configuracion', AuthMiddleware.isLogged, Controllers.UserContr
   oller.updateUserConfiguration);
```

Código 2.44 Rutas asociadas a la edición de información personal de un usuario

- **Cambio de contraseña**

Los métodos asociados al cambio de contraseña son *getEditPasswordView* y *changePassword* que se encuentran declarados en 'UserController.js'.

- *getEditPasswordView*: responde a las peticiones GET de la vista para cambio de contraseña, contiene un formulario con elementos *input* tipo *text* para el ingreso de la contraseña actual, de la nueva contraseña y su confirmación, y un botón que servirá para la petición PUT del usuario.
- *changePassword:* Este método, ejecuta el método *updatePassword* declarado en 'consultasDB.js', que utilizando Bcrypt, compara la contraseña actual ingresada con la almacenada en la base de datos, de ser correcta compara que la contraseña nueva no sea igual a la anterior, de ser diferente ejecuta la sentencia UPDATE en la base de datos. De ser exitoso el cambio de contraseña se redirige al perfil del usuario.

Las rutas declaradas en 'routes.js' se muestran en el Código 2.45.

```
1. router.get('/users/configuracion/changepass',AuthMiddleware.isLogged,Controllers.
   UserController.getEditPasswordView);
2. router.put('/users/configuracion/changepass', AuthMiddleware.isLogged,Controllers
   .UserController.changePassword);
```

Código 2.45 Rutas asociadas al cambio de contraseña de un usuario

- **Mostrar perfil de usuario**

El método asociado a mostrar el perfil de usuario es *getUserPanelView* y se muestra en el Código 2.46.

```
1. getUserPanelView: function (req, res, next) {
2.         res.render('users/panel', {
3.             isAuthenticated: req.isAuthenticated(),
4.             user: req.user,
5.             message: req.flash('info')
6.         });
7.     },
```

Código 2.46 Implementación del método *getUserPanelView*

En la línea cuatro del Código 2.46 se puede notar que se envía adjunto a la vista el objeto *req.user*, dado que en este objeto se guardan los datos asociados al usuario cuando se autenticó. La vista contiene un panel que muestra la información personal del usuario.

- **Página de Inicio/ Acerca De**
- **Página de Inicio**

El método asociado a la obtención de la página de inicio se declaró como *index* en el archivo 'HomeController.js', en el Código 2.47 se muestra la implementación de este método.

```
1.  index: function (req,res,next) {
2.          if(req.isAuthenticated()){
3.              res.redirect('/users/panel')
4.          }else{
5.          res.render('home',{
6.              isAuthenticated: req.isAuthenticated(),
7.              user : req.user
8.          });
9.      }}
```

Código 2.47 Implementación del método *index* para obtención de página de inicio

Se puede notar que en las líneas 2 a 4 se inserta un condicional que redirige al usuario autenticado a su perfil en caso solicitar la ruta de inicio.

La ruta asociada a la obtención de la página de inicio se declaró en el archivo 'routes.js' y el código completo asociado se incluye en el ANEXO C.

- **Acerca de la aplicación**

El método asociado a la obtención de la página: *acerca de*, se declaró como *getAcercaDeView* en 'UserController.js', este método se encarga de enviar la vista de la página '*acerca de*' la aplicación, sirve como una guía breve para inicio de sesión y registro de los usuarios. La ruta asociada se declaró en 'routes.js' y se detalla en el Código 2.48.

```
1.  router.get('/users/acercade',Controllers.UserController.getAcercaDeView);
```

Código 2.48 Ruta asociada a la página *acerca de*

2.3 Implementación del Prototipo en Google Cloud

Una vez terminada la fase de codificación de los módulos componentes del sistema y de comprobar su funcionamiento localmente, se procede a la implementación del prototipo en un ambiente *cloud,* usando la prueba gratuita de Google Cloud Platform[28], haciendo uso de los servicios Compute Engine[29] y Google Cloud DNS[30].

[28] *Google Cloud Platform*: Espacio virtual que reúne las aplicaciones de desarrollo web de Google y que utilizan la nube como única forma de acceso, almacenamiento y gestión de datos.
[29] *Compute Engine*: Es el componente de Infraestructura como servicio de Google Cloud Platform, permite al usuario crear y lanzar máquinas virtuales con imágenes estándar o imágenes personalizadas.
[30] *Google Cloud DNS*: Permite publicar nombres de dominio utilizando la infraestructura de Google para servicios DNS.

Google Compute Engine fue seleccionado para la implementación del proyecto debido a la facilidad de configuración de las máquinas virtuales que se crean y vienen predefinidas para adaptarse a diferentes necesidades. Adicionalmente, presenta compatibilidad con sistemas operativos tipo Windows y Linux, en los cuales se puede instalar todas las dependencias necesarias para el funcionamiento del prototipo. Por último, Google Cloud cuenta con gran cantidad de información de ayuda para poder poner en marcha un proyecto en su plataforma.

El objetivo principal de la implementación de este proyecto en un ambiente *cloud*, es el de facilitar la realización de pruebas, aportando también seguridad.

2.3.1 Creación de una instancia en Google Compute Engine

Para crear una instancia o máquina virtual, se debe ingresar a la plataforma de Google Cloud al módulo Compute Engine, a Instancias de VM, en este módulo se creó una nueva instancia, como se muestra en la Figura 2.31.

Para desplegar los servidores web, de señalización y MySQL se creó una instancia de Ubuntu 16.04 LTS, como se muestra en la Figura 2.32, cuya dirección IP asignada fue 35.196.122.38.

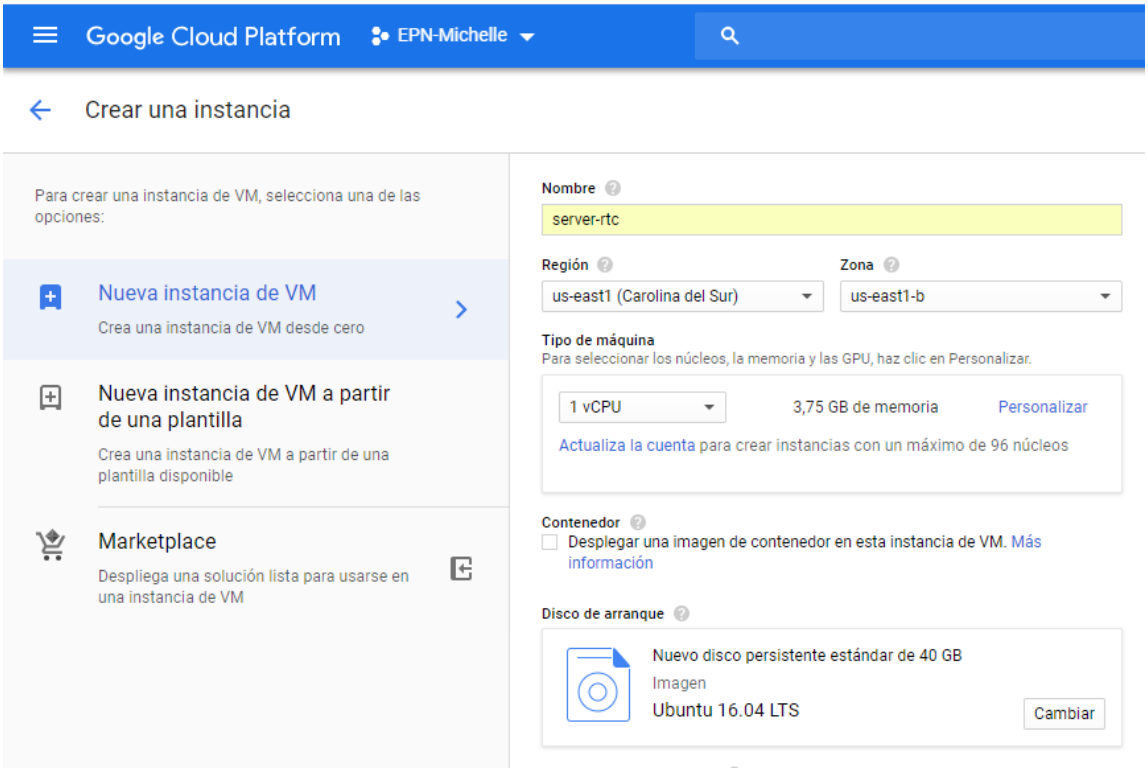

Figura 2.31 Creación de una instancia en Google Compute Engine

Figura 2.32 Instancia creada en Google Compute Engine

2.3.2 Asignación de nombre de dominio

Para fines de pruebas de la aplicación se ha creado un dominio gratuito: 'appwebrtc-epn.tk', en Freenom[31] y se han asignado los nombres de servidores de Google Cloud como se muestra en la Figura 2.33.

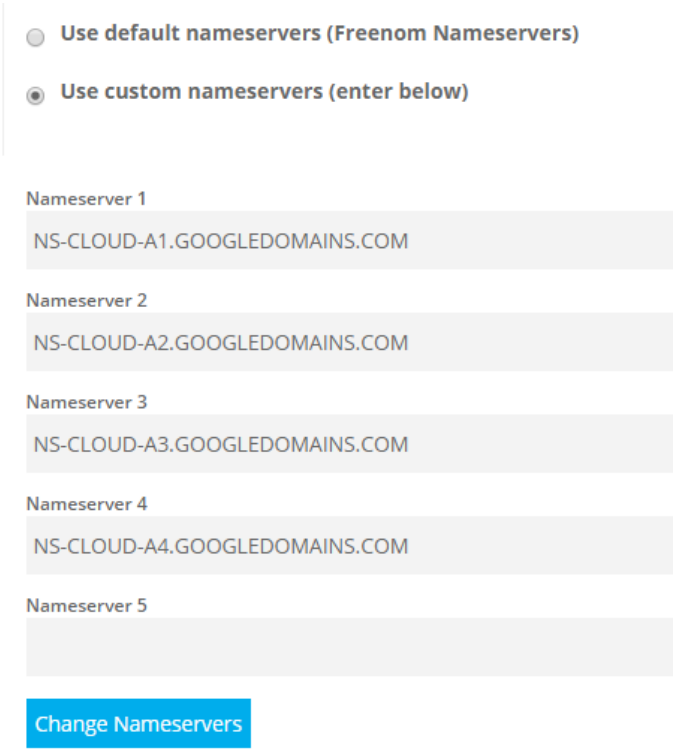

Figura 2.33 Asignación de nombres de servidores de Google Cloud en Freenom

Google Cloud DNS se usó para asociar el dominio creado a la dirección IP asignada para la instancia, la configuración de los registros en Cloud DNS se muestra en la Figura 2.34.

[31] Freenom: Proveedor de dominios gratuitos y de pago.

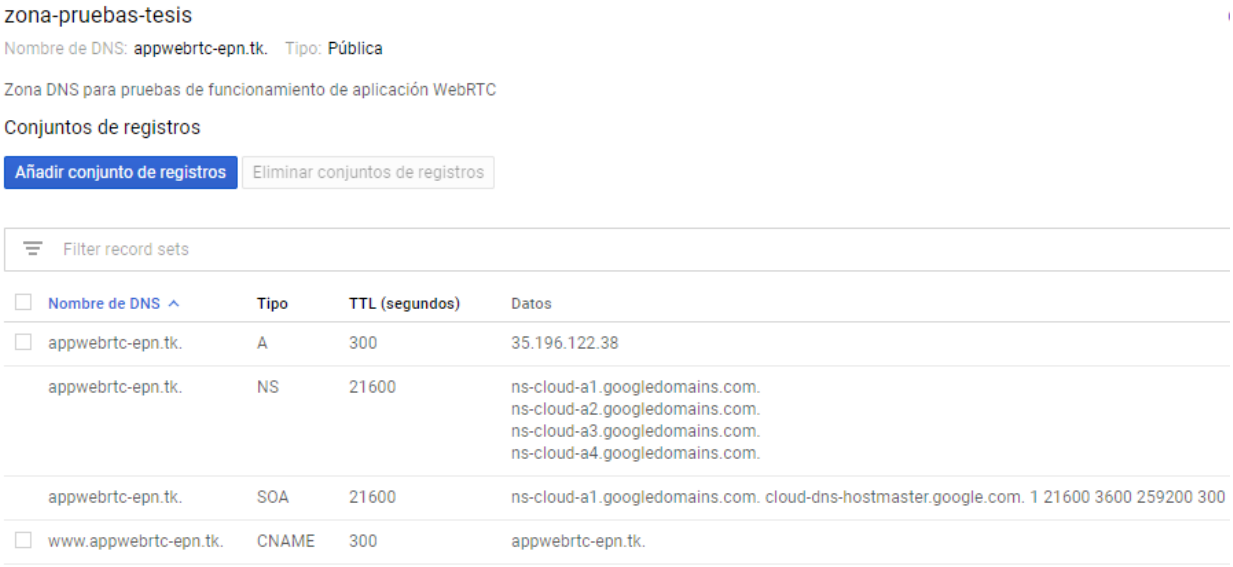

Figura 2.34 Configuración de registros para el dominio appwebrtc-epn.tk

2.3.3 Configuración de la instancia creada

Luego de la creación de la instancia se ingresó al servidor Ubuntu 16.04 para realizar las configuraciones y poner en marcha los servidores asociados para el funcionamiento de la aplicación. Los pasos seguidos fueron:

1. Subir al servidor los archivos necesarios, script para generación de base de datos, código asociado a servidor web y de señalización.
2. Instalación Node.js y npm.
3. Instalación de MySQL y creación de la base de datos y sus tablas en base a un *script* generado con anterioridad.
4. Ejecutar el servidor web y de señalización con Node.js.

En la Figura 2.35 se muestra la página de inicio de la aplicación luego de su implementación en Google Cloud.

La evidencia de la implementación como vistas asociadas a cada una de las actividades que realiza el prototipo se detallan en el capítulo siguiente.

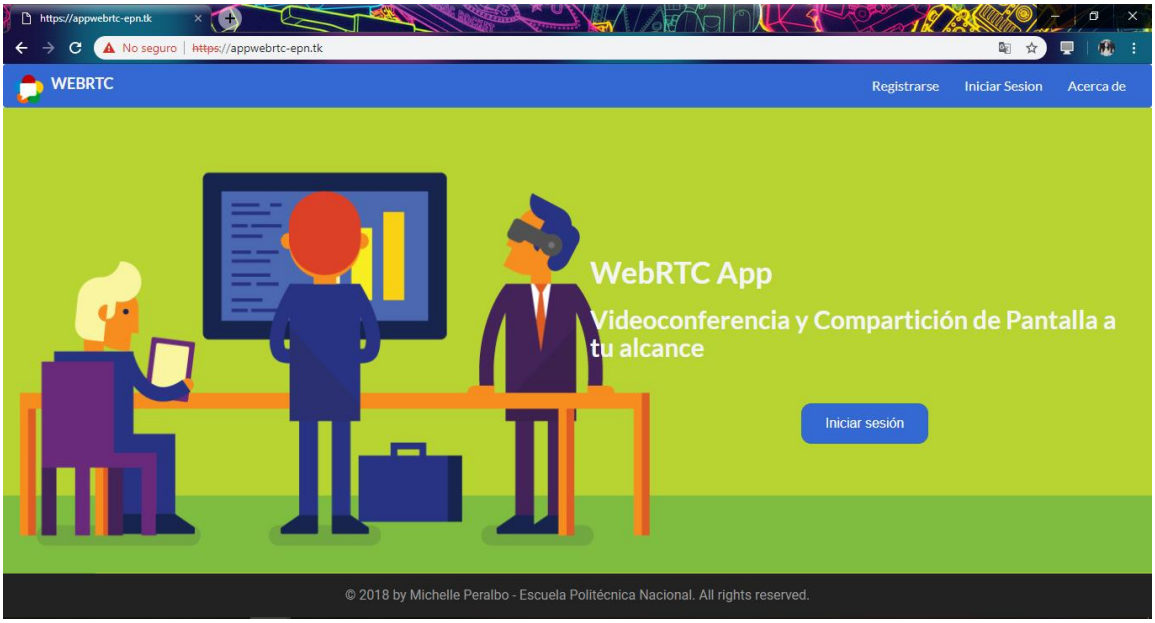

Figura 2.35 Página de inicio ‘https://appwebrtc-epn.tk’

3. RESULTADOS Y DISCUSIÓN

En este capítulo se describen los resultados de las pruebas realizadas al sistema, considerando que la metodología Scrum contempla las pruebas ágiles en paralelo al desarrollo, es decir las historias de usuario se probaron tan pronto fueron completadas quedando el *testing* inmerso en cada uno de los *sprints* realizados.

En cada historia de usuario se definieron criterios de aceptación, para la realización de las pruebas de funcionamiento del sistema que se presentan en esta sección.

3.1 Pruebas de Funcionamiento

Las pruebas de funcionamiento se realizaron para determinar si los requerimientos especificados en las historias de usuario fueron cumplidos en su totalidad.

Las pruebas definidas contemplan los campos descritos a continuación:

- **Requisitos:** descripción de condiciones que desencadenan el requerimiento.
- **Evento:** acción ejecutada por el usuario acorde a los requisitos definidos.
- **Resultado esperado:** comportamiento deseado del software de acuerdo con los requisitos y el evento desencadenado por el usuario.
- **Resultados obtenidos:** evidencia los resultados de la prueba de funcionamiento aplicada.
- **Evaluación:** describe la prueba como exitosa o fallida de acuerdo con el resultado que se haya obtenido.

En esta sección se han dividido estas pruebas de acuerdo con los requerimientos de los *sprints* presentados en el Capítulo 2.

3.1.1 Pruebas sprint 1

Para para las pruebas del *sprint* 1 se tomó en cuenta las historias de usuario que constan en la Tabla 3.1.

Tabla 3.1 Historias de usuario asociadas al *sprint* 1

Id Historia de Usuario	Id Prueba	Título HU
HU-01	PF-01	Administración de usuarios
HU-02	PF-02	Registro de usuarios en la aplicación
HU-05	PF-03	Autenticación en la aplicación

3.1.1.1 Pruebas de funcionamiento para administración de usuarios (PF-01)

A continuación, se muestran las pruebas de funcionamiento de aplicadas a la HU-01: Administración de usuarios, que corresponde a la creación, lectura, modificación y eliminación de usuarios del sistema. Para esta historia de usuario se definieron cuatro tareas principales: creación, listado, edición y eliminación de usuarios.

Requisitos:

- **Crear usuario:** Ninguno.
- **Listar usuarios:**
    - Tener asignado rol administrador.
- **Editar un usuario:**
    - Tener asignado rol administrador.
    - Contar con usuarios registrados en la aplicación.
- **Eliminar un usuario:**
    - Contar con usuarios registrados en la aplicación.

Eventos:

- **Creación de un usuario:**
    1. Introducir en el formulario los datos de un nuevo usuario del sistema.
    2. Intentar guardar un usuario sin llenar completamente los campos asignados.
    3. Intentar guardar un usuario con el mismo correo, o *username* que otro registrado.
    4. Presionar ‘Guardar’ después de ingresar todos los campos requeridos.
- **Listar usuarios:**
    1. Presionar ‘Administrar usuarios’ en la barra de navegación del sistema.
- **Editar un usuario:**
    1. Introducir los datos a actualizarse en el formulario de edición presentado.
    2. Intentar cambiar el campo correo electrónico.
    3. Intentar guardar los cambios con campos incompletos.

4. Presionar ‘Guardar Cambios’ después de ingresar todos los campos requeridos.

- **Eliminar un usuario:**

1. Presionar el botón ‘Eliminar’ asociado a algún usuario registrado.
2. Presionar ‘Cancelar’ frente a la confirmación de eliminación.
3. Presionar ‘Aceptar’ frente a la confirmación de eliminación.

Resultados esperados:

- **Creación de un usuario:**

1. Permitir el ingreso de datos de un nuevo usuario en el sistema.
2. Realizar validaciones de campos vacíos en el formulario y de usuarios duplicados.

- **Listar usuarios:**

1. Permitir visualizar todos los usuarios registrados en la aplicación y los botones de acción para editar y eliminar.

- **Editar un usuario:**

1. Permitir el ingreso de datos en los campos para actualización de información.
2. Realizar validaciones de campos vacíos en el formulario, no permitir cambio en el correo electrónico.
3. Actualizar el usuario en la base de datos.

- **Eliminar un usuario:**

1. Eliminar un usuario al presionar ‘Aceptar’ en la ventana de confirmación.

Resultados obtenidos:

- **Creación de un usuario:**

1. La Figura 3.1 muestra el formulario para ingreso de un nuevo usuario en el sistema.

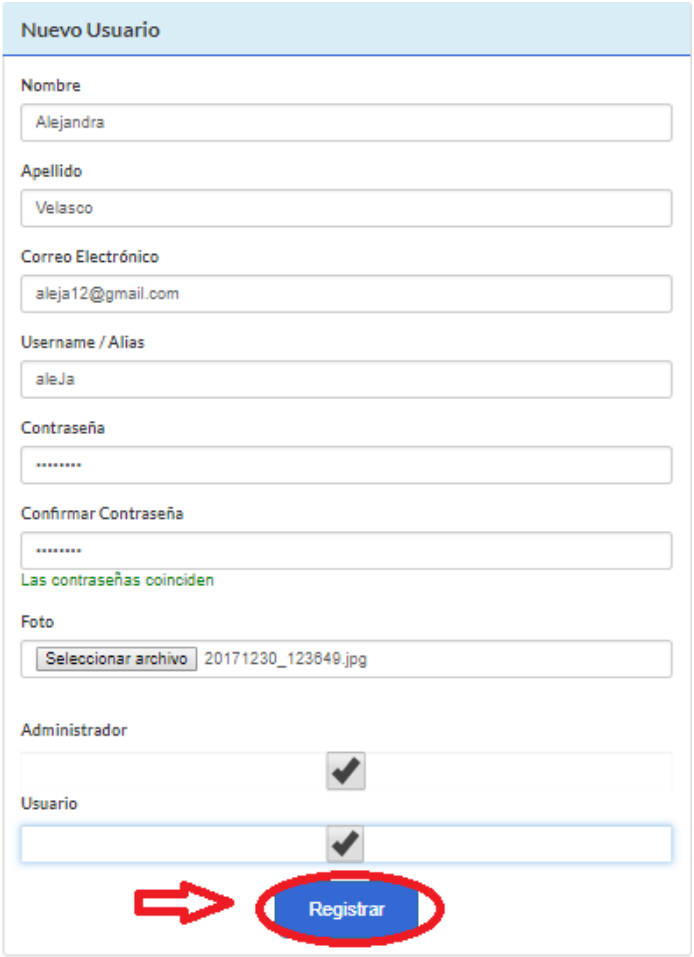

Figura 3.1 Ingreso de datos para creación de un nuevo usuario

En la Figura 3.2 se muestra que el usuario fue registrado correctamente en la base de datos.

idUsuario	firstName	lastName	username	correo
5	Alejandra	Velasco	aleJa	aleja12@gmail.com

idUsuario	idPerfil	idPerfil	descripcion
5	1	1	administrador
5	2	2	usuario

Figura 3.2 Usuario ingresado en la base de datos

2. Las validaciones tanto de campos vacíos como de ingreso de duplicados se observan en la Figura 3.3 y Figura 3.4 respectivamente.

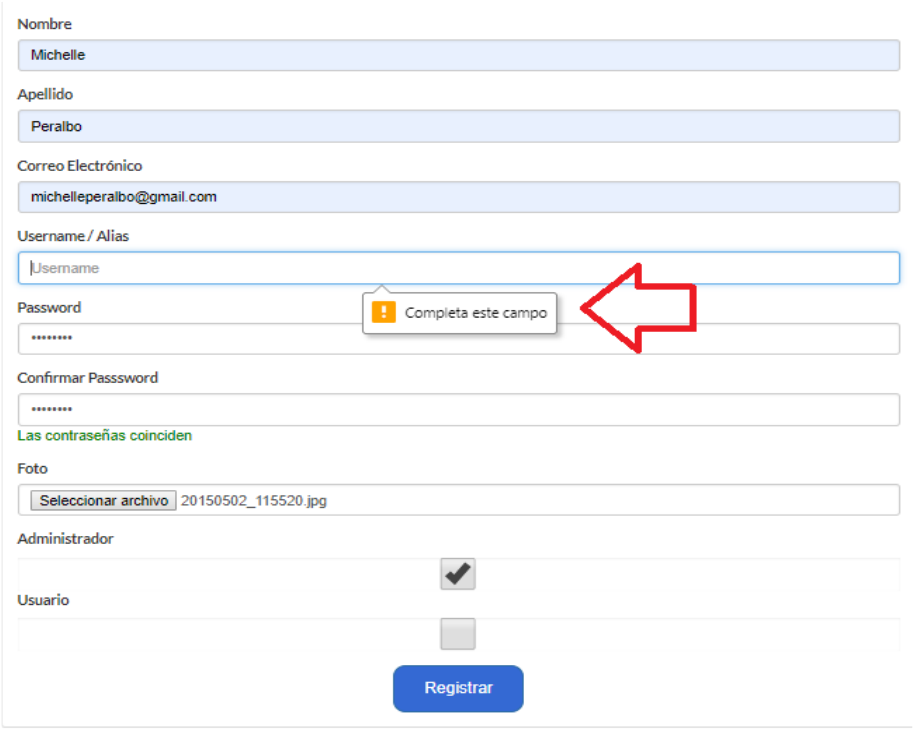

Figura 3.3 Validación de campos incompletos de nuevo usuario

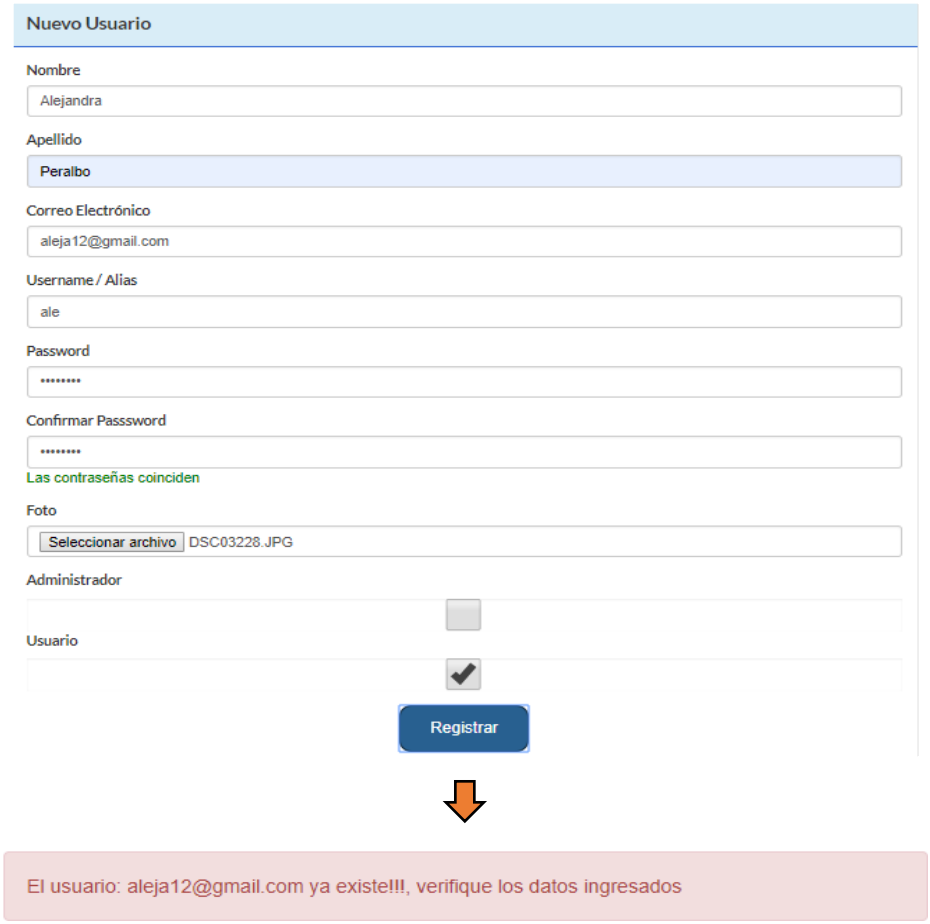

Figura 3.4 Validación de usuarios duplicados

- **Listar usuarios:**

  1. La Figura 3.5 permite visualizar todos los usuarios registrados en la aplicación y los botones de acción para editar y eliminar.

Figura 3.5 Panel de administración de usuarios

- **Editar un usuario:**

  1. La Figura 3.6 muestra el proceso de edición de un usuario de manera exitosa, y la Figura 3.7 muestra el resultado en la base de datos.

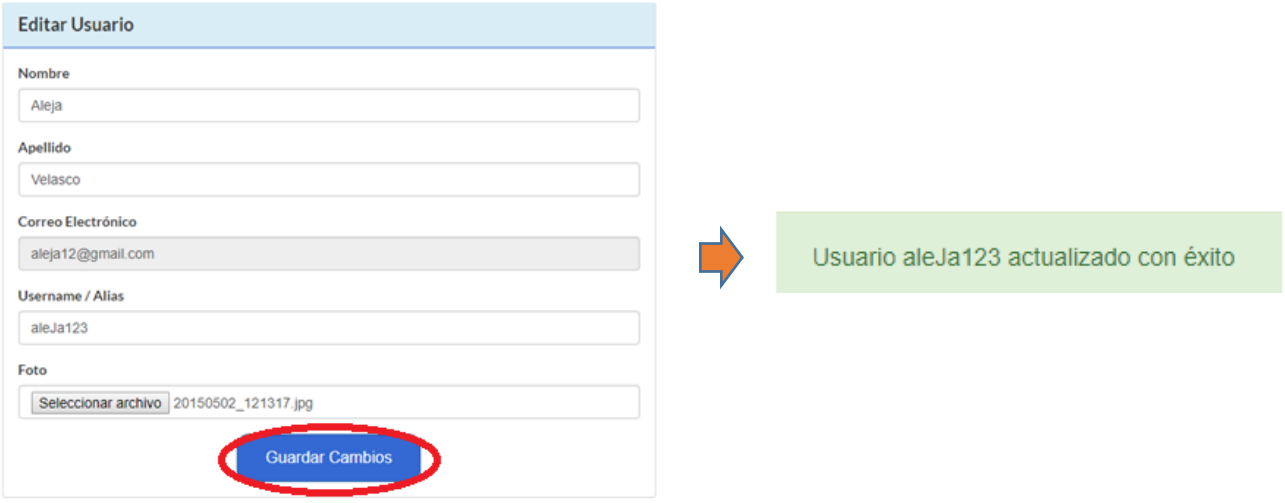

Figura 3.6 Proceso de edición de un usuario

firstName	lastName	username	correo	foto
Aleja	Velasco	aleJa123	aleja12@gmail.com	jpg

Figura 3.7 Usuario actualizado en la base de datos

2. La Figura 3.8 muestra las validaciones del campo correo electrónico y de campos incompletos para el formulario de edición de usuario.

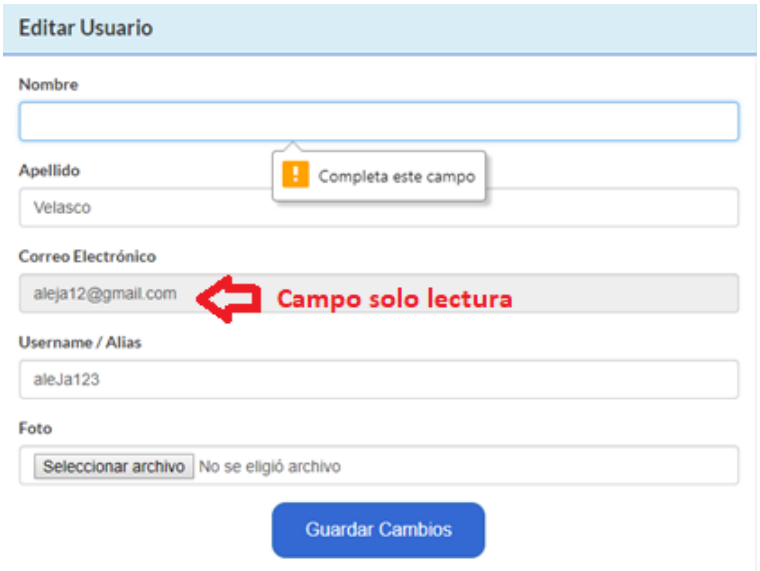

Figura 3.8 Validación de campo correo electrónico y campos incompletos

- **Eliminar un usuario:**

1. La Figura 3.9 muestra el proceso de la eliminación de un usuario.

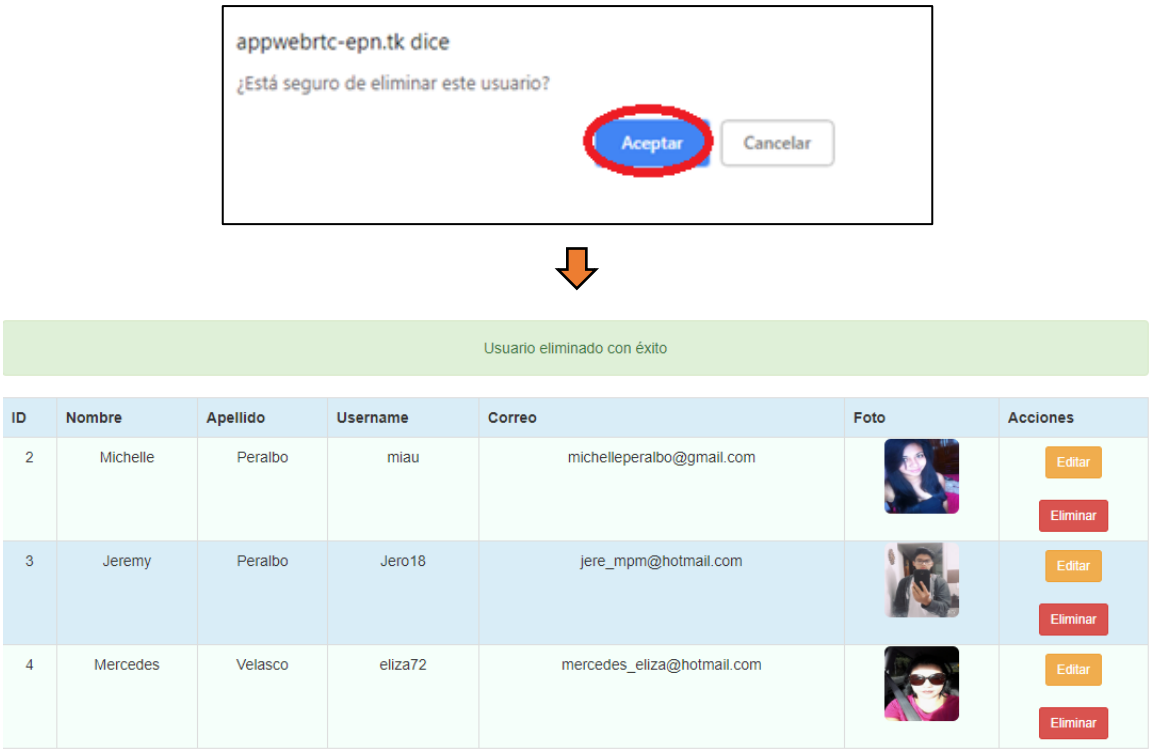

Figura 3.9 Proceso de eliminación de un usuario

Evaluación: Exitosa.

3.1.1.2 Pruebas de funcionamiento del registro de usuarios (PF-02)

A continuación, se muestran las pruebas de funcionamiento de aplicadas a la HU-02: Registro de usuarios en la aplicación.

Requisitos:

- Ingresar a la página web de la aplicación.

Eventos:

- Ingresar al formulario de registro e introducir los datos de un nuevo usuario.
- Intentar guardar un usuario sin llenar completamente los campos asignados.
- Intentar guardar un usuario con el mismo correo, o *username* que otro registrado.

Resultados esperados:

1. Mostrar el formulario de registro y permitir el ingreso los datos del nuevo usuario.
2. Realizar validaciones de campos vacíos en el formulario y de usuarios duplicados.

Resultados obtenidos:

1. La Figura 3.10 muestra el proceso de ingreso al formulario y el registro exitoso de un nuevo usuario. La Figura 3.11 muestra el nuevo usuario en la base de datos.

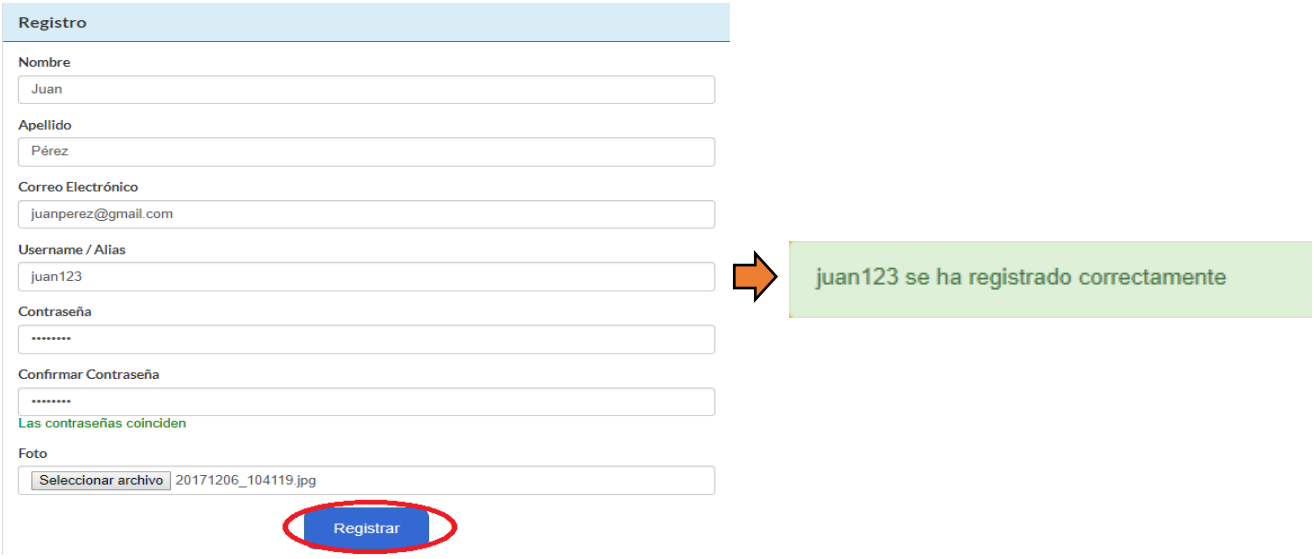

Figura 3.10 Proceso de registro de un nuevo usuario

idUsuario	firstName	lastName	username	correo
10	Juan	Pérez	juan123	juanperez@gmail.com

Figura 3.11 Consulta del usuario insertado en la base de datos

2. Las validaciones de campos vacíos y de usuarios duplicados se muestran en la Figura 3.12 y Figura 3.13 respectivamente.

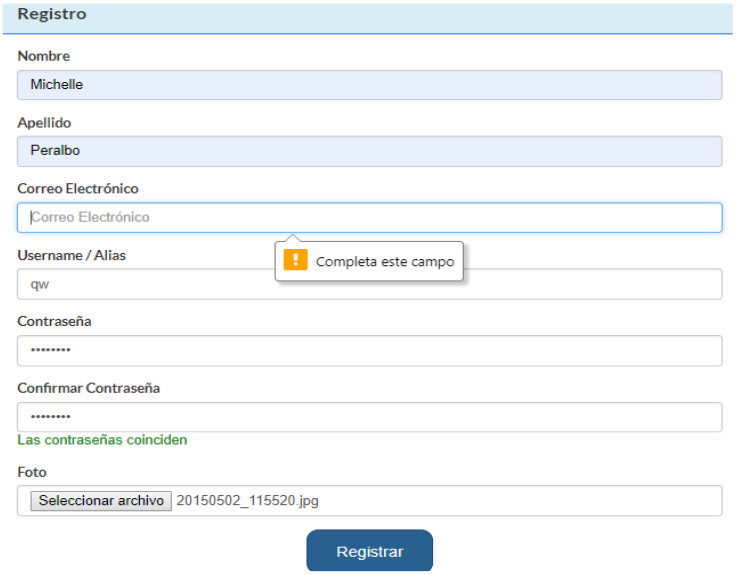

Figura 3.12 Validación de campos incompletos(Registro)

Figura 3.13 Validación de campos duplicados

Evaluación: Prueba exitosa

3.1.1.3 Pruebas de funcionamiento de la autenticación del sistema (PF-03)

En esta sección se detallan las pruebas de funcionamiento aplicadas a la HU-05: Autenticación en la aplicación.

Requisitos:

- Haberse registrado previamente en la aplicación.

Eventos:

- Seleccionar el botón de 'Inicio de sesión' en la página principal de la aplicación.
- Introducir correo y contraseña para iniciar sesión.
- Intentar iniciar sesión sin llenar todos los campos.
- Intentar iniciar sesión con correo equívoco.
- Intentar iniciar sesión con contraseña incorrecta.

Resultados esperados:

1. Mostrar formulario de inicio de sesión.
2. Iniciar sesión correctamente.
3. Validar campos vacíos en el formulario, validar usuarios no registrados y contraseñas equívocas.

Resultados obtenidos:

1. La Figura 3.14 muestra el ingreso del usuario al formulario de inicio de sesión desde la barra de navegación.

Figura 3.14 Formulario de inicio de sesión

2. Para un inicio de sesión exitoso, luego de ingresar el correo y contraseña correctos se accederá a la página principal de la aplicación que muestra la barra de navegación con opciones de acceso al perfil de usuario, al módulo de comunicación y al módulo de ayuda como se muestra en la Figura 3.15.

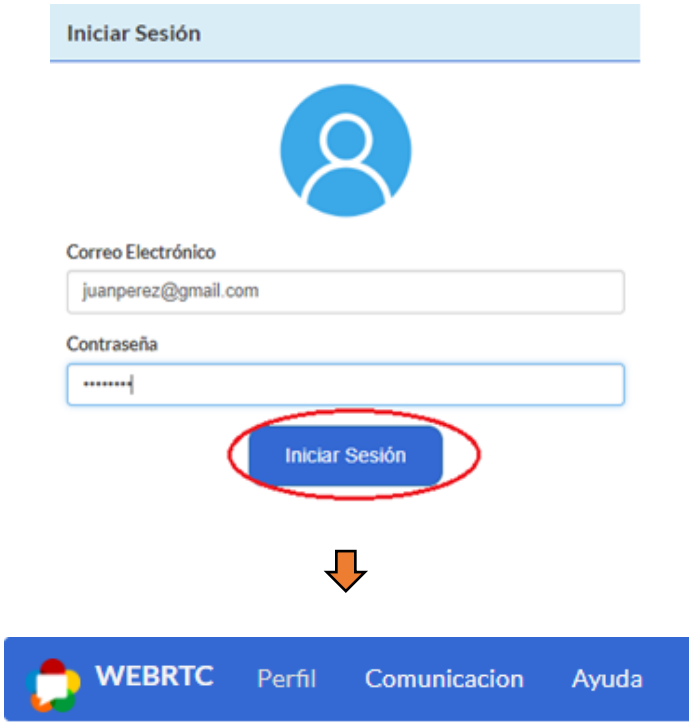

Figura 3.15 Inicio de sesión exitoso

3. Las Figura 3.16, Figura 3.17 y Figura 3.18 muestran las validaciones de campos vacíos, correo no encontrado y contraseña incorrecta.

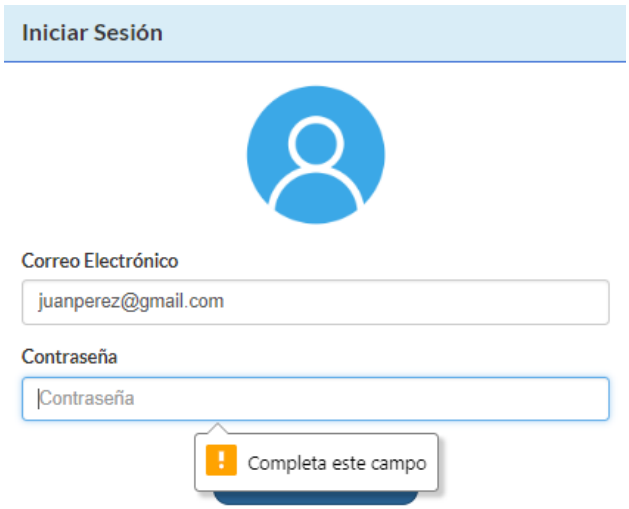

Figura 3.16 Validación de campos incompletos (Inicio de sesión)

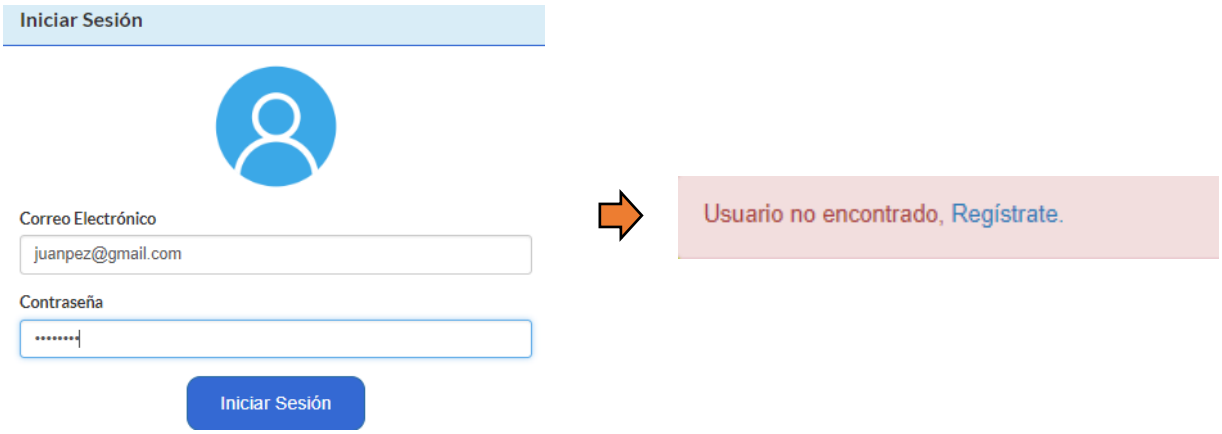

Figura 3.17 Validación de usuario no encontrado

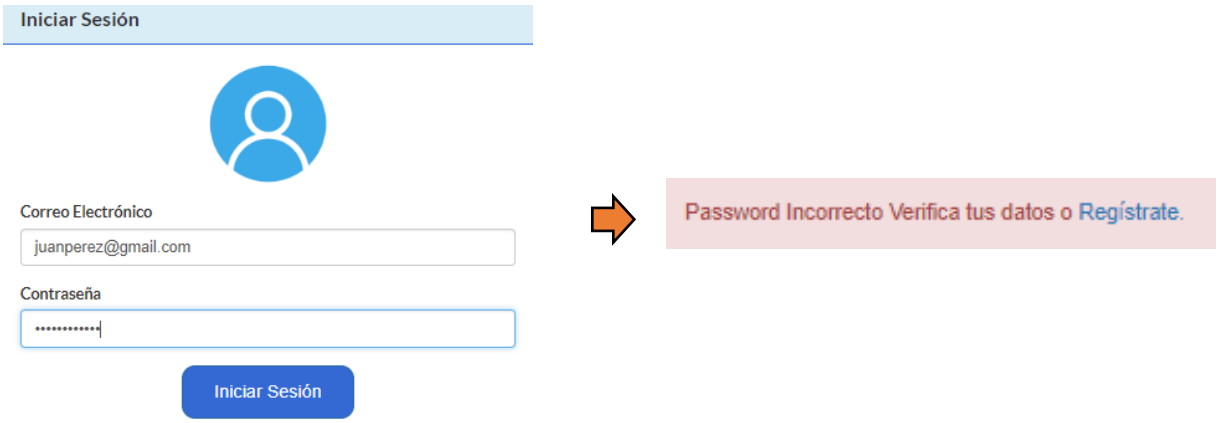

Figura 3.18 Validación de contraseña

Evaluación: Prueba Exitosa

3.1.2 Pruebas sprint 2

Para las pruebas realizadas en el *sprint* 2 se tomó en cuenta las historias de usuario que constan en la Tabla 3.2, que corresponden al módulo de comunicación.

Tabla 3.2 Historias de usuario asociadas al *sprint* 2.

Id Historia de Usuario	Id Prueba	Título HU
HU-07	PF-04	Videoconferencia
HU-08	PF-05	Compartición de pantalla
HU-09	PF-06	Transferencia de mensajes

Para las pruebas de funcionamiento del módulo de comunicación se desplegó el escenario de pruebas presentado en la Figura 1.1.

El servidor web y de base de datos están alojados en una instancia de Google Compute Engine. El navegador utilizado por todos los clientes fue Google Chrome.

La resolución de video de cada participante es VGA (640x480), adaptada a los contenedores de video que se muestran en el navegador, el número de fotogramas por segundo o *frame rate* fue de 30. Estas características fueron asignadas en el constructor de *getUserMedia*.

Tabla 3.3 Características de hardware

Dispositivo	Características
Cliente uno	*Marca:* HP *Procesador:* Intel Core i5 *Memoria RAM:* 8 GB
Cliente dos	*Marca:* ACER *Procesador:* Intel Core i5 *Memoria RAM:* 6GB
Cliente tres	*Marca:* ACER *Procesador:* Core i5 *Memoria RAM:* 8 GB
Cliente cuatro	*Marca:* ASUS *Procesador:* Core i7 *Memoria RAM:* 32GB

Tabla 3.4 Características de software

Dispositivos	Características
Servidor	*OS:* Ubuntu Server 16.04 LTS *Servidor Web:* Node.js
Cliente uno	*OS:* Windows 10 *Versión del navegador:* 71.0.3578.80
Cliente dos	*OS:* Windows 8.1 *Versión del navegador:* 70.0.3538.110
Cliente tres	*OS:* Windows 10 *Versión del Navegador:* 70.0.3538.110
Cliente cuatro	*OS:* Windows 10 *Versión del Navegador:* 71.0.3578.80

En la Tabla 3.3 se muestra las características de hardware de los equipos que se utilizaron para las pruebas de las funciones de: videoconferencia, compartición de pantalla y transmisión de mensajes, mientras que en la Tabla 3.4 se presentan las características de software de los equipos incluyendo la versión del navegador utilizado.

3.1.2.1 Pruebas de funcionamiento de la función de videoconferencia (PF-04)

En esta sección se detallan las pruebas de funcionamiento de aplicadas a la HU-07: Videoconferencia.

Requisitos:

- Haber iniciado sesión en el sistema.
- Tener un servidor de señalización disponible.
- Tener instalada la extensión de Google Chrome Screen Capturing [38].

Eventos:

- Ingresar al módulo de comunicación dando clic en ‘Comunicación’ en la barra de navegación.
- Ingresar el nombre del cuarto para la comunicación, y dar clic en ‘Iniciar video’.
- Intentar ingresar al módulo de comunicación sin tener un servidor de señalización disponible.
- Intentar ingresar al módulo de comunicación sin tener la extensión instalada.
- Intentar iniciar video sin ingresar el nombre del cuarto.

Como usuario que se une a un cuarto de videoconferencia creado por otro usuario:

1. Ingresar satisfactoriamente al cuarto creado por otro usuario.

Resultados esperados:

1. Mostrar la vista del módulo de comunicación, luego de dar clic en la barra de navegación.
2. Como organizador: iniciar la videoconferencia y obtener el video local y una ventana que permita copiar el enlace de la videoconferencia para compartirlo con otras personas.
3. Realizar validaciones de:
    - Servidor de señalización no disponible.

- Extensión no instalada.
- Ingreso del nombre del cuarto de comunicación.

Como usuario que se une a un cuarto de videoconferencia creado por otro usuario:

1. Permitir a los usuarios secundarios unirse a la comunicación.

Resultados obtenidos:

1. La Figura 3.19 muestra el ingreso a la vista del módulo de comunicación.

Figura 3.19 Ingreso al módulo de comunicación

2. Como organizador: La Figura 3.20 muestra el proceso de inicio de la videoconferencia y la obtención exitosa del video local.

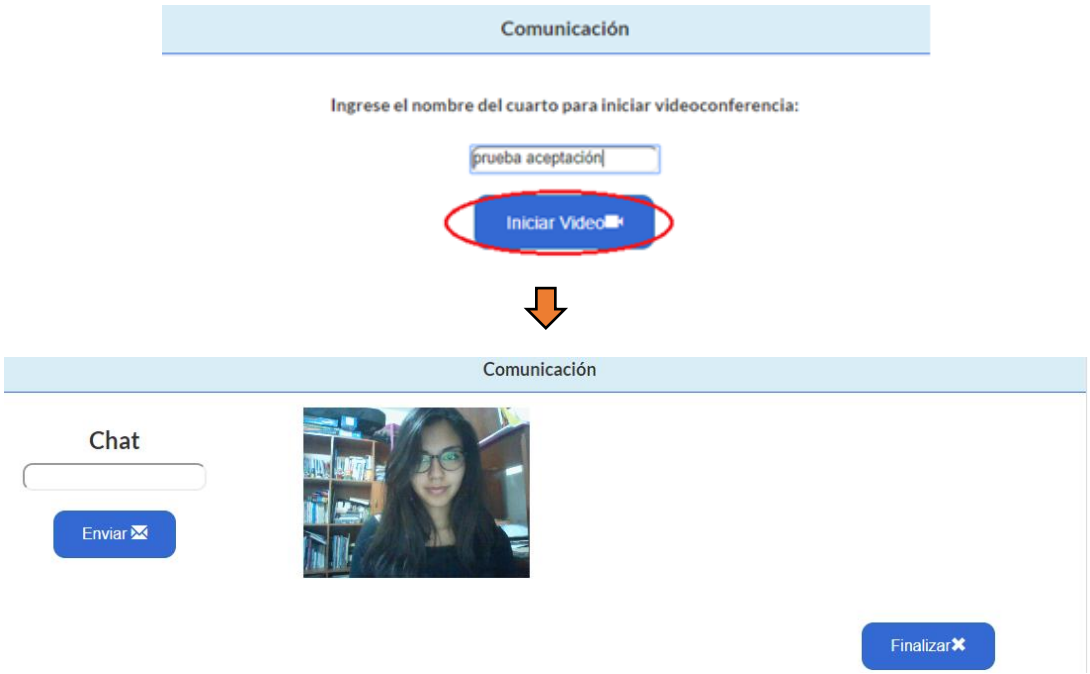

Figura 3.20 Inicio de videoconferencia

La Figura 3.21 muestra la ventana para compartir el enlace a más participantes.

Figura 3.21 Ventana de ayuda para compartir el enlace generado

3. La Figura 3.22 muestra la respuesta del sistema cuando el servidor de señalización no se encuentra disponible o falla la conexión entre el cliente y el servidor; en este caso no permitirá iniciar la comunicación, la Figura 3.23 muestra el mensaje que visualiza el usuario al no tener instalada la extensión que es un requisito para la funcionalidad de compartición de pantalla y la Figura 3.24 muestra la validación de ingreso del nombre del cuarto de videoconferencia.

Figura 3.22 Validación de servidor de señalización

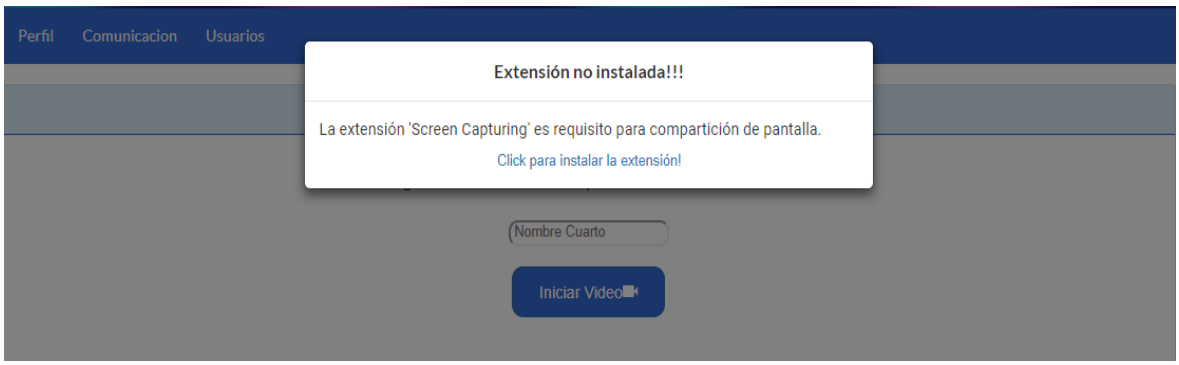

Figura 3.23 Validación de instalación de extensión

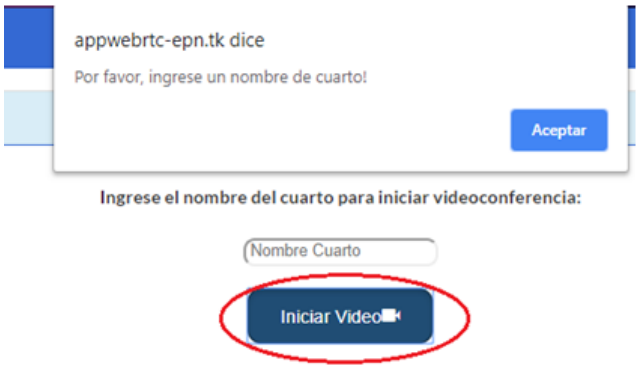

Figura 3.24 Validación de nombre de cuarto de comunicación

Usuario que se une a un cuarto de videoconferencia creado por otro usuario:

1. La Figura 3.25 muestra la respuesta la opción de unirse a la videoconferencia iniciada por otro usuario. En la Figura 3.26 se muestra a cuatro usuarios participando de la función de videoconferencia.

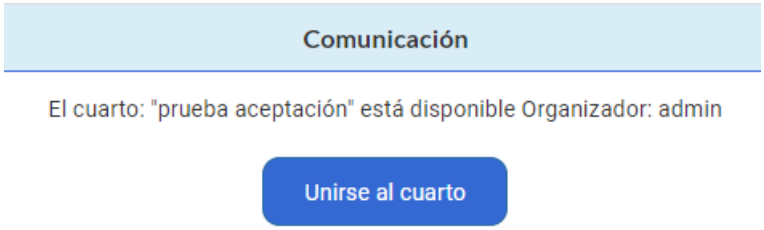

Figura 3.25 Vista para unirse a un cuarto creado

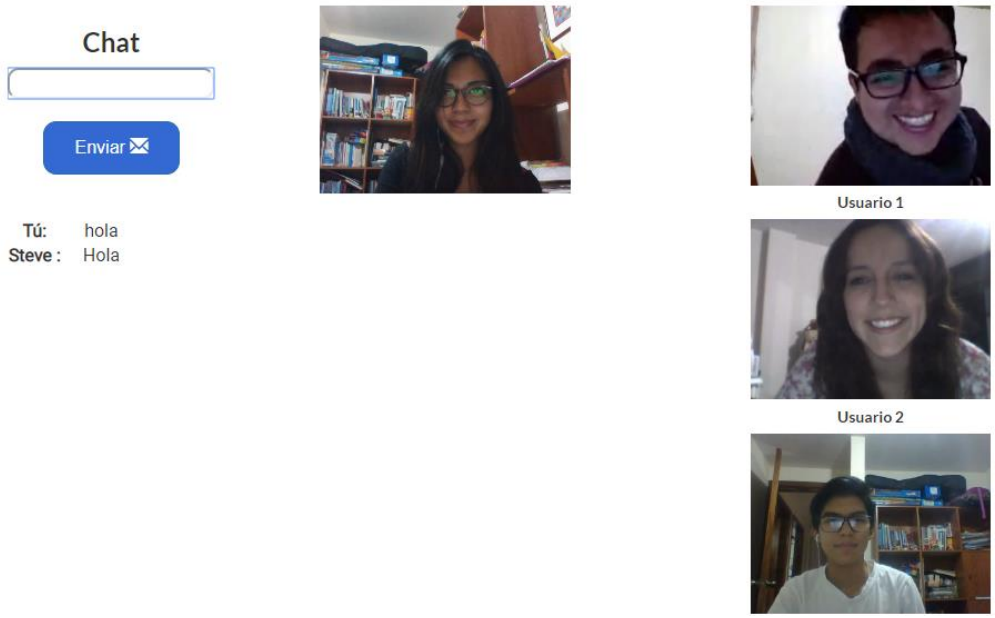

Figura 3.26 Función de videoconferencia con cuatro participantes

Evaluación: Prueba Exitosa

3.1.2.2 Pruebas de funcionamiento para compartición de pantalla (PF-05)

En esta sección se detallan las pruebas de funcionamiento aplicadas a la HU-08: Compartición de pantalla.

Requisitos:

- Haber iniciado sesión.
- Ser el organizador de la videoconferencia.
- Tener instalada la extensión Screen Capturing [38].

Eventos:

- Dar clic en el botón ‘Compartir pantalla’.
- Detener compartición de pantalla mediante los botones asignados y volver a la función de video.

Resultado esperado:

1. Poder compartir pantalla a los usuarios secundarios.
2. Permitir terminar compartición de pantalla, y volver a la función de video mediante cámara web.

Resultados obtenidos:

1. La Figura 3.27 muestra el proceso que lleva a cabo el organizador de la videoconferencia para compartir su pantalla al resto de participantes. La Figura 3.28 muestra la recepción exitosa de la pantalla compartida por el organizador.

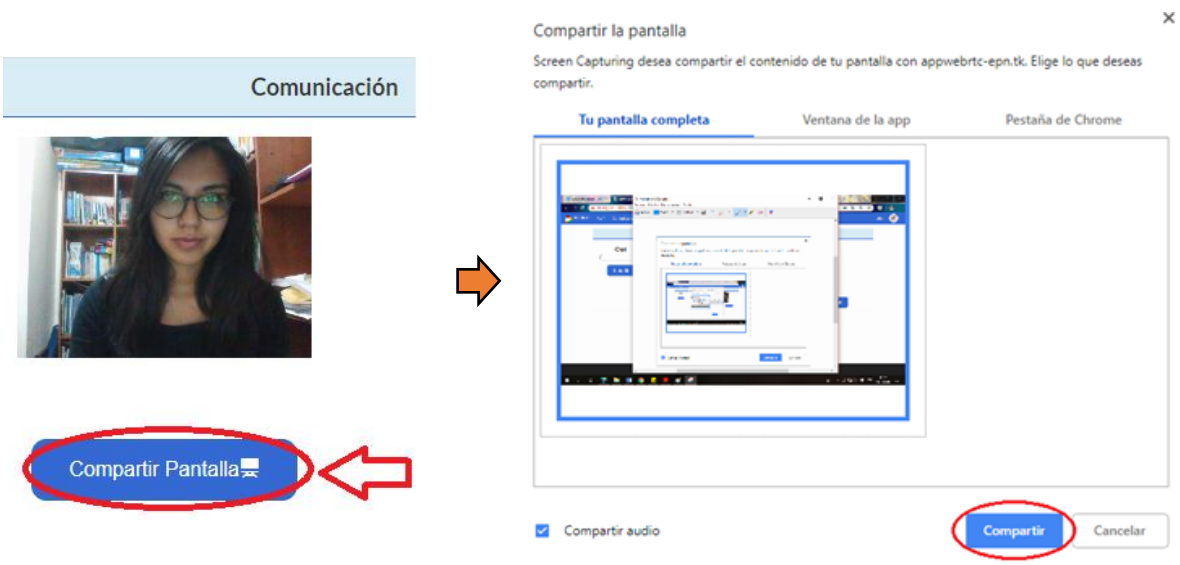

Figura 3.27 Ventana de compartición de pantalla

Figura 3.28 Recepción del video correspondiente a compartición de pantalla.

2. En la Figura 3.29 se muestran los botones asignados para terminar compartición de pantalla, y volver a la función de video mediante cámara web.

Figura 3.29 Opciones para detener compartición de pantalla

Evaluación: Prueba exitosa.

3.1.2.3 Pruebas de funcionamiento de transferencia de mensajes (PF-06)

En esta sección se detallan las pruebas de funcionamiento aplicadas a la HU-09: Transferencia de mensajes.

Requisitos:

- Haber iniciado sesión en la aplicación.
- Ser parte de un cuarto de videoconferencia.

Eventos:

- Mostrar el panel de mensajes al iniciar videoconferencia.
- Ingresar un mensaje en el cuadro de texto y enviarlo.
- Recibir mensajes de otros participantes de la videoconferencia.

Resultado esperado:

1. Mostrar el panel de mensajes al iniciar videoconferencia.
2. Enviar mensajes.
3. Recibir mensajes.

Resultados obtenidos:

1. La Figura 3.30 muestra el panel de mensajes al iniciar videoconferencia.

Figura 3.30 Panel para envío de mensajes

2. La Figura 3.31 muestra la función de envío de mensajes.

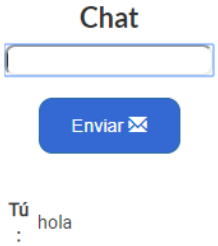

Figura 3.31 Envío de mensajes

3. La Figura 3.32 muestra la recepción de mensajes.

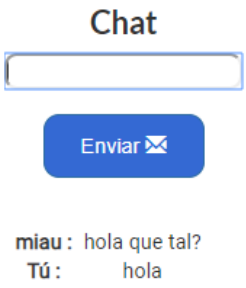

Figura 3.32 Recepción de mensajes

Evaluación: Prueba Exitosa.

3.1.3 Pruebas sprint 3

Para las pruebas del *sprint* 3 se tomó en cuenta las historias de usuario que constan en la Tabla 3.5.

Tabla 3.5 Historias de usuario asociadas al *sprint* 3

Id Historia de Usuario	Id Prueba	Título HU
HU-10	PF-07	Información de desarrollo de la aplicación y manual de usuario
HU-04	PF-08	Administración de roles

3.1.3.1 Pruebas de funcionamiento al módulo de ayuda (PAF07)

En esta sección se describen las pruebas de funcionamiento de la HU-10: Información de la aplicación y manual de usuario.

Requisitos:

- Haber iniciado sesión en la aplicación.
- Tener asignado el rol usuario.

Eventos:

- Acceder al módulo de ayuda y sus subsecciones por medio de la barra de navegación.

Resultados esperados:

1. Mostrar en la barra de navegación la opción de ‘Ayuda’.
2. Desplegar información del diagrama del prototipo.
3. Desplegar información de las características principales del prototipo.
4. Desplegar información de los tipos de usuario existentes en el sistema.
5. Desplegar el manual de usuario de la aplicación.

Resultados obtenidos:

1. La Figura 3.33 muestra la opción de ayuda en la barra de navegación.

Figura 3.33 Módulo de ayuda

2. La Figura 3.34 muestra el despliegue de la información del diagrama del prototipo.

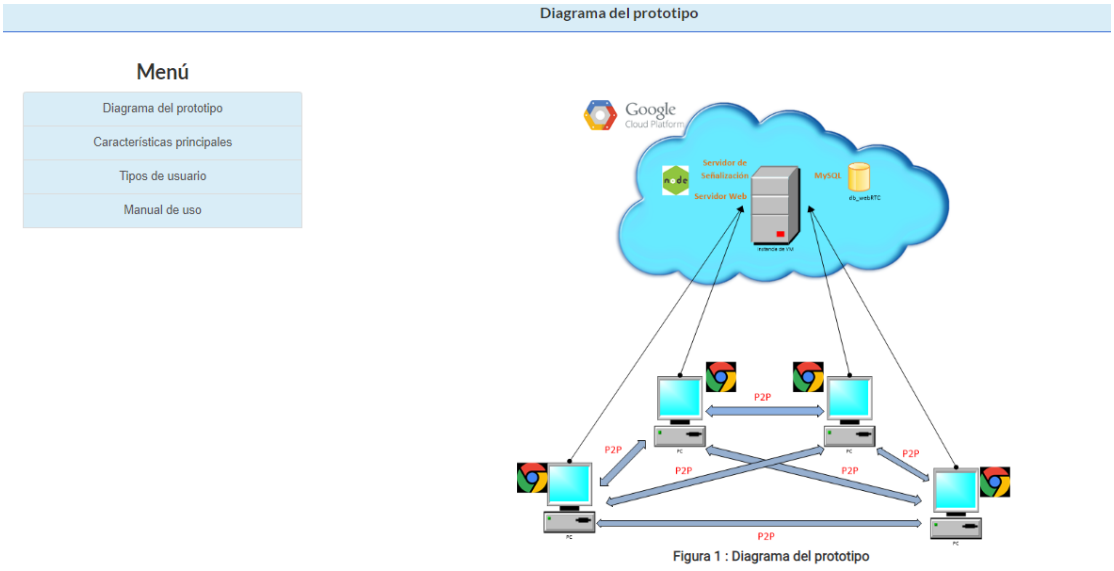

Figura 3.34 Información de diagrama del prototipo

3. La Figura 3.35 muestra el despliegue de la información de las características principales del prototipo.

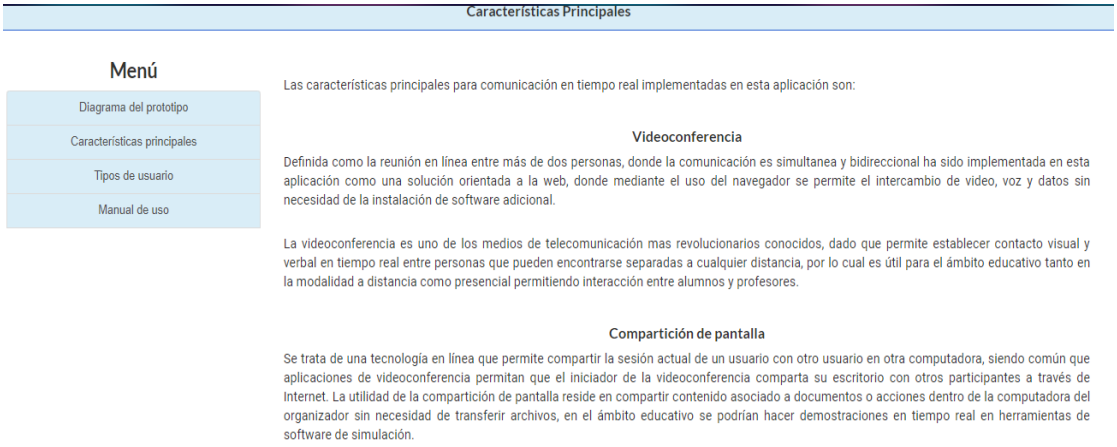

Figura 3.35 Características del prototipo

4. La Figura 3.36 muestra el despliegue de la información de los tipos de usuario existentes en el sistema.

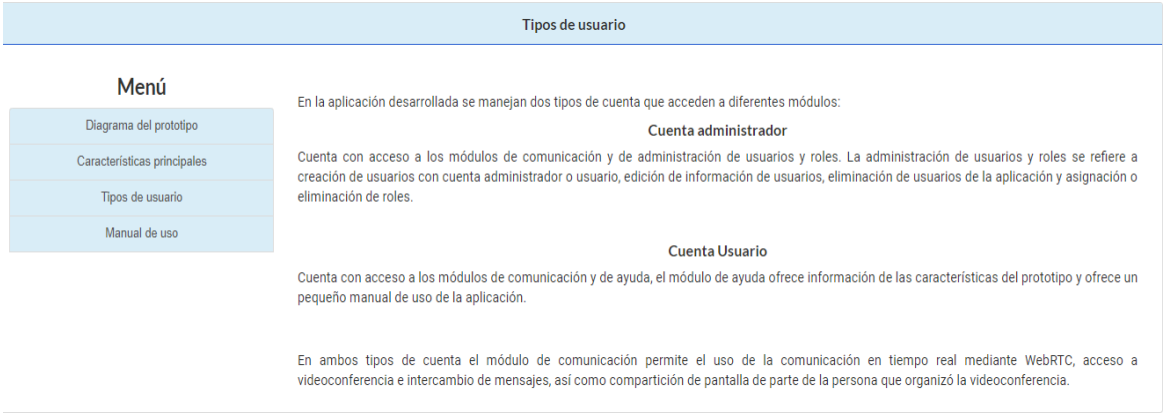

Figura 3.36 Tipos de usuario

5. La Figura 3.37 muestra el despliegue del manual de usuario de la aplicación. Toda la información contenida en el manual de usuario se encuentra en el ANEXO A.

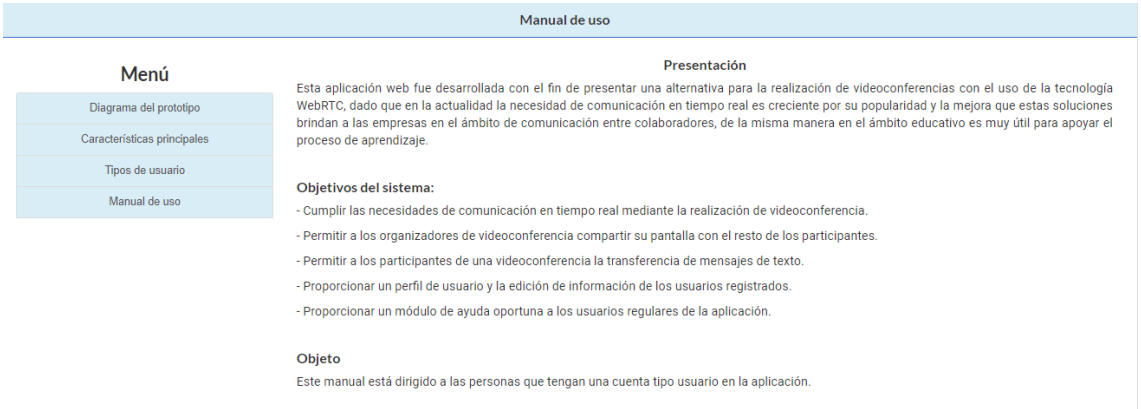

Figura 3.37 Manual de usuario

Evaluación: Prueba exitosa

3.1.3.2 Pruebas de funcionamiento de la administración de roles (PF-08)

En esta sección se muestran las pruebas de funcionamiento aplicadas a la HU-04: Administración de roles.

Requisitos:

- Tener rol administrador asignado.

Eventos:

- **Listar roles:**
    - Acceder al panel de administración de roles de usuario.
- **Añadir rol:**
    - Seleccionar un rol para añadir asociado a un usuario específico.
    - Presionar añadir un nuevo rol.
    - Aceptar añadir un nuevo rol.
- **Eliminar rol:**
    - Presionar el botón ‘Eliminar’ desde el panel de administración de roles.
    - Acceder al formulario de eliminación.
    - Seleccionar el rol a eliminar.
    - Presionar el botón eliminar.

Resultados esperados:

- **Listar roles:**
    1. Acceder al panel de administración de roles desde de la barra de navegación.
- **Añadir rol:**
    1. Seleccionar un rol de usuario y añadirlo luego de confirmarlo.
- **Eliminar rol:**
    1. Eliminar un rol correctamente.

Resultados obtenidos:

- **Listar roles:**
    1. La Figura 3.38 muestra el acceso al panel de administración de roles desde la barra de navegación.

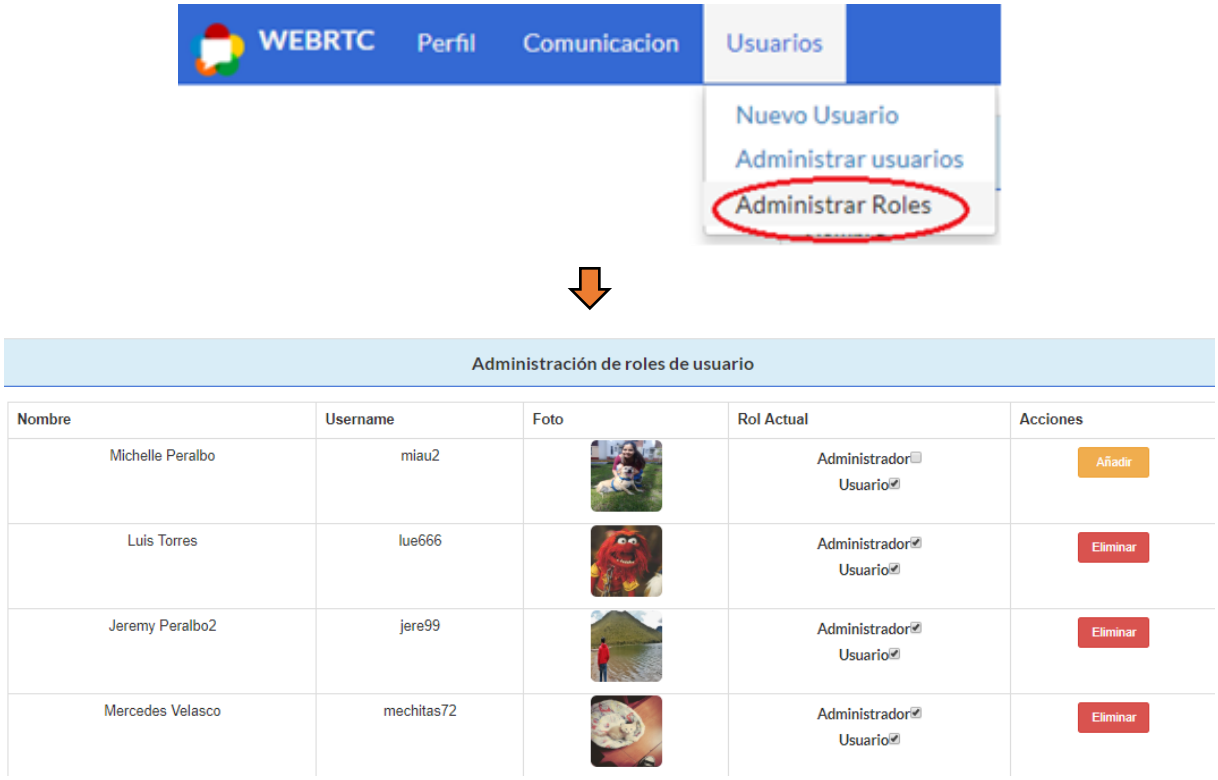

Figura 3.38 Panel de administración de roles

- **Añadir rol:**

1. La Figura 3.39 muestra el proceso para añadir un rol de usuario.

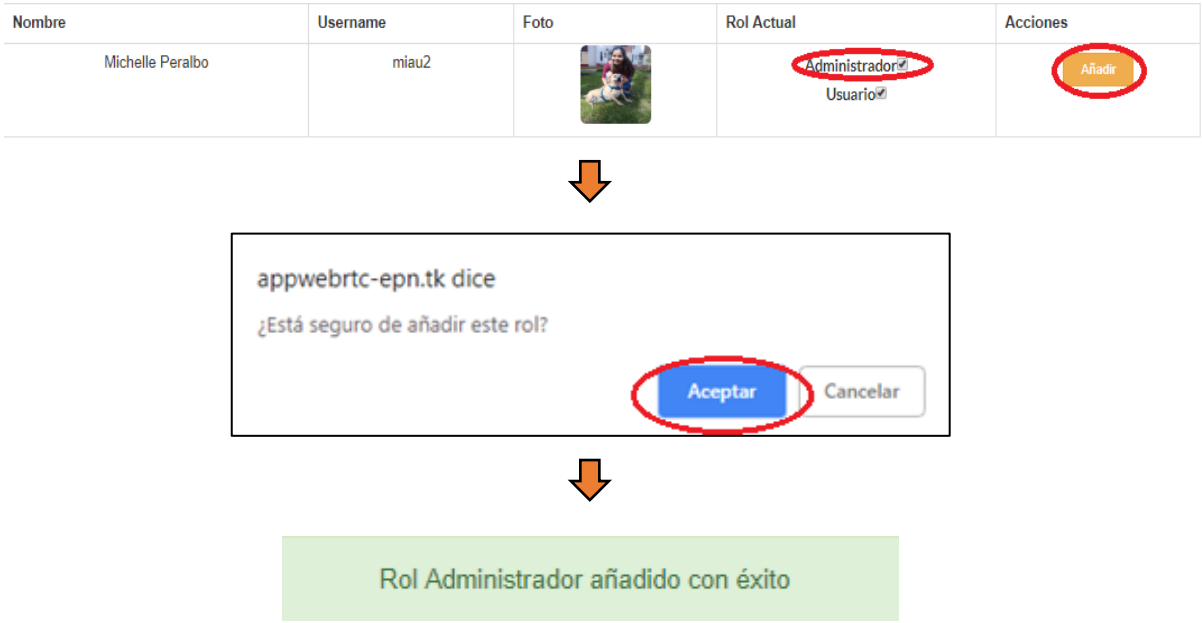

Figura 3.39 Proceso para añadir un rol

- **Eliminar rol:**

1. La Figura 3.40 muestra el proceso de eliminación de un rol de manera exitosa. Por su parte la Figura 3.41 muestra el resultado de la consulta a la base datos para comprobar que el rol ha sido eliminado.

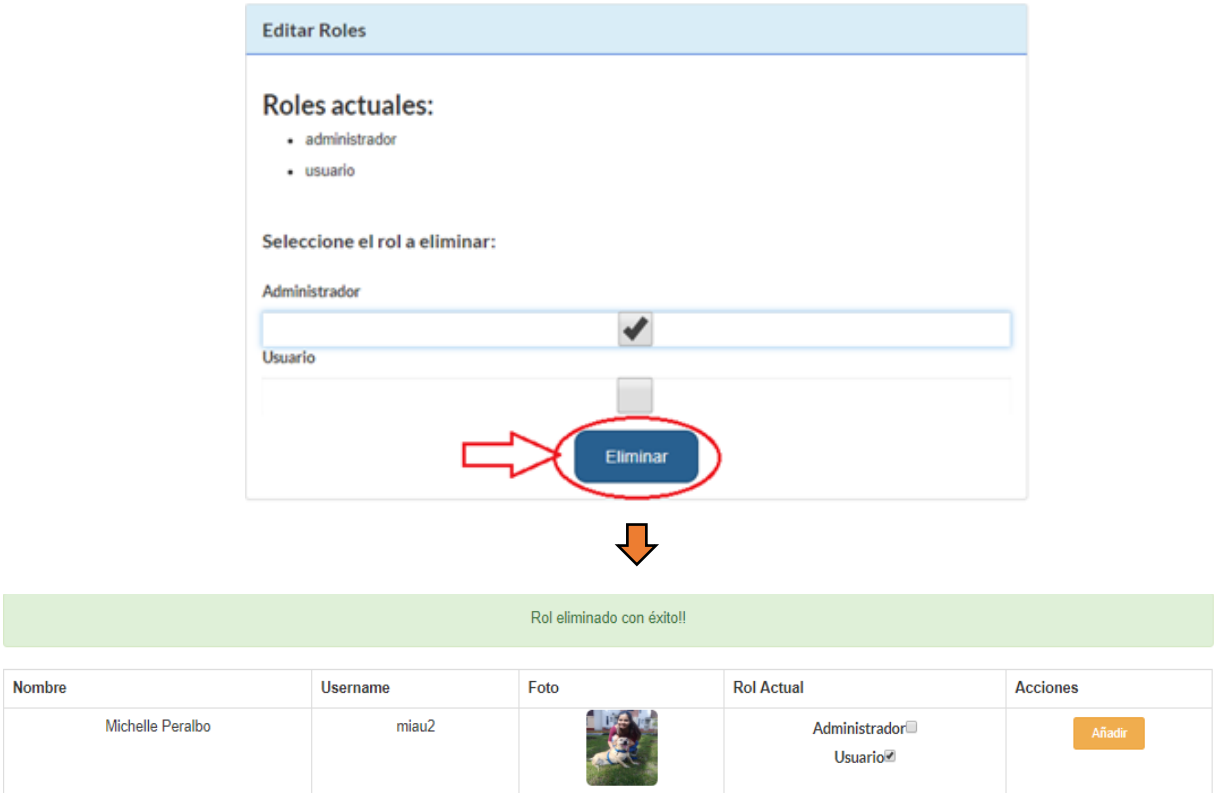

Figura 3.40 Eliminación de un rol

```
+-----------+----------+--------------------------+-------------+
| firstName | lastName | correo                   | descripcion |
+-----------+----------+--------------------------+-------------+
| Michelle  | Peralbo  | michelleperalbo@gmail.com | usuario     |
+-----------+----------+--------------------------+-------------+
```

Figura 3.41 Búsqueda del rol eliminado

Evaluación: Prueba exitosa

3.1.4 Pruebas sprint 4

Para las pruebas del *sprint* 4 se tomó en cuenta las historias de usuario que constan en la Tabla 3.6.

Tabla 3.6 Historias de usuario asociadas al *sprint* 4

Id Historia de Usuario	Id Prueba	Título HU
HU-03	PF-09	Edición de información personal y contraseña
HU-06	PF-10	Mostrar perfil de usuario
HU-11	PF-11	Home Page/ Acerca De

3.1.4.1 Pruebas de funcionamiento para edición de información personal y contraseña (PF-09)

Esta sección describe las pruebas de funcionamiento aplicadas a la HU-03: Edición de información personal y contraseña.

Requisitos:

- Haber iniciado sesión en el sistema.

Eventos:

- **Edición de información personal:**
    - Seleccionar la opción ‘Configuración’ de la barra de navegación.
    - Llenar todos los campos requeridos y guardar los cambios.
    - Tratar de guardar los cambios sin llenar todos los campos.
    - Tratar de editar el campo correo electrónico.
- **Edición de contraseña:**
    - Seleccionar la opción de cambiar contraseña en la vista de configuración de usuario.
    - Llenar los campos requeridos para el cambio de contraseña.
    - Intentar guardar los cambios con campos incompletos.
    - Introducir una contraseña actual errónea.
    - Introducir una contraseña nueva igual a la actual.

Resultados esperados:

- **Edición de información personal:**
    1. Ingresar al formulario de edición de datos luego de presionar ‘Configuración’ en la barra de navegación y editar un usuario correctamente.

2. Realizar validaciones de campos vacíos y no permitir editar el correo electrónico.

- **Edición de contraseña:**

1. Ingresar al formulario de cambio de contraseña y cambiar contraseña de manera exitosa.
2. Realizar validaciones de campos vacíos, de contraseña actual errónea y de contraseña nueva igual a la anterior.

Resultados obtenidos:

- **Edición de información personal:**

1. La Figura 3.42 muestra el proceso de edición de un usuario.

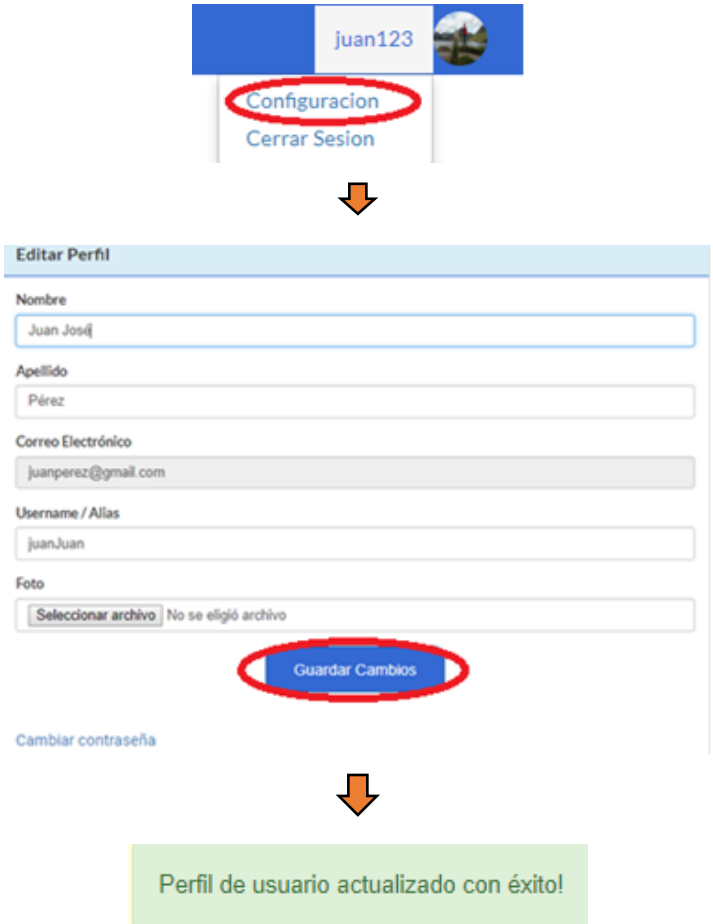

Figura 3.42 Actualización de datos personales

2. La Figura 3.43 corresponde a las validaciones de campos vacíos y de correo electrónico no editable.

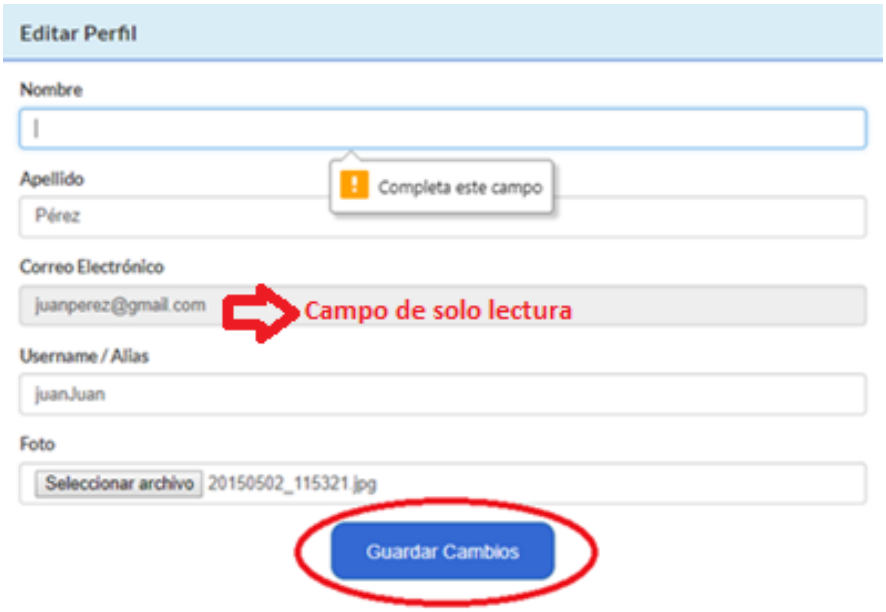

Figura 3.43 Validación de campos vacíos y de correo electrónico

- **Edición de contraseña:**

1. La Figura 3.44 muestra el proceso de cambio de contraseña.

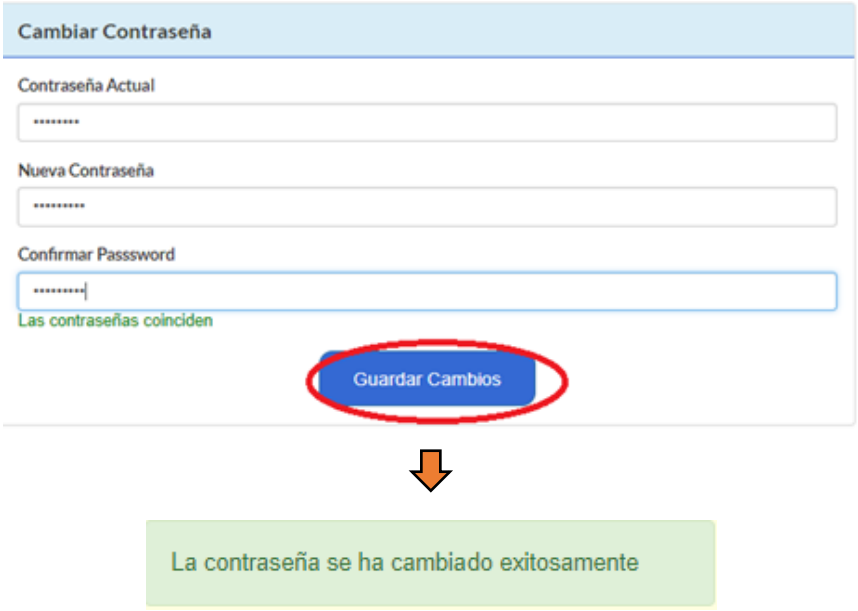

Figura 3.44 Proceso para cambio de contraseña

2. Las Figura 3.45, Figura 3.46 y Figura 3.47 muestran las validaciones de campos vacíos en el formulario, contraseña actual errónea y contraseña nueva igual a la anterior.

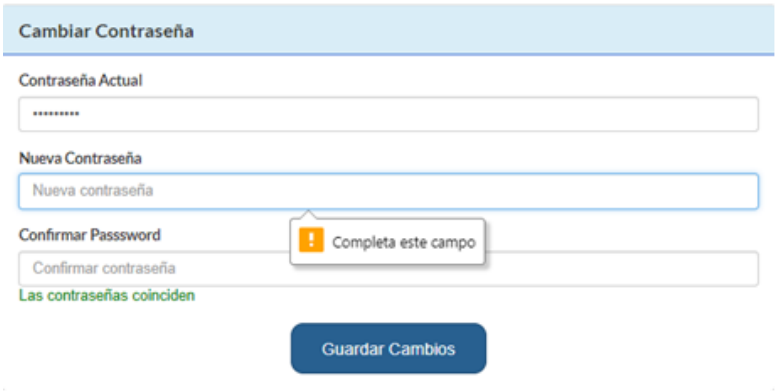

Figura 3.45 Validación de campos vacíos

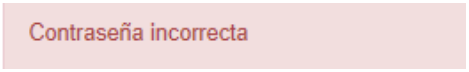

Figura 3.46 Validación de contraseña actual incorrecta

Figura 3.47 Validación de contraseña nueva

Evaluación: Prueba exitosa

3.1.4.2 Pruebas de funcionamiento para perfil de usuario (PF-10)

Esta sección muestra las pruebas de funcionamiento aplicadas a la HU-06 correspondiente a mostrar el perfil de un usuario.

Requisitos:

- Haber iniciado sesión en la aplicación.

Eventos:

- Seleccionar la opción ‘Perfil’ desde la barra de navegación.

Resultados esperados:

1. Mostrar el perfil de usuario al presionar ‘Perfil’ en la barra de navegación.

Resultados obtenidos:

1. La Figura 3.48 muestra el acceso al perfil de usuario.

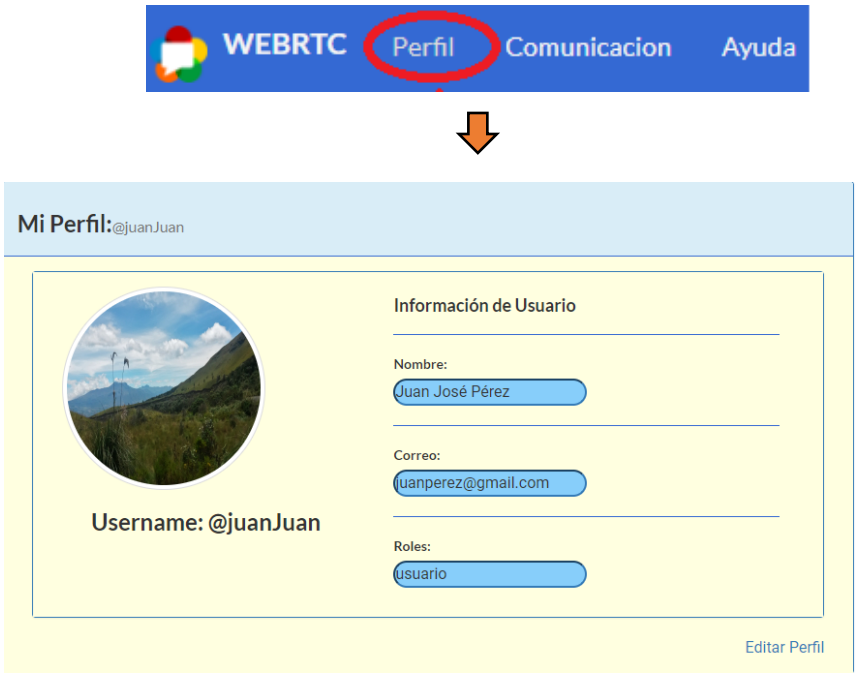

Figura 3.48 Perfil de usuario

Evaluación: Prueba exitosa.

3.1.4.3 Pruebas de funcionamiento para Home Page y ‘Acerca de’ (PF-11)

En esta sección se presentan las pruebas de funcionamiento aplicadas a la HU-11: Home Page/ Acerca de.

Requisitos: Ninguno

Eventos:

- **Home Page:**
  - Ingresar a la página principal de la aplicación.
- **Acerca de:**
  - Seleccionar la opción ‘Acerca de:’ en la barra de navegación inicial.

Resultados esperados:

- **Home Page:**

  1. Obtener la vista de la página de inicio al solicitar la aplicación.

- **Acerca de:**

  1. Mostrar información general de la aplicación al seleccionar la opción ‘Acerca de’.

Resultados obtenidos:

- **Home Page:**

  1. La Figura 3.49 muestra la página de inicio de la aplicación.

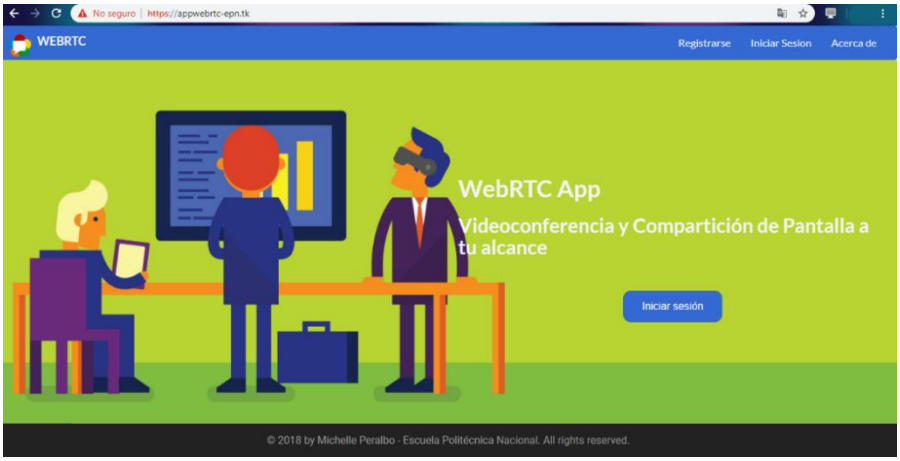

Figura 3.49 Página de inicio de la aplicación

- **Acerca de:**

  1. La Figura 3.50 muestra información general de la aplicación al seleccionar la opción ‘*Acerca de’*.

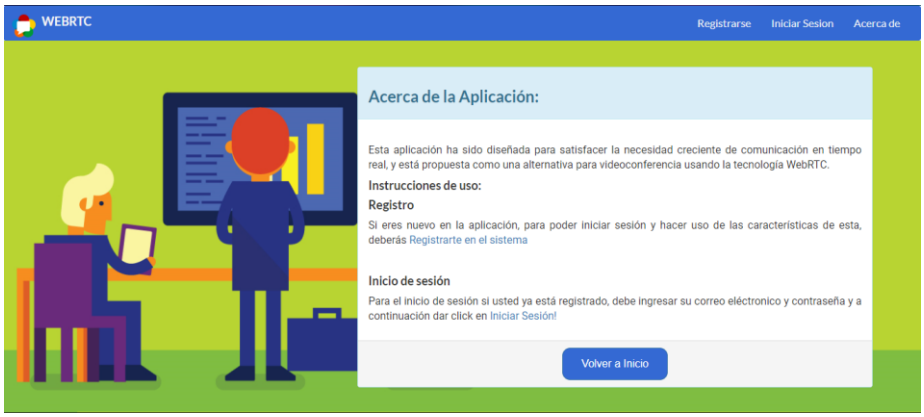

Figura 3.50 ‘Acerca de’ la aplicación

Evaluación: Prueba exitosa.

3.2 Análisis de los Resultados

Los resultados que arrojaron las pruebas fueron satisfactorios, debido a que el prototipo cumple con todos los criterios de aceptación definidos en cada una de las historias de usuario presentes en el *product backlog*, por lo cual los requerimientos planteados en este proyecto se han cumplido de la manera esperada. La Tabla 3.7 detalla el cumplimiento de los ítems del *product backlog*.

Tabla 3.7 Cumplimiento de los ítems del *product backlog*

Prioridad	Historia de Usuario	Título historia de usuario	Cumple	Horas
Alta	HU-00	Creación Base de datos, instalación de IDE de desarrollo.	Si	15
	HU-01	Administración de usuarios	Si	20
	HU-02	Registro de usuarios en la aplicación	Si	5
	HU-05	Autenticación en la aplicación	Si	10
Alta	HU-07	Videoconferencia	Si	40
	HU-08	Compartición de pantalla	Si	20
	HU-09	Transferencia de mensajes	Si	5
Alta	HU-10	Información de desarrollo de la aplicación y manual de usuario	Si	25
Media	HU-04	Administración de roles	Si	25
Media	HU-03	Edición de información personal y contraseña	Si	20
	HU-06	Mostrar perfil de usuario	Si	5
Baja	HU-11	Home Page/ Acerca De	Si	10
			TOTAL	**200**

Las correcciones del prototipo después de la fase de pruebas estuvieron relacionadas con la mejora de la experiencia de usuario.

En el módulo de comunicación, se pulieron detalles respecto a el tamaño de los contenedores donde se muestran los elementos de video, se implementó etiquetas para identificar al organizador de la comunicación y a los usuarios secundarios. Otra de las mejoras en este módulo fue la ventana para compartir el enlace con otros usuarios, en la cual se implementó la opción de enviar por correo el enlace generado.

En el módulo de administración de usuarios y roles se mejoró la presentación de las tablas que contenían los usuarios a administrar, así como los botones de acción sobre los mismos.

Finalmente en el módulo de ayuda se dividió la información en varias secciones para una mejor visualización y se implementó un submenú para ingresar a ellas.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La aplicación web desarrollada permite realizar videoconferencia y compartición de pantalla mediante el uso de la API de WebRTC. Además, cuenta con una base de datos para el almacenamiento de usuarios y sus roles, consiguiendo de esta manera el objetivo general de este proyecto técnico.

- El diseño de los módulos del sistema se llevó a cabo mediante el modelado tanto de clases como de base de datos utilizando los diagramas UML correspondientes. También mediante la agrupación de los requerimientos que compartieron funcionalidades similares como las de comunicación, administración de usuarios, administración de roles, autenticación y de ayuda.

- Las características de los componentes que conforman el prototipo fueron desarrolladas en base a requerimientos definidos en el alcance del proyecto, y que se detallaron en forma de historias de usuario.

- El uso de la metodología Scrum facilitó el desarrollo del prototipo debido a la mejora continua en cada una de las iteraciones y que las pruebas de funcionamiento de los requerimientos planteados se realizaron en cada uno de los *sprints*.

- El módulo de ayuda implementado ha permitido brindar información relevante acerca de la aplicación, la tecnología involucrada, las características de comunicación en tiempo real con las que cuenta, los tipos de usuario que maneja y un manual que facilitó al usuario la navegación en la aplicación.

- El servidor de señalización utilizado para el módulo de comunicación hace uso de la librería Socket.IO, para permitir que la aplicación WebRTC configure la videollamada mediante el intercambio de información de los pares, como: metadatos de medios, configuración de códecs, tipos de medios e información de red.

- El uso del lenguaje JavaScript facilitó el desarrollo del prototipo debido a que fue utilizado tanto en el lado del servidor como en el lado del cliente; en el lado del servidor utilizando Node.js fue útil para servir los archivos al cliente y realizar transacciones en la base de datos, mientras que en el lado del cliente habilitó las funcionalidades de comunicación en tiempo real mediante la API de WebRTC.

- El prototipo fue implementado en servidores de tipo HTTPS debido a que el equipo de seguridad de Chrome ha determinado que las características de la plataforma web como *getUserMedia,* utilizada por WebRTC para obtención de video local de un usuario, solo se admiten con orígenes seguros.
- A pesar de que WebRTC proclama ser una tecnología que no necesita de la instalación de software adicional para su funcionamiento, por motivos de seguridad del navegador fue necesaria la instalación de la extensión *Screen Capturing* [38] en Google Chrome para poder obtener el video asociado a la compartición de pantalla.
- En la implementación del prototipo fue necesario realizar el proceso conocido como "renegociación", que consiste en usar una conexión entre pares creada con anterioridad para alternar entre el video de la cámara web y el video asociado a la compartición de pantalla.
- Los resultados obtenidos en la realización de las pruebas según los criterios de aceptación definidos en las historias de usuario permitieron comprobar el funcionamiento del prototipo.

4.2 Recomendaciones

- Se recomienda el uso de Bootstrap para la creación de las interfaces web debido a que facilita la adaptación de los elementos HTML y ayuda en la organización de las vistas que se presentan a los usuarios.
- Es importante verificar la compatibilidad de los módulos de Node.js con los diferentes sistemas operativos, debido a que algunos módulos no tienen mantenimiento en sistemas operativos Linux.
- Para el correcto funcionamiento de la aplicación en un entorno *cloud*, con participantes en diferentes redes, se recomienda desactivar el *firewall* del ordenador, debido a que los puertos correspondientes a STUN/TURN y tráfico UDP suelen estar bloqueados.
- Para el uso del prototipo se recomienda considerar la capacidad del hardware que se utiliza, ya que el consumo de memoria RAM aumenta con cada participante que se une a la videoconferencia debido a la transmisión y recepción de varios *streams* de video en la arquitectura tipo *mesh*.
- Para manejar problemas de desconexión, latencia y degradación de la calidad del video se recomienda el uso de una conexión cableada.

- Para trabajos futuros sobre el tema, en los cuales se requiera agregar más de un flujo de video a una conexión entre pares WebRTC previamente creada, se deberá tener en cuenta los cambios para el formato SDP conforme a las normas del denominado “Plan Unificado”, que se refiere a la señalización de múltiples fuentes de medios.
- La compartición de pantalla mediante el uso de la extensión de Google Chrome capta únicamente los sonidos del sistema, si se deseara capturar audio proveniente del micrófono se deberá añadir como un flujo adicional a la conexión de pares creados previamente.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] A. B. Johnston y D. Burnett, “WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web,” Segunda Edición. St. Louis USA, 2013, p. 7.

[2] B.A. Jansen, "Performance Analysis of WebRTC-based Video Conferencing," M.S. Thesis, EEMCS, Delft University of Technology, Países Bajos, Delf, 2016, pp. 6-7, 16-19.

[3] W3C, “Web Architecture,” 2018, [Online]. Available: https://www.w3.org/standards/webarch/. [Accessed: 04- May-2018].

[4] Programación Web, “Arquitectura de las aplicaciones Web,” 2018, [Online]. Available: https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/ .[Accessed: 04- May-2018].

[5] S. Smith, “Arquitecturas de aplicaciones web comunes,” Microsoft,2018, [Online]. Available: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/common-web-application-architectures. [Accessed: 04- May-2018].

[6] IBM Knowledge Center, “Arquitecturas de tres niveles,” 2018, [Online]. Available: https://www.ibm.com/support/knowledgecenter/es/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/covr_3-tier.html. [Accessed: 04- May-2018].

[7] statcounter GlobalStats, “Browser Market Share Worldwide April-2018,” [Online]. Available: http://gs.statcounter.com/. [Accessed: 04- May-2018].

[8] MDN Web Docs, “Gecko,” [Online]. Available: https://developer.mozilla.org/es/docs/Gecko. [Accessed: 11- Jul-2018].

[9] Webkit, “A fast, open source web browser engine.,” [Online]. Available: https://webkit.org/. [Accessed: 12- Jul-2018].

[10] Microsoft 2019, “Microsoft Edge,” [Online]. Available: https://www.microsoft.com/en-us/windows/microsoft-edge. [Accessed: 12- Jul-2018]

[11] Polycom, “An Introduction to the Basics of Video Conferencing,” 2017, [Online]. Available:

http://www.polycom.co.in/content/dam/polycom/common/documents/whitepapers/intro-video-conferencing-wp-engb.pdf. [Accessed: 23- May- 2017].

[12] R. Vapenik, M. Michalko, J. Janitor and F. Jakab, "Secured web-oriented videoconferencing system for educational purposes using WebRTC technology," in Conf. International Conference on Emerging eLearning Technologies and Applications (ICETA), Starý Smokovec, Eslovaquia, 2014, pp. 495-500.

[13] XU, Yang, et al. “Video telephony for end-consumers: measurement study of Google+, iChat, and Skype,” 2012 ACM, p. 371-384.

[14] WebRTC, "WebRTC project," 2018, [Online]. Available: https://webrtc.org/ [Accessed 20-Abr-2018].

[15] S. Ouya, C. Seyed, A. Bamba Mbacke, G. Mendy, I. Niang, "WebRTC platform proposition as a support to the educational system of universities in a limited Internet connection context," in 5th World Congress Information and Communication Technologies (WICT), Marrakech, Morocco, 2015, pp 48.

[16] A. Mellen, “The history of WebRTC,” 2018, [Online]. Available: https://www.callstats.io/blog/2018/05/11/history-of-webrtc-infographic. [Accessed 25-Abr-2018].

[17] WebRTC.org, “What is WebRTC,” 2018, [Online]. Available: https://webrtc.org/faq/#what-is-webrtc. [Accessed 20-Abr-2018]

[18] R. R. Stewart, “Stream Control Transmission Protocol,” RFC 4960, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc4960.txt . [Accessed 22-Abr-2018].

[19] V. Kandari ,”Implementation and Performance Optimization of WebRTC Based Remote Collaboration System,” M.S Thesis, Faculty of Computing, Blekinge Institute of Technology, Sweden, Karlskrona, 2016 pp.15 -18.

[20] WebRTC, “Interop Notes, API differences,” 2018, [Online]. Available: https://webrtc.org/web-apis/interop/ . [Accessed 22-Abr-2018].

[21] MDN web docs, “API de MediaStream,” 2018, [Online]. Available: https://developer.mozilla.org/es/docs/WebRTC/MediaStream_API . [Accessed 21-Abr-2018].

[22] D.C Burnett, A. Bergkvist, C. Jennings, A Narayanan, B. Aboba, J. Bruaroey, W3C Editor's Draft, "Media Capture and Streams," 2018, [Online]. Available: https://w3c.github.io/mediacapture-main/#stream-api. [Accessed 21-Abr-2018].

[23] WebRTC, "WebRTC Native APIs," 2018, [Online]. Available: https://webrtc.org/native-code/native-apis/. [Accessed 21-Abr-2018].

[24] MDN web docs, "RTCPeerConnection," 2018, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection. [Accessed 20-Abr-2018].

[25] J.D. Gauchat, "El gran libro de HTML5, CSS3 y Javascript," primera edición, España: marcombo, 2012, pp. 18, 32.

[26] W3C,"HTML & CSS," 2018, [Online]. Available: https://www.w3.org/standards/webdesign/htmlcss. [Accessed 23-Abr-2018].

[27] D. Flanagan, "JavaScript: The Definitive Guide," sexta edición, USA: O'Reilly, 2011, pp. 19.

[28] M. Otto, "Bootstrap getting started," [Online]. Available: https://getbootstrap.com/docs/3.4/. [Accessed 22-Abr-2018].

[29] MDN web docs, "Introduction to web APIs", 2018, [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction. [Accessed 22-Abr-2018].

[30] Node.js Foundation, "Node.js," 2018, [Online]. Available: https://nodejs.org/es/. [Accessed 22-Abr-2018].

[31] socket.io, "socket.io 2.0," 2018, [Online]. Available: https://socket.io/ . [Accessed 22-Abr-2018].

[32] MySQL Documentation, "What is MySQL", 2018, [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. [Accessed 22-Abr-2018].

[33] O.A. Pérez, "Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP – SCRUM",2011, pp. 73-75.

[34] Ureus, Ingeniería de Software, "Desarrollo de sistemas y soluciones backend", 2018, [Online]. Available: http://www.ureus.cl/esp/Servicios.aspx?Desarrollo+de+Sistemas+y+Backend#. [Accessed 06-May-2018].

[35] Y. Liao, Z. Wang, Y. Luo, "The Design and Implementation of a WebRTC based Online Video Teaching System," in Electronic and Automation Control Conference (IMCEC), Xi'an, China, 2016.

[36] Y. Bandung, H. C. Tanuwidjaja, L.B. Subekti, K. Mutijarsa, "Development of Multimedia System for Supporting Education in Rural Areas," in International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Nusa Dua, Indonesia, 2015.

[32] Express Documentation, "Direccionamiento" ,2018, [Online]. Available: http://expressjs.com/es/guide/routing.html. [Accessed 05-Nov-2018].

[33] M. Khan, " WebRTC Experiments & Demos", 2018, [Online]. Available: https://github.com/muaz-khan/WebRTC-Experiment. [Accessed 05-Jun-2018].

[34] M. Khan, " Socket.io over Node.js ", 2018, [Online]. Available: https://github.com/muaz-khan/WebRTC-Experiment/tree/master/socketio-over-nodejs. [Accessed 05-Jun-2018].

[35] M. Khan , " Capture Screen on Any Domain ", 2018, [Online]. Available: https://github.com/muaz-khan/getScreenId [Accessed 24-Jun-2018].

[36] M. Khan , " DetectRTC ", 2018, [Online]. Available: https://github.com/muaz-khan/DetectRTC. [Accessed 24-Jun-2018].

[37] WebRTC, "WebRTC adapter", 2018, [Online]. Available: https://github.com/webrtc/adapter. [Accessed 24-Jun-2018].

[38] M. Khan , " Screen Capturing", 2018, [Online]. Available: https://chrome.google.com/webstore/detail/screen-capturing/ajhifddimkapgcifgcodmmfdlknahffk. [Accessed 24-Jun-2018].

6. ANEXOS

ANEXO A: Información del módulo de ayuda.

ANEXO B: *Script* para generación de base de datos y tablas.

ANEXO C: Código fuente de la aplicación web.

ANEXO D: *Script* y archivo .csv para carga de información del módulo de ayuda.

Nota: Dada su extensión, los anexos B a D se han incluido en el disco compacto adjunto a este documento.

ANEXO A

INFORMACIÓN DEL MÓDULO DE AYUDA

A.1 DIAGRAMA DEL PROTOTIPO

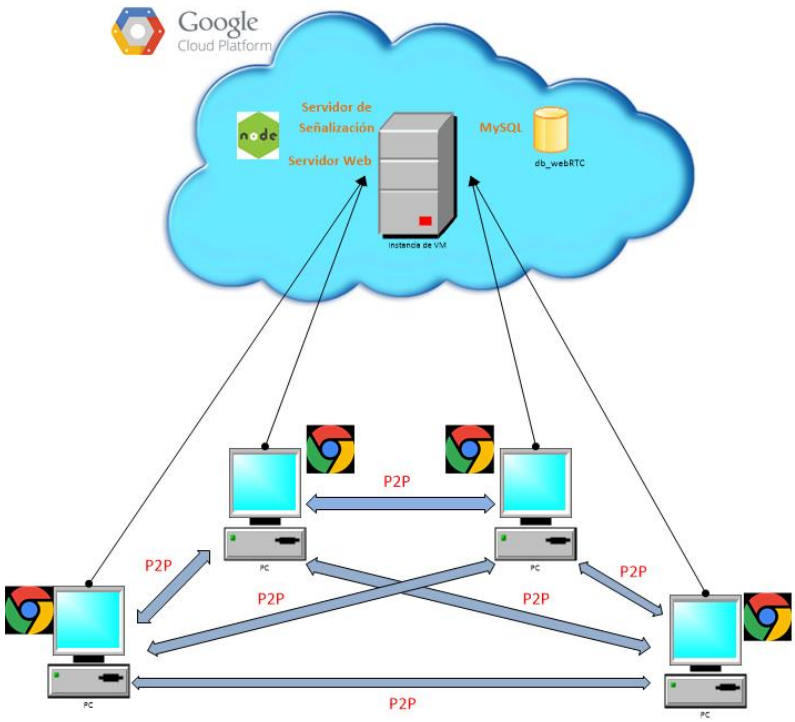

Fig A. 1 Diagrama del prototipo

Esta aplicación web, plantea ser una alternativa para videoconferencia y compartición de pantalla mediante el uso de la tecnología WebRTC. El diagrama del prototipo muestra la arquitectura que tendrá el mismo en funcionamiento, con la participación de cuatro usuarios ejecutando el navegador Google Chrome.

El servidor web y el servidor de señalización han sido desarrollados en Node.js y para el almacenamiento se ha utilizado el motor MySQL. Los servidores desarrollados han sido alojados en una instancia VM dentro de la plataforma Google Cloud.

Los usuarios acceden a la aplicación, se autentican y al requerir comunicarse el servidor web, devuelve los scripts necesarios para la comunicación en tiempo real con WebRTC. Una vez iniciado el requerimiento de comunicación el cliente se conecta con el servidor de señalización, creando un canal de comunicación y a continuación se empieza una serie de negociaciones hasta establecer el canal de tipo SCTP de WebRTC y el contenido ya sea video, texto, etc. fluye entre los pares involucrados en la comunicación.

La arquitectura de videoconferencia utilizada en el prototipo y que se puede observar en el diagrama, corresponde al tipo mesh que es ampliamente utilizada por aplicaciones que usan WebRTC para el envío de flujos de video. En este tipo de arquitectura cada cliente individual se conecta a otro cliente e intercambia los datos y los flujos de medios directamente entre los navegadores sin la participación de un servidor para procesar el video.

A.2 CARACTERÍSTICAS PRINCIPALES

Las características principales para comunicación en tiempo real implementadas en esta aplicación son:

Videoconferencia

Definida como la reunión en línea entre más de dos personas, donde la comunicación es simultanea y bidireccional ha sido implementada en esta aplicación como una solución orientada a la web, donde mediante el uso del navegador se permite el intercambio de video, voz y datos sin necesidad de la instalación de software adicional.

La videoconferencia es uno de los medios de telecomunicación más revolucionarios conocidos, dado que permite establecer contacto visual y verbal en tiempo real entre personas que pueden encontrarse separadas a cualquier distancia, por lo cual es útil para el ámbito educativo tanto en la modalidad a distancia como presencial permitiendo interacción entre alumnos y profesores.

Compartición de pantalla

Se trata de una tecnología en línea que permite compartir la sesión actual de un usuario con otro usuario en otra computadora, siendo común que aplicaciones de videoconferencia permitan que el iniciador de la videoconferencia comparta su escritorio con otros participantes a través de Internet. La utilidad de la compartición de pantalla reside en compartir contenido asociado a documentos o acciones dentro de la computadora del organizador sin necesidad de transferir archivos, en el ámbito educativo se podrían hacer demostraciones en tiempo real en herramientas de software de simulación.

Chat

Consiste en envío de mensajes de texto entre los participantes de la comunicación, es muy útil para comunicarse con grupos de personas y enviar mensajes instantáneos a los participantes de la sala de comunicación. Para su uso en el ámbito educativo es útil para

tutorías individuales y cuya temática no requiera gran cantidad de texto ni sea de gran complejidad, será útil para resolver pequeñas dudas.

A.3 TIPOS DE CUENTA

En la aplicación desarrollada se manejan dos tipos de cuenta que acceden a diferentes módulos.

- *Cuenta Administrador:* Cuenta con acceso a los módulos de comunicación y de administración de usuarios y roles. La administración de usuarios y roles se refiere a creación de usuarios con cuenta administrador o usuario, edición de información de usuarios, eliminación de usuarios de la aplicación y asignación o eliminación de roles.
- *Cuenta Usuario:* Cuenta con acceso a los módulos de comunicación y de ayuda, el módulo de ayuda ofrece información de las características del prototipo y ofrece un pequeño manual de uso de la aplicación.

En ambos tipos de cuenta el módulo de comunicación permite el uso de la comunicación en tiempo real mediante WebRTC, acceso a videoconferencia e intercambio de mensajes, así como compartición de pantalla de parte de la persona que organizó la videoconferencia.

A.4 MANUAL DE USO

Presentación

Esta aplicación web fue desarrollada con el fin de presentar una alternativa para la realización de videoconferencias con el uso de la tecnología WebRTC, dado que en la actualidad la necesidad de comunicación en tiempo real es creciente por su popularidad y la mejora que estas soluciones brindan a las empresas en el ámbito de comunicación entre colaboradores, de la misma manera en el ámbito educativo es muy útil para apoyar el proceso de aprendizaje.

Objetivos del sistema:

- Cumplir las necesidades de comunicación en tiempo real mediante la realización de videoconferencia.
- Permitir a los organizadores de videoconferencia compartir su pantalla con el resto de los participantes.
- Permitir a los participantes de una videoconferencia la transferencia de mensajes de texto.

- Proporcionar un perfil de usuario y la edición de información de los usuarios registrados.
- Proporcionar un módulo de ayuda oportuna a los usuarios regulares de la aplicación.

Objeto

Este manual está dirigido a las personas que tengan una cuenta tipo usuario en la aplicación.

Generalidades del sistema

Página de inicio

Al ingreso a la aplicación mediante su dirección web se mostrará una página de inicio de la aplicación como se muestra en la Fig A. 2 a continuación.

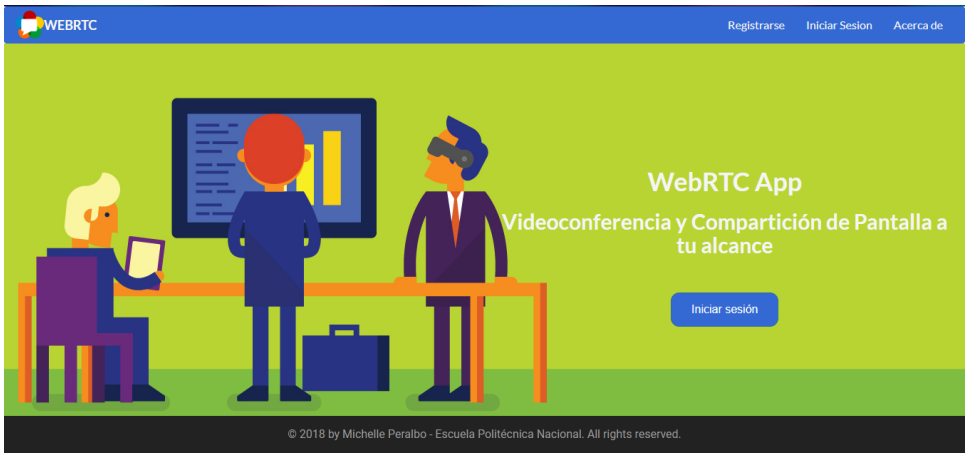

Fig A. 2 Página de inicio de la aplicación

Registro en la aplicación

Como un usuario nuevo, es necesario registrarse antes de poder acceder a las funciones principales de la aplicación web. Para ello se deberá presionar la opción ‘Registrarse’ en la página principal de la aplicación. Se ingresará a la página de registro que se presenta en la Fig A. 3.

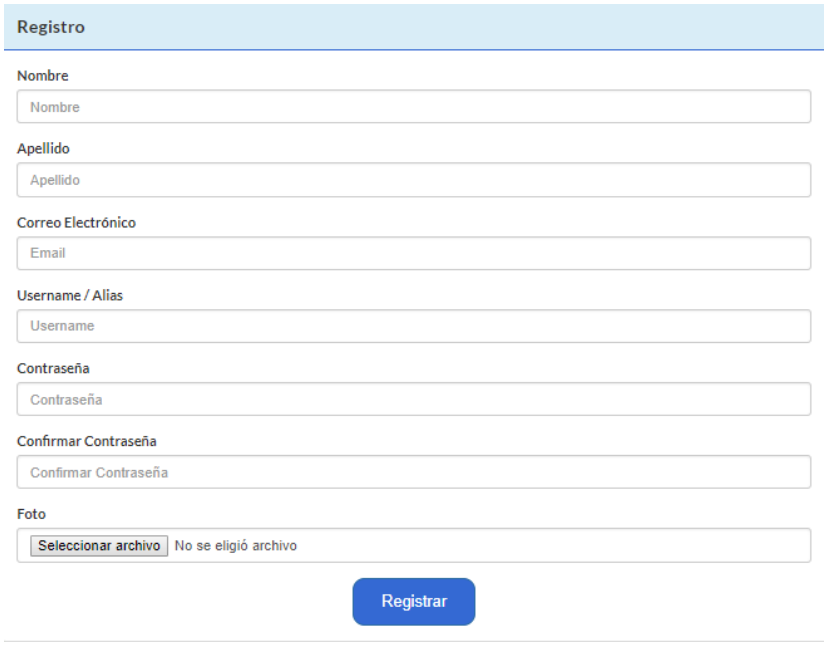

Fig A. 3 Formulario de registro

Es muy importante que todos los campos estén llenos, que se ingrese un correo electrónico válido, que no trate de registrarse con los mismos datos de una cuenta anterior; caso contrario se mostrarán mensajes de error si se incumplen una de estas reglas. Por ejemplo, en el caso de no coincidir las contraseñas se inhabilitará el botón de registro (Fig A. 4)

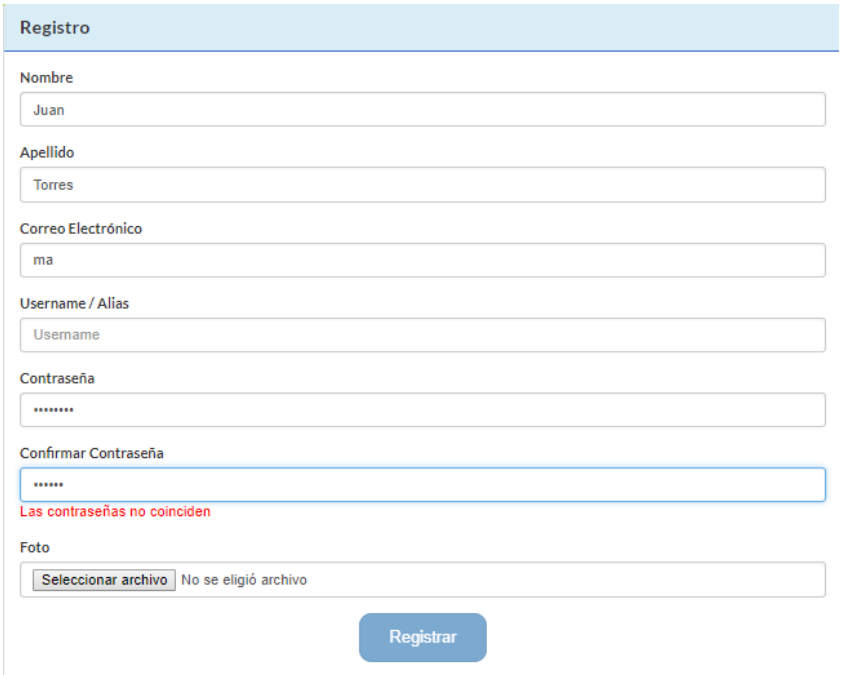

Fig A. 4 Contraseñas no coinciden

Inicio de sesión

Si el usuario ya se ha registrado en la aplicación, puede iniciar sesión, presionando los botones ‘Iniciar Sesión’ de la página de inicio como se muestra en la Fig A. 5.

Fig A. 5 Opciones de inicio de sesión

Posterior a presionar el botón se mostrará la vista de inicio de sesión Fig A. 6.

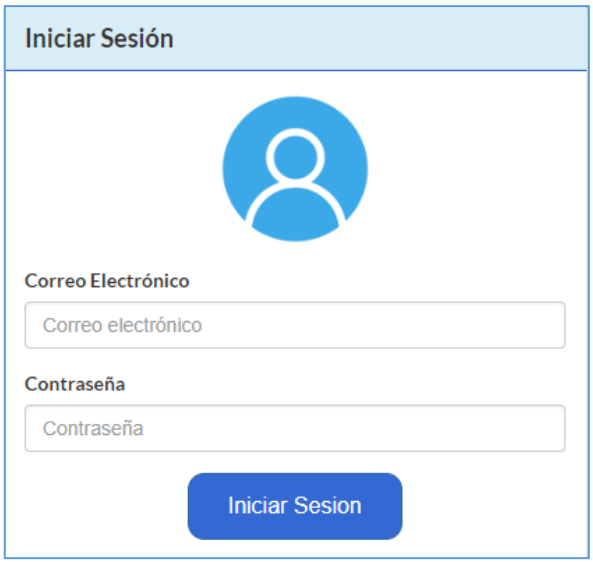

Fig A. 6 Formulario de inicio de sesión

En este formulario se deberá ingresar: *Email* y contraseña del usuario previamente registrado, al presionar el botón ‘Iniciar Sesión’ la aplicación validará que los campos estén llenos, que se ingrese un correo válido, y que el usuario ingrese una cuenta existente al igual que valida si la contraseña es correcta o no. En la Fig A. 7 se muestran los problemas más comunes en el inicio de sesión.

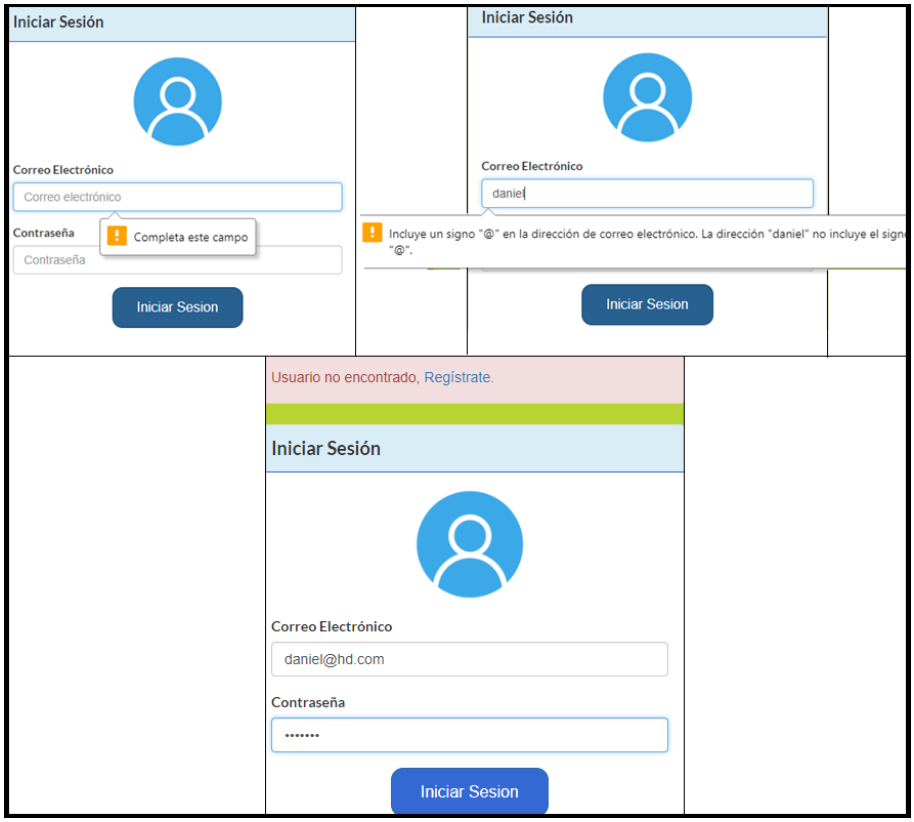

Fig A. 7 Problemas de inicio de sesión

Perfil

Una vez autenticado, el usuario será redirigido a su perfil de usuario en la aplicación, donde se mostrará su información asociada como se muestra en la Fig A. 8.

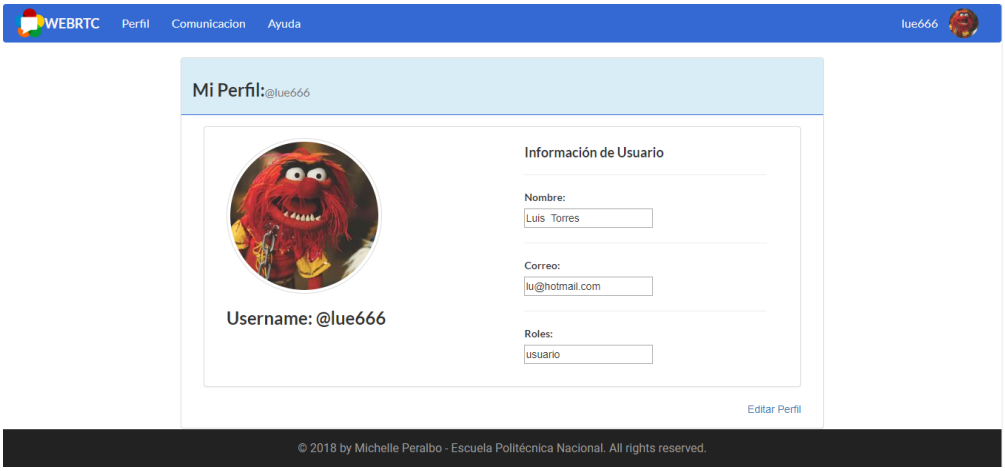

Fig A. 8 Perfil de usuario

Edición de datos personales

Para la edición de los datos personales como, por ejemplo, nombres, *username* y fotografía se deberá ingresar a Editar Perfil o Configuración como se muestra en la Fig A. 9.

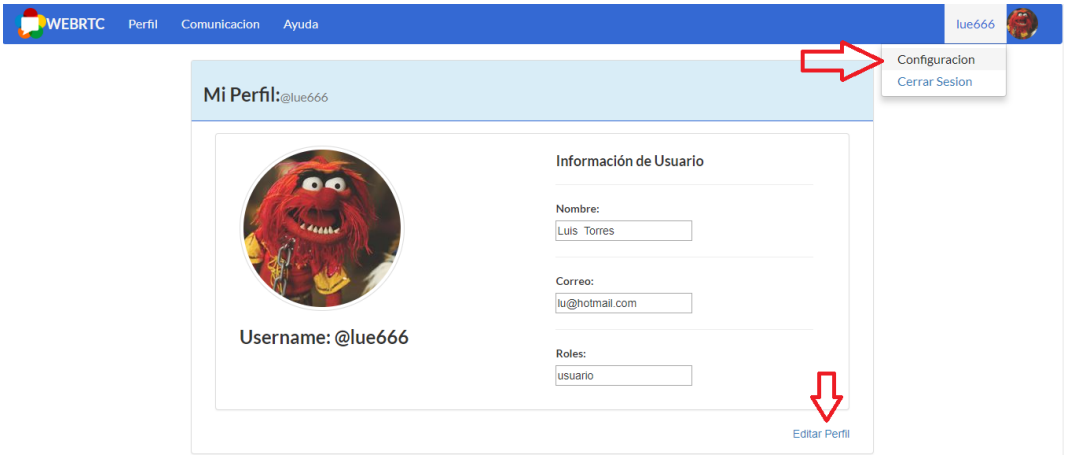

Fig A. 9 Opciones de edición de datos personales

Una vez seleccionada alguna de las opciones para edición de perfil se muestra el formulario de edición de datos personales mostrado en la Fig A. 10.La aplicación valida que los campos se encuentren llenos y si no lo están se le notifica al usuario.

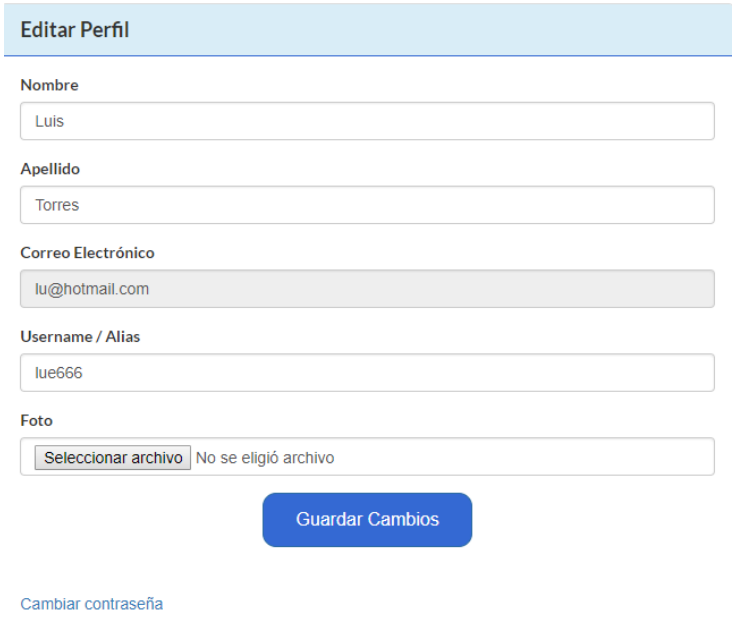

Fig A. 10 Formulario de edición de datos personales

Cambio de contraseña

Para cambio de contraseña se accederá desde el panel de edición de información de usuario como se muestra en la Fig A. 11.

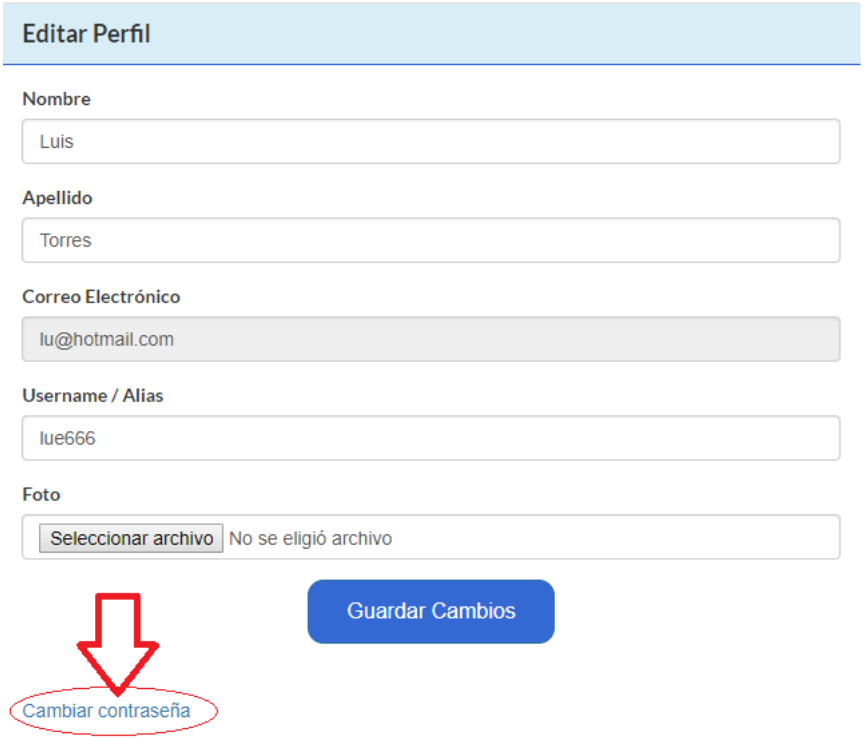

Fig A. 11 Opción de cambio de contraseña

En la Fig A. 12 se muestra el formulario de cambio de contraseña, la aplicación validará que la contraseña actual sea correcta, que la nueva sea diferente de la actual y se debe confirmar la contraseña nueva.

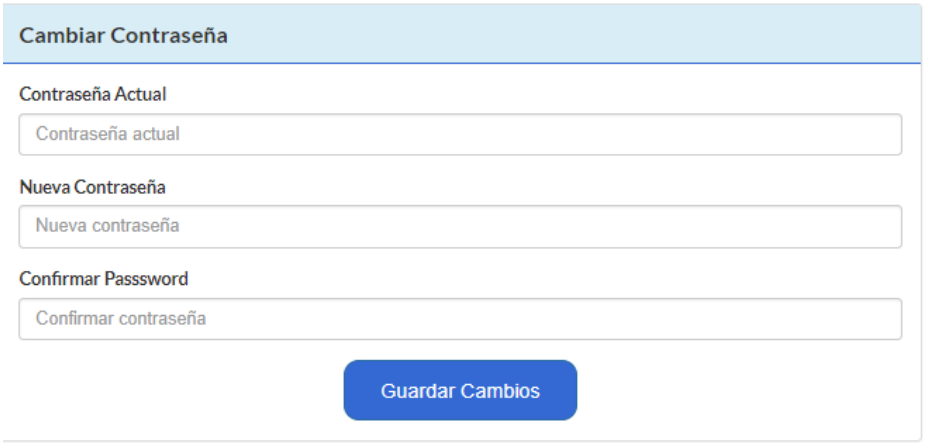

Fig A. 12 Formulario de cambio de contraseña

Comunicación

Videoconferencia

Para iniciar un cuarto de videoconferencia se debe acceder a la pestaña Comunicación de la barra de navegación ubicada en la parte superior. Se mostrará la página como la de la Fig A. 13.

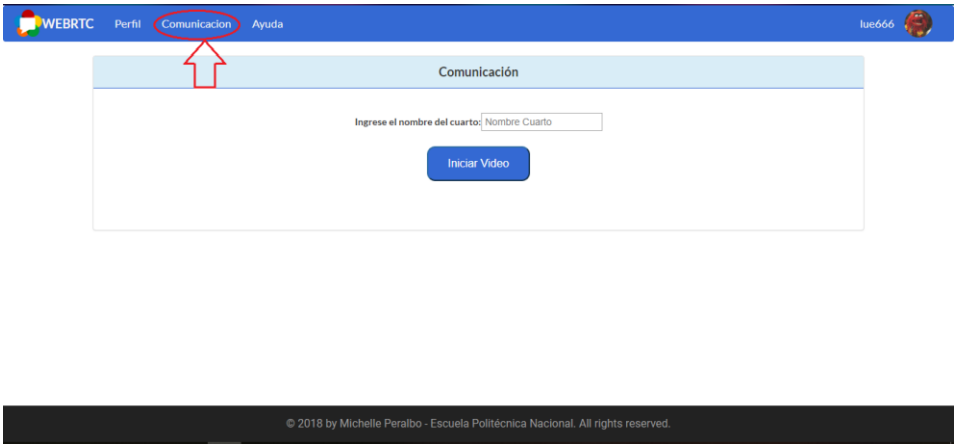

Fig A. 13 Acceso al módulo de comunicación

Después de ingresar un nombre al cuarto de videoconferencia y dar clic en el botón 'Iniciar video' mostrado en la Fig A. 13 se inicia la videoconferencia y se pedirá acceso al micrófono y cámara web del usuario, se debe dar clic en 'Permitir' (Fig A. 14).

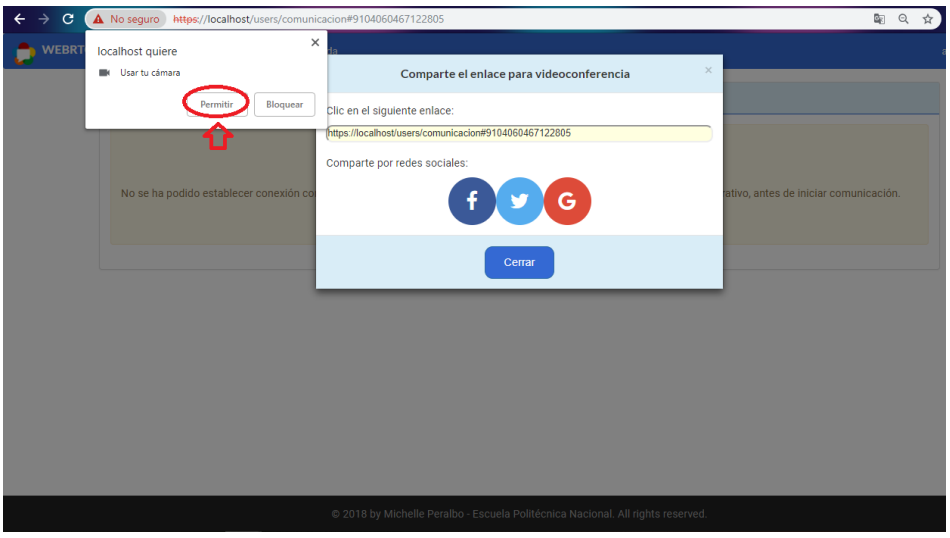

Fig A. 14

A continuación, el usuario que inició la videoconferencia deberá compartir el enlace generado como se muestra en la Fig A. 14 para que otros usuarios puedan unirse a la videoconferencia.

En el caso de ser el usuario que recibe el enlace, deberá iniciar sesión en la aplicación y pegar el enlace en la barra de navegación para poder unirse a la videoconferencia.

De ser así los usuarios se encontrarán en la misma sala y podrán realizar la comunicación en tiempo real (Fig A. 15).

Fig A. 15 Videoconferencia entre cuatro participantes

Compartición de pantalla

En caso de ser el organizador de la videoconferencia, el usuario podrá compartir su pantalla con el resto de los participantes, luego de establecida la comunicación, presionando el botón compartir pantalla. Al presionar este botón aparecerá una ventana emergente, para compartir la pantalla deberá darse clic en ‘Compartir’ como muestra la Fig A. 16.

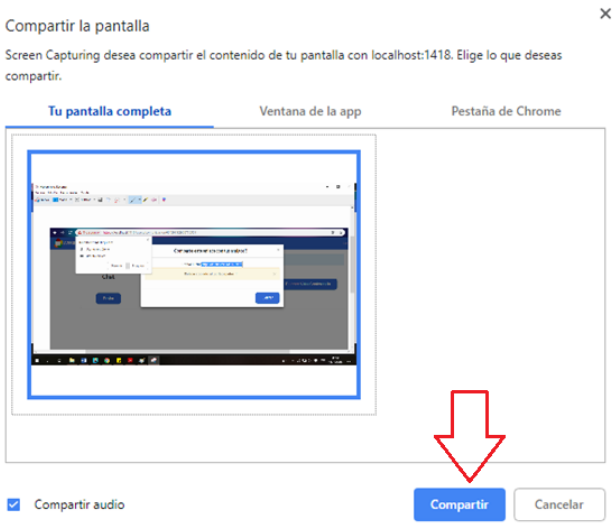

Fig A. 16 Ventana para compartición de pantalla

Transferencia de mensajes

Al iniciar la comunicación también se dará paso a la posibilidad de intercambio de mensajes, como se muestra en la Fig A. 17 **¡Error! No se encuentra el origen de la referencia.**, donde el usuario puede escribir en el cuadro de texto asignado y enviar el mensaje mediante el botón enviar o al presionar la tecla “Enter”, en la parte de debajo podrá visualizar los mensajes enviados y los recibidos de los usuarios remotos participantes de la videoconferencia.

Fig A. 17 Transferencia de mensajes

ORDEN DE EMPASTADO