



“Responsabilidad con pensamiento positivo”

UNIVERSIDAD TECNOLÓGICA ISRAEL

**TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:
INGENIERO EN ELECTRÓNICA DIGITAL Y
TELECOMUNICACIONES**

TEMA:

DESARROLLO DE UNA APLICACIÓN WEB DE VIDEO CONFERENCIA BASADA
EN LOS PROTOCOLOS WEBRTC PARA NAVEGADORES CHROME Y MOZILLA

AUTOR:

RAMIREZ OJEDA RONNY DAVID

TUTOR:

MSC. ELIZABETH PATRICIA CARRILLO ARMENDARIZ

QUITO, ECUADOR

2019

AGRADECIMIENTO

Agradezco a Dios por darme la vida, a mi padre por la educación y el apoyo en la carrera. A la Universidad Israel, mis maestros y tutores que hicieron posible este proyecto.

DEDICATORIA

Este trabajo está dedicado a Dios por darme la vida y a mi padre por su guía y apoyo.

CERTIFICACIÓN DEL TUTOR

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación **“DESARROLLO DE UNA APLICACIÓN WEB DE VIDEO CONFERENCIA BASADA EN LOS PROTOCOLOS WEBRTC PARA NAVEGADORES CHROME Y MOZILLA** presentado por el Sr. Ramírez Ojeda Ronny David estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D.M., 20 de agosto 2019

TUTOR

.....

MSC. ELIZABETH PATRICIA CARRILLO ARMENDARIZ

DECLARACIÓN DE AUTENTICIDAD

El abajo firmante, declaro que los contenidos y resultados obtenidos en el presente proyecto, como requerimiento previo para la obtención del Título de Ingeniería en Electrónica digital y Telecomunicaciones, son absolutamente originales, auténticos y personales, de exclusiva responsabilidad legal y académica del autor.

.....

Ramirez Ojeda Ronny David

C.I.: 1718160557

APROBACIÓN DEL TRIBUNAL DE GRADO

Proyecto de aprobación de acuerdo con el Reglamento de Títulos y Grados de la facultad.

Quito, septiembre 2019

Para constancia firman:
TRIBUNAL DE GRADO

F.....
PRESIDENTE

F.....
VOCAL

F.....
VOCAL

RESUMEN

El presente proyecto tiene como objetivo el diseño, desarrollo e implementación de una aplicación web que utilice el protocolo WebRTC para establecer un espacio compartido, conocido como room (sala virtual), en el que se pueda realizar video llamada, mensajería instantánea, compartición de archivos y de pantalla entre uno o varios usuarios.

Para la realización del proyecto se realiza un análisis de la situación actual, las variables involucradas y las necesidades de los usuarios mediante el uso de la metodología descriptiva y el diseño ocupa una metodología proyectiva. Un desarrollo organizado se aplica la metodología de software Scrum ágil y la implementación mediante la programación en lenguajes de aplicaciones web y la librería RTCMulticonnection.

La aplicación consta de dos páginas, la página index.html donde se puede crear o acceder a un room al ingresar el número de room, un nombre de usuario y una contraseña. Tras la autenticación se re direcciona a una página room.html donde se tiene implementado los módulos de videoconferencia, envío de mensajes y compartición de archivos o pantalla. La aplicación es desplegada en el domino web Heroku para las pruebas de funcionalidad.

Se comprueba que la aplicación realiza la validación de las credenciales para la creación y acceso a los rooms, así como presenta las funcionalidades descritas para los usuarios. La aplicación es compatible con Google Chrome 28+ y Mozilla Firefox 22+, el número de rooms para el dominio Heroku es de máximo 10 en la versión de prueba y el número de usuarios concurrentes en la aplicación es de 50.

PALABRAS CLAVE webrtc, rtcmulticonnection, scrum, heroku, node, jquery

ABSTRACT

The present project aims to design, develop and implement a web application using the WebRTC protocol to establish a shared room where users can video call, message, share files and share screen with each other.

The project uses the descriptive methodology to analyze the variables and needs of similar application users. The application in this project is designed using the Scrum methodology and it is implemented using web application languages and the RTCMulticonnection library.

The application has two pages, index.html where a user can either create or join a room providing the room id, a user name and a password. After the authentication the application redirects the user to the room.html page where two or more users can video chat, send messages, share files and share screens. For the purpose of testing, the application is deployed in the Heroku web application.

The application developed in this project can validate the room's access credentials and it allows the users to communicate using the previous mentioned functionalities. The characteristics of Google and Mozilla explorer are defined as well as the parallel rooms and simultaneous users limits.

KEY WORDS webrtc, rtcmulticonnection, scrum, heroku, node, jquery

TABLA DE CONTENIDO

AGRADECIMIENTO	ii
DEDICATORIA	iii
CERTIFICACIÓN DEL TUTOR.....	iv
DECLARACIÓN DE AUTENTICIDAD	v
APROBACIÓN DEL TRIBUNAL DE GRADO.....	vi
RESUMEN	vii
ABSTRACT	viii
TABLA DE CONTENIDO	9
LISTA DE FIGURAS	13
LISTA DE TABLAS	15
INTRODUCCIÓN.....	17
ANTECEDENTES	17
PLANTEAMIENTO Y JUSTIFICACIÓN	19
OBJETIVO GENERAL	19
OBJETIVOS ESPECÍFICOS	19
ALCANCE	19
DESCRIPCIÓN DE LOS CAPÍTULOS	20
CAPÍTULO 1	21
FUNDAMENTACIÓN TEÓRICA	21
<u>1.</u> Videoconferencia.....	21
1.1 Audio digital	22
1.1.1 Fundamentos de audio digital.....	22
1.1.2 Técnicas de compresión.....	23
1.1.3 Calidad de audio digital.....	24

1.2 Video digital	25
1.2.1 Muestreo	26
1.2.2 Cuantificación.....	27
1.2.3 Codificación	27
1.2.3.1 Técnicas de codificación.....	28
1.2.4 Códecs de video.....	30
1.2.4.1 Indeo (INtel viDEO).....	31
1.2.4.2 MPEG 1 y 2.....	31
1.2.4.3 MPEG4.....	32
2. Protocolo WebRTC	32
2.1 Códec de audio WebRTC	33
2.2 Códec de Video WebRTC	33
2.3 Seguridad en WebRTC.....	34
2.3.1 DTLS (Datagram Transport Layer Security, Seguridad en capa de transporte de datagramas).....	35
2.3.2 SRTP (Secure Real Time Protocol, Protocolo de seguridad en tiempo real)	35
2.3.3 Soporte WebRTC.....	36
3. Aplicación web	36
3.1 Protocolo HTTP.....	36
3.2 Lenguajes y herramientas de programación web	37
3.2.1 HTML.....	37
3.2.2 JAVA SCRIPT (JS).....	39
3.2.2.1 Node.JS.....	40
3.2.2.2 RTCMulticonnection	41
3.2.2.3 jQuery	41
3.2.3 CSS (Cascading Style Sheets, Hojas de estilo en cascada)	42
3.2.3.1 Semantic UI.....	43

CAPÍTULO 2	44
MARCO METODOLÓGICO	44
2.1 Metodología aplicada para las fases del proyecto	45
2.2 Metodología Scrum Ágil	45
CAPÍTULO 3	47
PROPUESTA	47
3.1 Esquemas	47
3.2 Módulos	49
3.2.1 Módulo de acceso	50
3.2.2 Módulo de room	50
3.3 Aspectos técnicos del proyecto.....	51
3.4 Sistema operativo, dependencias y herramientas de programación	51
3.5 Análisis de costos de la aplicación	52
3.5.1 Horas-hombre	52
3.5.2 Gastos pasivos e imprevistos	54
3.6 Análisis de tiempo	56
3.7 Ventajas del proyecto	58
CAPÍTULO 4	60
IMPLEMENTACIÓN	60
4.1 Diseño.....	60
4.1.1 Sprints.....	60
4.1.2 Diseño de diagramas de clase	61
4.1.3 Diseño de diagrama de distribución de objetos	61
4.2 Implementación	64
4.2.1 Librería RTCMulticonnection	65
4.2.2 Server IO (Server Input Output, Servidor de entrada y salida)	70
4.2.3 Requisitos mínimos de navegadores Mozilla y Chrome	71

4.3 Implementación y programación de las páginas.....	72
4.3.1 La página index.html	72
4.3.2 La página room.html	75
4.3 Programación de la funcionalidad WebRTC.....	76
4.4 Pruebas de funcionalidad.....	77
4.5 Análisis de las pruebas	83
4.5.1 Desafíos y problemáticas de la implementación de una aplicación de alto nivel	83
4.5.2 Parámetros de conexión para un nivel adecuado.....	86
CONCLUSIONES Y RECOMENDACIONES	89
CONCLUSIONES.....	89
RECOMENDACIONES	91
BIBLIOGRAFÍA	92
ANEXOS	94
ANEXO 1	95
CÓDIGO DE INDEX.HTML.....	95
ANEXO 2	104
CÓDIGO DE ROOM.HTML.....	104
ANEXO 3	125
Manual Heroku (HerokuDevCenter, 2019).....	125
ANEXO 4	129
Manual de usuario	129
MANUAL DE USUARIO	130

LISTA DE FIGURAS

Figura 1.1	Representación simplificada del proceso de digitalización	24
Figura 1.2	Componentes YUV para imagen base	26
Figura 1.3	Muestreo para una señal analógica	27
Figura 1.4	Esquema de codificación temporal	29
Figura 1.5	Diagrama de codificación bidireccional	30
Figura 1.6	Segmento de código de ejemplo de código JavaScript	40
Figura 1.7	Segmento de código de ejemplo de jQuery	42
Figura 1.8	Ejemplo de código CSS	42
Figura 3.1	Diagrama UML de casos de uso	47
Figura 3.2	Diagrama UML de flujo para aplicación tradicional	48
Figura 3.3	Diagrama UML de flujo para aplicación propuesta.....	48
Figura 3.4	Módulos de la aplicación	49
Figura 3.5	Versión de Google Chrome	51
Figura 3.6	Consola de Google Chrome	52
Figura 3.7	Costo de horas programación promedio JavaScript.....	53
Figura 3.8	Costo de horas programación promedio Web.....	54
Figura 3.9	Costo de un proyecto web con similares características	56
Figura 3.10	Diagrama de Gantt para el proyecto	58
Figura 4.1	Diagrama de clase de la aplicación	62
Figura 4.2	Esquema de objetos index.html	62
Figura 4.3	Esquema de objetos para room.html	63
Figura 4.4	Esquema de sección de videoconferencia.....	63
Figura 4.5	Esquema de sección de envío de mensajes	64
Figura 4.6	Esquema de Server.js	66
Figura 4.7	Esquema de package.json	67
Figura 4.8	Instalador de node JS para Windows	68
Figura 4.9	Comandos para instalar RTCMulticonnection en Linux	68

Figura 4. 10	Comandos para instalar el servidor RTCMulticonnection.....	70
Figura 4. 11	Esquema html de la página principal index.html.....	73
Figura 4.12	Lista de referencias en index.html	74
Figura 4.13	Vista de la página index.html.....	74
Figura 4.14	Vista de room.html con dos usuarios. a) Zona de videoconferencia b) Zona de mensajería c) Zona de botón de envío de archivos y compartición de pantalla d) Zona de pantalla compartida.....	75
Figura 4.15	Vista de room.html con varios usuarios.....	76
Figura 4. 16	Plataforma de Heroku	78
Figura 4. 17	Aplicación desplegada en Heroku	79
Figura 4. 18	Mensaje de validación para un room creado.....	79
Figura 4.19	Mensaje de validación cuando no existe un room.	80
Figura 4. 20	Funcionalidades de la aplicación WebRTC.....	81
Figura 4. 21	Prueba de carga de la aplicación	81
Figura 4. 22	Diagrama de aplicación Big Blue Button	84
Figura 4. 23	Aplicación webrtc-internals	86
Figura 4. 24	Captura de datos de una sesión	86
Figura 4. 25	Datos de video de la aplicación.....	87
Figura 4. 26	Cálculo de ancho de banda.....	87

LISTA DE TABLAS

Tabla 1. 1 Principales etiquetas HTML.....	38
Tabla 3. 1 Cálculo de horas/hombre	53
Tabla 3. 2 Cálculo de costo de horas de desarrollo	54
Tabla 3. 3 Análisis de costos de la aplicación	55
Tabla 3. 4 Recursos físicos	55
Tabla 3. 5 Costo final del proyecto.....	55
Tabla 4. 1 Sprints para el desarrollo del proyecto	60
Tabla 4. 2 Archivos y carpetas de un proyecto base.....	65
Tabla 4. 3 Funciones de RTCMulticonnection.....	69
Tabla 4. 4 Referencias WebRTC	76
Tabla 4. 5 Resumen de la prueba de carga para la aplicación.	82
Tabla 4. 6 Funciones y herramientas de Big Blue Button	84

INTRODUCCIÓN

Las comunicaciones en la actualidad se realizan en gran parte gracias a las redes de datos y telecomunicaciones a través la transmisión y recepción de datos, audio y video. Mediante las redes de comunicaciones, millones de personas están conectadas al usar un dispositivo inteligente con conexión a Internet y hacen cada vez más uso de la red, por tanto, se incrementa la necesidad de implementar conexiones más eficientes, rápidas y confiables.

Existen varias aplicaciones para comunicar a diversos usuarios, entre ellas las más relevantes: aplicación de llamadas, mensajes, correos, y videoconferencia, donde se transmite audio y video en tiempo real. Gran parte de las aplicaciones de videoconferencia tienen ciertos límites en las cuentas gratuitas, como el número máximo de usuarios o la posibilidad de compartir pantallas. Dichas aplicaciones están desarrolladas sobre protocolos privados de voz y video sobre IP para Skype o RTP (Real time protocol o protocolo de tiempo real) para WhatsApp web. Estas aplicaciones ofrecen servicios de video llamada sobre aplicaciones de escritorio o aplicaciones propias que se instalan en los dispositivos.

ANTECEDENTES

La videoconferencia es una tecnología que permite el encuentro entre dos o más personas que se encuentren ubicadas en distintas locaciones geográficas en tiempo real, la comunicación que se establece es de tipo bidireccional y puede involucrar audio, video, medios escritos y más. Las aplicaciones de videoconferencia han atravesado por cambios significativos a lo largo de los años, y en la actualidad son utilizadas en varios campos como el educativo, laboral, de las capacitaciones, comunicación interpersonal privada.

Para el presente proyecto se propone una aplicación web de comunicaciones bidireccionales en tiempo real que cuenta con algunas funcionalidades de WebRTC como videoconferencia y chat, adicionalmente se define un módulo de administración propio de esta aplicación en el caso que se requiera un mediador de comunicaciones o para almacenar

los datos de conexiones. Para esto, se han recopilado trabajos que se presentan como antecedentes a continuación.

Entre los trabajos de investigación en este tema se tienen los siguientes: “Estudio de las características de nuevas arquitecturas web basadas en WebRTC alojada en la nube y factible implementación para aplicaciones de voz sobre IP (VoIP).” Autor: Dannyll Michelle Zambrano. Pontificia Universidad Católica del Ecuador. El trabajo presenta un estudio de una arquitectura WEB para aplicaciones de red como es el caso de VoIP (Voice over IP, Voz sobre IP), mediante servidores WebRTC en la nube, que ofrecen un servicio gratuito o de pago. Mediante un navegador web, por ejemplo, Google Chrome, Mozilla Firefox y Opera, que soporta WebRTC; se han realizados llamadas desde dos ubicaciones geográficas distantes y mediante las herramientas “WebRTC -*internals*” y Wireshark se han tomado valores de *Jitter*, RTT y pérdidas de paquetes para establecer QoS (Quality of service, calidad de servicio) en la transmisión de paquetes de VoIP. Con base a los parámetros analizados se define la capacidad de establecer comunicación de VoIP mediante WebRTC, valiéndose de servidores WebRTC en la Nube.

En este proyecto se realiza un estudio con resultados favorables de la factibilidad de implementar una aplicación web que utiliza el protocolo WebRTC para transmisión de audio y video que puede ser desplegado en un servidor gratuito. Se enfatiza la alta calidad de la voz al momento de establecer la comunicación y se recomienda la investigación de las nuevas características que se presente en el protocolo WebRTC.

Otra investigación previa es: “Diseño e implementación de un prototipo web para comunicación en tiempo real que permite realizar video llamadas, transferencia de archivos y chat, mediante el uso de WebRTC”. Autor: Mendoza Quichimbo Carlos Alfonso, Escuela Politécnica Nacional, en donde se establece un precedente de la factibilidad de realizar un proyecto basado en WebRTC. La aplicación web desarrollada está compuesta por módulos de administración para la gestión de los usuarios y de los grupos de usuarios. Además, cuenta con un módulo de autenticación que maneja el ingreso de usuarios registrados en el sistema y se ejecuta sobre navegadores web.

PLANTEAMIENTO Y JUSTIFICACIÓN

En la actualidad se realiza la videoconferencia mediante aplicaciones de escritorio o instaladas previamente (Skype, WhatsApp, Hangouts), la limitante de estas aplicaciones es que, si la aplicación no se encuentra instalada, se debe instalar y configurar en cada una de las máquinas en las que se quiera trabajar. Las tecnologías actuales apuntan a la migración a una aplicación web que pueda ser accedida desde un explorador de Internet, particularmente para este proyecto desde Mozilla o Chrome.

OBJETIVO GENERAL

Desarrollar e implementar una aplicación web de videoconferencia basada en los protocolos WebRTC para navegadores Chrome y Mozilla.

OBJETIVOS ESPECÍFICOS

- Definir los requisitos mínimos de navegadores Mozilla, Chrome para Windows y Android.
- Definir los módulos y la funcionalidad de los componentes de la aplicación.
- Diseñar los módulos y la funcionalidad de los componentes de la aplicación.
- Desarrollar la aplicación de videoconferencia al usar el protocolo WebRTC.
- Realizar pruebas unitarias de funcionamiento y validación de la aplicación.

ALCANCE

El alcance está en el diseño, desarrollo e implementación de la aplicación de videoconferencia que utiliza el protocolo WebRTC. En el proyecto se definirá los requisitos mínimos para los navegadores Mozilla y Chrome en los dispositivos que utilizarán la aplicación, luego se diseña los módulos de la aplicación: videoconferencia, chat, envío de archivos y compartir pantalla; para el diseño de la aplicación se utilizará diagramas de clases y diagramas UML (Unified Modeling Language, Lenguaje unificado de modelado).

Las características de la aplicación como usuarios simultáneos y ancho de banda se definen por el protocolo WebRTC, sin embargo, para el propósito de muestra de la aplicación se utilizará un host gratuito, por lo que el número de usuarios simultáneos, almacenamiento y ancho de banda se definirán de acuerdo al host en el que se despliegue la aplicación. El host se escoge en función de la compatibilidad con las características de la aplicación.

Se utilizará una metodología SCRUM ágil para el desarrollo de la aplicación y para la programación se utilizarán los lenguajes HTML, PHP y JavaScript junto a las librerías del protocolo WEBRTC. La aplicación se entregará funcional, publicada en un servidor para demostración y con las respectivas pruebas que validen la entrega.

DESCRIPCIÓN DE LOS CAPÍTULOS

Para el proyecto de titulación en el capítulo 1 se detallan y argumenta desde una visión científica y tecnológica los conceptos básicos que permitirán la comprensión del desarrollo e implementación del programa de videoconferencia.

En el capítulo 2 se definen las metodologías utilizadas a lo largo del proyecto para un desarrollo eficaz, tanto en la construcción del marco teórico, en los diseños del proyecto y aun en la etapa de conclusiones y recomendaciones.

En el capítulo 3, con base a la teoría del capítulo 1 y la metodología del capítulo 2 se crea la propuesta al usar variables iniciales, representaciones gráficas, y el desarrollo integral de la solución al problema planteado.

En el capítulo 4 se implementa la solución, donde se describe el proceso del desarrollo de la solución, se explica y detalla la solución implementada y se muestran pruebas de funcionamiento y el análisis de resultados de acuerdo a los objetivos.

Finalmente, se muestran las conclusiones como una valoración de cumplimiento de los objetivos y el alcance planteado en el plan integrador de carrera. Además, se escriben recomendaciones para el uso y posible mejora de la solución presentada.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

El presente proyecto tiene como objetivo proveer servicios de videoconferencia, mensajería instantánea, compartición de archivos y pantalla sobre un navegador sin requerir programas de terceros o plugins adicionales. Para el desarrollo, se debe definir primero los conceptos relacionados a cada una de las tecnologías como videoconferencia, protocolo WebRTC, protocolos web, lenguajes de programación web, servidor de despliegue de prueba. Estos conceptos serán la base para la propuesta e implementación del proyecto.

1. Videoconferencia

La videoconferencia es un medio de comunicación donde el sonido está acompañado por una imagen de video, es también conocida como video teleconferencia o VTC. (Chacón, 2003) La videoconferencia no es una invención reciente, pues ya fue prevista por uno de los inventores del teléfono, Bell, que mencionó en 1924 “el día vendrá en que la persona al teléfono podrá ver a la persona en el otro lado con la cual se habla.” Citado por (Chacón, 2003)

A pesar de ser visionada desde los finales de 1920, la videoconferencia ha crecido en la actualidad por las tecnologías nuevas y conexiones de internet más rápidas con menores costos. Las unidades de videoconferencia proveen audio y video de calidad por los métodos de compresión, modulación y transporte, además, con la mejora en los dispositivos relacionados a la videoconferencia, tanto equipos de computación como videocámaras, permite que la calidad aumente mientras que los costos disminuyen.

El sistema de videoconferencia está compuesto por dos componentes digitales de audio y video.

1.1 Audio digital

El audio digital es la representación de señales sonoras mediante un conjunto de datos binarios. (Chacón, 2003) El sistema tiene en su extremo inicial un transceptor de audio (micrófono) y transforma la onda de sonido a una señal analógica. La señal analógica pasa por un sistema de proceso analógico en el que se realiza limitaciones en frecuencia, ecualización, amplificación. La ecualización tiene como objetivo contrarrestar la respuesta en frecuencia del transceptor para que la forma analógica de salida tenga una forma de audio similar a la original, la ecualización permite modificar el contenido de frecuencias de la señal y la amplificación permite una señal con más potencia.

El proceso analógico de la señal realiza muestreo, cuantificación y codificación. Los muestreos toman valores de señal analógico por segundo, la cuantificación asigna valores analógicos a esas muestras lo que implica pérdida de información y la codificación realiza la asignación de secuencias de bits a cada valor, la longitud de la palabra de bits está relacionada al número de niveles analógicos. El número de bits y la tasa de muestreo son los parámetros básicos para procesar una señal de audio.

Finalmente, para entregar el audio al receptor se plantea el proceso inverso, en el que las muestras digitales se transforman a señales analógicas que llegan como ondas de sonido al completar así el ciclo de comunicación.

1.1.1 Fundamentos de audio digital

El audio digital se conforma por una secuencia de muestras que representan el sonido. Los parámetros de esta secuencia del audio digital son: el número de canales con un canal para mono, dos para estéreo, cuatro para el cuadrafónico. La tasa de muestreo corresponde al número de muestras tomadas cada segundo en cada canal y el número de bits por muestra que se usan son 8 o 16. Se considera audio digital multicanal al audio que trabaja con tres o más canales, las muestras de audio multicanal suelen organizarse en conjuntos llamados tramas, los cuales se envían a través del canal.

1.1.2 Técnicas de compresión

De acuerdo a (Loza, 2014) las técnicas de compresión están relacionadas a los formatos de audio digital y son fundamentales para alcanzar el compromiso entre capacidad de almacenamiento y procesamiento. Entre las técnicas de compresión más relevantes se tienen:

a) ADPCM (Adaptive Differential Pulse Code Modulation, Modulación adaptativa diferencial de pulsos de código). Es una codificación conocida como diferencial ya que almacena la diferencia entre los valores de muestras consecutivas, en el caso de la señal de audio la diferencia suele ser pequeña. ADPCM se articula en los estándares CCITT G.721, CCITT G.723 y en el CCITT G.726, este último reemplaza a los dos anteriores y define estándares para 16, 24, 32 y 40 kbits por segundo que corresponden a tamaños de muestra de 2, 3, 4 y 5 bits respectivamente. (Loza, 2014)

b) LPC-10E (Linear Predictive Coder, Codificador predictivo lineal). Este algoritmo hace corresponder la señal de audio con un modelo lineal simple y recupera los parámetros que se ajustan mejor al modelo a la señal, esta señal generada no es una representación exacta de la original por el tipo de codificación. Este codificador aún se utiliza en servicios de voz de telefonía básica (Loza, 2014)

c) CELP (Code Excited Linear Prediction, Codificador predictivo de salida de código). Es un estándar similar a LPC-10E, con la diferencia de que en este estándar se toma en cuenta el error entre la señal original y la señal aproximada con lo que se crea una tabla de errores. La señal se compone el parámetro modelo y el parámetro índice del error en cada muestra. (Loza, 2014)

d) GSM 06.10 (Global Solution Mobile, Solución móvil global). Es una modificación de LPC conocida como RPE-LPC (Regular Pulse Excited – Linear Predictive Coder, Codificador lineal predictivo de salida de pulso regular). La compresión puede ser muy elevada, hasta 13kbits/s por muestra de 260 bits, pero requiere también de un alto índice de procesamiento. (Loza, 2014)

e) MPEG (Moving Picture Experts Group, Grupo expertos en imágenes en movimiento). Es un estándar de para audio y vídeo con alta compresión de datos, por lo que requiere de alta potencia de cálculo en su mayoría para el proceso de codificación. Se definen tres capas (layers) para las versiones MPEG-1 y MPEG-2: capa 1 desde 32 a 448 kbps, capa 2 desde 32 a 384 kbps, capa 3 desde 32 a 320 kbps. (Loza, 2014)

1.1.3 Calidad de audio digital

La calidad de audio digital se puede medir al comparar el audio original de tipo análogo con respecto al audio digitalizado para obtener una relación señal a ruido concreta. El número de bits por muestra que se utilice para la codificación corresponde a un número de niveles de cuantificación que están asociados a esa relación señal a ruido.

La relación señal a ruido es un parámetro empírico en el que se mide la calidad de la señal de audio original en relación a la calidad percibida por el oído humano, al depender del factor humano en esta relación se sabe que puede variar al depender del sujeto que escuche las señales para hacer la comparación.

En la figura 1.1 se puede ver una representación simplificada de la digitalización, donde la señal continua toma un valor discreto en cada punto del eje x.

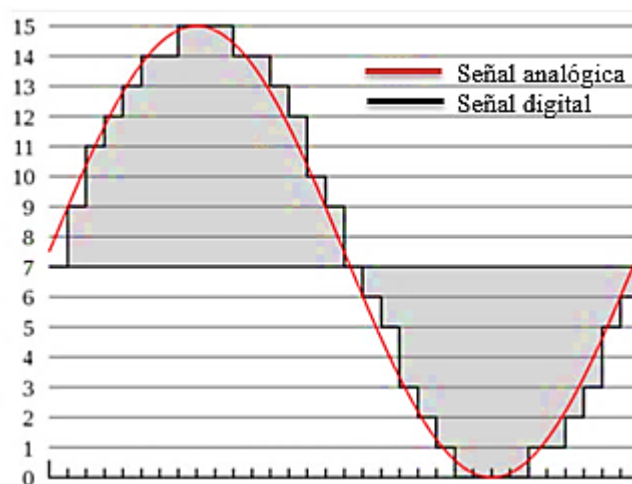


Figura 1.1 Representación simplificada del proceso de digitalización

Fuente: (Rasheed, 2012)

1.2 Video digital

El video digital es una representación digital de una sucesión de imágenes que conforman el video analógico, la calidad de reproducción de un sistema digital de video que está bien diseñado es independiente del medio y depende solamente de la calidad de los procesos de conversión. Existen varios tipos de video que se emplean en el campo digital, con RGB y YUV como los más representativos.

El término RGB se deriva de los tres colores básicos Red-Green-Blue (Rojo-Verde-Azul), estos colores son utilizados en el tipo de video RGB, obteniéndose el resto de colores como combinación de ellos. Con este formato se puede conseguir una máxima calidad, sin embargo, requiere de una gran cantidad de ancho de banda. Este formato se especifica al usar las siglas seguido de un número que indica la profundidad del color en bits (Ej. RGB32, profundidad de color de 32 bits).

Debido a la gran demanda de ancho de banda que requiere el formato previo, surge la idea de realizar una transformación lineal las componentes RGB a unas equivalentes Y, CB, CR conocidas como (Y, U, V); este sistema es la base de los sistemas digitales actuales y se considera como un sistema de compresión al transformar la señal RGB a la cual se comprime las componentes de color y se preserva la luminosidad que es la componente más sensible el ojo humano.

El sistema de compresión YUV viene incluido en las tarjetas de video para dar una funcionalidad sin la necesidad de instalar un códec, así mismo varios de los códecs utilizados para lograr mayor compresión. En la figura 1.2 se puede ver una comparación entre las componentes YUV y RGB desde una imagen original.

Después del proceso de compresión se debe realizar el proceso de digitalización para enviar la imagen por el medio digital se usa tres herramientas: muestreo, cuantificación y codificación.

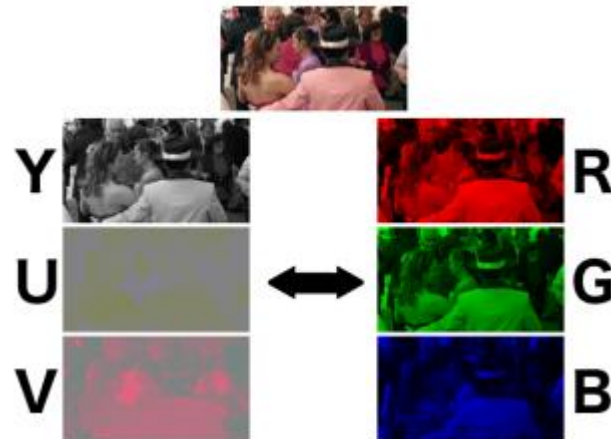


Figura 1. 2 Componentes YUV para imagen base

Fuente: (Rodríguez y Fernández, 2016)

1.2.1 Muestreo

Para la señal analógica de la figura 1.3 se realiza el proceso de muestreo que consiste en la toma de muestras en un intervalo pequeño de tiempo, 2 -3 segundos en el ejemplo, con amplitud correspondiente a la forma de la señal.

Una señal de video tiene varias componentes de frecuencia que van en el rango de 0 a 5 MHz, al momento del muestreo de la señal las frecuencias de video se tienen por encima o debajo de la frecuencia de muestreo, que es el número de muestras por unidad de tiempo que se toman de la señal. La banda base de referencia es el armónico 0, es decir, la frecuencia con mayor amplitud de la señal. Este proceso también genera pérdidas en la señal, debido a las discriminaciones de frecuencia que se realizan. En la figura 1.3 se puede ver el muestreo de una señal analógica, donde en cada intervalo se ejecuta un pulso que adquiere la amplitud de la señal original.

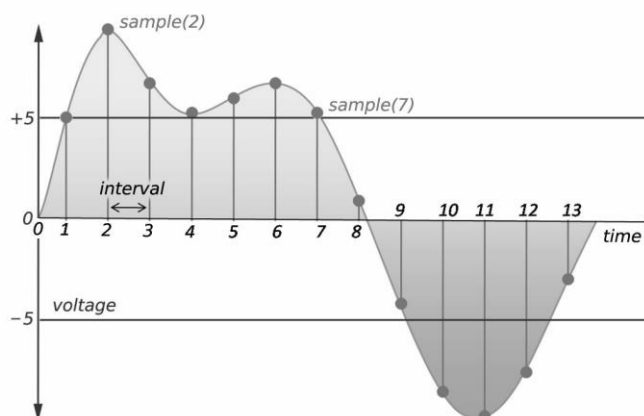


Figura 1.3 Muestreo para una señal analógica

Fuente: (Rasheed, 2012)

1.2.2 Cuantificación

Al usar los valores obtenidos en el proceso de muestreo se realiza el proceso de cuantificación; este proceso consiste en la asignación de un conjunto de bits que está relacionado al número de profundidad de color. Se usa muestras de 8, 10 o 12 de bits.

El proceso de cuantificación puede producir un error conocido como ruido, en el que se introducen frecuencias espurias a la entrada o salida del proceso de cuantización. El ruido puede ser blanco por presencia de señales aleatorias y se aproxima a las propiedades estadísticas de distribución del error, si el error no se encuentra en el rango estadístico se considera la existencia de distorsión en la señal.

1.2.3 Codificación

Posterior a la etapa de cuantificación se aplica la codificación para reducir el flujo binario que puede encontrarse en el orden de los cientos de megabits por segundo. La codificación facilita la transmisión de video por internet ya que la compresión reduce el ancho de banda requerido y al mismo tiempo mantiene una buena calidad. Además, la codificación permite la compatibilidad para asegurar que los videos puedan ser utilizados por diferentes servicios o programas que requieren ciertas especificaciones de codificación conocidas como códecs.

Los códecs son estándares de video compresión que se realizan mediante hardware o software, y no deben ser confundidos con los contenedores que se usan para encapsular el producto final, entre los cuales se tiene, MOV, AVI, etc.

Para realizar la codificación se debe distinguir entre dos tipos de datos presentes en materiales de video

a) Los datos entrópicos. Se tratan de los datos nuevos y son realmente la información necesaria.

b) Los datos redundantes. Son datos irrelevantes debido a redundancias y pueden ser anticipados al ser datos comunes para cada frame.

Al tener en cuenta esta distinción entre datos necesarios e innecesarios se puede establecer una clasificación de carácter general para los sistemas de codificación:

a) En el sistema sin pérdida de codificación. Trata de enviar únicamente los datos entrópicos, pues como se puede prescindir de los datos redundantes.

b) En el sistema con pérdida. Al igual que ocurre en los sistemas sin pérdida el codificador envía únicamente los datos entrópicos al decodificador; la diferencia radica en que en este sistema se prescinde también de otros datos que, si bien no son redundantes, no son tan esenciales y se puede prescindir de ellos sin alterar excesivamente la calidad del vídeo final. (Rodríguez y Fernández, 2005)

1.2.3.1 Técnicas de codificación

Se pueden mencionar tres técnicas de codificación principales: la compresión espacial, la compresión temporal y la codificación bidireccional. (Rodríguez y Fernández, 2005)

a) La codificación espacial. Se apoya en que la similitud de píxeles próximos en imágenes fijas y tiene como resultado un envío de información irrelevante. Mediante un

análisis de las frecuencias espaciales y a través de una transformada de Fourier bidimensional se puede aprovechar este efecto.

b) La codificación temporal. Se basa en que, a menudo, las imágenes que se suceden en el tiempo son muy similares, de forma que el procesado individual de cada una de ellas puede resultar un tanto ineficiente. En la figura 1.4 se tiene un diagrama de la codificación temporal.

Al aprovechar este aspecto, se propone tratar la diferencia que se produce entre dos imágenes consecutivas.

c) La codificación bidireccional. En esta se tiene una previsión de una imagen basada en la imagen anterior y la siguiente. Este proceso complica los algoritmos con los que se modifica, pero se compensa al obtener altos niveles de compresión.

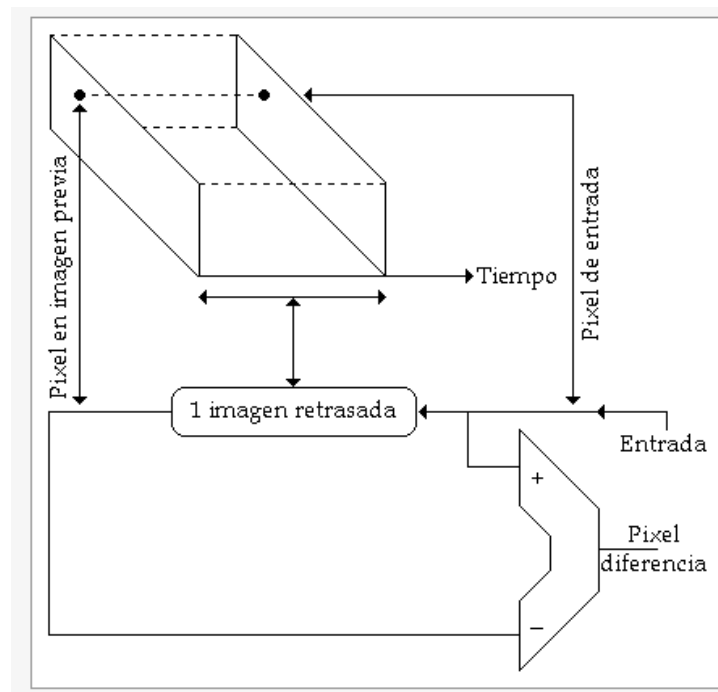


Figura 1. 4 Esquema de codificación temporal

Fuente: (Rodríguez y Fernández, 2005)

Esta codificación define que si un objeto superpone a otro en una imagen y el objeto que está oculto se muestra lentamente entonces existe transmisión de nueva información de

forma constante al revelarse del objeto. Este fenómeno es aprovechado por la codificación bidireccional para disponer de imágenes previas o posteriores. En la figura 1.5 se puede observar un diagrama de la codificación bidireccional.

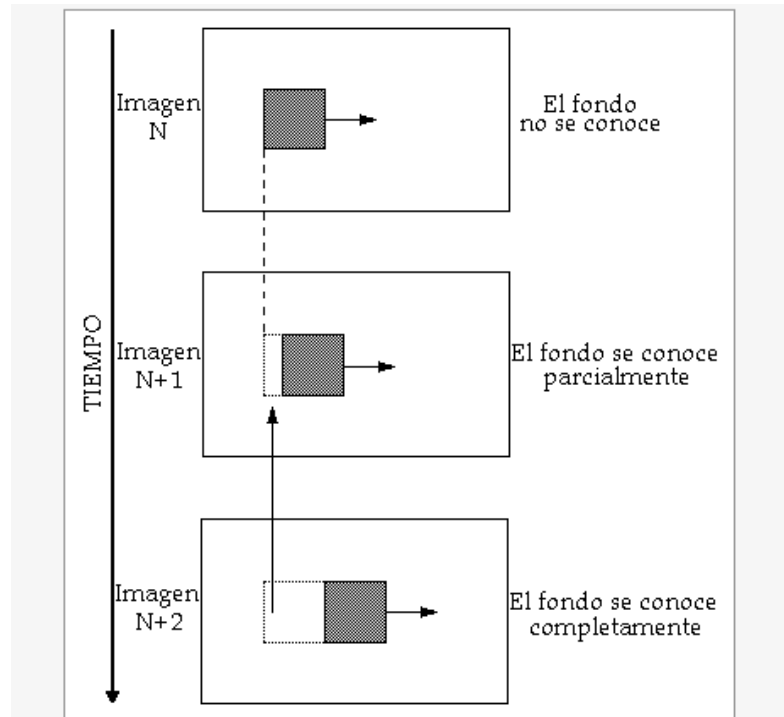


Figura 1. 5 Diagrama de codificación bidireccional

Fuente: (Rodríguez y Fernández, 2005)

1.2.4 Códecs de video

El término códec proviene de la contracción de Coder y Decoder, es decir el programa que permite la codificación y decodificación en un formato digital específico para el usuario. (Yáñez, 2011) Para realizar este proceso se utilizan algoritmos que disminuyen la longitud del flujo digital y lo adecuan a los medios de almacenamiento y transmisión suponible.

Como todo proceso que disminuye el tamaño, está sujeto a pérdidas que pueden afectar la calidad del producto final, por esto se desarrollaron códecs con pérdida y sin pérdida. Entre los códecs más relevantes se tienen Indeo, MPEG 1,2 y MPEG 4.

1.2.4.1 Indeo (INtel viDEO)

Basado en Wavelet, Indeo es un códec que realiza una compresión temporal que permite la reproducción de vídeo a una velocidad máxima sin necesidad de utilizar aceleración de hardware.

El licenciamiento de los códecos Indeo es propiedad de Ligos, aunque fue desarrollada por Intel. La última versión es el Indeo 5.10, mucho más rápida y eficaz que la Indeo 4.5, las pruebas muestran que efectivamente se obtiene una calidad muy buena de imagen y además tanto el tiempo de codificación como el tamaño del vídeo resultante son menores.

1.2.4.2 MPEG 1 y 2

MPEG1 se usa en tasas de bits más bajas que MPEG2, ya que este último está dirigido a vídeos de alta calidad (DVD, por ejemplo). Para una tasa de datos baja, MPEG 1 ofrece mejores resultados que MPEG 2, la hacer una analogía, MPEG2 es a DVD como MPEG1 es a CD, ya que es la base del VCD, Video CD, un formato de compresión de vídeo cuya tasa de datos es similar a la de un CD de audio (por lo que la capacidad suele compararse con la de este: "un CD virgen de 74 minutos de música" o 74 minutos de vídeo codificado con VCD). La calidad del VCD es comparablemente menor a la de una cinta VHS, por lo que en ocasiones se utiliza el formato SVCD (Super Video Contact Disk, Disco de contacto de super video), el cual es un formato de codificación basado en MPEG2, al ofrecer una calidad de imagen superior (Rodríguez y Fernández, 2005). Esto también implica que la capacidad necesaria para almacenar la misma cantidad de video es mayor.

Se puede concluir que MPEG 2 implica una calidad superior mientras que MPEG 1 mantiene una tasa de datos baja.

1.2.4.3 MPEG4

La codificación basada en MPEG4 realiza codificación temporal y espacial. Utiliza la implementación de diversas técnicas que se verán más adelante.

En la compresión temporal de MPEG 4 se utilizan 3 tipos de imágenes:

a) I-frames/keyframes: estos son los cuadros "clave" son imágenes enteras, individuales, comprimidas de forma independiente del resto.

b) P-frame: son imágenes deducidas de la imagen anterior o de la posterior, estas imágenes no se almacenan por completo debido a que son generadas a partir de otros P-frames e imágenes clave.

c) B-frame: Al igual que los I-frames y P-frames estas imágenes se crean al usar información de los frames anteriores y sucesivos.

2. Protocolo WebRTC

El protocolo WebRTC (Web Real Time Communication, Comunicación web en tiempo real) es un estándar abierto que embebe las capacidades de comunicación multimedia en tiempo real directamente en un navegador web (WebRTC, 2011). El protocolo y los estándares que maneja son abiertos y eliminan la necesidad de descargar software cliente y el uso de plugins.

Este protocolo se desarrolla y adecua de acuerdo a las herramientas de despliegue en las que se va a implementar, en este caso los exploradores. En la actualidad, tiene implementaciones avanzadas para Chrome, Firefox y Opera; para el presente trabajo se utilizará específicamente los exploradores Mozilla y Chrome.

En un análisis disruptivo que se realizó se determina que para finales del año 2018 los usuarios individuales de WebRTC llegarán a los 1.000 millones y las computadoras, teléfonos inteligentes y tabletas que soporten e implementen el protocolo WebRTC incrementará el número a 4.700 millones. (WebRTC, 2011)

El protocolo Web RTC maneja sus propios códecs libres, que dan el tratamiento de los datos de audio y video a ser transmitidos al establecer una conexión punto a punto.

2.1 Códec de audio WebRTC

WebRTC utiliza tres tipos de audio códec:

a) Opus audio códec. Es un códec libre definido en el IETF RFC 6176 que permite codificación de bitrate (tasa de bit) constante y variable de 6 kbit/s a 510 kbit/s con tamaños de trama entre 2.5 ms a 60 ms y varias muestras de velocidad desde 8 kHz a 48 kHz con un ancho de banda de 20 kHz que cubren el rango completo del sistema auditivo humano.

b) iSAC (Internet Speech Audio Codec, Códec de audio de habla para Internet). Es un códec de voz robusto de ancho de banda adaptativo, banda ancha y súper banda ancha desarrollado por Global IP Solution, usado en varias aplicaciones VoIP y streaming de audio. Es utilizado en con industrias líderes de VoIP y como parte del proyecto WebRTC.

c) iLBC (Internet low bit rate code, Código de tasa baja de internet). Es un códec de banda angosta desarrollado por Global IP Solution, al usar en varias aplicaciones de VoIP y streaming. En el año 2004, el códec se registró en IETF RFC y se volvió disponible, de igual manera este código está incluido en WebRTC.

2.2 Códec de Video WebRTC

Debido a que el protocolo utilizado en este proyecto es WebRTC, es necesario especificar el códec que se utiliza en el mismo. El códec de Web RTC es conocido como V8

y permite a los navegadores transmitir video en tiempo real sin la necesidad de complementos adicionales.

El códec V8 es relativamente nuevo, fue lanzado en el 2013 como un códec libre para desarrollo por parte de la empresa Google. La calidad de video de V8 está entre las mejores, ya que no tiene un límite de frame o datos. Utiliza 14 bits tanto para ancho y alto, lo que hace que la máxima resolución llegue a 16384 x 16384 pixeles, que es muy alta comparada a la resolución de otros códecs, se puede tomar como referencia YouTube, con su máxima resolución en formato 4K es de 4096 x 2160; otra referencia puede ser Dailymotion, cuya máxima resolución es de 920 x 1080.

2.3 Seguridad en WebRTC

WebRTC al ser una tecnología de código abierto, genera inseguridad en los usuarios que desean compartir información confidencial en llamadas, videoconferencia y compartición de pantalla. Para esto, Web RTC tiene características nativas de seguridad que manejan problemas de seguridad, así como cifrada de extremo a extremo para garantizar videoconferencias privadas y seguridades

Para que WebRTC transfiera datos en tiempo real, los datos primero se cifran con el método DTLS (Datagram Transport Layer Security, Seguridad en capa de transporte de datagramas). Este es un protocolo integrado en todos los navegadores soportados por WebRTC desde el principio (Chrome, Firefox y Opera). En una conexión DTLS cifrada no se puede manipular la información o acceder a ella de forma ilegal.

Aparte de DTLS, WebRTC también encripta los datos de vídeo y audio a través del método SRTP (Secure Real-Time Protocol, Protocolo en tiempo real seguro) que garantiza seguridad para las comunicaciones IP – su voz y el tráfico de vídeo – no puedan ser escuchados o vistos por terceros no autorizados. (WebRTC, 2011)

2.3.1 DTLS (Datagram Transport Layer Security, Seguridad en capa de transporte de datagramas).

El DTLS es un protocolo de comunicaciones que se deriva del protocolo SSL y provee los mismos servicios de seguridad los cuales son integridad, autenticación y confidencialidad bajo el protocolo UDP (User datagram protocol, Protocolo de datagrama de usuario).

El protocolo UDP es un protocolo que corresponde a la capa transporte en el modelo OSI. Este protocolo se caracteriza por el envío de datagramas sin previo establecimiento de conexión, tampoco se envía información de control ni control de flujo, por lo que se utiliza principalmente para el envío de voz donde se la relación pérdida/velocidad es justificada.

Por tanto, el protocolo DTLS ofrece un nivel de seguridad en el transporte de voz para el protocolo Web RTC y asegura que los datagramas enviados por UDP estén protegidos. DTLS está definido en RFC 6347 (RFC6347, 2012)

2.3.2 SRTP (Secure Real Time Protocol, Protocolo de seguridad en tiempo real)

El SRPT es un protocolo de seguridad en tiempo real definido especialmente para las comunicaciones de voz sobre IP. Está definido en la RFC 3711 (RFC3711, 2004)Es una extensión de RPT, el protocolo de transporte en tiempo real, que incorpora seguridades sobre las características de RPT, al ser las principales:

- a) Diseñado para transmisión de información en tiempo real
- b) Compensación de latencia y detección de pérdida de paquetes, para compensar transmisiones en UDP.
- c) Permite transferencia de datos por IP multicast (RFC3711, 2004)

Las funciones de seguridad que se implementan sobre RPT son encriptación y autenticación que evitan ataques DOS (Denegación de servicio, por sus siglas en inglés). Los ataques DOS pueden afectar las comunicaciones debido a que llenan el enlace de comunicación con paquetes y evitan la transmisión de información entre las conexiones extremo a extremo.

2.3.3 Soporte WebRTC

WebRTC está acondicionado para su implementación sobre exploradores de Internet, por tanto, se debe definir cuáles son los exploradores que soportan la ejecución de una aplicación diseñada sobre este protocolo. En la actualidad, WebRTC se encuentra soportado por Google Chrome, Mozilla Firefox y Opera.

WebRTC apunta especialmente al desarrollo de aplicaciones web de videoconferencia, por lo que resta definir los lenguajes de programación y el alojamiento para la aplicación de videoconferencia.

3. Aplicación web

Una aplicación web es una aplicación informática que se ejecuta en un entorno web exclusivamente. Es una aplicación de tipo cliente/servidor, donde el cliente es el navegador y se conecta al servidor web mediante el protocolo HTTP. (Conallen, 2000)

3.1 Protocolo HTTP

El protocolo HTTP (Hyper Text Transfer Protocol, Protocolo de transferencia de hipertexto) es un protocolo que permite la comunicación entre clientes y servidores en la web. La comunicación se realiza mediante peticiones y respuestas. La petición es emitida por el cliente y procesada por el servidor, que devuelve una respuesta. Las peticiones y respuestas se realizan mediante URLs.

El protocolo HTTP es un protocolo de la capa aplicación del modelo OSI y que trabaja sobre el protocolo TCP para la información; por su extensibilidad, es usado para presentar páginas web de texto, elementos multimedia, audio, video e incluso estado de funcionamiento del servidor. (RFC2616, 1999)

En una sesión cliente/ servidor, el protocolo HTTP permite controlar:

- a) Autenticación, mediante el uso de cabeceras y cookies
- b) Cache, memoria local donde se almacenan copias de una página que se visita frecuentemente, para que sea accesible, aunque no se tenga acceso al servidor en el momento de realizar la petición.
- c) Sesiones, a pesar de que el protocolo por definición no tiene estado, permite establecer sesiones con el uso de cookies para mantener una conexión permanente entre un cliente y un servidor. Esto es particularmente útil para aplicaciones de bancos y comercio electrónico

3.2 Lenguajes y herramientas de programación web

Para el desarrollo de páginas y aplicaciones web se requiere lenguajes de programación dinámicos que permitan al usuario la interacción con los elementos de la aplicación, así como la presentación de elementos multimedia, sesiones de usuario y comunicaciones entre usuarios mediante la aplicación.

Las herramientas de programación corresponden a las librerías o códigos creados con base a lenguajes de programación que permiten crear una interfaz de usuario, darle funcionalidad y realizar acciones en el lado del cliente y del servidor con el objetivo de desarrollar una página o aplicación web. A continuación, se detalla los lenguajes y herramientas que se aplicarán al desarrollo del proyecto actual.

3.2.1 HTML

HTML (HyperText Markup Language, Lenguaje de Marcado para Hipertextos) es el lenguaje de programación más básico para la construcción de una página web, al determinar el contenido, los espacios y el estilo y al permitir la creación y visualización de los elementos que conforman la página.

Los vínculos en las páginas web son los elementos clave que establece la diferencia con las aplicaciones de escritorio que no permiten la navegación entre aplicaciones como se puede hacer en páginas web, el hipertexto hace referencia a los enlaces que conectan la

página web con otras. HTML utiliza el marcado para etiquetar los elementos que componen una página, se tiene por ejemplo <head>, <body>, <footer> que son las etiquetas que conforman la cabecera, cuerpo y pie de la página web respectivamente. Las principales etiquetas HTML se listan en la tabla 1.

Tabla 1. 1 Principales etiquetas HTML

ETIQUETA	USO
<html> </html>	Define el inicio y final de la página, es la primera etiqueta que se usa en el código HTML
<head></head>	La etiqueta encierra el contenido de la cabecera, aquí también se define las hojas de estilo y las librerías de JavaScript si son utilizadas.
<body></body>	Encierra el contenido del cuerpo de la página, donde se tiene el diseño que va a ser presentado al usuario
<p> texto </p>	La etiqueta se usa para delimitar un párrafo
 Nombre del enlace	La etiqueta permite crear un enlace que puede ser interno en el mismo sitio web o externo hacia otro enlace de internet.
	Permite incluir una imagen en la página, el atributo scr corresponde a la dirección de la carpeta de la imagen.
 	Con ayuda de la etiqueta <a> se puede hacer que la imagen también se convierta en un enlace
<div> ELEMENTO </div>	La etiqueta div permite encerrar un elemento que puede ser de tipo texto, imagen o un conjunto de elementos. Esta etiqueta es utilizada para distribuir el contenido de la página de forma ordenada con posiciones y estilos.

<pre><table > <td> <tr></tr> <tr></tr> </td> </table></pre>	La etiqueta permite crear una tabla, para crear una columna se usa la etiqueta <td> y para crear filas se usa la etiqueta <tr>
---	--

Fuente: El autor

3.2.2 JAVA SCRIPT (JS)

JavaScript es un lenguaje de programación con el cual se realiza actividades complejas para páginas y aplicaciones web, por ejemplo, mostrar actualizaciones de contenido en tiempo real, interactuar con usuario mediante alertas, animaciones gráficas 2D/3D etc. Es la tercera capa de los estándares en las tecnologías para la web, dos de las cuales son HTML y CSS. (Developer Mozilla, 2019) Entre las características de JavaScript se tienen las siguientes:

- a) Maneja el almacenamiento de valores en variables, que luego pueden ser usadas para un método o enviadas al servidor.
- b) Permite desarrollar la programación para el lado del cliente, donde se puede mostrar al cliente mensajes de confirmación sobre acciones (por ejemplo, al cerrar la ventana que se muestre un mensaje de confirmación) y manejo de acciones (por ejemplo, controlar que acción se ejecuta al dar clic en un botón).
- c) Se apoya en API's (Application programming interfaces, Interfaz de programación en aplicaciones) que son códigos en forma de librerías para implementar funcionalidades existentes mediante una referencia al API. Como ejemplo de API's se tiene DOM (Document Object Model, Modelo de objeto de documento) que manipula el código HTML y CSS, Localización-Geo ayuda a establecer la información geográfica y Canvas para la creación de gráficos 3D y animaciones.

En la figura 1.7 se puede ver un ejemplo de código escrito en JavaScript. En este, se recupera selecciona una etiqueta <p> y se actualiza el texto de la etiqueta con el nombre ingresado por el usuario en un cuadro de dialogo mediante una función updateName().

```
var para = document.querySelector('p');  
  
para.addEventListener('click', updateName);  
  
function updateName() {  
    var name = prompt('Enter a new name');  
    para.textContent = 'Player 1: ' + name;  
}
```

Figura 1. 6 Segmento de código de ejemplo de código JavaScript

Fuente: (Developer Mozilla, 2019)

JavaScript es un código cinámico pues permite la actualización de una página de acuerdo a las acciones que el usuario realiza. Para definir el código JS se utiliza la etiqueta <script> dentro de la cabecera de la página.

3.2.2.1 Node.JS

Node JS es un entorno de ejecución para JavaScript desarrollado por el motor de JavaScript V8 de Google Chrome (NodeJS, 2019) orientado a eventos asíncronos. Se utiliza para la construcción de aplicaciones con conexiones concurrentes, es decir, que tienen varios usuarios al acceder simultáneamente a la aplicación.

Node JS no maneja librerías, sino que presenta un bucle de eventos al que se accede mediante un *callback* (llamada) con todas las funcionalidades ocultas al usuario, este entorno es usado para aplicaciones de streaming donde se maneja audio y video por la baja latencia. Para la gestión de paquetes se utiliza NPM (Node Package System, Sistema de paquetes de Nodos) con el cual se puede instalar y actualizar las API's y dependencias con comandos simples.

3.2.2.2 RTCMulticonnection

RTCMulticonnection es una librería JavaScript para aplicaciones punto a punto que permite compartición de pantalla, videoconferencia, compartición de archivos, mensajería instantánea entre otras funciones. Está disponible en GitHub y desarrollada por el MIT (Massachusetts Technologic Institute, Instituto tecnológico de Massachusetts) de código abierto para el manejo de las funcionalidades de WebRTC de una forma sencilla. (GitHub, 2016) Entre las características de la API se tiene que:

- a) Está basada en Node JS y comprende una librería que se instala mediante NPM en entornos de desarrollo Windows, Linux y MacOS.
- b) Soporta canales para datos y streaming de audio o video. Estos canales se establecen y permiten separar comunicaciones entre usuarios para evitar el solapamiento de la funcionalidad de la aplicación.
- c) Actúa como un wrapper de código (portabilidad de código) para que este mantenga su funcionalidad en los diferentes exploradores sin necesidad de escribir comandos específicos para cada explorador.

3.2.2.3 jQuery

jQuery es una API de JavaScript que facilita la interacción con HTML en una página web, al simplificar los comandos de código original de JavaScript con soporte en una gran cantidad de exploradores. Permite manipular objetos de dominio HTML, desarrollo de animación e interacciones.

jQuery es software libre bajo la licencia MIT para proyectos públicos y privados, la sintaxis de jQuery es sencilla de aprender e implementar y permite crear páginas web dinámicas y aplicaciones web. (jQuery Foundation, 2019). En la figura 1.8 se muestra un ejemplo de código escrito en jQuery que recupera los elementos de una tabla HTML.

```
$("#tablaAlumnos"); // Devolverá el elemento con  
id="tablaAlumnos"  
$(".activo");      // Devolverá una matriz de  
elementos con class="activo"
```

Figura 1. 7 Segmento de código de ejemplo de jQuery

Fuente: (jQuery Foundation, 2019)

3.2.3 CSS (Cascading Style Sheets, Hojas de estilo en cascada)

CSS es un lenguaje de programación para el diseño de la presentación de una página o aplicación web. CSS utiliza el atributo class de las etiquetas HTML para definir estilos de cada uno de los elementos por separado. Los estilos se utilizan las propiedades de stlye (estilo) de los diferentes elementos HTML por ejemplo width (ancho), height (alto), color (color de fondo), font (tipo de letra). No todos los elementos tienen los mismos atributos por lo que se recomienda crear estilos separados en dependencia del elemento que se va a diseñar. En la figura 1.9 se muestra un estilo para una etiqueta <p> donde se establece el color de fondo rojo y la alineación del texto centrada.

```
p {  
  color: red;  
  text-align: center;  
}
```

Figura 1. 8 Ejemplo de código CSS

Fuente: (W3Schools, 2019)

3.2.3.1 Semantic UI

Semantic UI es un framework con más de 3000 variables CSS, elementos y niveles de jerárquicos para hojas de estilo. Permite a los desarrolladores construir sitios web con un diseño profesional, código HTML conciso, uso de JavaScript y depuración.

Semantic también se caracteriza por ser escalable a diferentes dispositivos, ya que no es lo mismo un diseño para una pantalla de computadora que para una pantalla de un teléfono inteligente, por lo que Semantic ayuda a la transición del sitio web entre diferentes tamaños de pantalla al mantener el diseño y la funcionalidad. (GitHub, 2019)

CAPÍTULO 2

MARCO METODOLÓGICO

El proyecto utilizará una metodología descriptiva que se usará para detallar los aspectos de la situación actual, las variables a ser consideradas y las necesidades que se presentan en el ambiente de prueba.

La metodología descriptiva está conformada por una descripción, un registro, el análisis e interpretación de la naturaleza actual, la composición o proceso un fenómeno. El enfoque de la metodología establece conclusiones dominantes sobre un grupo de personas, un grupo o cosas y se conduce o funciona en tiempo presente. (Tamayo, 2015)

Los estudios realizados por metodología descriptiva describen características de una población, lugar o fenómeno, sin importar las razones por las cuales se han dado estas características. Toda la información obtenida será utilizada para definir el objeto de estudio y así encontrar variables que permitirán desarrollar el proyecto de manera que satisfaga las necesidades y se ajuste al estado actual.

Con la información y variables obtenidas en la etapa de investigación descriptiva, se realizará el diseño al usar la metodología proyectiva, la cual se define como una modalidad de la ciencia determinada por el propósito de elaborar propuestas susceptibles de ser llevadas a feliz término. Con la metodología proyectiva se proponen soluciones a problemas encontrados para el desarrollo e implementación del proyecto. (Barrera, 2010)

El diseño e implementación del presente proyecto permitirá dar una solución viable y eficaz para el uso de tele consultas por videoconferencia, que sean útiles para la población de estudio. Para asegurar el cumplimiento de los objetivos planteados, se utilizará la investigación confirmatoria, que compone la etapa de pruebas que validan el correcto funcionamiento del proyecto, y se complementa con las conclusiones y recomendaciones.

2.1 Metodología aplicada para las fases del proyecto

La primera fase del proyecto es la fase exploratoria donde se realiza la revisión de los conceptos de videoconferencia, protocolo WebRTC y las librerías que se pueden utilizar para su implementación, lenguajes de programación web y posibles servidores de dominio para el despliegue y prueba de la aplicación.

En la fase de descripción explicativa se revisan las metodologías tradicionales y metodología de desarrollo SCRUM para el desarrollo de software

La fase de encuestas no se aplica a este proyecto, en su lugar se realiza un análisis de la funcionalidad de las aplicaciones de videoconferencia tradicionales, particularmente Skype y mediante un diagrama de flujo se describe el proceso que un usuario debe seguir para el uso de la aplicación. También se realiza el análisis de la funcionalidad esperada de la aplicación WebRTC y se realiza un diagrama de flujo, la comparación de ambos diagramas mostrará la mejora en la eficiencia del uso entre la aplicación tradicional y la aplicación propuesta.

En la fase de construcción de la propuesta se utiliza la metodología SCRUM para el desarrollo de sprints que permiten un desarrollo iterativo de la aplicación, además se utilizan diagramas UML de casos de uso para modelar el uso y las acciones de los usuarios de la aplicación y diagramas de clase para identificar las variables y métodos que se requieren en cada una de las páginas que componen la aplicación.

Las variables y los métodos se definen de acuerdo a los requerimientos que se identifican en los casos de uso, los métodos incluyen también validaciones y el acceso a la aplicación.

2.2 Metodología Scrum Ágil

El desarrollo de software ágil se refiere a un grupo de metodologías de desarrollo basadas en un desarrollo iterativo donde los requerimientos y las soluciones evolucionan a

medida que avanzan las iteraciones. Una de las características más llamativas es que permite el desarrollo de software con eficiencia de tiempo y recursos.

Scrum es un subconjunto de ágil que consiste de un framework para desarrollo ampliamente usado que incrementa la calidad de los entregables en cada interacción y permite mantener un control de la organización del proyecto y el estado actual.

Normalmente la metodología Scrum organiza a un equipo de desarrolladores, pero se puede aplicar para un solo desarrollador, que aplica para este proyecto con la consideración de que el desarrollador asume el rol de Scrum master al momento de definir los sprints (actividades) y al hacer las revisiones.

El Scrum master es el facilitador del proyecto, que se encarga de definir los sprints (actividades) de acuerdo al tiempo disponible y las tareas que se deban cumplir. Para los sprints se necesita definir todas las tareas y colocarlas en un orden de acuerdo a las necesidades del proyecto. Los sprints para este proyecto deber incluir la recopilación de información sobre la librería para WebRTC, los lenguajes de programación que se utilizan y también la instalación de las herramientas para el desarrollo de la aplicación.

Los sprints tienen una duración de dos semanas y la revisión se hace al final del sprint, si alguna tarea queda pendiente se debe retomar en el siguiente sprint. Los sprints permiten también que en cada finalización se tenga un avance significativo y en el caso de la programación un entregable funcional al que se le puede aplicar pruebas de funcionalidad y corregir el código en la revisión, de manera que se disminuye el tiempo de corrección de errores al final del desarrollo.

CAPÍTULO 3

PROPUESTA

3.1 Esquemas

Para los esquemas del proyecto se utilizan diagramas UML de casos de uso, clases y flujo. En la figura 3.1 se puede ver el diagrama de casos de uso para la aplicación. La aplicación permite al actor A crear un espacio de trabajo para compartir con el actor B, el actor B recibe el número de espacio de trabajo y un enlace para conectarse. El actor B puede entonces decidir si se une al espacio de trabajo o crea su propio espacio.

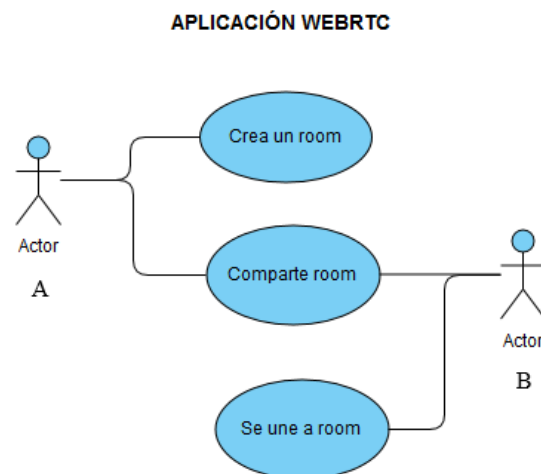


Figura 3. 1 Diagrama UML de casos de uso

Fuente: El autor

La dinámica se mantiene simple para favorecer la comunicación de los usuarios sin necesidad de registro en la aplicación, enviar una invitación a la persona, esperar que se confirme la invitación, establecer el espacio de trabajo mediante una llamada, esperar que la otra persona conteste y finalmente la comunicación. El diagrama de flujo de la operación tradicional de comunicación por aplicaciones de video llamada se muestra en la figura 3.2.

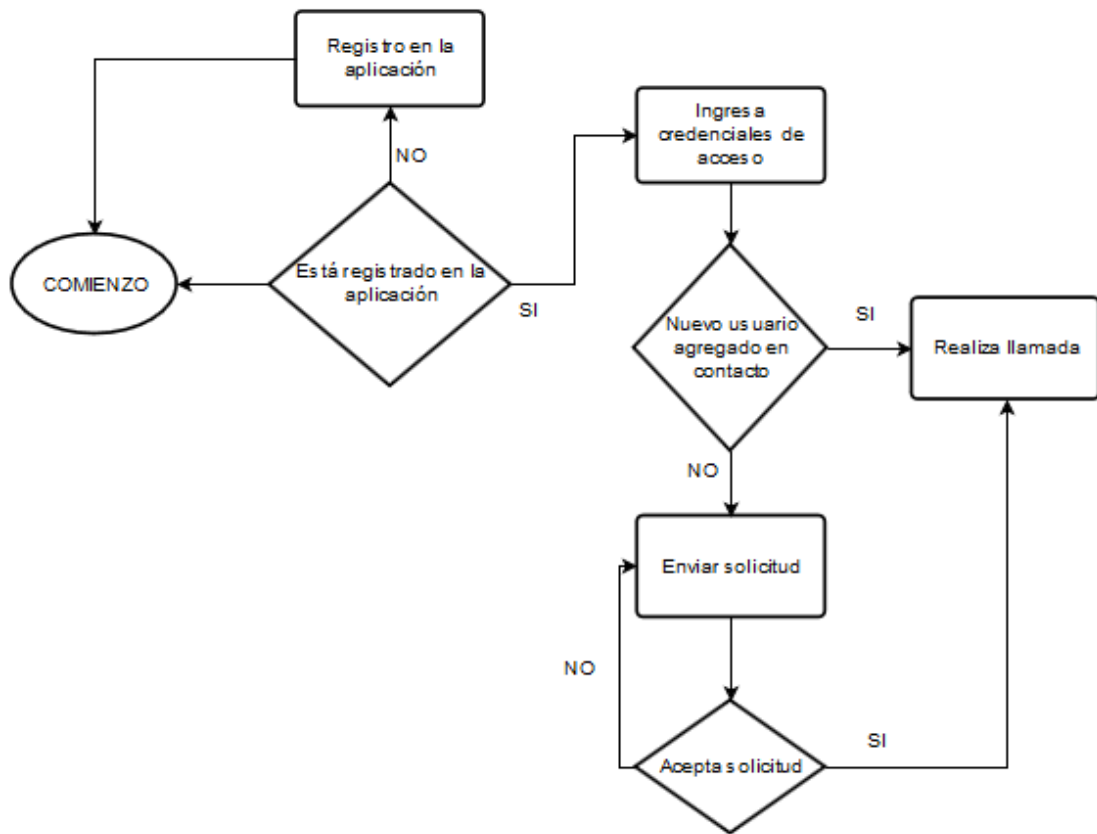


Figura 3. 2 Diagrama UML de flujo para aplicación tradicional

Fuente: El autor

En la figura 3.3 se tiene el diagrama de flujo para la propuesta del presente proyecto, donde el usuario genera un grupo de trabajo y envía el enlace con el que el otro usuario puede acceder al espacio de trabajo, sin necesidad de los pasos intermedios que se pudieron apreciar en la figura 3.2

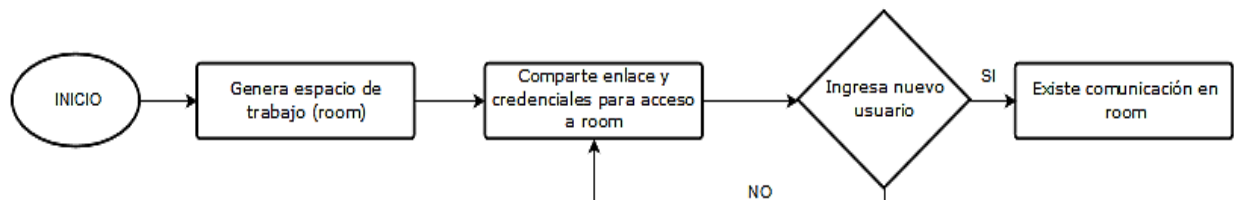


Figura 3. 3 Diagrama UML de flujo para aplicación propuesta

Fuente: El autor

Los diagramas de clase son esquemas que contienen variables y atributos. Los diagramas se presentarán durante la etapa de implementación cuando se determine cómo se utilizarán las herramientas en el desarrollo de la aplicación.

3.2 Módulos

La aplicación consta de dos módulos, el módulo de acceso donde se realiza el ingreso de credenciales para acceder al room y el módulo de room que a su vez define cuatro módulos internos que corresponden a las funcionalidades que los usuarios pueden aplicar cuando ingresan al espacio de trabajo: videoconferencia, chat, envío de archivos y compartición de pantalla. En la figura 3.4 se puede un diagrama de los módulos que comprenden la aplicación.

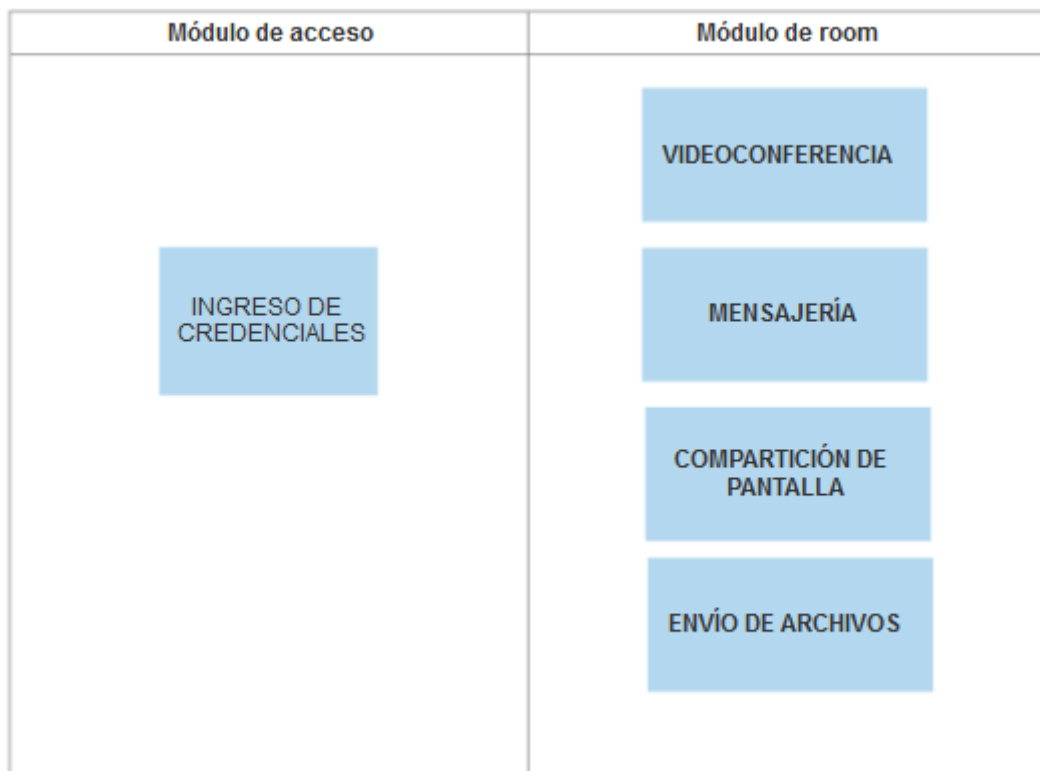


Figura 3. 4 Módulos de la aplicación

Fuente: El autor

3.2.1 Módulo de acceso

El módulo de acceso contiene los elementos que permiten la creación y el acceso a un room para lo cual implementa el submódulo de ingreso de credenciales que funciona para dos acciones: la creación de un room y el acceso a un room ya creado.

- Para la creación del room se ingresa un número para identificar al room, un nombre para el usuario y una contraseña asociada al room.
- Para el acceso a un room se requiere conocer el identificador de room y la contraseña, el nombre de usuario es configurable.

Si el ingreso de credenciales es correcto, se establece una conexión y se crea el room. La configuración de usuarios es maestro-esclavo, donde el creador del room tiene el rol del moderador.

3.2.2 Módulo de room

El módulo de room permite la comunicación entre los usuarios que acceden a la aplicación. Las funcionalidades se dividen en cuatro submódulos.

- En el módulo de videoconferencia maneja la transmisión de audio y video al usar la cámara y micrófono incorporado por el dispositivo al que se va a acceder a la aplicación.
- En el módulo de mensajería implementa el envío de mensajes instantáneos, se identifica el emisor con el nombre de usuario de cada uno de los participantes.
- El módulo de compartición de pantalla permite que el moderador y los usuarios comparta la pantalla de su dispositivo.
- El módulo de envío de archivos maneja la carga y envío de archivos a través del espacio destinado al envío de mensajes.

3.3 Aspectos técnicos del proyecto

- La aplicación es de tipo web, se puede acceder desde equipos y teléfonos inteligentes que tengan un explorador Chrome o Firefox con conexión a internet.
- La aplicación ofrece la creación de 10 rooms simultáneos con 5 usuarios en cada uno en la versión inicial.
- La aplicación requiere para su uso un ancho de banda mínimo de 2 Mbps para un funcionamiento óptimo.
- La aplicación permite el envío de archivos de hasta 1 GB, con las extensiones: pdf, docx, xlsx, jpg, png, mp3, mp4 e incluso archivos ejecutables (.exe)
- La aplicación necesita que el usuario acepte permisos para el uso de la cámara, micrófono y que se acepte las ventanas pop-up.

3.4 Sistema operativo, dependencias y herramientas de programación

Para el desarrollo se utiliza una máquina con sistema operativo Mac OS sobre la cual se instala el navegador Google Chrome que se utilizará para las pruebas durante el desarrollo. La versión instalada es la 76.0 como se muestra en la figura 3.5

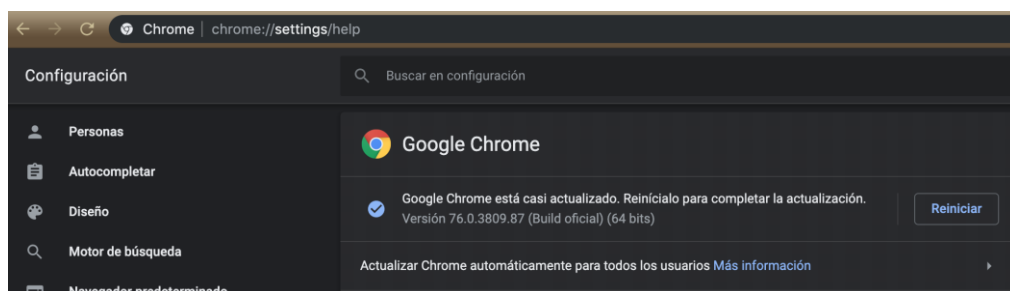


Figura 3.5 Versión de Google Chrome

Fuente: El autor

El navegador instalado tiene la ventaja de tener dentro de sus herramientas una consola que permite hacer análisis de elementos HTML, depuración del código JavaScript y también errores de sintaxis. En la figura 3.6 se puede ver una imagen de la consola de Google Chrome.

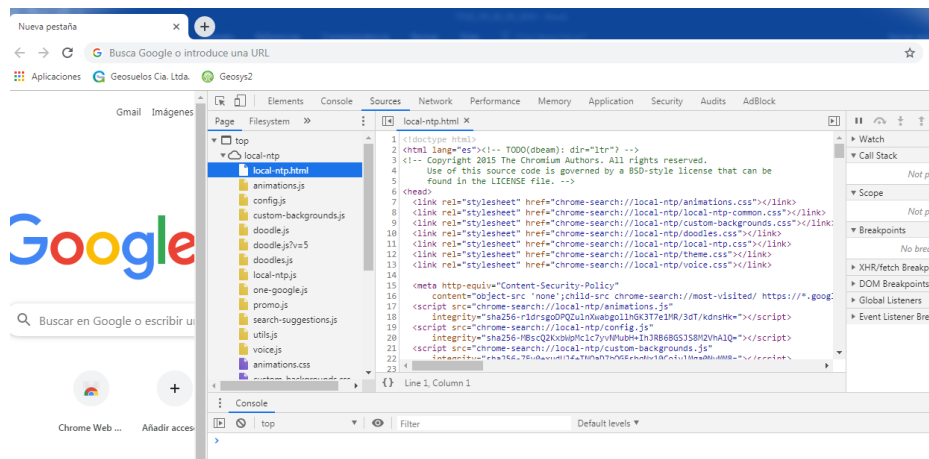


Figura 3. 6 Consola de Google Chrome

Fuente: El autor

Para la escritura del código se utiliza un IDE (Integrated Development Enviroment, Entorno de desarrollo integrado) que muestra errores de sintaxis, despliega ayuda al momento de integrar librerías y ordena el código con indentación y códigos de colores que hacen más fácil el seguimiento del código. Los IDE pueden ser instalados o usar uno online, para este proyecto se usa <https://repl.it/languages/nodejs> que es un IDE online que trabaja especialmente con Node y JavaScript, por lo que se ajusta mejor a los requerimientos del proyecto.

3.5 Análisis de costos de la aplicación

3.5.1 Horas-hombre

Para el análisis de costos se plantean los siguientes ítems del proyecto para el cálculo de las horas hombre. Las horas/desarrollador corresponden a las horas invertidas en el la programación y las horas/QA (Quality analysis, Análisis de calidad) a las horas de revisión de código, pruebas intermedias y pruebas finales de la aplicación. En el caso del proyecto, tanto las horas/desarrollador y las horas QA son responsabilidad del autor, por lo que se tiene un total de 209 horas para el proyecto como se puede ver en la tabla 3.1

Tabla 3. 1 Cálculo de horas/hombre

TAREAS	HORAS/DESAROLLO	HORAS/QA
Diseño de la página de inicio	10	2
Diseño de la página room	10	2
Implementación de la página de inicio	20	5
Implementación de la página room	50	10
Pruebas	30	30
Imprevistos de diseño y programación	30	10
TOTAL	150	59

Fuente: El autor

De acuerdo al análisis de costos provisto por (Arc, 2019) para una aplicación desarrollada en JavaScript que es la mayor parte del desarrollo se tiene un promedio de costo de hora de 40\$ para un programador de nivel medio en Latinoamérica. Un programador web para el diseño de las páginas está en el rango de 60\$ con las mismas consideraciones, ambos valores se pueden ver en la figura 3.7 y 3.8

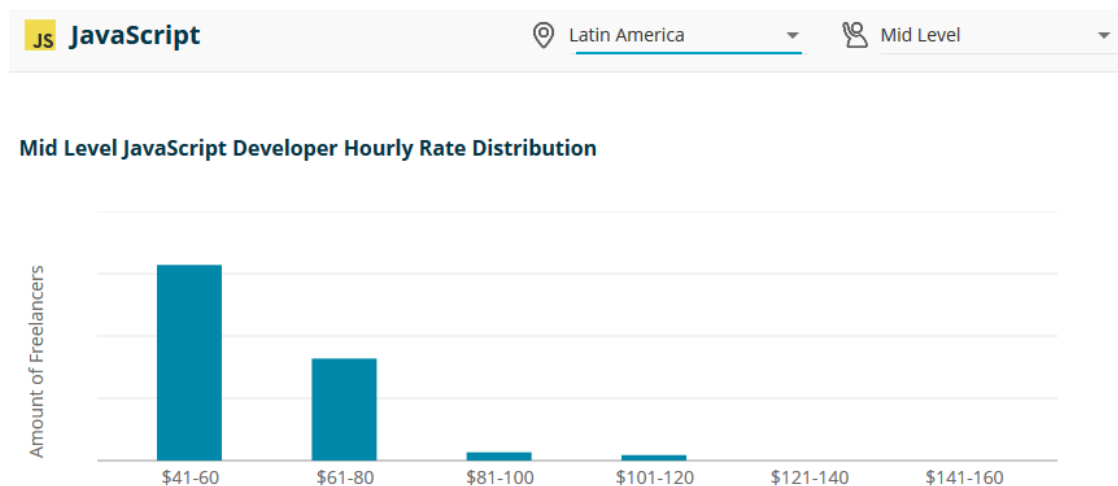


Figura 3. 7 Costo de horas programación promedio JavaScript

Fuente: (Arc, 2019)

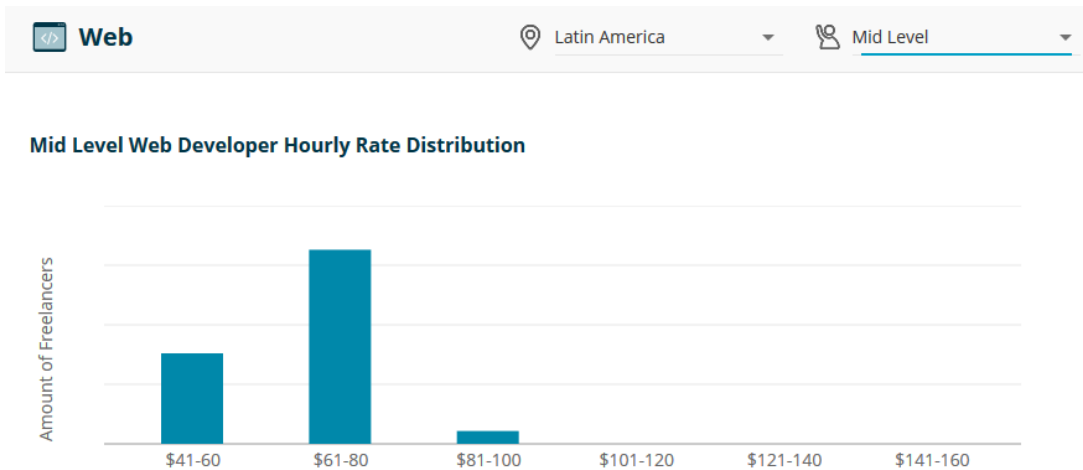


Figura 3. 8 Costo de horas programación promedio Web

Fuente: (Arc, 2019)

Para un análisis simple se asume el valor más bajo de 40\$ para todas las horas de programación y 20\$ para las horas de QA lo que da como resultado 7180\$ para programación como se puede ver en la tabla 3.2.

Tabla 3. 2 Cálculo de costo de horas de desarrollo

Horas desarrollador	150 horas	6000\$
Horas QA	59 horas	1180\$
Total	209 horas	7180\$

Fuente: El autor

3.5.2 Gastos pasivos e imprevistos

Para los gastos pasivos e imprevistos se considera primero el tiempo de desarrollo del proyecto, que incluye desde el planteamiento de requerimientos, diseño, programación e implementación con un estimado de 4 meses. Los programas que se utilizan se puede ver en la tabla 3.3 y se han seleccionado de código abierto para minimizar los gastos. Adicional a los programas, también se considera los recursos físicos que se utilizan los cuales se detallan en la tabla 3.4.

Tabla 3. 3 Análisis de costos de la aplicación

Parámetro	Uso	Costo
SO Mac	Sistema operativo	0 \$
Node y Node JS	Framework	0 \$
RTC Multiconnection	Librería JS	0\$
IDE online	Entrono de desarrollo	0\$
Heroku	Servidor de aplicación	0\$ en plan gratis

Fuente: El autor

Tabla 3. 4 Recursos físicos

Parámetro	Uso	Consideración	Costo
MacBook Pro	Desarrollo	Depreciación de 16.5% en 4 meses	247.5\$
Conexión a internet	Desarrollo	Se asume un aproximado de 200 horas al mes	30\$
Servicios	Desarrollo y pruebas	Equipos adicionales para pruebas, servicios.	100\$
TOTAL			377.5

Fuente: El autor

Para el calcular el costo del proyecto se debe ajustar las horas de desarrollador, pues el autor del proyecto no puede avaluar la hora de trabajo al mismo nivel que un desarrollador semi-senior, por lo que establece el valor de 20\$ para horas de desarrollo y QA, así mismo se cotiza un adicional de 70 horas para ajustes de programación en caso de requerir un cambio. En la tabla 3.5 se tiene el costo del proyecto con el ajuste de precio.

Tabla 3. 5 Costo final del proyecto

Horas desarrollador	209 horas	4180\$
Horas adicionales	70 horas	1740\$

Gastos pasivos e imprevistos		377.5\$
Total		6297.5\$

Fuente: El autor

Para evaluar la viabilidad y determinar si el costo calculado está dentro de un rango de costo adecuado, se contrasta este costo con el cálculo para un proyecto de similares características provisto por Arc, donde en la estimación más baja se calcula un costo de 13000 dólares como se puede ver en la figura 3.9. Con esto se puede evidenciar que valor propuesto para el proyecto está dentro de un rango aceptable y tiene viabilidad para ser implementado.

		LOW	HIGH
Number of pages	1 - 10	\$ 1,000	\$ 2,000
Style of design	Moderately stylized	\$ 3,000	\$ 5,000
Copywriting # of pages	None	\$ 0	\$ 0
SEO w/ Placement Guarantee	30 keywords	\$ 2,000	\$ 4,000
Responsive Design	Yes	\$ 3,000	\$ 3,000
Database Integration	None	\$ 0	\$ 0
e-Commerce Functionality	Basic	\$ 2,000	\$ 4,000
CMS	Standard	\$ 2,000	\$ 4,000
Total Estimated Cost		\$ 13,000	\$ 22,000

Figura 3. 9 Costo de un proyecto web con similares características

Fuente: (Arc, 2019)

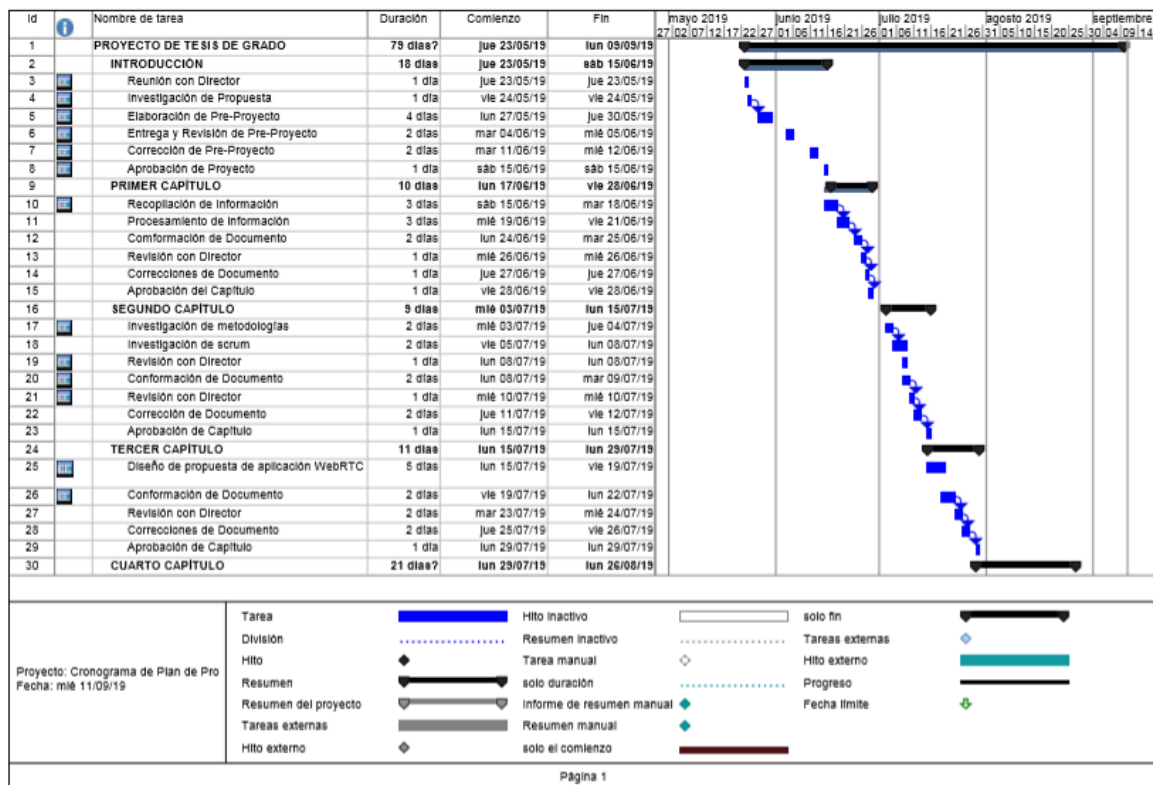
3.6 Análisis de tiempo

Para el análisis de tiempo se considera las actividades del proyecto de tesis de grado. La primera etapa consiste en la introducción que tiene una duración de 18 días en la que se realiza reunión con el director, investigación de la propuesta, elaboración del pre-proyecto, entrega, revisión, corrección y aprobación de la propuesta.

La siguiente etapa consiste en el desarrollo del capítulo uno con una duración de 10 días en los cuales se realiza la recopilación y procesamiento de la información, la conformación del documento, revisiones, correcciones y aprobación del capítulo.

A continuación, se tiene el segundo capítulo con una duración de 9 días donde se realiza la investigación de metodologías relacionadas al proyecto. En la cuarta etapa se desarrolla el capítulo tres donde se realiza el diseño de la propuesta que incluye diagramas, módulos, aspectos técnicos, análisis de costos, análisis de tiempos y ventajas en un período de tiempo de 11 días.

En la quinta etapa se realiza el diseño e implementación del proyecto usando los sprints definidos al inicio del capítulo, esta etapa tiene una duración de 20 días. En la etapa final se utiliza 11 días para las conclusiones, recomendaciones, bibliografía, revisión general del documento, correcciones y aprobación. En la figura 3.10 a) y b) se tiene el diagrama de Gantt para el proyecto.



a)

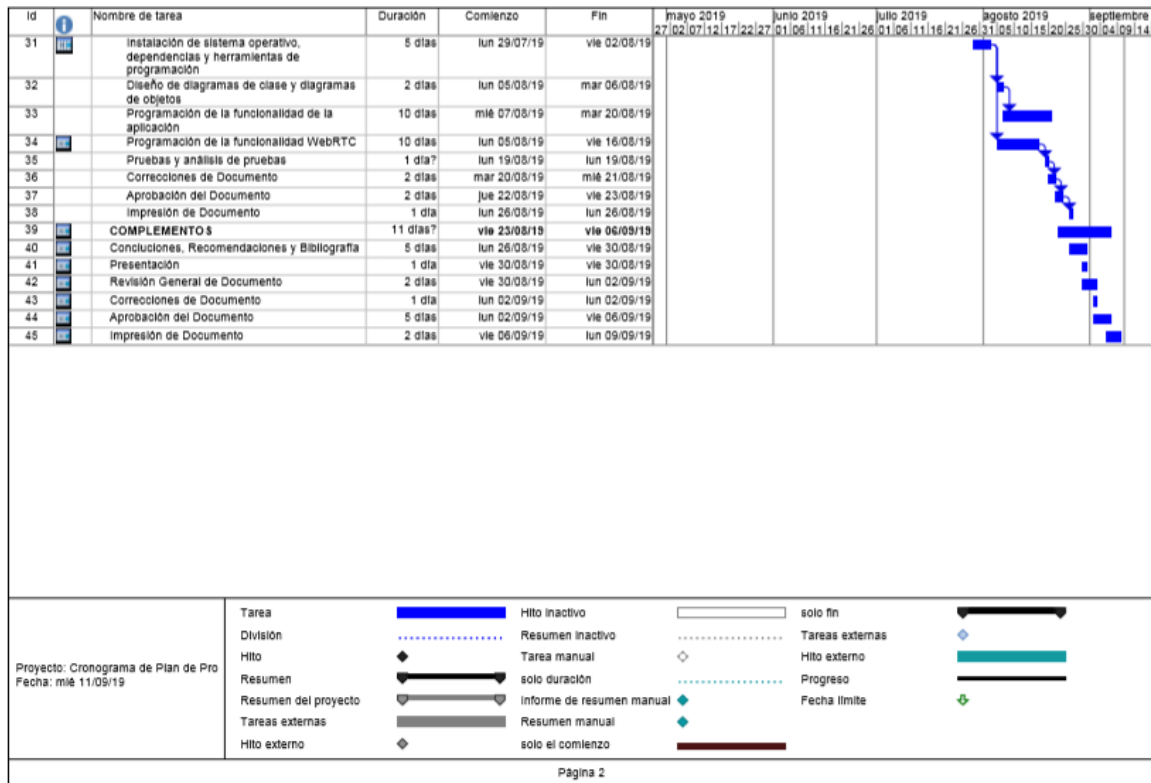


Figura 3. 10 b) Diagrama de Gantt para el proyecto

Fuente: El autor

3.7 Ventajas del proyecto

- La aplicación evita los procesos tradicionales que requieren el ingreso de credenciales y suscripciones para su uso, en cambio, provee solamente de un enlace en el que se debe ingresar los datos correspondientes al room proporcionados por el creador.
- La aplicación ofrece una alternativa a las aplicaciones que requieren instalación de un programa dedicado, para esta propuesta solamente se requiere un explorador Chrome o Firefox compatible con el protocolo.
- La aplicación utiliza para su desarrollo e implementación software libre, por lo que los costos de desarrollo del programa son mínimos. El despliegue de la aplicación se realiza sobre un servidor gratuito con limitaciones, sin embargo, se puede adquirir en este mismo

servidor un plan de pago que va desde los 7\$ a los 500\$ si se desea implementar la aplicación en producción.

- La aplicación no depende de los recursos de la computadora o dispositivo inteligente sino solamente de la calidad de la conexión a Internet.

CAPÍTULO 4

IMPLEMENTACIÓN

4.1 Diseño

4.1.1 Sprints

El desarrollo e implementación del proyecto utiliza sprints, un sprint es un periodo de tiempo determinado que puede ser de una a cuatro semanas para el desarrollo de una característica del producto de software. Para este proyecto se tiene una duración entre una y tres semanas al depender de la complejidad de la tarea y un día de revisión del producto al final de cada sprint, el seguimiento es parte de la metodología SCRUM y permite que el proyecto se desarrolle de forma ordenada y con seguimiento del avance. Los sprints para el proyecto se planifican de acuerdo a la tabla 4.2.

Tabla 4. 1 Sprints para el desarrollo del proyecto

SPRINT	ACTIVIDAD
	Diseño de diagramas de clase y diagramas de distribución de elementos
2	Selección de sistema operativo, lenguajes de programación y herramientas.
3	Instalación y configuración de sistema operativo, dependencias y herramientas de programación
5	Programación de la funcionalidad de la aplicación
8	Programación de la funcionalidad WebRTC
12	Pruebas de funcionalidad

Fuente: Del autor

4.1.2 Diseño de diagramas de clase

Los diagramas de clase son la representación de los objetos que conforman el programa y como se relacionan entre ellos, para lo cual se debe definir la funcionalidad de la aplicación de acuerdo al alcance donde se indica que tras definir los requisitos mínimos para navegadores Mozilla y Chrome se debe diseñar los módulos de la aplicación: videoconferencia, chat, envío de archivos y compartir pantalla, para el diseño de la aplicación se utilizará diagramas de clases y diagramas UML.

El alcance no contempla la creación de una base de datos o manejo de usuarios registrados, como parte de la innovación del programa, para el acceso a la aplicación se requiere el id del room y la contraseña que provee el creador del room, el nombre de usuario que es solo para la identificación puede ser variable.

Para el proyecto se utilizan dos páginas: `index.html` y `room.html` que usan métodos de la librería `RTCMulticonnection` como se puede ver en la figura 4.1. En `index.html` se tienen tres funciones, `joinARoom()` que maneja la función de unirse a un room ya creado, `openRoom()` que permite crear un nuevo room, y `looper` que permite hacer la cuenta de los room que están abiertos y en línea.

En `room.html` se tiene las funciones `appendChatMessage` para el manejo de mensajería instantánea, `getFileHTML()` para el manejo de carga y descarga de archivos, `getFullName()` y `updateLabel()` para actualizar y presentar los nuevos usuarios, `addStreamStopListener()` y `replaceTrack()` para la presentación del stream de video. Las funciones y métodos en `RTCMulticonnection` fueron definidas con anterioridad y aquí se hace referencia a las usadas por la aplicación.

4.1.3 Diseño de diagrama de distribución de objetos

Definidas las funcionalidades en el diagrama de clase, se presenta un esquema de la distribución de objetos en cada una de las páginas. Estos objetos corresponden a los controles (cuadros de texto, botones, etc) y a los espacios destinados para cada una de las funcionalidades de la aplicación.

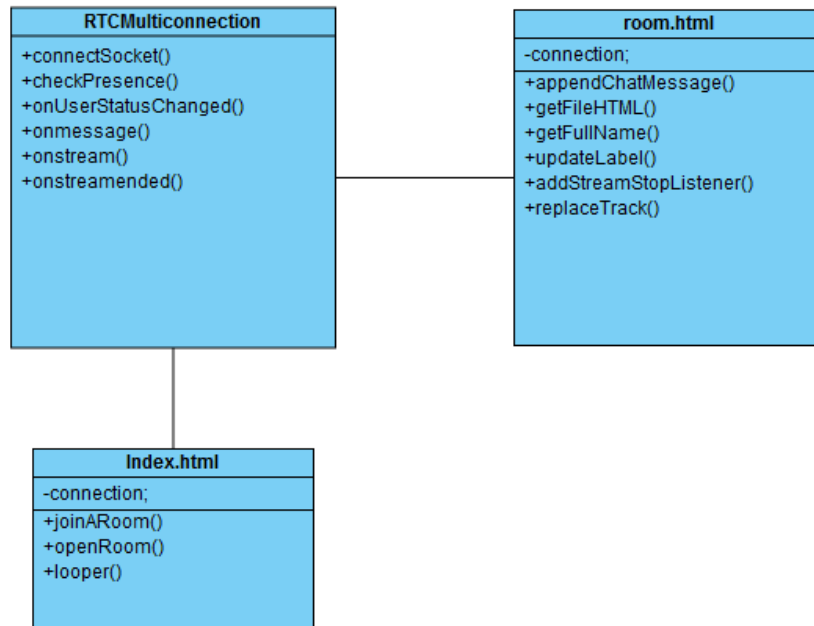


Figura 4. 1 Diagrama de clase de la aplicación

Fuente: El autor

En la figura 4.2 se puede ver el esquema de objetos para index.html, este tiene tres cuadros de texto para el ingreso de la información de rooms. Se tiene también dos botones que permiten crear o acceder a un room creado.

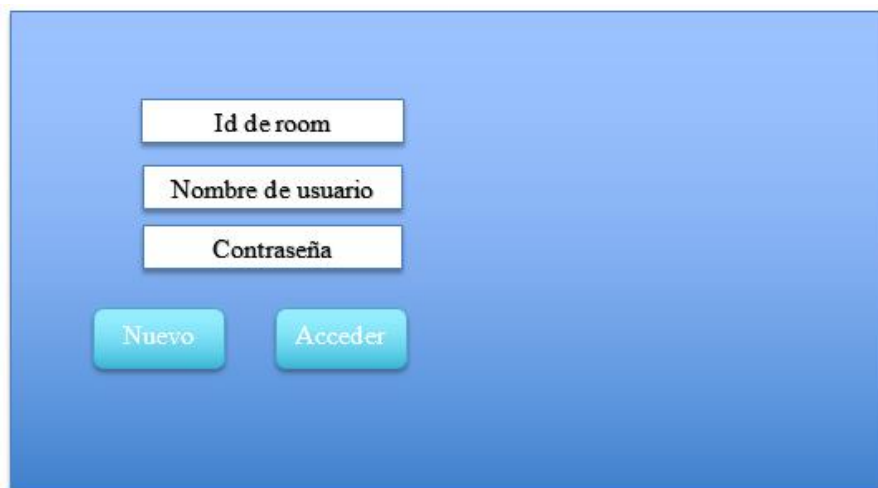


Figura 4. 2 Esquema de objetos index.html

Fuente: El autor

En la figura 4.3 se puede ver el esquema para room.html donde se identifican tres secciones. La sección de videoconferencia contiene elementos de video que muestran la imagen de la cámara de cada uno de los participantes. La sección de mensajería y envío de archivos es un

contenedor que debe adaptarse a la llegada de nuevos mensajes y también que permite cargar y descargar los archivos compartidos en el grupo. La sección de compartición de pantalla contiene un elemento de video para presentar la pantalla.



Figura 4. 3. Esquema de objetos para room.html

Fuente: El autor

Para la sección de compartición de pantalla es evidente su contenido, sin embargo, la sección de videoconferencia y mensajería pueden detallarse más. En la figura 4.4 se tiene un esquema de la sección de videoconferencia, se puede ver que el video principal es más grande que los videos secundarios pues el moderador/creador del room tiene prioridad en la imagen.

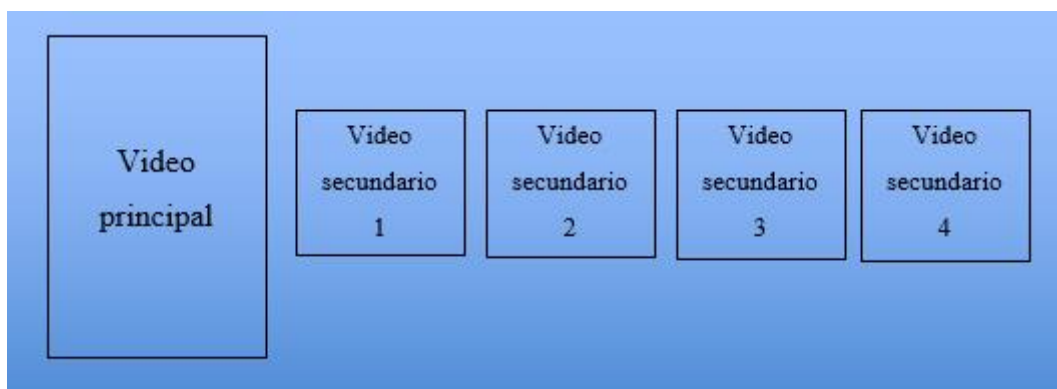


Figura 4. 4 Esquema de sección de videoconferencia

Fuente: El autor

La sección de mensajería y envío de archivos se detalla la figura 4.5. Para diferenciar los mensajes se usa el nombre de usuario como prefijo y el contenedor debe mostrar en cierto momento una barra de desplazamiento para poder navegar en los mensajes enviados. Además del cuadro de texto para escribir el mensaje y el botón de envío, se tiene un ícono que al ser presionado despliega la ventana de selección de archivos en la computadora o el dispositivo móvil y permite seleccionar el archivo el mismo que ocupará un espacio de mensaje con un enlace para ser descargado.

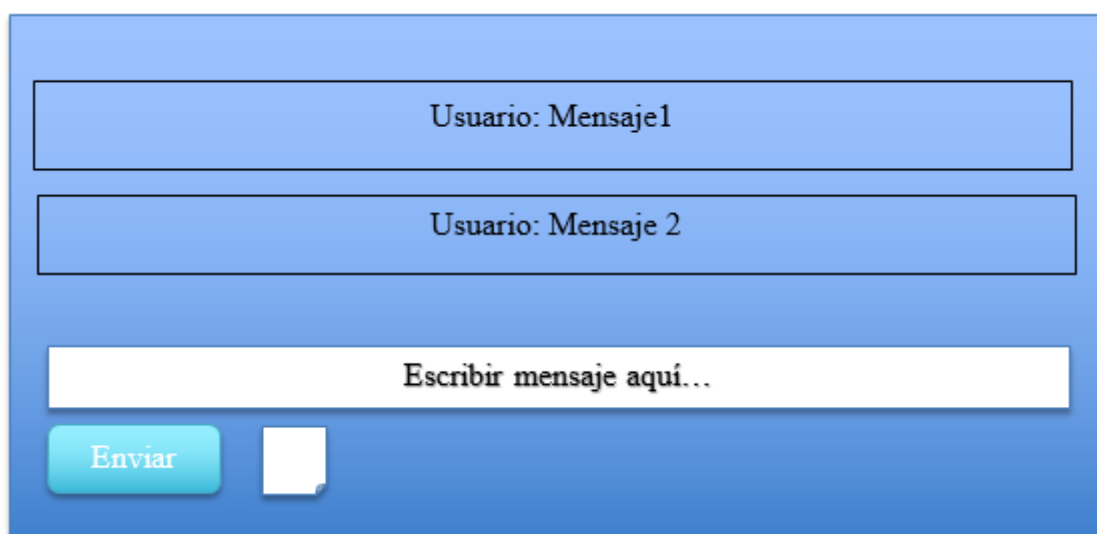


Figura 4. 5 Esquema de sección de envío de mensajes

Fuente: El autor

4.2 Implementación

En primera instancia se detalla los programas seleccionados y los procesos de instalación y configuración. La base del proyecto es la librería RTCMulticonnection, en los repositorios de información de Web RTC se indica que todas las funcionalidades que se proponen se pueden cumplir con el protocolo. (WebRTC Experiment, 2019) En estos repositorios también se disponen enlaces a tutoriales y librerías de código abierto disponibles en GitHub y otros repositorios de compartición de información.

4.2.1 Librería RTCMulticonnection

En el repositorio GitHub se encuentra la librería RTCMulticonnection como parte de un proyecto del MIT de código abierto y con soporte continuo. Además de instrucciones para utilizar la librería también se muestra un esquema de las carpetas y archivos que se requiere para un proyecto con RTCMulticonnection. Adicionalmente, se tiene carpetas con códigos demostrativos para las diferentes funcionalidades que provee la librería, archivos de configuración de servidores JS y servidores IO para establecimiento de conexión y archivos de configuración de despliegue. El esquema de carpetas y archivos del proyecto, como se tiene en la tabla 4.3, se basa en esquema de la librería RTCMulticonnection.

Tabla 4. 2 Archivos y carpetas de un proyecto base

ARCHIVO	FUNCIONALIDAD
Server.js	Manejo de servidores RTCMulticonnection y Server IO. Se tiene un ejemplo en la figura 4.1
Config.json	Detalles de configuración del proyecto para control de versiones, despliegue local, manejo de errores. Se tiene un ejemplo en la figura 4.2
Index.html	Es el archivo principal del proyecto, es la página a la que se llama para iniciar la aplicación y desde la cual se controla las llamadas secundarias.
CARPETAS	
Css	Contiene, si fuera necesario, las plantillas de estilos para la aplicación
Node.js	Contiene los scripts de la librería RTCMulticonnection que se van a llamar desde las páginas de la aplicación

Fuente: Del autor

En la figura 4.6 se puede ver un esquema del documento Server.js provisto en la documentación de la librería RTCMulticonnection. En este archivo se realiza la configuración de los servidores que se utilizan para la aplicación, en el caso de una aplicación

con la librería `RTCMulticonnection` que implementa el servidor RTC, el servidor IO (Input Output, entrada-salida) para configuración de las conexiones punto a punto y el servidor local de node JS.

```
var server = require('http'),
    url = require('url'),
    path = require('path'),
    fs = require('fs');
function serverHandler(request, response) { }

var config = {
};

var RTCMultiConnectionServer = require('rtcmulticonnection-server');
var ioServer = require('socket.io');

var app = server.createServer(serverHandler);
RTCMultiConnectionServer.beforeHttpListen(app, config);
app = app.listen(process.env.PORT || 9001, process.env.IP || "0.0.0.0", function() {
    RTCMultiConnectionServer.afterHttpListen(app, config);
});
```

Figura 4. 6 Esquema de Server.js

Fuente: El autor

El esquema del archivo `package.json` se muestra en la figura 4.7 Este archivo tomado de la librería `RTCMulticonnection` contiene información sobre el proyecto: el nombre, la versión, comentarios, nombre y contacto del desarrollador y el repositorio que maneja. En la sección de scripts define el servidor con el que se ejecuta y los prerequisites que deben instalarse para ejecutar y desplegar la aplicación en el servidor donde será alojada.

La librería `RTCMulticonnection` comprende una gran cantidad de funcionalidades relacionadas a WebRTC, para este proyecto se implementará videoconferencia, compartición de pantalla, envío de mensajes instantáneos y compartición de archivos. Para instalación de la librería se requiere instalar primero Node JS y NPM.

```
{
  "name": "aplicacion webrtc",
  "preferGlobal": false,
  "version": "1.0",
  "author": {
    "name": "creador",
    "email": "creador@gmail.com",
    "url": "https://creador.com/"
  },
  "description": "ejemplo webrtc",
  "repository": {
    "type": "",
    "url": ""
  },
  "scripts": {
    "start": "node server.js",
    "contrib-prerequisites": "npm install grunt@0.4.5 && npm install grunt-bump@0.7.0 &&
    npm install grunt-cli@0.1.13 && npm install grunt-contrib-clean@0.6.0 && npm install
    grunt-contrib-concat@0.5.1 && npm install grunt-contrib-copy@0.8.2 && npm install
    grunt-contrib-uglify@0.11.0 && npm install grunt-contrib-watch@1.1.0 && npm install
    grunt-jsbeautifier@0.2.10 && npm install grunt-replace@0.11.0 && npm install load-
    grunt-tasks@3.4.0",
    "build": "npm install"
  },
  "engines": {
    "node": ">=8.0.0 <11.0.0"
  }
}
```

Figura 4. 7 Esquema de package.json

Fuente: El autor

El proceso de instalación en Windows comprende la descarga de la versión ejecutable más reciente de Node JS de la página oficial <https://nodejs.org/en/>. Tras la ejecución del programa se verifica mediante el terminal mediante el comando `node -v`. En la figura 4.8 se puede ver la ventana de instalación de node JS para Windows.

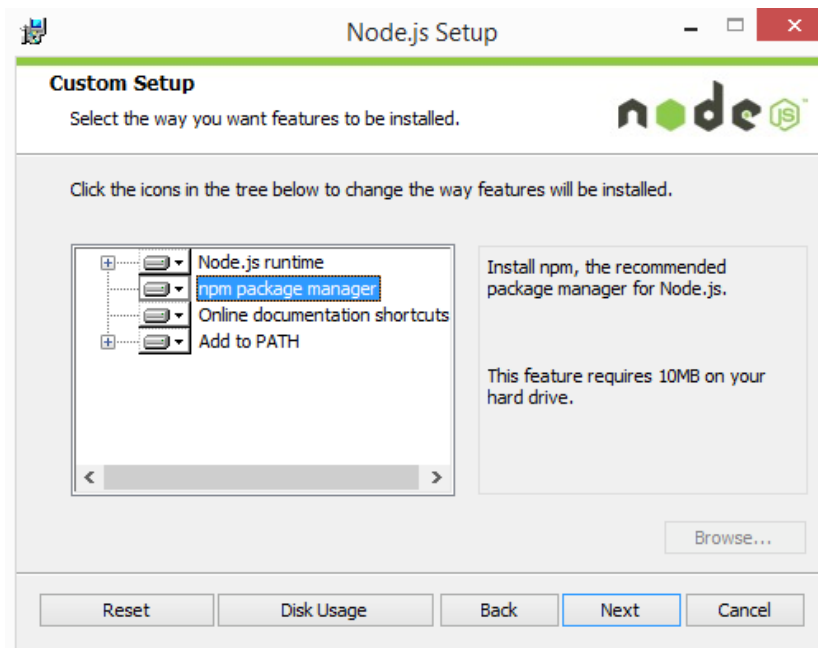


Figura 4. 8 Instalador de node JS para Windows

Fuente: El autor

El proceso de instalación en Linux y MacOS se realiza mediante la terminal con el comando `sudo apt-get install nodejs`, para verificar la instalación correcta se utiliza el mismo comando que en Windows. NPM es parte del paquete Node JS y se instala en conjunto, para verificar la instalación y versión se utiliza el comando `npm -v`.

Tras la instalación de Node JS y NPM se puede descargar la librería RTCMulticonnection, al usar los comandos mostrados en la figura 4.9

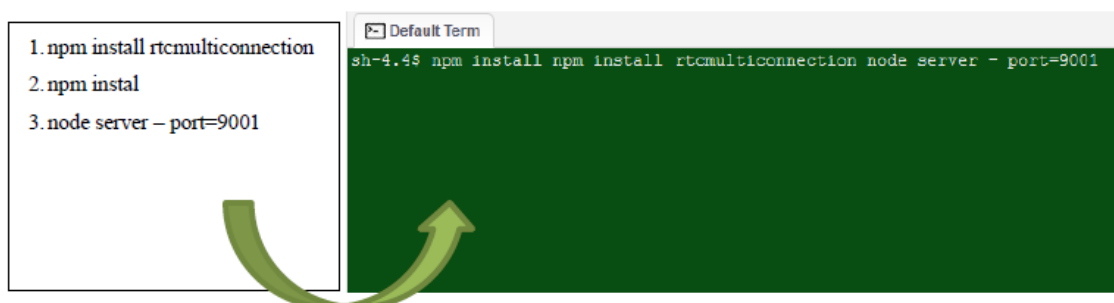


Figura 4. 9 Comandos para instalar RTCMulticonnection en Linux

Fuente: El autor

El primer comando descarga la librería a la máquina local, para lo que se recomienda primero definir una carpeta sobre la que se realizará el proyecto, el segundo comando permite compilar todas las dependencias del proyecto y los cambios realizados, el tercer comando inicializa el servidor JS local y define el puerto sobre el que se despliega.

La librería RTCMulticonnection es un archivo de JavaScript que contiene múltiples variables y métodos asociados al protocolo WebRTC. A continuación, se detalla las funciones principales definidas por la clase en la tabla 4.3

Tabla 4. 3 Funciones de RTCMulticonnection

FUNCION	
SocketConnection	Recibe como parámetro la conexión y una llamada de retorno. Discrimina el tipo de conexión que existe si es audio, video o datos.
MultiPeers	Recibe como parámetro de ingreso la conexión y realiza el manejo para una sesión de acceso múltiple.
getBrowserInfo	Identifica y recupera la información de la versión y configuración del explorador de red utilizado.
isIE10OrLater	Verifica que el explorador usado sea IE 10 o superiores
detectPrivateMode	Detecta modo privado en exploradores
isMobile	Detecta si el explorador usado viene desde un dispositivo celular.
detectDesktopOS	Detecta el sistema operativo en el que se ejecuta el navegador que accede a la aplicación.
DetectLocalIPAddress	Detecta la IP local del equipo.
getIPs	Obtiene las direcciones IP de todos los pares que acceden
PeerInitiator	Permite que se inicie conexión entre pares que acceden al Socket

connectSocket	Crea el socket y habilita las conexiones de audio, video o datos entre los extremos.
---------------	--

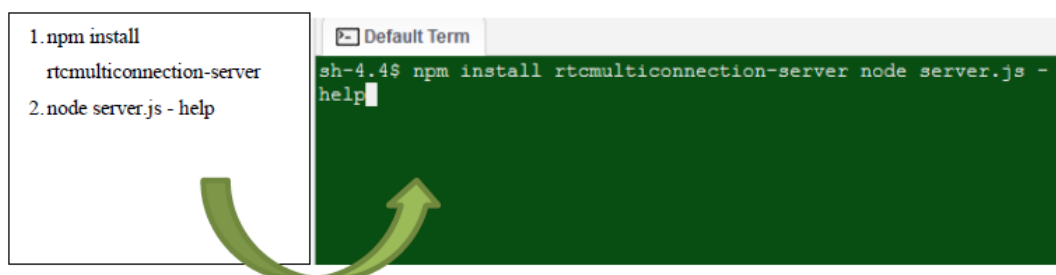
Fuente: Del autor

Adicionalmente se define en concepto de ROOM (sala virtual) en RTCMulticonexction como un espacio de comunicación donde acceden dos o más usuarios. Se utiliza una nomenclatura alfanumérica o numérica para definir al room, el creador del room puede luego compartir la identificación para que otros usuarios accedan a su espacio

4.2.2 Server IO (Server Input Output, Servidor de entrada y salida)

RTCMulticonnection utiliza para el establecimiento de conexión un servidor IO el cual permite virtualizar un servidor físico con múltiples entradas y salidas por cables y reemplazarlo con entradas y salidas virtuales. Un socket es el establecimiento de conexión entre el servidor IO y el cliente remoto.

El servidor IO permite el despliegue de aplicaciones que requieran accesos simultáneos y se encarga de la asignación y liberación de los sockets. La librería utiliza su propio servidor RTCMulticonnection-Server, para el enlace al servidor se usa el comando requiere ('socket.io'). Se realiza la instalación con los comandos mostrados en la figura 4.10.



The image shows a terminal window with a dark green background. On the left, there is a white box containing a list of commands: 1. npm install rtcmulticonnection-server and 2. node server.js - help. A green arrow points from this box to the terminal window. The terminal window shows the command 'sh-4.4\$ npm install rtcmulticonnection-server node server.js - help' being entered.

Figura 4. 10 Comandos para instalar el servidor RTCMulticonnection

Fuente: El autor

El lenguaje HTML es la base de las aplicaciones web, por lo que este será al usar en el proyecto para el esquema visual de la aplicación, sin embargo, el diseño y presentación se

complementan con JS y CSS para que el producto tenga una presentación amigable al usuario y de calidad.

JavaScript es el núcleo del proyecto, con este lenguaje se realizan todas las interacciones de la aplicación con el usuario y también se utiliza la librería de Node JS para el desarrollo y despliegue del programa.

La librería RTCMulticonnection es un elemento fundamental del proyecto, pues permite la simplificación del uso del protocolo WebRTC al implementar clases y métodos de las funcionalidades: videoconferencia, mensajería instantánea, compartición de archivos y pantalla.

4.2.3 Requisitos mínimos de navegadores Mozilla y Chrome

La información del protocolo WebRTC, los requerimientos para su operación y el soporte al desarrollador se encuentran en los repositorios de WebRTC y en la página oficial de WebRTC. Los requerimientos mínimos para el funcionamiento del protocolo WebRTC se listan a continuación. (WebRTC Organization, 2019)

Exploradores web para computadoras

Google Chrome 28+

Mozilla Firefox 22+

Safari 11+

Microsoft Edge 12+

Vivaldi 1.9+

Opera 18+

Exploradores web para Android

Google Chrome 28+ (habilitado por defecto desde la versión +29)

Mozilla Firefox 24+

Opera Mobile 12+

Chrome OS

Firefox OS

Exploradores web para iOS 11

MobileSafari / WebKit

Tizen 3.0

Con esta información se define los requisitos mínimos de versión para Mozilla y Chrome con soporte que incluye los dispositivos móviles Android y computadoras de escritorio. La aplicación también puede ser ejecutada desde el navegador Safari en iOS, aunque no se haya contemplado para este proyecto.

4.3 Implementación y programación de las páginas

4.3.1 La página index.html

En la figura 4.11 se observa el esquema base html para index.html que es la pantalla de inicio para la creación y acceso a rooms. Con base a los diagramas de objetos definidos en el diseño, se inserta los elementos HTML de etiquetas, cajas de texto y botones y las posiciones de los objetos se realizan con las etiquetas <div> y las propiedades de alto, ancho, color, posición y márgenes. Cada elemento es posicionado de acuerdo al diseño mediante las propiedades y las hojas de estilo.

La página de index.html requiere de la referencia para las hojas de estilo CSS y scripts JS, lo que se realiza con la etiqueta link para las hojas de estilo CSS y la etiqueta script para los scripts de java, la lista de referencias completas se puede ver en la figura 4.12. El código completo de index.html se encuentra en el Anexo 1, en la figura 4.13 se tiene una vista de la página en el explorador.


```

<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
<body>
<header style="margin-bottom: 20px;">
  <div style="margin-bottom: 40px;">
    
    <span class="icono-text">Aplicación WebRTC</span>
  </div>
  <div style="margin-top: 10px; margin-left: 20px;">
    <p>
      <label for="txt-roomid" style="width:200px;" >Número de room:</label>
      <input type="text" id="txt-roomid">
    </p>
    <p>
      <label for="txt-user-name" style="width:200px;">Nombre:</label>
      <input type="text" id="txt-user-name">
    </p>
    <p>
      <label for="txt-room-password" style="width:200px;">Contraseña:</label>
      <input type="password" id="txt-room-password-hidden"> <br><br>
      <input type="checkbox" onclick="mostrarContrasena()" style="margin-
right:5px;">Mostrar contraseña
    </p>
  </div>
  <div style="float: left; margin-top: 15px; margin-left: 20px;">
    <button id="nuevoRoom" class="btnVerde">Nuevo room</button>
    <button id="unirseRoom" class="btnVerde">Unirse a room</button>
    <span class="top-span">Rooms activos: <span id="active-rooms">0</span></span>
  </div>

```

Figura 4. 11 Esquema html de la página principal index.html

Fuente: El autor

```
<link rel="stylesheet" href="/css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="/css/estilo.css">
<script src="/js/jquery-3.3.1.slim.min.js"></script>
<script src="/js/popper.min.js"></script>
<script src="/js/bootstrap.min.js"></script>
<script src="/js/RTCMultiConnection.min.js"></script>
<script src="/socket.io/socket.io.js"></script>
```

Figura 4.12 Lista de referencias en index.html

Fuente: El autor



Número de room:

Nombre:

Contraseña:

Mostrar contraseña

Rooms activos: 0

Figura 4.13 Vista de la página index.html

Fuente: El autor

Los pasos para el acceso a la aplicación en index.html son:

- Ingresar el número de room, un nombre y una contraseña para crear un room
- Si desea unirse a un room se debe solicitar al creador del room el número ya la contraseña, el usuario es solo identificativo.

- c) Se puede marcar el check si se desea ver la contraseña para verificar que esté bien escrita.

4.3.2 La página room.html

La página room.html contiene los elementos para la videoconferencia, mensajería y compartición de pantalla, el código completo de room.html se encuentra en el anexo 2. Para que se pueda hacer uso de las funciones se requiere al menos dos personas dentro del room. Al momento de crear el room, si se accede desde un explorador nuevo se solicita permiso para el uso de cámara y teléfono en la aplicación y luego se despliega un ícono en la parte superior que indica que tanto la cámara como el audio están al ser usados, el video de la cámara del creador del room se encuentra en la parte superior izquierda. En la figura 4.14 se puede observar la vista de la aplicación al unirse dos o más usuarios, los videos de las cámaras de los usuarios se ubican junto al video del creador del room y además se habilitan los cuadros para envío de mensajes, compartición de archivos y compartición de escritorio.

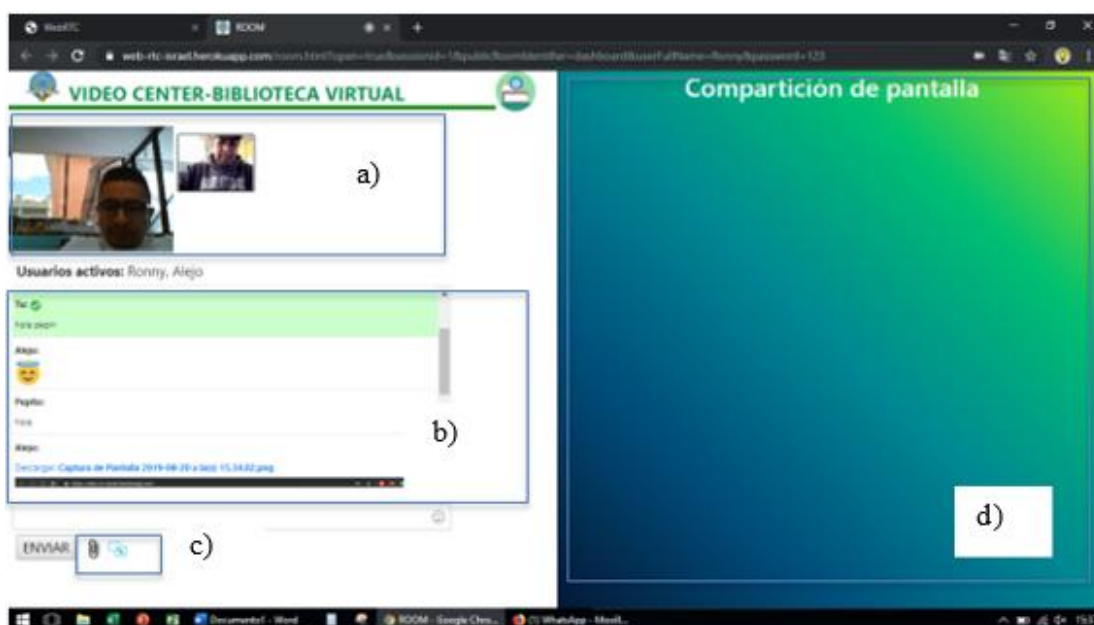


Figura 4.14 Vista de room.html con dos usuarios. a) Zona de videoconferencia b) Zona de mensajería c) Zona de botón de envío de archivos y compartición de pantalla d) Zona de pantalla compartida

Para la aplicación, se ha limitado a cinco usuarios máximos al usar el método `maxParticipantAllowed`. Los usuarios pueden incrementarse hasta 256 para los exploradores Google Chrome y Mozilla Firefox. En la figura 4.15 se puede observar cuatro usuarios simultáneos.

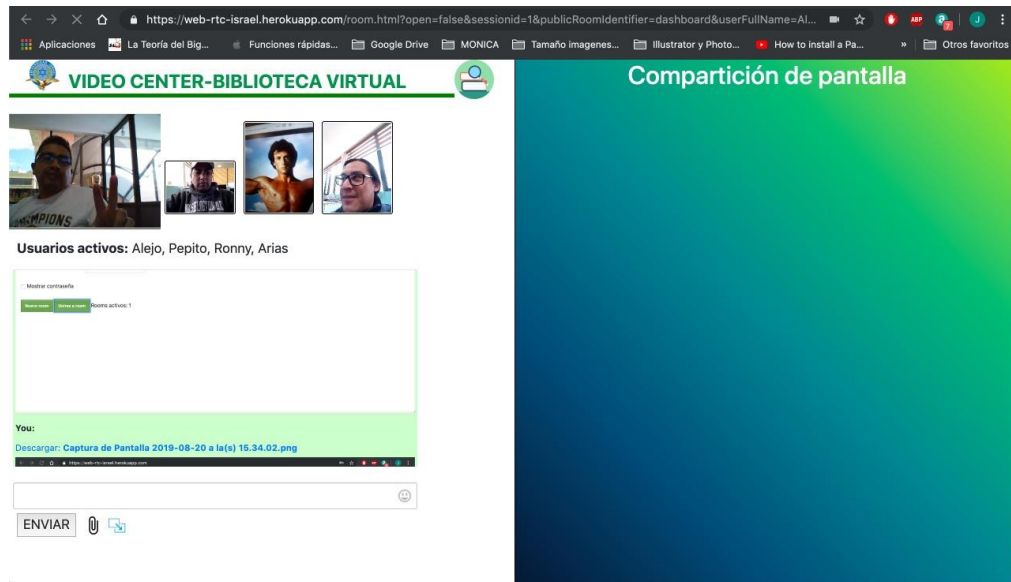


Figura 4.15 Vista de room.html con varios usuarios

Fuente: El autor

4.3 Programación de la funcionalidad WebRTC

WebRTC está incorporado a la aplicación a través de la librería `RTCMulticonnection`, en la tabla 4.4 se tiene un resumen de las principales referencias al protocolo en las páginas `index` y `room`.

Tabla 4. 4 Referencias WebRTC

REFERENCIA	FUNCIÓN
index.html	
<code>var connection = new RTCMultiConnection();</code>	Crea un objeto de la clase <code>RTCMulticonnection</code>
<code>connection.publicRoomIdentifier</code>	Define al room de tipo publico

<pre>var href = location.href + 'room.html?open=' + connection.isInitiator + '&sessionid=' + connection.sessionid + '&publicRoomIdentifier=' + connection.publicRoomIdentifier + '&userFullName=' + connection.extra.userFullName;</pre>	<p>Crea un enlace que se abre en una ventana nueva al llamar a room.html y envía los parámetros de sesión, identificadores de room y nombre de usuario.</p>
room.html	
<pre>var connection = new RTCMultiConnection();</pre>	<p>Crea un objeto de la clase RTCMultiConnection</p>
<pre>connection.maxParticipantsAllowed = 1000;</pre>	<p>Número máximo de participantes en el room</p>
<pre>connection.enableFileSharing = true;</pre>	<p>Habilita la compartición de archivos</p>
<pre>connection.session = { audio: true, video: true, data: true };</pre>	<p>Permite que en la sesión se habilite audio, video y datos</p>
<pre>function(file) connection.send(file, connection.getAllParticipants()[recentFile.userIndex]);</pre>	<p>Permite el envío de un archivo desde un usuario al room</p>
<pre>navigator.mediaDevices.getDisplayMedia({ video: true}).then(stream</pre>	<p>Permite desplegar una ventana de selección para la opción de compartir pantalla.</p>
<pre>function appendChatMessage(event, checkmark_id)</pre>	<p>Maneja el envío de mensajes instantáneos.</p>

Fuente: Del autor

4.4 Pruebas de funcionalidad

Para las pruebas se utiliza el servidor de despliegue y dominio Heroku que permite compilar, desplegar y escalar aplicaciones web en el servidor propio de Heroku, esta

aplicación tiene una cuenta gratuita que luego puede ser de paga al depender de la cantidad de recursos que requiera la aplicación.

Para crear una cuenta se necesita un correo y una contraseña, una vez verificado el correo se accede a la plataforma como se puede ver en a figura 4.16

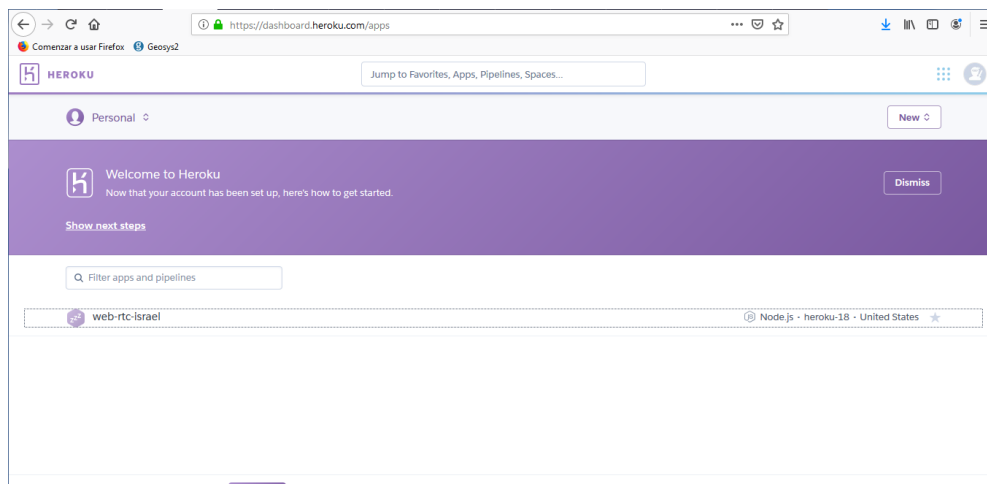


Figura 4. 16 Plataforma de Heroku

Fuente: El autor

Para desplegar la aplicación se debe instalar Heroku CLI en la máquina donde se realizó en desarrollo de la aplicación con las instrucciones provistas en el anexo 3.

Una vez instalado se debe abrir una ventana de terminal y se utiliza el comando `$ heroku login` para ingresar a la cuenta desde el terminal, para lo que se debe ingresar el correo y contraseña de la cuenta Heroku. El despliegue se realiza con los siguientes comandos:

```
$ heroku git:clone -a web-rtc-israel
$ cd web-rtc-israel
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

Al momento del despliegue Heroku entrega una URL que se utilizará para acceder a la aplicación desde cualquier explorador, para este proyecto es **https://web-rtc-**

israel.herokuapp.com/. En la figura 4.17 se puede ver la aplicación desplegada y la URL asignada por Heroku.



Figura 4. 17 Aplicación desplegada en Heroku

Fuente: El autor

Para realizar las pruebas de funcionalidad, se crea un room y se envía a otra persona la URL del proyecto donde se debe ingresar el mismo número de room que el creado, la contraseña e ingresar cualquier nombre que permita identificar al nuevo usuario. Para evitar que alguien intente crear un room que ya se hizo previamente se tiene una validación para este caso, en la figura 4.18 se puede ver el mensaje de validación.

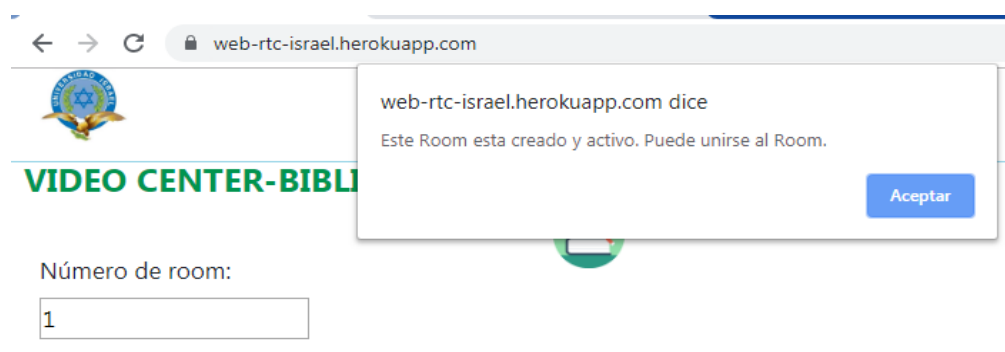


Figura 4. 18 Mensaje de validación para un room creado

Fuente: El autor

También puede pasar que alguien intente unirse a un room que no ha sido creado todavía o la contraseña es incorrecta, para lo cual también se tiene validación como se puede ver en la figura 4.19

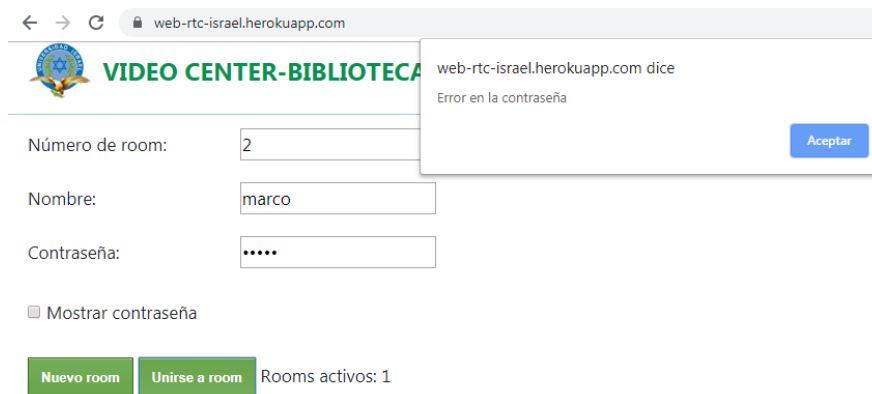
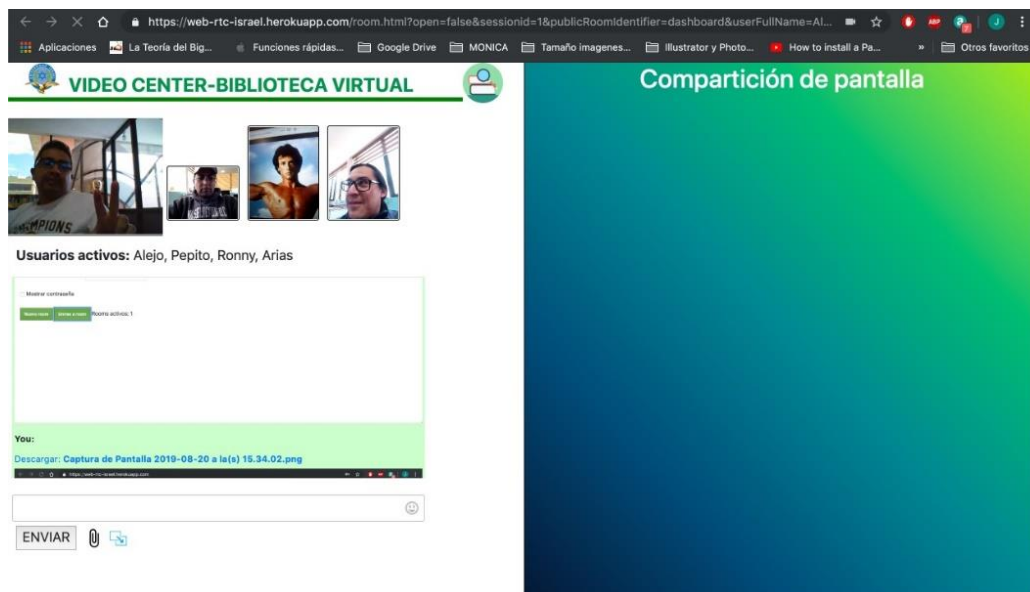


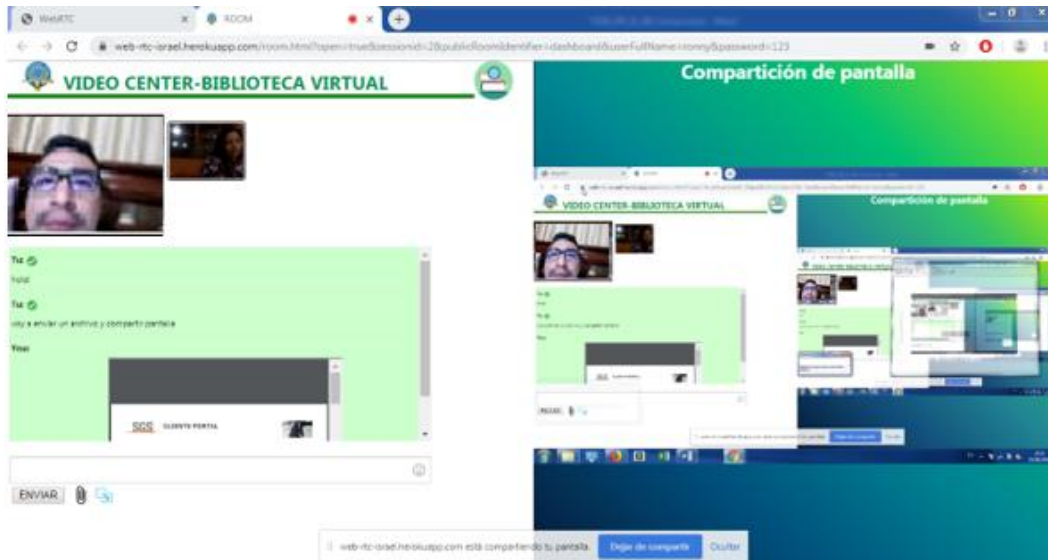
Figura 4.19 Mensaje de validación cuando no existe un room.

Fuente: El autor

Finalmente, al conectarse los dos usuarios se puede realizar la video llamada, intercambio de archivos, envío de mensajes instantáneos y compartición de pantalla como se puede ver en las figuras 4.20 a) y b)



a) Envío de mensajes instantáneos y compartición de archivos



b) Compartición de pantalla del creador

Figura 4. 20 Funcionalidades de la aplicación WebRTC

Fuente: El autor

Para las medir los niveles de calidad de servicio y la eficiencia se utiliza una herramienta de pruebas de carga Stress Stimulus. En la figura 4.21 se puede observar la prueba de carga realizada.

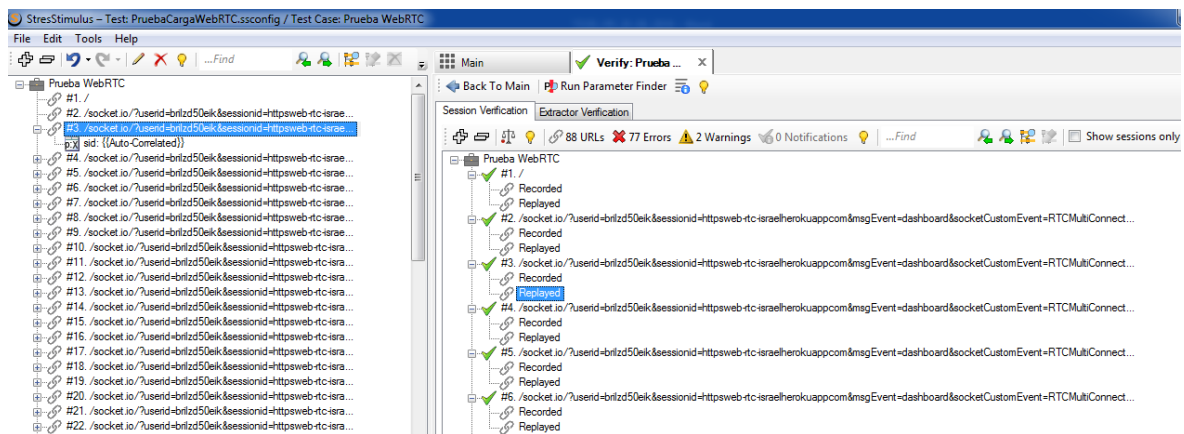


Figura 4. 21 Prueba de carga de la aplicación

Fuente: El autor

En la tabla 4.5 se observan los resultados resumidos. En el campo dirección se detalla el tipo de tráfico si es entrante o saliente y detalla en número de fallas y los usuarios concurrentes en el momento, se especifica solamente un servidor de prueba que es Heroku donde está la aplicación desplegada.

Tabla 4. 5 Resumen de la prueba de carga para la aplicación.

Parámetro	Tipo de tráfico	Fallas	Usuarios	Servidor
Datos de baja prioridad	Entrante	1	4	Heroku
Datos de baja prioridad	Saliente	0	3	Heroku
Conferencia multimedia (datos y video)	Entrante	0	5	Heroku
Conferencia multimedia (datos y video)	Saliente	0	5	Heroku
Multimedia streaming	Entrante	1	4	Heroku
Multimedia streaming	Saliente	1	3	Heroku
Todas las anteriores	Entrante	2	2	Heroku
Todas las anteriores	Saliente	3	5	Heroku

Fuente: El autor

Como se observa, en la tabla para la última iteración de la prueba se tuvo una alta tasa de error, esto debido a que Heroku en su versión libre asigna solamente un Dyno, que es una unidad de medida de ancho de banda cuyo valor numérico no está publicado en la documentación de Heroku por cuestiones de privacidad. Esto influye en el hecho de que, en el octavo intento, el error sea incrementado. Sin embargo, las primeras pruebas reflejan buenos parámetros de corrección de errores para la aplicación.

4.5 Análisis de las pruebas

Tras el desarrollo y las pruebas se realiza el análisis de la viabilidad de implementar las funcionalidades que ofrece el protocolo web RTC en una aplicación creada desde cero como se indica en el alcance del proyecto.

El protocolo WebRTC define para comunicaciones distribuidas las llamadas de audio y video, conocidas como video llamadas, las conferencias online con configuración maestro-esclavo, las presentaciones en vivo con compartición de pantalla, la compartición de archivos y una configuración en malla para las comunicaciones. Por tanto, la aplicación desarrollada se enfoca en implementar todas las funcionalidades del protocolo, apoyándose en lenguajes de programación de aplicaciones web para el diseño de la interfaz web y la interacción con el usuario. La aplicación debe ser también open source y usar herramientas y lenguajes abiertos para que el producto final pueda ser replicado y mejorado a costo cero o el menor posible.

4.5.1 Desafíos y problemáticas de la implementación de una aplicación de alto nivel

Para definir los desafíos de la implementación de la aplicación mediante el protocolo WebRTC se considera como elemento de comparación la aplicación de alto nivel Big Blue Button, que se encuentra disponible en la web al enlace <https://bigbluebutton.org>. La infraestructura se puede ver en la figura 4.13 y las características en la tabla 4.5.

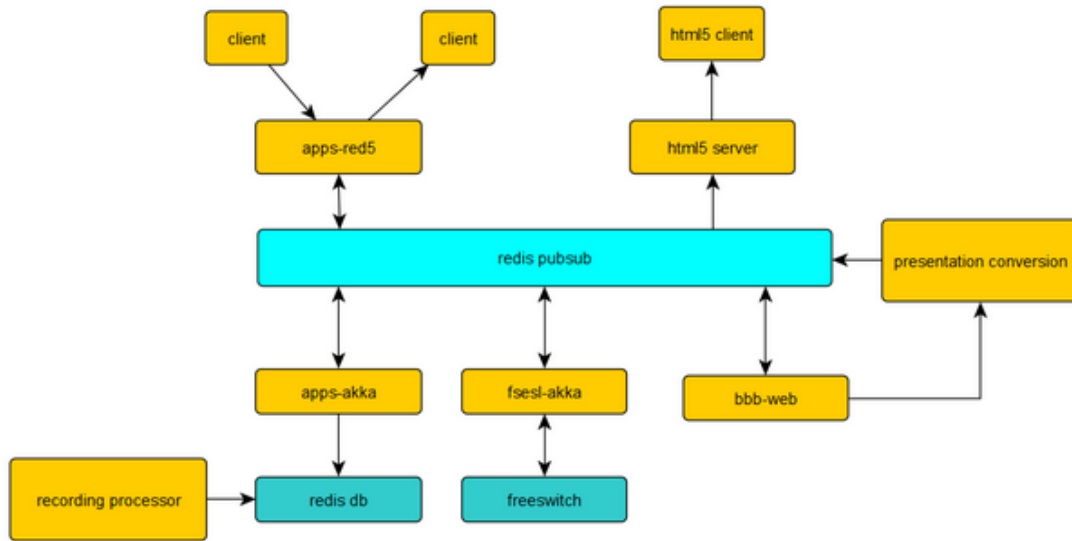


Figura 4. 22 Diagrama de aplicación Big Blue Button

Fuente: (bigbluebutton.org, 2019)

Tabla 4. 6 Funciones y herramientas de Big Blue Button

FUNCION	HERRAMIENTA
Comunicación cliente servidor	Meteor.js
Estado de cliente	Mongo DB
Interfaz de usuario	React.js
Recepción de audio y video	WebRTC
Integración con aplicaciones	BBB Web
Canal de comunicación	Redis PubSub
Grabar una reunión	Redis DB
Compartición de pantalla, aplicación de mensajes	Conjunto de aplicaciones Red5
Conferencia de voz	FreeSWITCH
Carga de presentación	FreeSWITCH

Fuente: (bigbluebutton.org, 2019)

La aplicación de Big Blue Button utiliza una serie de herramientas para implementar las funcionalidades y no está basada completamente en el protocolo WebRTC, sino que implementa sub aplicaciones como Red5 y FreeSWITCH que fueron desarrolladas

previamente. En contraste, la aplicación desarrollada para el presente proyecto utiliza una librería e implementa todas las funcionalidades a través de WebRTC como se definió en el tema y alcance.

A continuación, se listan los desafíos encontrados durante el desarrollo.

- Se necesitan conocimientos previos de programación web para el manejo de las páginas HTML, las hojas de estilo y la interacción con el cliente.
- Al ser un proyecto de titulación, no se puede depender de aplicaciones de terceros para implementar las funcionalidades definidas en el tema del proyecto, por tanto, el uso del protocolo tiene su base en la biblioteca JavaScript RTCMulticonnection. Esta biblioteca de código abierto desarrollada por un grupo MIT permite implementar videoconferencia, envío de mensajes, compartición de archivos, compartición de pantalla mediante las funciones que define la librería. La librería tiene más de 5000 líneas de código y requiere un conocimiento previo de JavaScript y node js, el estudio de los métodos y funciones es indispensable para desarrollar el proyecto.
- Para que la aplicación pueda ser desplegada, se requiere un servidor que permita el manejo de versiones y pruebas al menor costo. El servidor Heroku que es utilizado en el proyecto requiere un conocimiento de Git para el manejo de versiones, la lectura de manuales para la instalación del enlace al servidor desde la computadora donde se desarrolla la aplicación y el uso de comandos de consola para la carga de las páginas y archivos que conforman la aplicación.
- Una aplicación como Big Blue Button se desarrolla con un equipo de trabajo que constituye varios programadores con fortalezas especializadas en diferentes lenguajes de programación que pueden entregar un producto final de alto nivel, para un programador que se inicia en desarrollos alcanzar dicho nivel puede llevar mucho tiempo de aprendizaje y programación.

4.5.2 Parámetros de conexión para un nivel adecuado

- Ancho de banda

El ancho de banda que requiere la aplicación es el mismo que se requiere para el códec V8, cuyo valor mínimo es de 100 kbits/s. El audio en tiempo real tiene una tasa de 40-200 kbits/s y el video requiere al menos 200 kbits/s para una imagen reconocible, pero pixelada. Para que la imagen se pueda distinguir se requiere al menos 500 kbits/s.

Con estos datos se realizan pruebas de la aplicación con la herramienta webrtc-internals de Chrome. En la figura 4.14 se tiene una captura de la herramienta, la cual reconoce el uso del protocolo al momento de acceder a la pantalla de ingreso de información. Al crear el room y conectarse los usuarios, la herramienta empieza a desplegar peticiones y recolectar datos como se puede ver en la figura 4.15.

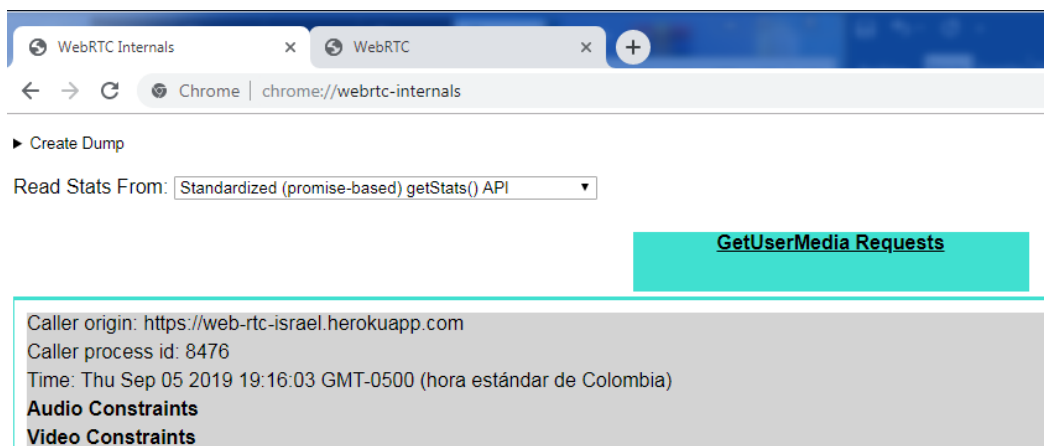


Figura 4. 23 Aplicación webrtc-internals

Fuente: (Chrome, 2019)

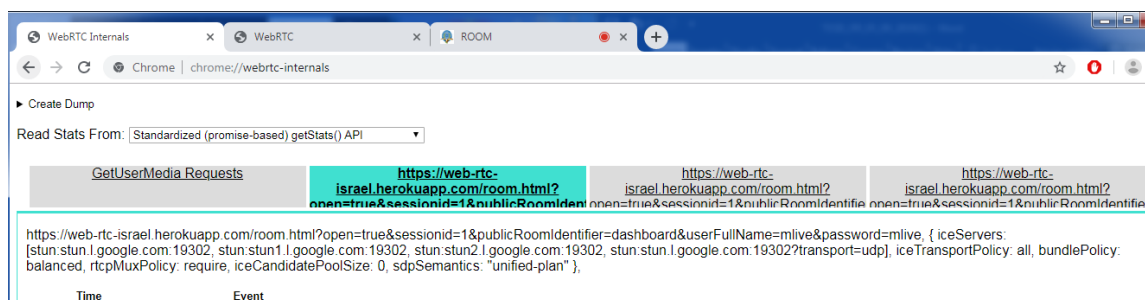


Figura 4. 24 Captura de datos de una sesión

Fuente: El autor

Dentro de la lista de datos que recoge la aplicación se puede ver la sección de video, como se muestra en la figura 4.16. La resolución del video para la aplicación es de 640 x 480 para el cuadro más grande, con 6 tramas por segundo durante la transmisión. Para el cálculo del ancho de banda requerido se asume 5 cámaras con la máxima resolución, una compresión de calidad media y las 6 tramas por segundo. El resultado del cálculo se realiza mediante la herramienta Bandwidth Calculator como se muestra en la figura 4.17 con el resultado de 0.6 Mbps

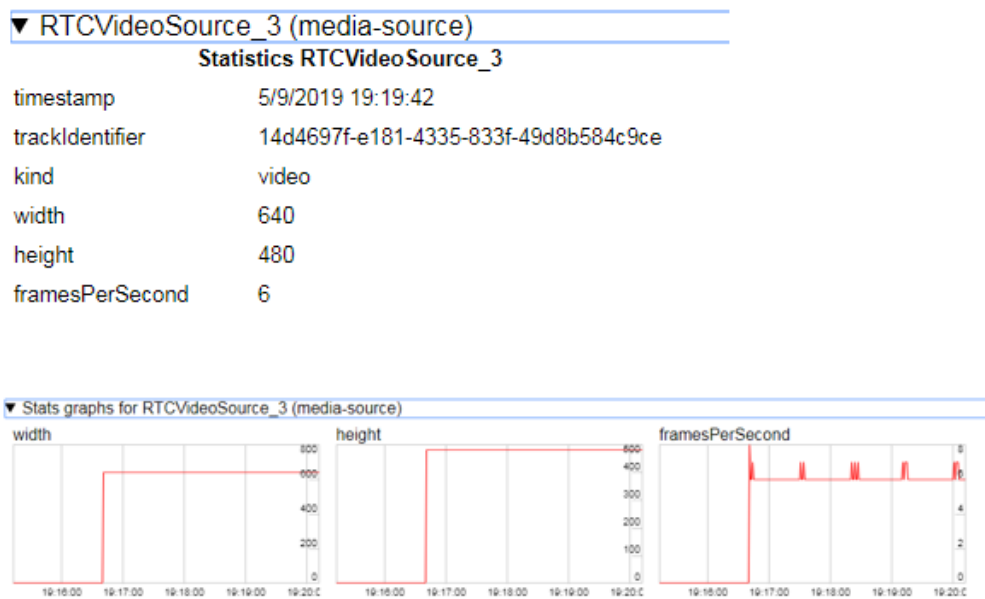


Figura 4. 25 Datos de video de la aplicación

Fuente: El autor

number of cameras:	<input type="text" value="5"/>
resolution:	<input type="text" value="0,3 MPx (640 × 480)"/> ▼
compression:	<input type="text" value="H.264 Base - medium quality"/> ▼
frame rate (fps):	<input type="text" value="6"/>
storage bandwidth (MB/s):	<input type="text" value="0.1"/>
network bandwidth (Mb/s):	<input type="text" value="0.6"/>
	<input type="button" value="calculate"/>

Figura 4. 26 Cálculo de ancho de banda

Fuente: El autor

Se considera que, durante las pruebas se evidencia una variación en la conexión y con los datos del protocolo, se acepta el valor de 0.6 Mbps para un nivel adecuado de calidad de video para una videoconferencia de cinco usuarios en una calidad adecuada.

- Verificación de compatibilidad de explorador

Durante las pruebas, en algunas ocasiones se evidencia que algún usuario tiene problemas para acceder y ser visto por los otros usuarios dentro del room. La razón es que el explorador que tiene desde el cual se intenta el acceso. Se debe recordar que de acuerdo a los requerimientos tanto para computadoras como para dispositivos es la versión 28 para Chrome y 24 para Firefox y si estos requisitos no se cumplen no se pueden acceder a las funcionalidades de la aplicación.

- Número de usuarios

El número de usuarios es programable antes del despliegue de la aplicación en el servidor mediante el cambio en una línea de código. Para el proyecto se determinó el número máximo de usuarios por las características del servidor Heroku en su versión gratuita. Heroku provee 5 MB para base de datos, 1 dyno que permite que la aplicación esté activa pendiente a solicitudes con un máximo de 600 peticiones por minuto. Los recursos de la aplicación están limitados a 300 MB en total y un límite de 1 TB /mes en ancho de banda y se debe considerar que cada usuario ocupa aproximadamente 1 Mb/s como se calculó anteriormente por lo que en una conferencia de 1 hora con el máximo de 5 usuarios se ocupa 1.8 GB, lo que se traduce en aproximadamente 900 GB en 500 horas de prueba al mes. El número de 500 horas al mes es adecuado para la realización de pruebas y esta es la razón por la cual se trabaja con un número bajo de usuarios.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Con el uso de RTCMulticonnection se define un espacio de trabajo conocido como room donde se puede realizar videoconferencia, compartición de archivos y pantalla y envío de mensajes instantáneos. El número máximo de rooms de acuerdo a RTCMulticonnection es ilimitado y de acuerdo a los parámetros de Heroku se puede trabajar hasta con 10 rooms simultáneos con 5 usuarios cada uno.

En cada room se puede definir el número máximo de usuarios que pueden acceder a un room mediante la propiedad maxParticipantAllowed, pero debe considerarse las limitaciones de los navegadores. De acuerdo a la documentación de la librería RTCMulticonnection para Google Chrome es de 256 y para Mozilla por defecto es 24 que puede incrementarse hasta 256.

El número máximo de usuarios también depende del ancho de banda que provee el servidor en el cual está desplegado, para Heroku se dispone de un ancho de banda de 1 TB/mes por lo que se limita el número de usuarios a 5 por room para disponer de 500 horas de acceso en 1 room y 50 horas para 10 rooms.

La implementación de una aplicación que usa el protocolo WebRTC requiere de un conocimiento previo y práctica en la programación de aplicaciones web con los lenguajes de programación JavaScript, node y HTML. Es necesario el análisis del protocolo como tal y el estudio del código y las funciones de la librería RTCMulticonnection, para conocer como se establece las conexiones que permiten la creación de la sala virtual, donde se realiza la videoconferencia, el envío de mensajería instantánea, la compartición de archivos y pantalla.

El proyecto muestra la viabilidad y los desafíos al desarrollar una aplicación que utilice solamente el protocolo WebRTC para el establecimiento de una sesión multimedia. Se conoce que existen aplicaciones de alto nivel que ya incluyen varias de las funcionalidades

presentadas, pero requieren un conjunto de programadores y utilizan para los módulos pequeñas aplicaciones previamente desarrolladas que son incorporadas a un proyecto mayor.

WebRTC tiene una librería de código abierto SimpleWebRTC, sin embargo, esta librería ahora está descontinuada y obsoleta, al inicio del proyecto se quiso utilizar esta librería y luego debió cambiarse al mostrar problemas de compatibilidad por lo que se recomienda hacer una pequeña prueba de compatibilidad para evitar la pérdida de tiempo en estudiar y aplicar una librería descontinuada.

RECOMENDACIONES

RTCMulticonnection tiene un repositorio en GitHub en el cual también se pueden realizar preguntas a la comunidad creadora de la librería en caso de presentarse un bug (error en el código original de la librería) o en caso de alguna duda en la implementación de código por lo tanto se recomienda que si se presenta una duda se pueda acudir a la comunidad.

El proyecto contempla la creación de rooms para la comunicación con una interfaz inicial simple, sin embargo, se puede añadir más funcionalidades a la interfaz inicial de modo que el administrador pueda llevar un control de que rooms están creados, cuantos participantes tienen y el tiempo que llevan enlazados, todos estos parámetros se pueden recuperar del objeto RTCMulticonnection.

El paso a producción de la aplicación requiere la adquisición de una cuenta premium en Heroku, que incrementa los valores de ancho de banda, almacenamiento y el estado de escucha de peticiones de la aplicación.

BIBLIOGRAFÍA

- Arc. (2019). *Development Resources*. Obtenido de www.arcdevelopment.com
- Barrera, J. (2010). *Metodología de la investigación*. Bogota: Quiron.
- Chacón, A. (2003). *La videoconferencia: conceptualización, elementos y uso educativo*. España: Universidad de Granada.
- Conallen, J. (2000). *Modelling web applications architectures with UML*. Massachusset: Communications of the ACM.
- Duke, C. (2009). *Beyond Nyquist*. Durham : CS Workshop.
- GitHub. (2016). *GitHub*. Obtenido de RTCMulticonnection: <https://github.com/muaz-khan/RTCMultiConnection>
- HerokuDevCenter. (2019). *Manual de instalación CLI*. Obtenido de <https://devcenter.heroku.com/articles/heroku-cli#download-and-install>
- JQueryFoundation. (2019). *JQuery* . Obtenido de <https://js.foundation/>
- Loza, J. (2014). *Introducción al audio digital*. Obtenido de <https://en.calameo.com/>
- Mendoza, C. (2016). *Diseño e implementación de un prototipo web para comunicación en tiempo real que permite realizar video llamadas, transferencia de archivos y chat, mediante el uso de WebRTC*. Quito: Escuela Politécnica Nacional.
- Rasheed, R. (2012). *Image and Sound Editing*. Obtenido de <https://slideplayer.com/slide/4584469/>
- RFC2616. (1999). *Hypertext Transfer Protocol -- HTTP/1.1*. Obtenido de <https://tools.ietf.org/html/rfc2616>
- RFC3711. (2004). *The Secure Real-time Transport Protocol (SRTP)*. Obtenido de <https://tools.ietf.org/html/rfc3711>
- RFC6347. (2012). *Datagram Transport Layer Security Version 1.2*. Obtenido de <https://tools.ietf.org/html/rfc6347>
- Rodríguez, J., & Fernández, R. (2005). *Códecs de video*. España: Escuela Técnica de Ingeniería Informática de Gijón.
- Tamayo, M. (20015). *El proceso de la identificación científica*. México: Editoriales Noriega.
- ThePHPGroup. (2019). *Qué es PHP*. Obtenido de <https://www.php.net/>

- W3Schools. (2019). *CSS Tutorial*. Obtenido de <https://www.w3schools.com/css/>
- WebRTC. (2011). *The WebRTC Project*. Obtenido de <https://webrtc.org/>
- Yáñez, A. (2011). *Formatos de audio y video: códecs*. Obtenido de <http://www.edu.xunta.gal>. México
- Zambrano, D. (2015). *Estudio de las características de nuevas arquitecturas web basadas en WebRTC alojada en la nube y factible implementación para aplicaciones de voz sobre IP (VoIP)*. Quito: Pontificia Universidad Católica del Ecuador.

ANEXOS

ANEXO 1

CÓDIGO DE INDEX.HTML

```
<!-- Versión 1.0 -->
```

```
<!DOCTYPE html>
```

```
<html lang="en" dir="ltr">
```

```
<!--Cabecera del código HTML-->
```

```
<head>
```

```
  <meta charset="utf-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

```
  <title>WebRTC</title>
```

```
  <meta name="description" content="Aplicacion WebRTC">
```

```
<!--Lista de enlaces a los archivos CSS y JS -->
```

```
  <link rel="stylesheet" href="/css/bootstrap.min.css">
```

```
  <link rel="stylesheet" type="text/css" href="/css/estilo.css">
```

```
  <script src="/js/jquery-3.3.1.slim.min.js"></script>
```

```
  <script src="/js/popper.min.js"></script>
```

```
  <script src="/js/bootstrap.min.js"></script>
```

```
  <script src="/js/RTCMultiConnection.min.js"></script>
```

```
  <script src="/socket.io/socket.io.js"></script>
```

```
</head>
```

```
<body>
```

```
<!--Cabecera de la aplicación-->
```

```
  <header style="margin-bottom: 20px;">
```

```
    <div style="margin-bottom: 40px; width:46%;">
```

```
      
```

```
      <span class="icono-text"> VIDEO CENTER-BIBLIOTECA VIRTUAL</span>
```



```


</div>
<!--Campos para ingreso de usuario, numero de room y contraseña -->
<div style="margin-top: 10px; margin-left: 20px;">
    <p>
        <label for="txt-roomid" style="width:200px;" >Número de room:</label>
        <input type="text" id="txt-roomid">
    </p>

    <p>
        <label for="txt-user-name" style="width:200px;">Nombre:</label>
        <input type="text" id="txt-user-name">
    </p>

    <p>
        <label for="txt-room-password" style="width:200px;">Contraseña:</label>
        <input type="password" id="txt-room-password-hidden"> <br><br>
        <input type="checkbox" onclick="mostrarContrasena()" style="margin-
right:5px;">Mostrar contraseña
    </p>

</div>

<!--Botones para acceder o unirse al room -->
<div style="float: left; margin-top: 15px; margin-left: 20px;">
    <button id="nuevoRoom" class="btnVerde">Nuevo room</button>
    <button id="unirseRoom" class="btnVerde">Unirse a room</button>
    <span class="top-span">Rooms activos: <span id="active-rooms">0</span></span>
</div>

<script>

// Muestra el password escrito
function mostrarContrasena() {
    var x = document.getElementById("txt-room-password-hidden");

```

```

if (x.type === "password") {
    x.type = "text";
} else {
    x.type = "password";
}
}
//
var publicRoomIdentifier = 'dashboard';

// Crea una objeto de la clase RTCMulticonnection
var connection = new RTCMultiConnection();

//inicializa el socket de comunicación
connection.socketURL = '/';

/// Room actual de tipo publico
connection.publicRoomIdentifier = publicRoomIdentifier;
connection.socketMessageEvent = publicRoomIdentifier;

// Mantener Room abierto
connection.autoCloseEntireSession = true;

connection.connectSocket(function(socket) {
    looper();
    socket.on('disconnect', function() {
        location.reload();
    });
});

//Crea nuevo room
$('#nuevoRoom').click(function() {
    var roomid = $('#txt-roomid').val().toString();
    if (!roomid || !roomid.replace(/ /g, "").length) {
        alert('Se requiere número de Room');
    }
});

```

```

        return;
    }

//Recupera el nombre del usuario que se ingresa
    var fullName = $('#txt-user-name').val().toString();
    if (!fullName || !fullName.replace(/ /g, "").length) {
        alert('Se requiere su nombre');
        return;
    }

    var roomPassword = $('#txt-room-password-hidden').val().toString();
    if (!roomPassword || !roomPassword.replace(/ /g, "").length) {
        alert('Ingrese una contraseña para crear el room');
        return;
    }

    connection.password = roomPassword;
    connection.extra.userFullName = fullName;

    var initialHTML = $('#btn-create-room').html();

    $('#nuevoRoom').html('Espere...').prop('disabled', true);

// verifica si el room fue creado previamente y envía un mensaje al usuario
    connection.checkPresence(roomid, function(isRoomExist) {
        if (isRoomExist === true) {
            alert('Este Room esta creado y activo. Puede unirse al Room.');
```

```

    $('#nuevoRoom').html(initialHTML).prop('disabled', false);
  });
});

// Función para verificar datos antes de unirse al room
// se valida si el usuario ingresa el número y el usuario
$('#unirseRoom').click(function() {
  var roomid = $('#txt-roomid').val().toString();
  if (!roomid || !roomid.replace(/ /g, "").length) {
    alert('Se requiere número de Room');
    return;
  }

  var fullName = $('#txt-user-name').val().toString();
  if (!fullName || !fullName.replace(/ /g, "").length) {
    alert('Se requiere su nombre');
    return;
  }

  var roomPassword = $('#txt-room-password-hidden').val().toString();
  if (!roomPassword || !roomPassword.replace(/ /g, "").length) {
    alert('Se requiere una contraseña para acceder el room');
    return;
  }

  connection.extra.userFullName = fullName;
  connection.password = roomPassword;

  connection.socket.emit('is-valid-password', connection.password, roomid,
function(isValidPassword, roomid, error) {
  if(isValidPassword === true) {
    joinARoom(roomid);
  }
}

```

```

    else {
        alert('Error en la contraseña');
    }
});
return;

});

// Función para unirse al room
function joinARoom(roomid) {
    var initialHTML = $('#unirseRoom').html();

    $('#unirseRoom').html('Espere...').prop('disabled', true);

    connection.checkPresence(roomid, function(isRoomExist) {
        if (isRoomExist === false) {
            alert('No existe el Room. Room-id: ' + roomid);
            $('#unirseRoom').html(initialHTML).prop('disabled', false);
            return;
        }

        var roomPassword = $('#txt-room-password-hidden').val().toString();
        if (!roomPassword || !roomPassword.replace(/ /g, "").length) {
            alert('Se requiere una contraseña para acceder el room');
            return;
        }

        connection.password = roomPassword;
        connection.sessionid = roomid;
        connection.isInitiator = false;
        openRoom();

        $('#unirseRoom').html(initialHTML).prop('disabled', false);
    })
}

```

```
}
```

```
// Función de apertura de room
```

```
function openRoom() {
```

```
// crea el enlace al room con el id de sesión, el usuario y el id de room. Apunta a room.html
```

```
    var href = location.href + 'room.html?open=' + connection.isInitiator + '&sessionId=' +  
connection.sessionid + '&publicRoomIdentifier=' + connection.publicRoomIdentifier +  
'&userFullName=' + connection.extra.userFullName;
```

```
    if(!connection.password) {
```

```
        href += '&password=' + connection.password;
```

```
    }
```

```
// abre la ventana de room en una pestaña nueva.
```

```
    var newWin = window.open(href);
```

```
    if (!newWin || newWin.closed || typeof newWin.closed == 'undefined') {
```

```
        var html = "";
```

```
        html += '<p>Siga el enlace:</p>';
```

```
        html += '<p><a href="' + href + '" target="_blank">';
```

```
        if(connection.isInitiator) {
```

```
            html += 'Abrir un Room';
```

```
        }
```

```
    } else {
```

```
        html += 'Unirse a Room';
```

```
    }
```

```
    html += '</a></p>';
```

```
    alertBox(html, 'Popups estan deshabilitados');
```

```
    }
```

```
}
```

```
// La página principal verifica de acuerdo al timeout por si se han creado nuevos rooms
```

```
function looper() {
```

```
    connection.socket.emit('get-public-rooms', publicRoomIdentifier, function(listOfRooms) {
```

```
        updateListOfRooms(listOfRooms);
```

```
        setTimeout(looper, 3000);
```

```
    });
```

```
}  
  
var roomCount;  
  
// actualiza la etiqueta del número de rooms creados  
function updateListOfRooms(rooms) {  
    if (rooms.lenght != 0)  
        roomCount = rooms.length;  
    roomCount = roomCount;  
    $('#active-rooms').html(roomCount);  
    $("#active-rooms").text(roomCount);  
    return;  
}  
</script>  
</body>
```

ANEXO 2

CÓDIGO DE ROOM.HTML


```
<!-- Version 1.0 -->
<!DOCTYPE html>
<html lang="en" dir="ltr">

<!--Cabecera del código HTML-->
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>ROOM</title>
  <meta name="description" content="Aplicacion WebRTC">

<!--Enlaces a los archivos CSS y JS -->
  <link rel="shortcut icon" href="/images/uisrael.png">
  <link rel="stylesheet" type="text/css" href="/css/emojionearea.min.css">
  <link rel="stylesheet" type="text/css" href="/css/estiloRoom.css">
  <script src="/js/jquery.min.js"></script>
  <link href="/css/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="/css/estiloRoom.css">
  <script src="/node_modules/webrtc-adapter/out/adapter.js"></script>
  <script src="/js/RTCMultiConnection.min.js"></script>
  <script src="/socket.io/socket.io.js"></script>
  <script src="/node_modules/fbr/FileBufferReader.js"></script>
  <script src="/js/emojionearea.min.js"></script>

</head>
<body>

<!-- Contenedor de compartición de pantalla -->
<div id="widget-container" style="position: fixed;bottom: 0;right: 0;left: 50%;height:
100%;border: 1px solid black; border-top:0; border-bottom: 0; background-image: linear-
gradient(to right top, #051937, #004d7a, #008793, #00bf72, #a8eb12);">
  <div">
    <h2 style="text-align: center; color:#ffffff;"> Compartición de pantalla </h2>
    <video id="screen-viewer" controls playsinline autoplay></video>
```

```

</div>
</div>
<header style="margin-bottom: 5px;">
  <div style="margin-bottom: 20px; border-bottom: 5px solid green; width:48%;">
    
    <span class="icono-text"> VIDEO CENTER-BIBLIOTECA VIRTUAL</span>
    
  </div>
  <!-- Contenedor de videos -->
  <div style="overflow:hidden;">
    <div style="border-top: 1px solid #E5E5E5; float: left; width:15%">
      <video id="main-video" controls playsinline autoplay></video>
    </div>
    <div id="other-videos" style="float:left; width:70%;"></div>
  </div>
  <div style="padding: 5px 10px;">
    <div id="onUserStatusChanged"></div>
  </div>
  <!-- Contenedor de mensajes -->
  <div style="margin-top: 10px;position: absolute;bottom: 200px;background: white;
padding-bottom: 20px; width: 40%; height: 30%; margin-left:5px;">
    <div id="conversation-panel"></div>
    <div id="key-press" style="text-align: right; display: none; font-size: 10px;">
      <span style="vertical-align: middle;"></span>
      
    </div>
    <textarea id="txt-chat-message"></textarea>
    <button class="btnVerde" id="btn-chat-message" disabled>ENVIAR</button>
  </div>
  <!--Botones dee compartición de archivos y compartición de pantalla -->
  

```

```

```

```
</div>
```

```
<canvas id="temp-stream-canvas" style="display: none;"></canvas>
```

```
<script>
```

```
(function() {
```

```
  var params = {},
```

```
  r = /([\&=]+)=?([\&]*)/g;
```

```
// Recupera la URL que viene desde index.html y le da un formato adecuado
```

```
function d(s) {
```

```
  return decodeURIComponent(s.replace(/\+/g, ' '));
```

```
}
```

```
var match, search = window.location.search;
```

```
while (match = r.exec(search.substring(1)))
```

```
  params[d(match[1])] = d(match[2]);
```

```
  // Recupera los parámetros que vienen en la url
```

```
  window.params = params;
```

```
})();
```

```
// Crea un objeto RTCMulticonnection
```

```
var connection = new RTCMultiConnection();
```

```
connection.socketURL = '/';
```

```
// Recupera el nombre de usuario
```

```
connection.extra.userFullName = params.userFullName;
```

```
// Recupera el identificador de room
```

```
connection.publicRoomIdentifier = params.publicRoomIdentifier;
```

```
connection.socketMessageEvent = 'room_connection';

connection.autoCloseEntireSession = true;

// Número máximo de participantes en el room
connection.maxParticipantsAllowed = 5;

// Librería RTCMultiConnection
connection.chunkSize = 16000;

// Habilita la compartición de archivos
connection.enableFileSharing = true;

// Establece para la sesión el uso de audio, video y datos
connection.session = {
  audio: true,
  video: true,
  data: true
};

// Solicita permiso para uso de audio, video y datos
connection.sdpConstraints.mandatory = {
  OfferToReceiveAudio: true,
  OfferToReceiveVideo: true
};

// Revisa si un usuario se ha añadido al room
connection.onUserStatusChanged = function(event) {
  var infoBar = document.getElementById('onUserStatusChanged');
  var names = [];
  connection.getAllParticipants().forEach(function(pid) {
    names.push(getFullName(pid));
  });
};
```

```

if (!names.length) {
    names = ['Creador!'];
} else {
    names = [connection.extra.userFullName || 'Tu'].concat(names);
}

infoBar.innerHTML = '<b>Usuarios activos:</b>' + names.join(', ');
};

// Cambia la presentación del room cuando se tiene al menos dos usuarios
connection.onopen = function(event) {
    connection.onUserStatusChanged(event);

    document.getElementById('btn-chat-message').disabled = false;
    document.getElementById('btn-attach-file').style.display = 'inline-block';
    document.getElementById('btn-share-screen').style.display = 'inline-block';
};

// Maneja la salida de usuarios del room
connection.onclose = connection.onerror = connection.onleave = function(event) {
    connection.onUserStatusChanged(event);
};

// Maneja el formato para el envío de mensajes
connection.onmessage = function(event) {
    if(event.data.showMainVideo) {
        // $('#main-video').show();
        $('#screen-viewer').css({
            top: $('#widget-container').offset().top,
            left: $('#widget-container').offset().left,
            width: $('#widget-container').width(),
            height: $('#widget-container').height()
        });
    }
};

```

```
$('#screen-viewer').show();  
return;  
}
```

```
if(event.data.hideMainVideo) {  
    $('#main-video').hide();  
    $('#screen-viewer').hide();  
    return;  
}
```

// Muestra una notificación de que usuario escribe un mensaje

```
if(event.data.typing === true) {  
    $('#key-press').show().find('span').html(event.extra.userName + ' está escribiendo');  
    return;  
}
```

```
if(event.data.typing === false) {  
    $('#key-press').hide().find('span').html("");  
    return;  
}
```

```
if (event.data.chatMessage) {  
    appendChatMessage(event);  
    return;  
}
```

```
if (event.data.checkmark === 'received') {  
    var checkmarkElement = document.getElementById(event.data.checkmark_id);  
    if (checkmarkElement) {  
        checkmarkElement.style.display = 'inline';  
    }  
    return;  
}
```

```
};
```

```
// Maneja el stream de video al iniciar la conexión
```

```
connection.onstream = function(event) {
```

```
  if (event.stream.isScreen && !event.stream.canvasStream) {
```

```
    $('#screen-viewer').get(0).srcObject = event.stream;
```

```
    // $('#screen-viewer').hide();
```

```
  }
```

```
  // Presentación de video principal para el usuario creador del room
```

```
  else if (event.extra.roomOwner === true) {
```

```
    var video = document.getElementById('main-video');
```

```
    video.setAttribute('data-streamid', event.streamid);
```

```
    // video.style.display = 'none';
```

```
    if(event.type === 'local') {
```

```
      video.muted = true;
```

```
      video.volume = 0;
```

```
    }
```

```
    video.srcObject = event.stream;
```

```
    $('#main-video').show();
```

```
  }
```

```
  // Presentación de videos secundarios para los demás usuarios del room
```

```
  else {
```

```
    event.mediaElement.controls = true;
```

```
    var otherVideos = document.querySelector('#other-videos');
```

```
    otherVideos.appendChild(event.mediaElement);
```

```
  }
```

```
connection.onUserStatusChanged(event);
```

```
};
```

```
// Maneja la finalización del stream tanto si se desconecta un usuario como si se termina la  
compartición de pantalla
```

```
connection.onstreamended = function(event) {  
    var video = document.querySelector('video[data-streamid="' + event.streamid + "']");  
    if (!video) {  
        video = document.getElementById(event.streamid);  
        if (video) {  
            video.parentNode.removeChild(video);  
            return;  
        }  
    }  
    if (video) {  
        video.srcObject = null;  
        video.style.display = 'none';  
    }  
};
```

```
// revisa que la conexión se encuentre abierta y el room esté activo
```

```
if (params.open === true || params.open === 'true') {
```

```
    // Almacena en un stream temporal el video de la cámara para intercambiarlo con  
    el de compartición de pantalla
```

```
        var tempStreamCanvas = document.getElementById('temp-stream-  
canvas');  
        var tempStream = tempStreamCanvas.captureStream();  
        tempStream.isScreen = true;  
        tempStream.streamid = tempStream.id;  
        tempStream.type = 'local';  
        connection.attachStreams.push(tempStream);  
        window.tempStream = tempStream;  
  
        connection.extra.roomOwner = true;
```



```

connection.open(params.sessionid, function(isRoomOpened, roomid, error) {
// si alguien trata de ingresar al room solo al usar la url se envía un mensaje también
    if (error) {
        if (error === connection.errors.ROOM_NOT_AVAILABLE) {
            alert('Alguien ya creo este room. Únete o crea otro room!');
            return;
        }
        alert(error);
    }
// desconecta la sesión
    connection.socket.on('disconnect', function() {
        location.reload();
    });
});
} else {
    connection.join(params.sessionid, function(isRoomJoined, roomid, error) {
        if (error) {
            if (error === connection.errors.ROOM_NOT_AVAILABLE) {
                alert('El room no existe. Por favor crea un room o espera a que el moderador
inicie el grupo');
                return;
            }
            if (error === connection.errors.ROOM_FULL) {
                alert('El room está lleno');
                return;
            }
            if (error === connection.errors.INVALID_PASSWORD) {
                connection.password = prompt('Please enter room password.') || "";
                if(!connection.password.length) {
                    alert('Invalid password. ');
                    return;
                }
            }
            connection.join(params.sessionid, function(isRoomJoined, roomid, error) {
                if(error) {

```

```

        alert(error);
    }
    });
    return;
}
alert(error);
}

connection.socket.on('disconnect', function() {
    location.reload();
});
});
}

```

```
var conversationPanel = document.getElementById('conversation-panel');
```

```
function appendChatMessage(event, checkmark_id) {
```

```
    var div = document.createElement('div');
```

```
    div.className = 'message';
```

```
    if (event.data) {
```

```
        div.innerHTML = '<b>' + (event.extra.userFullName || event.userid) + ':</b><br>' +
event.data.chatMessage;
```

```
        if (event.data.checkmark_id) {
```

```
            connection.send({
                checkmark: 'received',
                checkmark_id: event.data.checkmark_id
            });
        }
    }
} else {
```

```
    // Maneja el enunciado de los mensajes para distinguir los usuarios
```

```
div.innerHTML = '<b>Tu:</b> <br>' +  
event;
```

```
div.style.background = '#cbffcb';  
}
```

```
conversationPanel.appendChild(div);
```

```
conversationPanel.scrollTop = conversationPanel.clientHeight;  
conversationPanel.scrollTop = conversationPanel.scrollHeight -  
conversationPanel.scrollTop;  
}
```

```
var keyPressTimer;  
var numberOfKeys = 0;
```

```
$('#txt-chat-message').emojioneArea({  
  pickerPosition: "top",  
  filtersPosition: "bottom",  
  tones: false,  
  autocomplete: true,  
  inline: true,  
  hidePickerOnBlur: true,  
  events: {  
    focus: function() {  
      $(''.emojionearea-category').unbind('click').bind('click', function() {  
        $(''.emojionearea-button-close').click();  
      });  
    },  
    keyup: function(e) {  
      var chatMessage = $(''.emojionearea-editor').html();  
      if (!chatMessage || !chatMessage.replace(/ /g, "").length) {  
        connection.send({
```

```

        typing: false
    });
}

clearTimeout(keyPressTimer);
numberOfKeys++;

if (numberOfKeys % 3 === 0) {
    connection.send({
        typing: true
    });
}

keyPressTimer = setTimeout(function() {
    connection.send({
        typing: false
    });
}, 1200);
},
blur: function() {
    // $('#btn-chat-message').click();
    connection.send({
        typing: false
    });
}
}
});

window.onkeyup = function(e) {
    var code = e.keyCode || e.which;
    if (code == 13) {
        $('#btn-chat-message').click();
    }
}

```

```
};
```

```
document.getElementById('btn-chat-message').onclick = function() {
```

```
    var chatMessage = $('#emojionearea-editor').html();
```

```
    $('#emojionearea-editor').html("");
```

```
    if (!chatMessage || !chatMessage.replace(/ /g, "").length) return;
```

```
    var checkmark_id = connection.userid + connection.token();
```

```
    appendChatMessage(chatMessage, checkmark_id);
```

```
    connection.send({
```

```
        chatMessage: chatMessage,
```

```
        checkmark_id: checkmark_id
```

```
    });
```

```
    connection.send({
```

```
        typing: false
```

```
    });
```

```
};
```

```
var recentFile;
```

```
document.getElementById('btn-attach-file').onclick = function() {
```

```
    var file = new FileSelector();
```

```
    file.selectSingleFile(function(file) {
```

```
        recentFile = file;
```

```
        if(connection.getAllParticipants().length >= 1) {
```

```
            recentFile.userIndex = 0;
```

```
            connection.send(file, connection.getAllParticipants()[recentFile.userIndex]);
```

```
        }
```

```
    });
```

```
};
```

```

function getFileHTML(file) {
    var url = file.url || URL.createObjectURL(file);
    var attachment = '<a href="' + url + '" target="_blank" download="' + file.name +
">Descargar: <b>' + file.name + '</b></a>';
    if (file.name.match(/\.(jpg|png|jpeg|gif|gi)) {
        attachment += '<br>';
    } else if (file.name.match(/\.(wav|mp3|gi)) {
        attachment += '<br><audio src="' + url + '" controls></audio>';
    } else if (file.name.match(/\.(pdf|js|txt|sh|gi)) {
        attachment += '<iframe class="inline-iframe" src="' + url + '"></iframe></a>';
    }
    return attachment;
}

```

```

function getFullName(userid) {
    var _userFullName = userid;
    if (connection.peers[userid] && connection.peers[userid].extra.userFullName) {
        _userFullName = connection.peers[userid].extra.userFullName;
    }
    return _userFullName;
}

```

```

connection.onFileEnd = function(file) {
    var html = getFileHTML(file);
    var div = progressHelper[file.uuid].div;

    if (file.userid === connection.userid) {
        div.innerHTML = '<b>You:</b><br>' + html;
        div.style.background = '#cbffcb';

        if(recentFile) {
            recentFile.userIndex++;
            var nextUserId = connection.getAllParticipants()[recentFile.userIndex];

```

```
    if(nextUserId) {
        connection.send(recentFile, nextUserId);
    }
    else {
        recentFile = null;
    }
}
else {
    recentFile = null;
}
} else {
    div.innerHTML = '<b>' + getFullName(file.userid) + ':</b><br>' + html;
}
};
```

// to make sure file-saver dialog is not invoked.

```
connection.autoSaveToDisk = false;
```

```
var progressHelper = {};
```

```
connection.onFileProgress = function(chunk, uuid) {
    var helper = progressHelper[chunk.uuid];
    helper.progress.value = chunk.currentPosition || chunk.maxChunks || helper.progress.max;
    updateLabel(helper.progress, helper.label);
};
```

```
connection.onFileStart = function(file) {
    var div = document.createElement('div');
    div.className = 'message';
```

```
    if (file.userid === connection.userid) {
        var userFullName = file.remoteUserId;
        if(connection.peersBackup[file.remoteUserId]) {
            userFullName = connection.peersBackup[file.remoteUserId].extra.userFullName;
```

```

    }

    div.innerHTML = '<b>You (to: ' + userFullName + '):</b><br><label>0%</label>
<progress></progress>';
    div.style.background = '#cbffcb';
    } else {
        div.innerHTML = '<b>' + getFullName(file.userid) + ':</b><br><label>0%</label>
<progress></progress>';
    }

    div.title = file.name;
    conversationPanel.appendChild(div);
    progressHelper[file.uuid] = {
        div: div,
        progress: div.querySelector('progress'),
        label: div.querySelector('label')
    };
    progressHelper[file.uuid].progress.max = file.maxChunks;

    conversationPanel.scrollTop = conversationPanel.clientHeight;
    conversationPanel.scrollTop = conversationPanel.scrollHeight -
conversationPanel.scrollTop;
};

function updateLabel(progress, label) {
    if (progress.position == -1) return;
    var position = +progress.position.toFixed(2).split('.')[1] || 100;
    label.innerHTML = position + '%';
}

if(!params.password) {
    connection.password = params.password;
}

```



```

function addStreamStopListener(stream, callback) {
  stream.addEventListener('ended', function() {
    callback();
    callback = function() {};
  }, false);
  stream.addEventListener('inactive', function() {
    callback();
    callback = function() {};
  }, false);
  stream.getTracks().forEach(function(track) {
    track.addEventListener('ended', function() {
      callback();
      callback = function() {};
    }, false);
    track.addEventListener('inactive', function() {
      callback();
      callback = function() {};
    }, false);
  });
}

function replaceTrack(videoTrack, screenTrackId) {
  if (!videoTrack) return;
  if (videoTrack.readyState === 'ended') {
    alert('Can not replace an "ended" track. track.readyState: ' + videoTrack.readyState);
    return;
  }
  connection.getAllParticipants().forEach(function(pid) {
    var peer = connection.peers[pid].peer;
    if (!peer.getSenders) return;
    var trackToReplace = videoTrack;
    peer.getSenders().forEach(function(sender) {
      if (!sender || !sender.track) return;

```

```

    if(screenTrackId) {
        if(trackToReplace && sender.track.id === screenTrackId) {
            sender.replaceTrack(trackToReplace);
            trackToReplace = null;
        }
        return;
    }

    if (sender.track.kind === 'video' && trackToReplace) {
        sender.replaceTrack(trackToReplace);
        trackToReplace = null;
    }
});
});
}

```

```

function replaceScreenTrack(stream) {
    stream.isScreen = true;
    stream.streamid = stream.id;
    stream.type = 'local';
    // connection.attachStreams.push(stream);
    connection.onstream({
        stream: stream,
        type: 'local',
        streamid: stream.id,
        // mediaElement: video
    });
    var screenTrackId = stream.getTracks()[0].id;
    addStreamStopListener(stream, function() {
        connection.send({
            hideMainVideo: true
        });
        $('#main-video').hide();
        $('#btn-share-screen').show();
    });
}

```

```

    replaceTrack(tempStream.getTracks()[0], screenTrackId);

});
stream.getTracks().forEach(function(track) {
    if(track.kind === 'video' && track.readyState === 'live') {
        replaceTrack(track);
    }
});
connection.send({
    showMainVideo: true
});
// $('#main-video').show();
$('#screen-viewer').css({
    top: $('#widget-container').offset().top,
    left: $('#widget-container').offset().left,
    width: $('#widget-container').width(),
    height: $('#widget-container').height()
});
$('#screen-viewer').show();
}
$('#btn-share-screen').click(function() {

if(navigator.mediaDevices.getDisplayMedia) {
    navigator.mediaDevices.getDisplayMedia({ video: true }).then(stream => {
        replaceScreenTrack(stream);
    }, error => {
        alert('Please make sure to use Edge 17 or higher.');
```

```
    });  
  }  
  else {  
    alert('getDisplayMedia API is not available in this browser.');
```

```
  }
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

ANEXO 3

Manual Heroku (HerokuDevCenter, 2019)

Standalone installation

The standalone install is a simple tarball with a binary that is useful in scripted environments or where there is restricted access (non-sudo). It contains its own node.js binary and will autoupdate like the above install methods. **We encourage this method inside docker containers.**

To quickly setup into `/usr/local/lib/heroku` and `/usr/local/bin/heroku`, run this script (script requires sudo and not Windows compatible):

```
curl https://cli-assets.heroku.com/install.sh | sh
```

Otherwise, download one of the tarballs below and extract it yourself.

Tarballs

These are available in `gz` or `xz` compression. `xz` is much smaller but `gz` is more compatible.

macOS

Linux (x64)

Linux (arm)

Windows (x64)

Windows (x86)

Ubuntu / Debian apt-get

```
curl https://cli-assets.heroku.com/install-ubuntu.sh | sh
```

This version does not autoupdate and must be updated manually via `apt-get`. Use the `snap` or standalone installation for an autoupdating version of the CLI.

Arch Linux

This package is community maintained and **not** by Heroku.

```
yay -S heroku-cli
```

npm

The CLI is built with Node.js and is installable via `npm`. This is a manual install method that can be used in environments where autoupdating is not ideal or where Heroku does not offer a prebuilt Node.js binary.

It's strongly recommended to use one of the other installation methods if possible.

This installation method does not autoupdate and requires you to use your system's version of Node.js, which may be older than the version Heroku develops the CLI against. Heroku uses very current releases of Node.js and does not back-support older versions.

If you use any of the other installation methods the proper version of Node.js is already included, and it doesn't conflict with any other version on your system.

Also, this method won't use the yarn lockfile for dependencies like the others do (even if you install with yarn). This may cause issues if the CLI's dependencies become incompatible in minor or patch releases.

This installation method is required for users on ARM and BSD. You must have `node` and `npm` installed already.

```
npm install -g heroku
```

Verifying your installation

To verify your CLI installation, use the `heroku --version` command:

```
heroku --version
```

```
heroku/7.0.0 (darwin-x64) node-v8.0.0
```

You should see `heroku/x.y.z` in the output. If you don't, but you have installed the Heroku CLI, it's possible you have an old `heroku` gem on your system. Uninstall it with these instructions.

Getting started

After you install the CLI, run the `heroku login` command. You'll be prompted to enter any key to go to your web browser to complete login. The CLI will then log you in automatically.

```
heroku login
```

```
heroku: Press any key to open up the browser to login or q to exit
```

```
> Warning: If browser does not open, visit
```

```
> https://cli-auth.heroku.com/auth/browser/***
```

```
heroku: Waiting for login...
```

```
Logging in... done
```

```
Logged in as me@example.com
```

If you'd prefer to stay in the CLI to enter your credentials, you may run `heroku login -i`

```
heroku login -i
```

```
heroku: Enter your login credentials
```

```
Email: me@example.com
```

```
Password: *****
```

```
Two-factor code: *****
```

```
Logged in as me@heroku.com
```

The CLI saves your email address and an API token to `~/.netrc` for future use. For more information, see [Heroku CLI Authentication](#).

Now you're ready to create your first Heroku app:

```
cd ~/myapp
```

heroku create

Creating app... done, ● sleepy-meadow-81798

<https://sleepy-meadow-81798.herokuapp.com/> |

<https://git.heroku.com/sleepy-meadow-81798.git>

Check out your preferred language's getting started guide for a comprehensive introduction to deploying your first app.

Staying up to date

The Heroku CLI keeps itself and its plugins (except linked plugins) up to date automatically, *unless* you installed the Debian/**Ubuntu** package or used `npm install`.

When you run a `heroku` command, a background process checks for the latest available version of the CLI. If a new version is found, it's downloaded and stored in `~/.local/share/heroku/client`. This background check happens at most once every 4 hours.

The `heroku` binary checks for an up-to-date client in `~/.local/share/heroku/client` before using the originally installed client.

ANEXO 4

Manual de usuario

MANUAL DE USUARIO

INTRODUCCIÓN

Este manual es un documento para el usuario de la aplicación Video-Center Biblioteca virtual. La aplicación tiene como objetivo la agilización del proceso de registro de los usuarios en biblioteca.

Para el acceso a la aplicación se utiliza el enlace <https://web-rtc-israel.herokuapp.com/>.

1. Acceso a la aplicación

Para el acceso a la aplicación se utiliza el enlace <https://web-rtc-israel.herokuapp.com/>.

Al ingresar, se puede ver la página principal de la aplicación como se tiene en la figura 1.

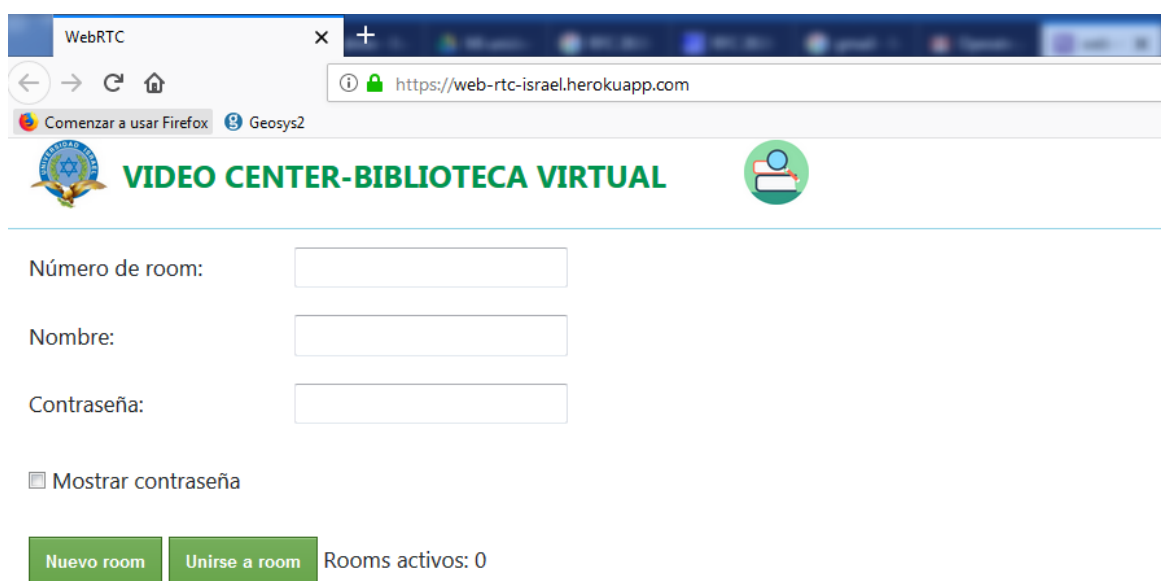


Figura 1. Página principal de la aplicación

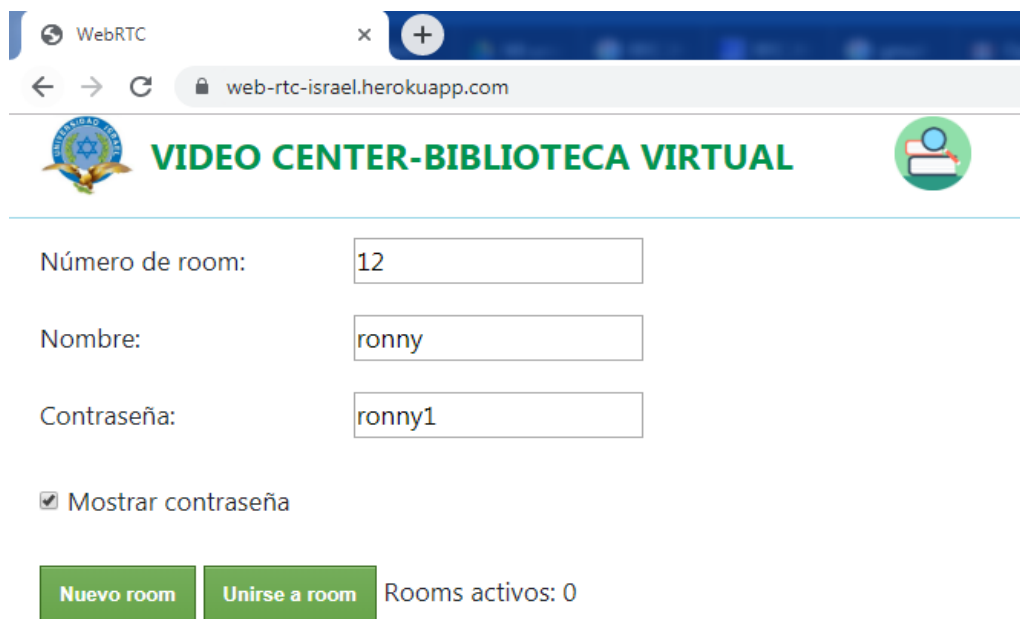
La aplicación permite dos funcionalidades en esta pantalla: creación de un nuevo room y el unirse a un room.

2. Crear un room

Para crear un room en la pantalla principal se ingresan los siguientes datos:

- número de room desde el valor 0 hasta 9999
- nombre se debe ingresar un usuario identificativo alfanumérico de hasta 20 caracteres
- contraseña del room que debe ser de 5 caracteres.

Se puede ver la contraseña para revisar que este ingresada correctamente al marcar el checkbox que dice mostrar contraseña como se puede ver en la figura 2. Si el usuario no ingresa alguno de estos campos se muestran mensajes de validación como se puede ver en la figura 3.



The screenshot shows a web browser window with the URL 'web-rtc-israel.herokuapp.com'. The page title is 'VIDEO CENTER-BIBLIOTECA VIRTUAL'. The form contains the following fields and controls:

- Número de room:
- Nombre:
- Contraseña:
- Mostrar contraseña
- Buttons: 'Nuevo room' and 'Unirse a room' (highlighted in green)
- Status: 'Rooms activos: 0'

Figura 2. Mostrar contraseña

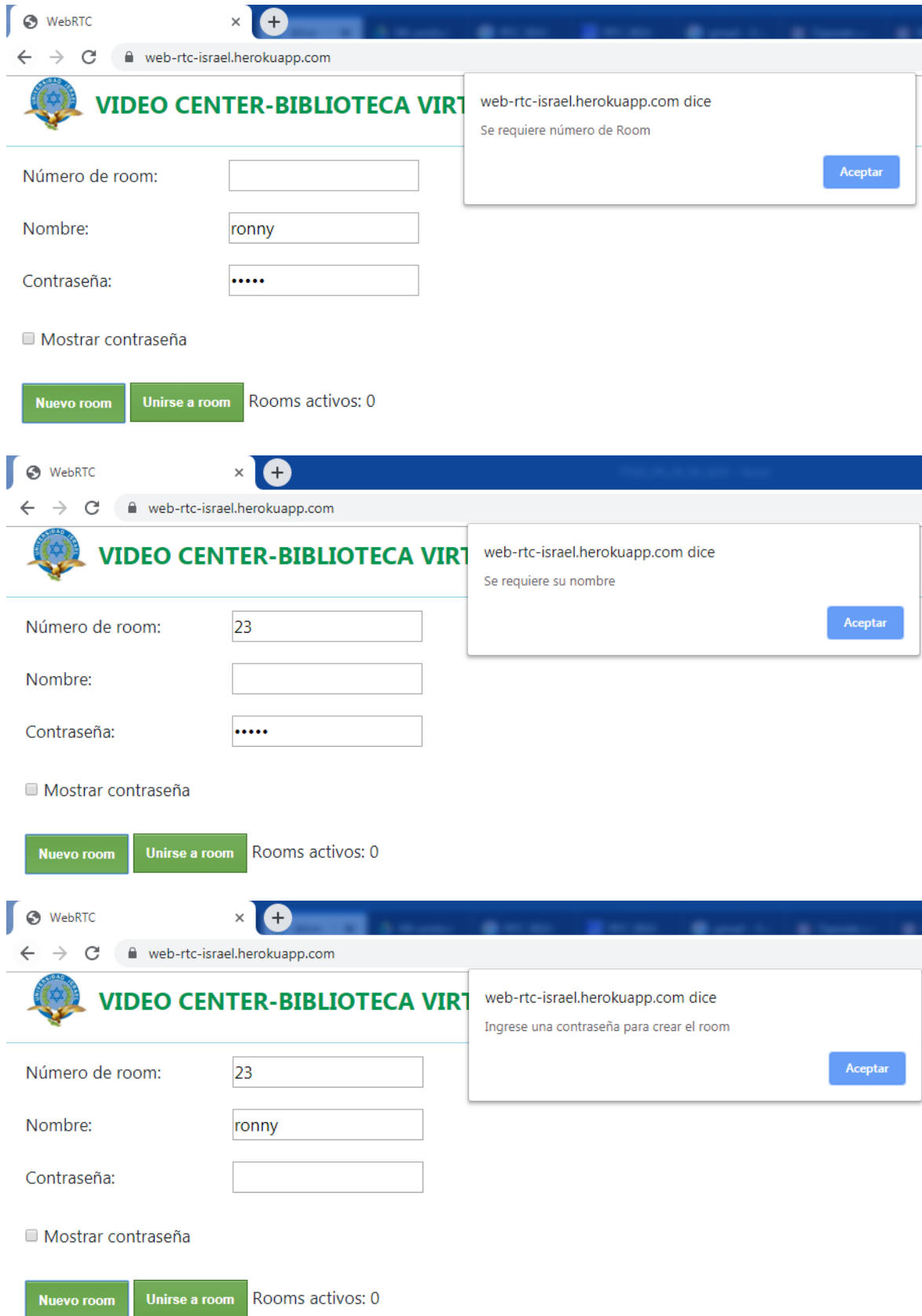


Figura 3. Mensajes de validación

Si se ingresan correctamente los datos al crear el room se despliega la pantalla de room.html como se puede ver en la figura 4.

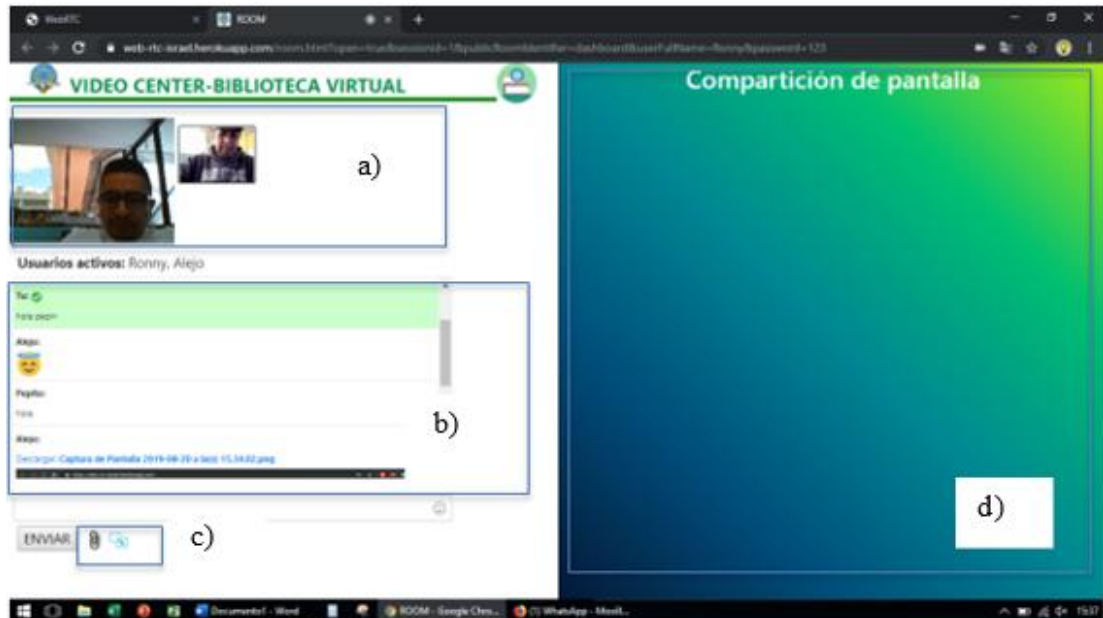


Figura 4. Vista de room.html con dos usuarios. a) Zona de videoconferencia b) Zona de mensajería c) Zona de botón de envío de archivos y compartición de pantalla d) Zona de pantalla compartida

3. Acceder a un room creado

Para ingresar al room se ingresan los siguientes datos en la pantalla principal:

- número de room desde el valor 0 hasta 9999
- nombre se debe ingresar un usuario identificativo alfanumérico de hasta 20 caracteres
- contraseña del room que debe ser de 5 caracteres.

Estas credenciales deben ser proporcionados por el creador del room mediante un correo electrónico que se envía al presionar el botón de envío de invitaciones.

4. Compartir archivos y pantalla

La compartición de archivos y pantalla, así como el envío de mensajes solo se puede ejecutar cuando al menos dos personas se encuentran en el room.

Para enviar un archivo se presiona en el botón mostrado en la figura 5, se escoge un archivo el cual se carga y se muestra en el panel de mensajes instantáneos.

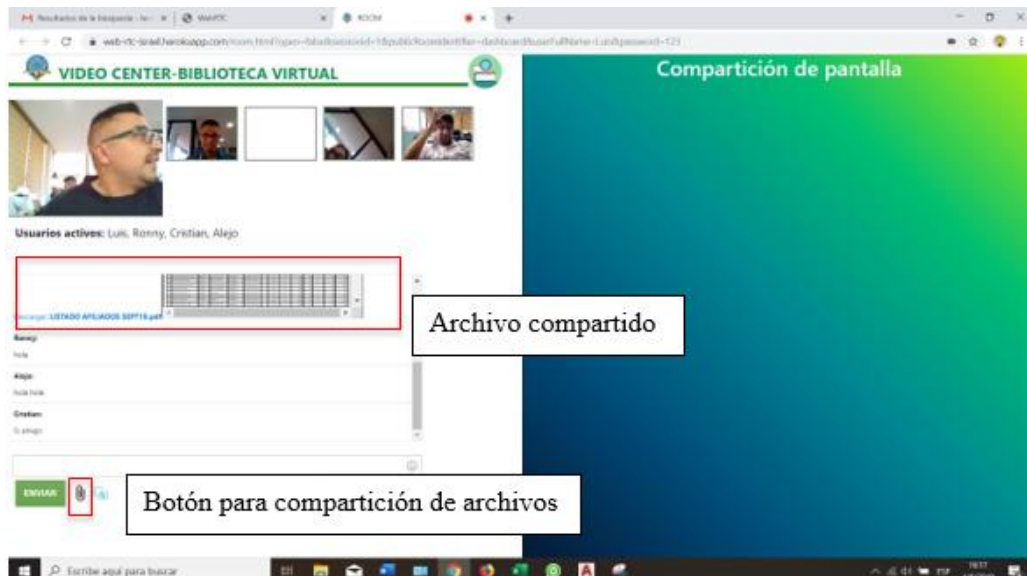


Figura 5. Compartición de archivos

Para la compartición de pantalla se presiona el botón mostrado en la figura 6, el explorador despliega un mensaje que permite seleccionar la pantalla que se quiere compartir. En el caso del creador del room la pantalla compartida se muestra en el espacio designado, si uno de los asistentes del room desea compartir pantalla se reemplaza el video de la cámara con la pantalla compartida.



Figura 6. Compartición de pantalla