MECANISMO PARA EL ENRIQUECIMIENTO DE SERVICIOS EN TIEMPO REAL SOBRE UN ENTORNO DE IPTV USANDO LA TECNOLOGÍA WEBRTC

CARLOS JULIAN DELGADO MUÑOZ

LEINER JOHAN MOSQUERA ARANDA

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telemática

Grupo de Ingeniería Telemática

Línea de Investigación en Servicios Avanzados de Telecomunicaciones

Popayán, 2017

MECANISMO PARA EL ENRIQUECIMIENTO DE SERVICIOS EN TIEMPO REAL SOBRE UN ENTORNO DE IPTV USANDO LA TECNOLOGÍA WEBRTC.

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

CARLOS JULIAN DELGADO MUÑOZ

LEINER JOHAN MOSQUERA ARANDA

Directora: Mg(C) Mary Cristina Carrascal Reyes

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telemática

Grupo de Ingeniería Telemática

Línea de Investigación en Servicios Avanzados de Telecomunicaciones

Popayán, 2017

TABLA DE CONTENIDO

LISTA DE ILUSTRACIONES

LISTA DE TABLAS

LISTA DE ACRÓNIMOS

API	*Application Programming Interface.*
CoD	*Content on Demand*
HTML5	*HyperText Markup Language, v5*
HTTP	*HyperText Transfer Protocol*
ICE	*Interactive Connectivity Establishment*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPTV	*Internet Protocol Television*
Js	*JavaScript*
JSEP	*JavaScript Session Establishment protocol*
NAT	*Network Address Translation*
OIP	*Open Internet Profile*
OIPF	*Open IPTV Forum*
OTT	*Over The Top*
P2P	*Peer To Peer*
PSTN	*Public Switched Telephone Network*
SIP	*Session Initiation Protocol*
STUN	*Session Traversal Utilities for NAT*
TCP	*Transmission Control Protocol*
TURN	*Traversal Using Relays around NAT*
UDP	*User Datagram Protocol*
VoD	*Video on Demand*
VoIP	*Voice on IP*
W3C	*World Wide Web Consortium*
WebRTC	*Web Real-Time Communication*
XHR	*XMLHttpRequest*
XMPP	*Extensible Messaging and Presence Protocol*

INTRODUCCIÓN

Actualmente existe una gran demanda sobre los servicios disponibles a través de internet (e.g., videollamadas, chats y servicios multimedia, entre otros). En promedio 2.700 millones de personas alrededor del mundo usaron internet a finales del 2013 [1], provocando que el uso de tecnologías de la información y las comunicaciones haya aumentado considerablemente en los últimos años y es estimado que estas cifras seguirán subiendo con el transcurrir del tiempo, es por ello que las empresas proveedoras de los servicios de comunicación buscan desarrollar o adoptar nuevas tecnologías que ofrezcan mayores beneficios tanto al usuario como a sí mismos.

Con la tendencia actual hacia la adopción de servicios convergentes [2], tecnologías como IPTV toman fuerza al prestar servicios de video sobre las redes de telecomunicaciones clásicas, convirtiéndose así en un rival para las empresas de TV por cable y satélite, ya que los operadores de IPTV pueden prestar una alta gama de servicios ligados a la televisión sobre redes tradicionales [3] (e.g., servicios de comunicación en tiempo real).

Los servicios en tiempo real, dadas sus características, han permitido que distintas formas de comunicación (e.g., texto, multimedia, video y voz) puedan combinarse para mejorar el nivel de productividad y rendimiento en la comunicación entre individuos, empresas, gobiernos y sociedades, además de agregar portabilidad, ubicuidad y movilidad de los servicios ofrecidos al igual que nuevas maneras de trabajo a distancia [4].

La aparición de diversas iniciativas como: WebRTC (*Web Real Time Conmunication*) y CU-RTC-WEB (*Customizable, Ubiquitous Real Time Communication over the Web*), como también la formación de algunos grupos de trabajo como ORTC (*Object Real time Communications*), da respuesta a la alta demanda de servicios de comunicación en tiempo real que puedan ser desplegados en plataformas convergentes como IPTV.

La comunicación de video en tiempo real es uno de los grandes retos para prestadores del servicio como Google y Microsoft, o para proveedores de IPTV, por ello, muchos de ellos participan en la iniciativa WebRTC, la cual presenta una opción sencilla y flexible para la implementación de este tipo de servicios.

El enriquecimiento del entorno IPTV, usando la tecnología WebRTC, permite que clientes IPTV incorporen servicios y aplicaciones con funcionalidad en tiempo real. Lograr esta integración genera numerosas ventajas, entre ellas pueden ser resaltadas: poca carga en la red, sencillez de uso para el cliente, implementación en gran variedad de dispositivos, reducción de tiempo en el desarrollo y menores costos en la implementación.

El presente proyecto de investigación plantea la inclusión de la tecnología WebRTC sobre una plataforma IPTV a través de un mecanismo software para lograr el enriquecimiento anteriormente mencionado de los servicios de comunicación en tiempo real.

Para el desarrollo de este proyecto y su organización dentro de la monografía fue usado el modelo en Cascada o modelo “Lineal secuencial”, este modelo establece que las diferentes etapas necesarias para la construcción de un producto deben desarrollarse de manera

lineal, por lo que facilita la documentación y descripción de características de una forma más fácil, en esta metodología cada etapa debe esperar la finalización de la etapa anterior, además en cada etapa es necesario establecer objetivos, tareas y actividades que la caracterizan, la escogencia de esta metodología fue realizada debido a que es una de las más simples y fáciles de entender permitiendo una buena organización y el desarrollo de proyectos que son puedan ser fácilmente duplicados en el futuro [5], las etapas escogidas del modelo en Cascada para el desarrollo del actual proyecto de investigación con el cual los objetivos planteados pudieron ser cumplidos son presentadas a continuación (Ver figura 1).

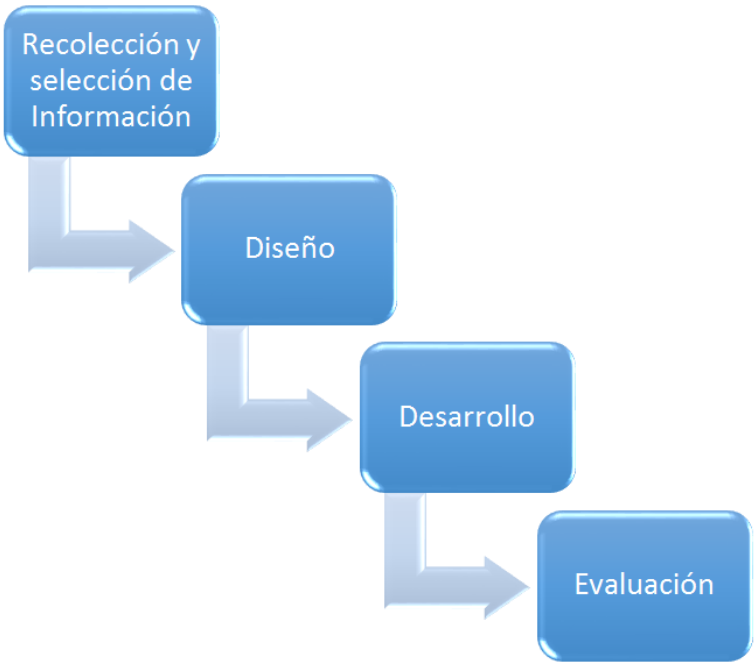

Figura 1: Etapas escogidas modelo en cascada

Cada una de las fases fue desarrollada de la siguiente manera:

- Recolección y selección de la información: esto fue llevado a cabo durante la construcción del anteproyecto y en el primer capítulo del actual documento, aunque esta fase es transversal a todo el trabajo de grado.
- Diseño: esta fase tiene énfasis en la selección de componentes y características para el diseño del mecanismo propuesto el cual es mostrado en el capítulo II.
- Desarrollo: esta fase es la de construcción, es en donde se especifican los parámetros y tecnologías usadas para la implementación y ejecución del mecanismo. Esta fase se desarrolla en el capítulo III.
- Evaluación: aquí son ejecutadas las pruebas de funcionalidad sobre el sistema solución construido y las pruebas de desempeño al mecanismo propuesto, esta fase es mostrada en el capítulo IV.

CAPÍTULO I

GENERALIDADES DE WEBRTC E IPTV

1.1. INTRODUCCIÓN

Para el desarrollo de la presente investigación fue necesario establecer una base conceptual con la cual realizar la mejor selección de características del mecanismo a diseñar. Este capítulo condensa la información utilizada durante las fases de diseño e implementación, dividiéndola en dos grandes núcleos temáticos: descripción de la tecnología WebRTC y ambientación del entorno IPTV.

1.2. CARACTERISTICAS GENERALES WEBRTC

Desde su creación las comunicaciones en tiempo real han sido costosas, complejas y privadas, los primeros intentos por construir aplicaciones interactivas con la mayor facilidad posible dependieron de redes, hardware y software personalizado especiales que frecuentemente tenían altos costos y baja calidad, es por ello que en los últimos años han surgido plataformas para el desarrollo de servicios [6]; el estándar WebRTC permite acceder a este tipo de comunicaciones, trayendo códigos de fuente libre y permitiendo a los desarrolladores elaborar sistemas usando lenguajes y estándares comunes como HTML5 y JavaScript.

WebRTC es un estándar que extiende el modelo de navegación web [7]. Provee a los navegadores y aplicaciones móviles capacidades de comunicación en tiempo real sobre una arquitectura punto a punto [8], posibilitando el intercambio directo de contenido multimedia.

WebRTC define una *API*[1] que permite la ejecución de aplicaciones web en tiempo real sobre dispositivos con acceso a internet, usando de manera segura las entradas periféricas como cámaras web y micrófonos [7]; es importante resaltar que WebRTC no es solo una *API*, sino también un conjunto de protocolos [10], estas dos especificaciones tienen como objetivo: proveer un entorno donde las aplicaciones puedan ser ejecutadas en los navegadores que las soportan [11], por tanto, WebRTC está basada en estándares definidos por el IETF (*Internet Engineering Task Force*) para los protocolos que son usados por las funciones de comunicación en tiempo real (RTC) del navegador y en la *API* desarrollada por el W3C (*World Wide Consortium*) requerida por las aplicaciones web para interactuar con dicha función RTC mencionada anteriormente [12] [13].

Ahora bien, gracias a WebRTC proveer aplicaciones de audio y video sobre la Web es mucho más fácil, además de proporcionar, en comparación con tecnologías anteriores como VoIP usando Asterisk por ejemplo, una mejor experiencia de usuario y posibilitar el desarrollo de servicios de comunicación a un nivel más alto, ya que es posible cumplir con casos de uso que siempre habían requerido ser satisfechos. En cuanto a los proveedores de servicios, reciben beneficios en aspectos como reducción de costos, facilidad, seguridad, una solución simple en todas las plataformas, la integración a dispositivos y un rápido posicionamiento en el mercado [12], sumado a esto, WebRTC es libre en todos los

[1]**Interfaz de programación de aplicaciones:** conjunto de funciones y procedimientos que son ofrecidos para su uso por otro software, permite la comunicación entre componentes software [9].

aspectos ya que cuenta con una licencia software BSD (*Berkeley Software Distribution*), lo que permite a los desarrolladores incluir la tecnología en aplicaciones de gran escala donde el número de clientes es grande con costos de acceso muy bajos con muy pocas restricciones, permitiendo así el uso de código tanto de fuentes privadas como libres [14].

Aunque los servicios de comunicación ofrecidos por WebRTC tradicionalmente son puestos en marcha en navegadores web, usados como puntos terminales, existen alternativas para utilizar WebRTC en otros dispositivos tales como *Set Top Box (STB)*, *Video Room Systems* y *dongles*[2] como *ChromeCast*[3], ya que cualquier dispositivo o componente que necesite ser capaz de procesar, enviar y recibir multimedia puede lograrlo con WebRTC embebido dentro del dispositivo.

Existen dos principales razones por las cuales seleccionar WebRTC para los dispositivos mencionados anteriormente [17]:

- WebRTC tiene una licencia *Open Source*, es decir, es de fuente libre, haciéndola fácil de usar como base para componentes de procesamiento multimedia.
- El uso de WebRTC en *STB*, *Dongles*, *ChromeCast* facilita la interacción de estos con otro tipo de dispositivos como celulares, computadores portátiles y navegadores.

Debido a que fue diseñado de manera tan genérica, el estándar WebRTC puede ser integrado con sistemas de comunicación ya existentes tales como Voz sobre IP (VoIP), diversos clientes SIP, e incluso la Red Telefónica Conmutada (PSTN) entre otras, lo que abre grandes posibilidades de implementación en diversos entornos.

1.2.1. Funciones de la API WebRTC [7 [18]

La W3C en su documento "*WebRTC 1.0: Real-time Communication Between Browsers*" [19], define aspectos que los servicios en tiempo real[4], como por ejemplo el servicio de videollamada, deben considerar; para el establecimiento de una comunicación, entre las cuales destacan: la captura del flujo multimedia y la conexión entre puntos remotos, esta última es realizada usando tecnologías NAT-Transversal como ICE, STUN y TURN que cumplen las siguientes funciones:

- Envío de rutas establecidas localmente a puntos remotos y recepción de rutas de dichos puntos.
- Envío de datos directamente, es decir, sin servidor intermedio a puntos remotos.

La *API* WebRTC fue diseñada tomando en consideración tres conceptos principales, los cuales son descritos a continuación:

---

[2] Dongle: Un dispositivo hardware unido a un computador sin el cual un software no puede funcionar, usado especialmente para prevenir uso no autorizado, puede ser conectado en un puerto USB para permitir el acceso inalámbrico de un computador a un aparato externo Wi-Fi, como un teléfono móvil, o a internet a través de ancho de banda de alta velocidad [15].

[3] Chromecast: es un dispositivo de transmisión de contenido multimedia del tamaño de un pulgar que se conecta al puerto HDMI de tu televisor. Solo necesitas un teléfono o tablet Android, un iPhone, un iPad, un portátil Mac o Windows o un Chromebook para enviar tus aplicaciones y contenido favoritos directamente a tu TV [16].

[4] Servicio en tiempo real: Es un modelo de telecomunicaciones que permite a los usuarios el intercambio de información instantáneamente, es decir con latencia despreciable [20].

1.2.1.1. MediaStream:

Representa un flujo de datos de audio y video, por lo tanto, para conseguir los datos multimedia se habilita la captura de dispositivos periféricos tales como cámara y micrófono, dando acceso a ellos sin la necesidad de *plugins* tales como A*dobe Flash* o *Microsoft Silverlight*. Cabe notar que esto debe llevarse a cabo una vez el sistema haya preguntado al usuario por los permisos de uso y acceso para así evitar problemas de privacidad, este componente también debe contener una entrada y una salida, la entrada viene desde los dispositivos periféricos y la salida puede ser una etiqueta video para insertar y mostrar el contenido de manera local, o a través de la conexión P2P para que en un punto remoto pueda ser visualizado.

1.2.1.2. RTCPeerConnection:

En una red P2P, '*Peers*' o puntos hace referencia a sistemas computacionales los cuales están conectados mutuamente a través de internet, cualquier archivo o dato puede ser compartido entre estos sistemas sin hacer uso de un servidor central [21], además cada máquina en la red tiene el rol tanto de cliente como de servidor. Un *RTCPeerConnection* es un componente de WebRTC que permite a los usuarios comunicarse directamente navegador a navegador, estableciendo una comunicación estable y eficiente entre los puntos. "Las comunicaciones son coordinadas a través de un canal de señalización, el cual es proporcionado por medios no especificados, pero generalmente usando *XMLHttpRequest* (*XHR*) o *WebSockets* o algún otro protocolo" [19].

En algunos textos este componente es descrito como el corazón de las conexiones punto a punto entre cada uno de los clientes WebRTC [17], debido a que es el que permite establecer la comunicación entre dichos puntos directamente. Además, es el responsable de manejar el ciclo de vida completo para cada conexión punto a punto.
A continuación, es explicado a grandes rasgos el proceso ejecutado por esta *API*:

Conexión de Usuarios.
Cada usuario accede al servicio a través de la URL de la aplicación y el sistema le muestra quienes además de él están actualmente conectados. Para realizar la conexión es necesario asignar un identificador único a cada usuario, este identificador representa el nombre de una '*Room*' que puede ser interpretado como un número telefónico o un *Id* de conversación. La primera función que debe ejecutar la API *RTCPeerConnection* es identificar cada navegador por la '*Room*' del usuario que está utilizándolo y conectar a cada uno con el servidor de señalización.

Búsqueda de Pares.
Otra tarea delegada a *RTCPeerConnection* es encontrar la forma de conectar dos puntos (clientes), para esto, cada navegador es relacionado con una ruta de acceso directo a un puerto y una interfaz de red en el equipo terminal donde se encuentra alojado, esto es necesario ya que por lo general cada navegador está situado detrás de un dispositivo de red como un enrutador que puede estar usando el mecanismo NAT (*Network Address Translation*) para conectar la red local a internet lo que le impide al navegador ser encontrado de manera directa desde una red externa.

El mecanismo NAT traduce las IP privadas de la red a una IP pública para que esta pueda enviar y recibir paquetes por fuera de la red local: su objetivo principal es simular el uso de una red con direcciones diferentes a las reales, por lo que estos dispositivos pueden

también imponer restricciones en el Firewall [22] que bloquean ciertos puertos y conexiones entrantes.

Encontrar una manera de conectar a través de esos tipos de enrutadores es comúnmente conocido como '*NAT Traversal'*, un modo común para lograr esto es usar el servidor *'Session Traversal Utilities for NAT'* (STUN) [23], este suministra el medio para que un punto terminal pueda determinar la dirección IP y el puerto asignado por una NAT, que como fue mencionado corresponde a su dirección IP y puerto privados. Es importante notar que actualmente en la red existen distintos proveedores de servidores STUN que pueden ser usados en la implementación de sistemas de comunicación. Ahora bien, si este servidor no puede encontrar el modo para conectarse a otro navegador, no hay otra opción que usar un tipo de solución que retransmite los datos multimedia, esta solución es el servidor '*Traversal Using Relay NAT'* (TURN) [24], este protocolo es una extensión de STUN, su modo de funcionamiento está basado en la organización y transmisión de paquetes a puntos determinados través de él, la desventaja presentada al usar este método es el retorno a una arquitectura cliente-servidor, aunque en redes estrictamente seguras este método puede llegar a ser la única opción.

Debido a que cada una de las técnicas mencionadas tienen pros y contras, ya que pueden ser óptimas en ciertas topologías de red y fallar en otras, la decisión de usar un protocolo u otro es tomada por el *Framework* ICE [25] (*Interactive Connectivity Establishment*) que basado en el análisis de diversos parámetros define el mejor método en una situación determinada.

1.2.1.3. RTCDataChannel:

Este componente negocia la transmisión de datos entre los clientes, es decir, permite a los navegadores enviar datos a través de una conexión punto a punto, en otras palabras, facilita el intercambio de datos de manera bidireccional entre los puntos. Cabe resaltar que esta subrutina es configurable de dos maneras posibles; la primera es tener una entrega confiable o parcialmente confiable de mensajes, es decir, similar a TCP y la segunda es la entrega de mensajes enviados en orden o fuera de orden, en otras palabras, de un modo menos seguro similar a UDP. De lo anterior es posible deducir que el modo menos confiable tiene como ventaja que no presenta sobrecarga y permite un funcionamiento más ágil, pero sin la certeza ni la garantía de una entrega segura, así mismo el modo más confiable puede presentar una sobrecarga mayor y puede llegar a ser más lento lo que lo hace menos idóneo para implementaciones en tiempo real, sin embargo, es posible configurar la cantidad de retransmisiones o el tiempo límite.

Por otro lado, es importante notar que la creación de un canal de datos puede darse de dos modos: 1) Dejar que WebRTC haga el trabajo, cree el transporte y lo anuncie al punto remoto con una notificación. 2) Escribir código propietario con el fin de notificar al punto remoto la necesidad de conectarse a un nuevo canal [26].

1.2.2. Mecanismos de señalización WebRTC [27][10]

La señalización es el proceso de coordinación de la comunicación, es un mecanismo en el cual los puntos intercambian mensajes de control con el fin de negociar protocolos, canales y medios de comunicación, es una parte esencial en el desarrollo de cualquier aplicación interactiva que requiera de alguna conexión con aplicaciones remotas, por ejemplo: chats,

juegos etc. En la especificación de la tecnología WebRTC no es definido el proceso de señalización debido a que busca maximizar la compatibilidad con otras tecnologías, por lo tanto es mejor permitir que el desarrollador seleccione la tecnología de red y los protocolos de mensajería apropiados, dándole así flexibilidad a su implementación.

Para el proceso de señalización, WebRTC puede hacer uso de variados mecanismos tales como *HTTP/REST*, *JSON* vía *XHR, JSON via WebSocket SIP (Session Initiation Protocol), XMPP (Extensible Messanging and Presence Protocol); Websockets* y *SIP* sobre *WebSockets* [28].

1.2.3. Arquitectura

La tecnología WebRTC combina el modelo cliente-servidor con el paradigma de comunicación punto a punto (P2P) concibiendo su modelo arquitectónico imitando la propuesta realizada por la IETF para el trapezoide de comunicaciones en tiempo real para un navegador [7].

En el más común de los escenarios para las aplicaciones WebRTC, cada terminal de usuario ejecuta la aplicación descargada previamente desde un mismo servidor web, reduciendo el modelo de trapezoide a una forma triangular como es mostrado en la *figura 2* [11].

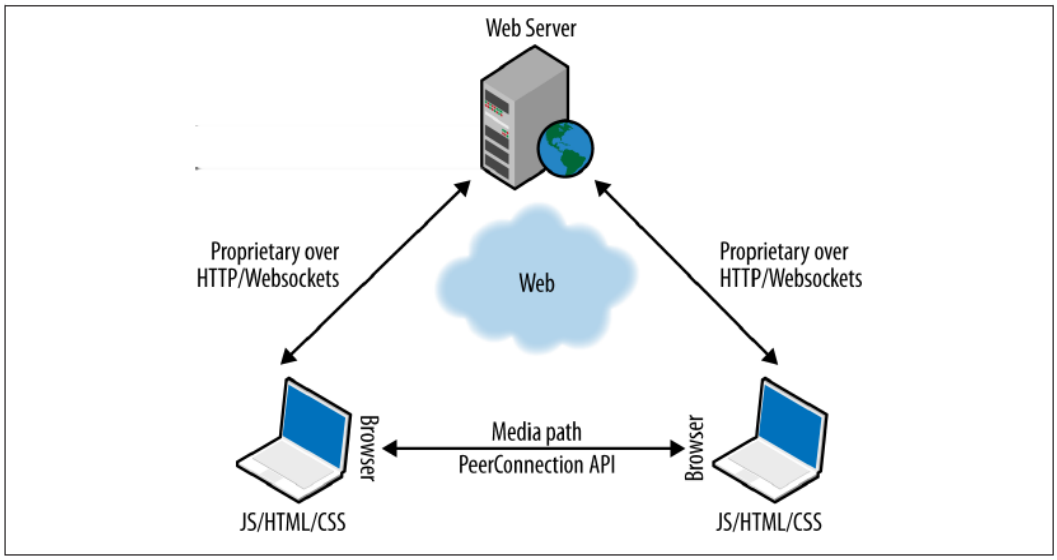

Figura 2: Triangulo WebRTC.

1.2.4. Proceso de ejecución de una aplicación WebRTC

A pesar de que la comunicación entre clientes es punto a punto en este tipo de aplicaciones, siempre es necesario establecer la configuración de la comunicación antes de cualquier otro proceso, por lo general esta configuración es realizada por un servidor Web o de señalización, realizando un intercambio de mensajes cliente-servidor con el objetivo de ajustar detalles de contacto y negociar la sesión para la comunicación, es decir, definir cómo los clientes se comunicarán. Después de ser negociada la configuración de la

comunicación, finalmente es posible establecer la trasmisión directa punto a punto de los datos multimedia entre clientes.

Para llevar a cabo lo anterior, WebRTC ejecuta una serie de procesos que permiten la configuración y el posterior establecimiento de la comunicación entre clientes. A continuación, se hará una breve explicación de estos procesos:

1.2.4.1. Conectar Usuarios

El primer paso en este proceso es que los usuarios puedan conectarse entre sí de algún modo, para esto acceden al servicio web a través de una página, una vez el servidor identifica el usuario, RTCPeerConnection conecta al navegador con el servidor de señalización tal como fue mencionado en la sección *1.2.1.2* donde se describe la función de conexión de usuarios que realiza la *API* RTCPeerConnection. Este proceso es llevado a cabo entre la aplicación cliente, el servidor web y el servidor de señalización [29].

1.2.4.2. Solicitud de periféricos

Después de identificado el usuario, es solicitado el acceso a los periféricos del dispositivo, esto es llevado a cabo usando la *API* MediaStream la cual solicita permisos al usuario para que el navegador pueda hacer uso de estas características [7].

1.2.4.3. Comienzo de señalización

Una vez ambos usuarios hayan compartido algunos identificadores ellos pueden intercambiar mensajes de señalización para negociar la configuración de su conexión WebRTC. En este contexto los mensajes de señalización son simplemente una forma de comunicación que ayuda a los puntos a establecer y controlar la comunicación.

1.2.4.4. Encontrar candidatos [22]

En el siguiente paso los navegadores intercambian información acerca de sus redes, y cómo ellos pueden contactarse el uno al otro, este proceso es comúnmente descrito en textos como “Encontrar candidatos”. En este paso vuelve a ponerse en funcionamiento RTCPeerConnection con su tarea Búsqueda de Pares descrita en la sección *1.2.1.2* del actual capítulo. El objetivo de este paso es poder establecer un camino de comunicación directo entre los usuarios evitando el uso de servidores.

1.2.4.5. Negociar sesiones multimedia

Ahora que ambos navegadores saben cómo hablarse, deben negociar el tipo y formato multimedia; estos intercambiarán características como *codecs*[5], resolución del contenido multimedia, así como también la tasa de bits, esto es usualmente negociado usando un modelo basado en oferta y respuesta construido sobre el protocolo SDP[6], que permite describir la información de las sesiones de comunicación multimedia, este modelo ha sido definido como JSEP (*JavaScript Session Establishment Protocol*) [31],además estas negociaciones son llevadas a cabo a través de un canal de datos creado con *RTCDataChannel* y el cual es asociado al *PeerConnection* del paso anterior.

[5] **Códec**: es utilizado para comprimir un archivo, para que ocupe el menor espacio posible, y descomprimirlo cuando tiene que ser reproducido [30].

[6] **SDP:** protocolo de descripción de sesiones, usado para describir los parámetros de inicialización de los flujos multimedia.

1.2.4.6. Inicio de transmisión

Al finalizar los pasos anteriores, los navegadores están en la capacidad de transmitir el contenido multimedia a través de su conexión *peer-to-peer*, los mensajes de señalización que pudiesen llegar a transmitirse durante el transcurso de la comunicación serán enviados a través del servidor de señalización común entre los puntos.

En las *figura 3* es posible apreciar los pasos anteriormente descritos organizados de manera secuencial, su objetivo es mostrar al lector la relación que existe entre los procesos y la arquitectura de las aplicaciones WebRTC, exponiendo el orden y las entidades involucradas en la ejecución de cada proceso.

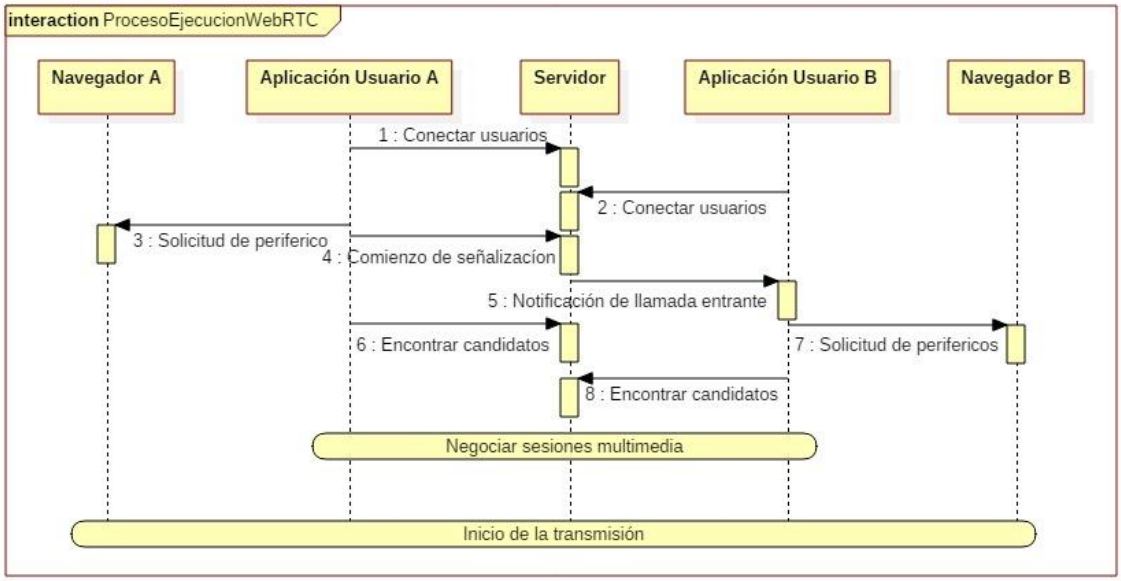

Figura 3: Diagrama de secuencia ejecución WebRTC

1.3. CARACTERÍSTICAS GENERALES IPTV

El "*Open IPTV Forum*" contempla la prestación de servicios de IPTV tanto en redes gestionadas como en no gestionadas [32]. En general, IPTV es un conjunto de servicios multimedia (televisión, video, audio, texto, gráficos y datos) distribuido por una red IP. Sobre una red gestionada debe poseer calidad de servicio, seguridad, interactividad y fiabilidad. Sobre una red no gestionada no es asegurado calidad de servicio" [33].

1.3.1. Perfiles de IPTV según Open IPTV Forum (OIPF)

En [34] son definidos tres perfiles para IPTV, con el objetivo de suministrar la mejor capacidad y flexibilidad posible a los proveedores para el despliegue de servicios. A continuación, es realizada una breve descripción de cada uno de ellos:

- **Perfil de Internet Abierto (OIP):** destinado para servicios OTT (*Over The Top*) que no usa ninguna provisión de QoS o administración de características en los terminales.

- **Perfil básico de Administración (BMP):** adiciona soporte para servicios de entrega de contenido *multicast*[7] y *unicast*[8] incluyendo todas las características asociadas que faciliten la provisión de QoS para la distribución de contenido en redes gestionadas.
- **Perfil de Administración Mejorado (EMP):** adiciona soporte para la administración avanzada de redes gestionadas, así como también características como las definidas para IMS.

1.3.2. Categorías de servicios IPTV

En la segunda publicación del Open IPTV Forum (OIPF) son descritas algunas categorías de servicios para IPTV y las características que deberán tener, esto es aplicado tanto para redes gestionadas como para modelos de internet abierto como es el perfil OIP [36]. A continuación, son presentadas algunas de estas categorías:

- Servicio de contenido programado.
- Contenido bajo Demanda.
- Grabador de video personal.
- Guía de Contenido.
- Servicio de notificación.
- Integración con servicios de comunicación.
- Acceso a la Web.
- Servicio de Información.
- Aplicaciones interactivas.

1.3.3. Requerimientos definidos por el Open IPTV Forum

En el documento "*Service and Platform Requirements*" [37] el OIPF define algunos requerimientos obligatorios que debe tener una solución de IPTV, estos requerimientos son definidos tanto para redes gestionadas como no gestionadas; además también existen algunos requerimientos opcionales y recomendaciones.

1.3.3.1. Generales

1. La solución IPTV deberá basarse en interfaces y estándares abiertos.
2. Los servicios definidos por OIPF deberán estar disponibles en varios dispositivos finales, que son adecuados para el consumo del servicio. La lista de dispositivos deberá incluir:
   - TV
   - PC
   - Teléfonos móviles

   NOTA: La experiencia de usuario y servicios disponibles pueden variar dependiendo de la capacidad de los dispositivos.

[7] **Multicast**: hace referencia a un método de direccionamiento de red en las cuales la fuente transmite un paquete a múltiples destinos simultáneamente [35].

[8] **Unicast**: hace referencia al envío de paquetes o información desde un único emisor hacia un único receptor [35].

3. Los servicios definidos por OIPF deben ser accedidos sobre redes fijas y móviles que sean adecuadas para la entrega del servicio, cabe notar que la QoE[9] del usuario puede variar dependiendo de las características de las redes mencionadas.

1.3.3.2. Navegación por el servicio y contenidos

1. La solución deberá habilitar portales para ser accedidos a través de internet.
2. La solución debe contar con GC[10] que provean información de los contenidos disponibles en VoD.

1.3.3.3. Despliegue y ejecución de la aplicación

1. La solución deberá incluir un componente de presentación que permita el despliegue de las aplicaciones en navegadores.
2. La solución debe ser capaz de: a) posicionar el contenido de video sobre la pantalla; b) permitir el control de la reproducción, c) permitir el inicio o terminación de una sesión de VoD.

1.3.3.4. Seguridad

El OIPF define para la seguridad de la solución los siguientes aspectos a tener en cuenta:

1. Control de acceso:
   - La solución de IPTV debe restringir el acceso a la aplicación con base a un grupo de políticas establecidas por el proveedor de servicios de IPTV.
2. Autenticación de usuario
   - La solución debe soportar un mecanismo de inicio de sesión único que proteja la privacidad de los usuarios a través de los servicios de IPTV.
   - La solución debe soportar un único mecanismo de inicio de sesión que permita que un servicio de IPTV solicite información acerca del usuario.
   - La solución IPTV debe soportar un mecanismo de inicio de sesión único para el acceso a múltiples servicios al mismo tiempo.
   - La solución debe soportar la capacidad, a través de un mecanismo apropiado como por ejemplo nombre de usuario/contraseña, para distinguir y autenticar usuarios individuales.
   - Los usuarios podrán adquirir, acceder y consumir contenido a través de internet abierto, sin la participación de un proveedor de plataforma de servicio.

1.3.3.5. Perfiles

La solución IPTV debe poseer mecanismos para que los usuarios puedan añadir, eliminar y modificar su perfil.

[9] QoE: calidad de experiencia: calidad del proceso de comunicación según es percibido por el usuario [38].

[10] **GC:** *Content Guides* o Guías de Contenido: usadas para describir contenido multimedia

CAPÍTULO II

DISEÑO DEL MECANISMO PROPUESTO

2.1. INTRODUCCIÓN

Como resultado del análisis realizado, en este capítulo es mostrado el proceso llevado a cabo para el diseño del mecanismo propuesto y el sistema que lo pondrá en marcha, así como también de la toma de decisiones hechas con el objetivo de conseguir un enriquecimiento en los servicios de comunicación en tiempo real del entorno IPTV.

2.2. METODOLOGÍA DE REFERENCIA.

Para la descripción del mecanismo propuesto en el contexto del trabajo de grado, es tomado como referencia las recomendaciones realizadas por la IEEE en su publicación "*Recommended Practice for Architectural Description of Software-Intensive Systems*" [39], de donde son consideradas tres partes fundamentales para realizar la descripción:

1. La selección y exposición de las características propias de los entornos donde esta arquitectura habitará, primero para el entorno WebRTC y en seguida para el entorno IPTV, estas características son tomadas como requisitos que el sistema debe cumplir.
2. Partiendo de los resultados obtenidos en el punto anterior, en los siguientes pasos en la metodología son descritas las características que deberán contener tanto el sistema como el mecanismo propuesto para poderse implementar en el entorno deseado.
3. Para finalizar, la arquitectura propuesta es descrita usando el modelo 4+1 de Phlllppe B. Krucht [40], con las siguientes vistas "Vista Lógica", "Vista de Desarrollo", "Vista de Procesos", "Vista Física" y "Vista de Escenario"; las cuales serán abordadas en mayor detalle posteriorment

La metodología anterior fue utilizada como base conceptual, aplicada inicialmente al diseño del mecanismo propuesto y posteriormente para describir los servicios de videollamada y VoD. La metodología es escogida debido al enfoque que esta tiene hacia la influencia del software en la arquitectura de los sistemas de software intensivos, garantizando de esta manera un diseño donde la tecnología WebRTC es de gran relevancia para la arquitectura del sistema solución.

Durante la fase de diseño se toma en consideración el patrón modelo-vista-controlador (MVC), para llevar a cabo la descomposición modular del sistema solución, garantizando con esto una arquitectura ordenada y bien definida, su selección está basada en la habilidad que este modelo posee para ser aplicado a subsistemas aislados[11] sección del sistema solución.sección del sistema solución.

2.3. DESCRIPCIÓN Y SELECCIÓN DE CARACTERÍSTICAS WEBRTC

2.3.1. Cliente WebRTC

Para la entrega de aplicaciones de comunicación en tiempo real es necesario definir un conjunto de funcionalidades en la aplicación cliente; tales como los motores encargados de soportar/procesar audio y video, además de los protocolos establecidos en las especificaciones para la capa de transporte, es por ello que son escogidos los navegadores como cliente debido a que suplen en gran parte estas funcionalidades, así mismo para

[11] Bucanek, J. (2009). Learn Objective-C for Java developers. 1st ed. [Berkeley, Calif.]: Apress, pp.353-402.

asegurar una interoperabilidad entre las distintas funciones de tiempo real la IETF trabaja en la selección de criterios mínimos para el soporte de los códecs de audio y video [7] en dichos navegadores. Por lo que a continuación, es realizada una comparación entre algunos navegadores enfatizando en que características soportan de la tecnología WebRTC. Adicionalmente en la siguiente imagen (Ver figura 4) son resumidos los elementos soportados por cada uno, donde el color verde representa las API soportadas, el color rojo los componentes no soportados, y el color amarillo son las partes de WebRTC que se encuentran bajo desarrollo o parcialmente desarrolladas.

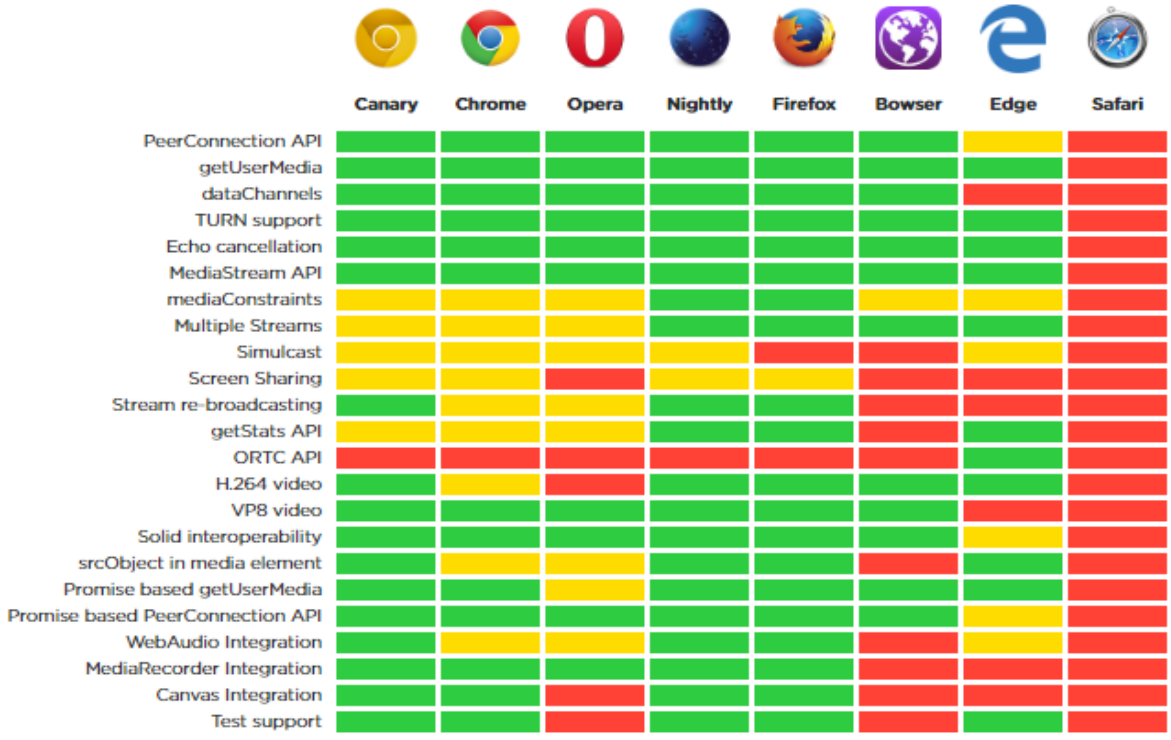

Figura 3: Soporte elementos WebRTC [41]

Como conclusión de la comparación realizada y mostrada en la figura 4 es posible inferir que hasta la fecha la mayoría de navegadores están trabajando en la implementación de los elementos propios de la tecnología llegando a soportar ya más del 50% de las partes.

Con el fin de lograr enriquecer las aplicaciones multimedia en tiempo real sobre la web y satisfacer las funcionalidades expuestas en la figura 4, en las especificaciones fue definida una arquitectura en la cual tanto los desarrolladores web, como los constructores de navegadores puedan basar sus implementaciones.

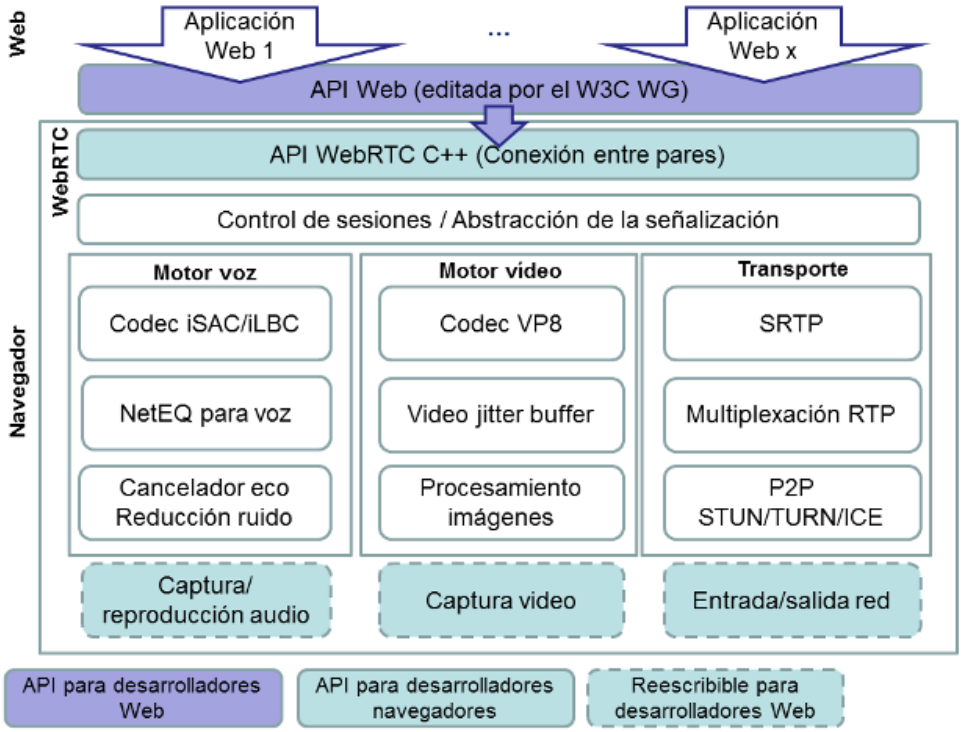

Figura 4: Arquitectura Api WebRTC [42] [43]

En la figura 5, de color verde y azul, se puede evidenciar las características y protocolos usados por un navegador para soportar la tecnología WebRTC permitiendo así la implementación de aplicaciones mediante el uso de JavaScript de forma sencilla sin la necesidad de conocer a fondo los requerimientos para el establecimiento de una comunicación, dicha API implementada por los desarrolladores web va incrustada en el recuadro de nivel superior.

2.3.2. Señalización [44] [45]

Como fue mencionado en la sección 1.2.2, la señalización fue dejada por fuera de las especificaciones de WebRTC, aunque existen variadas opciones de protocolos de transporte y mecanismos de señalización que acepta WebRTC.

La conexión entre un cliente y el servidor de señalización es denominado canal de señalización [46], parte esencial de una aplicación ya que permite tener un continuo intercambio de datos entre partes; un canal de señalización requiere de un protocolo de trasporte que indique el modo en el cual pueden comunicarse las dos partes, independiente del mensaje o la estructura del mismo, y un mecanismo de señalización que defina los tipos de mensajes, su significado y la estructura que tendrán.

A continuación, son mostradas las características más relevantes de los protocolos de transporte y sus correspondientes mecanismos de señalización:

2.3.2.1. Comet/XHR/SSE [44]

Estos protocolos de transporte son considerados los enfoques clásicos para la señalización web, dichos modelos están basados en el protocolo HTTP para posibilitar que un servidor web pueda enviar mensajes a sus clientes, algo que es totalmente necesario para negociar o establecer comunicación entre puntos; cada cliente envía una solicitud al servidor, el cual la mantiene presente hasta cuando haya datos disponibles para ser enviados al cliente y el proceso es repetido nuevamente. Sin embargo, existe una gran limitante con estos protocolos: su escalabilidad[12], ya que su uso implica el consumo de una gran cantidad de recursos del lado del servidor, por ende, habrá menos conexiones hacia él y los costos tendrán una tendencia a elevarse, sumado a esto el desarrollador debe definir sus propios mensajes de señalización.

2.3.2.2. WebSockets

Websockets representan la nueva evolución de los protocolos comet (Ver 2.3.2.1), estos definen una conexión *Socket full-duplex* para el envío y recepción de información entre el cliente y el servidor por lo que da solución a los problemas que presenta Comet de portabilidad y conexión [47]. Cada *WebSocket* suministra una conexión persistente entre cliente y servidor para enviar datos en cualquier momento y en ambas direcciones simultáneamente, lo que permite la creación de aplicación en tiempo real con mayor facilidad, además el uso de estas evitará una sobrecarga adicional en el establecimiento de nuevos enlaces TCP/IP para cada solicitud del cliente [48]. Para usar este protocolo es necesario tener en cuenta que el navegador debe soportarlo, así como también servidores web y *proxies*; cabe notar que para este protocolo de transporte se requiere definir mensajes de señalización, o adoptar un mecanismo de señalización estándar como los siguientes:

2.3.2.2.1. Objetos JSON

Es un formato de intercambio de datos basado en un subconjunto del lenguaje JavaScript útil para escribir cualquier tipo de aplicación basada en JavaScript, como por ejemplo páginas web. JSON está compuesto por dos estructuras: una colección de pares clave-valor conocido también como objeto o diccionario y una lista ordenada de valores también conocida como arreglos, vectores o listas; JSON es independiente del lenguaje de programación que lo utilice convirtiéndolo en un fuerte candidato al trabajar con diversas tecnologías [49].

2.3.2.2.2. SIP sobre WebSockets [50]

La ventaja de esta técnica de conexión es que no aporta tráfico en la red cuando no es usada, es decir, solamente es activada cuando es requerido enviar solicitudes. Este mecanismo es recomendable cuando es requerido conectar un *Backend* de telefonía como los servidores tipo *Asterisk* buscando satisfacer algún caso de uso que implemente llamadas a PSTN, de lo contrario no es muy usado debido a que es necesaria una infraestructura SIP ya implementada, además para aplicaciones móviles es realmente complicado mantener la conexión siempre disponible, como otra desventaja es la pérdida de conexión eventual que puede acarrear la pérdida de mensajes de notificación, debido a

[12] Escalabilidad: capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa el número de usuarios del mismo.

esto los sistemas requieren de maneras para detectar fallos y restablecer las sesiones evitando así pérdida de datos.

2.3.2.2.3. XMPP/Jingle

Es similar a SIP sobre WebSockets, con la diferencia que este usa otro protocolo estándar llamado XMPP. Cuando es escogida esta opción es necesario tener algún tipo de instalación XMPP y conocer acerca de ella. XMPP es usado para mensajería instantánea, chat grupal, llamadas de voz y video, esto lo vuelve una buena opción para ser usado junto a WebRTC, además su *Framework Jingle* para el establecimiento de sesiones P2P, interactua muy bien con los objetivos de comunicación punto a punto de la tecnología WebRTC [51] [52].

Tomando en consideración los mecanismos de señalización, los protocolos de transporte y las consideraciones técnicas de cada uno de estos, los cuales fueron descritas anteriormente, para esta investigación fue adoptado JSON para los mensajes de señalización y WebSockets como protocolo de transporte, sumado a esto, existen herramientas que permiten la migración automática en tiempo de ejecución a mecanismos COMET en caso tal de que el protocolo WebSockets no esté disponible como por ejemplo la herramienta híbrida socket.io [53] [54] que facilita el desarrollo de aplicaciones que implementan JSON sobre WebSockets y adiciona nuevas ventajas a las aplicaciones basadas en WebRTC.

2.4. DESCRIPCIÓN Y SELECCIÓN DE CARACTERÍSTICAS DEL ENTORNO IPTV

2.4.1. Perfiles de IPTV según Open IPTV Forum

Dado el análisis de las características del proyecto, es posible inferir que el Perfil de Internet Abierto (OIP) (ver capítulo 1 apartado *1.3.1)* es suficiente para lograr los objetivos propuestos y satisfacer los casos de estudio desarrollados en este proyecto.

Los servicios del OIP son accedidos desde Internet sin ninguna garantía de QoS y típicamente desde plataformas de servicio como portales Web, además permite la compatibilidad de servicios que no ofrecen garantías de calidad durante todo el recorrido entre el prestador de servicios y el terminal de usuario, es decir, de modo OTT [34]. Si se tienen en cuenta estas características favorecen el objetivo del mecanismo, ya que permiten la construcción de aplicaciones web que enriquezcan los servicios de comunicación del entorno IPTV.

2.4.1.1. Servicios de IPTV para OIP

Para el proyecto se acogen las directrices o requerimientos establecidos por el OIPF para servicios de IPTV en el que indican que:

- OIP debe admitir el uso de servicios de contenido programado, usando el método de transporte HTTP.
- OIP debe soportar el servicio de CoD con el método de transporte HTTP.

De lo anterior es importante resaltar la necesidad de trabajar con el método de transporte HTTP, el cual será clave para la construcción del sistema.

2.4.1.2. Métodos de autenticación para OIP

OIP debe soportar métodos de autenticación como HTTP básico y autenticación implícita. En el primer método, las credenciales son enviadas codificadas en Base64, esta tiene como ventaja que forma parte de las especificaciones de HTTP, pero puede ser fácilmente descifrada; el segundo método proporciona una mejora en términos de seguridad debido a que transmite usando un método de cifrado. En ambos casos se asume un alias y una contraseña, los cuales son conocidos por el usuario de IPTV y el proveedor de plataforma [55].

En la siguiente imagen es ilustrado el proceso de autenticación HTTP sugerido para el perfil abierto de IPTV:

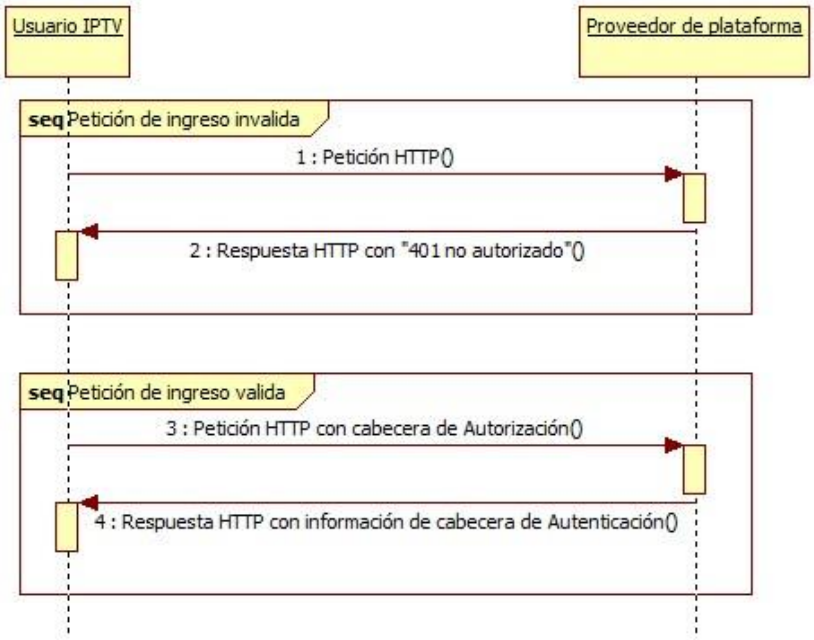

Figura 5: Secuencia de acceso servicio web

En la figura 6 existen dos secuencias diferentes, la primera describe el intento por acceder a un servicio web llevado a cabo por un usuario sin autorización, al cual le es negado el acceso con un mensaje “401 no autorizado”; la segunda secuencia muestra a un usuario registrado previamente enviar sus credenciales al proveedor de plataforma para poder acceder a un servicio web, el proveedor devuelve como resultado un mensaje de confirmación de acceso.

2.4.2. Arquitectura IPTV según OIPF

En [56], el Open IPTV Fórum define una arquitectura de alto nivel que encierra los dominios previamente mencionados y es definida a través de un conjunto de entidades usadas para el perfil abierto de internet, estas entidades sirvieron como referencia para el diseño de la arquitectura. A continuación, será dada una breve descripción de ellas, fueron

seleccionadas aquellas que se adaptan a las condiciones de la investigación y se encuentran dentro del marco del perfil escogido en la sección 2.4.1.

1. **Autenticación de Acceso al Servicio:** esta entidad funcional es la responsable de la protección del acceso al servicio y la autenticación, los usuarios son identificados con parámetros previamente definidos como el nombre de usuario y su correspondiente contraseña.
2. **Aplicaciones IPTV:** esta entidad incluye todo lo relacionado con la lógica de los servicios y aplicaciones de IPTV, tales como CoD (contenido bajo demanda) seleccionado para el contexto de este proyecto.
3. **Base de Datos del usuario:** la base de datos central de los perfiles de suscripción es administrada por el proveedor de plataforma del servicio, los campos incluidos en cada base de datos pueden variar, los más comunes son datos relacionados con la autenticación de usuario.
4. **Función de Gestión de Contenido:** esta entidad maneja el ciclo de vida del contenido, es decir, es responsable del almacenamiento del servicio y de los contenidos.
   En el Anexo A está explicado en mayor detalle la descripción de los dominios y la arquitectura de IPTV definidos por el OIPF y la ITU.

2.4.3. Categorías de servicios de IPTV

Para mostrar algunas características del enriquecimiento de los servicios en tiempo real en un entorno IPTV, fueron seleccionados dos servicios del entorno tomando como referencia las categorías mencionadas en el apartado *1.3.2*, los cuales son: “Contenido bajo demanda” e “Integración con servicios de comunicación”.

Para la categoría “Contenido bajo demanda” fue seleccionado el servicio de Video bajo demanda o VoD, esta elección es realizada por ser uno de los servicios más populares en IPTV y con características de funcionamiento apropiadas para mostrar la interacción a nivel de interfaz con los servicios enriquecidos por WebRTC, además de ser una categoría de servicios propia del perfil de IPTV escogido en la sección *2.2.1.* A continuación, son descritas las características generales de la categoría “Contenido bajo demanda” y algunas recomendaciones que posteriormente son tomadas como requerimientos para la construcción del servicio.

2.4.3.1. Contenido bajo demanda (CoD)

Servicio de IPTV donde el usuario puede seleccionar individualmente el contenido que desea ver de una lista disponible, el contenido es consumido por petición del usuario y transmitido vía *unicast* desde el servidor. Específicamente el consumo de video en el servicio de VoD puede usar uno de estos tres métodos para su transmisión: “*Monomedia*”, “*Multiplex format*” y “*Streaming*”; cada uno con sus respectivos formatos [57]. En el presente proyecto de investigación es seleccionado el formato MP4 (especificado en ISO/IEC 14496-14), usado en el método “*Multiplex format*”, ya que es comúnmente usado en la actualidad para la distribución de contenido [57] [58] y permite su uso sobre dispositivos con recursos limitados como es el caso de algunos clientes IPTV.

El Open IPTV Forum, además de definir los requerimientos generales expuesto en la sección *1.3.3*, también consigna en su documento “*Service and Platform Requirements*” [59] requisitos para la puesta en marcha de un servicio de CoD en un entorno IPTV, estos requisitos son tenidos en cuenta para la construcción del sistema donde será evaluado el mecanismo propuesto, a continuación, es enumerado lo sugerido por el OIPF [60]:

- Requerimientos comunes:
  1. La solución IPTV deberá soportar los servicios CoD
  2. La solución IPTV deberá soportar el acceso libre para ver CoD.
  3. La solución IPTV deberá soportar acceso a CoD basada en suscripción.
  4. La solución IPTV deberá proveer un mecanismo para reanudar la reproducción en un punto específico.
  5. La solución IPTV deberá proveer un mecanismo para reanudar la reproducción en el mismo o algún otro ITF[13] donde el usuario tiene registrado.
- Requerimientos para el envío de CoD:
  1. La solución IPTV deberá soportar el consumo directo de la distribución de CoD de los servidores de contenido.
  2. La solución IPTV deberá soportar la distribución de CoD a modo de descarga progresiva.

2.4.3.2. Integración con servicios de comunicación [59]

Proporciona a los usuarios IPTV el acceso a servicios de comunicación entre clientes que pueden ser integrados con otros servicios de IPTV otorgando una experiencia más rica, algunos ejemplos de los servicios de comunicación son:

- Identificación llamada entrada.
- Presencia.
- Mensajería.
- Chat.
- Telefonía de voz y video.

La categoría "Integración con servicios de comunicación" busca facilitar la implementación de servicios de comunicación en tiempo real para los usuarios del entorno IPTV. Es por ello que fue seleccionado este tipo de servicios considerando el enfoque de WebRTC hacia servicios en tiempo real como videoconferencia, mensajería instantánea, compartición de escritorio, entre otros. Por lo tanto es escogido el servicio de videollamada para la implementación y pruebas del mecanismo, debido a su popularidad y mejor adaptación a los clientes IPTV, en comparación con los demás servicios.

Cabe aclarar que la definición técnica de comunicación en tiempo real se basa en la recomendación de la ITU –T [G.-114] donde se define los tiempos requeridos para tener un servicio en tiempo real, es decir, un tiempo aceptable de retardo en una vía el cual es de hasta 400 milisegundos.

A continuación, son consignados algunos requisitos propuestos por el Open IPTV Forum para la implementación de servicios de comunicación, específicamente para la Telefonía de Voz y Video, el cual es semejante al servicio de videollamada [59]. Estos requisitos son usados posteriormente para la construcción del sistema donde es puesto a prueba el mecanismo propuesto. La solución IPTV debe:

1. Soportar la habilidad para que un usuario pueda interactuar con aplicaciones que provean el establecimiento y gestión de llamadas de voz.

[13] ITF: *IPTV Terminal Function*, funcionalidad dentro de la red del consumidor o móvil terminal.

2. Tener la habilidad para que un usuario pueda interactuar con aplicaciones que provean establecimiento y control de llamadas telefónicas de video.
3. Proveer un mecanismo para que el usuario pueda cambiar de una ITF basada en llamada de voz a una ITF basada en videollamada y viceversa.
4. Suministrar un mecanismo para la gestión de los dispositivos de captura de audio y video.
5. Mostrar la cámara de video local siendo vista en una ventana previa.

A continuación, es presentado el resultado de la ejecución del segundo paso de la metodología usada para la descripción del sistema, en este son obtenidas las características que deberán tener los componentes del sistema para cumplir con los requisitos planteados anteriormente. El sistema está dividido en cuatro partes: la primera y más importante el **MECANISMO PROPUESTO**, en esta es condensado el principal aporte del trabajo de grado; posteriormente se tiene el **ENTORNO IPTV** y el **SERVICIO DE VIDEOLLAMADA PROPUESTO**, usados para poner en marcha el mecanismo y destacar sus cualidades.

2.5. MECANISMO PROPUESTO

Tomando en consideración los atributos que otorga la tecnología WebRTC e IPTV es caracterizado el mecanismo propuesto, el cual permite el funcionamiento de los servicios en tiempo real basados en WebRTC y posibilita su operación en entornos de IPTV, aportando a los servicios de comunicación ventajas intrínsecas de la tecnología WebRTC, como la comunicación punto a punto o la omisión de *plugins* para la prestación de servicios, además, el mecanismo utiliza estándares abiertos que permiten enmarcar los servicios de comunicación en un entorno IPTV, satisfaciendo los requerimientos exigidos para esta categoría de servicios.

2.5.1. Caracterización del mecanismo

El objetivo del mecanismo es permitir el enriquecimiento y la implementación de servicios en tiempo real basados en WebRTC que puedan caber dentro de la categoría de servicios de comunicación de IPTV. Este mecanismo sirve como un intermediario entre ambas tecnologías permitiendo la inclusión de una dentro de la otra a nivel lógico y de manejo de datos y así maximizar la prestación de servicios en IPTV trayendo beneficios tanto al prestador como al usuario.
Con base a los requerimientos establecidos para los servicios de comunicación y las características entregadas por la tecnología WebRTC se proponen los siguientes atributos para el mecanismo:

1. RNF1: el mecanismo ejecuta la solicitud de los periféricos disponibles en el dispositivo del usuario final.
2. RNF2: el mecanismo establece canales de comunicación punto a punto (conforme a lo establecido por la API WebRTC) con otros usuarios conectados y su posterior cierre al terminar el proceso de comunicación.
3. RNF3: las funciones provistas por el mecanismo deben ser ejecutadas dentro de páginas HTML que sirvan como interfaz para el cliente, con el fin de permitir una interacción entre servicios a nivel de presentación.

4. RNF4: establece mensajes propietarios para identificar el estado de la comunicación, los cuales posibilitan el control de la misma.
5. RNF5: el mecanismo conmuta mensajes de control entre los usuarios conectados al servicio.
6. RNF6: adiciona al perfil de usuario la clave “room” que representa el identificador único para los servicios de comunicación WebRTC.
7. RNF7: adiciona al perfil de usuario la clave “state” la cual es el estado de conexión o desconexión al servicio.

2.5.2. Modularización del mecanismo propuesto

Realizando una agrupación de las funciones anteriormente descritas, es construido el siguiente diagrama (ver figura 7) donde es posible evidenciar la asociación lógica entre grupos de funciones.

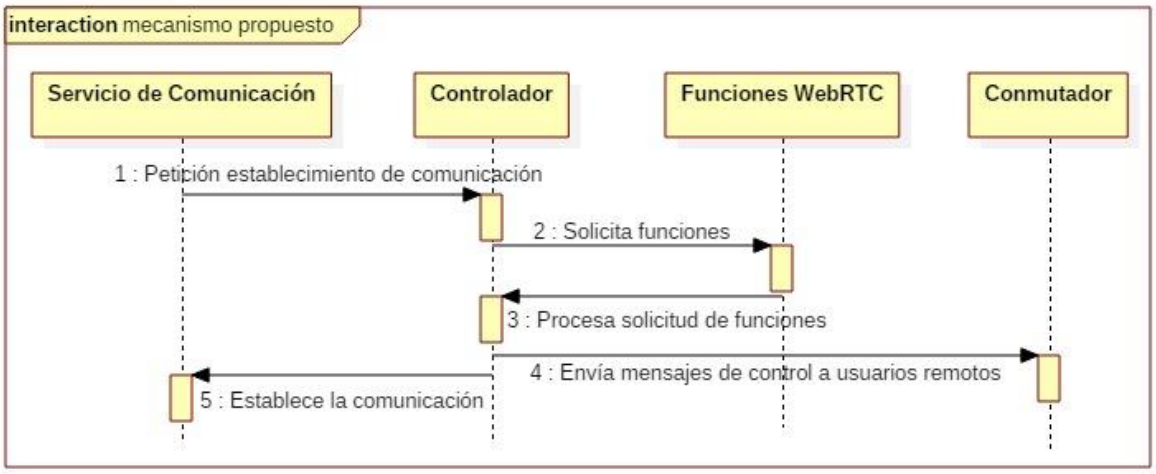

Figura 6: Módulos del mecanismo

Bloques:

a) Funciones WebRTC

Este módulo es el encargado de hacer posible el inicio de una comunicación, en él es realizada la captura multimedia y la creación del canal de datos por el cual es enviado el flujo de información durante la ejecución del servicio

b) Conmutador

En este módulo fue realizada la implementación de un servidor Node JS, encargado de la recepción, control y envío de los mensajes de notificación a cada usuario que inicia una comunicación a través de una Videollamada. Estos mensajes permiten al proveedor de servicio controlar e identificar el estado de las comunicaciones establecidas, posibilitando así, una mejor gestión de los usuarios y un enriquecimiento en los servicios en tiempo real de WebRTC.

c) Controlador

En este módulo se realiza la gestión centralizada del software, es decir, es el corazón de la solución que realiza la toma de decisiones dependiendo de las acciones realizadas por un usuario conectado al sistema.

d) Servicio de Comunicación

Este bloque representa el servicio de Videollamada implementado como caso de estudio en el sistema para poner en marcha todos los módulos del mecanismo.

2.6. ENTORNO IPTV PROPUESTO

El entorno IPTV sobre el cual es puesto en funcionamiento el mecanismo propuesto, provee los servicios necesarios para resaltar las cualidades del mismo como es el enriquecimiento en los servicios de comunicación en tiempo real y la posibilidad de interactuar con otros servicios dentro del entorno; es por eso que son seleccionados los servicios de Video bajo Demanda y videollamada expuestos en las secciones *2.4.3.1* y *2.4.3.2* respectivamente. Por otro lado, una de las características más atractivas de IPTV es la gestión de usuarios y servicios, esta solventa una debilidad de WebRTC, ya que este último no provee herramientas para la administración de sus aplicaciones, por lo tanto, la **Gestión centralizada** expuesta a continuación (ver sección *2.6.1*) no solo cumple con los requisitos establecidos para el entorno IPTV, sino que refleja una mejoría en las aplicaciones de comunicación en tiempo real desarrolladas usando WebRTC.

2.6.1. Gestión centralizada

La gestión centralizada del sistema toma en consideración requisitos del perfil abierto de IPTV, como los métodos de autenticación por HTTP para la identificación de los actores ante el sistema, tal como fue explicado en la sección *2.4.1.2*, resaltando que para el presente trabajo se consideraron dos tipos de actores: administrador y usuario.

Para permitir la entrada de aplicaciones en tiempo real que usen WebRTC, los prestadores de servicios IPTV no tendrían que realizar mayores cambios en sus sistemas de gestión, solamente sería necesario ampliar la información del perfil de usuario introduciendo una variable que almacene el estado de conectividad a la aplicación WebRTC y un identificador único para el establecimiento de la comunicación con el usuario.

Tomando en consideración los requisitos generales y de gestión del perfil abierto de IPTV definidos por el OIPF y expuestos en la sección *1.3.3* y *2.4.1* respectivamente, junto con las adiciones necesarias para la implementación de un servicio de videollamada con WebRTC en el entorno IPTV, se realiza la siguiente caracterización de la gestión centralizada del sistema:

1. El sistema debe permitir a los usuarios la creación de un perfil que servirá para su identificación ante el sistema en futuras ocasiones.
2. La identificación de usuarios debe ser llevada a cabo usando el protocolo clásico de autenticación descrito en la sección 2.2.1.2.
3. Por defecto cada usuario identificado por la plataforma tendrá un estado de inactividad para el servicio de videollamada.
4. Todos los usuarios identificados ante el sistema pueden acceder a los contenidos provistos por el servicio de VoD.
5. Durante el registro del perfil de usuario es asignado un identificador único construido con el alias de usuario y un número aleatorio, este identificador será útil para el establecimiento de la comunicación entre usuarios con el servicio de videollamada.
6. El acceso al sistema es controlado, dependiendo si es un usuario o un administrador, se permitirán acciones acordes con cada uno de sus roles en el sistema.

7. La autenticación requiere de dos variables, nombre de usuario y contraseña, estas variables son proporcionales al sistema durante el registro del usuario.
8. Las variables de autenticación son almacenadas en la base de datos al momento del registro, estos datos son protegidos usando algoritmos de cifrado.

2.6.2. Caracterización del servicio de Video bajo Demanda propuesto como caso de estudio

Como fue dicho anteriormente, el servicio de VoD propuesto se construye con el objetivo de probar el mecanismo y apreciar la forma en cómo son integrados los servicios de comunicación como videollamadas al entorno IPTV. El tener contenido multimedia que puede pausarse y reanudar su reproducción permite la inclusión del servicio de videollamada sin correr el riesgo de que al usuario no le sea posible ver parte del contenido multimedia; como sucedería en servicios '*Live Stream'*, donde no es posible controlar el flujo multimedia.

Para la caracterización del servicio VoD en un entorno IPTV, se toman en consideración algunos requisitos generales definidos por el OIPF para la estructuración de un entorno IPTV, junto con características de la categoría "Contenido bajo Demanda" de IPTV. A continuación, se enumeran las características asignadas para este servicio:

1. Solo los usuarios registrados en la base de datos del sistema podrán acceder al servicio de VoD.
2. El sistema provee al usuario la posibilidad de seleccionar el contenido que desea consumir dentro de la lista de posibilidades mostrada en pantalla.
3. Los videos dispuestos para el servicio de VoD son listados con vistas miniatura del contenido e información relevante del video.
4. Cada video que es presentado al usuario contiene información general como un título y una breve descripción de su contenido, e información técnica como duración, formato y calidad.
5. Son proveídos controles de reproducción para el video seleccionado dando al usuario la posibilidad de pausar, reanudar, adelantar o atrasar el video.
6. El formato de los videos usados para el servicio es MP4.
7. Cada usuario podrá reanudar la reproducción del video consumido desde cualquier cliente con acceso a internet, la reanudación considerará el tiempo de reproducción que llevaba el video hasta el cierre de la página donde es reproducido.

2.6.3. Modularización del entorno IPTV y su servicio VoD

En la figura 8 es presentada la agrupación de funciones para llevar a cabo la implementación del servicio VoD en un entorno IPTV usando el mecanismo propuesto.

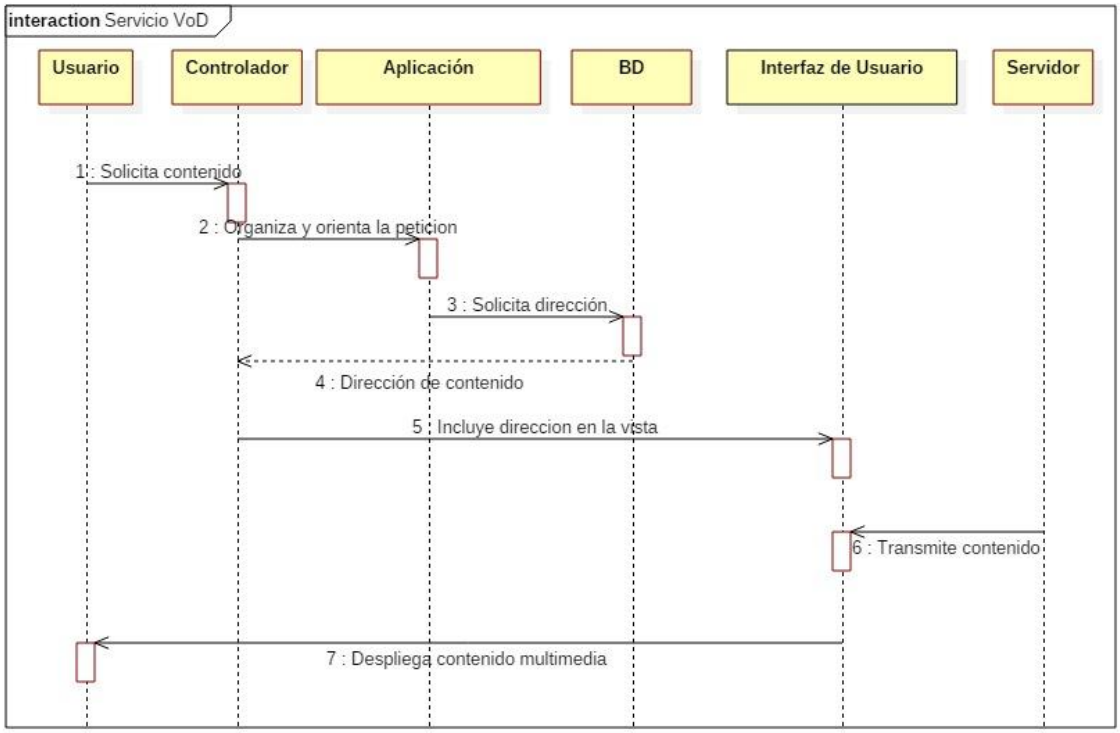

Figura 7: Módulos entorno IPTV

La base de datos almacena la información de cada contenido, entre las cuales se encuentra la dirección de alojamiento en el servidor de contenidos.

1. **Solicita contenido multimedia**

El terminal del usuario realiza la petición del contenido multimedia al controlador, dependiendo de que video el usuario desee mirar.

2. **Organiza y orienta petición**

El controlador recibe peticiones del usuario desde el navegador web, este la procesa y solicita la respuesta a la función correspondiente.

3. **Solicita dirección de contenido multimedia**

Dentro de la lógica de la aplicación, están los métodos encargados de hacer la consulta a la base de datos de cada contenido, la dirección de estos contenidos es almacenada en una lista para que puedan ser accedidos cuando sea requerido.

4. **Responde con dirección contenido multimedia**

Una vez obtenida la información de la base de datos, son armadas las direcciones completamente, adicionando la IP del servidor de contenidos a la ruta del archivo que el usuario ha solicitado.

5. **Incluye dirección de contenido multimedia en la vista**

El controlador una vez conseguida la dirección correspondiente llama al archivo XHTML y le pasa lo obtenido para que sea desplegado.

6. **Trasmite contenido multimedia**

El servidor de contenidos envía el video seleccionado a la página XHTML con la cual es respondida la petición del usuario.

7. **Despliegue contenido multimedia**

El navegador cliente interpreta el archivo XHTML con la respuesta del sistema permitiendo que el usuario puede disfrutar del contenido solicitado.

2.6.4. Cumplimiento de los requisitos generales establecidos por el OIPF

En la *tabla 1* son condensados los requisitos descritos en la sección *1.3.3*, los cuales son cubiertos por las características seleccionadas para el diseño del entorno IPTV; en la tabla es apreciable la agrupación de estos requisitos según la categoría a la cual pertenecen.

Tabla 1 Requisitos generales OIPF

CATEGORÍA	REQUISITOS
Generales	La solución IPTV debe basarse en interfaces y estándares abiertos.
	Los servicios definidos por OIPF deben estar disponibles en varios dispositivos finales, que son adecuados para el consumo del servicio. La lista de dispositivos deberá incluir: • TV • PC • Teléfonos móviles
	Los servicios definidos por OIPF deben ser accedidos sobre redes fijas y móviles que sean adecuadas para la entrega del servicio.
Navegación por el servicio y los contenidos	La solución deberá habilitar portales para ser accedidos a través de internet.
	La solución debe proveer GC que provean información de los contenidos disponibles en VoD, dentro del contexto de este trabajo fueron especificados: el título, descripción, duración y calidad del contenido.
	La solución debe incluir un componente de presentación que permita el despliegue de las aplicaciones en navegadores.
Despliegue y ejecución de la aplicación	La solución debe ser capaz de: • Posicionar el contenido de video sobre la pantalla. • Permitir el control de la reproducción. • Permitir el inicio o terminación de una sesión de VoD.
Seguridad	Autenticación de usuario
Perfiles	La solución IPTV debe poseer mecanismos para que los usuarios puedan añadir, eliminar y modificar su perfil.

2.6.5. Cumplimiento de los requisitos establecidos por el OIPF para el servicio VoD de IPTV

Al igual que para la *tabla 1*, en la *tabla 2* son condensados los requisitos que fueron incluidos en el diseño realizado para el servicio VoD, en este caso solo fue considerada la categoría "**Contenido bajo demanda**".

Tabla 2 Requisitos para VoD OIPF

CATEGORÍA	REQUISITOS
Contenido bajo demanda: Video bajo demanda	La solución IPTV deberá soportar los servicios CoD
	La solución IPTV deberá soportar acceso a CoD basada en suscripción.
	La solución IPTV deberá proveer un mecanismo para reanudar la reproducción en un punto específico.
	La solución IPTV deberá proveer un mecanismo para reanudar la reproducción en el mismo o algún otro ITF donde el usuario tiene registrado.
	La solución IPTV deberá soportar el consumo directo de la distribución de CoD de los servidores de contenido.
	La solución IPTV deberá soportar la distribución de CoD a modo de descarga progresiva.

2.7. SERVICIO DE VIDEO LLAMADA PROPUESTO

El servicio básico de videollamada diseñado bajo los criterios presentados por el OIPF, es construido con el fin de observar como un servicio de comunicación en tiempo real basado en WebRTC puede caber dentro de la categoría de servicios de comunicación y a su vez interactuar con otros servicios del entorno de IPTV, demostrando que al utilizar la tecnología WebRTC es posible mejorar y adicionar características a los servicios de comunicación propios del entorno IPTV.

2.7.1. Caracterización del Servicio de videollamada propuesto

Tomando en consideración las características de la tecnología WebRTC, las propiedades intrínsecas del entorno IPTV seleccionado y los requisitos para la puesta en marcha de servicios de comunicación en este entorno, son puntualizadas las siguientes funciones para el servicio de videollamada:

1. El acceso al servicio de videollamada será concedido únicamente a usuarios registrados, estos deberán autenticarse antes de poder acceder al servicio.
2. Durante el consumo de servicios de IPTV como VoD, el sistema le permite al usuario cambiar su estado entre "activo" e "inactivo" en el servicio de videollamada.
3. La aplicación permite al usuario iniciar, finalizar y responder videollamadas.
4. Para el inicio de una videollamada el usuario puede visualizar que usuarios están disponibles. Solo es posible establecer una comunicación si ambos usuarios están activos para el servicio.
5. En una página previa al establecimiento de la llamada, el usuario del servicio puede visualizar la captura de video de la cámara local.

2.7.2. Modularización del servicio de videollamada usado como caso de estudio

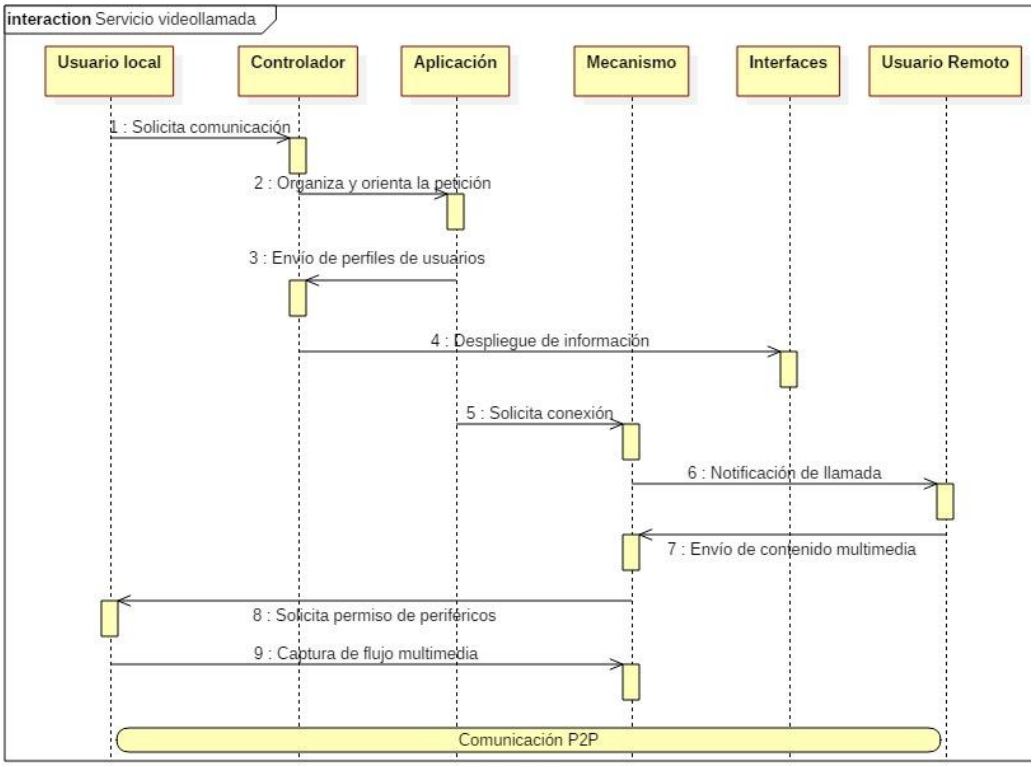

Figura 8: Módulos servicio videollamada

1. **Solicita establecimiento de comunicación**

El usuario a través de la aplicación web desarrollada puede visualizar a otros usuarios conectados, además puede elegir llamar a quien desee.

2. **Organiza y orienta petición.**

El controlador recibe peticiones del usuario desde el navegador web, este las procesa y solicita la respuesta a las funciones correspondientes.

3. **Envía información usuario local y remoto**

El sistema consulta en la base de datos los perfiles de usuario y los tiene a disposición cuando estos se requieran, en este caso, cuando se solicita el establecimiento de una videollamada es cargada la información tanto del usuario local, quien llama, como del usuario remoto, quien es llamado.

4. **Incluye información de los usuarios dentro de contenido XHTML**

Despliega información del usuario local y remoto en las interfaces como sus nombres e incluye datos dentro del código JavaScript para ejecutar efectivamente el proceso de comunicación.

5. **Solicita establecimiento de comunicación**

Dentro del sistema solución es ejecutada la lógica encargada de invocar funciones del mecanismo que, permiten establecer la comunicación entre usuarios. Como primer paso está el envío de notificación de llamada entrante al usuario remoto para que este decida si contestar o rechazarla, mientras esto sucede, la API WebRTC se prepara para iniciar la comunicación.

6. **Usuario Remoto**

- **6.1: Notifica solicitud de comunicación**
  Es enviada una notificación al usuario remoto indicándole que el usuario local desea establecer una comunicación, por lo que tiene dos opciones: contestar o rechazar, si el consumidor rechaza la llamada entonces puede seguir disfrutando de la aplicación.
- **6.2: Envía contenido multimedia usuario remoto**
  Si el usuario remoto acepta la llamada entrante y permita el uso de sus dispositivos periféricos, se envía el contenido multimedia para que sea desplegado en la interfaz correctamente y se visualice usando contenedores HTML.

7. **Usuario Local**

- **7.1: Solicita permiso de periféricos**
  Cuando un usuario local inicia una llamada, es invocada la función encargada de hacer la gestión de los dispositivos periféricos, en ella se le pregunta al usuario si da acceso a estos. Si el usuario no permite el uso de dispositivos periféricos entonces la llamada se cancela o se despliega solamente con audio o video dependiendo si comparte su cámara web o su micrófono.
- **7.2: Captura flujo multimedia de periféricos**
  Si un usuario acepta dar permiso a sus dispositivos periféricos, inicia a la captura de video y/o audio.

8. **Envía flujo multimedia de comunicación**

Una vez el usuario remoto ha aceptado la comunicación y ambos usuarios otorguen permisos al sistema sobre sus dispositivos, el contenido multimedia como audio y video es transmitido directamente entre ellos y se agrega en la interfaz web desarrollada para el servicio de videollamada.

9. **Despliega contenido multimedia.**

Tanto el usuario local como el usuario remoto visualizan el contenido multimedia de audio y video, transmitido en el paso anterior. En este punto es posible decir que la llamada ha sido establecida satisfactoriamente.

2.7.3. Cumplimiento de los requisitos establecidos por el OIPF para el servicio de videollamada

En la *tabla 3* son consignados los requisitos cubiertos por el servicio de videollamada caracterizado anteriormente. Del entorno IPTV descrito en la sección *2.4.* es tomada la categoría "**Integración con servicios de comunicación**", de la cual son extraidos los siguientes requisitos:

Tabla 3 Requisitos para servicio Video-llamada OIPF [60]

CATEGORÍA	REQUISITOS
Integración con servicios de comunicación: Videollamada	La solución IPTV debe soportar la habilidad para que un usuario pueda interactuar con aplicaciones que provean el establecimiento y gestión de llamadas de voz.
	La solución de IPTV debe soportar la habilidad para que un usuario pueda interactuar con aplicaciones que provean establecimiento y control de llamadas telefónicas de video.

	La solución IPTV deberá proveer un mecanismo para la gestión de los dispositivos de captura de audio y video.
	La solución IPTV debe soportar la cámara de video local siendo vista en una ventana previa.

2.8. DESCRIPCIÓN DE LA ARQUITECTURA PROPUESTA PARA EL SISTEMA

Como se ha venido mencionando, es de suma importancia para cualquier sistema definir una arquitectura en la cual evidenciar las características y la estructura para la construcción precisa del sistema caracterizado; tener dicha arquitectura posibilita comprender las restricciones del sistema, dar solución a eventuales problemas que puedan presentarse y consolidar una base sobre la cual desarrollar trabajos futuros.
El diseño de la arquitectura se basa en un esquema hibrido, en el cual se evidencia una arquitectura P2P y cliente/servidor al mismo tiempo, lo anterior se debe a que al tener un entorno IPTV, la implementación de los servicios se hace en un esquema cliente/servidor y con WebRTC el servidor central no es usado para el intercambio de información y contenido multimedia por lo que la comunicación es realizada de manera directa de modo punto a punto.
Con el fin de evitar confusiones y facilitar su comprensión, fue adoptada, para la presentación de la arquitectura del sistema que incorpora el mecanismo propuesto, el modelo de vistas 4+1 [40]; este modelo permite la descomposición y descripción de arquitecturas software mediante diferentes vistas o perspectivas las cuales son ilustradas con la herramienta StarUML [61].

2.8.1. Vista de escenario

En este diagrama es contemplado los posibles usos que se le pueden dar al sistema desde el punto de vista de cada uno de los actores del sistema, para este caso en particular "*User*" y "*Manager*". En la *figura 10* son expuestos los Casos de Uso considerados durante el diseño del sistema.

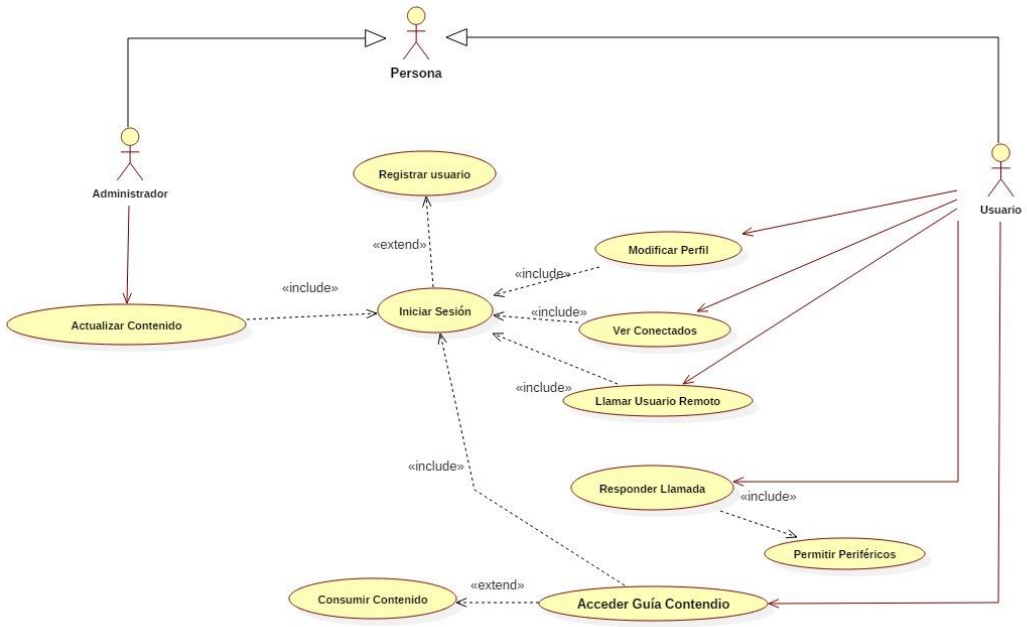

Figura 9: Diagrama de casos de uso UML

Con el objetivo de dar mayor claridad sobre los actores del sistema y los casos de uso anteriormente expuesto, a continuación, es realizado un breve resumen de cada uno de ellos:

a) PERSONA:

- **Administrador:** es el único actor encargado de gestionar la información y los contenidos disponibles para el servicio de VoD, sus funciones son asociar contenido multimedia y adicionar contenido informativo de cada video asociado.
- **Usuario:** actor encargado de consumir los servicios prestados por el sistema.

b) CASOS DE USO:

- **Iniciar sesión:** todo actor relacionado con el sistema solución debe iniciar sesión, este es el primer paso que es necesario para que el sistema identifique que tipo de actor (Administrador o Usuario) intenta acceder y verifique si ha sido previamente registrado, esto último en el caso de ser Usuario.
- **Actualizar contenido:** caso de uso exclusivo para el **Administrador**, permite adicionar contenido multimedia e información al sistema para su visualización en el servicio de VoD.
- **Acceder Guía Contenido:** después de **Iniciar sesión** o realizar un nuevo registro; los usuarios accederán automáticamente a la página principal de la aplicación, en ella podrán observar los contenidos multimedia disponibles en la aplicación mostrados en una **Guía Contenido** y seleccionar el que desea ver.
- **Consumir Contenido:** después de **Acceder Guía Contenido** y seleccionar el contenido que desea consumir, el usuario ejecuta el caso de uso actual, en el podrá reproducir el video seleccionado e interactuar con un grupo de funciones dedicadas al control del mismo.

- **Registrarse:** Una persona tiene la capacidad de registrarse en el sistema si no lo ha hecho previamente para que pueda hacer uso de los servicios.
- **Modificar perfil:** una vez el usuario haya realizado su proceso de registro, tendrá la posibilidad de realizar modificaciones en el perfil creado durante su registro.
- **Ver Conectados:** estando en la página de Guía de contenido, el usuario tiene la posibilidad de ver quienes se encuentran conectados al servicio de videollamada. Para que un usuario esté conectado a este servicio deberá presionar el botón “Online” en la interfaz principal del sistema.
- **Llamar Usuario Remoto:** si el usuario a elegido **Ver Conectados** podrá efectuar una videollamada con cualquier usuario que se encuentre conectado en ese momento.
- **Enviar Notificaciones:** una vez el usuario indique que desea realizar una llamada, el sistema enviará de manera automática una notificación al usuario remoto, por medio del servidor *Node*, para el establecimiento de la videollamada.
- **Solicitar Periféricos:** la **Solicitud de Periféricos**, al igual que el caso de uso anterior, es ejecutado de manera automática por el sistema cuando el usuario decide llamar, aquí el usuario puede decidir si permitirle o no al sistema usar sus periféricos (cámara y micrófono) para llevar a cabo la videollamada.
- **Responder Llamada:** en el caso de ser el receptor, el usuario podrá contestar o rechazar la petición de comunicación realizada por otro usuario, de ser aceptada la petición será presentada la interfaz dedicada a la comunicación entre usuarios donde también se llevará a cabo el caso de uso **Solicitar Periféricos**.

2.8.2. Vista lógica

En la vista lógica son plasmados los requerimientos funcionales del sistema, es decir, se indican que servicios puede proveer el sistema a los usuarios. Para esta vista son construidos diagramas de secuencia de clases. Las descripciones completas de los diagramas de secuencia se pueden ver en el Anexo B adjunto a este documento.

El diagrama de clases es un referente para esta vista, permite dividir en abstracciones del mundo real (clases) los componentes del sistema y describir en cada clase las acciones que realiza. También permite observar cómo interactúan estas clases para ejecutar de manera satisfactoria sus funciones.

A continuación, se muestra el diagrama de clases (ver *figura 11*) elaborado para el sistema que contiene el mecanismo propuesto.

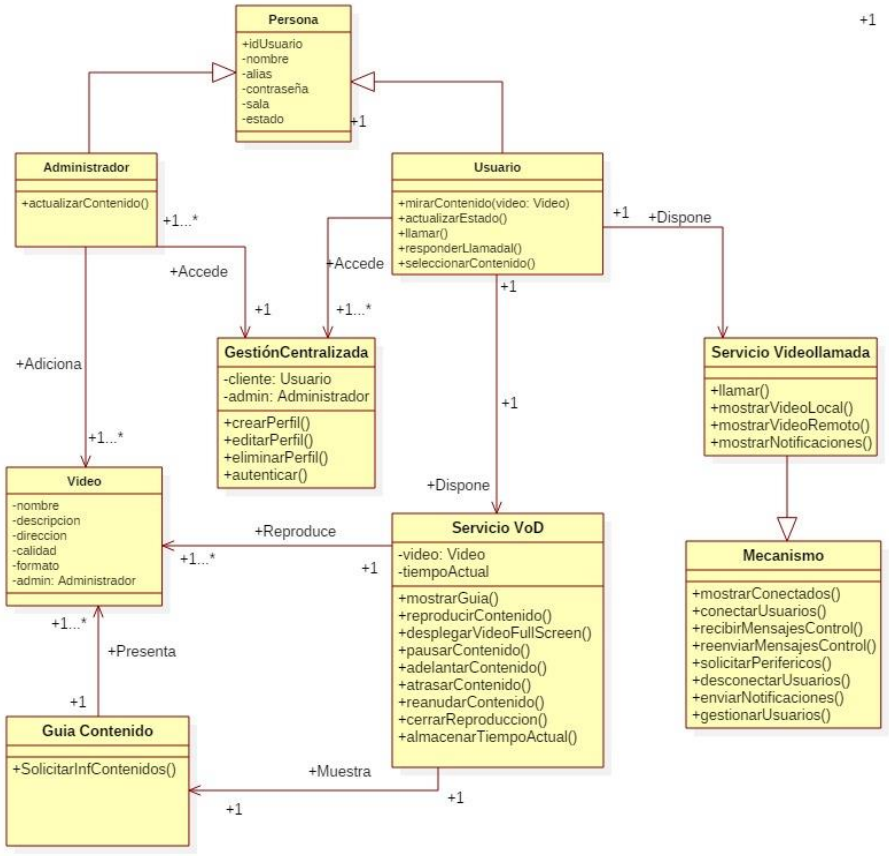

Figura 10: Diagrama de clases UML

A continuación, es descrito las clases usadas en el modelo anterior con el fin de dar mayor claridad al lector.

- **Persona**: es una superclase que abstrae las características generales de los usuarios del sistema, sus atributos reflejan el perfil almacenado de cada uno de ellos en la base de datos.
- **Administrador:** clase hija de la superclase **Usuario** que abstrae las características del usuario encargado de los procesos administrativos del sistema, esta clase adiciona la función **actualizarContenido** la cual le permite al administrador manipular el contenido multimedia que será reproducido en el **Servicio VoD**, pudiendo subir contenido, eliminar contenido y actualizar información adicional al contenido.
- **Usuario**: es la segunda clase hija de la superclase Persona, creada para abstraer las características del usuario cliente quien finalmente consume los servicios provistos por el sistema, para tal fin adiciona las siguientes funciones:
  - **seleccionarContenido:** permite al usuario seleccionar el contenido que desea consumir de un grupo de videos provistos por el **Servicio de VoD**.
  - **mirarContenido:** función que le permite al usuario reproducir el contenido que haya seleccionado previamente.
  - **actualizarEstado:** otorga al consumidor la posibilidad de cambiar su estado de conectado a desconectado o viceversa para el **Servicio Video llamada**.

- **llamar:** permite a los clientes seleccionar a quien desea llamar de un grupo de usuarios con estado conectado al **Servicio Video llamada**.
- **responderLlamada:** le entrega al cliente la facultad de decidir si desea o no responder una llamada entrante de otro cliente conectado al **Servicio videollamada**.

- **GestiónCentralizada:** clase encargada del control de los actores y el acceso al sistema:
  - **crearPerfil, editarPerfil, eliminarPerfil:** este grupo de funciones le permiten al usuario manipular su perfil de usuario.
  - **autenticar:** refleja la acción de verificar los datos entregados por los **Actores** durante el proceso de autenticación (alias y contraseña) para ser identificado por el sistema.
- **Video:** clase concebida para representar algunos atributos inherentes a los videos en el mundo real como la calidad y el formato junto con una descripción de su contenido dedicada al cliente, adicionalmente cuenta con un atributo **admin** de tipo **Administrador** que nos indica qué usuario administrador ha adicionado el contenido al sistema.
- **Guia Contenido:** esta clase está enfocada en solicitar el contenido multimedia disponible para el sistema de la mano de los atributos establecidos en la clase **Video**.
- **Servicio VoD:** clase dedicada a proporcionar el servicio de video bajo demanda del entorno IPTV a los usuarios del sistema. Dentro de sus atributos tenemos el **cliente** que nos indica qué usuario está conectado al servicio, de igual forma el **video** nos indica que video el usuario ha seleccionado para ser visualizado y por último el **tiempoActual** nos permite almacenar el tiempo de ejecución del video que el cliente está visualizando para que en una posterior ocasión él pueda reanudar la reproducción del video justo donde la dejó la última vez, para conseguir esto y algunas otras características se adicionan a esta clase las siguientes funciones:
  - **mostrarGuia:** presenta los contenidos multimedia disponibles para el servicio de VoD y su correspondiente información en una tabla para que el cliente pueda seleccionar el que desee ver, esta información es tomada de la clase **Guia Contenido**.
  - **desplegarVideoFullScreen:** le permite al sistema desplegar el contenido multimedia seleccionado en una interfaz dedicada a la reproducción del contenido donde se podrá apreciar de una mejor manera ocupando un gran porcentaje de la página web.
  - **reproducirContenido:** como su nombre lo indica esta función permite la reproducción automática del contenido multimedia una vez el usuario lo haya seleccionado de la guía de contenido, además de reanudarlo en el tiempo de ejecución que haya sido almacenado en la última visualización de este video por el usuario en cuestión.
  - **pausarContenido, adelantarContenido, atrasarContenido, reanudarContenido:** este conjunto de funciones se crean para el control de reproducción del video seleccionado por el cliente, estas mismas son ejecutadas exclusivamente en la interfaz **VideoFullScreen**.
  - **cerrarReproducción:** este método le permite al cliente detener la reproducción del contenido y ejecutar otras funciones.
  - **almacenarTiempoActual:** cuando el cliente decide cerrar la reproducción del contenido multimedia y retornar al menú principal de la aplicación, se ejecuta esta función, su objetivo es guardar el tiempo de ejecución que el video tenía cuando se finalizó su reproducción en el atributo **tiempoActual** junto con **idCliente** y **idVideo**.

- **Mecanismo:** esta superclase encierra la lógica del mecanismo planteado para este trabajo de grado, puede ser heredada por cualquier servicio en tiempo real basado en WebRTC como por ejemplo la clase hija **Servicio videollamada**; sus funciones mostradas a continuación permiten el establecimiento, control y terminación de una comunicación entre usuarios clientes del sistema.
  - **mostrarConectados:** para dar inicio con el establecimiento de la comunicación es necesario conocer qué clientes se encuentran conectados, es por ello que se dispone de esta función, su objetivo es mostrar que usuarios se encuentran disponibles para ser llamados y permitir su selección.
  - **conectarUsuarios:** ejecuta las funciones provistas por la API WebRTC para el establecimiento del canal de comunicación entre puntos, cuando un usuario A intente dar inicio a una comunicación, se inicializa un objeto *PeerConection*, este objeto a través de mecanismos de NAT-Traversal, permite la comunicación a través de *Firewalls* y *NAT* y es posible conocer la topología de la red para usar el mejor camino de comunicación posible, una vez hecho esto, las funciones JavaScript generan un mensaje de señalización para ser enviado al usuario B.
  - **enviarMensajesControl:** después del establecimiento de los canales de comunicación es necesario intercambiar mensajes para el control de la comunicación, estos permiten invitar a otros clientes a establecer una video llamada o informales su terminación. Esta función provee al sistema la posibilidad de enviar mensajes de control que contienen cliente origen, cliente destino y mensaje.
  - **recibirMensajesControl:** una vez enviados los mensajes de control, es necesario tener una función que pueda recibir e interpretar dichos mensajes, con este fin se crea el actual método.
  - **reenviarMensajesControl:** esta función es propia del conmutador, permite la recepción y redistribución de los mensajes enviados por los clientes, esto es necesario debido a que fue usado un servidor de señalización en la nube y no se tenía mensajes que permitieran al proveedor gestionar el servicio provisto por el proyecto WebRTC para el envío de mensajes de control directos entre clientes.
  - **solicitarPerifericos:** este método se encarga de ejecutar las funciones de la API MediaStream para la petición de los periféricos del dispositivo usado por el cliente para el establecimiento de la comunicación.
  - **desconectarUsuarios:** para la finalización de la comunicación además de enviar el mensaje de control es necesario liberar los recursos usados y desplazar al usuario a la interfaz principal del sistema, esto es llevado a cabo en el actual método.
  - **gestionarUsuarios:** este método representa las acciones que ejecuta el mecanismo para permitir un enriquecimiento en términos de la gestión de los usuarios que usan el servicio en tiempo real de la videollamada, es decir, el establecimiento, control y finalización de la comunicación.

- **Servicio videollamada:**
  - **llamar:** cuando el cliente ha seleccionado a quien llamar, este método ejecuta funciones de la superclase **Mecanismo** necesarias para establecer una comunicación.
  - **mostrarVideoLocal:** antes de iniciar una videollamada, esta función le permite al cliente observar la captura de video que realiza la cámara de su dispositivo.
  - **mostrarVideoRemoto:** despliega en un contenedor HTML el flujo multimedia enviado por el cliente remoto.

- **mostrarNotificaciones:** esta función le permite al sistema desplegar ventanas emergentes para informar al cliente que está siendo llamado y permitiéndole decidir si acepta o no la llamada entrante.

2.8.3. Vista de desarrollo

Para lograr una organización modular del entorno de desarrollo software es concebida la vista de desarrollo en la cual el sistema es presentado en pequeños paquetes software organizados en capas y con interfaces de comunicación entre capas bien definidas, además la vista de desarrollo permite organizar el equipo encargado de la construcción del sistema, evaluar la seguridad y monitorear el progreso del proyecto.

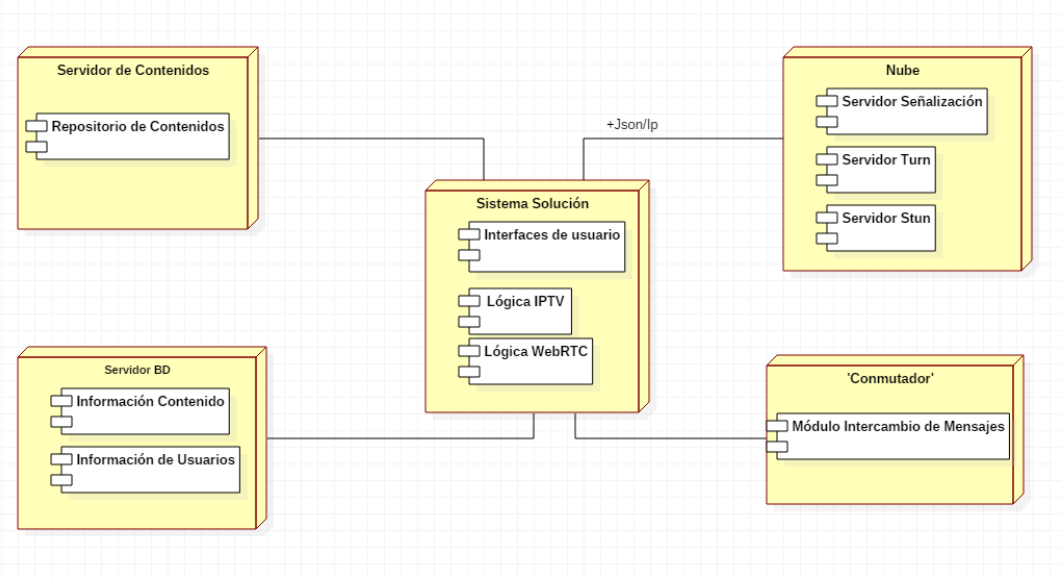

Figura 11: Diagrama de componentes UML

En la figura anterior (ver *figura 12*) es expuesto el diagrama de nodos y componentes, por claridad del lector a continuación se describen las tareas de cada componente:

1) **Nodo Servidor de contenidos.**
   - **Repositorio de contenidos:** encargado de almacenar el contenido para el servicio de VoD, los cuales son accedidos cuando el usuario lo solicite.
2) **Nodo servidor de Bases de datos(BD)**
   - **Información de Contenido:** es creada una tabla en la base de datos exclusivamente para almacenar los datos correspondientes al contenido multimedia almacenado en el **Repositorio de contenidos**, datos como nombre, descripción, dirección en el repositorio, duración, resolución y formato.
   - **Información de los Usuarios:** también existe una tabla dedicada a la información de cada usuario registrado en el sistema, estos datos son usados para la identificación y prestación de los servicios en el sistema propuesto.
3) **Nodo Sistema solución**

- **Interfaces de Usuario:** cada interfaz permite al usuario navegar por los servicios prestados, en ellas se visualiza el contenido para el servicio de VoD y los datos multimedia de la videollamada, así como también es posible saber que usuarios están conectados y por ende a quien es posible llamar.
- **Lógica IPTV:** este componente tiene como función ejecutar toda la lógica relacionada con el entorno de IPTV y específicamente del servicio de VoD, por lo que organiza la información en las interfaces dependiendo de las acciones del usuario, además consulta al **Nodo servidor de Bases de datos** la información requerida cuando esta es solicitada.
- **Lógica WebRTC:** este componente es el encargado de gestionar el servicio videollamada, su función está orientada a la toma de decisiones dependiendo de lo que el usuario realice, además él, es el núcleo que administra las conexiones a otros componentes que se requieren para prestar el servicio.

4) Nodo Nube

- **Servidor de señalización:** este servidor permite el intercambio de mensajes necesarios para establecer la comunicación punto a punto.
- **Servidor *Stun*:** este servidor es usado por el sistema para encontrar al usuario remoto, esto se hace, como se explicó en capítulos anteriores (ver 1.2.4.1), debido a que, el usuario puede estar en redes privadas y es difícil contactarlo directamente para así poder establecer la comunicación punto a punto sin el uso de un servidor central.
- **Servidor *Turn*:** este servidor es usado en casos particulares, es decir, es invocado en caso de emergencias, cuando el servidor *Stun* falle o el usuario remoto se encuentre detrás de NATs simétricas y sea imposible llegar a él a través de otro mecanismo.

5) En el servidor Node.js

- **Módulo de intercambio de mensajes:** este módulo es el encargado de recibir y re-enviar notificaciones entre usuarios, es construido para comunicar eventos durante el proceso de ejecución del servicio, es decir, por medio de él se puede transmitir, a modo de alertas, la intención de establecer una nueva conexión o la finalización de una comunicación ya establecida entre usuarios.

2.8.4. Vista de procesos

La vista de procesos incluye los requerimientos no funcionales del sistema como el rendimiento y la disponibilidad, además organiza los grupos de tareas que deben ejecutarse como unidad para formar procesos del sistema y al mostrar la relación entre procesos permite observar con mayor claridad la integración del sistema.

Uno de los diagramas más relevantes para esta vista es el diagrama de Actividad, en él se puede observar el flujo de trabajo a través de una serie de tareas y decisores [62], en la *figura 13* se presenta el diagrama de Actividades elaborado para el presente trabajo en el cual es mostrado el proceso general que un usuario hace del sistema solución que incorpora el los servicios y el mecanismo propuesto.

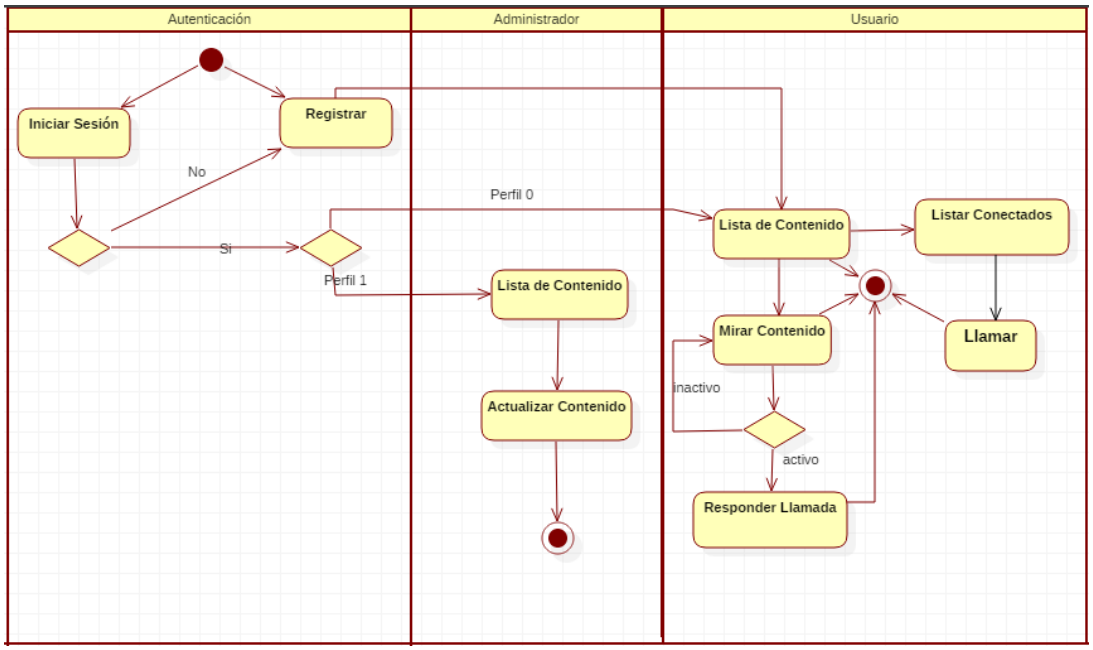

Figura 12: Diagrama de Actividad UML

El anterior diagrama muestra la secuencia de acciones realizadas por el sistema cuando un usuario hace uso de él, partiendo del nodo inicial en donde la persona puede elegir entre iniciar sesión o registrarse en el sistema; una vez este ingrese sus datos, la aplicación resuelve que decisión tomar a través de un nodo decisor, cuando inicie sesión se pregunta si es un administrador (Perfil 1) o si es un usuario común (Perfil 0), ahora bien si el usuario desea registrarse solo podrá hacerlo como cliente, debido a que existe solo un administrador previamente registrado.
Después de que haya iniciado sesión un administrador, el sistema ejecuta la acción de listar el contenido para que este lo actualice si lo desea, de lo contrario se da por finalizada la actividad del administrador.

Por otro lado, si el usuario inicia sesión el sistema también realiza la acción de mostrar la guía de contenidos disponibles, una vez allí cada usuario tiene la opción de realizar dos acciones: elegir y mirar un contenido, o ver que otras personas están conectadas al servicio, es importante resaltar que si el usuario se encuentra mirando un contenido y está activo, puede recibir llamadas en cualquier momento y el contenido es reanudado una vez haya finalizado la videollamada en el mismo tiempo de reproducción, si el usuario decide llamar a otro usuario es iniciado el proceso de ejecución del servicio como es mostrado en la sección 1.2.4.

2.8.5. Vista física

Al igual que para la vista de Procesos, la Vista Física toma en consideración los requerimientos no funcionales como la tolerancia a fallos y la escalabilidad, esta vista es utilizada para mapear el software del sistema a el hardware que es usado para su ejecución, cada proceso puede mapearse a dispositivos como computadoras o a nodos de procesamiento dentro de un mismo dispositivo.

Para la Vista Física se construye el diagrama de Despliegue desarrollado haciendo uso de la herramienta Microsoft Visio el cual es mostrado en la *figura 14*, en donde es posible observar claramente la topología hardware del sistema.

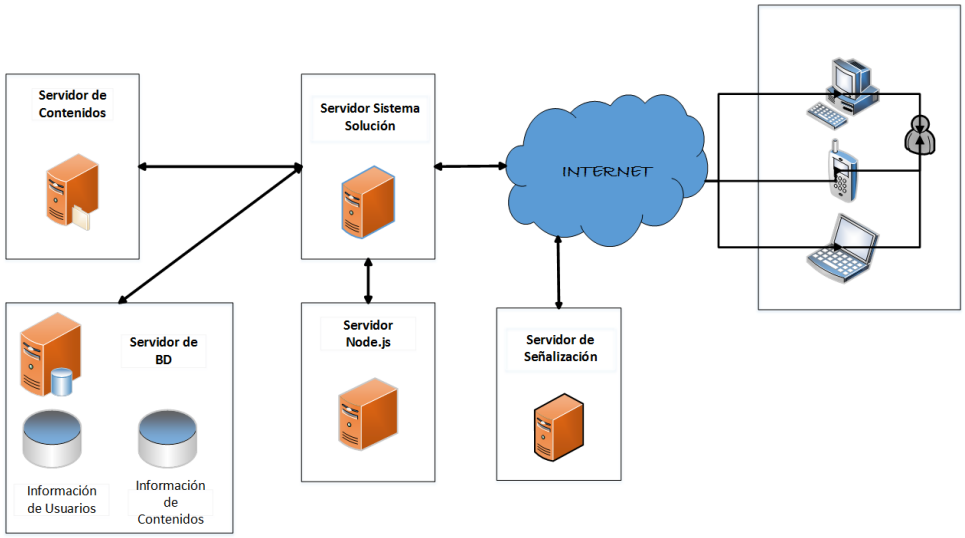

Figura 13: Diagrama de despliegue

A partir de lo expuesto en el capítulo II es posible dar por cumplido el objetivo número 1: "Diseñar la arquitectura del mecanismo que permita la comunicación en tiempo real haciendo uso de la tecnología WebRTC en un entorno IPTV", aunque es de suma importancia resaltar que la arquitectura propuesta y las consideraciones aquí tomadas deben ser asumidas como referencia o como base conceptual para una eventual implementación en entornos reales y comerciales, por lo tanto la arquitectura debe ser adaptada debido a que los criterios de selección y requerimientos del sistema pueden variar dependiendo de cada caso de uso particular.

CAPÍTULO III

DESARROLLO E IMPLEMENTACIÓN

3.1. INTRODUCCIÓN

En los capítulos anteriores fueron descritas las características de las tecnologías utilizadas WebRTC e IPTV y sus entornos correspondientes (ver secciones 1.2, 1.3, 2.3 y 2.4) para el planteamiento de un mecanismo que permita el enriquecimiento de los servicios en tiempo real sobre un entorno de IPTV; además en el capítulo II fue realizado el diseño de la arquitectura para el sistema que contendrá al mecanismo propuesto (ver secciones 2.5, 2.6, 2.7 y 2.8), basándose en la descripción de características y requerimientos de las tecnologías usadas. Con el resultado del trabajo y la captura de requisitos realizada en los anteriores capítulos, es realizada la construcción del sistema solución, el cual incluye los servicios escogidos como apoyo al mecanismo propuesto, y su posterior puesta en marcha en el Laboratorio de Televisión Digital de la Universidad del Cauca.

3.2. DEFINICIÓN DEL ESCENARIO Y ENTORNO DE DESARROLLO

Para la fase de desarrollo fueron seleccionadas herramientas acordes con las tecnologías usadas en el sistema propuesto, a continuación, son descritos brevemente los recursos software usados para la fase de desarrollo:

3.2.1. Proyectos Base

La construcción de un proyecto WebRTC puede llevarse a cabo de muchas maneras, dependiendo del nivel de control y experiencia en el desarrollo de aplicaciones VoIP y WebRTC, en la *figura 15* son mostradas algunas de las posibles maneras de construir un nuevo proyecto WebRTC [79].

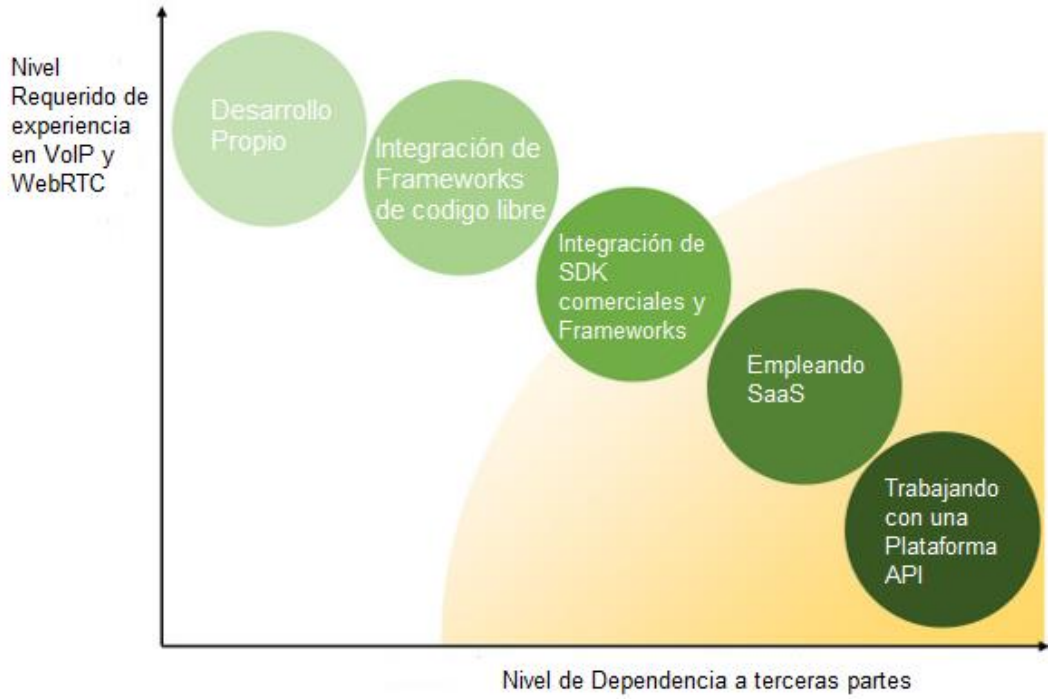

Figura 14: Alternativas para la construcción de proyectos WebRTC [63]

Dada la necesidad de controlar gran parte de la aplicación WebRTC, en el presente trabajo de investigación se adoptó el modo de construcción "Integración de *Frameworks* de código libre". Con el método de integración de Frameworks, WebRTC puede ser visto como un motor de medios[14] [64], dejando a consideración del desarrollador el proceso de averiguar con quien conectarse y la negociación de la sesión.

En la actualidad existen *Frameworks* (e.g., PeerJS, EasyRTC) que cubren aspectos de la señalización en WebRTC, utilizan tecnologías como las descritas en la sección *2.3.2* permitiendo la comunicación a través de navegadores. En su gran mayoría los *Frameworks* para WebRTC son de código libre y es posible encontrarlos inmersos en muchos proyectos, a continuación, son presentados algunos de los más relevantes:

- PeerJS: envuelve la implementación de WebRTC sobre los navegadores para proporcionar una API de conexión P2P completa, configurable y fácil de usar [65].
- EasyRTC: centrado en las videoconferencias, es una forma sencilla de implementar aplicaciones WebRTC seguras de video, audio y datos [66].
- SimpleWebRTC: es uno de los *Frameworks* con más años de antigüedad, con un resiente enfoque por parte de sus desarrolladores originales. [67]

Desde su liberación, muchas iniciativas han surgido con base en los *Frameworks* anteriormente nombrados, uno de los más relevantes, el "Proyecto oficial de WebRTC" (WebRTC Project) cumple ya 5 años de trabajo [8], este cuenta con un '*codelab'*, es decir, una herramienta en línea para aprender y practicar sobre WebRTC, allí es posible encontrar

[14] Motor de medios: componente software encargado del procesamiento de medios.

información necesaria para iniciar en este mundo, tutoriales para el establecimiento de un stream de video usando un *RTCPeerConection*, la configuración de un servicio de señalización para el intercambio de mensajes y el uso *RTCDataChannel* para el envío de datos; además en su cuenta GitHub[15] está disponible el código fuente desarrollado con base en el *Framework* SimpleWebRTC, el cual contiene funciones asociadas a la implementación y establecimiento de una comunicación punto a punto sobre las cuales es viable construir aplicaciones enfocadas a la comunicación en tiempo real, como es el caso del presente proyecto de investigación.

Por otro lado, una de las primeras implementaciones fue mostrada por Ericsson en el 2011 [68], otros tantos menos reconocidos como el proyecto EasyRTC desarrollaron una aplicación para llevar a cabo una comunicación en tiempo real usando WebRTC, el código de esta aplicación también es de fuente libre y está disponible en su página oficial [66].

Debido a la información presentada por el "*WebRTC project*", la facilidad para acceder a su código y la buena documentación elaborada alrededor de este; para el presente proyecto de investigación se decidió trabajar con los códigos fuente entregados por este equipo de desarrollo junto con el *Framework* SimpleWebRTC, usándolos en la construcción del mecanismo propuesto.

3.2.2. Selección tecnologías de programación

La selección correcta del lenguaje y/o *Frameworks* de programación es fundamental durante el desarrollo de un proyecto, ya que condicionan las herramientas que el desarrollador usará, el rendimiento que la aplicación tendrá y la fluidez con la que el desarrollador realizará su tarea.

A continuación, son expuestos algunos de los lenguajes de desarrollo más populares para aplicaciones web.

- **Java:** este lenguaje de programación orientado a objetos fue desarrollado por *Sun Microsystems*, usado para el desarrollo de aplicaciones web, convirtiéndolo en la base para la mayoría de tipos de aplicaciones en red; el fuerte del lenguaje Java es su portabilidad, está diseñado para permitir el desarrollo de aplicaciones comúnmente sobre las plataformas informáticas disponibles actualmente [69], debido a su máquina virtual de java "JVM".

  Gran parte del lenguaje Java es basado en el lenguaje C, con la diferencia que este es un lenguaje de alto nivel y simplifica la programación orientación a objetos, facilitando el trabajo al desarrollador.

  Con más de 9 millones de desarrolladores en todo el mundo, Java ha llegado a convertirse en uno de los lenguajes más populares hoy en día [70].

- ***RUBY*:** es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su sintaxis es fácil tanto al leerla como al escribirla [71].

[15] **GitHub**: es una plataforma de desarrollo para el control de versiones y el trabajo colaborativo, permite trabajar a equipos conjuntamente sobre un mismo proyecto.

- **PHP:** acrónimo de *Hypertext Preprocessor***,** lenguaje de código abierto usado para la construcción de contenido dinámico en las páginas Web, es uno de los lenguajes de escritura más rápidos en el mundo, de fácil uso, y además puede ser usado en IDE (*Integrated Development Environment*), es usado en sitios web como Facebook, Oracle y Yahoo [72].

  Es un lenguaje netamente del lado del servidor, es decir se ejecuta en la máquina donde esta almacenada la aplicación web, realiza sus procesos justo antes de que la página HTML sea enviada al cliente.

- **PYTHON (DJANGO):** Django es un *Framework* web de alto nivel basado en Python, generalmente usado para desarrollos rápidos y competitivos, este es un *Framework* de código abierto que permite a los desarrolladores centrarse en la escritura de su aplicación, sin el uso de una sintaxis compleja para la implementación de la lógica de la aplicación a desarrollar [73].

- **JAVASCRIPT [74]:**
  Este es un lenguaje de programación usado para crear páginas web interactivas, especialmente contenido dinámico, es decir, que se actualice constantemente una vez se haya cargado la página web. Es importante notar que actualmente la mayoría de navegadores web lo soportan permitiendo su uso por los desarrolladores.

  Aunque es posible usar cualquiera de las anteriores tecnologías para desarrollar una misma aplicación, hay características como la curva de aprendizaje, la documentación disponible, la solidez, entre otras, que distinguen una tecnología de otra.

  La decisión de trabajar con el lenguaje de programación Java se toma considerando las características como escalabilidad, robustez y respaldo, sumado a esto, Java al ser orientado a objetos posibilita hacer un uso eficiente del código, con Java es posible desarrollar aplicaciones en distintas plataformas como Linux, Windows y MacOS; Java contiene una gran cantidad de librerías disponibles para la implementación de aplicaciones de este tipo y la mayoría de ellas son de fuente abierta y cuentan con soporte y documentación [75], además considerando las estadísticas TIOBE se ratifica la decisión, debido a que Java es el lenguaje de programación más popular hasta Julio del 2016. [76].

3.2.3. Selección del entorno de desarrollo

La selección de un IDE (*Integrated Development Environment*) va de la mano con la selección del lenguaje de programación, al escoger Java como lenguaje de programación para el proyecto el entorno más propicio para trabajar puede ser escogido entre Netbeans y Eclipse, él usado fue Netbeans ya que este ya incluye los módulos y plugins necesarios.

Una de las ventajas de trabajar con Netbeans es que provee la plataforma Java EE para la creación de aplicaciones empresariales de la cual se hablará a continuación:

JAVA EE [77]: esta plataforma permite el desarrollo de aplicaciones empresariales más fácil y rápido, proporciona la lógica del negocio para una empresa, centralizando la gestión e interactuando con otro tipo Software empresarial.

El objetivo más importante de la plataforma Java EE es simplificar el desarrollo, entregando una base común para los diversos tipos de componentes en la plataforma.

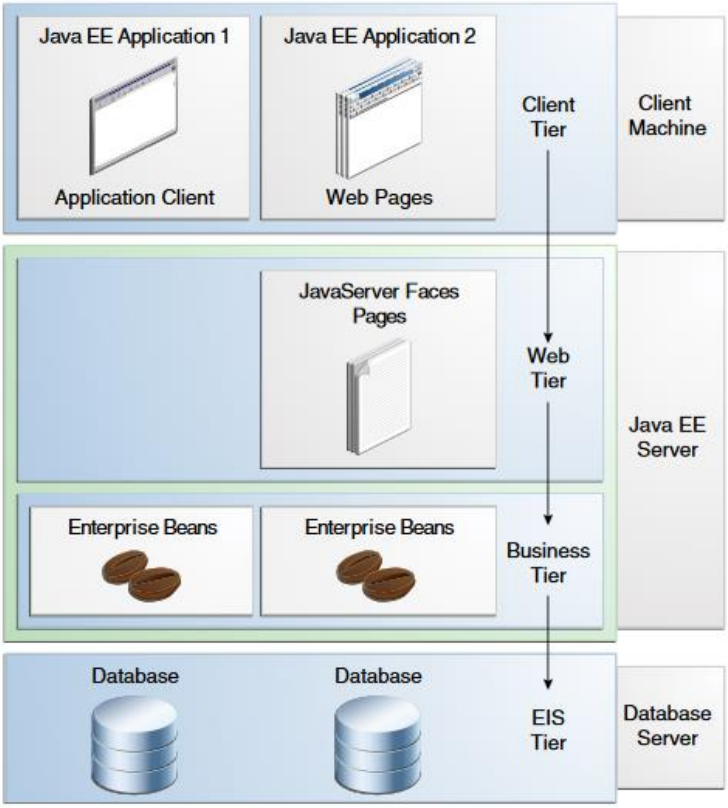

Figura 15: Niveles plataforma Java EE [77]

Es seleccionada la plataforma Java EE para la implementación de la arquitectura propuesta en este trabajo de grado por las siguientes razones:

- **Interacción con Software empresariales:** dada su facilidad para integrar diferentes tipos de software a la aplicación empresarial en construcción, permite la interacción de tecnologías diferentes como WebRTC e IPTV en una sola plataforma.

- **Modelo de aplicación distribuida:** permite la modularización durante el desarrollo de la aplicación adaptando de mejor manera la arquitectura propuesta a la aplicación en desarrollo, además permite la separación de la aplicación en diferentes máquinas clarificando la arquitectura de despliegue y permitiendo una mejor evaluación de la solución planteada.

3.2.4. Selección Gestor de bases de datos [78]

Debido a los módulos del mecanismo propuesto y del entorno de IPTV en los cuales es usado características de gestión y almacenamiento de la información del perfil de usuario,

se construyó una base de datos relacional que fue elaborada haciendo uso del gestor MySQL, siendo esta una de las más populares en gestión de bases de datos para entornos web, usada por Facebook, Twitter, LinkedIn entre otros. Esta herramienta es de código abierto liberada por Oracle y perteneciente al mismo conjunto de herramientas junto con Netbeans; dadas sus características de escalabilidad y facilidad de uso fue seleccionado para el desarrollo del sistema solución que engloba al mecanismo propuesto en este trabajo de grado.

3.3. PROCESO DE CONSTRUCCIÓN DEL SISTEMA

Para la fase de desarrollo del mecanismo, se decidió optar por la instalación de la plataforma de software libre XAMPP (Apache, MySQL, PHP, Perl), ya que incorpora un motor de base de datos MySQL, y un servidor web. Igualmente fue instalado el entorno de desarrollo integrado NetBeans para el trabajo con la plataforma Java EE, además fue de gran ayuda el uso de herramientas de edición de texto e imágenes para las interfaces gráficas tales como SublimeText, Microsoft Visio, Paint.
La implementación de los módulos del mecanismo definidos en la sección2.5 (Conmutador, Funciones WebRTC y Controlador) fue realizada de la siguiente manera:

- **Conmutador**
  A partir del diseño del mecanismo propuesto expuesto en la sección 2.5.2. en el cual de definió este módulo que fue construido haciendo uso de Node.Js, el cual es “un entorno de ejecución para JavaScript”, con él fue posible establecer y manejar eventos asíncronos para el envío, recepción y control de mensajes entre clientes, una de las ventajas de usar esta herramienta es que permite manejar muchas conexiones concurrentes [79]. En la siguiente imagen es mostrado un ejemplo de conexión entre dos usuarios, los mensajes vistos en la captura de pantalla son aquellos que se muestran al proveedor para controlar la actividad de cada usuario registrado en el sistema.

Figura 16: Mensajes del servidor Node.js

- **Funciones WebRTC**
  Este bloque fue implementado dentro del mecanismo propuesto (Ver sección 2.5) usando el lenguaje JavaScript, en él fueron implementadas las funciones necesarias para la prestación del servicio en tiempo real de la video llamada, estas funciones son las encargadas de hacer la captura multimedia (audio y video), el establecimiento de la conexión en tres puntos remotos y el intercambio de datos punto a punto.

- **Controlador**
  Este módulo representa la lógica del sistema y es el encargado de tomar decisiones dependiendo de las acciones del usuario, es usado para enlazar la lógica de los servicios y poder ejecutar tanto las funciones de WebRTC como las del conmutador, puede ser considerado como el corazón del mecanismo propuesto (Ver sección 2.5), en otras palabras, este bloque ejecuta el código para decidir cuándo es requerido recibir y reenviar mensajes y es el encargado de la gestión centralizada de los usuarios.
  Su implementación fue realizada usando programación orientada a objetos con un núcleo en Java y sub rutinas de control del lado del cliente usando JavaScript.

3.3.1. Aplicación IPTV

Después de instalado el entorno de desarrollo con base a las guías disponibles en la página oficial de NetBeans [80], el siguiente paso fue la estructuración de la información para la gestión de los posibles servicios IPTV, para llevar a cabo esto fue tomado en cuenta los siguientes requerimientos:

- Sección **2.6.2** ítem 4: para poder presentar la información general y técnica de los contenidos multimedia, es necesario que de cada video sean almacenados datos como: título, descripción, duración y calidad del contenido, esto se logró estructurando una tabla específica para el contenido multimedia dentro de la base de datos, allí se almacenó tanto la ruta de los videos como los campos anteriormente nombrados.
- Sección **2.6.1** ítem 1, 7 y 8: la seguridad de la aplicación está basada en la utilización de un alias y una clave para cada usuario, con este método se realiza un control de acceso, autenticación de usuarios y en cierto grado una protección al servicio prestado. El almacenamiento de los alias y claves fue realizado en una tabla de la base de datos dedicada a los usuarios donde además es almacenado nombres, estados de conexión o desconexión y el rol que cada usuario desempeña en la aplicación, entre otros.
- Sección **2.6.4** tabla 1: para el control de los perfiles de usuario descritos en la sección **1.3.3.5** se crea una sección dentro de la aplicación donde cada usuario puede crear un nuevo perfil o modificar los datos almacenados de su perfil previamente creado.

Ya estructurada la información que maneja la aplicación, el siguiente paso fue proceder a la construcción del servicio de VoD, para este fin fueron creadas dos páginas web construidas con HTML5 y CSS3, en donde además fue adicionado JavaScript y PrimeFaces para suministrar la presentación de las interfaces de manera más interactiva. Su construcción fue guiada por los siguientes requerimientos:

- La utilización de HTML5, CSS3, JavaScript y PrimeFaces es acorde a lo sugerido en la sección **1.3.3.1.** específicamente en el primer ítem y en la sección **1.3.3.3** ítem 2.
- El usuario podrá seleccionar el video que desea consumir de una lista provista por la aplicación, esta lista mostrará información de cada video almacenada en la base de datos según fue acordado en la sección **2.6.2** ítem 2,3 y 4.
- Los videos disponibles en el servicio de VoD tendrán una resolución de 720p y un formato MP4 según lo acordado en la sección **2.6.2** ítem 6.
- Fue necesario la creación de una relación entre la tabla usuario y la tabla video para almacenar el tiempo de reproducción del contenido que el usuario está consumiendo en el momento de inicio del servicio de llamada; su objetivo es poder reanudar la reproducción del contenido justo en el momento donde fue interrumpido, evitando de esta manera que el usuario pierda de ver parte del mismo, esto fue creado con el objetivo de cumplir el requerimiento expuesto en la sección **2.6.2.** ítem 7.

Para esta sección del sistema solución, correspondiente al entorno IPTV, también fue construido el módulo encargado de la gestión de la aplicación, este módulo fue desarrollado es su gran mayoría usando Java, aunque fue necesario adicionarle una interfaz de identificación, construida con las herramientas web anteriormente nombradas; cabe resaltar que dicha página está disponible para todos los usuarios de la aplicación. El módulo de gestión fue creado en primera medida, para identificar a los usuarios que accedan a la aplicación, y así después proveerles los servicios disponibles en el sistema.

3.3.2. Entorno WebRTC

Para la construcción de la aplicación WebRTC (Ver Anexos), que incluye el módulo de Funciones WebRTC diseñado en el mecanismo propuesto (Ver Sección 2.5.1) y servicio de videollamada básico especificado (Ver sección 2.7.) fue necesario empezar con una modificación a la base de datos estructurada para IPTV (ver figura 18), fue necesario adicionar en la tabla dedicada a los usuarios un campo, donde fue almacenado el nombre de la *Room* correspondiente a cada uno de ellos, esto fue necesario para poder llevar a cabo la conexión entre usuarios a través de la función de WebRTC: **RTCPeerConnection** descrita en la sección **1.2.1**. Dicho campo está compuesto de un número aleatorio, asignado por la aplicación, unido con el alias escogido por el usuario, por lo que este debe ser único para evitar confusiones durante el proceso de llamada ya que como se explicó antes este nombre puede interpretarse como un número telefónico.

id	name	login	password	profile	room	state 0: Desconectado
1	Administrador	admin	YWRtaW4=	1	admin	0
16	johan	johanmos	am9oYW4=	0	840johanmos	1
17	julian	cjdelgado	ZGVsZ2Fkbw==	0	175cjdelgado	1
18	Pedro	pedro	cGVkcm8=	0	350pedro	1
19	Juan	juan	anVhbg==	0	975juan	1

Figura 17: Base de datos perfil de usuario

Después de tener los datos necesarios para establecer una videollamada entre diferentes usuarios de la aplicación, es iniciado el proceso de construcción del servicio de videollamada: Inicialmente fue implementado una videollamada básica (Ver figura 19), donde ambos usuarios debían tener el mismo nombre de '*Room'* para poder comunicarse, es decir, cualquier persona que conociera dicho identificador tenía la posibilidad de acceder a la videollamada establecida entre dos usuarios, presentando desventajas en términos de seguridad e imposibilitando su implementación en aplicaciones reales. Por lo cual para la construcción del sistema solución fue necesario la utilización de diferentes nombres de '*Room'* únicos, una para cada usuario, las cuales son asignados de forma automática después de que el usuario se registre ante el sistema, y eran controlados por el módulo de gestión centralizada, por otro lado la mayoría de los códigos base encontrados en la investigación literaria (ver sección *3.2.1*) carecen de una interfaz dinámica que permitiera su utilización intuitiva e interactiva por parte de un usuario, y de un sistema de gestión para la conexión entre usuarios.

A continuación, es mostrado la implementación de un servicio básico utilizando el proyecto base seleccionado, sobre este servicio básico fue desarrollado posteriormente el servicio mejorado de videollamada para el sistema solución.

Figura 18: videollamada básica

Después del trabajo realizado fueron obtenidas las siguientes interfaces: la primera creada para gestionar y poder observar que otros usuarios están conectados y con los cuales es posible establecer una Video-llamada (ver figura 20); la segunda interfaz es dedicada al proceso de comunicación en la Video-llamada donde es posible observar la captura de video tanto de la cámara propia como la del otro usuario partícipe en la comunicación (ver figura 21). Aquí es posible observar el servicio implementado, descrito en los dos capítulos anteriores.

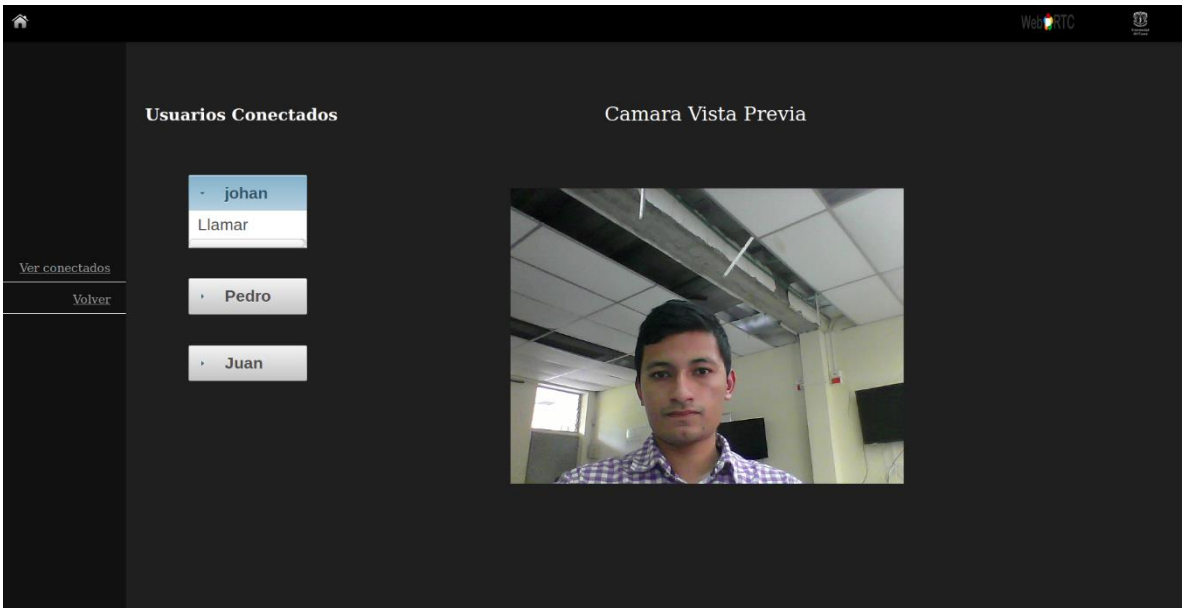

Figura 19: Interfaz "Ver Conectados"

Figura 20: Interfaz WebRTC, videollamada sistema desarrollado

Para tener la interacción de ambas aplicaciones a disposición del cliente, fue necesario incluir dentro de una misma interfaz, el catálogo de vídeos disponibles y la opción de ver a los usuarios conectados al sistema, esta interfaz es mostrada en la *figura 22*. Una vez el usuario haya realizado el proceso de autenticación ante la aplicación, es presentado el contenido multimedia en la sección principal de la página y a su lado izquierdo, en un menú lateral, un enlace llamado “Ver conectados”, donde como su nombre lo indica, podrá observar a todos los usuarios conectados al sistema. A continuación es mostrada la interfaz principal del sistema.

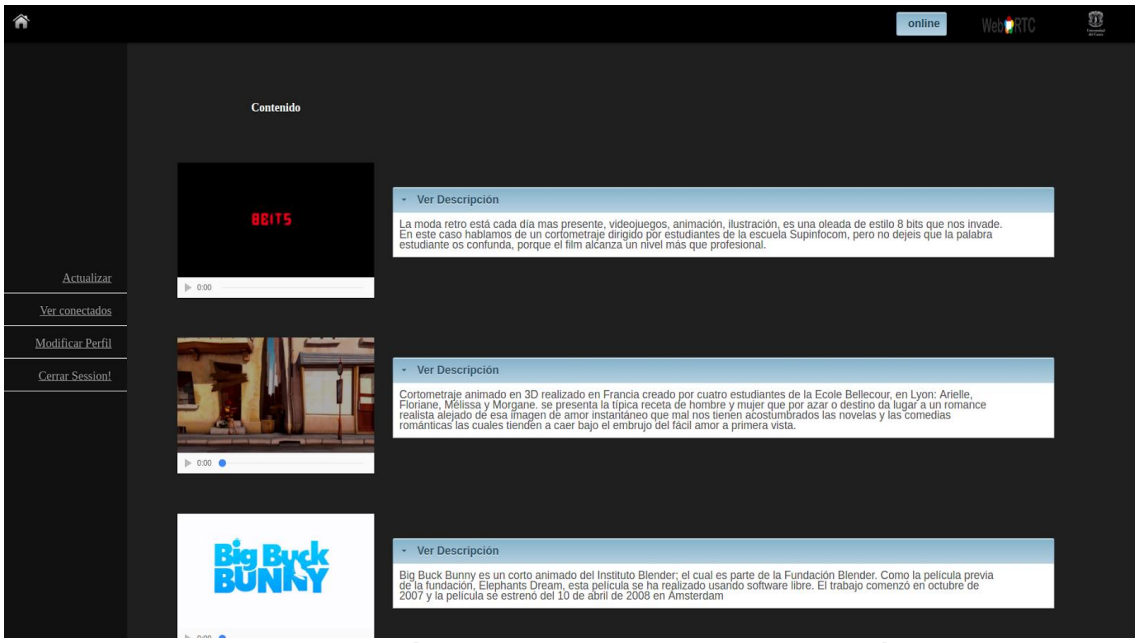

Figura 21: Página principal del sistema solución

CAPÍTULO IV

EVALUACIÓN Y ANALISIS DE RESULTADOS

4.1. INTRODUCCIÓN

Para dar cumplimiento al tercer objetivo planteado en el presente trabajo de grado, fueron evaluadas las características funcionales del sistema desarrollado con base en el mecanismo propuesto y el desempeño que el mismo puede presentar al ser evaluado en un laboratorio. En el actual capítulo son expuestas las pruebas a ejecutar sobre el sistema desarrollado; una vez seleccionado el tipo de evaluaciones es realizado el proceso de construcción de un plan de pruebas, su ejecución y finalmente las conclusiones de los resultados obtenidos.

Según [81], las pruebas realizadas a un sistema pueden clasificarse según la etapa en la cual es desarrollada, por ejemplo “Prueba unitaria”, “Prueba de integración” o “Pruebas del sistema”, estas últimas son llevadas a cabo después del proceso de desarrollo del sistema y son realizadas con el objetivo de evaluar funcionalidad y desempeño que son las requeridas para aplicar al presente trabajo de grado y así dar cumplimiento al tercer objetivo específico “Evaluar la funcionalidad y el desempeño del mecanismo propuesto”.

Pruebas del sistema [81]

Usando las “Pruebas del sistema” es posible evaluar el diseño a un alto nivel y los requisitos planteados para el sistema desarrollado, por medio de pruebas que comprueben que este hace lo que es esperado que haga, para tal fin es usada la técnica de caja negra, en la cual no es necesario conocer los componentes internos del elemento a evaluar, es decir, esta técnica ignora los componentes internos para centrarse directamente sobre las salidas generadas y determinar la respuesta a las entradas en ciertas condiciones de ejecución.

Las características asignadas para las “Pruebas del sistema” son:

- Opacidad: prueba de caja negra
- Requisitos: diseño alto nivel y especificación de los requisitos.
- Objetivo: verificar que las funcionalidades definidas en la especificación de requisitos de la aplicación sean cumplidas.

Los tipos de “Pruebas del sistema” más importantes para este trabajo son:

- **Prueba de funcionalidad:** su objetivo es verificar que las funcionalidades que presenta el sistema cumplan con los requisitos establecidos para el mismo, para llevar a cabo esta prueba es analizado el resultado obtenido cuando es aplicada una entrada a la solución, posteriormente es realizada una comparación entre el resultado esperado proveniente de los requisitos y el resultado obtenido al aplicar la entrada.
- **Prueba de desempeño:** en esta prueba se busca comprobar que el sistema cumple con los requisitos de operación acordados, es decir, que tiene tiempos de respuesta apropiados para su correcto funcionamiento.

4.2. PRUEBA DE FUNCIONALIDAD

Para evaluar la funcionalidad del sistema solución, es necesario comprobar que este cumpla con los requisitos funcionales planteados en capítulos anteriores, para tal fin fue adoptada una técnica de evaluación denominada caja negra **[99]**, inicialmente es necesario

separar los requisitos en casos de uso y agrupar estos casos de uso en el correspondiente componente del sistema al que afecta.

A continuación, son descritos los casos de uso involucrados en la evaluación funcional del sistema desarrollado, estos son divididos en dos grupos: “Características del mecanismo propuesto” y “Requisitos para la implementación de un servicio de videollamada en un entorno IPTV”, ya que las pruebas son ejecutadas solo sobre componentes del sistema directamente relacionados con el mecanismo propuesto.

4.2.1. Agrupación de Requisitos

Tabla 4 Vinculación Casos de Uso y Requerimientos

Componente	Caso de Uso	Requisitos	N° Prueba
Características del mecanismo propuesto	Solicitar Periféricos	El mecanismo ejecuta la solicitud de los periféricos disponibles en el dispositivo del usuario final	01
	Llamar Usuario Remoto	El mecanismo establece canales de comunicación punto a punto (conforme a lo establecido por la API WebRTC) con otros usuarios conectados y su posterior cierre al terminar el proceso de comunicación	02
		Las funciones provistas por el mecanismo deben ser ejecutadas dentro de páginas HTML que sirvan como interfaz para el cliente, con el fin de permitir una interacción entre servicios a nivel de presentación	03
	Enviar Notificaciones	Establece mensajes propietarios para identificar el estado de la comunicación, los cuales posibilitan el control de la misma	04
	Enviar Notificaciones	El mecanismo conmuta mensajes de control entre los usuarios conectados al servicio	05
	Registrar	Adiciona al perfil de usuario la clave “*room*” que representa el identificador único para los servicios de comunicación WebRTC	06
	Registrar	Adiciona al perfil de usuario la clave “*state*” la cual es el estado de conexión o desconexión al servicio.	07
Requisitos para la implementación	Registrar	El acceso al servicio de videollamada será concedido únicamente a usuarios registrados,	08

Componente	Caso de Uso	Requisitos	N° Prueba
de un servicio de videollamada en un entorno IPTV		estos deberán autenticarse antes de poder acceder al servicio	
	Acceder Guía Contenido	Durante el consumo de servicios de IPTV como VoD, el sistema le permite al usuario cambiar su estado entre “activo” e “inactivo” en el servicio de videollamada	09
	Llamar Usuario Remoto	La aplicación permite al usuario iniciar, finalizar y responder videollamadas	10 equivalente a prueba 2
	Ver Conectados	Para el inicio de una videollamada el usuario puede visualizar que usuarios están disponibles. Solo es posible establecer una comunicación si ambos usuarios están activos	11
	Ver Conectados	En una página previa al establecimiento de la llamada, el usuario del servicio puede visualizar la captura de video de la cámara local	12

4.2.2. Especificación de pruebas

Después de la agrupación y asociación de los requisitos y los casos de uso, el siguiente paso es la construcción de un plan de pruebas o batería de pruebas para verificar el cumplimento de los requisitos. En las tablas 5 a 16 son definidos una serie de pasos que deben llevarse a cabo en el sistema desarrollado para corroborar el cumplimiento de las características que son objeto de evaluación, la validación es realizada comparando el resultado esperado descrito en la tabla, que corresponde a los requisitos establecidos para el servicio y el resultado obtenido al ejecutar los pasos en el sistema solución.

Tabla 5 Prueba funcional N°1

Prueba N° 01	
Título	El mecanismo ejecuta la solicitud de los periféricos disponibles en el dispositivo del usuario final
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados” Seleccionar usuario conectado presionando el botón “Llamar”
Resultado esperado	El usuario podrá permitir o negar el uso de los periféricos disponibles en su dispositivo.

Tabla 6 Prueba funcional N°2

Prueba N° 02	
Título	El mecanismo establece canales de comunicación punto a punto (conforme a lo establecido por la API WebRTC) con otros usuarios conectados y su posterior cierre al terminar el proceso de comunicación
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados” Seleccionar usuario conectado presionando el botón “Llamar” Dar clic en el botón “Colgar”
Resultado esperado	Un usuario puede establecer una llamada con otro que esté conectado al sistema. El usuario puede finalizar la llamada establecida previamente

Tabla 7 Prueba funcional N°3

Prueba N° 03	
Título	Las funciones provistas por el mecanismo deben ser ejecutadas dentro de páginas HTML que sirvan como interfaz para el cliente, con el fin de permitir una interacción entre servicios a nivel de presentación
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados” Seleccionar usuario conectado presionando el botón “Llamar”
Resultado esperado	Es presentado al usuario la interfaz del servicio de videollamada en páginas basadas en HTML.

Tabla 8 Prueba funcional N°4

Prueba N° 04	
Título	Establece mensajes propietarios para identificar el estado de la comunicación, los cuales posibilitan el control de la misma
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados” Seleccionar usuario conectado presionando el botón “Llamar” Dar clic en el botón “Colgar”
Resultado esperado	El sistema puede informar al usuario remoto que está siendo llamado. El sistema puede informar al usuario remoto que se ha finalizado la llamada establecida.

Tabla 9 Prueba funcional N°5

Prueba N° 05	
Título	El mecanismo conmuta mensajes de control entre los usuarios conectados al servicio
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados” Seleccionar usuario conectado presionando el botón “Llamar”
Resultado esperado	Se envía un mensaje de notificación al usuario remoto indicándole que ha sido seleccionado para establecer la llamada.

Tabla 10 Prueba funcional N°6

Prueba N° 06	
Título	Adiciona al perfil de usuario la clave “*room*” que representa el identificador único para los servicios de comunicación WebRTC
Pasos	Ingresar a la dirección de la aplicación web Clic en “Registrarse” Completar los campos del formulario. Clic en “Registrarse”

Resultado esperado	El perfil de usuario debe contener el valor de "*room*" asignado automáticamente junto con los demás campos en la base de datos tales como nombre, nombre de usuario y contraseña.

Tabla 11 Prueba funcional N°7

Prueba N° 07	
Título	Adiciona al perfil de usuario la clave "*state*" la cual es el estado de conexión o desconexión al servicio
Pasos	Ingresar a la dirección de la aplicación web Clic en "Registrarse" Completar los campos del formulario. Clic en "Registrarse"
Resultado esperado	El perfil de usuario debe contener el valor de "*state*" asignado automáticamente en la tabla usuario de la base de datos

Tabla 12 Prueba funcional N°8

Prueba N° 08	
Título	El acceso al servicio de videollamada será concedido únicamente a usuarios registrados, estos deberán autenticarse antes de poder acceder al servicio
Pasos	Ingresar a la dirección de la aplicación web Clic en "Iniciar sesión" Escribir alias y clave Clic en "Entrar"
Resultado esperado	Solo los usuarios que se hayan registrado previamente podrán acceder al sistema con el alias y la clave provistas durante el registro.

Tabla 13 Prueba funcional N°9

Prueba N° 09	
Título	Durante el consumo de servicios de IPTV como VoD, el sistema le permite al usuario cambiar su estado entre "activo" e "inactivo" en el servicio de videollamada
Pasos	Ingresar a la dirección de la aplicación web Clic en "Iniciar sesión" Escribir alias y clave Clic en "Entrar" Clic en botón "Offline"
Resultado esperado	El usuario puede modificar su estado de "Offline" a "Online" y viceversa para el servicio de videollamada.

Tabla 14 Prueba funcional N°10

Prueba N° 10	
Título	La aplicación permite al usuario iniciar, finalizar y responder videollamadas
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados” Seleccionar usuario conectado presionando el botón “Llamar” Dar clic en el botón “Colgar”
Resultado esperado	Un usuario puede establecer una llamada con otro que esté conectado al sistema. El usuario puede finalizar la llamada establecida previamente.

Tabla 15 Prueba funcional N°11

Prueba N° 11	
Título	Para el inicio de una videollamada el usuario puede visualizar que usuarios se encuentran disponibles. Solo es posible establecer una comunicación si ambos usuarios se encuentran activos
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados”
Resultado esperado	El usuario puede visualizar quienes además de él se encuentran conectados al servicio de videollamada.

Tabla 16 Prueba funcional N°12

Prueba N° 12	
Título	En una página previa al establecimiento de la llamada, el usuario del servicio puede visualizar la captura de video de la cámara local.
Pasos	Ingresar a la dirección de la aplicación web Clic en “Iniciar sesión” Escribir alias y clave Clic en “Entrar” Clic en “Ver Conectados”
Resultado esperado	El usuario podrá visualizar la captura de su cámara antes de iniciar la llamada

4.2.3. Resultados obtenidos

En esta sección son consignados los resultados obtenidos para cada caso de uso al ejecutar los pasos correspondientes, como resultado de algunas pruebas son agregadas imágenes de apoyo tomadas de la interfaz gráfica construida para el sistema desarrollado, también son adicionadas capturas de tráfico para corroborar el flujo de información entre puntos o entre un punto y el servidor.

- **Prueba N°1:**

Como resultado de esta prueba son obtenidas las siguientes capturas de pantalla en donde se observa como el navegador pregunta al usuario si desea compartir sus periféricos, en este caso cámara y micrófono, con la aplicación descargada.

Figura 22: Mensaje de petición de periféricos

Debido a que el usuario tiene la posibilidad de aceptar o negar la utilización de sus periféricos, la prueba **N°1** es cumplida.

- **Prueba N°2:**

Después de identificarse ante el sistema dar clic en el botón “Ver Conectados”, es presentada al usuario una ventana similar a la expuesta en la *figura 23*.

Figura 23: Interfaz Ver Conectados, opción de llamar

En la anterior imagen es mostrado, enmarcado en un cuadro rojo, el botón “Llamar” que le permite al usuario conectado iniciar el proceso para el establecimiento de una videollamada con el usuario remoto que desee, para este ejemplo el usuario remoto es “johan”.

Después de establecida la comunicación el usuario podrá consumir el servicio de videollamada en una página web similar a la presentada a continuación.

Figura 24: Interfaz WebRTC, opción de colgar interfaz "WebRTC"

Remarcado por un cuadro rojo en la imagen anterior es posible observar el botón “Colgar”, este le permite al usuario dar inicio al proceso de finalización de la comunicación establecida y retornar a la actividad que realizaba sobre el sistema antes de iniciar el proceso de comunicación.

Con la presentación de las dos acciones anteriormente descritas la prueba **N°2** es cumplida.

- **Prueba N°3:**

Con la selección de Java EE en la sección *3.2.3*, es establecido el uso de páginas web construidas con XHTML para la presentación del sistema desarrollado. Para verificar el uso de este lenguaje es posible observar en la parte superior de la *figura 26*, en la URL de la página “conectados” se puede detallar el uso de la extensión “xhtml” al final del nombre de la página, lo que corrobora la utilización del lenguaje XHTML para su construcción.

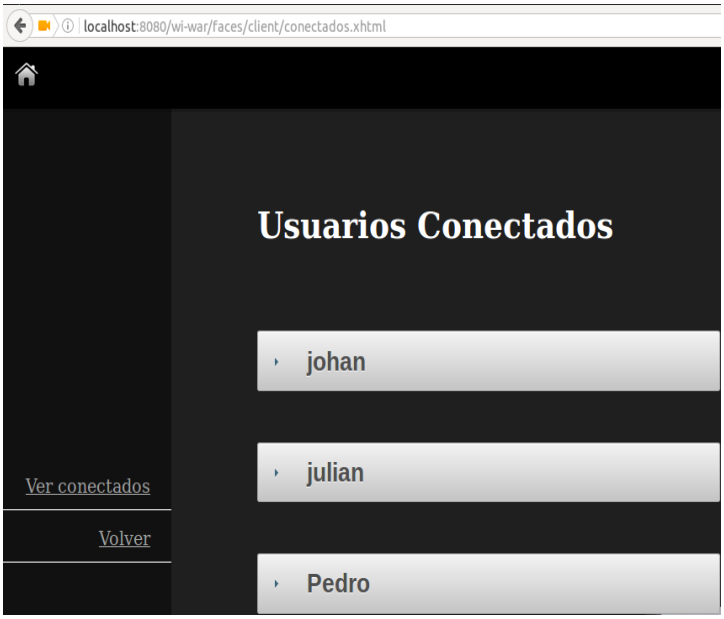

Figura 25: Interfaz Ver Conectados, URL con extensión XHTML

Habiendo verificado por medio de la interfaz mostrada en la imagen anterior el uso del lenguaje XHTML, es posible concluir que la prueba **N°3** es cumplida.

- **Prueba N°4:**

La prueba N°4 refleja una de las acciones ejecutadas por el sistema durante el establecimiento o finalización de una llamada. En las siguientes imágenes (Ver Ilustraciones 27 y 28) es posible observar los mensajes propietarios definidos para el establecimiento y la finalización de la llamada, estos mensajes son enviados desde y hacia el servidor Node.js. Para efectos de la prueba se adicionó una sección de código que permite organizar e imprimir los mensajes transmitidos.

```
root@Machine:/home/julian/wi/wi-war/web/client/webrtc# node server.js
listening on *:3000
Usuario Conectado
Usuario Conectado
Solicitud de Llamada
De:
175cjdelgado
Para:
840johanmos
Llamada:
activa
```

Figura 26: Mensajes Propietarios, solicitud de inicio de llamada

```
Solicitud de Llamada
De:
175cjdelgado
Para:
840johanmos
Llamada:
inactiva
```

Figura 27: Mensajes Propietarios, Informe finalización de llamada

Ambas imágenes presentan un patrón de mensajes similar, inicia con la descripción de quien efectúa la notificación y a quien intenta notificar, en el ejemplo mostrado el usuario con el campo “room = 175cjdelgado” envía un mensaje de notificación al usuario con el campo “room = 840johanmos”; después de identificar los actores es mostrada la intención de la notificación, el sistema solo necesita transmitir dos tipos de mensajes por medio del servidor Node.js, el primero “activa” indica la intención de establecer una comunicación con el usuario quien será notificado y el segundo “inactiva” notifica la terminación de la comunicación.

Con la definición e implementación de mensajes propietarios que permitan establecer y terminar una comunicación entre usuarios conectados es cumplida la prueba **N°4**.

- **Prueba N°5:**

En la *figura 29* se visualiza cuando un usuario estaá navegando en la guía de contenidos y recibe una notificación de llamada entrante, la cual se muestra encerrada en el círculo n°1; es importante notar que un usuario puede ser llamado si y solo sí este se encuentra conectado a la aplicación WebRTC como es mostrado en el círculo n°2.

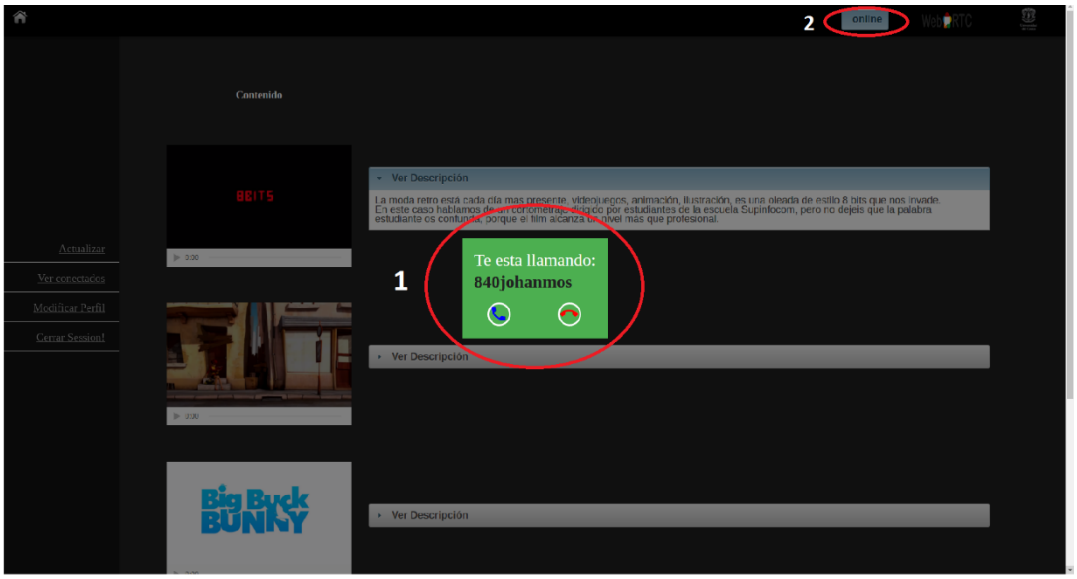

Figura 28: Interfaz principal, notificación llamada entrante

Por otro lado, en la *figura 30* es posible visualizar los paquetes recibidos cuando es recibida una notificación de llamada entrante.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	192.168.0.102	192.168.0.101	TCP	133	3000→52809 [PSH, ACK] Seq=1 Ack=1 Win=239 Len=79
2	0.00006400	192.168.0.101	192.168.0.102	TCP	54	52809→3000 [ACK] Seq=1 Ack=80 Win=255 Len=0

Figura 29: Paquetes del usuario llamado

El paquete N°1 es el enviado por el servidor Node.js notificando la solicitud de llamada, en él van incluidas dos banderas PSH (*Push*) y ACK (*Acknowledgement*), mostradas en la siguiente ilustración (ver *figura 31*), la primera es usada para forzar a un envío de datos tan pronto como sea posible, es decir, que el receptor cuando identifique esta bandera tiene que responder de manera inmediata.

```
⊞ Internet Protocol Version 4, Src: 192.168.0.102 (192.168.0.102), Dst: 192.168.0.101 (192.168.0.101)
⊟ Transmission Control Protocol, Src Port: 3000 (3000), Dst Port: 52809 (52809), Seq: 1, Ack: 1, Len: 79
    Source Port: 3000 (3000)
    Destination Port: 52809 (52809)
    [Stream index: 0]
    [TCP Segment Len: 79]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 80    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  ⊟ .... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 239
    [Calculated window size: 239]
    [Window size scaling factor: -1 (unknown)]
```

Figura 30: Paquete capturado N°1

En la siguiente imagen (ver *figura 32*) es mostrada el paquete capturado N°2 que representa el acuse de recibo (ACK), enviado desde el equipo del usuario llamado hasta el servidor Node.js para indicar que el paquete N°1 ha sido recibido con éxito.

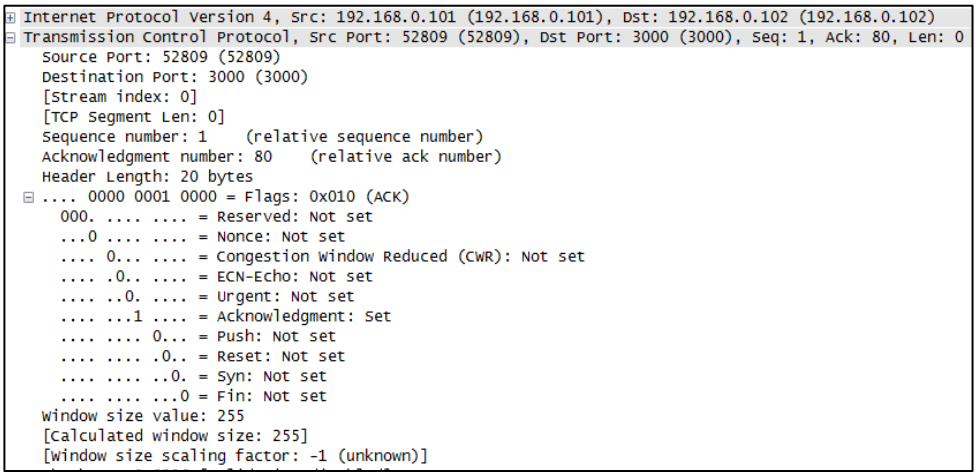

Figura 31: Paquete capturado N°2

Con lo expuesto anteriormente es posible inferir que la notificación de llamada entrante a un usuario remoto es realizada con el protocolo TCP/IP, esto garantiza que la notificación sea realizada exitosamente y además que sea confirmada su llegada en el menor tiempo posible al ser usada la bandera PSH.

Por lo anterior se concluye que la prueba **N°5** es cumplida satisfactoriamente por el sistema solución.

- **Prueba N°6:**

La información de usuarios agregados en la base de datos contiene el campo "*room*", representa un identificador único, el cual es agregado automáticamente cuando un usuario consiga registrarse en el sistema, en la *figura 33* es posible ver una porción de código donde es agregado este campo al objeto del nuevo usuario que posteriormente será guardado en la base de datos.

```
public String createUser(){
   int num= (int) (Math.random()*1000+1);
   newUser.setRoom(num+newUser.getLogin());
```

Figura 32: Código registro nuevo usuario

En la siguiente imagen (ver *figura 34*), es mostrado encerrado en el recuadro rojo la columna que guarda la información previamente mencionada.

id	name	login	password	profile	room	state 0: Desconectado
1	Administrador	admin	YWRtaW4=	1	admin	0
16	johan	johanmos	am9oYW4=	0	840johanmos	1
17	julian	cjdelgado	ZGVsZ2Fkbw==	0	175cjdelgado	1
18	Pedro	pedro	cGVkcm8=	0	350pedro	1
19	Juan	juan	anVhbg==	0	975juan	1

Figura 33: Base de datos perfil de usuario, adición columna "room"

Al verificar en la base de datos que el campo "*room*" ha sido agregado al perfil de usuario se da por cumplida la prueba **N°6**.

- **Prueba N°7:**

Para esta prueba se realiza un proceso similar al anterior, en donde previo al registro en la base de datos se agrega un atributo al perfil del usuario llamado "*state*", el cual representa una variable que indica la disponibilidad de un usuario cuando este haga uso del sistema, este parámetro es muy importante en la implementación de la lógica de los servicios.
En la *figura 35* es visualizado el registro hecho durante el almacenamiento de los usuarios del sistema.

id	name	login	password	profile	room	state 0: Desconectado
1	Administrador	admin	YWRtaW4=	1	admin	0
16	johan	johanmos	am9oYW4=	0	840johanmos	1
17	julian	cjdelgado	ZGVsZ2Fkbw==	0	175cjdelgado	1
18	Pedro	pedro	cGVkcm8=	0	350pedro	1
19	Juan	juan	anVhbg==	0	975juan	1

Figura 34: Base de datos perfil de usuario, adición columna "state"

Al igual que para la prueba anterior mediante la base de datos es garantizado el cumplimiento satisfactorio de la prueba **N°7**.

- **Prueba N°8:**

En esta prueba es requerida la verificación de la gestión de usuarios realizada en el sistema, con el objetivo de dar cumplimiento a los lineamientos establecidos para la implementación de servicios IPTV. En la *figura 36* es mostrado el modo por el cual un usuario, a través de un formulario, puede registrarse ante el proveedor de los servicios, su información será guardada en la base de datos y con ella es posible hacer la posterior identificación que permite su acceso al sistema.

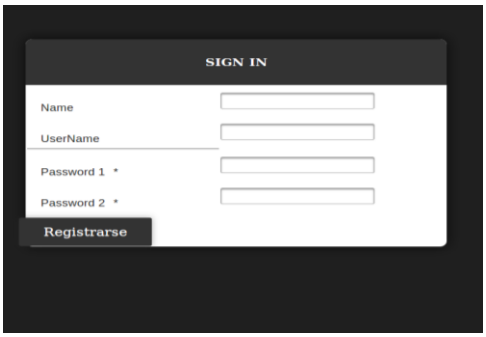

Figura 35: Interfaz de registro

Por otro lado, la interfaz mostrada en la *figura 37* de ingreso al sistema, es usada para que el actor ingrese su alias (nombre de usuario) y contraseña registrados previamente en la base de datos.

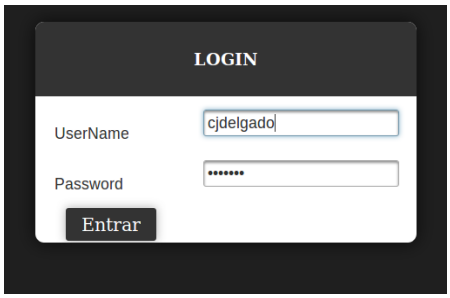

***Figura* 36: Interfaz de identificación**

Con lo anterior, es posible evidenciar que el sistema construido permite el acceso al sistema solo a usuarios registrados, dando por cumplida la prueba **N°8**.

- **Prueba N°9:**

El sistema construido le permite al usuario interactuar con los diferentes servicios, una muestra de esto es el modo como el usuario puede cambiar su estado de disponibilidad, para el servicio de videollamada, desde la interfaz principal del sistema (ver *figura 22*).

En la *figura 38* y *39* es mostrado el botón dispuesto para cambiar el estado de disponibilidad, concluyendo que la prueba **N°9** es cumplida a cabalidad por el sistema construido.

Figura 37: Interfaz principal, estado inactivo para videollamada

Figura 38: Interfaz principal, estado activo para videollamada

- **Prueba N°10:**

Dada la estrecha relación entre el mecanismo propuesto y el servicio de videollamada, la característica utilizada en la prueba **N°10** es una particularización de la característica usada en la prueba **N°2**, es decir, a partir de lo demostrado en esa prueba con la implementación de las funciones es posible comprobar que un usuario puede iniciar y finalizar una llamada cuando accede al sistema.

- **Prueba N°11:**

En la figura 26 es posible evidenciar el modo en el cual un consumidor de los servicios del sistema solución puede consultar a otros usuarios del sistema que están activos (función comprobada en Prueba N° 9), el establecimiento de una comunicación es exclusivamente para aquellos que se encuentren conectados, con ello es posible dar cumplimiento al requerimiento y el resultado de la prueba fue el esperado.

- **Prueba N°12:**

En el sistema solución se implementó la visualización de la cámara web en vista previa, con la cual se tenía previsto dar cumplimiento al requerimiento especificado, en la *figura 40* es posble evidenciar lo dicho, y con ella inferir que el resultado de la prueba N°12 es satisfactorio.

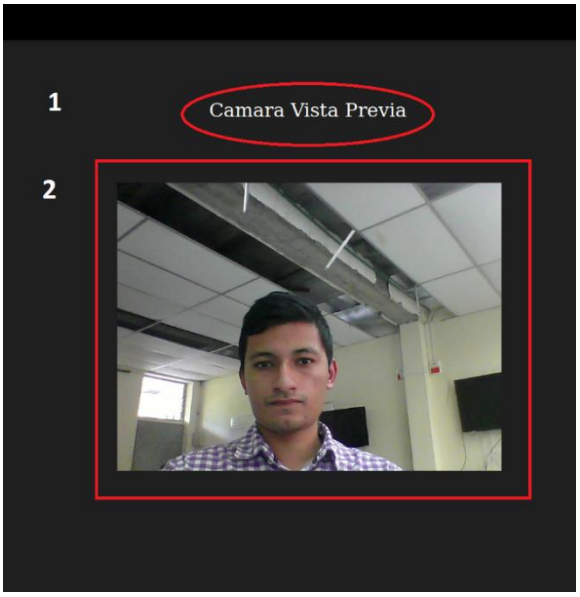

Figura 39: Interfaz Ver conectados, cámara vista previa

Dado el resultado de las pruebas funcionales descritas anteriormente, es viable concluir que tanto el mecanismo propuesto como el servicio de videollamada utilizado como caso de estudio satisfacen las características consideradas en el capítulo 2 como requisitos de un entorno IPTV.

4.3. PRUEBA DE DESEMPEÑO

El mecanismo implementado en el marco de esta investigación es puesto a prueba para determinar su comportamiento en situaciones adversas, es por ello que los módulos (Conmutador, Funciones WebRTC y Controlador) fueron evaluados en una prueba de

estrés. Las pruebas fueron realizadas mediante funciones JavaScript para los módulos Conmutador y Funciones WebRTC, en el **Anexo D** se muestra las funciones básicas utilizadas para el establecimiento de una comunicación y se especifica el modo de obtención de los datos de medición de cada una de ellas. Para la evaluación del módulo Controlador se realizó un proceso similar con utilidades incluidas en el lenguaje JAVA.

Las pruebas fueron llevadas a cabo en el "Laboratorio de Televisión Digital" de La Universidad del Cauca utilizando un escenario de pruebas compuesto por tres equipos como se puede apreciar en la *figura 41*.

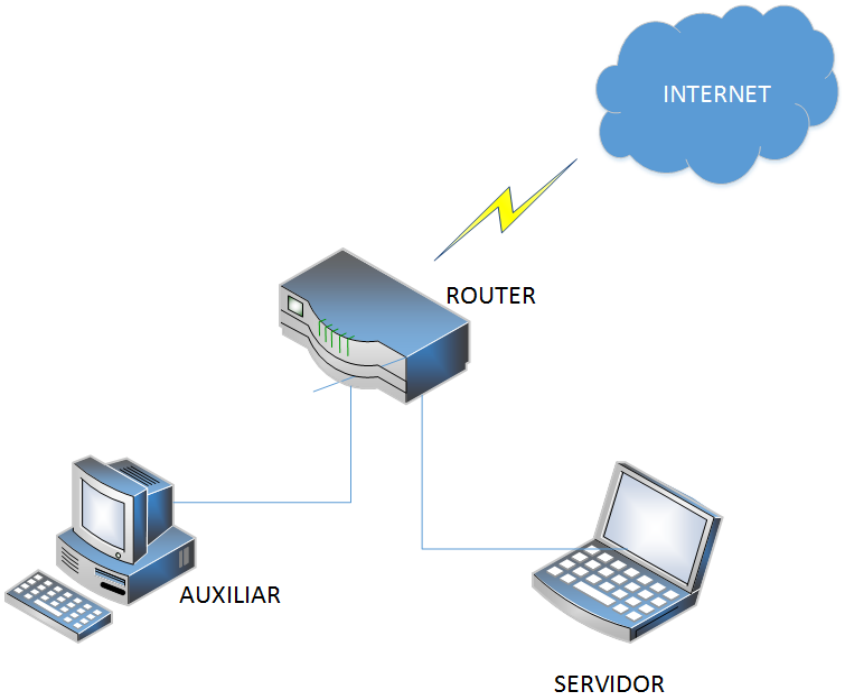

Figura 40: Arquitectura Pruebas de desempeño

El dispositivo "ROUTER" fue configurado para conectar los dispositivos en una red aislada del resto del laboratorio, buscando evitar posibles interferencias externas durante las pruebas, sin embargo, fue necesario conectar el "ROUTER" a internet ya que el módulo Funciones WebRTC requiere de servidores de señalización y librerías externas; las características de este son:

- Router Tenda W316R
- Wireless-N 150 Mbps
- Cable UTP Cat 5e 100Mbps

El equipo "Servidor" fue destinado para el almacenamiento y ejecución del sistema desarrollado, albergando también los tres módulos del mecanismo puesto a prueba, las características del equipo portátil son:

- Sistema operativo Ubuntu 14.04 LTS x64
- Procesador Intel Core i5 2.6 GHz

- Memoria RAM 6GB

Por último, el equipo "Auxiliar" utilizado como "*tester*[16]" del mecanismo, encargado de ejecutar las peticiones y almacenar los datos resultantes de las pruebas cuenta con las siguientes características:

- Sistema operativo Ubuntu 14.04 LTS x64
- Procesador Inter Core i7 3.4 GHz
- Memoria RAM 4GB

El proceso de mapeo gráficamente de los datos obtenidos fue realizado con la herramienta libre SCiDAVis [82].

4.3.1. Módulo Conmutador

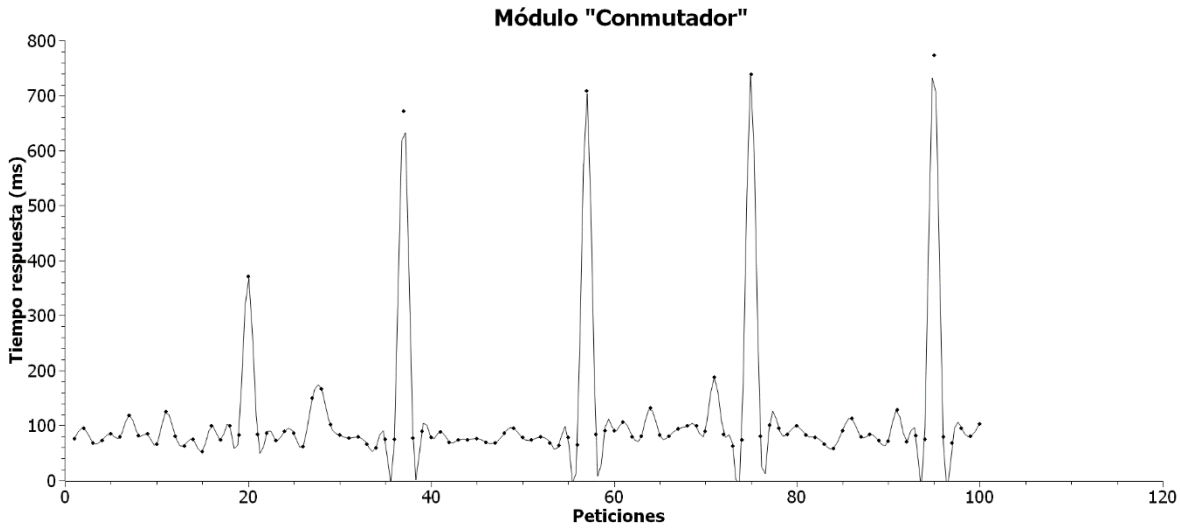

Figura 41: Peticiones Bloque "Conmutador"

El módulo conmutador, encargado de reenviar información entre usuarios, es puesto a prueba realizando hasta 100 peticiones secuenciales desde el cliente y cronometrando el tiempo que tarda en ir y volver una petición, la petición enviada desde el cliente hacia el conmutador lleva un mensaje comúnmente utilizado en el establecimiento de la comunicación y es enviado a un cliente receptor idéntico al emisor, es decir, el mensaje enviado retorna a la fuente.

Los resultados obtenidos durante la prueba son consignados en la *figura 42*, en donde se observa un tiempo de respuesta estable entre los 50 y 150 milisegundos, durante una hipotética puesta a producción del mecanismo, a este tiempo se le deberá sumar el retardo que genere la red al enviar la petición desde el cliente local hasta el cliente remoto.

4.3.2. Módulo Funciones WebRTC

[16] *Tester*: Dispositivo externo al sistema, utilizado para ejecutar el plan de pruebas sin invadir los equipos sobre los cuales se despliega la solución desarrollada.

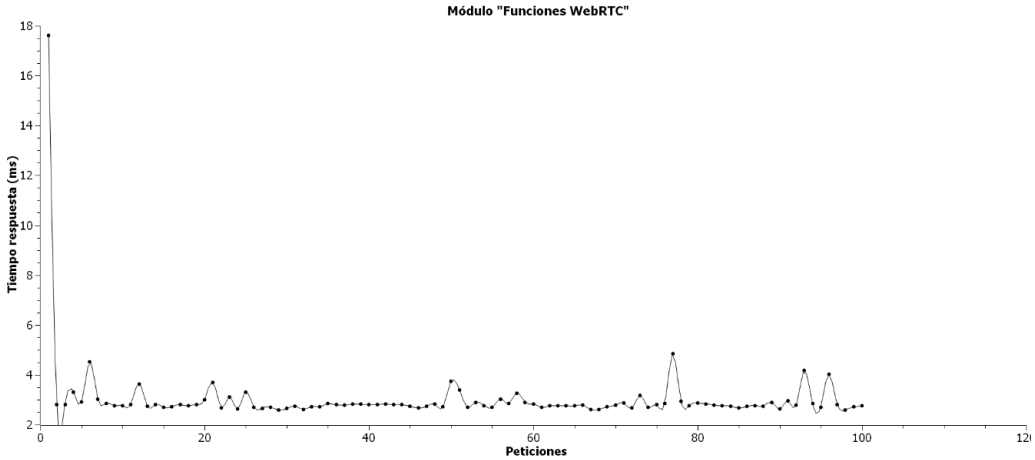

Figura 42: Peticiones Bloque "Funciones WebRTC"

La agrupación de las Funciones WebRTC fue evaluada por medio de ejecuciones secuenciales de los métodos utilizados para el establecimiento de una comunicación, para realizar este ejercicio fue necesario construir una nueva página web dedicada a la ejecución de las funciones, también fue necesario establecer como llamante y llamado al mismo usuario, eliminando así la interacción humana durante la prueba.

La primera petición realizada tardó aproximadamente 18 milisegundos, ya que debe descargar la librería construida por "*WebRTC Project*", después de tener este archivo localmente los tiempos de respuesta de las funciones WebRTC oscilan entre los 2 y 4 milisegundos, como es apreciado en la *figura 43*, la prueba fue realizada en la configuración mostrada en la figura 41.

El tiempo de procesamiento tan bajo obtenido con las funciones WebRTC permite inferir que este bloque no será un limitante para la prestación de los servicios de comunicación en tiempo real que dispongan del mecanismo propuesto.

4.3.3. Módulo Controlador

El módulo controlador agrupa las funciones lógicas para el inicio y terminación de los servicios implementados con base en el mecanismo desarrollado. Debido a que estos dos

procesos se realizan por separado, fue necesario probar cada uno independientemente, dando como resultado las ilustraciones *44 y 45*.

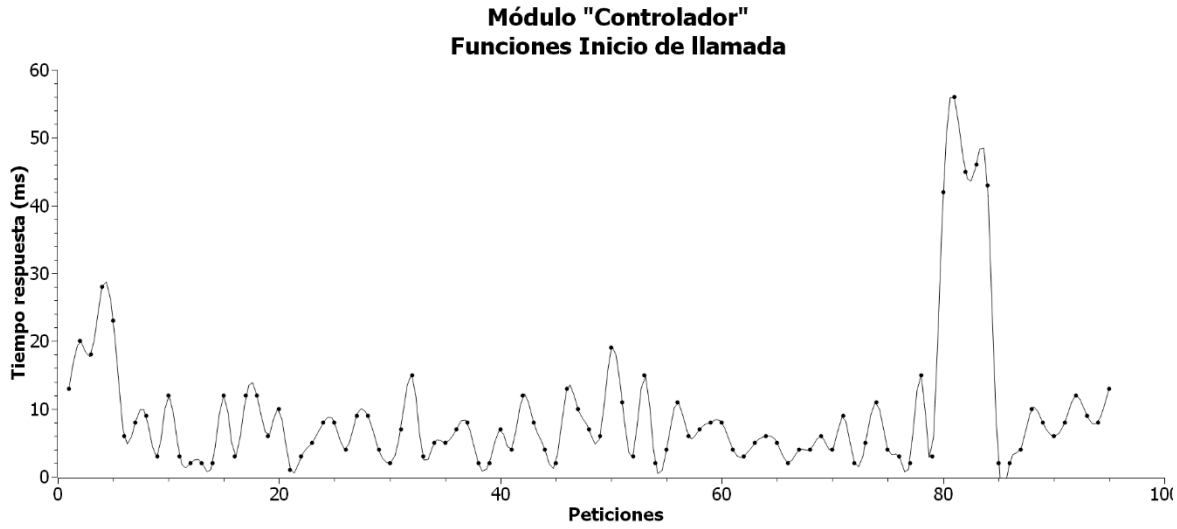

Figura 43: Peticiones Bloque "Controlador", Inicio de llamada

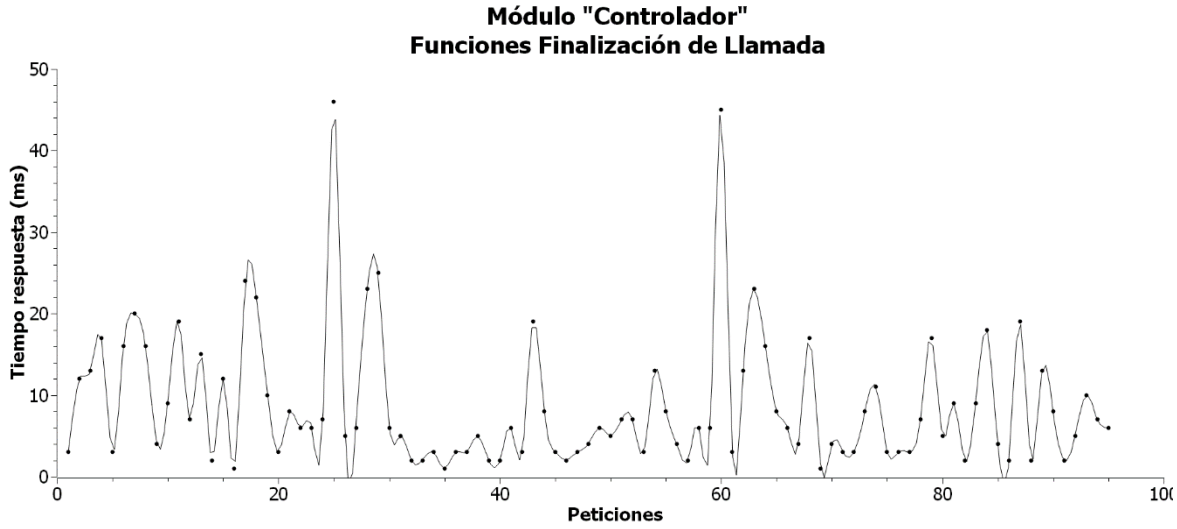

Figura 44: Peticiones Bloque "Controlador", Finalización de llamada

Los resultados obtenidos con la prueba del módulo controlador tienen una mayor variación que las pruebas anteriores, entre los 2 y los 60 milisegundos con una mayor concentración de datos alrededor de los 10 milisegundos.

Durante el funcionamiento normal en el sistema desarrollado, cada módulo es ejecutado secuencialmente, iniciando el Controlador y finalizando con el Conmutador, esto para el inicio de una llamada; por lo que los tiempos anteriormente cronometrados deben sumarse dando un tiempo promedio de 91,27 milisegundos para la ejecución del sistema desarrollado, cabe resaltar que este tiempo está dentro de los parámetros considerados como aceptables para la comunicación en tiempo real establecidos por la ITU-T y definidos en la sección 2.4.3.

CAPITULO V

CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo de grado fue presentado el potencial de la tecnología WebRTC y una nueva perspectiva de su implementación, al ser incluido en un entorno IPTV, con el fin de lograr enriquecer los servicios de comunicación en tiempo real.

Para verificar el correcto funcionamiento del sistema solución implementado, el cual incorpora el mecanismo propuesto, fueron realizadas dos tipos de pruebas.
La primera con el propósito de constatar el correcto funcionamiento de los servicios implementados con base en las funciones WebRTC y los requisitos establecidos para el entorno IPTV.
La segunda prueba realizada fue necesaria para determinar cuan capaz es el sistema para su implementación en entornos reales.

A continuación, son presentadas las conclusiones que resultaron de la investigación y desarrollo de este trabajo de grado. Posteriormente son propuestos trabajos futuros que según nuestro criterio permitirán nuevas investigaciones en temas relacionados.

5.1. CONCLUSIONES

A partir del trabajo y la investigación realizada, así como también de la experiencia adquirida durante el proceso, fueron planteadas las siguientes conclusiones:

1. La inclusión de la tecnología WebRTC en un entorno IPTV evidencia un impacto positivo en los servicios de comunicación en tiempo real. Esto se debe a las características de la tecnología WebRTC como la comunicación punto a punto, la utilización de software libre, la eliminación de plugins y la ejecución en navegadores web evitando la instalación de clientes. Dichas mejoras son validadas al mantener estos servicios de comunicación dentro de los límites definidos por la ITU-T en la sección 2.4.3.2 para la comunicación en tiempo real.
2. El mecanismo propuesto y el servicio de videollamada desarrollado como caso de estudio, cumplieron con las características asignadas en las secciones 2.5 y 2.7 respectivamente para la satisfacción de los requisitos inherentes al entorno IPTV. Validado por medio de la evaluación funcional la propuesta de un mecanismo que permita enriquecer los servicios en tiempo real en un entorno IPTV, y que además encaja dentro de los límites establecidos por la ITU-T como fue mostrado en las pruebas.
3. Los navegadores Google Chrome y Firefox Mozilla presentan un soporte adecuado de las funciones y protocolos necesarios para la implementación de los servicios de WebRTC e IPTV. Lo anterior fue inferido a partir del proceso de diseño del sistema solución y validado con el establecimiento de una comunicación punto a punto haciendo uso de estos navegadores.
4. El servicio de videollamada desarrollado con base en el mecanismo propuesto cumple a cabalidad los requisitos establecidos por el OIPF para la categoría “Servicios de Comunicación”, demostrando de esta manera que un servicio de comunicación construido con la tecnología WebRTC puede incluirse en un entorno IPTV.

5. El perfil abierto de internet permite la inclusión de aplicaciones web que enriquecen los servicios típicos en IPTV, dándole un valor agregado a los proveedores y acrecentando su portafolio de servicios.
6. Incluir WebRTC en una plataforma como IPTV, conlleva a un mejoramiento en las aplicaciones y/o servicios prestados por WebRTC. Debido a que IPTV proporciona características de gestión y manejo de información las cuales no son incluidas en las especificaciones de WebRTC, es posible evidenciar esta mejoría en la gestión centralizada del sistema solución que además de soportar el servicio de VoD presenta características de administración para el servicio de videollamada basado en WebRTC.
7. El mecanismo propuesto presenta una reducción de costos y tiempo en su construcción, convirtiéndose en una solución atractiva para ser implementada en entornos reales. Por la utilización de herramientas y software libre comúnmente utilizados en desarrollos web.
8. Ya que WebRTC no define un mecanismo de señalización, permite que los desarrollos en este tipo de tecnologías utilicen el mecanismo de señalización y los protocolos adecuados que mejor puedan amoldarse al entorno donde es requerida la implementación de la solución. El mecanismo de señalización WebSockets mostró ser una buena alternativa para el establecimiento de una comunicación, además esta superioridad fue afirmada por su uso frecuente en proyectos WebRTC como es el caso del proyecto guía seleccionado.
9. El uso de la herramienta híbrida Socket.io ofrece una mejor disponibilidad y prestación de los servicios de comunicación en tiempo real a los usuarios. Este beneficio es conseguido al tener mecanismos de señalización como respaldo que puedan actuar en el caso que WebSockets falle.
10. Es posible desarrollar servicios de comunicación en tiempo real sobre IPTV usando la tecnología WebRTC y que sean compatibles con VoIP, diversos clientes SIP y la red telefónica conmutada. La versatilidad de WebRTC para ser integrado con sistemas de comunicación ya existentes permite este tipo de mejoras, un ejemplo práctico de esta cualidad es evidenciada en los proyectos desarrollados por Telestax Inc.
11. WebRTC permite crear aplicaciones escalables, proporcionando ventajas en torno a la reducción de costos y experiencia de usuario, mejorando así la prestación de servicios reduciendo la carga en los servidores. Lo anterior se debe a la implementación de los desarrollos WebRTC en arquitecturas P2P.
12. Los servicios de comunicación en tiempo real construidos con la tecnología WebRTC pueden interactuar en el cliente con otros servicios de la plataforma IPTV. Esto fue validado por medio de los resultados obtenidos del plan de pruebas, diseñado para la validación de funcionalidades del mecanismo propuesto en un entorno IPTV.
13. Los operadores de servicio de IPTV pueden aumentar su portafolio de servicios integrando tecnologías como WebRTC, agregando nuevas formas de satisfacer a los usuarios con servicios en tiempo real, económicos y muy solicitados.
14. La metodología para la descripción de sistemas intensivos de software escogida permitió diseñar la arquitectura del mecanismo y del sistema solución en general. Posibilitando el cumplimiento de los requisitos establecidos para el entorno IPTV y además adicionar características a la tecnología WebRTC.
15. El aprendizaje obtenido durante el desarrollo del presente trabajo sirve como guía en la selección de requisitos y asignación de características para la integración y enriquecimiento de servicios en tiempo real en un entorno IPTV basándose en la tecnología WebRTC y además como como punto de partida para futuras investigaciones.

5.2. TRABAJOS FUTUROS

Con base en los resultados obtenidos y la investigación realizada durante la elaboración de este trabajo se presentan los posibles trabajos futuros que pueden ayudar a implementar nuevos y mejores servicios y/o sistemas.

1. Utilización del Core IMS para la implementación de la plataforma IPTV usando perfiles de administración. Para llevar a cabo esto es necesario cambiar las tecnologías para el desarrollo de la solución y el mecanismo de señalización usado por WebRTC, una posible alternativa seria SIP sobre WebSockets.
2. Implementación de la tecnología WebRTC usando distintos mecanismos de señalización en entornos de IPTV para establecer cuál de ellos es más útil en determinados casos y facilitar así su inclusión en futuras implementaciones.
3. Proponer un modelo de plataforma de Streaming de video online usando tecnologías como WebRTC.
4. Usar la tecnología WebRTC para enriquecer otras categorías de servicios en tiempo real del entorno IPTV como por ejemplo la categoría “Contenido bajo demanda”.
5. Utilizar un perfil de administración para IPTV, donde se toma en consideración la QoS, con el objetivo de evaluar el impacto que tiene WebRTC en la calidad de los servicios de comunicación en tiempo real sobre este entorno.

CAPITULO VI

REFERENCIAS

[1] “Lo más destacado de El mundo en 2013: datos y cifras relativos a las TIC - El uso de Internet – la población en línea", Itunews.itu.int, 2016. [Online]. Available: https://itunews.itu.int/es/3781-Lo-mas-destacado-de-El-mundo-en-2013-datos-y-cifras-relativos-a-las-TIC.note.aspx. [Accessed: 27- Nov- 2014].

[2] Fedesarrollo, "Análisis de proceso de integración y convergencia en telecomunicaciones: experiencia internacional, el caso colombiano", 2011.

[3] INICTEL-UNI, 'Sistema de Comunicaciones IPTV para Redes de Investigación', Lima, 2009. [Online]. Available: http://aat.inictel-uni.edu.pe/files/AAT-03_Sistema_IPTV_para_Investigacion.pdf

[4] Power More, 'Future of real-time communication: Cloud, mobile, and video', 2015. [Online]. Available: https://powermore.dell.com/technology/cloud-collaboration-and-the-future-of-ad-hoc-real-time-communication/.

[5] Laboratorio Nacional de Calidad del Software de INTECO, “Ingeniería Del Software: Metodologías Y Ciclos De Vida”, 1st ed. 2009. [Online] Available: http://datateca.unad.edu.co/contenidos/301569/guia_de_ingenieria_del_software.pdf

[6] "draft-ietf-rtcweb-overview-16 - Overview: Real Time Protocols for Browser-based Applications", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtcweb-overview-16.

[7] S. Loreto and S. Romano, Real-time communication with WebRTC. Sebastopol, CA: O'Reilly Media, 2014.

[8] "WebRTC Home | WebRTC", Webrtc.org, 2016. [Online]. Available: https://webrtc.org. [Accessed: 13- Sep- 2016].

[9] M. Merino, "¿Qué es una API y para qué sirve?", TICbeat, 2014. [Online]. Available: http://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/.

[10] B. Sredojev, D. Samardzija and D. Posarac, "WebRTC technology overview and signaling solution design and implementation," Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on, Opatija, 2015.

[11] "draft-ietf-rtcweb-overview-15 - Overview: Real Time Protocols for Browser-based Applications", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtcweb-overview-15.

[12] GSM Association, WebRTC to complement IP Communication Services, V 1.0 216.

[13] Adham Zeidan, Armin Lehmann, Ulrich Trick, “WebRTC enabled multimedia conferencing and collaboration solution.” Research Group for Telecommunication Networks, University of Applied Sciences Frankfurt am Main, Germany

[14] "Frequent Questions | WebRTC", Webrtc.org, 2016. [Online]. Available: https://webrtc.org/faq/. [Accessed: 01- Nov- 2016].

[15] “the definition of dongle”, Dictionary.com2016. [Online]. Available: http://www.dictionary.com/browse/dongle. [Accessed: 17- Sep- 2016]

[16] "Chromecast", Google.es, 2016. [Online]. Available: https://www.google.es/chrome/devices/chromecast/. [Accessed: 01- Nov- 2016].

[17] *WebRTC for Business People*, 2014. Available: https://bloggeek.me/webrtc-business-people/

[18] I. Grigorik, High-performance browser networking. Sebastopol, CA: O'Reilly, 2013.

[19] WebRTC 1.0: Real-time Communication Between Browsers", W3.org, 2016. [Online]. Available: https://www.w3.org/TR/webrtc/#intro. [Accessed: 17- Sep- 2016].

[20] “What is real-time communications (RTC)? - Definition from WhatIs.com", SearchUnifiedCommunications, 2016. [Online]. Available: http://searchunifiedcommunications.techtarget.com/definition/real-time-communications. [Accessed: 17- Sep- 2016].
[21] "P2P (Peer To Peer) Definition", Techterms.com, 2016. [Online]. Available: http://techterms.com/definition/p2p. [Accessed: 01- Nov- 2016].
[22] Gabriela Alexandra Coppiano Marin, “Análisis de técnicas para atravesar NAT/FIREWALLS en una red extremo a extremo” Universidad Politécnica de Madrid, 2011.
[23] "RFC 5389 - Session Traversal Utilities for NAT (STUN)", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/rfc5389. [Accessed: 17- Sep- 2016].
[24] "RFC 5766 - Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/rfc5766. [Accessed: 19- Sep- 2016].
[25] "RFC 5245 - Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/rfc5245#page-7. [Accessed: 22- Sep- 2016].
[26] "Using WebRTC data channels", Mozilla Developer Network, 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Using_data_channels [Accessed: 19- Sep- 2016].
[27] "Lifetime of a WebRTC session", Mozilla Developer Network, 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Session_lifetime. [Accessed: 19- Sep- 2016].
[28] Altanai., WebRTC integrator's guide. Birmingham, UK: Packt Pub., 2014.
[29] R. Manson, Getting started with WebRTC. Birmingham, UK: Packt Pub., 2013.
[30] "Buscar en el diccionario informático", Lawebdelprogramador.com, 2016. [Online]. Available: http://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=codec. [Accessed: 01- Nov- 2016].
[31] "draft-ietf-rtcweb-jsep-00 - JavaScript Session Establishment Protocol", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtcweb-jsep-00. [Accessed: 19- Sep- 2016].
[32] OIPF, “Overview Volume 1 V 2.3”, OIPF, 2014.
[33] D. Durán Dorado and J. Arciniegas, "Arquitectura para el Despliegue del servicio de Video bajo Demanda de IPTV, apoyada en Interactividad y Sistemas de Recomendaciones", ITECKNE, vol. 10, no. 1, 2013.
[34] OIPF, “OIPF Profiles V 2.0”, OIPF, FRANCE, 2014.
[35] P. Support, C. Firewalls and C. Guides, "Cisco Security Appliance Command Line Configuration Guide, Version 8.0 - Glossary [Cisco ASA 5500-X Series Firewalls]", Cisco, 2016. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/security/asa/asa80/configuration/guide/conf_gd/glossary.html#wp1022899. [Accessed: 19- Sep- 2016].
[36] OIPF, “Services and Functions V1.0”, OIPF, FRANCE, 2009 p5.
[37] OIPF, "Service and Platform Requirements V2.0," OIPF, Francia, 2009.
[38] P. Pérez, J. J. Ruiz, and N. García, “Calidad de Experiencia en servicios multimedia sobre IP” XX Jornadas Telecom I+D (Telecom I+D 2010), Valladolid, Spain, 27-29 Sep. 2010.
[39] Software Engineering Standards Committee, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," IEEE, New York, IEEE Std 1471-2000, Septiembre 2000.

[40] Ph. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, vol. 6, no. 12, pp. 45-50, 1995.
[41] "Is WebRTC ready yet?", Iswebrtcreadyyet.com, 2016. [Online]. Available: http://iswebrtcreadyyet.com/. [Accessed: 20- Sep- 2016].
[42] "Architecture | WebRTC", Webrtc.org, 2016. [Online]. Available: https://webrtc.org/architecture/. [Accessed: 26- Sep- 2016].
[43] WebRTC: comunicaciones en tiempo real en el navegador Web", Ramonmillan.com, 2016. [Online]. Available: http://www.ramonmillan.com/tutoriales/webrealtimecommunications.php. [Accessed: 01- Nov- 2016].
[44] W. Resources, C. me and T. Levent-Levi, "5 Different Signaling Protocol Options for WebRTC Services", BlogGeek.me, 2014. [Online]. Available: https://bloggeek.me/siganling-protocol-webrtc/. [Accessed: 20- Sep- 2016].
[45] "WebRTC Signaling Protocols and WebRTC Transport Protocols Demystified • BlogGeek.me", BlogGeek.me, 2016. [Online]. Available: https://bloggeek.me/webrtc-signaling-transport/. [Accessed: 01- Nov- 2016].
[46] A. Sergiienko, WebRTC blueprints. Birmingham, UK: Packt Pub., 2014.
[47] "HTML 5 Web Sockets vs. Comet and Ajax", InfoQ, 2016. [Online]. Available: https://www.infoq.com/news/2008/12/websockets-vs-comet-ajax. [Accessed: 20- Sep- 2016].
[48] "RFC 6202 - Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", Tools.ietf.org, 2016. [Online]. Available: https://tools.ietf.org/html/rfc6202#page-4. [Accessed: 20- Sep- 2016].
[49] "Introducing JSON", Json.org, 2016. [Online]. Available: http://www.json.org/json-es.html. [Accessed: 16- Nov- 2016].
[50] A. Atalay, “Signaling Mechanism Design for WebRTC”, ISITES2015 Valencia – Spain, 2015.
[51] "XMPP | WebRTC", *Xmpp.org*, 2016. [Online]. Available: https://xmpp.org/uses/webrtc.html. [Accessed: 20- Sep- 2016].
[52] "XMPP | An Overview of XMPP", *Xmpp.org*, 2016. [Online]. Available: https://xmpp.org/about/technology-overview.html. [Accessed: 20- Sep- 2016].
[53] Understanding Socket.IO", *The Back Channel by NodeSource | The Enterprise Node Company™ Providing Enterprise Node.js Training, Support, Software & Consulting, Worldwide*, 2014. [Online]. Available: https://nodesource.com/blog/understanding-socketio/. [Accessed: 20- Sep- 2016].
[54] "Real time communication with WebRTC", *Codelabs.developers.google.com*, 2016. [Online]. Available: https://codelabs.developers.google.com/codelabs/webrtc-web/#6. [Accessed: 20- Sep- 2016].
[55] OIPF, “Authentication, Content Protection and Service Protection V 2.3”, OIPF, FRANCE, 2014.
[56] OIPF, "OIPF Functional Architecture," OIPF, Munich, 2.3, 2014.
[57] ITU-T "IPTV terminal devices: Basic model “, H.721 04-2015.
[58] *Multimedia y Web 2.0*, "Módulo 4. Video y Animaciones". 2012.
[59] OIP, "Service and Platform Requirements V2.0", OIPF, Francia, 2008.
[60] OIP, "Service and Platform Requirements V2.5," OIPF, Francia, 2008.
[61] StarUML", *Staruml.io*, 2016. [Online]. Available: http://staruml.io/. [Accessed: 20- Sep- 2016].

[62] “Diagramas de actividades UML: Referencia", *Msdn.microsoft.com*, 2016. [Online]. Available: https://msdn.microsoft.com/es-es/library/dd409360.aspx. [Accessed: 21- Sep- 2016].
[63] T. Levent-Levi, "The 4 Different Approaches of Using WebRTC APIs", BlogGeek.me, 2016. [Online]. Available: https://bloggeek.me/approaches-using-webrtc/. [Accessed: 18- Nov- 2016].
[64] T. Levent-Levi, "Media Engine", WebRTC Glossary, 2016. [Online]. Available: https://webrtcglossary.com/media-engine/. [Accessed: 18- Nov- 2016].
[65] M. Bu and E. Zhang, "Simple peer-to-peer with WebRTC", Peerjs.com, 2016. [Online]. Available: http://peerjs.com/. [Accessed: 18- Nov- 2016].
[66] "The fastest way to build your own WebRTC apps", Easyrtc.com, 2016. [Online]. Available: https://easyrtc.com/. [Accessed: 18- Nov- 2016].
[67] "SimpleWebRTC.js from &yet", Simplewebrtc.com, 2017. [Online]. Available: https://simplewebrtc.com/. [Accessed: 08- Feb- 2017].
[68] S. Ålund, "Bowser – The World’s First WebRTC - Enabled Mobile Browser - Ericsson Research Blog", *Ericsson Research Blog*, 2012. [Online]. Available: https://www.ericsson.com/research-blog/context-aware-communication/bowser-worlds-first-webrtc-enabled-mobile-browser/. [Accessed: 20- Sep- 2016].
[69] "java.com: Java y Tú", *Java.com*, 2016. [Online]. Available: https://www.java.com/es. [Accessed: 21- Sep- 2016].
[70]ICM, Indra “Frameworks de desarrollo/Lenguajes de programación", Madrid, España, 2016.
[71] “Lenguaje de Programación Ruby", *Ruby-lang.org*, 2016. [Online]. Available: http://www.ruby-lang.org/es/. [Accessed: 21- Sep- 2016].
[72] R. Nagilla, "Comparison of Web Development Technologies - ASP.NET & PHP", Master, MÄLARDALEN UNIVERSITY SWEDEN.
[73] "The Web framework for perfectionists with deadlines | Django", *Djangoproject.com*, 2016. [Online]. Available: https://www.djangoproject.com/. [Accessed: 21- Sep- 2016].
[74] "All about JavaScript | distribly.com", *Distribly.com*, 2016. [Online]. Available: https://www.distribly.com/help/all-about-javascript.html. [Accessed: 21- Sep- 2016].
[75] M. H. Trejos, D. F. Zamora, “*Criterios de Evaluación de Plataformas de Desarrollo de Aplicaciones Empresariales para Ambientes Web*”, Universidad Tecnológica de Pereira.
[76] “TIOBE Index | TIOBE - The Software Quality Company", *Tiobe.com*, 2016. [Online]. Available: http://www.tiobe.com/tiobe-index/. [Accessed: 21- Sep- 2016].
[77] E. Jendrock, R. Cervera-Navarro, I. Evans, K. Haase and W. Markito, *The Java EE Tutorial*, 1Java Platform, Enterprise Edition, 2014.
[78] MySQL: 10 razones para elegir MySQL para las aplicaciones web de la próxima generación", *Mysql.com*, 2016. [Online]. Available: http://www.mysql.com/why-mysql/white-papers/10-razones-para-elegir-mysql-para-las-aplicaciones-web-de-la-proxima-generacion/. [Accessed: 21- Sep- 2016].
[79] F. Node.js, "Node.js", Nodejs.org, 2016. [Online]. Available: https://nodejs.org/es. [Accessed: 07- Dec- 2016].
[80] "NetBeans IDE 8.1 Installation Instructions Netbeans.org, 2017. [Online]. Available: https://netbeans.org/community/releases/81/install.html. [Accessed: 16- Feb- 2017]. 2017. [Online]. Available: https://netbeans.org/community/releases/81/install.html. [Accessed: 16- Feb- 2017].
[81] S. Mendoza Fariña, "Estudio de la metodología de automatización del testeo en aplicaciones web", Universitat Politècnica de Catalunya (UPC), Jun 2011.
[82] “SciDavis”, [Online]. Available: http://scidavis.sourceforge.net/.