



**Análisis de interactividad para televisión híbrida para el estándar de transmisión de la  
próxima generación ATSC 3.0**

Chitupanta Palomo, Víctor Hugo

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y  
Telecomunicaciones

Ing. Olmedo Cifuentes, Gonzalo Fernando, Ph.D.

11 de agosto del 2021

### Document Information


Analyzed document	TRABAJO-TITULACION-CHITUPANTA.pdf (D111182346)
Submitted	8/11/2021 8:13:00 PM
Submitted by	Olmedo Cifuentes Gonzalo Fernando
Submitter email	gfolmedo@espe.edu.ec
Similarity	5%
Analysis address	gfolmedo.espe@analysis.arkund.com



Formado digitalmente por  
GONZALO FERRANDO  
OLMEDO CIFUENTES

### Sources included in the report

Ing. Olmedo Cifuentes, Gonzalo Fernando Ph.D.

<b>SA</b>	<b>Universidad de las Fuerzas Armadas ESPE / Luis_Medivilla_tesis_final.pdf</b> Document Luis_Medivilla_tesis_final.pdf (D111181181) Submitted by: gfolmedo@espe.edu.ec Receiver: gfolmedo.espe@analysis.arkund.com	 12
<b>W</b>	URL: <a href="https://kupdf.net/download/analisis-de-un-head-end-de-television-digital-para-sistemas-iptv-y7a-ottpdf_5af3175be2b6f5f90b1daef3_pdf">https://kupdf.net/download/analisis-de-un-head-end-de-television-digital-para-sistemas-iptv-y7a-ottpdf_5af3175be2b6f5f90b1daef3_pdf</a> Fetched: 12/6/2020 9:05:42 AM	 1
<b>W</b>	URL: <a href="https://www.mediax.online/mpeg-%C2%AD%E2%80%90-la-solucion-al-problema-de-las-tecnologias-de-streaming-propietarias-para-la-television-online/">https://www.mediax.online/mpeg-%C2%AD%E2%80%90-la-solucion-al-problema-de-las-tecnologias-de-streaming-propietarias-para-la-television-online/</a> Fetched: 2/2/2021 12:00:32 AM	 2
<b>SA</b>	<b>75.572_20201_PEC 2. Las aplicaciones de red, la razón de ser de una red de ordenadores_13339781.txt</b> Document 75.572_20201_PEC 2. Las aplicaciones de red, la razón de ser de una red de ordenadores_13339781.txt (D86170037)	 2
<b>W</b>	URL: <a href="https://res-www.zte.com.cn/mediare/magazine/publication/com_en/article/201601/448973/P020160311295798100137.pdf">https://res-www.zte.com.cn/mediare/magazine/publication/com_en/article/201601/448973/P020160311295798100137.pdf</a> Fetched: 8/11/2021 8:14:00 PM	 3
<b>W</b>	URL: <a href="https://www.atsc.org/wp-content/uploads/2017/12/A331-2019-Signaling-Delivery-Sync-FEC.pdf">https://www.atsc.org/wp-content/uploads/2017/12/A331-2019-Signaling-Delivery-Sync-FEC.pdf</a> Fetched: 8/11/2021 8:14:00 PM	 1
<b>W</b>	URL: <a href="https://dspace.uclv.edu.cu/bitstream/handle/123456789/10201/Yoelvis%20Gonz%C3%A1lez%20P%C3%A9rez.pdf?sequence=1&amp;isAllowed=y">https://dspace.uclv.edu.cu/bitstream/handle/123456789/10201/Yoelvis%20Gonz%C3%A1lez%20P%C3%A9rez.pdf?sequence=1&amp;isAllowed=y</a> Fetched: 6/14/2020 3:56:15 PM	 1
<b>W</b>	URL: <a href="https://www.atsc.org/wp-content/uploads/2018/04/A344S34-306r1-Revision-of-Interactive-Content-Redline.pdf">https://www.atsc.org/wp-content/uploads/2018/04/A344S34-306r1-Revision-of-Interactive-Content-Redline.pdf</a> Fetched: 8/11/2021 8:14:00 PM	 1
<b>W</b>	URL: <a href="https://docplayer.es/amp/1070057-Implementacion-del-protocolo-p2psp-usando-webrtc.html">https://docplayer.es/amp/1070057-Implementacion-del-protocolo-p2psp-usando-webrtc.html</a> Fetched: 1/17/2020 5:07:43 PM	 1
<b>SA</b>	<b>ErickMunoz.docx</b> Document ErickMunoz.docx (D62487701)	 1



**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y  
TELECOMUNICACIONES

### CERTIFICACIÓN

Certifico que el trabajo de titulación, **"Análisis de interactividad para televisión híbrida para el estándar de transmisión de la próxima generación ATSC 3.0"** fue realizado por la señor **Chitupanta Palomo, Víctor Hugo** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 11 de agosto del 2021



Firmado electrónicamente por:  
GONZALO FERNANDO  
OLMEDO CIPUENTES

Ing. Olmedo Cifuentes, Gonzalo Fernando, Ph.D.

C.C.:1711696342



# ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y  
TELECOMUNICACIONES

RESPONSABILIDAD DE AUTORÍA

Yo, **Chitupanta Palomo, Victor Hugo**, con cédula de ciudadanía n° 1721706875, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Análisis de interactividad para televisión híbrida para el estándar de transmisión de la próxima generación ATSC 3.0"** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 11 de agosto del 2021

Chitupanta Palomo, Victor Hugo

C.C.:1721706875



DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y  
TELECOMUNICACIONES

#### AUTORIZACIÓN DE PUBLICACIÓN

Yo, **Chitupanta Palomo, Víctor Hugo**, con cédula de ciudadanía n° 1721706875, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Análisis de interactividad para televisión híbrida para el estándar de transmisión de la próxima generación ATSC 3.0"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 11 de agosto del 2021



Chitupanta Palomo, Víctor Hugo

C.C.:1721706875

## **Dedicatoria**

A Dios en primer lugar por haberme guiado por el camino correcto y ayudado a equilibrar mis pensamientos.

A mis queridos padres José Chitupanta y María Palomo quienes son el eje principal para mí, ya que con su gran amor y apoyo incondicional me han inculcado buenos valores, gracias a eso poder alcanzar mis objetivos.

A mis hermanos Patricio, Luis y Mayte por todo los consejos y cariño que siempre me dieron en los momentos difíciles.

A mis dos sobrinas Dayra y Danna quienes con sus ocurrencias siempre hacían que mis días difíciles sean más fáciles.

A todos mis amigos y compañeros de clases ya que de una u otra forma han contribuido al desarrollo de mis actividades en el transcurso de la universidad.

VÍCTOR HUGO CHITUPANTA PALOMO

## **Agradecimiento**

A la Universidad de las Fuerzas Armadas "ESPE" por todo el conocimiento adquirido, para mi formación profesional.

Al Doctor Gonzalo Olmedo, director de tesis gracias por compartir sus conocimientos, por la confianza y guía durante el desarrollo del proyecto.

A todos los docentes quienes forman parte del departamento de Eléctrica, Electrónica y Telecomunicaciones, gracias por todo los conocimientos que transmitían en cada clases.

También agradezco a todos mis compañeros con quienes a diario convivía en la universidad, ya que fue como nuestro segundo hogar, donde compartíamos ideas para fortalecer nuestros conocimientos.

VÍCTOR HUGO CHITUPANTA PALOMO

## Índice de contenidos

Urkund.....	2
Certificación.....	3
Responsabilidad de autoría .....	4
Autorización de publicación .....	5
Dedicatoria .....	6
Agradecimiento .....	7
Índice de contenidos.....	8
Índice de tablas .....	13
Índice de figuras .....	14
Resumen.....	17
Abstract.....	18
Capítulo I.....	19
Introducción.....	19
Antecedentes.....	19
Justificación e importancia .....	20
Alcance del proyecto.....	21
Objetivos.....	21
Objetivo general .....	21
Objetivos específicos.....	21
Capítulo II.....	22



Estado del arte .....	22
Televisión Digital y la interactividad.....	22
Estándar de la nueva generación ATSC 3.0 .....	24
DASH .....	27
Multimedia Presentation Description (MPD).....	29
Formato del segmento DASH .....	31
Señalización para ROUTE/DASH y MMTP/MPU.....	32
Escaneo rápido de servicios en ATSC 3.0.....	34
Identificación de servicios ATSC 3.0 en sesiones ROUTER y MMTP.....	35
Entrega de contenido en ATSC 3.0.....	36
Entrega de contenido de ATSC 3.0 en tiempo real.....	37
Entrega de contenido de ATSC 3.0 en tiempo no real (NRT) .....	39
Sistema de entrega híbrido de contenido en ATSC 3.0 .....	40
Entorno de ejecución para una aplicación en ATSC 3.0.....	42
Diferencias entre un entorno de web normal y un entorno ATSC 3.0 en HTML5	45
Modelo de receptor ATSC 3.0.....	47
Origen de un recurso web .....	48
Aplicación broadcaster.....	48
WebSocket Interface .....	50
Enlace de interfaz.....	52
Enlace de datos.....	54

	10
Métodos soportados en ATSC 3.0.....	56
Árbol de carpetas de la aplicación en ATSC 3.0.....	59
DASH INGEST .....	60
CAPÍTULO III .....	62
Desarrollo de la aplicación interactiva en base al estándar ATSC 3.0 .....	62
Software del sistema.....	62
Estructura de un documento HTML.....	62
Estructura de una aplicación web mediante bootstrap .....	63
Estructura de la aplicación interactiva .....	64
Barra de navegación .....	64
Sección de visualización y descripciones .....	66
Sección de visualización .....	67
Sección de descripción .....	69
Importación de archivos externos .....	70
Importación de bootstrap.....	71
Importación de chart.js .....	72
Importación de dash.js .....	72
Importación de clnt.js.....	72
Creación y Reproducción de contenido MPEG-DASH .....	73
FFMPEG .....	73
Bento4.....	74

	11
dash.js.....	75
Resultado de la página web .....	76
Etapa de interactividad .....	76
Document Object Model (DOM) .....	76
Métodos DOM utilizados.....	77
Eventos utilizados .....	78
Cambio de contenido HTML.....	79
Métodos de comunicación.....	79
Métodos de solicitudes HTTP .....	79
Método GET.....	80
Método POST .....	80
Interfaz de programación de aplicación (API) .....	80
Estructura JSON.....	80
APIs utilizadas en el proyecto .....	81
API Fetch .....	81
API de geolocalización .....	81
API de clima “open-weather-map” .....	81
Websocket .....	83
JSON-RPC.....	84
Algoritmos en JavaScript relacionados con el cliente.....	85
Servidores.....	92

	12
Módulos .....	92
Algoritmos en JavaScript relacionados con el servidor .....	94
Capítulo IV .....	98
Análisis de Resultados .....	98
Creación del contenido MPEG-DASH .....	98
Funcionamiento de la aplicación .....	99
Interactividad.....	100
Teclado del control remoto.....	100
Cambio del texto de los subtítulos .....	103
Adaptación de los segmentos según en ancho de banda de la red.....	103
Primer escenario .....	104
Segundo escenario .....	105
Tercer escenario .....	105
Análisis de tráfico con la herramienta wireshark, para ver el comportamiento de la información a transmitir.....	106
Capítulo V .....	110
Conclusiones y recomendaciones .....	110
Conclusiones .....	110
Recomendaciones .....	113
Referencias .....	114
Anexos .....	118

## Índice de tablas

<b>Tabla 1</b> <i>API para ejecución de aplicación</i> .....	46
<b>Tabla 2</b> <i>Función interfaz</i> .....	54
<b>Tabla 3</b> <i>Características de audio/video de cada versión</i> .....	73
<b>Tabla 4</b> <i>Códigos utilizados al activarse el evento keydown</i> .....	78

## Índice de figuras

<b>Figura 1</b> <i>Pila de protocolo de ATSC 3.0</i> .....	24
<b>Figura 2</b> <i>Estándares utilizados en la capa de entrega en ATSC 3.0</i> .....	27
<b>Figura 3</b> <i>Comunicación cliente-servidor con DASH</i> .....	28
<b>Figura 4</b> <i>Modelo de datos jerárquico MPD</i> .....	30
<b>Figura 5</b> <i>Formato del segmento DASH</i> .....	32
<b>Figura 6</b> <i>Referencias SLT a servicios vía SLS</i> .....	33
<b>Figura 7</b> <i>Escaneo rápido de servicios en ATSC 3.0</i> .....	34
<b>Figura 8</b> <i>Paquete enviado sobre dos sesiones MMT</i> .....	39
<b>Figura 9</b> <i>Funcionamiento híbrido dentro del cliente DASH</i> .....	40
<b>Figura 10</b> <i>Modo de entrega híbrido con MMTP/MPU</i> .....	41
<b>Figura 11</b> <i>Componentes lógicos del receptor en ATSC 3.0</i> .....	47
<b>Figura 12</b> <i>Relación del servidor web del receptor y agente de usuario</i> .....	49
<b>Figura 13</b> <i>Comunicación con receptor ATSC 3.0</i> .....	51
<b>Figura 14</b> <i>Árbol de carpetas</i> .....	60
<b>Figura 15</b> <i>Comunicación y peticiones con POST, GET</i> .....	61
<b>Figura 16</b> <i>Estructura básica de un documento HTML5</i> .....	63
<b>Figura 17</b> <i>Barra de navegación</i> .....	65
<b>Figura 18</b> <i>Código HTML para sección de visualización de videos y anuncio</i> .....	67
<b>Figura 19</b> <i>Código HTML para sección descriptiva e información</i> .....	69
<b>Figura 20</b> <i>Importación de archivo CSS para Bootstrap</i> .....	71
<b>Figura 21</b> <i>Importación de archivo JavaScript para Bootstrap</i> .....	71
<b>Figura 22</b> <i>Importación de archivo JavaScript chart.js</i> .....	72
<b>Figura 23</b> <i>Importación de archivo JavaScript dash.js</i> .....	72
<b>Figura 24</b> <i>Importación de archivo JavaScript clnt.js</i> .....	73

<b>Figura 25</b> Código en JavaScript para crear un reproductor MPEG-DASH .....	75
<b>Figura 26</b> Página web cargada en el navegador.....	76
<b>Figura 27</b> Modelo DOM estructura de árbol.....	77
<b>Figura 28</b> Ejemplo estructura JSON .....	80
<b>Figura 29</b> Ejemplo de consulta mediante API fetch.....	81
<b>Figura 30</b> Interfaz de pruebas open-weather-map .....	82
<b>Figura 31</b> Respuesta a solicitud API de clima.....	83
<b>Figura 32</b> Respuesta ante solicitud empleando protocolo JSON-RPC.....	84
<b>Figura 33</b> Diagrama de flujo de la primera parte de la función principal.....	85
<b>Figura 34</b> Diagrama de flujo función enviar_urls .....	86
<b>Figura 35</b> Diagrama de flujo función ad .....	87
<b>Figura 36</b> Diagrama de flujo primera parte función anuncio_ocenclck.....	88
<b>Figura 37</b> Diagrama de flujo segunda parte función anuncio_ocenclck.....	89
<b>Figura 38</b> Diagrama de flujo tercera parte función anuncio_ocenclck .....	90
<b>Figura 39</b> Diagrama de flujo función consultar_base_oceanos .....	91
<b>Figura 40</b> Diagrama de flujo función Ubicacion_geo.....	92
<b>Figura 41</b> Datos solicitados al ejecutar npm init.....	93
<b>Figura 42</b> Paquetes instalados mostrados en el archivo package.json .....	94
<b>Figura 43</b> Diagrama de flujo acerca del funcionamiento del servidor http parte1 .....	94
<b>Figura 44</b> Diagrama de flujo acerca del funcionamiento del servidor http parte2 .....	95
<b>Figura 45</b> Diagrama de flujo sobre el servidor websocket.....	96
<b>Figura 46</b> Diagrama de flujo de función wsmetodos encargada de enviar mensajes utilizando JSON-RPC.....	97
<b>Figura 47</b> Creación de las tres versiones del archivo con sus respectivas fragmentaciones.....	98

<b>Figura 48</b> Carpeta output y subcarpetas de los segmentos de audio/video y el archivo <i>stream.mpd</i> .....	98
<b>Figura 49</b> Resultado de la página web cargada en el navegador.....	99
<b>Figura 50</b> Respuesta de la conexión del servidor websocket y la preferencia de subtítulo.....	100
<b>Figura 51</b> Respuestas al presionar la tecla A, consultar idioma y consultar MPDUrl.	101
<b>Figura 52</b> Respuesta de la temperatura y Clima al presionar la tecla S.....	101
<b>Figura 53</b> Información al presionar la tecla D.....	102
<b>Figura 54</b> Gráfico de barra según los votos al presionar las teclas Q, W, E.....	102
<b>Figura 55</b> Notificación del cambio del texto de los subtítulos .....	103
<b>Figura 56</b> Ancho de banda con su respectiva escala de video extraído del archivo <i>stream.mpd</i> .....	103
<b>Figura 57</b> Perfiles creados con diferentes velocidades de descarga.....	104
<b>Figura 58</b> URL de los segmentos descargados con ancho de banda de 1.5Mbps/s....	104
<b>Figura 59</b> URL de los segmentos descargados con ancho de banda de 800 kbits/s...	105
<b>Figura 60</b> URL de los segmentos descargados un ancho de banda de 400 kbits/s.....	106
<b>Figura 61</b> Parámetros de los campos en las capas.....	106
<b>Figura 62</b> Contenido del archivo MPD .....	107
<b>Figura 63</b> Cabecera de la capa de aplicación .....	108



## Resumen

En el presente proyecto de investigación se define la estructura para la interactividad con base al nuevo estándar ATSC 3.0. De acuerdo a la norma A/331 son definidos los protocolos ROUTE/DASH, MMTP/MPU para la entrega de contenido y servicios en radiodifusión para banda ancha a través del protocolo HTTP.

Para evaluar la estructura se creó una aplicación interactiva compuesta de archivos multimedia, HTML5, JavaScript y CSS. De acuerdo a la norma A/344 se puede acceder a los recursos mediante solicitudes HTTP, por lo que se creó un servidor que emplea este protocolo para enviar contenido por banda ancha para intercambiar mensajes en formato JSON. Para entregar preferencias de idioma del usuario se creó una interfaz WebSocket. Debido a que ATSC 3.0 utiliza el formato de entrega MPEG DASH en la transmisión, se crearon 3 versiones de video con diferentes resoluciones y tasas de bit mediante el programa FFMPEG, para lo cual se utilizó la herramienta bento4 para fragmentar las 3 versiones de video creadas y empaquetar en una presentación MPEG DASH. Con la ayuda de las herramientas de google chrome se observa cómo se adapta el contenido al ancho de banda disponible, además se realizó la captura de paquetes para analizar la forma en la que se encapsula el descriptor de presentación de medios (MPD) en la carga útil de la respuesta HTTP. En las capas posteriores fueron analizados los campos más relevantes en los protocolos TCP e IP.

### **PALABRAS CLAVE:**

- **ATSC 3.0**
- **INTERACTIVIDAD**
- **HTML5**
- **JAVASCRIPT**

### **Abstract**

This research project defines the structure for interactivity based on the new ATSC 3.0 standard. According to the A/331 standard, the ROUTE/DASH, MMTP/MPU protocols are defined for the delivery of content and services in broadband broadcasting through the HTTP protocol.

To evaluate the structure, an interactive application composed of multimedia files, HTML5, JavaScript and CSS was created. According to the A/344 standard, resources can be accessed through HTTP requests, so a server was created using this protocol to send content over broadband to exchange messages in JSON format. A WebSocket interface was created to deliver user language preferences. Since ATSC 3.0 uses the MPEG DASH delivery format in transmission, 3 video versions with different resolutions and bit rates were created using the FFmpeg program, for which the bento4 tool was used to fragment the 3 video versions created and packaged into an MPEG DASH presentation. With the help of google chrome tools, it is observed how the content is adapted to the available bandwidth, also packet capture was performed to analyze the way in which the media presentation descriptor (MPD) is encapsulated in the HTTP response payload. In the subsequent layers, the most relevant fields in the TCP and IP protocols were analyzed.

### **KEYWORDS:**

- **ATSC 3.0**
- **INTERACTIVITY**
- **HTML5**
- **JAVASCRIPT**

## Capítulo I

### Introducción

#### Antecedentes

Hoy en día la tendencia de utilizar estándares nuevos para la televisión digital terrestre (TDT), es importante por lo cual se debe dar continuidad al estudio de los nuevos sistemas de DTV, para mejorar los resultados en cuanto al rendimiento, funcionalidad y eficiencia (Medina, et al., 2016). Con la llegada de la última versión de los estándares del Comité de Sistemas de Televisión Avanzada ATSC 3.0, se da flexibilidad para la adaptación a nuevas tecnologías y servicios de manera sencilla, implicando un mayor aprovechamiento y optimización del espectro radioeléctrico (Chernock & Whitaker, 2017).

ATSC 3.0 da la oportunidad para que los organismos de radiodifusión terrestre envíen servicios de contenido híbrido a receptores fijos/móviles sin interrupciones, combinando tanto la transmisión por broadcast como por broadband, así como también las opciones de múltiples pantallas en resoluciones, HD y Ultra HD (Ortiz, 2017).

Existen varias propuestas para los sistemas de Televisión Híbrida, en diferentes partes del mundo que están siendo desarrolladas entre ellas el estándar ATSC 3.0 el cual tiene conceptos similares al estándar HbbTV con relación a la interactividad, este estándar en su última versión HbbTV 2.0.1 tiene dos grandes cambios, la primera es la incorporación de la tecnología HTML5 para el desarrollo de aplicaciones asociadas a la Televisión Híbrida, disminuyendo el proceso del desarrollo de aplicaciones interactivas y ampliando el espectro RF. El segundo es la incorporación del estándar HEVC, para obtener una mejor eficiencia en la señal de video es decir resoluciones de 4K e inclusive superiores (Rondán, et al., 2016).

## **Justificación e importancia**

La Unión Internacional de Telecomunicaciones (ITU), adoptó ATSC 3.0 como un estándar de transmisión digital del futuro conocido también como NEXT GEN TV. Hoy en día los usuarios requieren tener la capacidad de ver cualquier contenido en cualquier dispositivo y en cualquier lugar, dicho contenido puede ser entregado por diferentes medios como aire, cable o satélite, a través de Internet o almacenado localmente (Starzynski & Thomsen, 2017). Por lo que la industria de la transmisión debe evolucionar con las demandas de los consumidores y consecuentemente proporcionar una extensibilidad que permita la adaptación futura a estos requerimientos.

El estado actual en el que se encuentra ATSC 3.0, hay mucho trabajo de investigación por realizar por lo que el presidente de ATSC 3.0, invita a todas las organizaciones interesadas en unirse y participar, generando grandes oportunidades para continuar trabajando en investigaciones referente a TV Digital (Fernández, 2019).

La ITU también ha admitido algunas recomendaciones asociadas con servicios de Televisión Digital, llamada *Integrated Broadcast and Broadband* (IBB). Estas recomendaciones han sido el producto de la colaboración entre ITU y organizaciones como ATSC, DVB, ARIB, SMPTE, OpenCable y otros, que han ayudado al proceso de uniformidad respecto a la interactividad en la DTV (UIT-R, 2017). Ya que los diferentes estándares actuales para Televisión Híbrida cumplen con las recomendaciones señaladas por la ITU, pero la compatibilidad entre ellos aún no es posible (León & González, 2008).

A demás la Comisión Federal de Comunicaciones (FCC), para los servicios de próxima generación enfatiza en crear un proceso abierto, colaborativo y transparente para la participación de las personas interesadas, además de crear un marco regulatorio que elimine los impedimentos para inversiones e innovaciones.

## **Alcance del proyecto**

El presente proyecto de titulación tiene como propósito el analizar y evaluar la estructura, para la interactividad con la tecnología HTML5 en base al nuevo estándar ATSC 3.0

En la primera etapa del proyecto se realizará el estudio del arte sobre los modos de transmisión, el manejo de la interactividad de ATSC 3.0. Para posteriormente realizar el respectivo análisis del proceso de la transmisión en base al estándar ATSC 3.0 a partir de la validación del protocolo de comunicación y la estructura de transmisión.

La segunda etapa consiste en el diseño y evaluación de una aplicación interactiva como modelo en base al nuevo estándar ATSC 3.0.

## **Objetivos**

### ***Objetivo general***

Analizar y Evaluar la estructura, para la interactividad en HTML5 con el estándar ATSC 3.0.

### ***Objetivos específicos***

- Recopilar información acerca de los modos de transmisión que existen en la actualidad y las plataformas de interactividad.
- Validar el protocolo de comunicación de datos y la estructura de transmisión broadcast.
- Analizar el proceso para la transmisión en base al estándar ATSC 3.0
- Diseñar y Evaluar aplicaciones interactivas para el estándar ATSC 3.0

## Capítulo II

### Estado del arte

#### Televisión Digital y la interactividad

Con el cambio de la televisión analógica a la televisión digital, surgieron nuevos estándares y normas para la transmisión, entre los más prácticos están ATSC, ISDB, DVB y DTMB. Existen varias versiones de algunas de estas normas, tal es el caso de la norma ISDB-Tb desarrollado en Brasil en base a la norma ISDB-T (Morales, 2010).

En la actualidad las normas para la DTV, indistintamente del medio de transmisión, emplean el formato MPEG-2 para el flujo de transporte o *transport stream* (TS), el cual contiene información multiplexada de todos los elementos a ser transmitidos como tablas de control, flujos de audio/video o contenidos multimedia (Jung, 2016).

Respecto a los contenidos interactivos son multiplexados dentro de estos TS, pueden estar como contenido independiente o añadido a un programa televisivo, para que el receptor pueda interpretar los contenidos transmitidos, cada norma de la DTV ha establecido su propio estándar para el manejo de la interactividad, por ejemplo la norma DVB estableció el estándar *Multimedia Home Platform* (MHP), basado en aplicaciones interactivas desarrolladas en Java y HTML. La norma japonesa ISDB-T estableció el estándar *Association of Radio Industries and Business* (ARIB), mientras que la norma implementada en Brasil ISDB-Tb decidió desarrollar el estándar Ginga. La norma ATSC estableció el estándar *Digital Television Application Software Environment* (DASE), y para su versión ATSC 2.0 estableció el estándar *Interactive Services Standard* (ISS) (Pinto & Barredo, 2017).

Para la transmisión de datos, indistintamente del estándar usado, los métodos más relevantes han sido el carrusel de datos y de objetos, estos carruseles se integran al TS aplicando la especificación *Digital Storage Media Command and Control* (DSM-CC), donde la información se ordena en módulos para luego ser divididos en bloques de igual tamaño para ser transmitidos (Graves & Guidobono, 2009).

Cabe recalcar que los estándares mencionados no son los únicos estándares para contenidos interactivos, pero son los más prácticos en cada una de las normas. Estos estándares han establecido la base para la creación de nuevos estándares, tal es el caso del estándar *Hybrid broadcast broadband TV* (HbbTV), la cual combina contenidos recibidos por la red broadcast como por broadband a través de la plataforma (browser) en la cual se ejecutan las aplicaciones (Gallegos, 2008). La última versión de HbbTV 2.0.1, incluye dos grandes cambios respecto a sus versiones anteriores (Rondán, et al., 2016). La primera es la incorporación de la tecnología HTML5 para el desarrollo de aplicaciones asociadas a la Televisión Híbrida, reduciendo el proceso para el desarrollo de aplicaciones y aumentando el espectro RF. El segundo es la incorporación del estándar *High Efficiency Video Coding* (HEVC), obteniendo una mejor eficiencia en la señal de video es decir resoluciones de 4K e inclusive superiores (Rondán, et al., 2016).

Sin embargo, MPEG-2 TS puede ser inadecuado para satisfacer las necesidades emergentes de un acceso más personalizado, como solución a las demandas el streaming adaptativo HTTP, se está utilizando actualmente para entregar todo el contenido IP de streaming de banda ancha. Mientras tanto, reconociendo los inconvenientes de los estándares y sistemas existentes MPEG ha desarrollado el estándar *Dynamic Adaptive Streaming* sobre HTTP (DASH), como una innovación de los estándares actuales.

## Estándar de la nueva generación ATSC 3.0

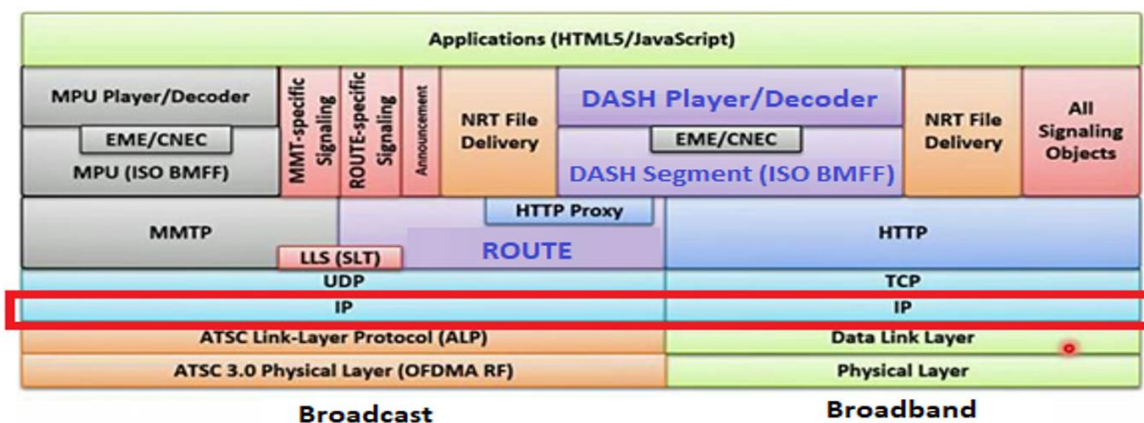
El Comité de Sistemas de Televisión Avanzada (ATSC), es una organización internacional sin fines de lucro que desarrolla estándares prácticos y voluntarios, en la actualidad está elaborando una especificación de próxima generación llamada ATSC 3.0, en Norteamérica y varios países de Asia (Zhang, 2016). El nuevo estándar 3.0 contiene una cantidad significativa de información y capacidades que no se encuentran en los estándares 1.0/2.0. Uno de los aspectos clave de 3.0 es que supera las barreras de tecnología patentada de la industria que se encuentran en 1.0, como MPEG-2 TS, ASI, SMPTE entre otros (González, 2002).

ATSC 3.0 es el primer estándar de radiodifusión basado en el protocolo de internet (IP), para el transporte y encapsulamiento de la transmisión en vez de utilizar el tren de transporte MPEG (ATSC, 2019). Respecto a la interactividad y creación de aplicaciones híbridas, está acogiendo conceptos similares a HbbTV por la incorporación de tecnologías como HTML5, JavaScript, HEVC, IP entre otros (Azarcoya, 2019).

ATSC 3.0 al trabajar con sistemas híbridos hace uso de dos redes: broadcast y broadband, como se muestra en la Figura 1.

**Figura 1**

*Pila de protocolo de ATSC 3.0*





*Nota.* El gráfico muestra la pila de protocolo de ATSC 3.0. Tomado de “*Signaling, Delivery, Synchronization, and Error Protection*”, (ATSC, 2019).

En la Figura 1 se muestra las múltiples capas que posee la pila de ATSC 3.0, a continuación se explica cómo funciona cada una de ellas para los nuevos servicios (Venkatraman, Vellingiri, & Prabhakaran, 2014).

**Capa de enlace:** Los protocolos de la capa de enlace optimizan el empaquetado para la capa física, es decir el protocolo de capa de enlace ATSC (ALP) optimiza la entrega de alta eficiencia a través de RF. El protocolo de transporte ALP (ALPTP) mantiene relaciones de ciertos componentes cuando el paquete ALP está separado del planificador-encuadre, como las estructuras ALP a PLP y los metadatos asociados.

**Capa de red:** La capa de red utiliza la misma IP para entregar paquetes apropiados entre la capa física y la capa de sesión. Es decir desde las capas físicas ATSC o de banda ancha a las capas de transporte UDP o TCP. La pila de banda ancha es una topología de red típica para las capas físicas asociadas.

**Capa de transporte:** La capa de transporte ATSC 3.0, para broadcast usa UDP en una conexión unidifusión o multidifusión, mientras para broadband usa TCP en una conexión bidireccional.

**Capa de sesión:** En esta capa se encuentra MMTP, ROUTE y HTTP, los cuales describen la relación de origen y destino para el transporte IP:

- El protocolo de transporte de medios MPEG (MMTP) es utilizado para la entrega de transmisión lineal, admitiendo características nuevas y dinámicas, como la inserción dinámica de anuncios o cambios de contenido.

- El protocolo ROUTE, brinda una mayor flexibilidad para las funciones de grabadora de video digital (DVR), video a pedido (VOD) y funciones de entrega en tiempo no real (NRT).
- El protocolo de transferencia de hipertexto (HTTP) es utilizado para la comunicación entre un cliente y un servidor a través de una red IP.

**Capa de administración:** Son las capas de presentación y aplicación donde se realiza la decodificación para los reproductores multimedia y el consumo de transmisiones y otros datos en tiempo no real (NRT).

Debido a la forma en que está estructurada ATSC 3.0 el streaming adaptativo HTTP se está utilizando para entregar todo el contenido IP de streaming de banda ancha. A pesar que la transmisión HTTP a través de TCP es adecuada para la entrega de banda ancha, no representa una adecuada entrega de extremo a extremo en la transmisión broadcast (Azarcoya, 2019).

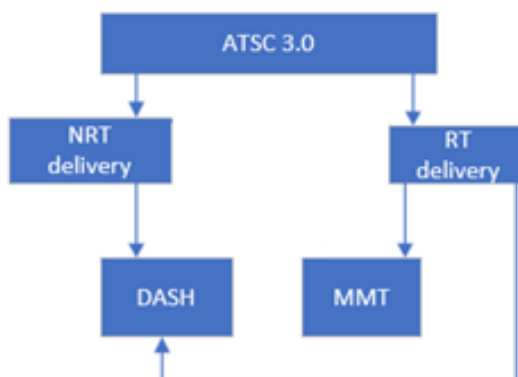
Para soportar los servicios híbridos de transmisión y banda ancha, ATSC 3.0 ha propuesto un método de transporte de transmisión mejorado denominado entrega de objetos en tiempo real sobre transporte unidireccional ROUTE/DASH, esto con el propósito de entregar contenido tanto en tiempo real y no en tiempo real (NRT) a través de canales de transmisión. Adicional, el estándar *MPEG Media Transport (MMT)* ha sido propuesto recientemente como parte del estándar ISO/IEC 23008, para el transporte y entrega de datos multimedia mediante redes IP, superando las deficiencias de los protocolos de transporte actuales (Venkatraman, Vellingiri, & Prabhakaran, 2014).

MMT asume redes IP con cachés inteligentes cerca de los receptores (Xu, Xie, Hao, & Le, 2016). Previamente las cachés inteligentes buscan el contenido mediante identificadores únicos empaquetando y enviando de forma adaptativa el contenido.

En resumen, ATSC 3.0 en la capa de entrega de datos utiliza dos estándares ROUTE/DASH y MMT para el canal broadcast y HTTP para el canal broadband. Los servicios pueden distribuirse como flujos escalables que constan de una capa base y varias capas de mejora. (Nakachi, 2017).

## Figura 2

*Estándares utilizados en la capa de entrega en ATSC 3.0*



*Nota.* El gráfico muestra los estándares utilizado en la capa de entrega en ATSC 3.0. Tomado de “*MPEG Media Transport technologies and its application to immersive media communication services*”, (Nakachi T. , 2007).

## **DASH**

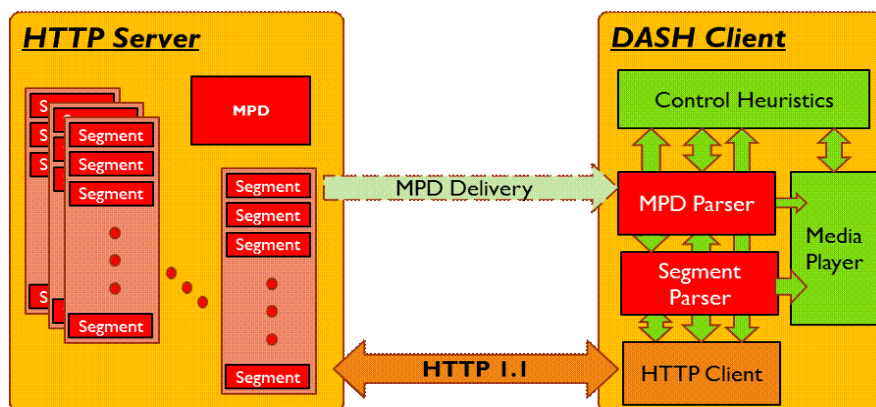
La Figura 3 muestra un escenario de transmisión simple entre un servidor HTTP y un cliente DASH, el contenido multimedia se captura y almacena en un servidor y se entrega mediante HTTP (Roberto, Gomez, & Whitaker, 2016). El contenido existe en el servidor en dos partes:

1. En un fichero de índice *Media Presentation Description* (MPD), el cual describe un manifiesto del contenido disponible, sus diversas alternativas, sus direcciones URL y otras características.

- En los segmentos se encuentran los flujos de bits multimedia reales en forma de fragmentos, en uno o varios archivos.

**Figura 3**

*Comunicación cliente-servidor con DASH*



*Nota.* Los formatos y funcionalidades de los bloques rojos son definidos por la especificación del estándar MPEG DASH (ISO 23009). Los clientes controlan la heurística y los reproductores multimedia. Tomado de “*Sistema para la integridad del vídeo en sistemas de streaming dash*”, (Euceda, 2020).

Para reproducir el contenido, comienza cuando el cliente DASH vía HTTP solicita con un GET el archivo MPD, el servidor atiende el requerimiento y lo envía. El cliente DASH aprende sobre la sincronización del programa, la disponibilidad del contenido de los medios, anchos de banda mínimos y máximos así como la existencia de varias alternativas codificadas de componentes multimedia, características de accesibilidad, ubicación de cada componente en la red y entre otras características del contenido.

Con esta información el cliente DASH selecciona la alternativa codificada adecuada y comienza a transmitir el contenido mediante la obtención de los segmentos. Después del almacenamiento en un buffer apropiado para permitir variaciones de rendimiento de la red, el cliente continúa obteniendo los segmentos subsiguientes y

también monitorea las fluctuaciones del ancho de banda de la red. Dependiendo de sus medidas, el cliente decide cómo adaptarse al ancho de banda disponible obteniendo segmentos de diferentes alternativas para mantener un buffer adecuado.

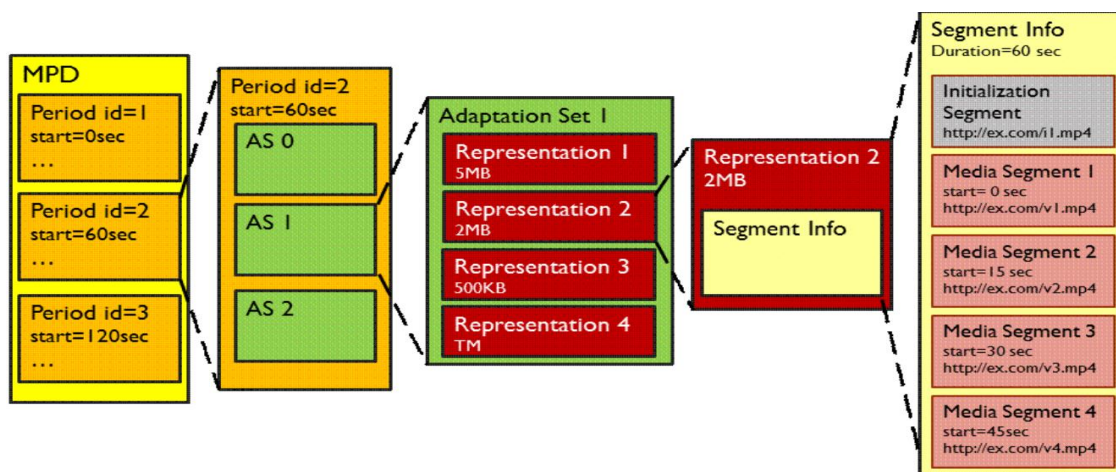
### **Multimedia Presentation Description (MPD)**

La transmisión de aplicaciones interactivas y dinámicas en HTTP necesita un servidor con diferentes alternativas de velocidad (bits) con el fin de reproducir el contenido. En MPEG-DASH, estas características las describe MPD, que es un archivo XML que detalla las particularidades y metadatos del contenido multimedia como vídeo, audio, subtítulos, entre otros; el cual se forma de subelementos jerárquicos, esto a su vez representa un aspecto para la gestión de segmentos vinculados entre sí como se muestra en la Figura 4, estos elementos son:

- **Period:** Son etapas temporales en las que se divide el contenido multimedia, por ejemplo, capítulos, escenas, insertar anuncios, entre otros.
- **Adaptation Set:** Son componentes del contenido ubicado al interior de cada período, por ejemplo, codificadores, subtítulos, idiomas, entre otros.
- **Representation:** Representa la calidad entregada del contenido del adaptador set, orientado al bit rate y la resolución.
- **Segment:** Son los fragmentos en los que se divide el contenido de representación y tiene un tiempo estimado.

Figura 4

Modelo de datos jerárquico MPD



*Nota.* El gráfico muestra el modelo de datos del fichero de índice *MPD*. Tomado de “Sistema para la integridad del vídeo en sistemas de streaming dash”, (Euceda, 2020).

Como se muestra en la Figura 4 el MPD consta de uno o varios períodos, donde un período es un intervalo del programa a lo largo del eje temporal, donde cada uno tiene una hora de inicio, una duración y consta de uno o varios set de adaptación. Un set de adaptación proporciona información sobre los componentes de los medios y sus diversas alternativas codificadas. Por ejemplo, un set de adaptación puede contener las diferentes velocidades en bits/s del componente de video, otro set de adaptación puede contener las diferentes velocidades de bits/s del componente de audio del mismo contenido multimedia, cada set de adaptación incluye múltiples representaciones. Una representación es una alternativa codificada del mismo componente de medios, que varía de otras representaciones por velocidad, resolución, número de canales u otras características, estas representaciones son elegidas a partir del algoritmo heurístico de selección del cliente DASH para cada segmento a transmitir, cada representación consta de uno o varios segmentos. Los segmentos son los fragmentos del flujo de medios, estos

son los elementos que se descargarán en el cliente DASH en los diferentes formatos existentes, inicia con un segmento de inicialización y posteriormente los segmentos multimedia en contenedores ISO BMFF o MPEG-2 TS de pequeñas duraciones, además cada segmento están definidos a través de URL numeradas de manera lógica para su ubicación en el servidor.

Para utilizar este modelo de datos, el cliente DASH primero analiza el documento XML MPD. Luego selecciona el conjunto de representaciones a utilizar considerando los elementos descriptivos en el MPD, las capacidades y las elecciones de su usuario. Luego, crea una línea de tiempo y comienza a reproducir el contenido multimedia solicitando segmentos de medios apropiados.

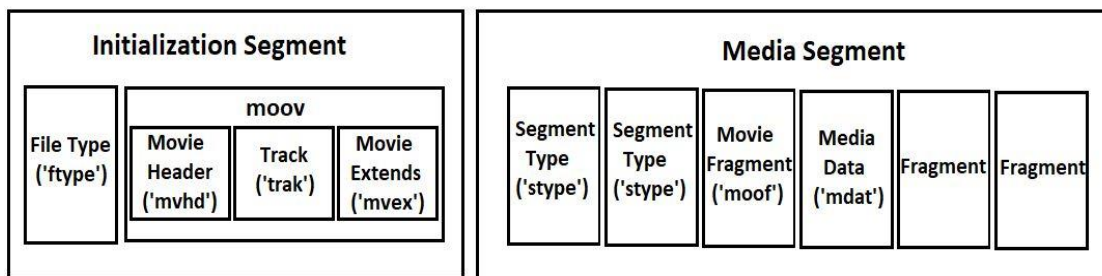
La descripción de cada representación incluye información sobre sus segmentos, formulando solicitudes para cada segmento en términos de URL- HTTP y rango de bytes. Para presentaciones en vivo, el MPD proporciona la hora de inicio de la disponibilidad del segmento y la hora de finalización, la hora de inicio de los medios y la duración fija o variable de los segmentos.

### **Formato del segmento DASH**

En la Figura 5 se muestra el formato de un segmento DASH, está se divide en dos partes. En primer lugar, se encuentra la inicialización del segmento, donde posee información del tipo, cabecera y la extensión del archivo. La segunda parte representa el contenido del segmento.

**Figura 5**

Formato del segmento DASH



*Nota.* El gráfico muestra el formato del segmento DASH. Tomado de “Sistema para la integridad del vídeo en sistemas de streaming dash”, (Euceda, 2020)

Para acceder al contenido multimedia se lo hace a través de una colección de segmentos. Un segmento representa la entidad de la respuesta al HTTP GET del cliente DASH. Un componente multimedia está codificado y dividido en varios segmentos. El primer segmento contiene la información necesaria para la inicialización del decodificador de medios del cliente DASH.

El flujo de medios está compuesta por varios segmentos consecutivos. A cada segmento de medios se le asigna una URL única, un índice, una hora de inicio y una duración. Cada segmento de medios contiene al menos un *Stream Access Point* (SAP), que es un punto de acceso aleatorio en el flujo de medios donde la decodificación puede comenzar utilizando solo datos a partir de ese punto en adelante. Para descargar segmentos en múltiples partes, la especificación define un método de señalización de subsegmentos mediante un cuadro de índice de segmento.

### **Señalización para ROUTE/DASH y MMTP/MPU**

La señalización de los servicios proporciona información de descripción y descubrimiento de los servicios. Consta de dos elementos funcionales: Listas de servicios

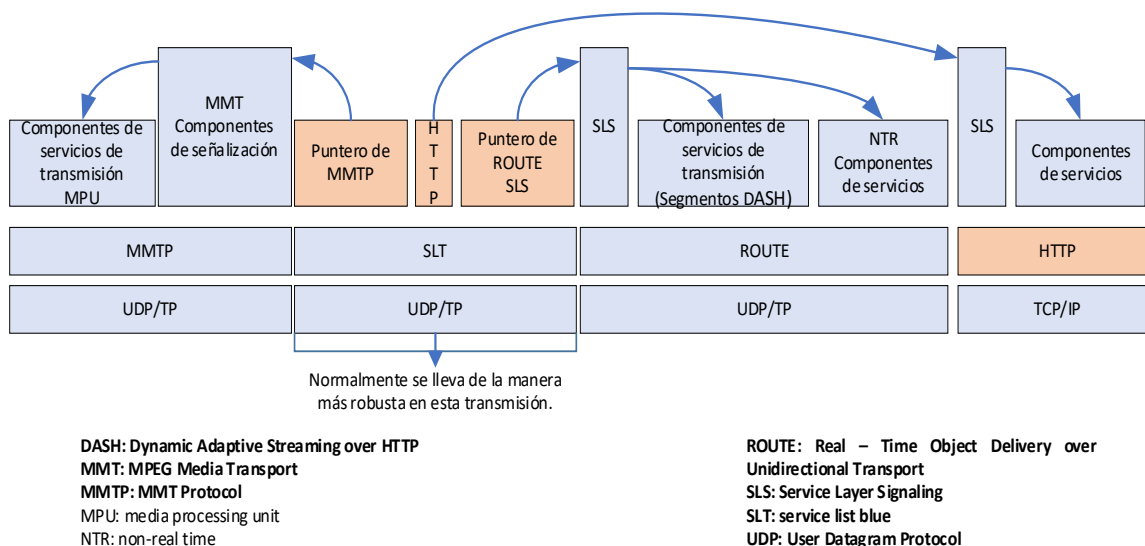


(SLT) y la señalización de capa de servicio (SLS). Estos proporcionan la información necesaria para descubrir y adquirir servicios ATSC 3.0. La información de señalización de ROUTE/DASH y MMTP/MPU se transporta en la carga útil de datos de los paquetes IP con una dirección/puerto bien definido, esto se conoce como *Low Level Signaling* (LLS), esta implica que la información de señalización admite escaneos rápidos de canales que permite al receptor encontrar primero la emisión que está transmitiendo para crear una lista de todos los servicios y arrancar el descubrimiento del SLS para cada servicio ATSC 3.0.

A continuación en la Figura 6 se muestra la relación entre los servicios SLT, SLS y ATSC 3.0. En donde la entrega de broadcasting de los servicios ROUTE/DASH (streaming y NRT), el SLS es transportado por ROUTE/UDP/IP en una de las sesiones de *Layered Coding Transport* (LCT). Mientras que para los servicios de streaming MMT/MPU, el SLS es transportado por mensajes de señalización MMTP. En la entrega de banda ancha, el SLS se transmite a través de HTTP (S)/TCP/IP.

**Figura 6**

*Referencias SLT a servicios vía SLS*





3. Al detectar una señal a una frecuencia determinada, el procesador de banda base extraerá la señalización L1 del preámbulo.
4. El procesador de banda base enviará PLP que contiene señalización de la capa de enlace y LLS al middleware que los extraerá de los datos de PLP.
5. El middleware contiene módulos funcionales como un administrador de señalización, un administrador de canales y otras partes. Después de recibir datos PLP que contienen señalización de capa de enlace y LLS del procesador de banda base, el middleware los pasará a cualquier módulo funcional que gestione cualquier tipo de datos (por ejemplo, señalización, audio / video / otros).
6. El módulo de middleware extraerá el LMT de la señalización de la capa de enlace y pasará el LMT al analizador LMT. El módulo de middleware extraerá el SLT de LLS mediante LLS\_table\_id, que tiene el valor "0x01". El módulo de middleware pasará el SLT al analizador de SLT.
7. El analizador LMT analiza el LMT y extrae la información que es esencial para producir un mapa de canales por ejemplo, PLPID, información de sesión: dirección IP y el número de puerto.
8. El analizador de SLT analiza el SLT y extrae la información que es esencial para producir un mapa de canales por ejemplo, identificación de servicio, nombre de servicio, información de señalización SLS de transmisión y oculta en el mapa.
9. La información se almacena en el mapa de canales.

### ***Identificación de servicios ATSC 3.0 en sesiones ROUTER y MMTP***

Una sesión de ROUTE está asociado con una o más sesiones LCT para transportar los componentes de contenido del servicio en ATSC 3.0, para servicios de transmisión continua la sesión LCT puede transportar componentes de servicio como: flujo de audio, video o subtítulos.

Una sesión ROUTE se identifica mediante la dirección IP origen/destino y el puerto de destino. Y la sesión LCT asociada es identificada por el identificador de sesión de transporte (TSI), el cual es único dentro de la sesión ROUTE principal, las propiedades comunes y exclusivas de las secciones LCT se da en una estructura de señalización de ROUTE denominada descripción de instancia de sesión de transporte basada en servicio (S-TSID) que es parte del SLS. Cada sección de LCT es realizada a través de una única tubería de la capa física (PLP). Las propiedades de S-TSID tienen el valor TSI y PLP\_ID para cada sesión LCT.

La sesión MMTP y el flujo de paquetes se identifican de manera similar a la sesión ROUTE. Es decir, utiliza la dirección IP, puerto de destino y el id\_paquete el cual es único dentro de la sesión MMTP principal. Las propiedades comunes a cada flujo de paquetes MMTP y las individuales se incluyen en el SLT. Las propiedades de cada sesión MMTP son dadas por mensajes de señalización MMT, que pueden ser transportados dentro de la misma. Estas propiedades incluyen el valor de packet\_id y PLP\_ID para cada flujo de paquete MMTP.

### **Entrega de contenido en ATSC 3.0**

En ATSC 3.0, el contenido multimedia incluye servicio de transmisión en tiempo real y no en tiempo real (NRT). El servicio de transmisión en tiempo real se puede entregar por ROUTE/DASH o MMTP/MPU, mientras que el contenido NRT solo puede entregarse por ROUTE/DASH porque el modo *Generic File Deliverance* (GFD) no se puede usar en el sistema ATSC 3.0.

El protocolo ROUTE proporciona capacidades generales de transmisión de difusión similares a los aplicados en la transmisión de banda ancha por HTTP. Incluyendo la admisión de técnicas de transmisión basadas en archivos que se utilizan en DASH, proporcionando una entrega de contenido compatible con los medios, lo que permite un

cambio de canal más rápido. En general, ROUTE/DASH es un protocolo de entrega de un rango de bytes con reconocimiento de medios y orientado a la transmisión basado en MPEG DASH y LCT sobre UDP.

MMTP es un protocolo diseñado para entregar archivos *ISO Base Media File Format* (ISOBMFF), útil en la transmisión en tiempo real a través de una red de entrega unidireccional, el cual tiene varias características como:

- Paquetización con reconocimiento de medios de archivos ISOBMFF.
- Multiplexación de varios componentes de medios en una sola sesión MMTP.
- Eliminación de la fluctuación de la red en el receptor bajo las restricciones establecidas por el remitente.
- Gestión del buffer del receptor por parte del servidor para evitar cualquier subdesbordamiento o desbordamiento de buffer y la fragmentación o agregación de datos de carga útil.
- Detección de paquetes perdidos durante la entrega.

A continuación, se va a analizar las formas de entrega de contenido las cuales son:

- Entrega de contenido en tiempo real (utilizando ROUTE/DASH o MMTP/MPU)
- Entrega de contenido en tiempo no real (utilizando ROUTE/DASH o MMTP/MPU)
- Entrega de contenido de manera híbrida (utilizando ROUTE/DASH y MMTP/MPU)

#### ***Entrega de contenido de ATSC 3.0 en tiempo real***

- **ROUTE/DASH en tiempo real**

Para servicio de transmisión de ROUTE/DASH, no importa si se trata de contenido en vivo o contenido pregrabado. Esto lo define el atributo MPD @tipo, para contenido en vivo se debe establecer en dinámico. En la entrega de servicios de transmisión mediante ROUTE/DASH se establecen tres tipos de entrega: entidad, archivo y empaquetado.

1. **Modo de entidad:** Se utiliza cuando no es posible determinar los parámetros de la tabla de distribución de archivos ampliada (EFDT), permite la entrega parcial o fragmentada de la misma manera que HTTP.
  2. **Modo de archivo:** Los metadatos de archivo-objeto representados por el EFDT se incrustarían o se harían referencia a ellos como un objeto de entrega independiente.
  3. **Modo de empaquetado:** Debe usarse como formato de objeto de entrega si el flujo de reparación se usa junto con el flujo de origen para la entrega de contenido de transmisión.
- **MMTP/MPU en tiempo real**

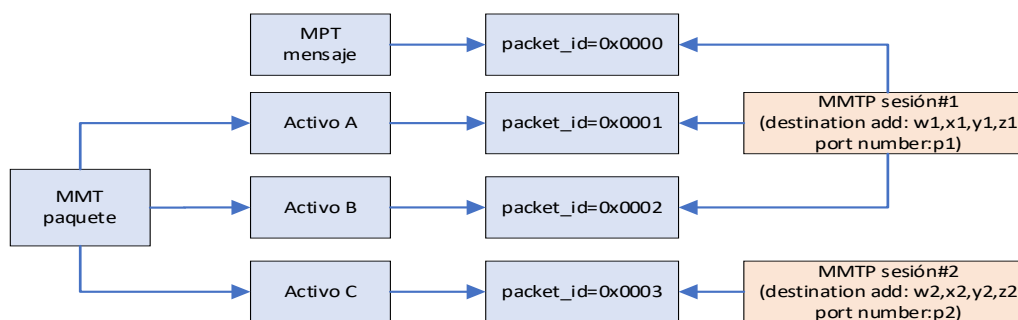
Cada componente de contenido se considera un elemento de MMT según MMTP/MPU. Cada elemento de MMT es una colección de una o más MPU con el mismo ID único. Un paquete de MMT representa una colección de uno o más elementos y un servicio ATSC 3.0 puede contener uno o más paquetes de MMT.

Tanto los paquetes MMT como las MPU no se superponen en su tiempo de presentación, pues, entregan múltiples elementos en una sola sesión de MMTP. Los elementos están relacionados con un packet\_id que es único dentro del alcance de la sesión MMTP, permitiendo un filtrado eficiente de paquetes MMTP que transportan un elemento específico. Además, los mensajes de señalización MMT entregados al receptor se utilizan para designar la información de mapeo entre paquetes MMT y sesiones MMT.

La Figura 8 muestra un mapeo entre un paquete MMT y una sesión MMTP. El paquete MMT tiene tres elementos (A, B y C) los cuales se entregan en dos sesiones de MMTP. Todos los mensajes de señalización MMT necesarios para consumir y entregar el paquete MMT se muestra en un único mensaje MPT para simplificar.

### Figura 8

*Paquete enviado sobre dos sesiones MMTP*



MMT: MPEG Media Transport    MMTP: MMT Protocol    MPT: MMT package table

*Nota.* El gráfico muestra el paquete MMT enviado sobre dos secciones MMTP, todos los elementos MMT y mensaje MPT pueden multiplexarse en una sola sección MMTP como en MPEG-2 TS. Tomado de “*Dash y MMT y sus aplicaciones en ATSC 3.0*”, (Xu, Xie, Hao, Le, & Jun, 2016).

### **Entrega de contenido de ATSC 3.0 en tiempo no real (NRT)**

Al entregar archivos en NRT se utiliza el modo de archivo ROUTE/DASH. Antes de la entrega del archivo, se deben informar al receptor los metadatos del archivo EFDT.

Respecto a la sincronización, DASH utiliza *Coordinate Universal Time* (UTC) para cumplir con los requisitos precisos del reloj de pared. Dado que UTC se establece sobre la capa física, En MMT, la sincronización de las MPU también se basa en marcas de tiempo que hacen referencia a UTC.

### **Sistema de entrega híbrido de contenido en ATSC 3.0**

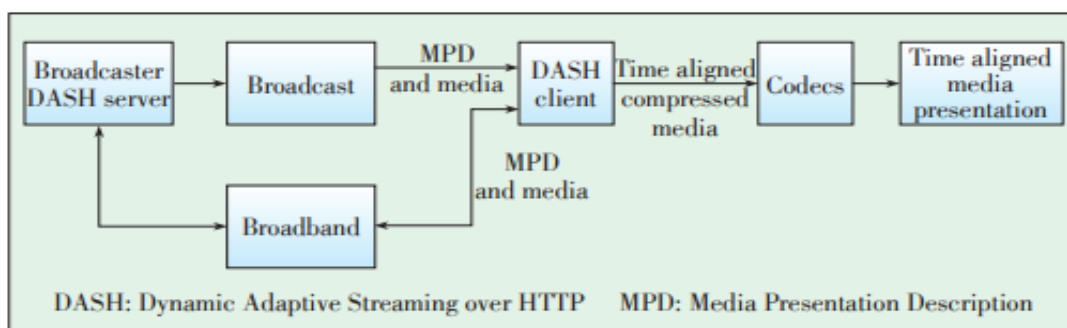
La especificación HbbTV define una plataforma híbrida broadcast/broadband para la señalización, transporte y presentación de aplicaciones mejoradas o interactivas en terminales híbridos, que incluyen tanto una conexión de transmisión compatible con DVB como una conexión de banda ancha a Internet. A continuación se presenta los modos de entrega híbridos.

- **Modo de entrega híbrido con ROUTE/DASH**

El modo de entrega de servicio híbrido con ROUTE/DASH combina la entrega de transmisión a través de ROUTE y la entrega de unidifusión a través de HTTP. En la Figura 9 se muestra el funcionamiento dentro del cliente DASH. ROUTE/DASH permite emplear técnicas especificadas en el estándar DASH para mejorar la experiencia del usuario en el caso de la entrega híbrida.

**Figura 9**

*Funcionamiento híbrido dentro del cliente DASH*



*Nota.* El gráfico muestra el funcionamiento híbrido con ROUTER/DASH. Tomado de “Dash y MMT y sus aplicaciones en ATSC 3.0”, (Xu, Xie, Hao, Le, & Jun, 2016).

En primer lugar, se tiene un servidor DASH que transmite información de manera broadcast y este está interactuando continuamente con la parte broadband de la red.



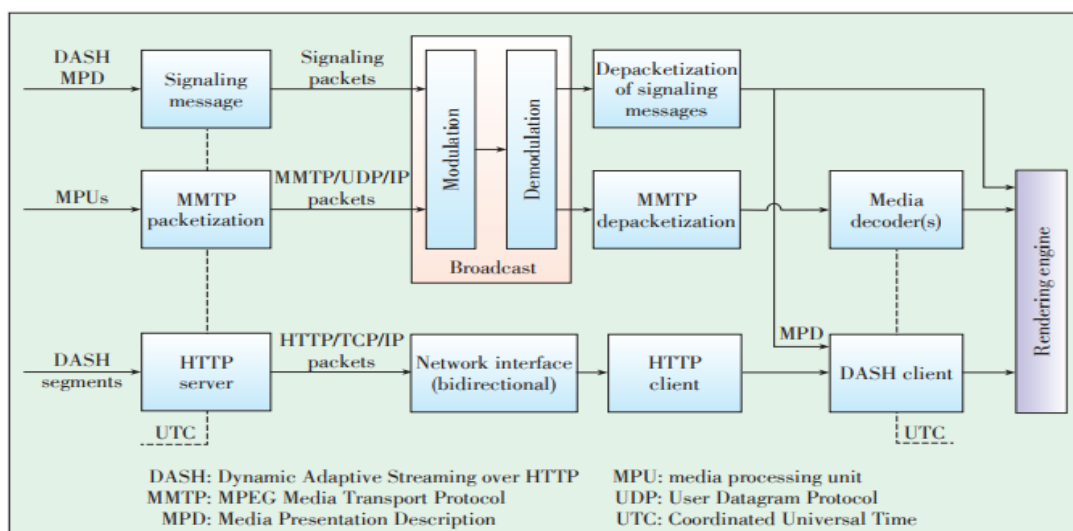
Las dos partes envían paquetes MPD al cliente DASH, para que exista una sincronización temporal se utilizan códecs y los paquetes lleguen de manera alineada a la capa de presentación.

- **Modo de entrega híbrido con MMTP/MPU**

La transmisión híbrida sobre la entrega de banda ancha se muestra en la Figura 10. Todos los componentes del sistema están bloqueados en UTC para la sincronización.

**Figura 10**

*Modo de entrega híbrido con MMTP/MPU*



*Nota.* El gráfico muestra modo de entrega híbrido con MMTP/MPU. Tomado de “Dash y MMT y sus aplicaciones en ATSC 3.0”, (Xu, Xie, Hao, Le, & Jun, 2016).

Para la red de transmisión, los datos de los medios se encapsulan en MPU, empaquetados en MMTP. Mientras que en la red de banda ancha se encapsulan en segmentos DASH. Cuando no se envían componentes por difusión, los DASH-MPD se entregan a través de la red mediante un mensaje de señalización y los segmentos se envían por banda ancha mediante una sesión HTTP de la interfaz de red de un servidor normal. Esta solución aumenta la complejidad del sistema y los costos operativos.

Al igual que el modo con ROUTE/DASH, cuando se cambia de transmisión a banda ancha, el cliente usa, antes de la transición, el último MPD contenido en un mensaje MPI para realizar una solicitud HTTP para un segmento DASH. El segmento DASH recibido se almacena en buffer y se pone a disposición del cliente DASH para su decodificación.

Para un traspaso sin interrupciones, el retraso en la entrega de radiodifusión debe ajustarse para que sea igual al retraso en la entrega de banda ancha. De lo contrario, el servicio puede congelarse al cambiar de transmisión de radiodifusión a banda ancha por primera vez. Mientras tanto, el segmento DASH y la MPU son incompatibles entre sí en formato, lo que da como resultado un mayor tiempo de procesamiento y un retraso en la conversión de datos. Además, para los servicios de transferencia, el límite de la MPU puede superponerse con el límite del segmento DASH, lo que puede crear barreras para una transferencia sin interrupciones y degradar aún más la calidad del servicio.

### **Entorno de ejecución para una aplicación en ATSC 3.0**

Los términos para el entorno de la ejecución de la aplicación se detallan de la siguiente manera:

- **Aplicación de broadcaster**

Se utiliza una aplicación de broadcaster para referenciar la funcionalidad de una colección de archivos compuestos por: documento HTML5, que sería la página de entrada y otros recursos de HTML5, CSS, JavaScript, imágenes y multimedia a los que se hace referencia, proporcionado por el emisor en un servicio ATSC 3.0.

El lado del servidor de la aplicación web se implementa mediante un receptor ATSC 3.0 y tiene una API estandarizada para todas las aplicaciones. El emisor no proporciona ningún código de aplicación del lado del servidor, pero sí documentos y

códigos basados en la web que funcionan junto con la Aplicación broadcaster a través de banda ancha. La colección de archivos que componen se puede entregar a través de la web de manera estándar o a través de la transmisión como paquetes a través del protocolo ROUTE.

- **Paquete de entrada**

Contiene uno o más archivos que comprenden la funcionalidad de la aplicación Broadcaster, donde incluye la página de entrada y archivos de apoyo adicionales, incluidos JavaScript, CSS, archivos de imagen y otro contenido.

- **Página de entrada**

Es el documento HTML5 inicial al que hace referencia la señalización de la aplicación que debe cargarse primero en el agente de usuario, ya que es el archivo del paquete de entrada.

- **Agente de usuario**

Software proporcionado por el receptor, para recupera y presenta contenido web. El agente de usuario interpreta HTML5, CSS y JavaScript, procesa medios, texto y gráficos, puede crear cuadros de diálogo de interacción con el usuario.

- **Caché de contexto de aplicación**

Área de almacenamiento conceptual donde se almacena la información de la transmisión para su recuperación a través del servidor web del receptor. Los archivos entregados a través de ROUTE contienen atributos que establecen con que caché se asociarán.

- **Identificador de contexto de aplicación**

Es un URI único que determina los recursos proporcionado por el receptor a una aplicación de difusor asociada. Los recursos están asociados con múltiples identificadores de contexto, pero una aplicación de radiodifusión solo se asociara con un único identificador de contexto de aplicación.

- **URI base**

La URI base determina la parte inicial de una URL utilizada por la aplicación de transmisión para acceder a los archivos dentro de la caché de contexto de la aplicación. Incluso esta URI base se antepone a la ruta URI relativa de un archivo para obtener la URL completa del archivo dentro de la caché de contexto de la aplicación. URI base es generado de forma exclusiva por el receptor en función del identificador de contexto de la aplicación definido para la aplicación del difusor.

- **Servidor web del receptor**

Componente conceptual del receptor para proporcionar un medio para el agente de usuario y obtenga acceso a los archivos entregados a través de ROUTE que se encuentren en la caché de contexto de la aplicación.

- **Servidor WebSocket del receptor**

Proporciona el medio para que el agente de usuario acceda a la información sobre el receptor y pueda controlar algunas funciones proporcionadas por el mismo como: número de puerto + dirección IP.

- **Mensajes JSON-RPC**

Proporciona las API que requiere la aplicación Broadcaster para acceder a los recursos, tales como consultar información recopilada en el receptor, así como recibir notificaciones mediante señalización de emisión y solicitar la realización de acciones.

### **Diferencias entre un entorno de web normal y un entorno ATSC 3.0 en HTML5**

Existen diferencias entre una aplicación HTML5 implementada en un entorno web normal y una implementada en un entorno de transmisión ATSC 3.0. En el entorno de transmisión ATSC 3.0, una aplicación de transmisión puede:

- Acceder a recursos de radiodifusión o banda ancha.
- Solicitar a los receptores que realicen determinadas funciones como: utilizar el reproductor de medios proporcionado por el receptor, MSE y EME para reproducir contenido multimedia por transmisión o banda ancha.
- Consultar información específica para la recepción de servicios de TV, por ejemplo, el estado de la visualización de subtítulos, referencias de idioma o recibir notificaciones de cambios en esta información.
- Recibir notificaciones de eventos de transmisión incrustados en el contenido de medios o en la señalización, cuando el receptor está reproduciendo ese contenido.

Otra diferencia notable entre los dos modelos es que en el entorno web normal, el espectador tiene el control directo del lanzamiento de una aplicación HTML5 definiendo la URL de un sitio web deseado. En el entorno ATSC 3.0, aunque el usuario inicia la acción seleccionando un servicio, el espectador no selecciona explícitamente la URL de la aplicación real y en su lugar es proporcionada mediante señalización de difusión.

La aplicación Broadcaster se basa en un conjunto de funciones derivadas del agente de usuario. La Tabla 1 muestra qué tipo de API utiliza una aplicación proporcionada por el emisor para acceder a las funciones proporcionadas por el receptor.

**Tabla 1***API para ejecución de aplicación*

<b>Acción solicitada por la aplicación</b>	<b>Tipo de API utilizada por la aplicación</b>
Solicitar descargar un archivo multimedia de banda ancha	W3C APIs utilizan el agente – usuario
Consultar información relacionada con la visualización del usuario y las preferencias de presentación, incluidos los idiomas, las opciones de accesibilidad y la configuración de subtítulos.	API de Receiver WebSocket Server
Solicitando transmitir el archivo multimedia descargado desde la transmisión.	A través del modelo GET o POST
Solicitando transmitir el archivo multimedia descargado desde banda ancha.	A través del modelo GET o POST
Solicitar al Receiver Media Player que reproduzca un flujo de medios entregado por banda ancha.	API de Receiver WebSocket Server
Recibir notificaciones de eventos de transmisión que se envían como parte de ROUTE/DASH durante la transmisión.	API de Receiver WebSocket Server
Consultar al receptor para conocer la identidad del servicio de transmisión seleccionado actualmente.	API de Receiver WebSocket Server
Recibir notificación de cambios en las preferencias de presentación y visualización del usuario.	API de Receiver WebSocket Server
Solicitar al receptor que seleccione un nuevo servicio de transmisión.	API de Receiver WebSocket Server

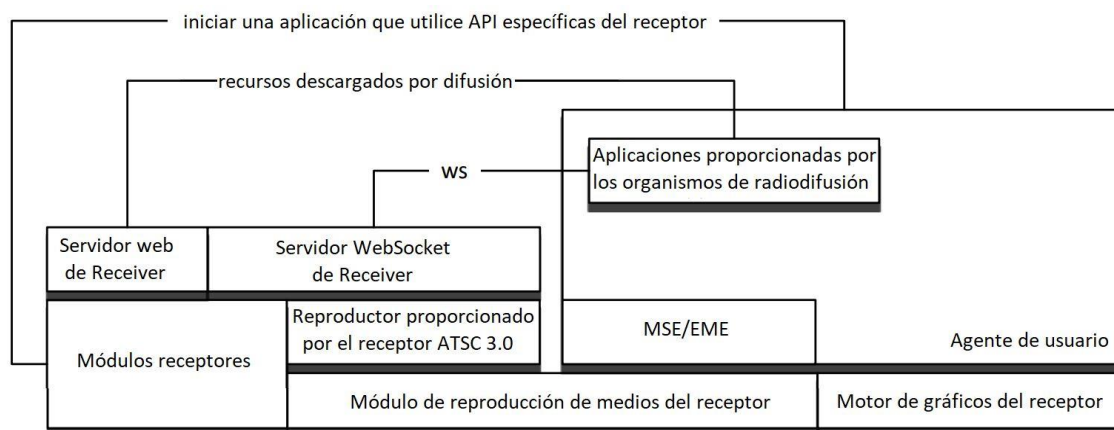
*Nota.* La tabla muestra las API para ejecutar la aplicación. Tomado de “ATSC 3.0 Interactive Content (A/344)”, (ATSC, 2019).

### Modelo de receptor ATSC 3.0

Un modelo de receptor de referencia ATSC 3.0 se muestra en la Figura 11, presenta una arquitectura de capas, el cual debe ejecutar un agente de usuario HTML5 que cumpla con los requisitos normativos de las API de medios web de CTA-5000.

### Figura 11

*Componentes lógicos del receptor en ATSC 3.0*



*Nota.* El gráfico muestra los componentes lógicos del receptor en ATSC 3.0. Tomado de “ATSC 3.0 Interactive Content (A/344)”, (ATSC, 2019).

Cada archivo que se envía a través de banda ancha tiene asociada la URL absoluta habitual. El archivo entregado mediante difusión tiene una referencia de URL, señalada en la transmisión, además tiene uno o más identificadores de contexto de aplicación. Los receptores asignan a cada archivo de transmisión un URI base que convierte la referencia relativa en una o más URL absolutas, considerando los identificadores de contexto de aplicación.

El identificador de contexto de aplicación es un URI único que determina qué recursos proporciona el receptor a una aplicación de difusor asociada. Cada identificador forma un entorno conceptual único en el que se espera que el receptor combine los recursos para que los utilicen las aplicaciones de radiodifusión asociadas. Este entorno conceptual único se denomina caché de contexto de aplicación.

### **Origen de un recurso web**

El origen de un recurso web se define en RFC 6454, las URL resultantes deben estar familiarizados con una parte del esquema “http” y una autoridad (dirección IP o nombre de host) o un número de puerto (10.2.12.45:8080).

El algoritmo utilizado por un receptor ATSC 3.0 para generar la porción de URL y determinar el origen de un archivo de transmisión deberá cumplir con las restricciones que se especifica a continuación.

Un recurso de banda ancha tiene un origen definido por el servidor web que aloja el recurso, la estructura de URL proporcionado por el receptor para cualquier recurso dentro de una caché de contexto de aplicación será:

`<baseURI>/<pathToResource>`

donde:

`<baseURI>` se define como `<scheme>://<authority>[/<pathPrefix>]`

`<scheme>` y `<authority>` Son elementos URI del estándar como se define en RFC 3986,

`<pathPrefix>` Prefijo de ruta opcional antepuesto a la ruta; y

`<pathToResource>` Es la ruta relativa proporcionada por ROUTE

### **Aplicación broadcaster**

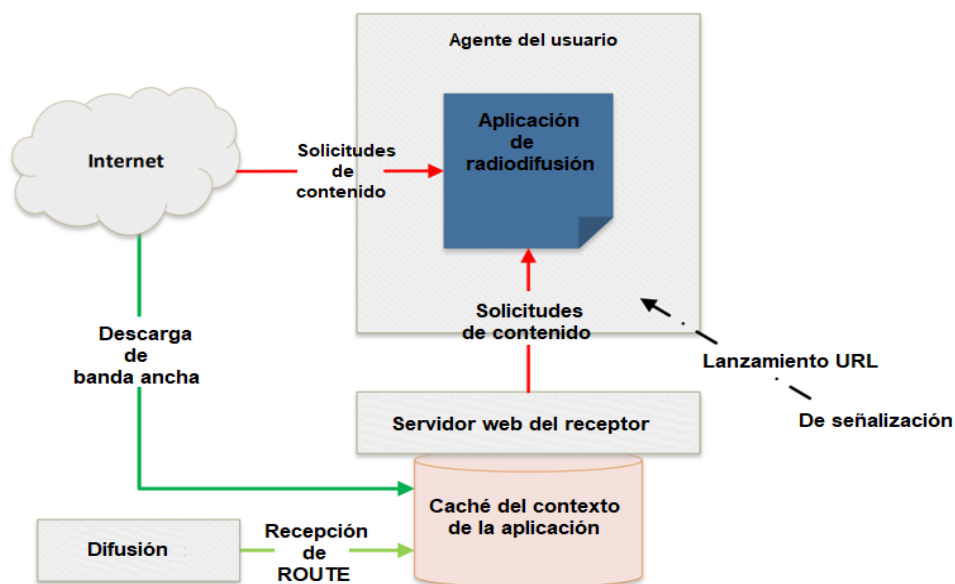
Para la gestión de aplicaciones en ATSC 3.0 se considera el conjunto de documentos de HTML5, JavaScript, CSS, XML, archivos de imagen y multimedia que se entregan por separado o juntos dentro de uno o más paquetes. En la Figura 12 se muestra las relaciones entre varios conceptos dentro de una arquitectura de receptor de referencia



distribuida, es decir, el servidor web del receptor está en un dispositivo físico separado del agente de usuario.

## Figura 12

*Relación del servidor web del receptor y agente de usuario*



*Nota.* El gráfico muestra la relación entre el servidor web del receptor y el agente de usuario. Tomado de “ATSC 3.0 Interactive Content (A/344)”, (ATSC, 2019).

El proceso para la gestión de la aplicación en ATSC 3.0 se describe de la siguiente manera:

- Inicia después de que el receptor obtiene la información de señalización de la aplicación para reenviar la URL de inicio al agente de usuario.
- El agente de usuario carga la página de entrada de la aplicación desde la URL en un servidor de internet o a un servidor web del receptor, dependiendo del formato en la señalización de la aplicación de servicio.
- El mecanismo específico de comunicar la URL de entrada de la aplicación que emite los canales al agente de usuario es un detalle de implementación del

receptor. Sin embargo, la URL de entrada tiene argumentos específicos que deben proporcionarse.

- Una vez que se ha cargado la página principal de entrada de la aplicación se puede comenzar a solicitar contenido de varias URL locales o externas a través de JavaScript o solicitudes href del estándar HTML5 en la forma compatible con W3C.
- Se asume que cualquier contenido recibido por difusión a través de la entrega de archivos ROUTE está disponible en la caché de contexto de aplicación y se accede por el servidor web del receptor. Esta especificación no hace afirmaciones sobre cómo se hace esto ni se implementa el caché. Sin embargo, describe cómo acceder a los recursos mediante solicitudes HTTP al servidor web del receptor.
- El agente de usuario admite varios mecanismos de almacenamiento W3C locales y en caché interno de contenido.
- Los mecanismos de almacenamiento internos compatibles con W3C implementados dentro del agente de usuario no deben confundirse con el caché de contexto de la aplicación que se muestra por separado.
- La aplicación puede utilizar interfaces W3C estándar para descubrir y utilizar las distintas instalaciones de almacenamiento del agente de usuario.

### ***WebSocket Interface***

Una interfaz webSocket está constituida por la dirección IP y el puerto. La aplicación que transmite la señal de los canales intercambia información con la plataforma del receptor con el fin de:

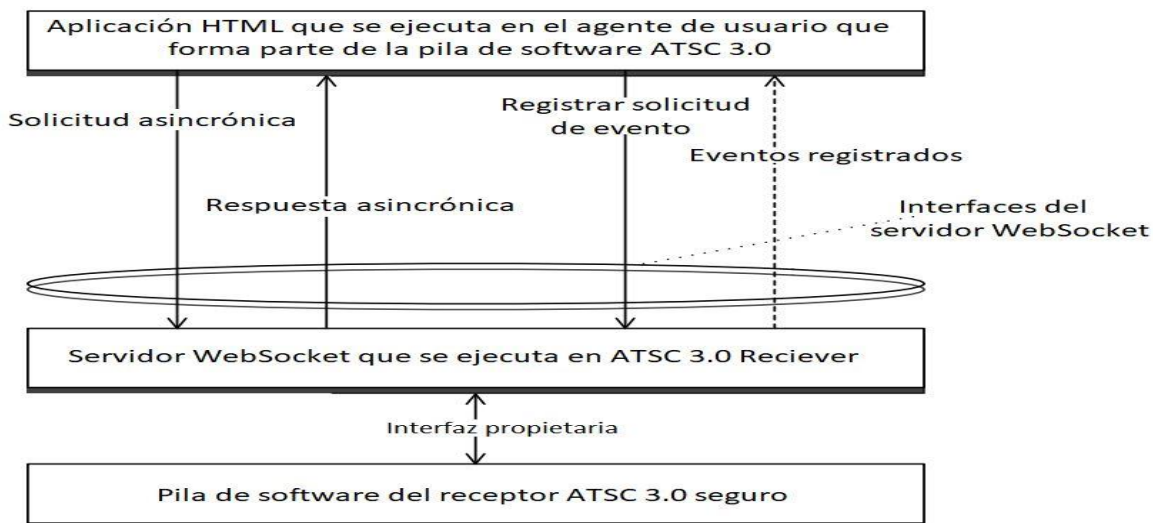
- Recuperar la configuración del usuario.

- Recibir un evento del receptor a la aplicación o plataforma que emite los canales basado en notificación de cambio en una configuración de usuario y *Event Stream* estilo DASH.
- Solicitar acciones del receptor.

El receptor admite estas funciones debido a que dispone un servidor web y un conjunto de llamadas a procedimiento remoto de WebSocket (RPC). Estas RPC se aplican para el intercambio de información de transmisión de canales entre la aplicación o plataforma, esto se ejecuta en el receptor. En la Figura 13 se muestra la arquitectura de receptor centralizada, donde solo se ingresa al servidor web desde Receiver por la aplicación en el agente de usuario.

**Figura 13**

*Comunicación con receptor ATSC 3.0*



*Nota.* El gráfico muestra la comunicación con receptor ATSC 3.0. Tomado de "ATSC 3.0 Interactive Content (A/344)", (ATSC, 2019).

El receptor presenta una o más interfaces ATSC 3.0 WebSocket y admitidos para comando y control. Algunos receptores utilizan hasta tres interfaces WebSocket adicionales, empleados en video, audio y datos binarios de subtítulos.

### ***Enlace de interfaz***

Las API utilizan una interfaz WebSocket para transmitir y recibir señales depende de la funcionalidad del navegador para abrir la conexión y no se requiere una funcionalidad específica. La comunicación con el servidor WebSocket proporcionado por el receptor para la transmisión y recepción de señales deben identificar la URL del servidor. Cabe mencionar que la ubicación depende de la topología de la red o según la implementación del receptor.

Para cubrir las diferencias en la transmisión y recepción de señales se ingresa con la URL de la página de entrada a través de uso del parámetro de consulta con el propósito de obtener información de la ubicación del servidor WebSocket del receptor. Al momento de cargar la página de entrada en la transmisión y recepción de señales, la URL posee un término de consulta para generar la URL base de la interfaz ATSC 3.0 WebSocket compatible con los receptores.

De igual modo, el receptor comunica la versión actual de la API de WebSocket compatible cuando genera consulta que contenga la fecha de publicación del estándar. Para la consulta de utiliza la sintaxis ABNF, tal como se detalla a continuación:

```
query = (wsQuery "&" revQuery) / (revQuery "&" wsQuery)
      wsQuery = "wsURL=" ws-url
      revQuery = "rev=" yyyyymmdd
```

El ws-url es la URL de WebSocket empleando según lo establecido en RFC 6455. El valor aaaammdd posee el año (aaaa), mes (mm) y el día (dd), es así que cuando la primera versión se publicó el 18 de diciembre del 2017, el valor se representó como “20171218”. Para esto se toma en cuenta la columna fecha que se tiene en la tabla historial. Respecto a la consulta del inicio de la transmisión y recepción de señales cuando la URL de la página se entrada se presenta como:

```
http://localhost/xbc.org/x.y.z/home.html
```

Las API de WebSocket utiliza el estándar publicado el 20 de julio de 2018, por lo que fue lanzada de la siguiente manera:

```
http://localhost/xbc.org/x.y.z/home.html?wsURL=wss://localhost2:8000&rev=20180720
```

Los parámetros de consulta wsURL y rev se emplean para cargar una URL de la página de entrada. Cuando aparece un parámetro de consulta wsURL en la dirección de la solicitud de HTTP el servidor web de banda ancha ignore la misma. En el lanzamiento de las aplicaciones basado en radiodifusión y banda ancha se utiliza el término de consulta rev.

El receptor admite el acceso a la interfaz WebSocket para comunicarse con las API. Cabe señalar que el receptor que admite la entrega en modo push de datos (video, audio, subtítulos) acepta tres interfaces WebSocket, para los tipos de datos multimedia.

Incluso, los receptores que aceptan el estándar de dispositivo complementario A/338 también generan una interfaz WebSocket adicional que permite la comunicación con el administrador de CD dentro del receptor. La Tabla 2 describe las cuatro interfaces, donde el término *WSPath* representa el valor del parámetro wsURL.

**Tabla 2***Función interfaz*

<b>Función interfaz WebSocket</b>	<b>URL</b>	<b>Soporte del receptor</b>
Comando y control	<i>WSPath/atscCmd</i>	Requerido
Video	<i>WSPath/atscVid</i>	Opcional
Audio	<i>WSPath/atscAud</i>	Opcional
Subtítulos	<i>WSPath/atscCap</i>	Opcional
Dispositivo complementario	<i>WSPath/atscCd</i>	Opcional

*Nota.* La tabla muestra la función de interfaz. Tomado de “ATSC 3.0 Interactive Content (A/344)”, (ATSC, 2019).

Cuando existe incompatibilidad de una URL de WebSocket, el receptor responderá con el mensaje de error de error “404 Not Found” de HTTP, esto significa que hay fallas en la conexión a la API. En el modelo push, los archivos del segmento de medios MPEG DASH enviado mediante la interfaz WebSocket se entregan en forma binaria, pues emplea tramas de texto.

### ***Enlace de datos***

La recepción y envío de mensajes se presenta cuando ya se tiene una conexión entre el servidor de control y comando del receptor WebSocket.

No obstante, WebSocket posee una interfaz bidireccional simple, por lo que es importante establecer una sintaxis de mensajes que se establecen en el esquema JSON. Estos se codifican en cadena como UTF-8, donde el receptor analiza el esquema JSON con el fin de direccionar el método al controlador para su posterior procesamiento. Los mensajes para la interfaz de WebSocket se dividen en:

- **Mensaje de solicitud:** Aplicada para requerir información o comenzar con una acción.
- **Respuesta síncrona:** Se basa en una respuesta definitiva e inmediata a un requerimiento.
- **Respuesta asincrónica:** Se basa en una respuesta definitiva a un requerimiento derivada de manera asíncrona.
- **Respuesta de error:** Se trata de un error presentado en el requerimiento.
- **Notificación:** Notificación unidireccional.

La codificación para la solicitud y respuesta es la siguiente: en la cual se envía la versión de jsonrpc, el método, los parámetros y el id de la solicitud. En la parte de recepción debe coincidir la versión de jsonrpc y el id:

```
--> {"jsonrpc": "2.0",  
     "method": "exampleMethod1",  
     "params": 1,  
     "id": 1  
}  
  
<-- {  
     "jsonrpc": "2.0",  
     "result": 1,  
     "id": 1  
}
```

La codificación para la notificación se muestra a continuación, para esto se utiliza el método update y un vector con los parámetros de la notificación.

```
<-- {  
     "jsonrpc": "2.0",  
     "method": "update",  
     "params": [1,2,3,4,5]  
}
```

### ***Métodos soportados en ATSC 3.0***

- **API: Consultas al Receptor**

El software del receptor presenta las API de WebSocket a la aplicación con el propósito de recuperar la configuración y datos del usuario. Cuando no se tiene las configuraciones en el receptor, la aplicación encargada de la transmisión de canales emplea valores predeterminados. Una plataforma encargada de la transmisión de señales de canales puede utilizar su propia configuración de interfaz de usuario y guardar como *cookies* en el receptor. La configuración o información que se recupera es la siguiente:

- La configuración de clasificación de asesorar el contenido.
- El estado de visualización de subtítulos habilitados, deshabilitados tamaños de fuente, estilos, colores, entre otros.
- ID de servicio actual.
- Preferencias de idioma (audio, interfaz de usuario, subtítulos, etc.).
- Las preferencias de visualización de subtítulos.
- Preferencias de accesibilidad de audio.
- La URL de la aplicación para transmisión y recepción de señales aplicadas para obtener el MPD de transmisión actual.
- La URL de servidor web del receptor para acceder a la caché de contexto de aplicación.
- Período de XLink.

Las siguientes API se emplean para que las aplicaciones que emiten la señal de canales recuperen la información:

- **API de preferencias de lenguaje**



Esta API se utiliza por la aplicación Broadcaster cuando se desee conocer la configuración de idioma en el receptor, incluido el idioma seleccionado para la salida de audio y subtítulos.

Para realizar la consulta, la aplicación Broadcaster podría emitir la siguiente API:

```
--> {
  "jsonrpc": "2.0",
  "method": "org.atsc.query.languages",
  "id": 95
}
```

El receptor utiliza la siguiente codificación: de la preferencia establecida en el idioma para las pistas de audio y los subtítulos.

```
<-- {
  "jsonrpc": "2.0",
  "result": {
    "preferredAudioLang": "es",
    "preferredUiLang": "en",
    "preferredCaptionSubtitleLang": "es"
  },
  "id": 95
}
```

- **API de notificación de cambio de idioma**

Cuando el usuario cambia el idioma definido: audio/ subtítulos. La notificación del cambio será emitida por el receptor a la aplicación con la siguiente API:

```
<-- {
  "jsonrpc": "2.0",
  "method": "org.atsc.notify",
  "params": {
    "msgType": "langPref",
    "preferredCaptionSubtitleLang": "fr-CA"
  }
}
```

Notificación que se proporciona cada vez que el usuario cambia el idioma preferido.

## API de solicitud de caché

Esta API se utiliza por la aplicación Broadcaster para solicitar que el receptor descargue los elementos u objetos de un servidor de banda ancha y ubicarlos en un caché. La URL permite identificar de forma individual cada archivo.

Por ejemplo, la aplicación Broadcaster solicita la descarga de tres archivos PNG desde un servidor de banda ancha en `http://foo.com/service1` y almacena en la cache de la aplicación en subdirectorios es decir en: `images/subdirectorio`. Los códigos de filtro que se asociarán con los archivos son 1007 y 1009. El origen de los tres archivos son los siguientes:

### 1. Archivo 1:

Fuente: `http://foo.com/service1/A/big-image1.png`

Ubicación de destino en la caché de la aplicación: `images / A / big-image1.png`

### 2. Archivo 2:

Fuente: `http://foo.com/service1/B/big-image2.png`

Ubicación de destino en la caché de la aplicación: `images / B / big-image2.png`

### 3. Archivo 3:

Fuente: `http://foo.com/service1/C/big-image3.png`

Ubicación de destino en la caché de la aplicación: `images / C / big-image3.png`

Para realizar la descarga, la aplicación Broadcaster podría emitir la siguiente API:

```
--> {
  "jsonrpc": "2.0",
  "método": "org.atsc.CacheRequest",
  "params": {
    "sourceURL": "https://foo.com/service1/",
    "targetURL": "imágenes /",
    "URL":["A / imagen-grande1.png", "B / imagen-grande2.png", "C / imagen-grande3.png"],
    "filtros": [1007, 1009]
  },
  "identificación": 37
}
```

Cuando en el proceso existen archivos no recuperados previamente, el receptor utiliza la siguiente codificación:

```
<-- {
  "jsonrpc": "2.0",
  "result": {"cached": false},
  "id": 37
}
```

Si la aplicación Broadcaster quiere verificar si los dos primeros archivos se han descargado correctamente, utiliza la codificación detallada:

```
--> {
  "jsonrpc": "2.0",
  "method": "org.atsc.CacheRequest",
  "params" {
    "targetURL": "images/",
    "URLs":["A/big-image1.png", "B/big-image2.png"]
  },
  "id": 38
}
```

Si ambos archivos están presentes, el receptor responde con la siguiente codificación:

```
<-- {
  "jsonrpc": "2.0",
  "result": {"cached": true},
  "id": 38
}
```

### ***Árbol de carpetas de la aplicación en ATSC 3.0***

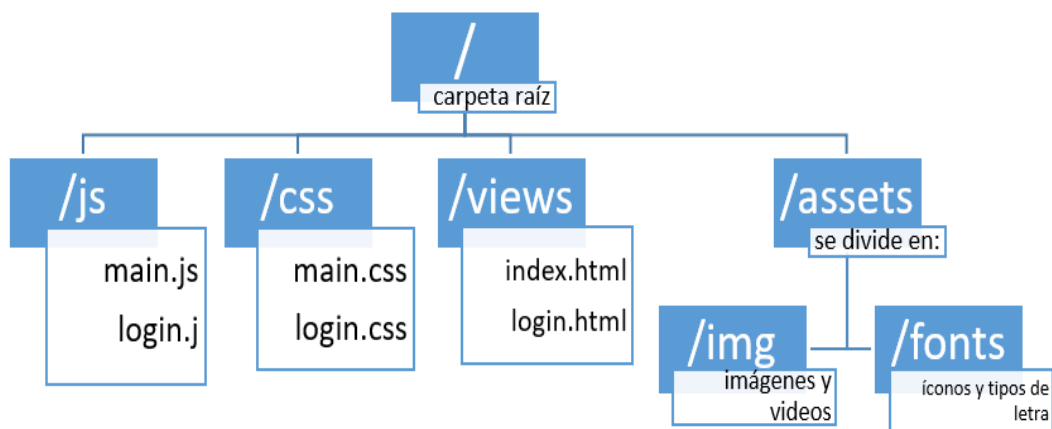
A continuación en la Figura 14, se muestra un ejemplo de árbol de las carpetas generado en un proyecto 3WC. En primer lugar, se tiene una carpeta raíz, seguido de una carpeta /js que correspondiente al código JavaScript que permite desarrollar la interactividad entre el cliente y la aplicación.

Después, se encuentra la carpeta /css, esta contiene librerías que dan estilo a las pantallas de la aplicación. Otra carpeta es /views donde se encuentra las vistas

desarrolladas en HTML5 y finalmente, se encuentra la carpeta `/assets`, la cual se subdivide en dos carpetas. La primera corresponde a `/img` para almacenar recursos visuales como imágenes, videos y `/fonts` que ayuda a guardar íconos y tipografía.

**Figura 14**

*Árbol de carpetas*



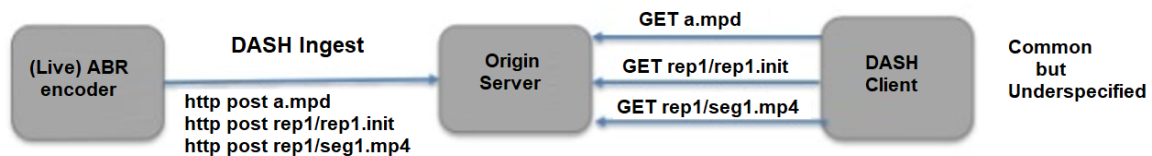
*Nota.* El gráfico muestra el árbol de carpetas generada de un proyecto 3WC. Tomado de “ATSC 3.0 Interactive Content (A/344)”, (ATSC, 2019).

### **DASH INGEST**

La comunicación entre los paquetes DASH y la capa de transporte se da a través de DASH Ingest, denominado también CMAF Ingest. La especificación CMAF o *Common Media Application Format*, es un conjunto de reglas que se utilizan para regular cómo se almacenan las pistas de audio, video y metadatos. Si bien CMAF tiene su propia especificación, muchas de las pautas son un subconjunto de la especificación MP4 mucho más grande, formalmente conocida como ISOBMFF. En la Figura 15 se muestra como el Dash Ingest utiliza el método POST para comunicarse con el servidor, mientras que utiliza el método GET para las peticiones entre el cliente y el servidor de origen.

**Figura 15**

*Comunicación y peticiones con POST, GET*



*Nota.* El gráfico muestra la comunicación y petición utilizando POST, GET respectivamente. Tomado de “ATSC 3.0 Interactive Content (A/344)”, (ATSC, 2019).

## CAPÍTULO III

### Desarrollo de la aplicación interactiva en base al estándar ATSC 3.0

#### Software del sistema

Para el desarrollo del sistema se utilizó el editor Vscodé para crear los documentos que se muestran en un navegador web, mediante el lenguaje HTML5. Para realizar solicitudes mediante los métodos GET, POST e interactuar con los elementos HTML de la página web se empleó JavaScript.

Con la ayuda de librerías como Bootstrap que internamente utiliza CSS y JavaScript se puede modificar las propiedades visuales de los elementos de la página web, dividir el documento en secciones para mostrar videos, imágenes, textos, barras de navegación, entre otros.

En el lado del servidor se utilizó node.js el cual es un entorno multiplataforma que permite crear servidores HTML, servidores websocket, bases de datos y requiere conocer JavaScript para proporcionar estas funcionalidades.

#### Estructura de un documento HTML

Un documento HTML consiste de varias etiquetas con un propósito específico, entre las principales tenemos la Cabecera **<head>**, contiene información que no se mostrara en la página web, pero contiene metadatos acerca del tipo de codificación de los caracteres, el área visible de la página (viewport), links de archivos CSS o librerías externas. El cuerpo **<body>**, contiene los elementos que se mostraran en la página web, dentro de esta se pueden crear contenedores de otros objetos mediante la etiqueta **<div>** dentro de ellos se presentan imágenes, videos, botones cada uno de ellos identificados por una etiqueta como se muestra en la Figura 16 (Mozilla, n.d.).

## Figura 16

### Estructura básica de un documento HTML5

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
  </head>
  <body>
    <div id = "ex1" class = "exclass">
      <video src=""></video>
      <img src="" alt="">
    </div>

    <div id = "ex2" class = "ex2class">
      <video src=""></video>
      <img src="" alt="">
    </div>
  </body>
</html>

```

### Estructura de una aplicación web mediante bootstrap

Para distribuir el contenido en la página web se utilizó el sistema de malla de bootstrap el cual emplea una serie de contenedores, filas y columnas (Bootstrap, n.d.).

Toda etiqueta dentro de un documento HTML puede contener atributos como el id que nos permite identificar un solo elemento o class que nos ayuda a categorizar varias etiquetas, el atributo style es utilizado para añadir propiedades que no estén definidas en la librería bootstrap como zindex que nos permite seleccionar que contenedor se mostrara en la capa superior.

La etiqueta **<div>** requiere utilizar el atributo "class" debido a que bootstrap tiene definido internamente una serie de clases a las cuales les aplica estilos y formatos diferentes mediante archivos CSS, por lo que en la documentación el objetivo del diseñador es seleccionar el nombre de la clase más apropiada. Una estructura básica requiere conocer tres clases:

1. **Container:** Esta clase en el sistema de mallas funciona como el contenedor padre de otras clases y puede ser anidado.
2. **Row:** Esta clase permite crear una nueva fila.
3. **Col-breakpoint-cantidad columnas:** Esta clase requiere especificar un punto de quiebre y la cantidad de columnas para definir el tamaño que ocuparan elementos dentro de este tipo de contenedor.
  - **Breakpoint:** Permite adaptar el diseño a un determinado tamaño de dispositivo.
  - **Cantidad columnas:** Permite ajustar el ancho que ocupara un contenedor, el valor máximo que puede ocupar es de 12 columnas.

### **Estructura de la aplicación interactiva**

En esta sección se describe la distribución de los contenedores y elementos que se mostraran en pantalla.

#### ***Barra de navegación***

En la parte superior de la página web se presentara una barra de navegación con 2 botones para obtener el clima en base a la ubicación geográfica y enviar mensajes a un servidor websocket en formato json.



## Figura 17

### Barra de navegación

```

<!-- barra de navegación -->
<nav class="navbar navbar-expand-sm navbar-dark bg-dark">
  <div class="container">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item m-sm-2">
        <button id = "per" type="button" class="btn btn-outline-info"><i class="fas fa-caret-square-right"></i> Mensajes</button>
      </li>
      <li class="nav-item m-sm-2">
        <button id = "clim" type="button" class="btn btn-outline-light"><i class="fas fa-cloud"></i> Climax</button>
      </li>
      <li class="nav-item m-sm-2 pt-3">
        <span class="badge badge-pill badge-primary bg-primary">Q</span>
      </li>
      <li class="nav-item m-sm-2 pt-3">
        <span class="badge badge-pill badge-success bg-success">W</span>
      </li>
      <li class="nav-item m-sm-2 pt-3">
        <span class="badge badge-pill badge-warning bg-warning">E</span>
      </li>
      <li class="nav-item m-sm-2 pt-3">
        <span class="badge badge-pill badge-info bg-info">A</span>
      </li>
      <li class="nav-item m-sm-2 pt-3">
        <span class="badge badge-pill badge-light bg-light" style="color: ■ black;">5</span>
      </li>
      <li class="nav-item m-sm-2 pt-3">
        <span class="badge badge-pill badge-danger bg-danger">D</span>
      </li>
    </ul>
    <!-- espacio para mostrar detalles del clima -->
    <span id = "clima" class="navbar-text"></span>
  </div>
</nav>

```

Como se muestra en la Figura 17 la etiqueta **<nav>** contiene varias clases separadas por espacios entre ellas `navbar-dark`, `bg-dark` que dan el color negro de fondo, `navbar`, `navbar-expand` de acuerdo a la documentación son necesarias para que se pueda aplicar colores y la barra de navegación se adapte al tamaño del dispositivo donde se visualiza la página.

Dentro de la etiqueta **<nav>** se puede observar la etiqueta **<div>** con su clase `container` que será el contenedor de una lista sin orden denominada **<ul>**, cada elemento de la lista debe tener la etiqueta **<li>** que contendrá las clases `nav-item` que será un identificador de los elementos que forman parte de la barra de navegación y `m-sm-2` equivale a dar un margen de 8px.

En las dos primeras etiqueta `<li>` se crean botones mediante la etiqueta `<button>` la misma que contendrá el atributo `id` para identificar a cada botón y reaccionar a eventos `click` mediante JavaScript, las clases `btn`, `btn-light` que tienen los botones permiten darle un borde de color blanco. Dentro de cada botón se crea la etiqueta `<i>` para mostrar un icono, las clases `fas fa-carret`, `fas fa-cloud` permiten mostrar símbolos de flechas o nubes respectivamente. Es importante mencionar que antes de cerrar la etiqueta `</button>` se escribe el nombre que se mostrara dentro del botón.

Luego se crea un grupo de etiquetas `<li>` para mostrar los elementos del control remoto con su respectivo color y valor de tecla, se puede notar que a todos los elementos se les asigna la clase `m-sm-2` para dar un margen de `8px` y `pt-3` para dar un margen interior de `16px`, al interior se crea una etiqueta `<span>` con las clases `badge badge pill` para que cada elemento ocupe el tamaño del elemento padre y `badge-$color` y `bg-$color` para cambiar el color de fondo del elemento, en este caso la palabra `$color` debe ser modificada con diferentes nombres que llevan asociado un color. De la misma manera antes de cerrar la etiqueta `</span>` se escribe el nombre que se mostrara dentro del botón.

Fuera de la etiqueta `<ul>` es decir fuera de la lista se crea una etiqueta `<span>` con su respectivo identificador para posteriormente mostrar la temperatura y clima.

### ***Sección de visualización y descripciones***

En la parte restante de la página se mostrara una sección para el video que ocupara 9 columnas y en las 3 restantes la parte de descripciones.

## Sección de visualización

En la Figura 18 se muestra la primera etiqueta `<div>` con la clase `container-fluid` para ocupar todo el ancho disponible en la pantalla, el siguiente `<div>` con la clase `row` para definir los elementos que se mostraran en la primer fila.

### Figura 18

Código HTML para sección de visualización de videos y anuncio

```
<div class="container-fluid">
  <div class="row">
    <div id = "videosection" class="col-md-9" >
      <div id="videosection1" class="ratio ratio-16x9" style="z-index: 1;">
        <video id="naturevid" muted controls>
          <track id = "es_track" label="Spanish" kind="subtitles" srclang="es" src="subtitles/es/nature_es.vtt" default>
          <track id = "it_track" label="Italian" kind="subtitles" srclang="it" src="subtitles/it/nature_it.vtt" default>
        </video>
      </div>
    <!-- fila para mostrar anuncio 1 -->
    <div id = "anuncio1" class="row" style="margin-top: -15%; z-index: 2 ;position: relative; visibility: hidden;"> <!--
    <div class="col-sm-6 offset-sm-3 border border-3 border-primary rounded"> <!-- tamaño pequeño ocupa 3 col y medi
      <div class="badge bg-secondary text-wrap text-center" style="width: 100%;" >
        <h8>Sabes que porcentaje de agua representan los oceanos en la tierra.</h8>
      </div>
      <!-- incluir grupo de botones para opciones -->
      <div class="offset-sm-4 btn-group-sm mt-4" role="group" aria-label="descripcion botones">
        <button id="oceana" type="button" class="btn btn-primary">75%</button>
        <button id="oceana" type="button" class="btn btn-success">50%</button>
        <button id="oceanc" type="button" class="btn btn-warning">35%</button>
      </div>
    </div>
  </div>
</div>
```

El siguiente `<div>` será el contenedor del video que ocupara 9 columnas, dentro de este se define un contenedor en el cual se usa la clase `ratio ratio-16x9` que permite especificar la relación de aspecto de video en este caso 16x9, también se puede notar que aquí se utiliza el atributo `style` que es muy utilizado para crear propiedades que no están incluidas en la librería bootstrap como es `zindex`, este nos permite indicar cual capa se mostrara primero dentro del contenedor de video, aquí se le dio el valor de 1, que es el valor más bajo y corresponde a una capa inferior. Finalmente se añade la etiqueta `<video>` con su identificador, atributos como: Control para mostrar botones de inicio, para opciones en el video y `muted` para que el video este silenciado por defecto. Las etiquetas `<track>` permiten incluir subtítulos en este caso en español o italiano.

El siguiente **<div>** con la clase row en la segunda fila se mostrara un anuncio por lo que se identificara como anuncio1, aquí el atributo más relevante es style, los valores que tiene el mismo son los siguientes:

- **Margin-top:** Esta propiedad permite especificar un valor porcentual del margen deseado en caso de ser un valor negativo el contenedor ocupara el mismo espacio que la sección superior por lo tanto es útil para mostrar anuncios, el valor seleccionado en este caso es de -15%.
- **Z-index:** Permite seleccionar el nivel de la capa del contenedor en este caso un valor mayor corresponde a una capa superior por lo que se seleccionó el valor de 2 para el anuncio de forma que se situé sobre el contenedor del video.
- **Position:** Esta propiedad define el tipo de posicionamiento seleccionado para un elemento (w3schools, n.d.). El valor seleccionado es relative ya que permite que el elemento se aleje de su posición normal y es necesario para que la propiedad z-index tenga efecto.
- **Visibility:** Controla la visibilidad del contenedor en este caso el valor seleccionado fue hidden de manera que este oculto por defecto.

Seguido se define un nuevo contenedor que presenta las siguientes clases: col-sm-6, offset-sm-3 que nos ayudan a que el anuncio ocupe 6 columnas con un offset de 3 columnas para centrar el anuncio. Border-primary rounded proporciona un borde de color azul redondeado al anuncio. El ultimo **<div>** utiliza la clase badge permitiendo que este contenedor presente el mismo tamaño que el elemento padre (Bootstrap, n.d.). De esta manera el texto ocupara el mismo espacio de 6 columnas. Las clases bg-secondary, text-center permiten proporcionar un color azul con texto centrado.

La última sección es para mostrar un grupo de botones por lo que se ha utilizado la clase btn-group, para que los 3 botones se centren se utiliza la clase offset-sm-4 es

decir se desplazaran 4 columnas de izquierda a derecha. Como se puede observar cada etiqueta **<button>** contiene 3 identificadores diferentes permitiendo responder a 3 eventos click diferentes, las clases btn btn-primary se utilizan para dar un color azul a los botones.

El anuncio informativo utiliza las mismas propiedades del primero pero con valores diferentes, por ejemplo: Margin-top: -50% para que se situé en la parte superior del video, zindex: 2 para que se muestre sobre el video, visibilty: hidden para que se mantenga oculto.

### Sección de descripción

En esta sección se mostrara una imagen, una descripción del programa, un botón de información y un gráfico de barras.

### Figura 19

*Código HTML para sección descriptiva e información*

```
<div class="col-md-3">
  <div class="row">
    <!-- logo -->
    <div class="col-sm-12">
      
    </div>
    <!-- descripcion -->
    <div class="col-sm-12 text-center">
      <div class="badge bg-gradient bg-secondary text-wrap" style="width: 15rem;">
        <h7>Noticiero Espe : Documentales relacionados con ciencia y tecnologia episodio1.</h7>
      </div>
    </div>
    <!-- publicidad -->
    <div class="col-sm-12 mt-4 text-center">
      <button id = "infor" type="button" class="btn btn-outline-danger">
        Info Acerca del Canal
      </button>
    </div>
    <!-- espacio para diagrama de barras -->
    <div class="col-sm-12 mt-4">
      <canvas id="myChart"></canvas>
    </div>
  </div>
</div>
```

En la Figura 19 el contenedor padre ocupara 3 columnas, dentro de esta se ubicará el primer **<div>** que será un contenedor que ocupe todo el espacio del elemento padre disponible en este caso 12 columnas, aquí se utiliza una etiqueta **<img>** con el atributo `src` para indicar la ubicación de la imagen. Las clases utilizadas `img-fluid`, `rounded` permite que la imagen se ajuste al tamaño del dispositivo que solicita la página, el atributo `alt` permite mostrar un texto en caso de no cargarse la imagen.

El siguiente contenedor de 12 columnas se usa para mostrar una descripción, la etiqueta **<div>** usa la clase `badge` permitiendo que este contenedor presente el mismo tamaño del elemento padre. Las clases `bg-secondary` y `text-wrap` permiten proporcionar un color azul de fondo con el texto en su interior, el atributo `style` con su valor `width` expresado en unidades `rem`, cada unidad `rem` equivale a 16 px y en este caso permite aumentar el ancho que ocupara el texto.

El siguiente contenedor de 12 columnas incluye un botón con su respectivo identificador, las clases `btn btn-outline-danger` permiten que el color del borde del botón sea rojo y se pinte de color rojo al pasar el cursor sobre él.

Finalmente se añade un contenedor de 12 columnas con un margen superior de 4 que equivale a 24 px, aquí se creara una etiqueta **<canvas>** con su identificador que posteriormente se utilizara para mostrar un gráfico de barras.

### **Importación de archivos externos**

Los archivos que se pueden importar provienen de una ruta local en caso de almacenarlos en el servidor o en un CDN que es una red de entrega de contenido soportada por un grupo de servidores para transferir archivos necesarios. Los tipos de archivos más comunes incluyen hojas de estilo CSS, JavaScript, videos e imágenes (cloudflare, n.d.).

Existen 2 etiquetas importantes para importar archivos externos dentro de la etiqueta **<head>** o **<body>** del documento HTML estos son:

- **Etiqueta script:** Esta etiqueta se utiliza para contener código JavaScript o importar archivos externos JavaScript mediante el atributo src se especifica la ruta local o la dirección url del CDN.
- **Etiqueta link:** Esta etiqueta se utilizó para importar hojas de estilo CSS mediante el atributo href se especifica la ruta local o la dirección url del CDN.

### ***Importación de bootstrap***

Para que funcione bootstrap es necesario importar archivos CSS y JavaScript especificando la ruta del CDN o descargando una copia de esta información de manera local.

### **Figura 20**

#### ***Importación de archivo CSS para Bootstrap***

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-+0n0xVW2e
```

### **Figura 21**

#### ***Importación de archivo JavaScript para Bootstrap***

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-gtEjrD/SeCtmISkJKNUaaKMoLD0,
```

Como se muestra en la Figura 20 mediante la etiqueta **<link>**, atributo href se especifica el url del CDN que contiene la hoja de estilos Bootstrap.min.css, además se puede notar que existe un atributo rel indicando que es una hoja de estilos y el atributo integrity utilizado en la actualidad para proveer de mayor seguridad ya que verifica que el recurso transferido no haya sido manipulado.

El segundo archivo mostrado en la Figura 21 se importa mediante la etiqueta `<script>`, el atributo `src` especifica el url del CDN que contiene el archivo `bootstrap.bundle.min.js` y el atributo `integrity` verifica que el archivo fuente no haya sido manipulado.

Es importante mencionar que el archivo CSS se importa dentro de la etiqueta `<head>`, mientras que el archivo JavaScript antes de cerrar la etiqueta `<body>`.

### ***Importación de chart.js***

En la Figura 22 se muestra la etiqueta `<script>` y el atributo `src` con el url del CDN de `chart.js` que permite crear diferentes tipos de gráficos en este caso se creara uno de barras con los datos de las votaciones recuperadas de una base de datos en el servidor.

### **Figura 22**

*Importación de archivo JavaScript chart.js*

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.3.2/chart.min.js"></script>
```

### ***Importación de dash.js***

En la Figura 23 se muestra la etiqueta `<script>` y el atributo `src` con el url del CDN de `dash.js` que permite crear un reproductor de contenido MPEG-DASH.

### **Figura 23**

*Importación de archivo JavaScript dash.js*

```
<script src="http://cdn.dashjs.org/latest/dash.all.min.js"></script>
```

### ***Importación de clnt.js***

En la Figura 24 se muestra la etiqueta `<script>` y el atributo `src` que incluye el nombre de la carpeta seguido el nombre del archivo `clnt.js`, este archivo será utilizado para interactuar con el DOM.



## Figura 24

*Importación de archivo JavaScript clnt.js*

```
<script src="javascript/clnt.js"></script>
```

## Creación y Reproducción de contenido MPEG-DASH

### FFMPEG

Esta herramienta fue diseñada para el procesamiento de archivos de audio/video, está basado en líneas de comandos. Se utilizará para crear 3 versiones del archivo nature.mp4, en la Tabla 3 se mostrará las características de cada versión.

**Tabla 3**

*Características de audio/video de cada versión*

<b>Versión</b>	<b>Máxima tasa bit</b>	<b>Escala de video</b>	<b>Tasa bit de audio</b>	<b>Frec. muestreo de audio (Hz)</b>	<b>Duración de GOP</b>
<b>Versión 1</b>	1500k	720	256k	48000	25
<b>Versión 2</b>	800k	540	128k	44100	25
<b>Versión 3</b>	400k	360	64k	22050	25

*Nota.* La tabla muestra las características de audio/video de las tres versiones creadas.

La última columna de la Tabla 3 muestra la duración del grupo de imágenes (GOP) es de 25 segundos, esto se debe a que en la documentación se menciona que los archivos mp4 deben ser codificados con la misma duración de GOP para que el comando mp4dash funcione como se verá a continuación.

## **Bento4**

Bento4 permitirá convertir varios archivos de entrada a una presentación MPEG DASH para lo cual se requiere los siguiente (Bento4, s.f.):

- Instalar python3.7 o superior.
- Descargar la carpeta zip que contiene archivos ejecutables y scripts batch disponible en: <https://www.bento4.com/downloads/>.
- Extraer la carpeta zip en cualquier directorio.

Luego se debe seguir los siguientes pasos:

1. Abrir una terminal en el directorio que contiene los archivos ejecutables de bento4.
2. Fragmentar las 3 versiones creadas especificando la duración de cada fragmento mediante el siguiente comando:

```
mp4fragment --fragment-duration 4000 version1.mp4 version1-fragmented.mp4
```

```
mp4fragment --fragment-duration 4000 version2.mp4 version2-fragmented.mp4
```

```
mp4fragment --fragment-duration 4000 version3.mp4 version3-fragmented.mp4
```

En este caso se seleccionó una duración de 4000 ms. Como resultado se obtendrá 3 versiones fragmentadas.

3. Empaquetar las 3 versiones fragmentadas en una sola presentación mediante el comando:
 

```
mp4dash version1-fragmented.mp4 version2-fragmented.mp4 version3-fragmented.mp4
```

Al completar el paso 3 se creará una carpeta llamada "output" que contendrá una carpeta de audio y otra de video cada una con 3 subcarpetas que almacenan varios segmentos correspondientes a las 3 versiones creadas, además se creará un archivo llamado streams.mpd que permitirá que el reproductor dash compute diferentes urls para cada segmento y así poder seleccionarlos dependiendo del ancho de banda disponible.

### **dash.js**

Es una librería basada en JavaScript que funciona como un reproductor MPEG DASH, su uso comercial es libre, cuenta con algoritmos de adaptación que permiten seleccionar segmentos de diferentes calidades en función del ancho de banda de la red.

Como se mencionó en la sección de importación el requisito es especificar la dirección url del CDN de dash.js para usar esta librería.

### **Figura 25**

*Código en JavaScript para crear un reproductor MPEG-DASH*

```
<script>
(function(){
  var url = "http://localhost:3000/output/stream.mpd";
  var TTMLRenderingDiv = document.querySelector("#caps");
  var player = dashjs.MediaPlayer().create();
  var video = document.querySelector("#naturevid");
  player.initialize(video, url, true);
})();
</script>
```

Como se observa en la Figura 25 dentro de la etiqueta **<script>** se deben seguir los siguientes pasos para que funcione el reproductor:

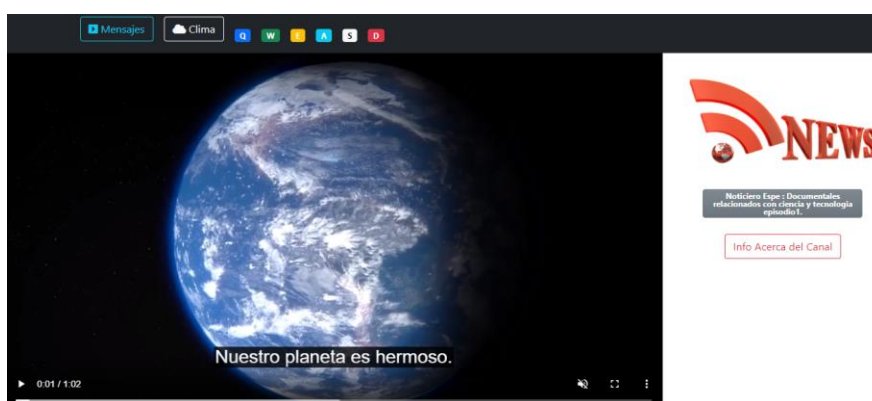
1. Almacenar en la variable url la dirección del archivo stream.mpd creada mediante bento4.
2. Crear una instancia del reproductor y almacenar este objeto en la variable "player".
3. En la variable video seleccionar el objeto con id "naturevideo" que representa a la etiqueta de video.
4. Llamar al método initialize y pasar como parámetro el objeto video, la url del archivo mpd y el valor booleano "true" para que se reproduzca el contenido automáticamente.

## Resultado de la página web

En la Figura 26 se puede visualizar la página cargada en el navegador Google Chrome, la barra de navegación en la parte superior muestra las teclas con su respectivo color que se deberán presionar cuando se añada la etapa de interactividad. Con la ayuda de bootstrap se logra aplicar estilos CSS que permiten que sea atractiva la presentación.

### Figura 26

*Página web cargada en el navegador*



## Etapa de interactividad

En esta etapa se utiliza JavaScript que es un lenguaje de alto nivel utilizado en el lado de cliente para interactuar con los elementos HTML mediante el modelo de objetos del documento (DOM).

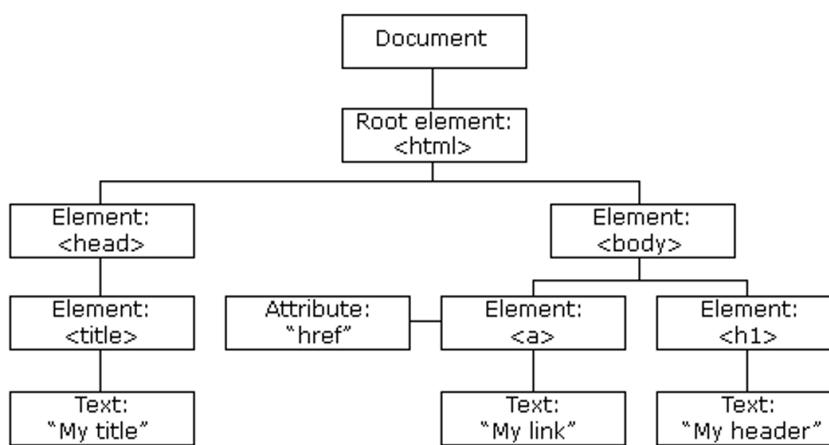
### ***Document Object Model (DOM)***

El DOM es una representación orientada a objetos de la página web permitiendo interactuar con la misma mediante JavaScript (Mozilla, s.f.). Dentro de las etiquetas script generalmente se inicia utilizando la API para los elementos Window o document.

En resumen, cada etiqueta del documento HTML es un objeto estructurado en forma de árbol como se muestra en la Figura 27, este modelo define las propiedades de todos los elementos HTML, los métodos para accederlos y sus respectivos eventos.

**Figura 27**

*Modelo DOM estructura de árbol*



Como se muestra en la Figura 27 hay objetos padres y dentro de ellos objetos hijos que pueden ser accedidos mediante métodos DOM o incluso se puede crearlos. Los objetos de mayor jerarquía son: Window y Document.

### **Métodos DOM utilizados**

Los métodos se definen como acciones que se aplican a los objetos, en el presente proyecto se utilizaron 3 métodos:

- **getElementById (id):** Este método permite obtener un objeto especificando su identificador "id", posteriormente retorna el objeto deseado y generalmente se lo asigna a una variable para modificar sus propiedades o reaccionar a eventos click y keypress, entre otros.
- **querySelector (selector):** Este método es similar al anterior permite obtener el primer elemento que coincide con el valor del selector en este caso el selector también será el identificador "id" y el resultado de este método se asigna a una variable para modificar propiedades y definir el código que se ejecutara de acuerdo a los eventos que disponga este objeto.

- **createElement (etiqueta):** Este método permite crear un nuevo elemento especificando el nombre de la etiqueta.
- **appendChild (objeto\_hijo):** Este método permite crear un objeto dentro de otro.
- **setAttribute (atributo, valor):** Permite especificar un atributo para un objeto, en el proyecto es utilizado para añadir la ruta de una imagen que representa el clima al atributo src.
- **addEventListener (nombre\_evento, función):** Permite asignar una función cuando se inicie un evento.

### **Eventos utilizados**

- **Click:** En este caso se empleó el evento click que se activara cuando el usuario presione un botón mediante el mouse.
- **Keydown:** Para asignar una tecla del control remoto cuando se presione una tecla.

En el caso de asignar una tecla se debe registrar el código de las mismas, los códigos correspondientes se muestran en la Tabla 4:

**Tabla 4**

*Códigos utilizados al activarse el evento keydown*

Tecla Presionada	Código
Q o q	KeyQ
W o w	KeyW
E o e	KeyE
A o A	KeyA
S o s	KeyS
D o d	KeyD

*Nota.* La tabla muestra las teclas asociadas a cada evento de keydown.

**Playing:** Este evento se activa mientras el video se encuentre en reproducción, será usado para definir el tiempo en el cual se mostrara un anuncio con 3 opciones de votos.

**Cuechange:** Este evento se activara cuando cambie el texto de los subtítulos, será usado para enviar un mensaje en formato JSON con el lenguaje seleccionado hacia un servidor websocket.

**Onload:** Este evento inicia cuando todo el contenido de la página web es cargado. Este contenido incluye imágenes, videos, archivos JavaScript y CSS.

### ***Cambio de contenido HTML***

Existen propiedades que permiten modificar el contenido y estilos que se mostraran en la página entre ellos:

**textContent:** Para cambiar el texto dentro de una etiqueta HTML por ejemplo `<span>`.

**style:** Permite asignar una propiedad CSS que modifique como se visualiza un elemento en la pantalla, en el proyecto es utilizado para ocultar un anuncio mediante la propiedad "visibility".

### ***Métodos de comunicación***

#### **Métodos de solicitudes HTTP**

HTTP es un protocolo que permite intercambiar información entre cliente y el servidor, está basado en un esquema de solicitud y respuesta. Los métodos usados tienen una función específica que se describe a continuación:

### ***Método GET***

Es usado para recuperar información de un recurso especificado (Mozilla, n.d.). Este método será utilizado para obtener resultados de la base de datos y consultar características acerca del clima en la ubicación geográfica del usuario.

### ***Método POST***

Es usado para enviar datos a un recurso especificado. (Mozilla, n.d.). En el proyecto es usado para enviar el resultado de un voto hacia el servidor.

### ***Interfaz de programación de aplicación (API)***

En el área de desarrollo web una API es un conjunto de métodos, eventos, propiedades usados para recuperar información en formato JSON.

El formato JSON se utiliza para intercambiar información, se caracteriza por su fácil lectura, escritura y tiene la misma sintaxis que un objeto creado en el lenguaje JavaScript. Es común encontrar que la información que se recibe de APIs esta en este formato.

### **Estructura JSON**

En la Figura 28 se muestra la respuesta ante una solicitud de consulta de votos en formato JSON, entre llaves están contenidas 3 propiedades que son: name, valor e id, con sus respectivos valores "ocean", 50, "0AiLuH1Woaqx061B". Esta es una estructura básica en JSON como se puede notar se ha utilizado dos tipos de datos que son strings y números, pero se pueden utilizar otros tipos como arreglos, booleans y otros objetos. La forma de acceder a estos valores es especificando su propiedad.

### **Figura 28**

*Ejemplo estructura JSON*

```
{name: "ocean", valor: 50, _id: "0AiLuH1Woaqx061B"}
```



## APIs utilizadas en el proyecto

### **API Fetch**

Es utilizada para consultar recursos de manera asíncrona. En la Figura 29 se muestra una consulta realizada mediante fetch en este caso la ruta “/naturaleza” definida en el servidor responderá con un string que contiene los votos almacenados en una base de datos. Por este motivo se utiliza el método json() para convertirlos a un objeto JavaScript y extraer los valores correspondientes a la cantidad de votos.

### **Figura 29**

*Ejemplo de consulta mediante API fetch*

```
const recv = await fetch('/naturaleza')
const datos_server = await recv.json()
```

### **API de geolocalización**

Proporciona la ubicación a las aplicaciones web que lo requieren (Mozilla, n.d.), para acceder a la misma se debe hacer una llamada a “navigator.geolocation” que retorna un objeto con su método getCurrentPosition(funcion), al ser llamado necesita que se especifique una función que obtendrá la latitud y longitud .Esta información será necesaria para usar la API de clima.

### **API de clima “open-weather-map”**

Para utilizar esta API se debe crear una cuenta en rapidapi.com. Al ingresar a la página en la barra de búsqueda escribimos “open weather map”, se presentara la interfaz mostrada en la Figura 30, en la parte izquierda incluye campos para ingresar latitud, longitud, entre estos campos cnt debe ser un valor mayor a cero para que funcione correctamente. En la parte derecha se observa el código JavaScript necesario para consultar el clima.

Figura 30

Interfaz de pruebas open-weather-map

The screenshot displays the API testing interface for the open-weather-map service. On the left, under 'Optional Parameters', the following values are set:

- cnt** (NUMBER): 1. Description: To limit number of listed cities please setup cnt parameter.
- mode** (STRING): null. Description: If left blank will default to JSON output. Ability to retrieve data in XML or HTML.
- lon** (NUMBER): 0. Description: Must be used with lat. Get current weather data when you know the longitude of the city.
- type** (STRING): link, accurate. Description: To setup accuracy level please use type parameter that have two values - accurate and like. In case of accurate value you will get results that exactly equivalent to your searching word. In case of like value the

On the right, the 'Code Snippets' tab is active, showing a Node.js Axios request:

```
(Node.js) Axios Copy Code
var axios = require("axios").default;

var options = {
  method: 'GET',
  url: 'https://community-open-weather-map.p.rapidapi.com/find',
  params: {
    q: 'quito',
    cnt: '1',
    mode: 'null',
    lon: '0',
    type: 'link, accurate',
    lat: '0',
    units: 'imperial, metric'
  },
  headers: {
    'x-rapidapi-key': '462d6fb6b6b67c89e54ec8add0p13d906jsn10ed19cc2bdd',
    'x-rapidapi-host': 'community-open-weather-map.p.rapidapi.com'
  }
};

axios.request(options).then(function (response) {
  console.log(response.data);
}).catch(function (error) {
  console.error(error);
});
```

Al dar click en “test Endpoint” en la Figura 31 se muestra la respuesta en formato JSON, las propiedades de interés son “main” que incluye la temperatura en grados kelvin y que será transformada a centígrados, “weather” que incluye una descripción del clima y el nombre del icono que lo representa. Cuando se ha recuperado esta información se enviara una respuesta con estos datos hacia el cliente.

## Figura 31

### Respuesta a solicitud API de clima

```

{
  "main": { 6 items
    "temp": 293.15
    "feels_like": 292.38
    "temp_min": 293.15
    "temp_max": 293.15
    "pressure": 1027
    "humidity": 45
  }
  "dt": 1624900165
  "wind": { ... } 2 items
  "sys": { ... } 1 item
  "rain": { ... } 1 item
  "snow": NULL
  "clouds": { 1 item
    "all": 75
  }
  "weather": [ 1 item
    0 : { 4 items
      "id": 500
      "main": "Rain"
      "description": "light rain"
      "icon": "10d"
    }
  ]
}

```

## Websocket

El protocolo websocket permite intercambiar información entre cliente y el servidor en dos direcciones, además no requiere interrumpir la conexión ni solicitudes http adicionales (Javascript.info, n.d.).

Para crear una conexión se utiliza la siguiente sentencia que emplea el objeto Websocket: `let socket = new WebSocket("ws://localhost:3000")`.

Se puede notar el url ingresado como parámetro inicia con "ws" que representa al protocolo websocket, seguido se especifica la dirección IP y puerto definido en el servidor.

Cuando se ha creado la conexión se debe atender los siguientes eventos:

- **Open:** Iniciado cuando se abre una conexión Websocket.
- **Mensaje:** Se inicia cuando se ha recibido datos.
- **Error:** Activado cuando la conexión se cierra debido a que los datos no pudieron ser enviados.
- **Close:** Inicia cuando la conexión ha sido cerrada.

## JSON-RPC

El estándar ATSC 3.0 define que se debe utilizar el protocolo JSON-RPC 2.0 para el intercambio de información mediante websockets. Este protocolo consiste en ejecutar una subrutina o conjunto de instrucciones en un computador remoto en este caso el servidor, para lo cual se requiere enviar un mensaje en formato JSON con las siguientes propiedades en el caso de una solicitud:

- **Id:** Un número para identificar solicitud y respuesta correspondiente.
- **Method:** Un string con el nombre del método que se desea llamar.
- **Parameters:** Un objeto o arreglo pasado como parámetros al método definido.

Y los siguientes parámetros en el caso de una respuesta:

- **Result:** Los datos correspondientes al método invocado.
- **Error:** Un mensaje en caso que se produzca un error en el método.
- **Id:** El mismo número de la solicitud que se responde.

En la Figura 32 se muestra una respuesta impresa en la consola del navegador ante una solicitud con el método "org.atsc.query.languages" e id 95. Como se puede notar la propiedad id tiene el mismo valor que la solicitud y la propiedad resultado presenta como valor otro objeto JSON con el lenguaje de preferencia.

### Figura 32

*Respuesta ante solicitud empleando protocolo JSON-RPC*

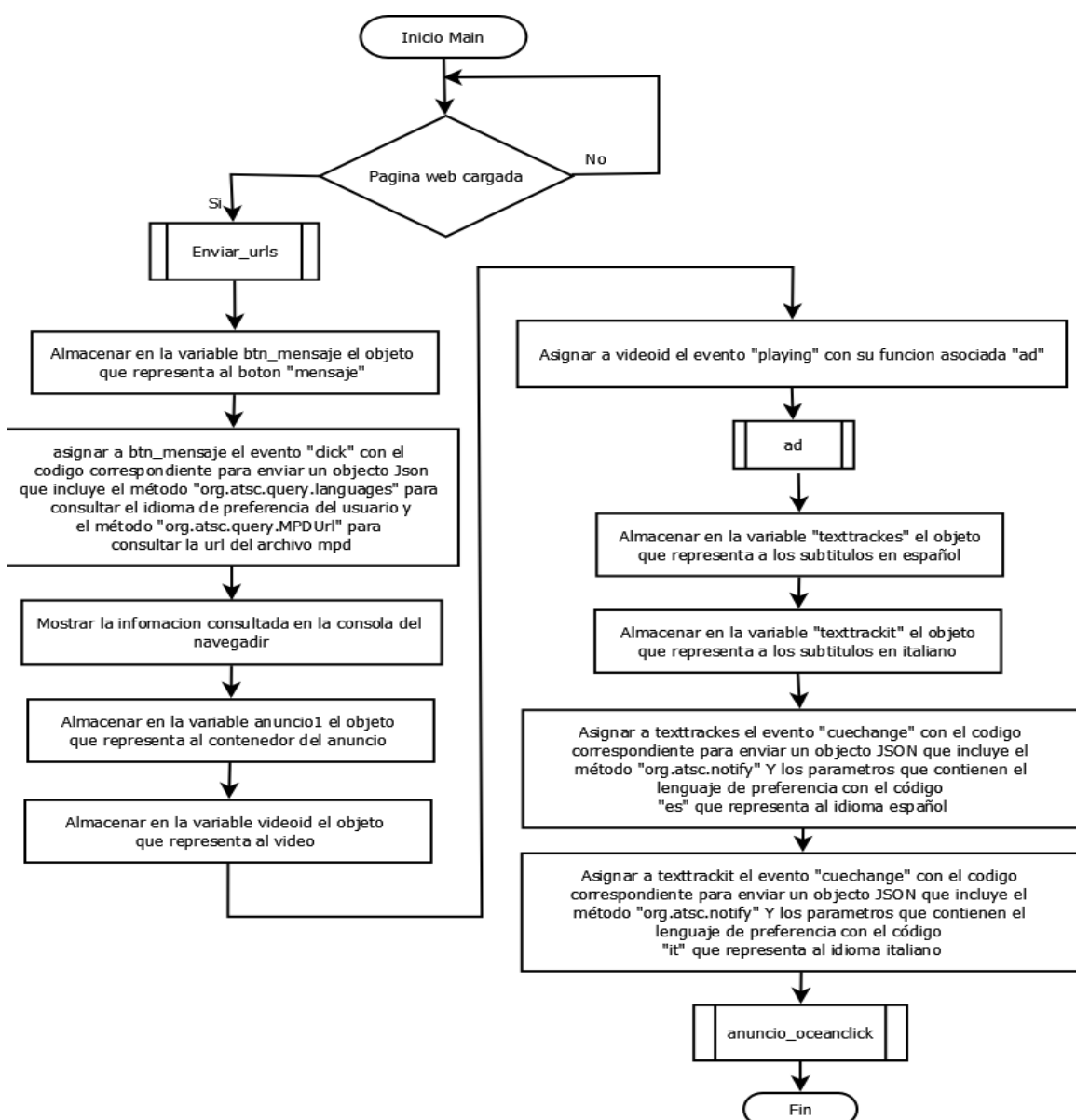
```
▼ {jsonrpc: "2.0", result: {...}, id: 95} ⓘ  
  id: 95  
  jsonrpc: "2.0"  
  ▶ result: {preferredAudioLang: "en", preferredUiLang: "en", preferredCaptionSubtitleLang: "es"}
```

## Algoritmos en JavaScript relacionados con el cliente

La función principal mostrada en la Figura 33 inicia cuando la página ha sido cargada, inicia con la llamada a varias subfunciones que se mostraran posteriormente, además de asignar eventos al botón de enviar mensajes y reaccionar a eventos asociados con la reproducción del video como mostrar subtítulos, mostrar anuncios.

**Figura 33**

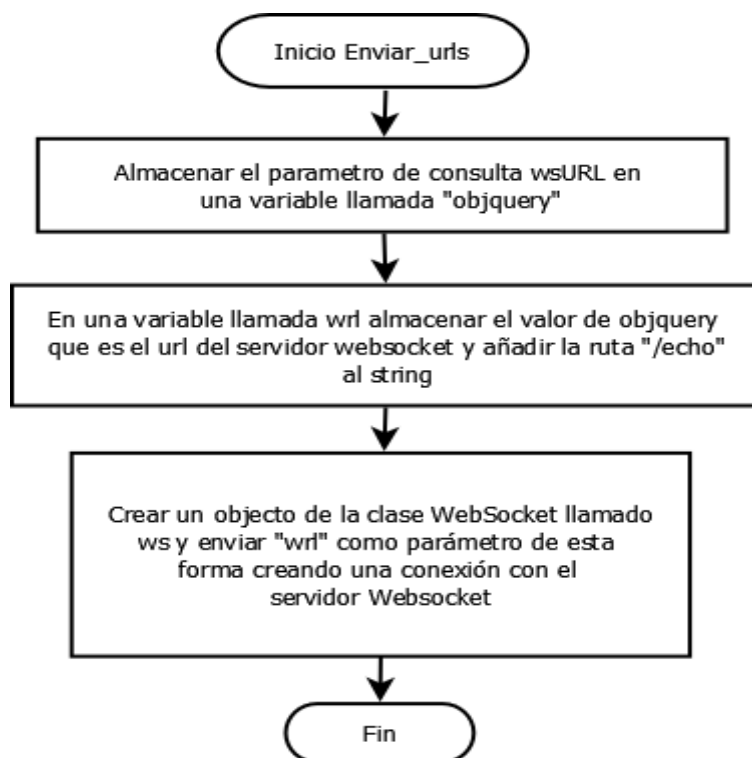
*Diagrama de flujo de la primera parte de la función principal*



La función `enviar_urls` mostrado en la Figura 34, consiste en leer los parámetros y consultas ingresadas en el navegador de esta forma se puede conectar al servidor websocket antes de enviar mensajes usando el protocolo JSON-RPC 2.0

### Figura 34

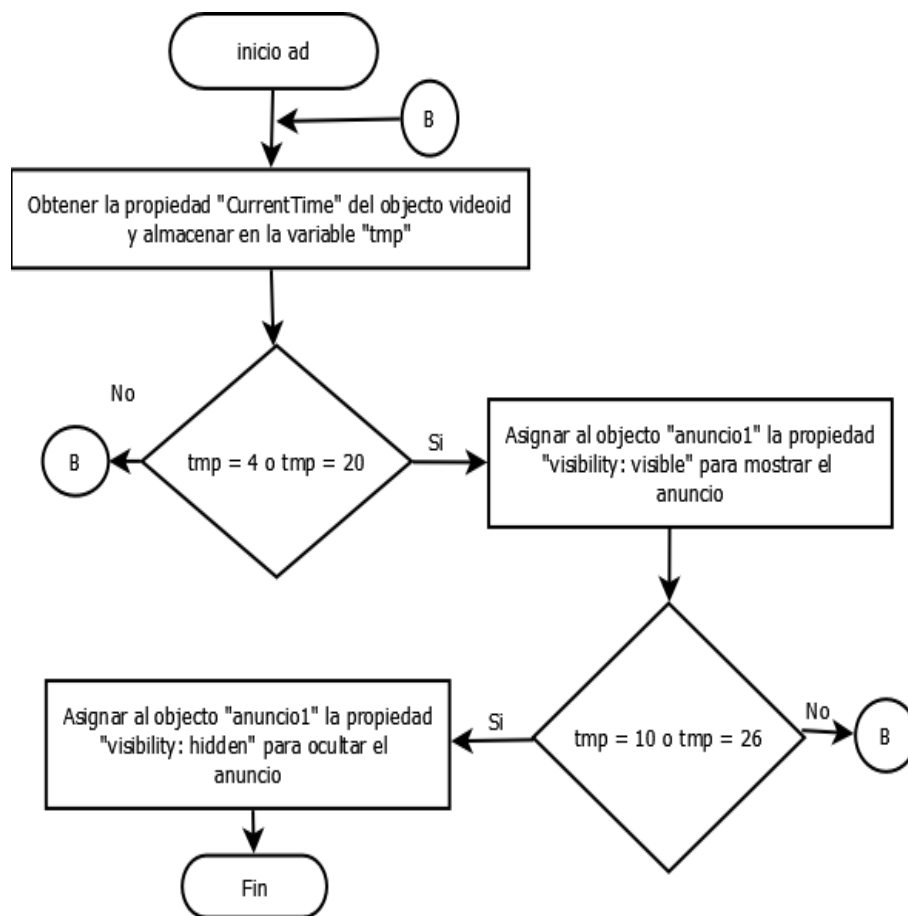
*Diagrama de flujo función `enviar_urls`*



La función `ad` mostrado en la Figura 35, consiste en determinar el tiempo exacto para mostrar un anuncio con 3 opciones de votación, el mismo que se ocultara después de 6 segundos.

Figura 35

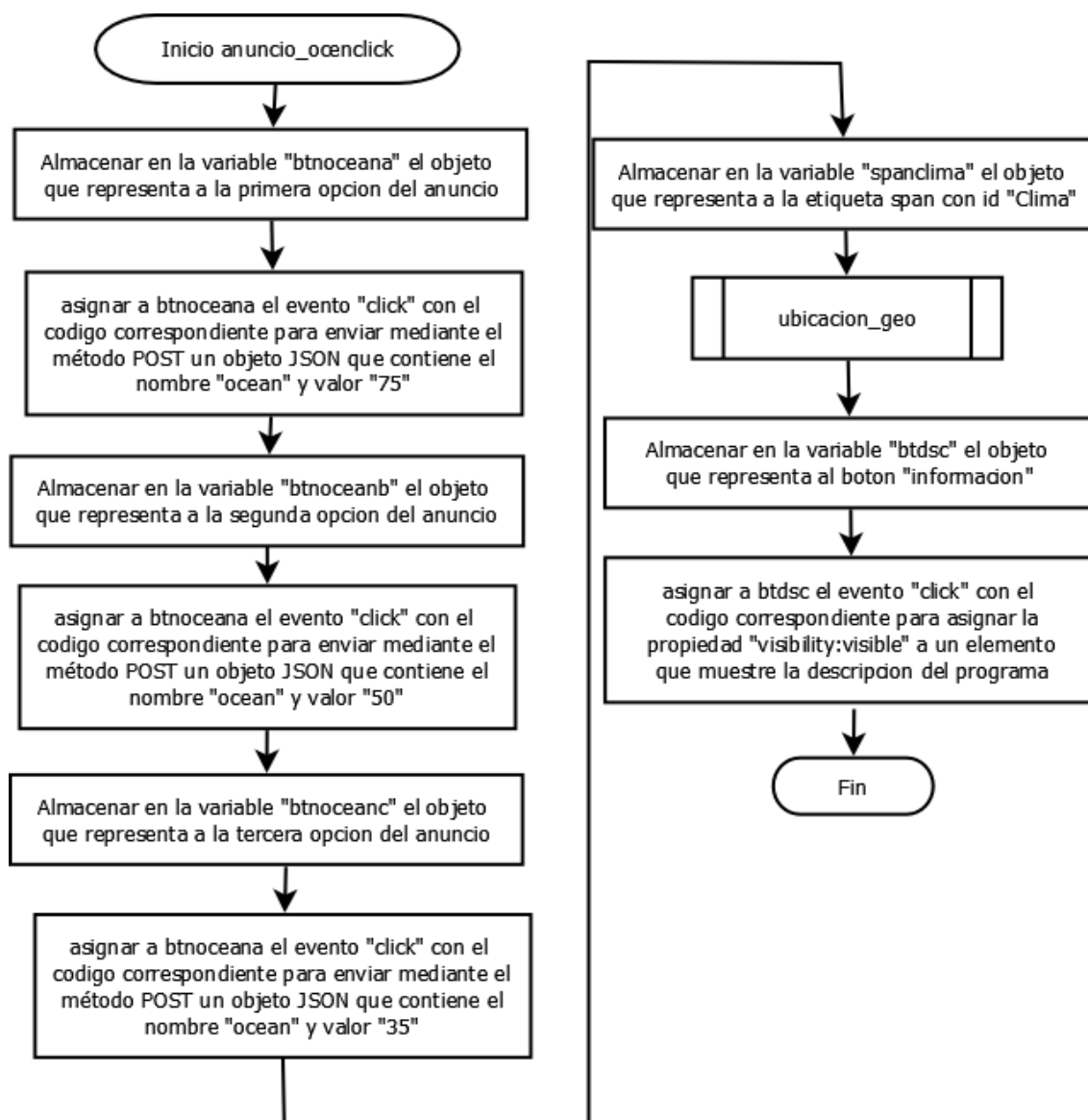
Diagrama de flujo función ad



La función *anuncio\_ocenclck* mostrado en la Figura 36, en su primera etapa consiste en asignar el evento click a los 3 botones para la selección del voto presentado en el anuncio, esta información requiere ser enviada al servidor para que se almacene en una base de datos, también se asigna el evento correspondiente al botón información.

Figura 36

Diagrama de flujo primera parte función anuncio\_ocencklick

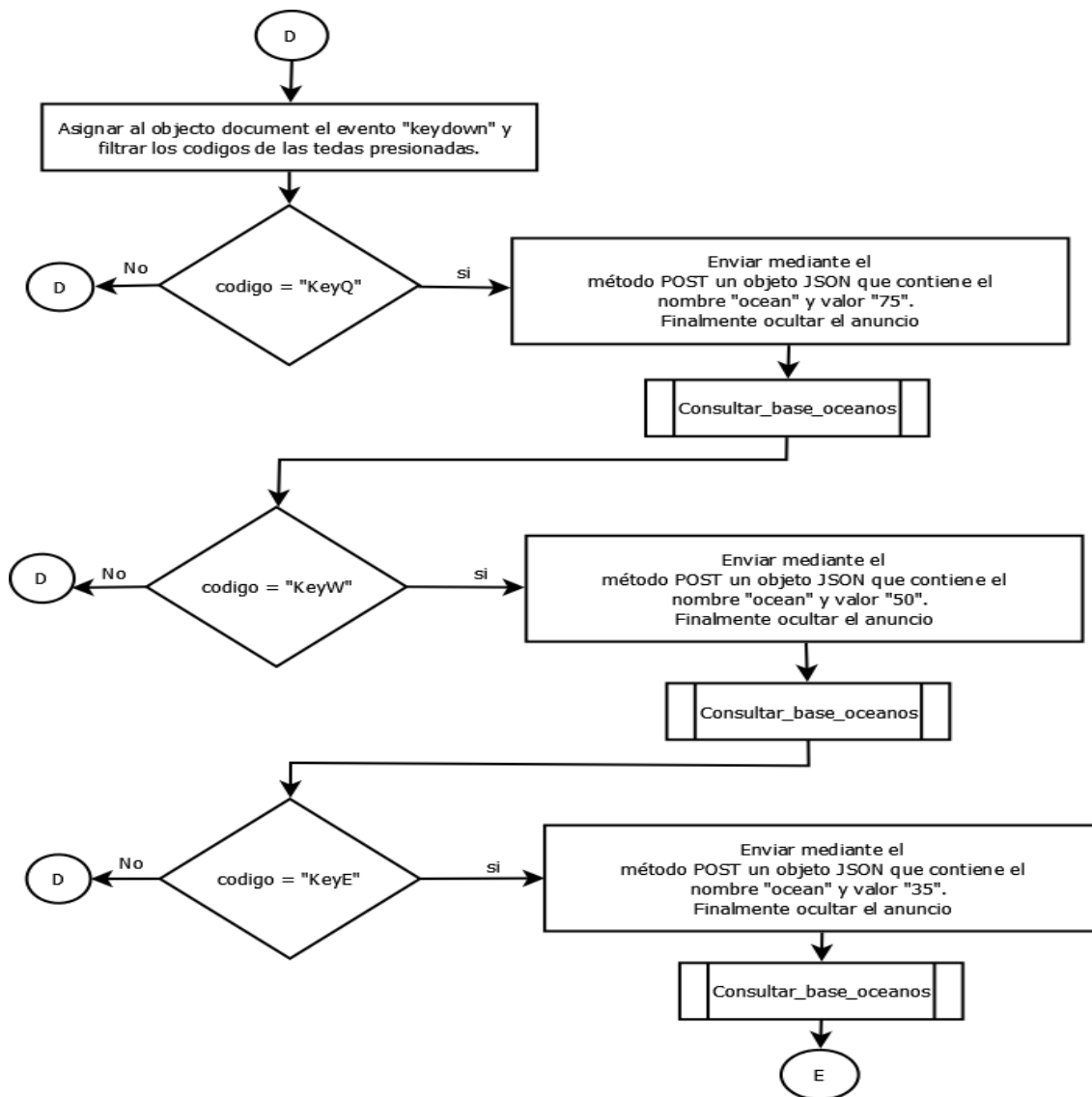


En la segunda parte se asigna el evento "keydown" como se muestra en la Figura 37 y los códigos correspondientes de las teclas que representan las 3 opciones de votación para enviar la opción seleccionar al servidor además de llamar a la función consulta\_base\_oceanos para recuperar la información de la base de datos.



Figura 37

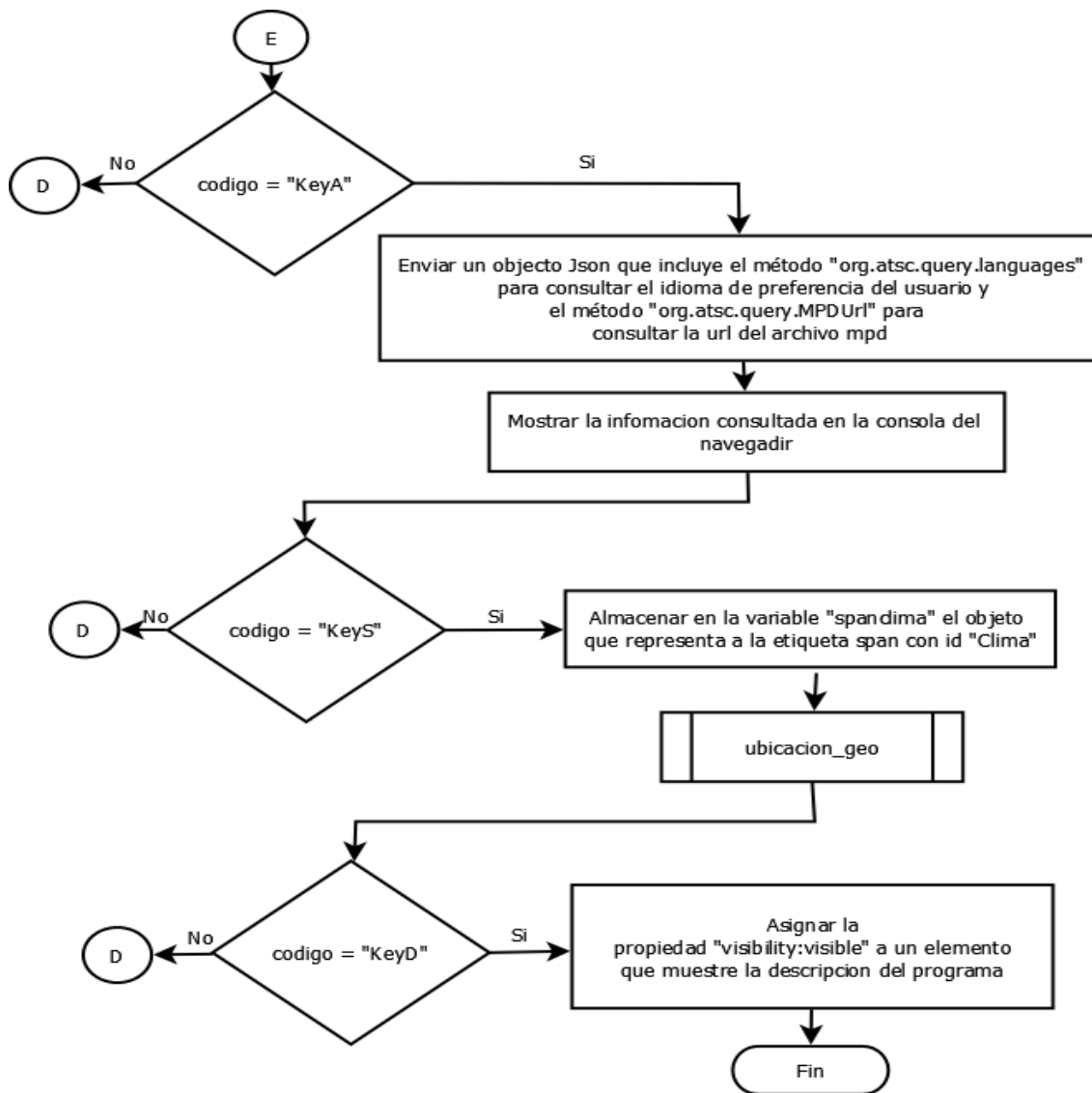
Diagrama de flujo segunda parte función anuncio\_ocenclick



En la tercera parte se asigna el evento “keydown” como se muestra en la Figura 38 y los códigos correspondientes de las teclas que enviarán mensajes mediante el protocolo JSON-RPC, mostrarán el clima y una descripción del programa.

Figura 38

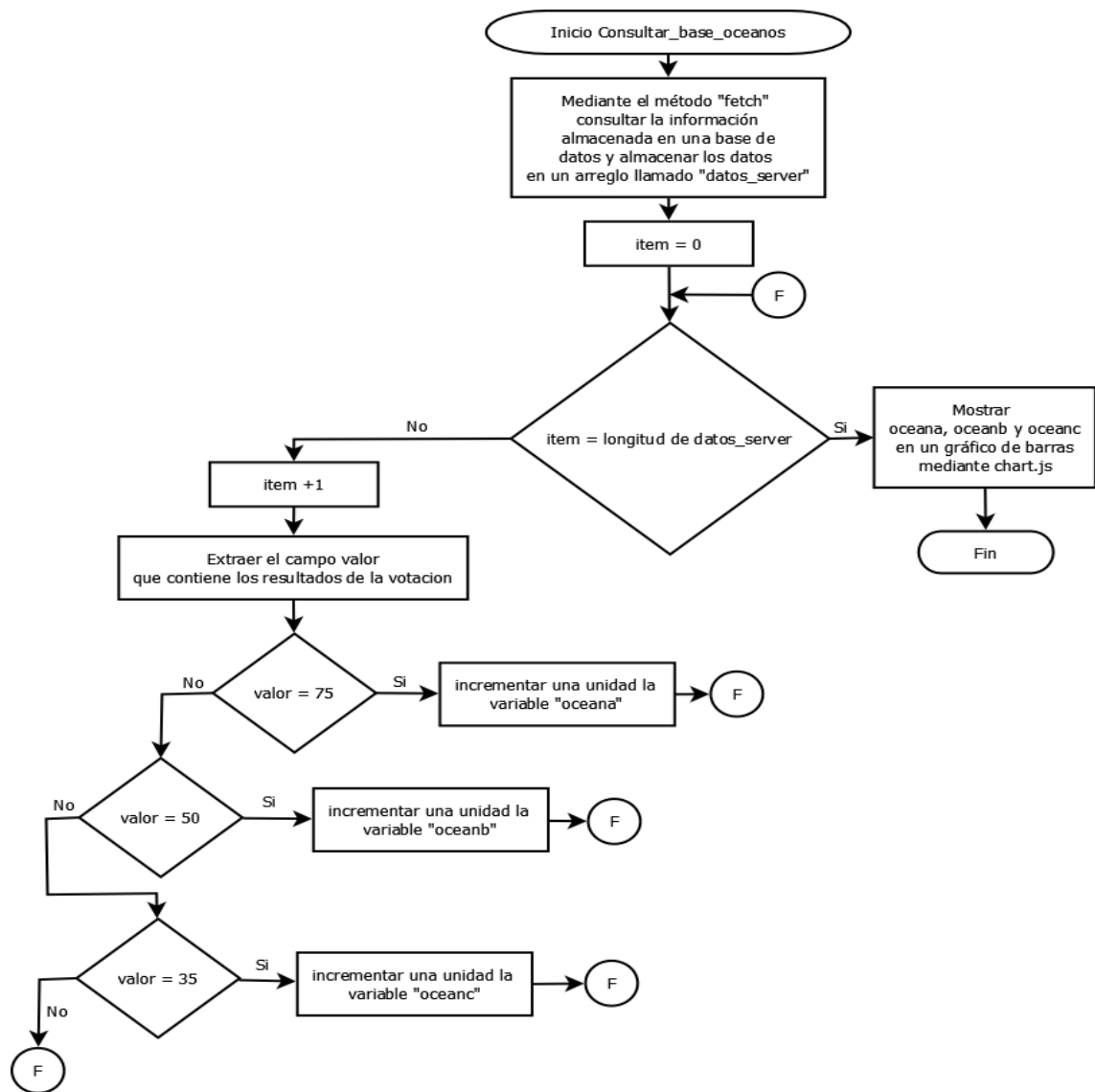
Diagrama de flujo tercera parte función *anuncio\_ocenlick*



La función *consultar\_base\_oceanos* mostrado en la Figura 39 consiste en consultar todos los datos almacenados en la base y mediante contadores determinar cuántos votos de cada opción han sido seleccionados para mostrarlos en un gráfico de barras mediante la librería *chart.js*

Figura 39

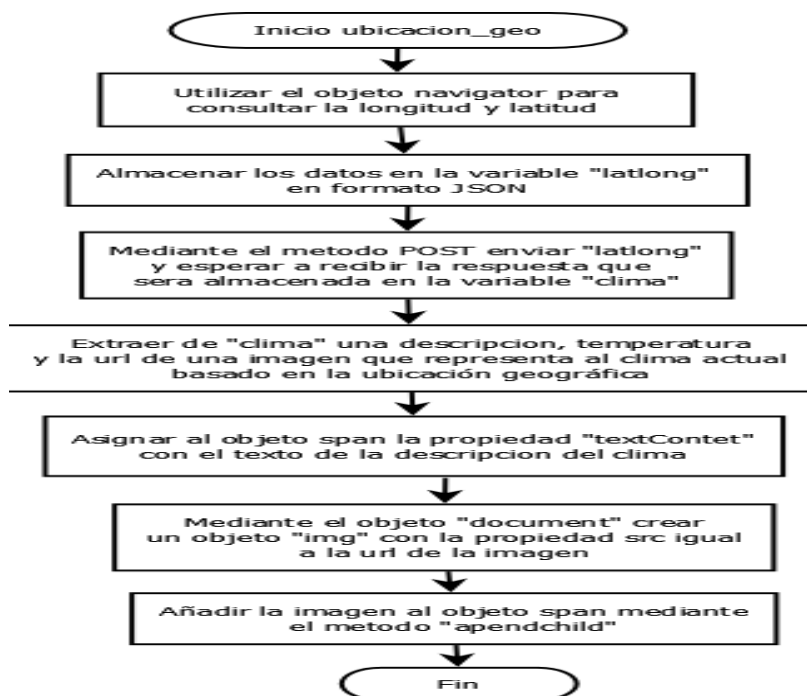
Diagrama de flujo función consultar\_base\_oceanos



La función ubicación\_geo mostrado en la Figura 40 consiste en obtener las coordenadas geográficas del usuario y enviar estos datos al servidor para que este utilice la api "open-weather-map" permitiendo mostrar una descripción del clima, temperatura y un icono en la barra de navegación.

Figura 40

Diagrama de flujo función *Ubicacion\_geo*



### Servidores

En el lado del servidor se utilizó Node.js, es un entorno que utiliza JavaScript en el servidor además de ser multiplataforma. Se caracteriza por usar programación asincrónica es decir es más eficiente ya que elimina los tiempos de espera para atender solicitudes.

El instalador se encuentra disponible en <https://nodejs.org/en/download/>, es recomendable descargar la versión LTS ya que proporciona mayor tiempo de soporte.

### Módulos

Los módulos son librerías de JavaScript que se pueden incluir en un proyecto, algunas vienen instaladas por defecto y otras requieren ser descargadas, en este caso mediante un administrador de paquetes llamado npm, los paquetes contienen los archivos necesarios para que funcione un módulo (w3schools, s.f.).

Previamente a instalar los paquetes se debe ejecutar el comando: `npm init` en la terminal y llenar los datos solicitados como se muestra en la Figura 41, la mayor parte de datos son creados por defecto, al finalizar se debe escribir “yes” para confirmar que los datos ingresados son los correctos. En esta etapa automáticamente se crea el archivo `package.json` el cual contiene una lista de los paquetes que han sido instalados.

### Figura 41

*Datos solicitados al ejecutar `npm init`*

```
package name: (victoratsc)
version: (1.0.0)
description: servidor atsc
git repository:
keywords:
author: Victor Chitupanta
license: (ISC)
```

Luego se debe instalar los siguientes paquetes:

1. **Express:** Sera utilizado para crear un servidor http que atienda a solicitudes GET y POST, comando de instalación: `npm install express`.
2. **Express-ws:** Se utiliza para crear un servidor websocket en una ruta definida, comando de instalación: `npm install express-ws`.
3. **Nedb:** Es una base de datos embebida que está escrita en Javascript, comando de instalación: `npm install nedb`.

Para incluir los módulos en el archivo `server.js` se usa la función `require()` y se asigna el valor retornado a una variable para que puedan ser accedido en la aplicación.

En la Figura 42 se puede confirmar bajo la propiedad “dependencies” que se han instalado las versiones más recientes de estos paquetes.

## Figura 42

Paquetes instalados mostrados en el archivo package.json

```

"name": "atsc_aplicacion",
"version": "1.0.0",
"description": "interactividad mediante tecnologias definidas en el estandar atsc 3.0",
"main": "server.js",
"dependencies": {
  "express": "^4.17.1",
  "express-ws": "^5.0.2",
  "nedb": "^1.8.0",
  "pkg": "^5.3.1",
  "unirest": "^0.6.0"
},
"devDependencies": {},
"scripts": {
  "test": "echo \\\"Error: no test specified\\\" && exit 1",
  "start": "node server.js"
},
"keywords": [
  "atsc3.0"
],
"author": "victor chitupanta palomo",
"license": "ISC"
}

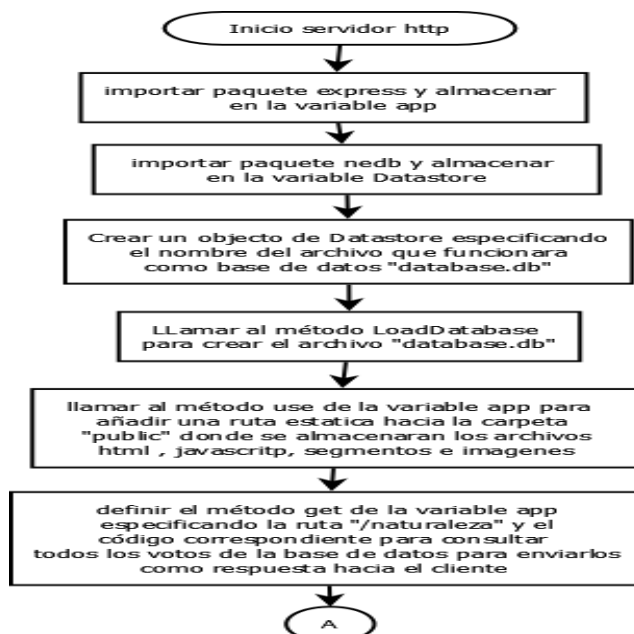
```

## Algoritmos en JavaScript relacionados con el servidor

En la Figura 43 se detallan los módulos utilizados para crear un servidor http que provea recursos al cliente y atienda a solicitudes que empleen el método GET para consultar información de la base de datos.

## Figura 43

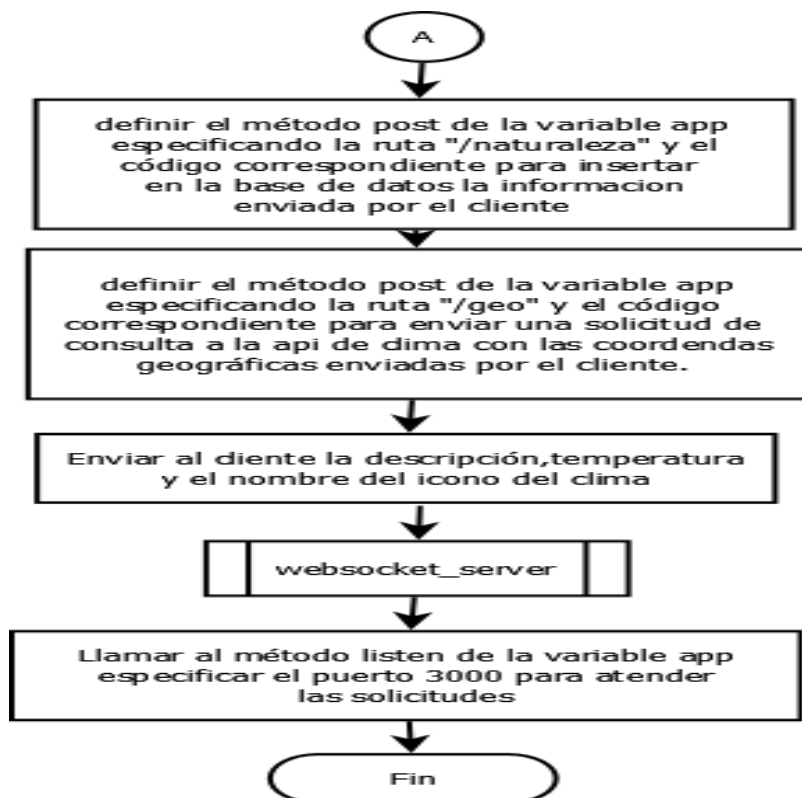
Diagrama de flujo acerca del funcionamiento del servidor http parte 1



En la Figura 44 se muestran la definición de métodos POST con sus respectivas rutas para insertar datos en la base y consultar detalles del clima, en esta etapa se especifica el puerto para responder ante solicitudes http.

#### Figura 44

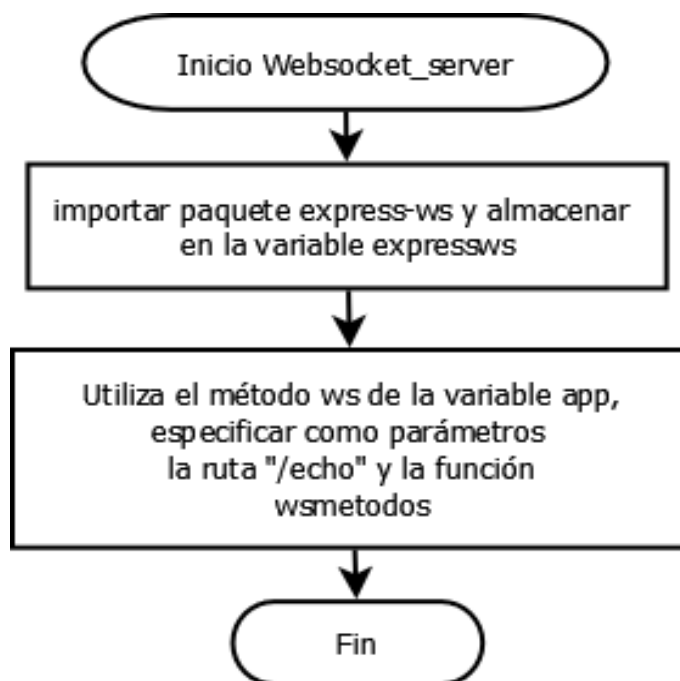
Diagrama de flujo acerca del funcionamiento del servidor http parte2



En la Figura 45 se menciona los paquetes importados para que la aplicación funcione como servidor websocket, en este caso es necesario especificar la ruta "/echo" a la cual se realiza la conexión.

**Figura 45**

*Diagrama de flujo sobre el servidor websocket*

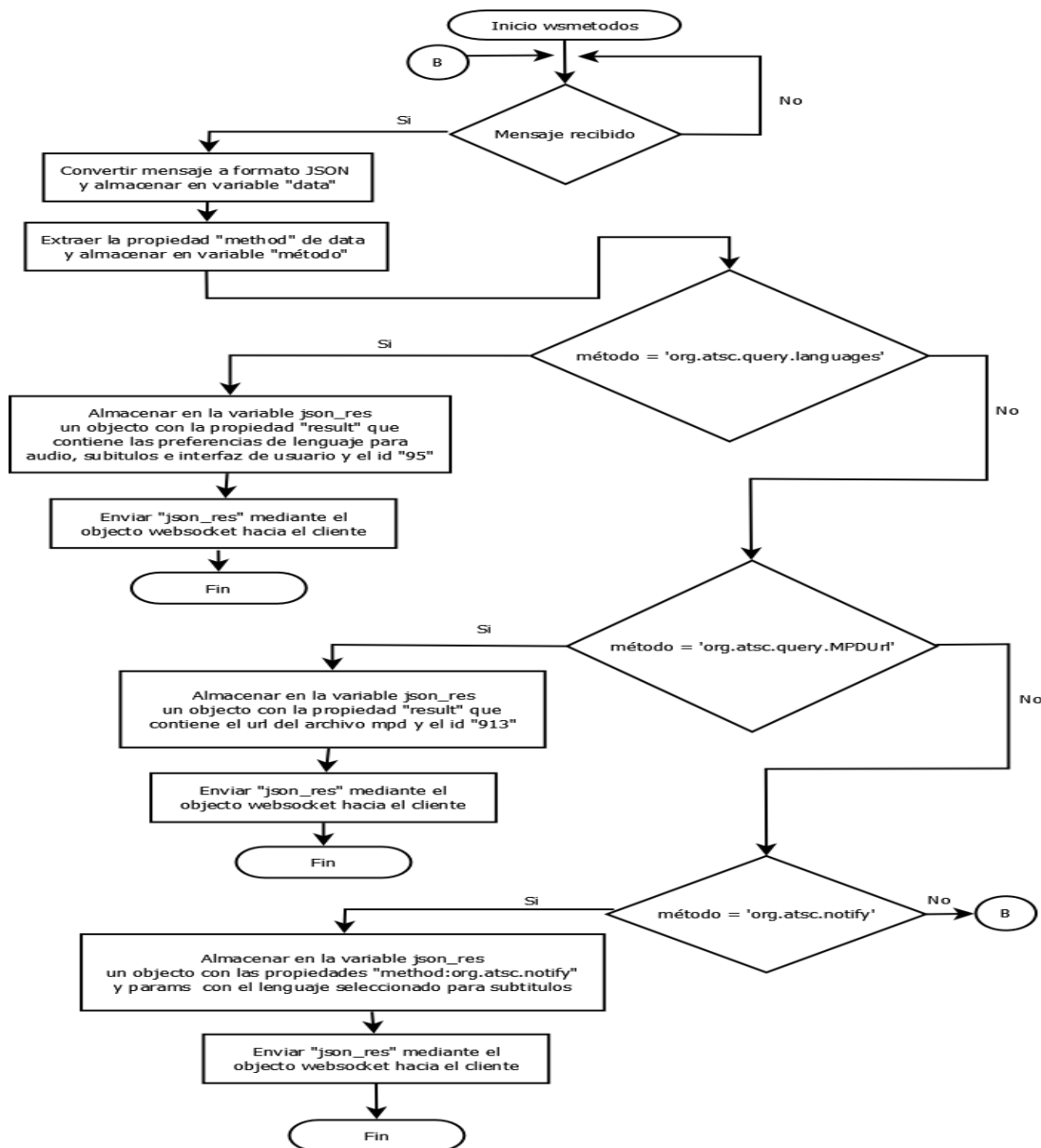


En la Figura 46 se muestra la función wsmetodos que permite enviar mensajes empleando el protocolo JSON-RCP ante 3 tipos de métodos enviados por el cliente.



Figura 46

Diagrama de flujo de función wsmetodos encargada de enviar mensajes utilizando JSON-RPC.



## Capítulo IV

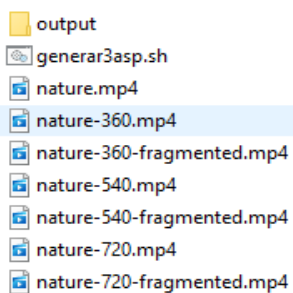
### Análisis de Resultados

#### Creación del contenido MPEG-DASH

A partir del ejecutable de Benton4 se crearon 3 versiones del archivo video nature.mp4 en las calidades de (360, 540, 720) con sus respectivas fragmentaciones como se muestra en la Figura 47.

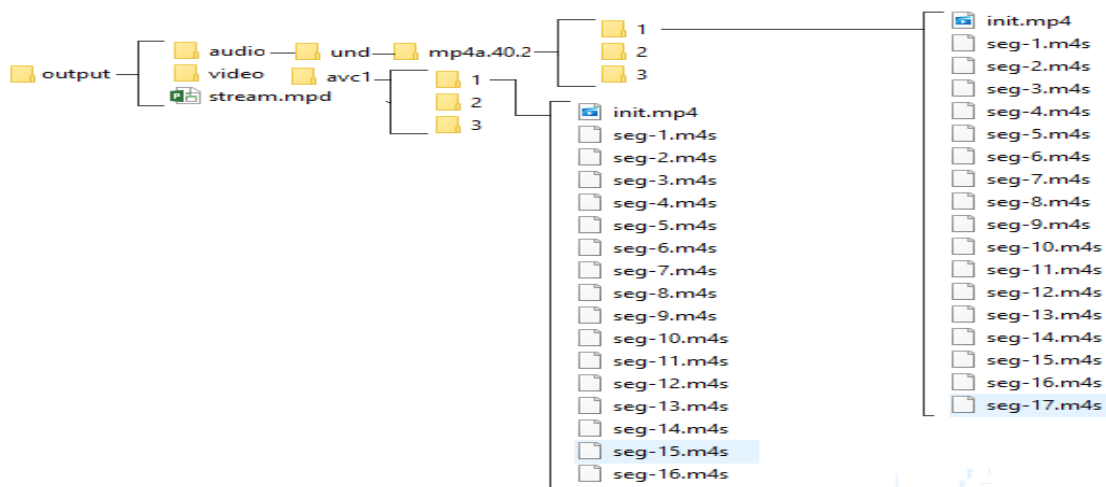
#### Figura 47

*Creación de las tres versiones del archivo con sus respectivas fragmentaciones*



#### Figura 48

*Carpeta output y subcarpetas de los segmentos de audio/video y el archivo stream.mpd*



Al empaquetar las 3 versiones fragmentadas en una sola presentación se crea una carpeta output como se muestra en la Figura 48, el cual contiene las carpetas de audio/video cada una con subcarpetas que contienen varios segmentos correspondientes a las 3 versiones creadas, siendo la carpeta 1 la de mejor calidad, además en la carpeta output se crea el archivo streams.mpd.

### Funcionamiento de la aplicación

Al finalizar la implementación de la aplicación, el resultado final de la página cargada en el navegador se puede observar en la Figura 49, con las 3 secciones principales que son: la barra de navegación, sección de visualización y la sección de descripción.

### Figura 49

*Resultado de la página web cargada en el navegador*



Al cargar la página podemos verificar en la consola del navegador si la conexión del servidor websocket fue correcto o no, como se muestra en la Figura 50 la respuesta impresa en la consola el url del servidor websocket inicia con "ws" que representa al protocolo websocket, seguido de la dirección IP y puerto definido en el servidor, cuando

este mensaje se imprime significa que la conexión fue exitosa, además se imprime una notificación que incluye como parámetro el idioma de preferencia seleccionado para subtítulos que en este caso es español.

## Figura 50

*Respuesta de la conexión del servidor websocket y la preferencia de subtítulo*

```
ws://localhost:3000/echo
▼ {jsonrpc: "2.0", method: "org.atsc.notify", params: {...}} ⓘ
  jsonrpc: "2.0"
  method: "org.atsc.notify"
  ▶ params: {msgType: "langPref", preferredCaptionSubtitleLang: "es"}
```

## Interactividad

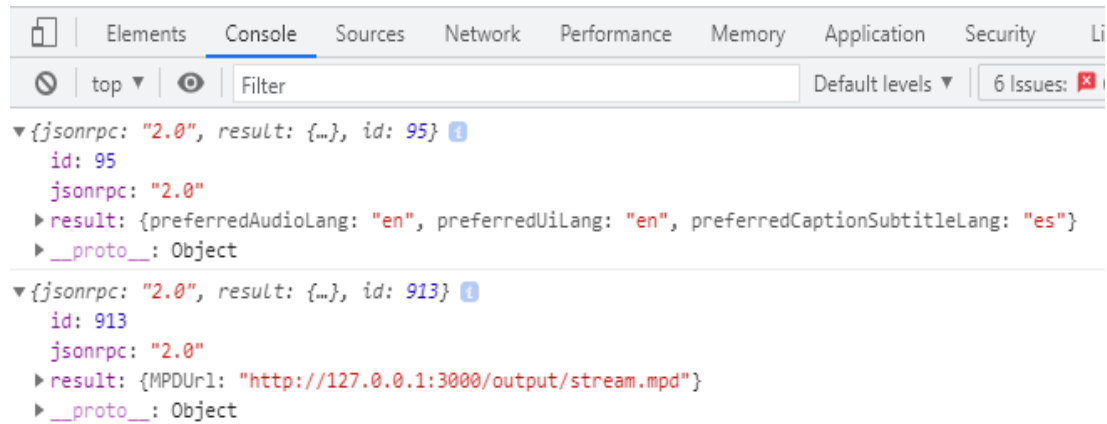
### Teclado del control remoto

En la barra de navegación se presenta las teclas con sus respectivos colores para el control remoto los cuales reaccionaran a los siguientes eventos:

- **Botón A o de Mensaje:** Al presionar la tecla A, se muestra 2 respuestas impresas en la consola del navegador como se muestra en la Figura 51, la primera corresponde a la respuesta ante una solicitud de consulta, el método especificado es "org.atsc.query.languages" e id 95, el cual presenta los parámetros de preferencia de idioma para la pista de audio y subtítulos. La segunda corresponde a la respuesta ante una solicitud de consulta, el método especificado es "org.atsc.query.MPDUrI" e id 913, el cual presenta la url del archivo mpd.

## Figura 51

*Respuestas al presionar la tecla A, consultar idioma y consultar MPDUrl*



- **Botón S o de Clima:** Al presionar la tecla S se obtiene la temperatura y clima en base a la ubicación geográfica del usuario, el cual se despliega en la parte superior derecha de la barra de navegación como se muestra en la Figura 52.

## Figura 52

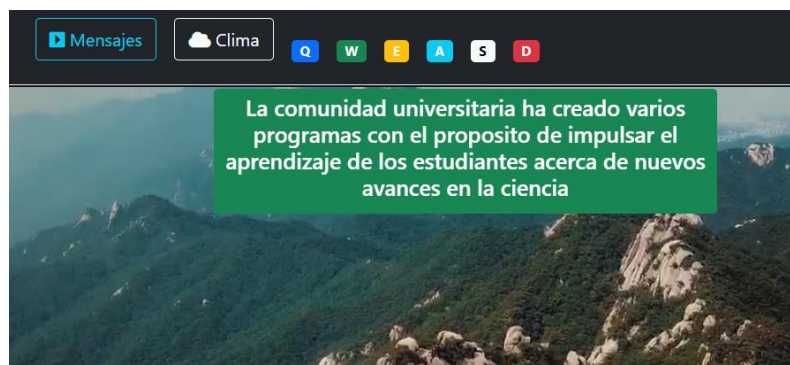
*Respuesta de la temperatura y Clima al presionar la tecla S*



- **Botón D:** Al presionar la tecla D se despliega una información el cual se encuentra en una capa superior al del video es decir un zindex 2 como se muestra en la Figura 53.

### Figura 53

Información al presionar la tecla D



- **Botones Q, W, E:** Cuando el video se encuentra en reproducción en el tiempo 4s se despliega un anuncio con 3 opciones de votos el mismo que se ocultara después de 6s, se encuentra en una capa superior al del video es decir un zindex 2. Al presionar las teclas Q, W, E se enviaran los datos de 75%, 50%, 35% respectivamente a la base de datos e ira actualizándose en el gráfico de barras que se encuentra en la parte inferior derecha de la pantalla como se muestra en la Figura 54.

### Figura 54

Gráfico de barra según los votos al presionar las teclas Q, W, E

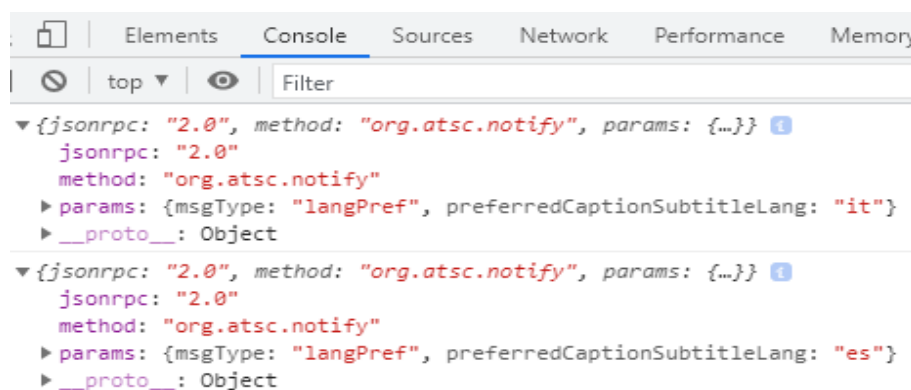


## Cambio del texto de los subtítulos

Cuando se cambie el texto de los subtítulos, se imprimirá en la consola del navegador una notificación de respuesta a la solicitud con el método "org.atsc.query.notify", los parámetros de respuesta incluyen el idioma para el subtítulo seleccionado. Puede ser en italiano o español como se muestra en la Figura 55.

**Figura 55**

*Notificación del cambio del texto de los subtítulos*



## Adaptación de los segmentos según en ancho de banda de la red

A continuación, se presenta tres escenarios en los cuales los segmentos generados se adaptarán según el ancho de banda de la red. En esta etapa se utilizó la opción "Network throttling" disponible en la opción inspeccionar del navegador Google Chrome, esta opción permitirá emular diferentes velocidades en la red y así analizar el resultado del algoritmo de selección adaptativo proporcionado en la librería dash.js.

**Figura 56**

*Ancho de banda con su respectiva escala de video extraído del archivo stream.mpd*

```

" height="720" scanType="progressive" frameRate="30000/1001" bandwidth="1558037"/>
height="540" scanType="progressive" frameRate="30000/1001" bandwidth="833509"/>
height="360" scanType="progressive" frameRate="30000/1001" bandwidth="419985"/>
  
```

Como se muestra en la Figura 56 para cada escala de video existe un determinado ancho de banda correspondiente por lo que esta información nos permitirá crear perfiles personalizados dentro “Network throttling” para emular 3 condiciones diferentes en la red, la información que se requiere ingresar es la velocidad de descarga como se muestra en la Figura 57.

### Figura 57

*Perfiles creados con diferentes velocidades de descarga*

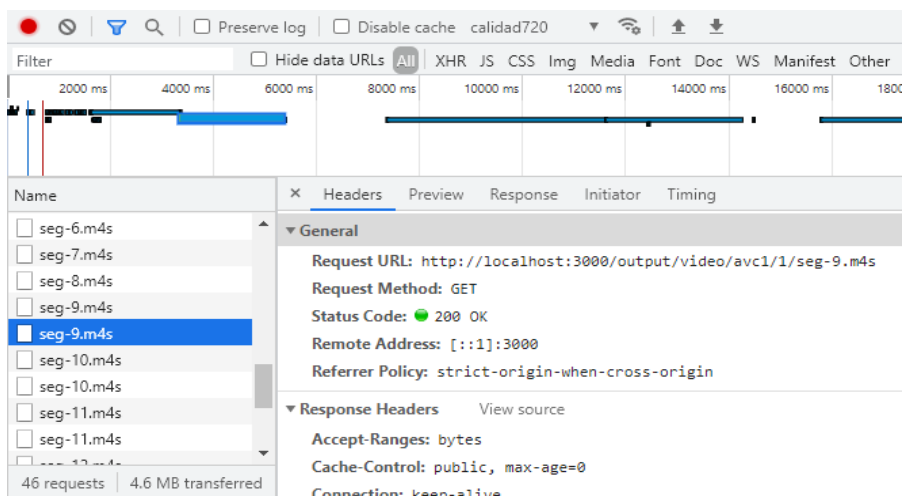
calidad720	1.5 Mbit/s
calidad540	800 kbit/s
calidad360	400 kbit/s

### Primer escenario

En la Figura 58 se muestra en la cabecera de la solicitud el segmento mostrado se encuentra en la carpeta “1” que contiene archivos con la mejor calidad correspondiente al aspecto de 720.

### Figura 58

*URL de los segmentos descargados con ancho de banda de 1.5Mbits/s*



The screenshot shows a browser's developer tools interface. At the top, there's a network waterfall chart with a time scale from 0 to 1800 ms. Below it, a list of network requests is shown, with 'seg-9.m4s' selected. The 'Headers' tab is active, displaying the following information:

- Request URL:** http://localhost:3000/output/video/avc1/1/seg-9.m4s
- Request Method:** GET
- Status Code:** 200 OK
- Remote Address:** [::1]:3000
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
  - Accept-Ranges: bytes
  - Cache-Control: public, max-age=0
  - Connection: keep-alive

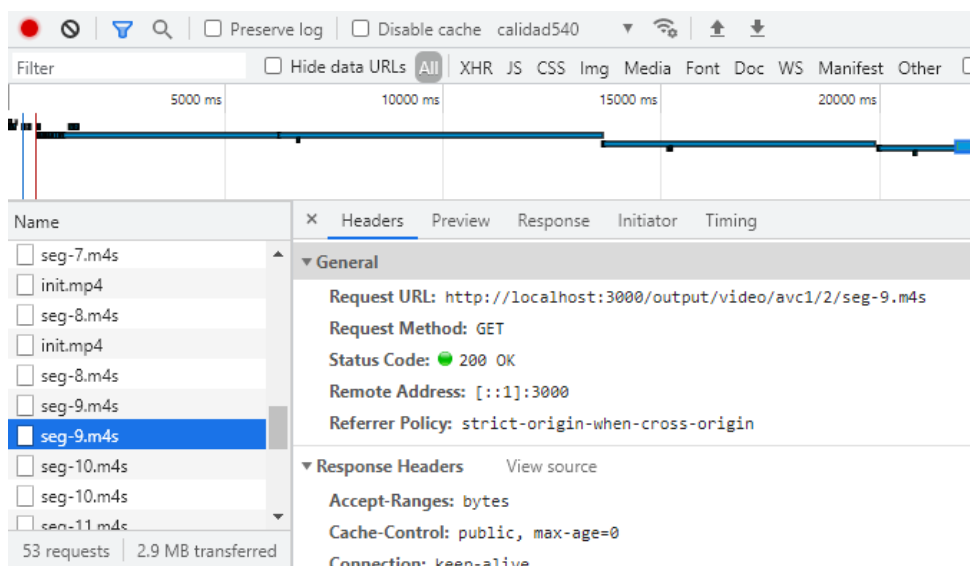


### Segundo escenario

En la Figura 59 se muestra en la cabecera de la solicitud el segmento mostrado se encuentra en la carpeta “2” que contiene archivos con calidad media correspondiente al aspecto de 540.

### Figura 59

*URL de los segmentos descargados con ancho de banda de 800 kbits/s*

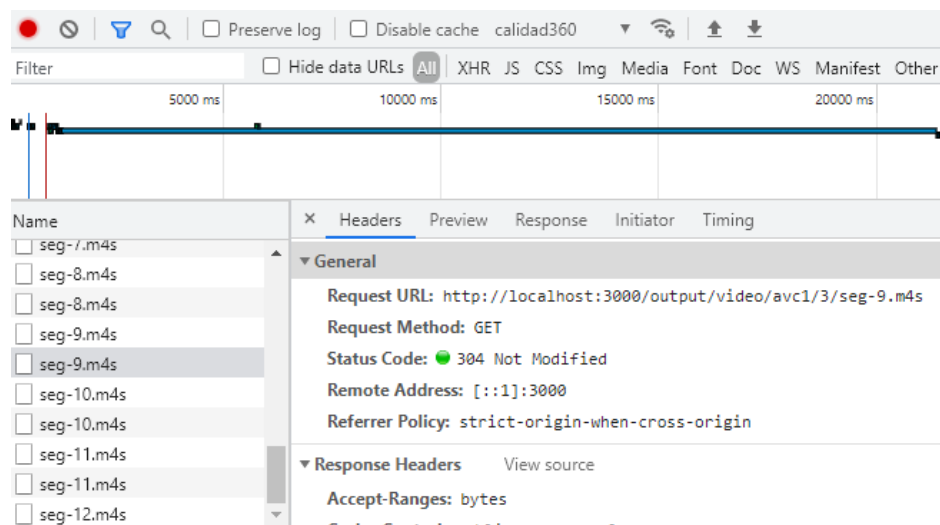


### Tercer escenario

En la Figura 60 se muestra en la cabecera de la solicitud el segmento mostrado se encuentra en la carpeta “3” que contiene archivos con calidad baja correspondiente al aspecto de 360.

**Figura 60**

*URL de los segmentos descargados un ancho de banda de 400 kbits/s*



**Análisis de tráfico con la herramienta wireshark, para ver el comportamiento de la información a transmitir.**

Se realizó una captura de tráfico para analizar cómo se encuentra encapsulada la información a transmitir, los paquetes capturados pueden contener: Archivos MPD, Segmentos dash, archivos html, JavaScript, imágenes, entre otros. En este caso se analizará un paquete que contiene un archivo MPD.

**Figura 61**

*Parámetros de los campos en las capas*

```

Ethernet II, Src: ChiconyE_84:5a                               Dst: MurataMa_9f:bf:ed
Internet Protocol Version 4, Src: 192.168.100.47, Dst: 192.168.100.2
Transmission Control Protocol, Src Port: 3000, Dst Port: 44210, Seq: 3341, Ack: 838, Len: 946
[2 Reassembled TCP Segments (2406 bytes): #863(1460), #864(946)]
Hypertext Transfer Protocol
Media Type
Media type: application/dash+xml (2093 bytes)

```

En la Figura 61 se muestra un paquete enviado por el servidor ante una solicitud para obtener el archivo MPD, con detalles de los campos en las capas de: Aplicación,

Transporte, red y enlace, considerando que las capas superiores encapsulan a las de nivel inferior, de esta manera se añade información importante antes de transmitir la aplicación.

En la Figura 62 se muestra el contenido del archivo MPD, cuyo formato es xml, se puede notar que existe un solo periodo y dentro de este 2 adaptationSet:

- El adaptationset de video tiene 3 representaciones, cada una tiene una resolución asociada a un ancho de banda y un identificador para agrupar segmentos de video en diferentes rutas.
- El adaptationset de audio tiene 3 representaciones, cada una tiene una frecuencia de muestreo asociada a un ancho de banda y un identificador para agrupar segmentos de audio en diferentes rutas.

## Figura 62

### Contenido del archivo MPD

```
<?xml version="1.0" ?>
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" profiles="urn:mpeg:dash:profile:isoff-live:2011" minBufferTime="PT3.995" mediaPresentationDuration="PT1M2.1295" type="static">
  <!-- Created with Bento4 mp4-dash.py, VERSION=2.0.0-638 -->
  <Period>
    <!-- Video -->
    <AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1" maxWidth="1280" maxHeight="720">
      <SegmentTemplate timescale="1000" duration="3992" initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/seg-$Number$.m4s" startNumber="1"/>
      <Representation id="video/avc1/1" codecs="avc1.64001F" width="1280" height="720" scanType="progressive" frameRate="30000/1001" bandwidth="1558037"/>
      <Representation id="video/avc1/2" codecs="avc1.64001F" width="960" height="540" scanType="progressive" frameRate="30000/1001" bandwidth="833509"/>
      <Representation id="video/avc1/3" codecs="avc1.64001E" width="640" height="360" scanType="progressive" frameRate="30000/1001" bandwidth="419985"/>
    </AdaptationSet>
    <!-- Audio -->
    <AdaptationSet mimeType="audio/mp4" startWithSAP="1" segmentAlignment="true">
      <SegmentTemplate timescale="1000" duration="3992" initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/seg-$Number$.m4s" startNumber="1"/>
      <Representation id="audio/und/mp4a.40.2/1" codecs="mp4a.40.2" bandwidth="256178" audioSamplingRate="48000">
        <AudioChannelConfiguration schemeIdUri="urn:mpeg:mpegB:cicp:ChannelConfiguration" value="2"/>
      </Representation>
      <Representation id="audio/und/mp4a.40.2/2" codecs="mp4a.40.2" bandwidth="139646" audioSamplingRate="44100">
        <AudioChannelConfiguration schemeIdUri="urn:mpeg:mpegB:cicp:ChannelConfiguration" value="2"/>
      </Representation>
      <Representation id="audio/und/mp4a.40.2/3" codecs="mp4a.40.2" bandwidth="64559" audioSamplingRate="22050">
        <AudioChannelConfiguration schemeIdUri="urn:mpeg:mpegB:cicp:ChannelConfiguration" value="2"/>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

En la Figura 63 se muestra la cabecera de la capa de aplicación el cual incluye campos relacionados con el protocolo http.

## Figura 63

### *Cabecera de la capa de aplicación*

```

Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
      X-Powered-By: Express\r\n
      Accept-Ranges: bytes\r\n
      Cache-Control: public, max-age=0\r\n
      Last-Modified: Thu, 01 Jul 2021 20:21:21 GMT\r\n
      ETag: W/"82d-17a63bba9f5"\r\n
      Content-Type: application/dash+xml\r\n
    > Content-Length: 2093\r\n
  
```

- **Response version:** Corresponde a la versión del protocolo http en este caso es la 1.1.
- **Status Code:** El código 200 indica que el recurso solicitado fue encontrado.
- **Content-type:** Indica el tipo el formato del archivo MPD en este caso dash+xml.
- **Content Length:** Indica el tamaño en bytes del archivo en este caso es de 2093.

La capa de transporte encapsulara la respuesta http que está en la capa de aplicación. En este caso el protocolo utilizado es TCP que garantiza que los paquetes lleguen al destino, en este nivel se agrega la cabecera TCP, algunos de los campos incluidos en la misma son:

```

Transmission Control Protocol, Src Port: 3000, Dst Port: 44210, Seq: 3341, Ack: 838, Len: 946
  
```

- **Seq:** Es el número de secuencia que corresponde al número de bytes enviados.
- **Ack:** Este campo confirmara que se recibió un segmento de datos y en caso de no recibirse después de que expire un temporizador se retransmiten los segmentos.

- **Src Port:** Corresponde al puerto de origen, el puerto 3000 fue configurado en el servidor http para que responda ante cualquier solicitud.
- **Dst Port:** Corresponde al puerto de destino 44210 está asociado a la aplicación web.

La capa de red encapsulara al segmento TCP que está en la capa de transporte, en esta capa se agrega la cabecera IP que incluye información acerca de la versión IP utilizada en este caso versión 4 y las direcciones IP de origen de un teléfono móvil y la IP de destino de una computadora que funciona como servidor http.

```
Internet Protocol Version 4, Src: 192.168.100.47, Dst: 192.168.100.2
```

La capa de enlace encapsula al paquete IP y es utilizada para transmitir datos entre nodos adyacentes o una red local, en esta capa se envían tramas que contiene campos que permiten detectar errores, en la Figura 61 se puede observar que se especifica la dirección mac de origen que coincide con la del servidor y la mac de destino que coincide con el dispositivo móvil.

De esta manera se puede transmitir la información a nivel de capa física por cualquier medio.

## Capítulo V

### Conclusiones y recomendaciones

#### Conclusiones

Se evaluó la estructura para la interactividad con la creación de una aplicación compuesta de archivos multimedia, HTML5, JavaScript y CSS. En el archivo de HTML se definió secciones de navegación, visualización y descripción que fueron distribuidas con el sistema de mallas de bootstrap, en el archivo de JavaScript se definió subrutinas que se ejecutan al iniciarse eventos como presionar una tecla, cambio de subtítulos entre otros.

En el estudio del arte en la norma A/331 se obtuvo información de los modos de transmisión que comprenden los protocolos ROUTE/DASH, MMTP/MPU para la entrega de contenido y servicios en radiodifusión y para banda ancha a través del protocolo HTTP.

Con ayuda de la herramienta de Wireshark se capturo varios paquetes y se pudo verificar que se utilizó los protocolos: HTTP, TCP e IP los cuales están relacionados con la transmisión de contenido en banda ancha, los archivos de textos y multimedia están contenidos en la carga útil de la capa de aplicación. También se notó que la capa de red contiene información acerca de las direcciones IP del cliente y servidor, la capa de transporte presenta información acerca de los puertos de origen o destino relacionados con la aplicación. En el caso de una transmisión en broadcast de acuerdo al estándar se presentan cambios en los protocolos utilizados por ejemplo se utiliza UDP en la capa de transporte y para reemplazar a HTTP se utiliza ROUTE o MMTP.

ATSC 3.0 indica que el protocolo JSON-RPC 2.0 debe ser usado en el intercambio de mensajes mediante websockets, el mismo fue utilizado para enviar un string en formato JSON con un id y método definido en el estándar de manera que en el servidor

se pueda extraer el método y se ejecute un conjunto de sentencias que consistieron en enviar notificaciones con las preferencias de lenguaje para subtítulos, audio y la dirección url del archivo mpd.

Se creó una presentación MPEG-DASH mediante los ejecutables disponibles en bento4.com. Se utilizó ffmpeg para crear 3 versiones de video, cada una codificada con diferentes tasas de bits, escalas de video y frecuencias de muestreo, el único valor que debe ser el mismo en las 3 versiones de video es la duración del grupo de imágenes que es un parámetro obligatorio indicado en la documentación de bento4, después de fragmentar las 3 versiones mediante el comando mp4fragment y con una duración de 4000 ms por cada fragmento se ejecutó el comando mp4dash el cual generó una carpeta que contiene los segmentos de audio/video en diferentes calidades y el archivo mpd necesario para que el reproductor seleccione los segmentos de acuerdo al ancho de banda disponible.

Se creó un archivo denominado clnt.js el cual fue importado en html. Las funciones que tiene este archivo permitieron acceder a ciertos objetos del DOM para asignarles eventos como presionar una tecla, dar un click y su respectiva subrutina ante estos eventos. Para lograr esto se requiere un identificador de elementos dentro del código html, en este caso el atributo id cumple esta función. Otros tipos de eventos utilizados están relacionados con el objeto que representa al video y se activan cuando el mismo se encuentre en reproducción permitiendo determinar cuándo se muestra un anuncio o en el caso de que se seleccione otro idioma para los subtítulos enviar un mensaje mediante el protocolo JSON-RPC.

La ventaja de utilizar Node.js en comparación a las funciones de otros servidores es que se puede descargar módulos para ofrecer servicios adicionales como base de datos, servidor http y servidor websocket, todos controlados y personalizados mediante

sentencias JavaScript. El módulo express permitió atender solicitudes http mediante el método GET en el caso de una consulta y POST cuando se envían datos hacia el servidor. Express requiere especificar una ruta a la cual se dirige la solicitud y una función que se ejecutara en respuesta. Nedb permitió crear una base de datos embebida, en el servidor se almaceno en un archivo llamado database.db y en formato JSON los votos seleccionados por el usuario. Express-ws se usó para crear un servidor websocket en la ruta "/echo", una vez que se crea la conexión entre cliente y servidor se intercambiaron mensajes siguiendo el protocolo JSON-RPC.

Se utilizó dash.js que es una librería escrita en el lenguaje JavaScript y funciona como reproductor de contenido MPEG-DASH, gracias a los algoritmos de adaptación que implementa se pudo evidenciar que dependiendo de la tasa de transmisión disponible en la red del usuario se seleccionaron segmentos con diferentes escalas de video y calidad de audio, esta información es evidenciable en la cabecera de la solicitud http donde se incluye el nombre de la carpeta que contiene los segmentos. La carpeta con el nombre "1" corresponde a los archivos de mayor calidad mientras que las carpetas "2" y "3" almacenan archivos de calidad media y baja respectivamente.

Las redes de entrega de contenido son útiles debido a que muchos archivos se encuentran disponibles en una red de servidores que los transfieren, de esta manera se puede utilizar librerías sin la necesidad de descargar estos archivos de manera local. En este proyecto se usó la CDN de bootstrap, Chart.js y Dash.js dentro del documento html, mediante las etiquetas script, link y el atributo src que requiere especificar la dirección url de cada CDN.

Se conoció que al consultar datos mediante APIs la información de repuesta se presenta en formato JSON y se almacenan en objetos dentro de JavaScript, estos pueden tener propiedades con tipos de variables numéricas, strings, booleans , arreglos e incluso



otros objetos por lo que JavaScript facilita su manipulación mediante funciones. La ventaja de utilizar APIs de la plataforma rapidAPI es que proporciono el código JavaScript para ser copiado y un botón de prueba para observar cómo será la respuesta en formato JSON, en el caso de la API open weather map fue muy útil para extraer las propiedades acerca del clima y la temperatura.

### **Recomendaciones**

Al ser el Ecuador un país en el cual esta tecnología aún no se encuentra desarrollada es necesario ampliar el campo de investigación en esta área.

Es importante que el espectro televisivo en el Ecuador desarrolle procesos de innovación y transformación para permitir al usuario hacer uso de nuevas tecnologías que contribuyan a la interacción con diferentes plataformas.

El estándar ATSC 3.0 se configura como uno de los más avanzados a nivel global, donde se puede aprovechar sus beneficios y características es esencial para que la televisión en el Ecuador se mantenga en línea con los estándares internacionales, brindando a los usuarios la oportunidad de aprovechar al máximo las ventajas gracias al uso de internet. Alcanzar este nivel de desarrollo facilitaría que a futuro se incorporen nuevas tecnologías, razón por la cual es fundamental explorar el campo investigativo en este ámbito

Respecto a la ejecución de la aplicación, debido a que se necesita reiniciar el servidor muy seguido mientras se hace pruebas con el código del lado del servidor se puede instalar el módulo nodemon mediante el administrador de paquetes npm, este módulo reinicia automáticamente el servidor cuando se detectan cambios en el código.

## Referencias

- Advanced Television Systems Committee. (2019). *ATSC 3.0 Interactive Content (A/344)*. Washington, D.C. : Advanced Television Systems Committee.
- Alvarez, M., Galeano, D., & Peña, J. (2017). *Manual del lenguaje de marcación de HTML5*. Sevilla : Saga Suite.
- ATSC. (2019). *ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection (A/331)*. Washington.
- Azarcoya, W. (2019). *La televisión de la nueva generación ATSC 3.0*. Washington D.C: ATSC Organization.
- Bento4. (s.f.). *Getting Started*. Recuperado el 7 de mayo de 2021, de <https://www.bento4.com/developers/dash/>
- Bootstrap. (s.f.). *Contenedores Bootstrap*. Recuperado el 2 de mayo de 2021, de <https://getbootstrap.com/docs/5.0/layout/containers/>
- Bootstrap. (s.f.). *Documentation Bootstrap*. Recuperado el 3 de mayo de 2021, de <https://getbootstrap.com/docs/4.0/components/badge/>
- Bootstrap. (s.f.). *sistem de malla bootstrap*. Recuperado el 1 de mayo de 2021, de <https://getbootstrap.com/docs/4.1/layout/grid/>
- Chasillacta, M. (2020). *Reconocimiento y seguimiento de plataformas para el aterrizaje automático de un vehículo aéreo no tripulado basado en inteligencia artificial y odometría visual [Tesis Pregrado]*. Universidad de las Fuerzas Armadas-ESPE.
- Chernock, R. (2018). *Next Generation TV: ATSC 3.0*. Nueva Jersey: IEEE Broadcast Technology Society.
- Chernock, R., & Whitaker, J. (2017). ATSC 3.0 The Next Step in the Evolution of Digital Television. 166-169. doi:10.1109/TBC.2017.2652018

- cloudflare. (s.f.). Recuperado el 4 de mayo de 2021, de CDN cloudflare:  
<https://www.cloudflare.com/es-es/learning/cdn/what-is-a-cdn/>
- DekTec. (19 de Octubre de 2019). *Herramientas ATSC 3.0: descripción general*.  
Recuperado el 14 de Diciembre de 2020, de <https://www.dektec.com/ATSC3/>
- Euceda, J. (2020). *Sistema para la integridad del vídeo en sistemas de streaming dash*.  
Universidad Politécnica de Valencia, Departamento de comunicaciones, Valencia.
- Fernández, X. (24 de 04 de 2019). *Estudio para la planificación de redes de difusión según el estándar ATSC 3.0*. Escuela de Ingeniería de Bilbao, estado de Iowa.
- Gallegos, C. (2008). *Características técnicas ISDB-T*. Asunción: DIREC.
- González, C. (2002). *En Televisión Digital: ATSC o DVB*. Londres : BG Engineering Desing .
- Graves, R., & Guidobono, J. (2009). *ATSC: La norma digital para todos los televidentes de la TV abierta libre y gratuita*. Buenos Aires: Honorable Senado de la República de Argentina.
- Javascript.info. (s.f.). *websockets*. Recuperado el 5 de mayo de 2021, de <https://javascript.info/websocket>
- Jung, B. C. (2016). The effects of second-screen viewing and the goal congruency of supplementary content on user perceptions. *Computers in Human Behaviour*, 347-354.
- León , O., & González, O. (2008). *Las telecomunicaciones de Banda ancha en la región Américas*. UIT-Unión Internacional de telecomunicaciones.
- Morales, A. (2010). *Diseño de la red para interactividad en Televisión digital terrestre e IPTV en el campus ESPE Sangolquí*. Ingeniería en electronica y telecomunicaciones. Sangolquí: ESPE.
- Mozilla. (s.f.). *Geolocation API*. Recuperado el 5 de mayo de 2021, de [https://developer.mozilla.org/en-US/docs/Web/API/Geolocation\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API)

- Mozilla. (s.f.). *HTTP Methods*. Recuperado el 5 de mayo de 2021, de <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
- Mozilla. (s.f.). *Introducción al DOM*. Recuperado el 5 de Mayo de 2021, de [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model/Introduction](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction)
- Mozilla. (s.f.). *Metadata Html*. Recuperado el mayo de 1 de 2021, de [https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\\_to\\_HTML/The\\_head\\_metadata\\_in\\_HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML)
- Nakachi, T. (2007). MPEG Media Transport technologies and its application to immersive media communication services. 111-116. doi:10.1109/ICDV.2017.8188649
- Nakachi, T. (2017). *MPEG Media Transport technologies and its application to immersive media communication services*. Toledo: 7th International Conference on Integrated Circuits, Design, and Verification (ICDV).
- Ortiz, M. (01 de 11 de 2017). *ATSC 3.0*. Escuela Superior Politécnica De Chimborazo, Ingeniería Electrónica, Riobamba. Obtenido de SCRIBD.
- Pinto, K., & Barredo, D. (2017). La interactividad en la televisión analógica y digital: Algunas ideas sobre diálogo audiovisual permanen. *Revista 100-Cs*, 59-69.
- Roberto, C., Gomez, D., & Whitaker, J. (2016). ATSC 3.0 Next Generation Digital TV Standard An Overview and Preview of the Issue. *IEEE Trans. Broadcasting*, 62(1), 154-158.
- Rodas, A., & Valencia, A. (2018). Desarrollo e implementación de un prototipo para una plataforma tecnológica para la transmisión de texto y video (streaming) en tiempo real empleando tecnología websocket. *Ingenierías USBMed*, IX(2), 2-10.
- Sotelo, R., Joskowicz, J., Uviedo, N., & Rondán, N. (2016). *Estándares de Interactividad en Televisión Híbrida*. Agencia Nacional de Investigación e Innovación, Investigaciones en Ingeniería, Uruguay.

- Starzynski, J., & Thomsen, P. (2017). The Technology Evaluation and Selection Process for ATSC 3.0. *IEEE Transactions on Broadcasting*, 63, 170-178. doi:10.1109/TBC.2017.2652118.
- T., N. (2007). MPEG Media Transport technologies and its application to immersive media communication services. 111-116. doi:10.1109 / ICDV.2017.8188649
- UIT-R. (2017). *Recomendación UIT-R BT.2075-1, Sistema integrado de radiodifusión-banda ancha*.
- Unión Internacional de Telecomunicaciones. (2020). *ATSC 3.0: Update*. Ginebra: UIT.
- Venkatraman, K., Vellingiri, S., & Prabhakaran, B. (2014). *MPEG Media Transport (MMT) for 3D Tele-Immersion Systems*. Texas: IEEE International Symposium on Multimedia.
- w3schools. (s.f.). *Node.js Npm*. Recuperado el 7 de mayo de 2021, de [https://www.w3schools.com/nodejs/nodejs\\_npm.asp](https://www.w3schools.com/nodejs/nodejs_npm.asp)
- w3schools. (s.f.). *w3schools CSS*. Recuperado el 3 de mayo de 2021, de [https://www.w3schools.com/css/css\\_positioning.asp](https://www.w3schools.com/css/css_positioning.asp)
- Xu, Y., Xie, S., Hao, C., & Le, Y. (2016). Dash and MMT an Their Applications in ATSC 3.0. *ZTE Communications*, 39-49.
- Xu, Y., Xie, S., Hao, C., Le, Y., & Jun, S. (2016). Dash and MMT an Their Applications in ATSC 3.0. *ZTE Communications*, 39-49.
- Zhang, L. (2016). Layerd-division-multiplexing: Theory and practice. *IEEE Transactions on Broadcasting*, 62(1), 216-232.

**Anexos**