

# *PRÁCTICA 3: DASH*

IRAC (2S) – Equipo 22

*Cristina González, Daniel Conde, Álvaro Ollero  
y Carlos Aznar*

*IRAC(2S) | 11.Mayo.2020*

# ÍNDICE

<b>1. RESUMEN</b>	<b>4</b>
<b>2. INTRODUCCIÓN AL DASH</b>	<b>5</b>
2.2 RESULTADOS	5
<b>3. GENERACIÓN DE VÍDEOS PARA STREAMING</b>	<b>9</b>
3.1 GENERACIÓN DE DIFERENTES CONFIGURACIONES DE VIDEO	9
3.2 PREPARACIÓN DE LOS VÍDEOS PARA DASH Y GENERACIÓN DEL FICHERO MPD	13
3.3 IMPLEMENTACIÓN DEL CLIENTE DE VISUALIZACIÓN Y ANÁLISIS DE MÉTRICAS	13
<b>4. GESTIÓN DRM CON DASH.JS</b>	<b>17</b>
4.1 IMPLEMENTACIÓN DEL CLIENTE DE VISUALIZACIÓN	20
<b>5. PARTE OPCIONAL</b>	<b>23</b>
<b>6. NOTA ADICIONAL</b>	<b>31</b>
<b>7. REFERENCIAS</b>	<b>32</b>

# ÍNDICE DE FIGURAS

Figura 1. URL del fichero MPD	5
Figura 2. Fichero index.html	5
Figura 3. Fichero manifiesto.MPD	6
Figura 4. Visualización del comportamiento MPEG-DASH	8
Figura 5. Métricas del vídeo descargado	10
Figura 6. Comando x264	11
Figura 7. Configuración de calidad baja con x264	11
Figura 8. Configuración de calidad media con x264	11
Figura 9. Configuración de calidad alta con x264	12
Figura 10. Comando de ejecución de MP4Box	12
Figura 11. Codificación en MPEG el video de calidad baja con MP4	12
Figura 12. Codificación en MPEG el video de calidad media con MP4	12
Figura 13. Codificación en MPEG el video de calidad alta con MP4	12
Figura 14. Comandos para la fragmentación de los videos a diferentes calidades	13
Figura 15. MPD asociado a la segmentación	13
Figura 16. Mensajes ofrecidos por el servidor al reproducir el video	14
Figura 17. Visualización del video en el lado del cliente	16
Figura 18. Información proporcionada por el comando mp4info del vídeo de baja calidad	17
Figura 19. Información proporcionada por el comando mp4info del vídeo de media calidad	18
Figura 20. Información proporcionada por el comando mp4info del vídeo de alta calidad	18
Figura 21. Comando de ejecución para la encriptación de los vídeos	19
Figura 22. Comandos para la encriptación de cada vídeo	19
Figura 23. Fichero MPD generado	19
Figura 24. Visualización del video encriptado en el lado del cliente	20
Figura 25. Modificación de las claves de encriptado en el index.html	21
Figura 26. Visualización video encriptado después de modificar las claves especificadas en protData	22
Figura 27. Formulario de la herramienta AXINOM Media Tools: Selección SystemID	24
Figura 28. Formulario de la herramienta Axinom Media Tools. Selección Data y Protection Scheme	24
Figura 29. Formulario de la herramienta Axinom Media Tools: Generated PSSH Box	25
Figura 30. Decodificación PSSH Box	25
Figura 31. Valor en Base-64 de PSSH Box	26

Figura 32. Codificación de la KEY a cadena de 32 caracteres	27
Figura 33. Encriptación de los ficheros de vídeo mediante el comando mp4dash para Widevine	27
Figura 34. Generación del fichero MPD con la codificación de los videos	28
Figura 35. Código JAVASCRIPT correspondiente a la protección DRM con Widevine	29
Figura 36. Visualización del vídeo en el lado del cliente, empleando el modelo de Widevine	29

## 1. RESUMEN

El objetivo de esta práctica es la creación de una aplicación web para proporcionar un servicio de Streaming basado en el protocolo MPEG-DASH. Para ello, se ha desarrollado un servidor que organiza los recursos a través del navegador y un cliente que ofrece el entorno de visualización al usuario.

Para la creación del contenido, se ha elegido un video adecuado para observar los efectos del uso de MPEG-DASH.

A continuación, se ha codificado este video de tres maneras distintas a través del codificador x264 creando tres representaciones, una de baja calidad, otra de calidad media y por último una de alta calidad, cada una con sus correspondientes características.

Posteriormente, se ha vuelto a codificar en MP4 mediante la herramienta MP4Box. Una vez conseguidos los 3 vídeos codificados en MP4 se han fragmentado y segmentado con la herramienta Bento4 mediante el comando *mp4dash*. Se explican los pasos que se han seguido, así como los resultados que se han obtenido.

Seguidamente, los vídeos codificados con MP4Box se han encriptado con el comando *mp4encrypt*, mediante el método MPEG-CENC y definiendo una KEY y una KID necesaria. Se ha comprobado que la desencriptación del video funciona correctamente, por lo que el cliente va a poder acceder al video. También, se ha realizado una prueba en la que se ha modificado las claves de encriptado y se ha comprobado qué pasa cuando el cliente va a querer acceder al video deseado.

Finalmente, se ha realizado la mejora de emplear otro modelo de encriptación, más concretamente, se ha empleado el método de encriptación para la gestión de DRM de *Widevine*. De esta manera, cuando el reproductor intenta reproducir el contenido protegido de *Widevine*, se enviará una solicitud al servicio de entrega de licencias para obtener la licencia. Si este servicio de licencias aprueba la solicitud, se le enviará al cliente dicha licencia y se le proporcionará una KID, una KEY y un token para poder reproducir el contenido especificado. Se detalla los pasos que se han seguido para poder obtener las diferentes claves de encriptado y desencriptado, así como los resultados obtenidos mediante este mecanismo.

## 2. INTRODUCCIÓN AL DASH

DASH (“Dynamic Adaptative Streaming over HTTP”) es un protocolo de streaming de video empleado para transmitir sobre Internet video en vivo o bajo demanda. Como indica su nombre, se adapta dinámicamente a las condiciones de la red.

Este protocolo genera fragmentos pequeños de audio y de video a los que llama “Media Presentation” (MP), y para informar sobre la estructura de los fragmentos crea un archivo llamado “Media Presentation Description” (MPD) en el que se encuentra información sobre la ubicación de los segmentos, codificación del vídeo, idioma del audio... [1].

En esta parte de la práctica se realiza el despliegue y uso de la herramienta dash.js para ofrecer un servicio de streaming básico de un vídeo codificado con diferentes calidades. El objetivo con esto es analizar el funcionamiento del protocolo DASH (Dynamic Adaptative Streaming over HTTP).

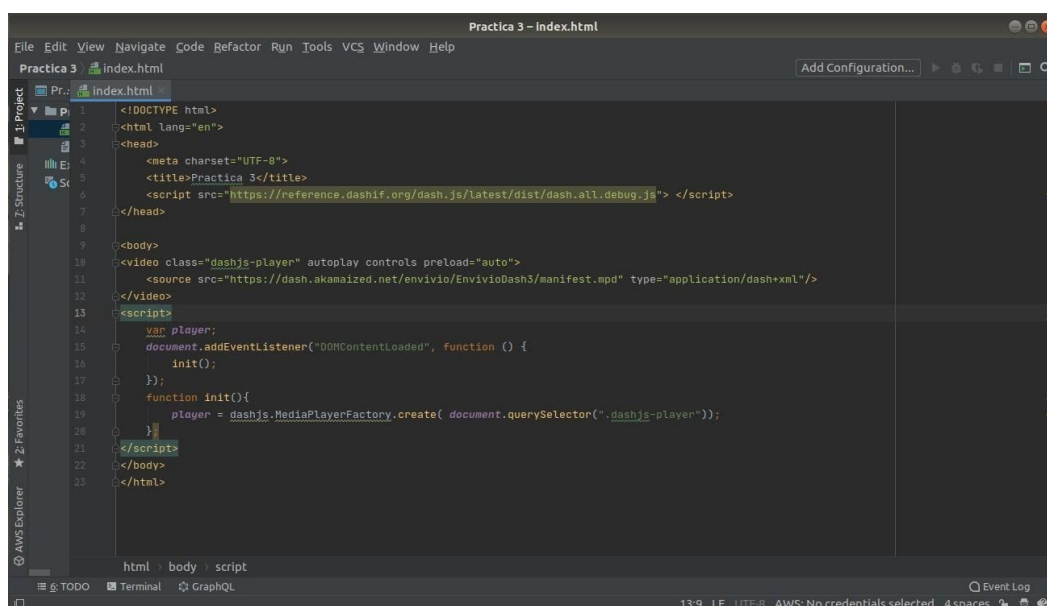
### 2.2 RESULTADOS

Para llevar a cabo el objetivo de esta primera parte comentado anteriormente, se utiliza el video ofrecido por **envivo** a través del MPD alojado en la siguiente URL:

```
https://dash.akamaized.net/envivo/EnvivioDash3/manifest.mpd
```

*Figura 1. URL del fichero MPD*

La etiqueta HTML necesaria para crear el elemento video es la que se muestra en la siguiente figura en la que se puede ver que se ha añadido la librería de la herramienta DASH.



*Figura 2. Fichero index.html*

En la siguiente figura vemos el contenido del fichero MPD en el que podemos encontrar los siguientes atributos:

```
<MPD xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd" profiles="urn:mpeg:dash:profile:isoff-live:2011" type="static" minBufferTime="PT5.000S"
maxSegmentDuration="PT2.005S" availabilityStartTime="2016-01-20T21:10:02Z" mediaPresentationDuration="PT193.680S">
  <Period id="period0">
    <AdaptationSet mimeType="video/mp4" segmentAlignment="true" startWithSAP="1" maxWidth="1920" maxHeight="1080" maxFrameRate="30000/1001" par="1:1">
      <SegmentTemplate timescale="90000" initialization="$RepresentationID$-Header.m4s" media="$RepresentationID$-270146-i-$Number$.m4s" startNumber="1" duration="179704"
presentationTimeOffset="0"/>
      <Representation id="v1_257" bandwidth="1200000" codecs="avc1.4D401E" width="768" height="432" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v2_257" bandwidth="1850000" codecs="avc1.4D401E" width="1024" height="576" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v3_257" bandwidth="2850000" codecs="avc1.4D401E" width="1280" height="720" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v4_257" bandwidth="2000000" codecs="avc1.4D401E" width="320" height="180" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v5_257" bandwidth="3000000" codecs="avc1.4D401E" width="320" height="180" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v6_257" bandwidth="4300000" codecs="avc1.4D401E" width="1280" height="720" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v7_257" bandwidth="5300000" codecs="avc1.4D401E" width="1920" height="1080" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v8_257" bandwidth="4800000" codecs="avc1.4D401E" width="512" height="288" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
      <Representation id="v9_257" bandwidth="750000" codecs="avc1.4D401E" width="640" height="360" frameRate="30000/1001" sar="1:1" scanType="progressive"/>
    </AdaptationSet>
    <AdaptationSet mimeType="audio/mp4" segmentAlignment="true" startWithSAP="1" lang="aa">
      <SegmentTemplate timescale="90000" initialization="$RepresentationID$-Header.m4s" media="$RepresentationID$-270146-i-$Number$.m4s" startNumber="1" duration="179704"
presentationTimeOffset="0"/>
      <Representation id="v4_258" bandwidth="130800" codecs="mp4a.40.2" audioSamplingRate="48000">
        <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Figura 3. Fichero manifiesto.MPD

La descripción de algunos de los atributos es la siguiente:

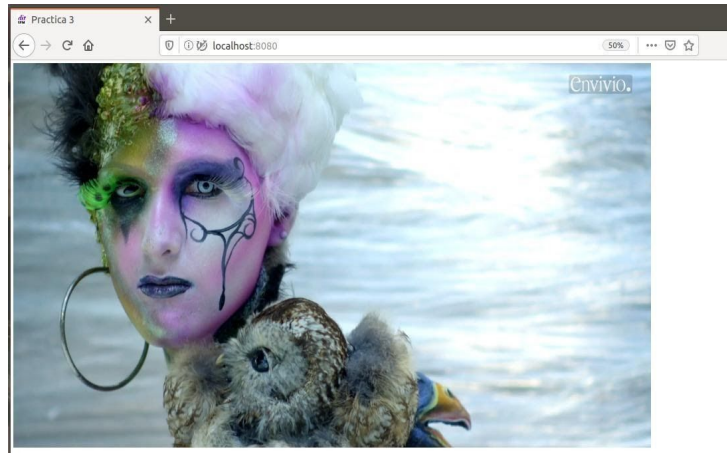
Campo	Significado	Valor
profiles	Especifica una lista de perfiles MP	Urn:mpeg:dash:profile:isoff-live:2011
type	Especifica si el MPD debe ser actualizado ("dynamic") o no ("static")	static
minBufferTime	Especifica una duración que se utiliza para saber cuándo se puede iniciar la reproducción del video	5 segundos
maxSegmentDuration	Es el tiempo que dura el segmento	2.005 segundos
availabilityStartTime	Al ser "static" especifica el tiempo en el que están disponibles todos los segmentos MPD. Si el atributo no aparece entonces todos los segmentos MPD están disponibles en el momento en el que esta disponible el MPD. En nuestro	21:10:02 horas
mediaPresentationDuration	Especifica la duración de todo el MPD	193.68 segundos
Period	Especifica la información de un Period. Donde un Period es un intervalo del contenido multimedia en el eje de tiempos.	Period0

AdaptationSet mimeType	Descripción del AdaptationSet. Donde un AdaptationSet proporciona información sobre uno o más contenidos multimedia y sus diferentes alternativas de codificación.	Video/mp4
SegmentAlign	Operación de ordenar los segmentos de los índices de la columna en una columna en particular.	True
startWithSAP	Tiene un valor de 1 o 2, que se corresponde con los tipos de SAP en [ISO-14496-12]	1
maxWidth	Máxima anchura	1920
maxHeight	Máxima altura	1080
maxFrameRate	Máximo framerate	30000/1001
par	Relación de aspecto de pixel	1:1
SegmentTemplate	Indica que el MPD está basado en templates: indican unas reglas que el cliente tiene que seguir para conseguir las URL	
Representation	Cada representación es una posible codificación de un contenido multimedia. Dentro de cada uno se especifica el ancho de banda, tipo de códec, ancho, alto, framerate, tipo de escaneado (progresivo o entrelazado)...	
Lang	Este campo indica que el audio es el audio original.	qaa
AudioSamplingRate	Frecuencia de muestreo de audio.	48000

Además, podemos observar que tenemos nueve representaciones que muestran codificaciones con calidades diferentes. Por otro lado, observamos como las resoluciones de las imágenes (height\*width) son mayores cuando el ancho de banda del canal (bandwidth) es más grande. Como se puede ver en todo lo comentado, se reflejan las principales características de MPEG-DASH.

En la siguiente imagen podemos ver el comportamiento del MPEG-DASH en el lado del cliente.





*Figura 4. Visualización del comportamiento MPEG-DASH*

En primer lugar, observamos que podemos acceder al servidor, el cual organiza el acceso al video, a través del navegador mediante el url: <https://localhost:8080/>.

En segundo lugar, el cliente nos ofrece la parte de visualización del vídeo en *Streaming*, el cual se corresponde al fichero MPD, descrito anteriormente.

Cabe destacar que, el video solo se puede ver en alta calidad, ya que el fichero MPD, solo permite esa representación.

### 3. GENERACIÓN DE VÍDEOS PARA STREAMING

En esta parte de la práctica, el objetivo principal es la generación de distintas calidades de un mismo video para ser ofrecido en streaming mediante MPEG-DASH y la visualización de las métricas en el cliente web.

Para llevar a cabo este objetivo se emplean:

- Las herramientas X264 y MP4Box para la configuración de resoluciones y bitrate de un video y su codificación final a MP4
- La herramienta Bento4 para la división de cada video en segmento de streaming y la creación del fichero de manifiesto MPD

#### 3.1 GENERACIÓN DE DIFERENTES CONFIGURACIONES DE VIDEO

Se ha llevado a cabo la instalación de las herramientas X264 y MP4Box en el sistema operativo Ubuntu 18.04 de la siguiente manera:

- **Instalación x264:** Al realizar las prácticas en un sistema operativo Ubuntu, solo ha sido necesario, para instalar esta herramienta, ejecutar el siguiente comando en el terminal del ordenador: *sudo apt-get install x264*.

La versión que se instaló fue la siguiente: 0.152.2854 e9a5903

- **Instalación MP4Box:** Al igual que ocurre con la instalación de x264, al estar en un sistema operativo de Ubuntu, esta herramienta la hemos podido instalar desde el terminal, siguiendo los siguientes pasos [3]:

1. Instalamos el software que nos permitirá usar *svn* en nuestro ordenador:  
*apt-get install -y subversion*
2. El siguiente paso es descargar la última versión de *gpac*:  
*svn co https://svn.code.sf.net/p/gpac/code/trunk/gpac gpac*
3. Accedemos al directorio donde hemos descargado la versión de *gpac*, modificamos los permisos de los ficheros y, finalmente, instalamos el paquete *gpac* el cual contiene la herramienta MP4.

La versión de la herramienta MP4 instalada es la 0.5.2.

Una vez instaladas se ha llevado a cabo la creación de los vídeos.

Cabe destacar que, el video utilizado en esta parte ha sido un video de Youtube con buena calidad (resolución en HD) para que después, durante la práctica, se pueda observar perfectamente las diferencias entre las codificaciones que más adelante se han realizado obteniendo una calidad baja, media y alta. Por otro lado, la duración del video es de 9 minutos y 58 segundos (598 segundos) y su contenedor de video y audio es Quicktime.

En cuanto al video, como ya se ha comentado anteriormente, es de resolución HD (1280x720), codificado con el códec H.264 y con un framerate de 25 fotogramas por segundo.

En cuanto al audio, está codificado con MPEG-4 AAC que usa codificación modular, su frecuencia de muestreo es de 44100 Hz y tiene dos canales (estéreo).



*Figura 5. Métricas del vídeo descargado*

El video elegido es el que se ve en la Figura 5 debido a todas las propiedades anteriormente mencionadas y a una adicional que permite observar el comportamiento del protocolo MPEG-DASH adecuadamente. Esa propiedad adicional es que el video dura más de 5 minutos. Por ello, se eligió una de aproximadamente 10 minutos debido a que preferimos dejar un margen suficientemente largo entre el límite (5 min) y la duración de nuestro vídeo.

Una vez descargado dicho vídeo, se ha generado diferentes configuraciones de codificación mediante la herramienta X264. Las configuraciones elegidas son, como especifica el enunciado, tres diferentes:

- **Calidad baja:** con una resolución de 160x90 y con un bitrate de 100 kbps.
- **Calidad media:** con una resolución de 640x360 y con un bitrate de 600 kbps.
- **Calidad alta:** con una resolución de 1280x720 y con un bitrate de 2400 kbps.

Como se puede observar, la calidad más alta tiene la misma resolución que tiene el video original y esto nos permite ver de una forma correcta los efectos del MPEG-DASH.

Para obtener estas calidades se ha tenido que descargar la herramienta x264 de Internet e instalar en Ubuntu empleando para ello un ordenador con sistema operativo Linux.

Posteriormente, hemos proseguido a su utilización para obtener las diferentes calidades de vídeo. Para ello, se ha empleado la siguiente línea de comando variando lo que se encuentra en **negrita** por lo requerido en cada calidad.

**[COMANDO DE EJECUCIÓN DE x264]** `--output Low_config.264 --fps 24 --preset slow --bitrate 100 --vbv-maxrate 4800 --vbv-buftype 9600 --min-keyint 48 --keyint 48 --scenecut 0 --no-scenecut --pass 1 --video-filter "resize:width=160,height=90" video_original.mp4`

*Figura 6. Comando x264*

Los comandos específicos para cada calidad se pueden observar en las siguientes imágenes:

```
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$ x264 --output low_config.264 --fps 24 --preset slow --bitrate 100 --vbv-maxrate 4800 --vbv-buftype 9600 --min-keyint 48 --keyint 48 --scenecut 0 --no-scenecut --pass 1 --video-filter "resize:width=160,height=90" Video.mp4
ffms [info]: 1280x720p 1:1 @ 25/1 fps (cfr)
resize [info]: resizing to 160x90
x264 [info]: using SAR=1/1
x264 [info]: using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
x264 [info]: profile Main, level 3.0
x264 [info]: frame I:312 Avg QP:15.56 size: 3348
x264 [info]: frame P:7508 Avg QP:20.63 size: 728
x264 [info]: frame B:7146 Avg QP:26.34 size: 167
x264 [info]: consecutive B-frames: 31.1% 13.1% 8.1% 47.7%
x264 [info]: mb I 16..4: 23.5% 0.0% 76.5%
x264 [info]: mb P 16..4: 4.9% 0.0% 0.0% P16..4: 52.5% 0.0% 0.0% 0.0% 0.0% skip:42.6%
x264 [info]: mb B 16..4: 0.3% 0.0% 0.0% B16..8: 16.2% 0.0% 0.0% direct: 7.9% skip:75.5% L0:18.3% L1:24.8% BI:56.8%
x264 [info]: final ratefactor: 18.92
x264 [info]: direct mvs spatial:84.3% temporal:15.7%
x264 [info]: coded y,u,vDC,uvAC intra: 73.3% 64.1% 50.8% inter: 21.2% 5.8% 2.4%
x264 [info]: i16 v,h,dc,p: 15% 66% 13% 6%
x264 [info]: i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 13% 37% 16% 5% 5% 5% 7% 4% 9%
x264 [info]: i8c dc,h,v,p: 44% 43% 9% 4%
x264 [info]: Weighted P-Frames: Y:2.1% UV:0.5%
x264 [info]: kb/s:98.77

encoded 14966 frames, 195.09 fps, 98.77 kb/s
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$
```

*Figura 7. Configuración de calidad baja con x264*

```
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$ x264 --output medium_config.264 --fps 24 --preset slow --bitrate 600 --vbv-maxrate 4800 --vbv-buftype 9600 --min-keyint 48 --keyint 48 --scenecut 0 --no-scenecut --pass 1 --video-filter "resize:width=640,height=360" Video.mp4
ffms [info]: 1280x720p 1:1 @ 25/1 fps (cfr)
resize [info]: resizing to 640x360
x264 [info]: using SAR=1/1
x264 [info]: using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
x264 [info]: profile Main, level 3.0
x264 [info]: frame I:312 Avg QP:15.04 size: 24719
x264 [info]: frame P:4854 Avg QP:19.02 size: 5841
x264 [info]: frame B:9800 Avg QP:23.07 size: 1041
x264 [info]: consecutive B-frames: 8.5% 9.4% 9.6% 72.5%
x264 [info]: mb I 16..4: 38.1% 0.0% 61.9%
x264 [info]: mb P 16..4: 11.2% 0.0% 0.0% P16..4: 38.5% 0.0% 0.0% 0.0% 0.0% skip:50.3%
x264 [info]: mb B 16..4: 1.1% 0.0% 0.0% B16..8: 15.7% 0.0% 0.0% direct: 3.5% skip:79.8% L0:31.1% L1:38.5% BI:30.5%
x264 [info]: final ratefactor: 20.86
x264 [info]: direct mvs spatial:88.3% temporal:11.7%
x264 [info]: coded y,u,vDC,uvAC intra: 56.1% 43.2% 21.7% inter: 9.3% 3.2% 0.4%
x264 [info]: i16 v,h,dc,p: 35% 42% 16% 8%
x264 [info]: i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 21% 34% 10% 5% 5% 5% 8% 5% 8%
x264 [info]: i8c dc,h,v,p: 53% 31% 13% 3%
x264 [info]: Weighted P-Frames: Y:1.8% UV:0.9%
x264 [info]: kb/s:593.53

encoded 14966 frames, 112.99 fps, 593.53 kb/s
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$
```

*Figura 8. Configuración de calidad media con x264*



```

carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$ x264 --output high_config.264 --fps 24 --preset slow --bitrate 2400 --vbv-maxrat
e 4800 --vbv-buFSIZE 9600 --min-keyint 48 --keyint 48 --scenecut 0 --no-scenecut --pass 1 --video-filter "resize:width=1280,height=720" Video.
mp4
ffms [info]: 1280x720p 1:1 @ 25/1 fps (cfr)
x264 [info]: using SAR=1/1
x264 [info]: using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
x264 [info]: profile Main, level 3.1
x264 [info]: frame I:312 Avg OP:10.61 size: 85408
x264 [info]: frame P:4681 Avg OP:14.50 size: 23588
x264 [info]: frame B:9973 Avg OP:18.88 size: 4631
x264 [info]: consecutive B-frames: 4.7% 14.5% 14.9% 66.0%
x264 [info]: mb I I16..4: 40.8% 0.0% 59.2%
x264 [info]: mb P I16..4: 14.8% 0.0% 0.0% P16..4: 39.8% 0.0% 0.0% 0.0% 0.0% skip:45.4%
x264 [info]: mb B I16..4: 1.6% 0.0% 0.0% B16..8: 20.2% 0.0% 0.0% direct: 4.3% skip:73.9% L0:35.6% L1:42.6% BI:21.8%
x264 [info]: final ratefactor: 17.45
x264 [info]: direct mvs spatial:96.4% temporal:3.6%
x264 [info]: coded y,uvDC,uvAC intra: 50.0% 37.6% 17.0% inter: 10.0% 4.5% 0.4%
x264 [info]: i16 v,h,dc,p: 40% 38% 14% 8%
x264 [info]: i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 24% 34% 9% 5% 5% 5% 7% 5% 7%
x264 [info]: i8c dc,h,v,p: 54% 29% 14% 4%
x264 [info]: Weighted P-Frames: Y:1.7% UV:1.0%
x264 [info]: kb/s:2350.86

encoded 14966 frames, 59.48 fps, 2350.86 kb/s
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$

```

*Figura 9. Configuración de calidad alta con x264*

Una vez realizadas las 3 configuraciones cada vídeo se tiene que codificar en MPEG-4. Para ello, se ha utilizado la herramienta de codificación MP4Box, en la cual se consigue codificar los videos en MPEG-4 con el comando que se presenta a continuación:

```
[COMANDO DE EJEC. MP4Box] -add [nombre_entrada].264-fps 24 [nombre_salida].mp4
```

*Figura 10. Comando de ejecución de MP4Box*

Por lo tanto, para cada uno de los videos se ha introducido los siguientes comandos para así generar las codificaciones MPEG-4 para las 3 configuraciones de vídeo:

```

carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$ MP4Box -add low_config.264 -fps 24 low_config.mp4
AVC-H264 import - frame size 160 x 90 at 24.000 FPS
AVC Import results: 14966 samples - Slices: 312 I 7508 P 7146 B - 1 SEI - 312 IDR
Stream uses forward prediction - stream CTS offset: 2 frames
Saving to low_config.mp4: 0.500 secs Interleaving
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$

```

*Figura 11. Codificación en MPEG el video de calidad baja con MP4*

```

carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$ MP4Box -add medium_config.264 -fps 24 medium_config.mp4
AVC-H264 import - frame size 640 x 360 at 24.000 FPS
AVC Import results: 14966 samples - Slices: 312 I 4854 P 9800 B - 1 SEI - 312 IDR
Stream uses forward prediction - stream CTS offset: 2 frames
Saving to medium_config.mp4: 0.500 secs Interleaving
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$

```

*Figura 12. Codificación en MPEG el video de calidad media con MP4*

```

carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$ MP4Box -add high_config.264 -fps 24 high_config.mp4
AVC-H264 import - frame size 1280 x 720 at 24.000 FPS
AVC Import results: 14966 samples - Slices: 312 I 4681 P 9973 B - 1 SEI - 312 IDR
Stream uses forward prediction - stream CTS offset: 2 frames
Saving to high_config.mp4: 0.500 secs Interleaving
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3$

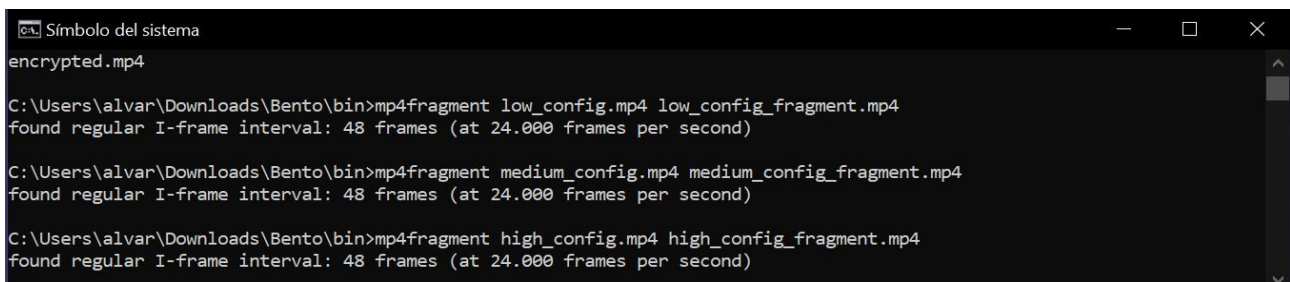
```

*Figura 13. Codificación en MPEG el video de calidad alta con MP4*

## 3.2 PREPARACIÓN DE LOS VÍDEOS PARA DASH Y GENERACIÓN DEL FICHERO MPD

En esta segunda parte, se realiza la segmentación de cada video mediante la herramienta Bento4. Para ello, se ha descargado la herramienta Bento4 en el sistema operativo Windows 10 mediante la página web proporcionada en el enunciado de la práctica.

Para la fragmentación de los videos se ha usado el comando proporcionado por el enunciado de la práctica, como se puede ver en la siguiente figura.



```
Símbolo del sistema
encrypted.mp4

C:\Users\alvar\Downloads\Bento\bin>mp4fragment low_config.mp4 low_config_fragment.mp4
found regular I-frame interval: 48 frames (at 24.000 frames per second)

C:\Users\alvar\Downloads\Bento\bin>mp4fragment medium_config.mp4 medium_config_fragment.mp4
found regular I-frame interval: 48 frames (at 24.000 frames per second)

C:\Users\alvar\Downloads\Bento\bin>mp4fragment high_config.mp4 high_config_fragment.mp4
found regular I-frame interval: 48 frames (at 24.000 frames per second)
```

Figura 14. Comandos para la fragmentación de los videos a diferentes calidades

Posteriormente, se ha creado el fichero MPD:

```
<MPD mediaPresentationDuration="PT10M23.583S" minBufferTime="PT2.00S" profiles="urn:mpeg:dash:profile:isoff-live:2011" type="static">
  <!--
    Created with Bento4 mp4-dash.py, VERSION=2.0.0-633
  -->
  <Period>
    <!-- Video -->
    <AdaptationSet maxHeight="720" maxWidth="1280" mimeType="video/mp4" segmentAlignment="true" startWithSAP="1">
      <SegmentTemplate duration="2000" initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/seg-$Number$.m4s" startNumber="1" timescale="1000"/>
      <Representation bandwidth="172106" codecs="avc1.4D401E" frameRate="24" height="90" id="video/avc1/1" scanType="progressive" width="160"/>
      <Representation bandwidth="1284260" codecs="avc1.4D401E" frameRate="24" height="360" id="video/avc1/2" scanType="progressive" width="640"/>
      <Representation bandwidth="4008939" codecs="avc1.4D401F" frameRate="24" height="720" id="video/avc1/3" scanType="progressive" width="1280"/>
    </AdaptationSet>
  </Period>
</MPD>
```

Figura 15. MPD asociado a la segmentación

Como se puede observar, la duración del fragmento es menor ya que se trata de un fichero de video fragmentado (fMP4). Además, podemos observar que tenemos tres representaciones ya que se ha codificado con tres calidades diferentes. También vemos como las resoluciones de las imágenes (height\*width) son mayores cuando el ancho de banda del canal o de la red (bandwidth) es más grande. Por lo tanto, vemos como se reflejan las principales características de MPEG-DASH.

## 3.3 IMPLEMENTACIÓN DEL CLIENTE DE VISUALIZACIÓN Y ANÁLISIS DE MÉTRICAS

Por último, se debe integrar lo que se ha hecho en los puntos 3.1 y 3.2 mediante la implementación de un cliente web capaz de ofrecer el video codificado en streaming.

Para ello, en primer lugar, lo que tenemos que hacer es ir a la carpeta donde se encuentra el fichero *index.html*, ya que contiene el código necesario para poder reproducir el video. Más concretamente, éste fichero se encuentra dentro del directorio *output*, el cual a su

vez, también contiene el fichero MPD, anteriormente descrito, para poder visualizar las diferentes representaciones de calidad que se han generado.

Una vez que estamos en el directorio correcto, el siguiente paso es arrancar el lado del servidor web en el puerto 8080, para ofrecer los recursos que se encuentran dentro del directorio. Estos recursos, como se ha dicho antes, es el fichero HTML que ofrecerá la reproducción del video a través de la web. Para poder arrancar el servidor se ejecuta el siguiente comando: `python -m SimpleHTTPServer 8080`.

Una vez arrancado el servidor, el último paso se realiza desde el lado del cliente. Dicho cliente debe acceder a un navegador web, ya sea *Google Chrome* o *Firefox*, y poner la siguiente dirección url, para poder tener acceso a la reproducción del video que se encuentra en el lado del servidor: <https://localhost:8080/>

Los resultados que hemos obtenido se pueden ver en las siguientes figuras:

```
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3/Bento4-SDK-1-6-0-633.x86_64-unknown-linux/utls_copia/output$ python -m SimpleHTT
PServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
127.0.0.1 - - [30/Apr/2020 20:52:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:30] "code 404, message File not found"
127.0.0.1 - - [30/Apr/2020 20:52:30] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [30/Apr/2020 20:52:30] "GET /stream.mpd HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/1/init.mp4 HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/1/seg-1.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/2/init.mp4 HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/2/seg-2.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/init.mp4 HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-3.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-4.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-5.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-6.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-7.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-8.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-9.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:31] "GET /video/avc1/3/seg-10.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-11.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-12.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-13.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-14.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-15.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-16.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-17.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-18.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-19.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-20.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-21.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-22.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-23.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-24.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-25.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-26.m4s HTTP/1.1" 200 -
127.0.0.1 - - [30/Apr/2020 20:52:32] "GET /video/avc1/3/seg-27.m4s HTTP/1.1" 200 -
```

Figura 16. Mensajes ofrecidos por el servidor al reproducir el video

Esta figura se corresponde con los datos que se obtienen en el lado del servidor. En primer lugar, se puede observar que está ofreciendo servicio en el puerto 8080, como habíamos establecido previamente. Este mensaje aparece nada más iniciar el servidor.

Una vez que hemos accedido desde el lado cliente a la dirección <https://localhost:8080/>, aparece el segundo mensaje `GET /HTTP/1.1 200-`, lo que nos indica que se ha establecido correctamente la conexión con el servidor por parte del cliente.

Una vez que aparece este mensaje, si todo funciona correctamente, el vídeo comenzará a reproducirse por pantalla. Como se puede apreciar en la imagen, aparecen dos mensajes de errores, los cuales después se solucionan correctamente. El primer mensaje de error (`code 400, message File not found-`), nos indica que en un primer momento no se había encontrado el fichero que contiene el código que debería de ejecutarse para reproducir el video. El segundo mensaje de error (`GET /favicon.ico HTTP/1.1 404-`) nos indica que el icono asociado al sitio a reproducir no se ha encontrado.



Una vez que se han solucionado estos errores, podemos ver un quinto mensaje (*GET /stream.mpd HTTP/1.1 200-*) que nos da información acerca de que el fichero MPD ha sido recuperado y se transmite el mensaje al body, es decir, se inicia la reproducción del video y, por tanto, su visualización por la pantalla, ya que este fichero es el que contiene toda la información para que los diferentes videos segmentados se reproduzcan.

Finalmente, los siguientes mensajes se corresponden con la reproducción del video. En ellos podemos ver cada segundo que se está reproduciendo, así como la calidad de la imagen que se reproduce en ese segundo. Podemos ver tres opciones diferentes:

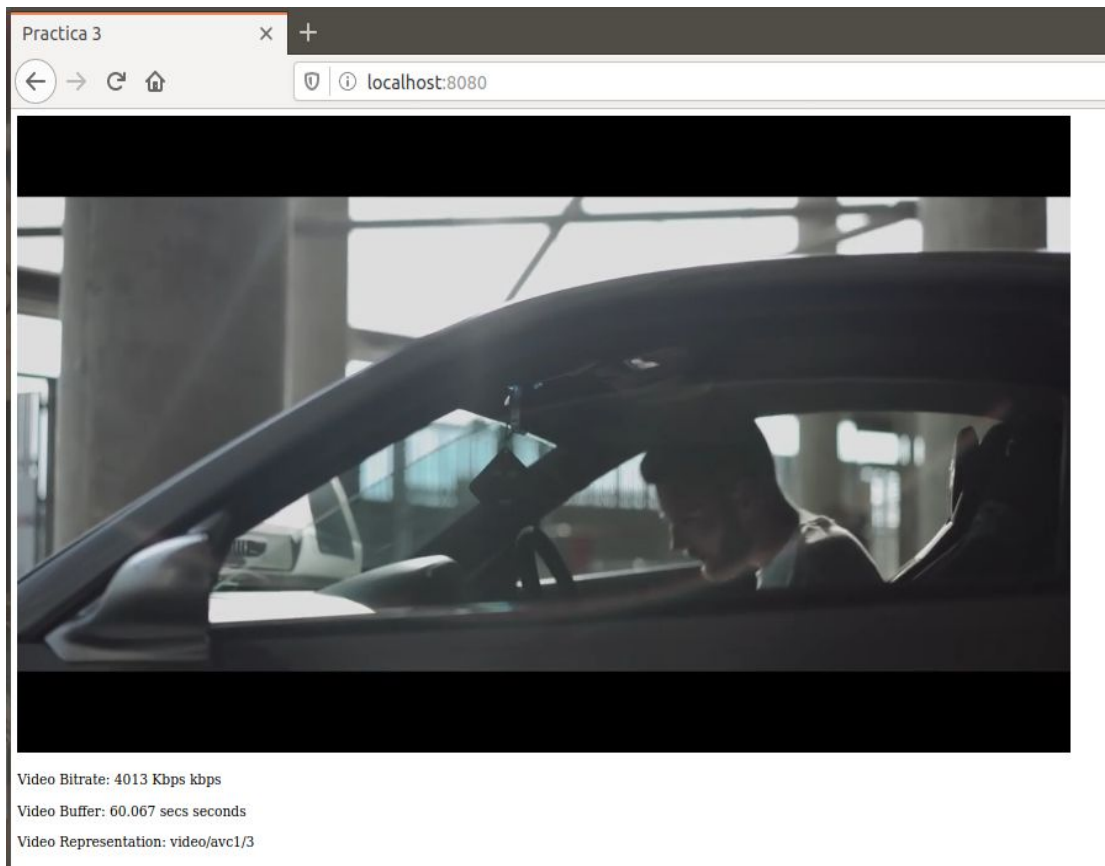
- **GET /video/avc1/1/seg-1.m4s HTTP/1.1 200-:** Esta petición nos indica que justo en el segundo video, el frame del video se corresponde con la calidad más baja que hemos establecido, ya que se está accediendo a la carpeta que contiene la representación de calidad más baja (*/video/avc1/1/*).

Esto se debe a que, al iniciar el video el buffer se encuentra vacío y se necesita que se llene para que la reproducción se realice correctamente y no haya congelaciones durante la reproducción. Por lo tanto, para dar tiempo a que el buffer se llene, el video se empieza a reproducir automáticamente con la calidad más baja.

- **GET /video/avc1/2/seg-2.m4s HTTP/1.1 200-:** En esta petición, que se corresponde con el segundo 2 de la reproducción del video, el frame correspondiente a este segundo tiene una calidad media. Lo sabemos porque la petición está accediendo al directorio que contiene la representación del video en dicha calidad (*/video/avc1/2/*). Esto se debe a que, el buffer está lo suficientemente lleno para ofrecer una mejor calidad de video sin que éste se vacíe totalmente y se produzcan congelaciones
- **GET /video/avc1/3/seg-3.m4s HTTP/1.2 200-:** Es la tercera petición y, si observamos la imagen, es la que más se repite. Se corresponde con la representación del video en alta calidad, ya que como hemos explicado en las anteriores peticiones, se ve que se está accediendo a la carpeta que contiene las representaciones con esta calidad (*/video/avc1/3/*). A partir del tercer segundo y en adelante, la reproducción del video se realizará con la máxima calidad posible, esto es, con la alta calidad. Esto es posible porque el tamaño del buffer que hemos establecido en la Figura 9 es mucho mayor que el bitrate que tiene el video en esta calidad, lo que permite que el buffer se vacíe totalmente y así poder establecer reproducciones de alta calidad sin congelaciones.

No solo existe la posibilidad de ver la reproducción del video en el terminal desde el lado del servidor y ver sólo si se establece bien la conexión y la calidad con la que el video se está representando en cada segundo. Lo mejor de todo es verlo desde el lado del cliente, es decir, ver el video desde el entorno de visualización del usuario. Esta representación desde el lado del usuario se puede apreciar en la siguiente imagen:





*Figura 17. Visualización del video en el lado del cliente*

En dicha imagen, se puede observar a simple vista que el video se está reproduciendo. En primer lugar, si nos fijamos en la url que aparece, efectivamente, nos encontramos accediendo al servidor que contiene el video desde el puerto deseado, el puerto 8080.

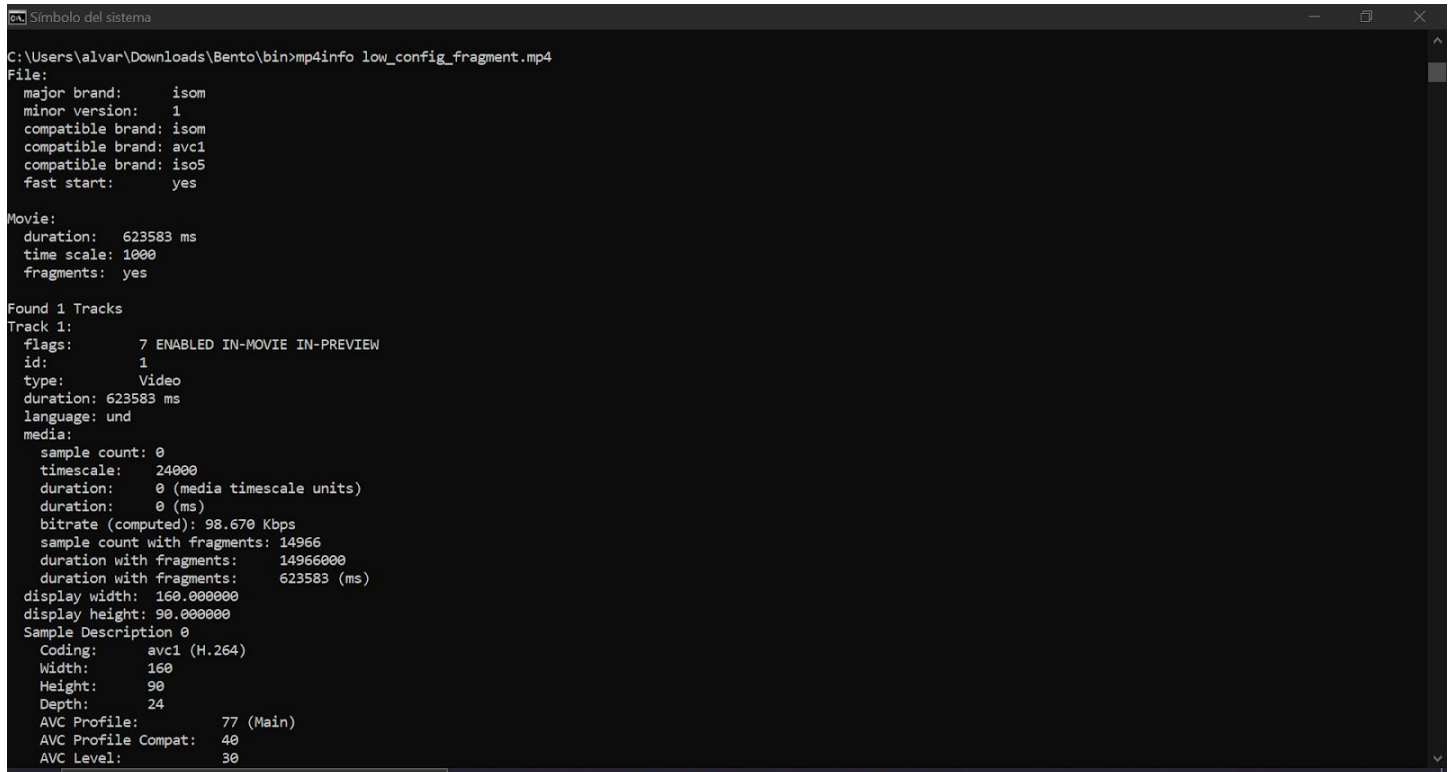
Por otro lado, para poder analizar mejor los datos de la reproducción, podemos analizar los valores que se encuentran debajo del video. En primer lugar, se encuentra el bitrate del video en ese momento, que este caso se corresponde con un valor de 4013 kbps. En segundo lugar, está la métrica del buffer la cual aparece con un valor de 60.067 segundos, lo que nos hace indicar que está lleno por lo que el riesgo de que haya congelaciones es bastante bajo. Finalmente, podemos analizar la métrica de la calidad de reproducción.

Justo en este momento, el video se está reproduciendo a la máxima calidad posible ya que la representación que se está teniendo acceso se encuentra en la carpeta número 3 del directorio `/video/avc1`. Esta carpeta, como hemos mencionado anteriormente, contiene la representación de los fragmentos de video en alta calidad.

## 4. GESTIÓN DRM CON DASH.JS

En esta parte de la práctica, el objetivo es adquirir conocimientos sobre la gestión de derechos digitales (DRM). Para conseguir esto se han utilizado las herramientas mp4fragment y mp4encrypt de Bento4.

Para encriptar los vídeos hay que ver cuántos tracks tiene cada archivo de vídeo generado. Para ello, se ha usado el comando *mp4info* a cada archivo de vídeo. La información que ofrece el comando es la que se puede ver en las siguientes figuras:



```
Símbolo del sistema
C:\Users\alvar\Downloads\Bento\bin>mp4info low_config_fragment.mp4
File:
  major brand: isom
  minor version: 1
  compatible brand: isom
  compatible brand: avc1
  compatible brand: iso5
  fast start: yes

Movie:
  duration: 623583 ms
  time scale: 1000
  fragments: yes

Found 1 Tracks
Track 1:
  flags: 7 ENABLED IN-MOVIE IN-PREVIEW
  id: 1
  type: Video
  duration: 623583 ms
  language: und
  media:
    sample count: 0
    timescale: 24000
    duration: 0 (media timescale units)
    duration: 0 (ms)
    bitrate (computed): 98.670 Kbps
    sample count with fragments: 14966
    duration with fragments: 14966000
    duration with fragments: 623583 (ms)
    display width: 160.000000
    display height: 90.000000
  Sample Description 0
    Coding: avc1 (H.264)
    Width: 160
    Height: 90
    Depth: 24
    AVC Profile: 77 (Main)
    AVC Profile Compat: 40
    AVC Level: 30
```

Figura 18. Información proporcionada por el comando mp4info del vídeo de baja calidad

```
Símbolo del sistema
C:\Users\alvar\Downloads\Bento\bin>mp4info medium_config_fragment.mp4
File:
major brand:      isom
minor version:    1
compatible brand: isom
compatible brand: avc1
compatible brand: iso5
fast start:       yes

Movie:
duration: 623583 ms
time scale: 1000
fragments: yes

Found 1 Tracks
Track 1:
flags:          7 ENABLED IN-MOVIE IN-PREVIEW
id:             1
type:           Video
duration: 623583 ms
language: und
media:
sample count: 0
timescale: 24000
duration: 0 (media timescale units)
duration: 0 (ms)
bitrate (computed): 593.419 Kbps
sample count with fragments: 14966
duration with fragments: 14966000
duration with fragments: 623583 (ms)
display width: 640.000000
display height: 360.000000
Sample Description 0
Coding:         avc1 (H.264)
Width:          640
Height:         360
Depth:          24
AVC Profile:    77 (Main)
AVC Profile Compat: 40
AVC Level:      30
```

Figura 19. Información proporcionada por el comando mp4info del vídeo de media calidad

```
Símbolo del sistema
C:\Users\alvar\Downloads\Bento\bin>mp4info high_config_fragment.mp4
File:
major brand:      isom
minor version:    1
compatible brand: isom
compatible brand: avc1
compatible brand: iso5
fast start:       yes

Movie:
duration: 623583 ms
time scale: 1000
fragments: yes

Found 1 Tracks
Track 1:
flags:          7 ENABLED IN-MOVIE IN-PREVIEW
id:             1
type:           Video
duration: 623583 ms
language: und
media:
sample count: 0
timescale: 24000
duration: 0 (media timescale units)
duration: 0 (ms)
bitrate (computed): 2350.746 Kbps
sample count with fragments: 14966
duration with fragments: 14966000
duration with fragments: 623583 (ms)
display width: 1280.000000
display height: 720.000000
Sample Description 0
Coding:         avc1 (H.264)
Width:          1280
Height:         720
Depth:          24
AVC Profile:    77 (Main)
AVC Profile Compat: 40
AVC Level:      31
```

Figura 20. Información proporcionada por el comando mp4info del vídeo de alta calidad

Se puede ver que, cada vídeo fragmentado (en el parámetro *fragmented*) solo tiene un track, por lo que solo se necesitará establecer un KEY y un KID para cada vídeo que podrá ser el mismo.

Posteriormente, se ha llevado a cabo la encriptación de cada vídeo mediante el comando que se puede ver en la Figura 21. Es importante destacar que, debido a que los vídeos solo tienen un track, solo tienen una clave KEY y una KID como se ha comentado anteriormente.

```
mp4encrypt --method MPEG-CENC --key 1:87237D20A19F58A740C05684E699B4AA:random --property 1:KID:A16E402B9056E371F36D348AA62BB749 --global-option mpeg-cenc.eme-pssh:true [INPUT] [OUTPUT]
```

Figura 21. Comando de ejecución para la encriptación de los vídeos

```
C:\Users\alvar\Downloads\Bento\bin>mp4encrypt --method MPEG-CENC --key 1:87237D20A19F58A740C05684E699B4AA:random --property 1:KID:A16E402B9056E371F36D348AA62BB749 --global-option mpeg-cenc.eme-pssh:true low_config_fragment.mp4 low_config_encrypted.mp4

C:\Users\alvar\Downloads\Bento\bin>mp4encrypt --method MPEG-CENC --key 1:87237D20A19F58A740C05684E699B4AA:random --property 1:KID:A16E402B9056E371F36D348AA62BB749 --global-option mpeg-cenc.eme-pssh:true medium_config_fragment.mp4 medium_config_encrypted.mp4

C:\Users\alvar\Downloads\Bento\bin>mp4encrypt --method MPEG-CENC --key 1:87237D20A19F58A740C05684E699B4AA:random --property 1:KID:A16E402B9056E371F36D348AA62BB749 --global-option mpeg-cenc.eme-pssh:true high_config_fragment.mp4 high_config_encrypted.mp4
```

Figura 22. Comandos para la encriptación de cada vídeo

Por último, se deben procesar del mismo modo que en los apartados anteriores para representarlos mediante el protocolo DASH, es decir, con el comando *mp4dash nombre\_vídeo* se genera la segmentación de cada video para su posterior presentación. Con esto, se genera una carpeta (*output* como nombre por defecto), con diferentes subcarpetas (numeradas por defecto) con todos los segmentos del vídeo y el documento MPD necesario para la presentación el cual se puede ver en la siguiente figura:

```
<MPD mediaPresentationDuration="PT10M23.583S" minBufferTime="PT2.00S" profiles="urn:mpeg:dash:profile:isoff-live:2011" type="static">
  <!--
    Created with Bento4 mp4-dash.py, VERSION=2.0.0-633
  -->
  <Period>
    <!-- Video -->
    <AdaptationSet maxHeight="720" maxWidth="1280" mimeType="video/mp4" segmentAlignment="true" startWithSAP="1">
      <SegmentTemplate duration="2000" initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/seg-$Number$.m4s" startNumber="1" timescale="1000"/>
      <Representation bandwidth="175813" codecs="avc1.4D401E" frameRate="24" height="90" id="video/avc1/1" scanType="progressive" width="160"/>
      <Representation bandwidth="1284284" codecs="avc1.4D401E" frameRate="24" height="360" id="video/avc1/2" scanType="progressive" width="640"/>
      <Representation bandwidth="4012985" codecs="avc1.4D401F" frameRate="24" height="720" id="video/avc1/3" scanType="progressive" width="1280"/>
    </AdaptationSet>
  </Period>
</MPD>
```

Figura 23. Fichero MPD generado

Como vemos, este fichero MPD es similar al anteriormente obtenido tras la segmentación. Las únicas variaciones son los valores del bandwidth de cada representación.

## 4.1 IMPLEMENTACIÓN DEL CLIENTE DE VISUALIZACIÓN

Una vez que hemos encriptado el video y hemos obtenido el fichero MPD correspondiente, el paso final es ver su representación en el lado del cliente y comprobar si efectivamente el cliente dispone de las claves adecuadas para descryptar el video y verlo correctamente o, por el contrario, no disponer de las claves correctas y que se produzca un error durante el descryptado y no poder reproducir el video.

Comenzamos con el primer caso:

Si analizamos el código que se encuentra en el fichero *index.html*, podemos ver que cuando el cliente acceda al servidor, éste último, dispondrá de las claves correctas que van a permitir descryptar el video y, por tanto, reproducirlo sin ningún tipo de problemas. Esto lo podemos ver en la siguiente imagen.

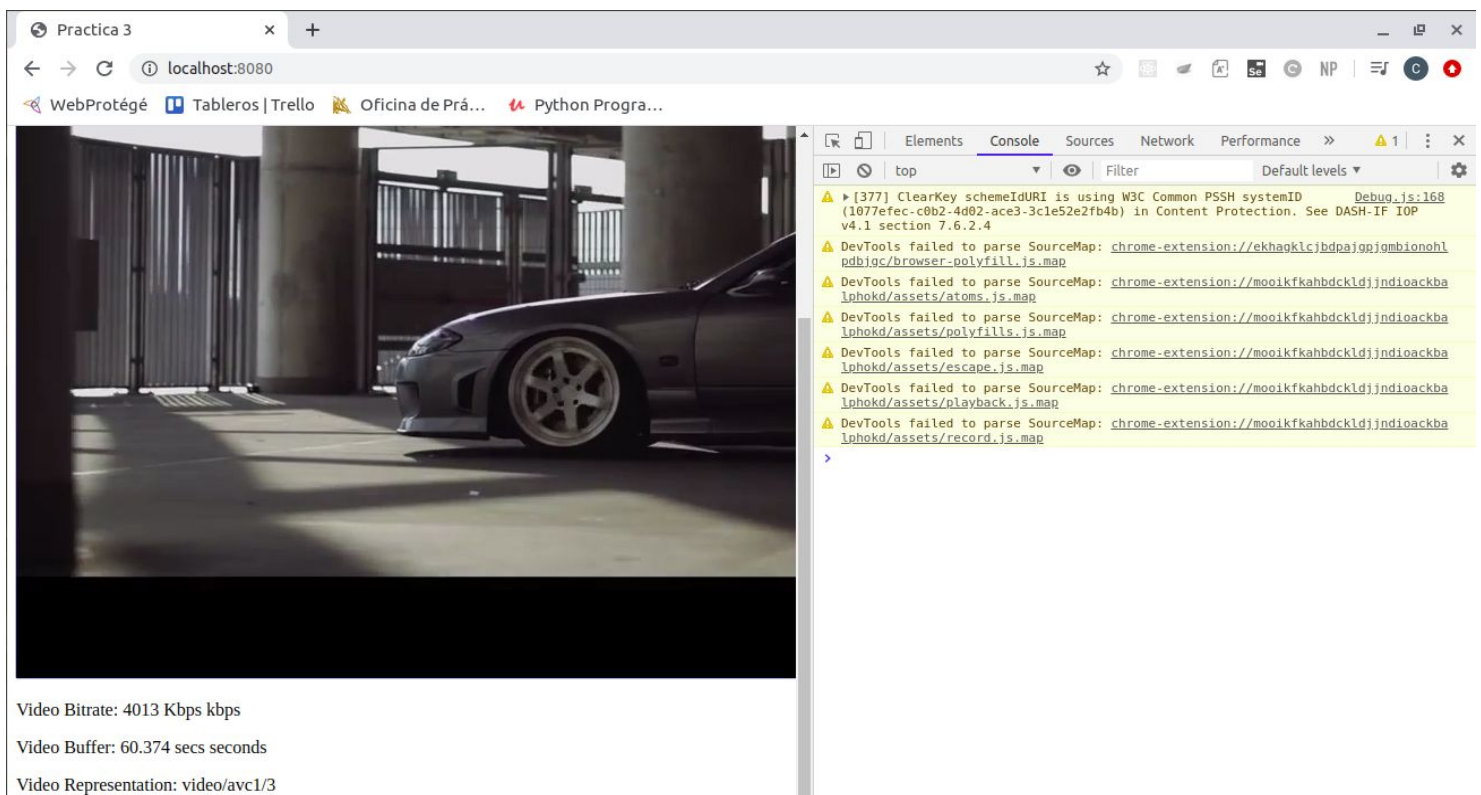


Figura 24. Visualización del video encriptado en el lado del cliente

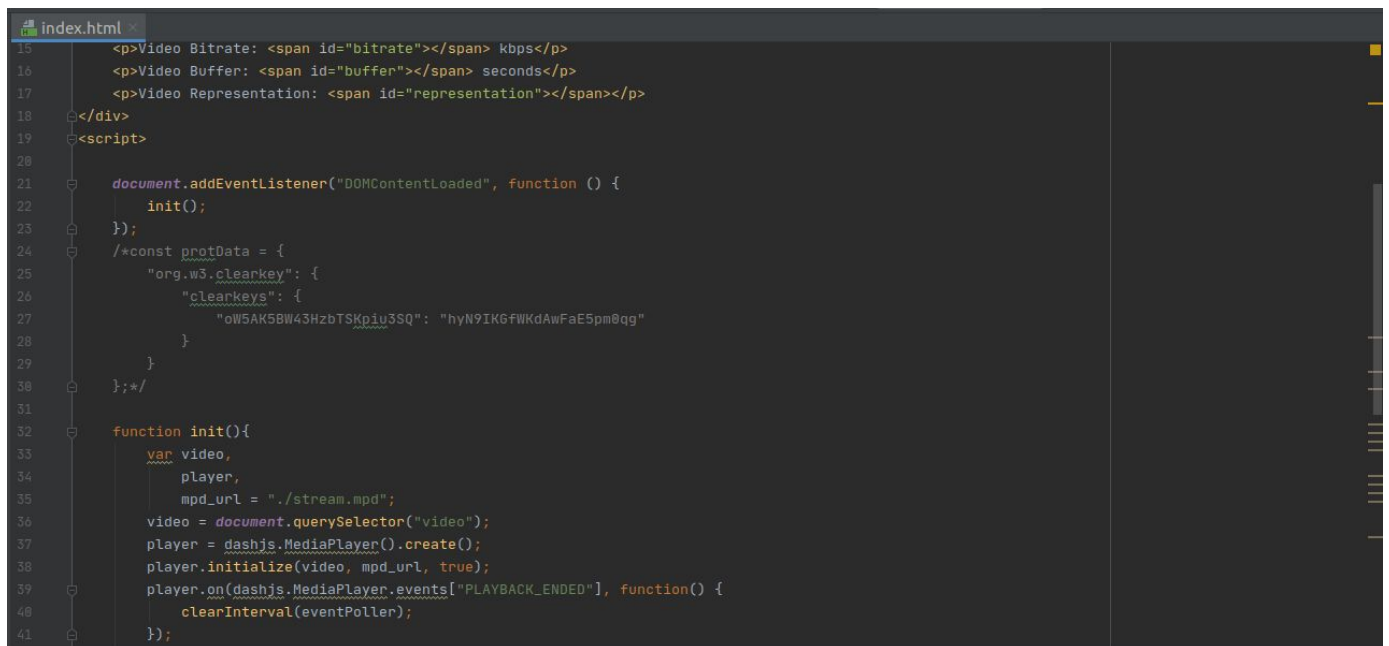
En esta imagen, podemos ver que estamos accediendo desde el lado del usuario y a través del navegador de *Google Chrome*, al servidor mediante la url <https://localhost:8080/>. Al igual que hemos explicado en el punto 2.3, antes de acceder desde el lado cliente, primero hay que arrancar el servidor desde el terminal del ordenador y poder tener acceso desde el puerto 8080 (igual que hemos hecho en ese punto). Una vez que vemos que el video se reproduce correctamente, podemos ver los mensajes que se ofrecen a través de la consola del navegador. Como era de esperar, no nos aparece ningún mensaje de error. Además, podemos ver que nos aparece un mensaje que nos indica que se ha empleado el método *Clearkey* para proteger el contenido el cual se ha encriptado con modo común.



Al igual que ha ocurrido con las representaciones anteriores, también podemos analizar los valores de las métricas para comprobar que el video se está reproduciendo correctamente. En este caso, se puede ver que se está viendo el video en la calidad más alta establecida.

Una vez que la descriptación ha funcionado correctamente, por lo que la visualización del video no ha dado ningún tipo de problema, el siguiente paso será modificar el código para que, cuando el cliente quiera acceder al video, el servidor no lo pueda descriptar.

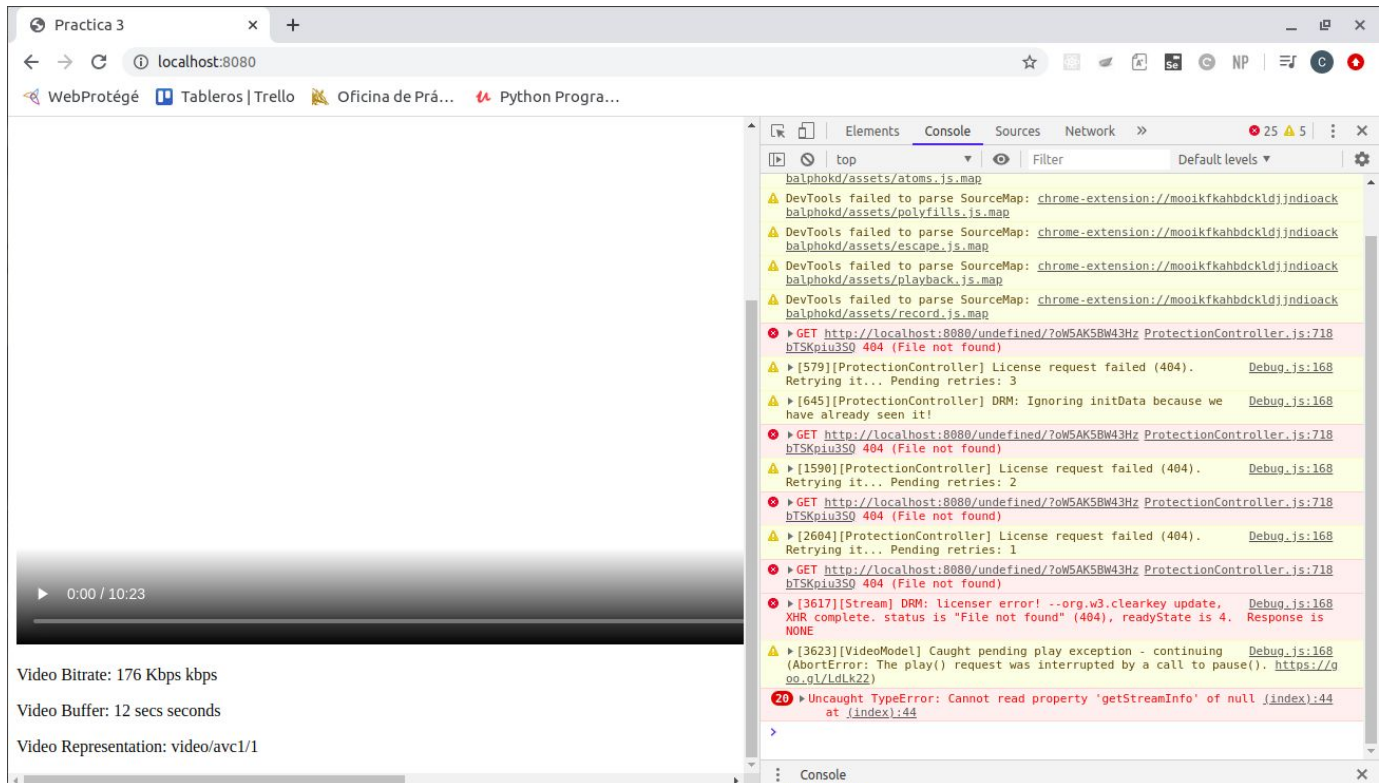
Por lo tanto, lo que hemos hecho ha sido quitar la parte del código relativa a la DRM, es decir, vamos a emplear el código que hemos empleado cuando el video no estaba encriptado. Esto se puede observar en la siguiente imagen.



```
15 <p>Video Bitrate: <span id="bitrate"></span> kbps</p>
16 <p>Video Buffer: <span id="buffer"></span> seconds</p>
17 <p>Video Representation: <span id="representation"></span></p>
18 </div>
19 <script>
20
21     document.addEventListener("DOMContentLoaded", function () {
22         init();
23     });
24
25     /*const protData = {
26         "org.w3.clearkey": {
27             "clearkeys": {
28                 "oW5AK5BW43HzbTSKpiu3SQ": "hyN9IK6fWKdAwFaE5pm0qg"
29             }
30         }
31     };*/
32
33     function init(){
34         var video,
35             player,
36             mpd_url = "./stream.mpd";
37         video = document.querySelector("video");
38         player = dashjs.MediaPlayer().create();
39         player.initialize(video, mpd_url, true);
40         player.on(dashjs.MediaPlayer.events["PLAYBACK_ENDED"], function() {
41             clearInterval(eventPoller);
42         });
43     }
44 </script>
```

*Figura 25. Modificación de las claves de encriptado en el index.html*

Obviamente, cuando el cliente acceda al servidor para poder ver el video, al estar el video encriptado, el servidor buscará las claves necesarias para descriptarlo. Como hemos empleado el código en el que el servidor no va a disponer de ningún tipo de claves, la descriptación no se realizará, por lo que el video no se podrá reproducir. Los resultados de esta parte se pueden ver en la siguiente imagen.



*Figura 26. Visualización video encriptado después de modificar las claves especificadas en protData*

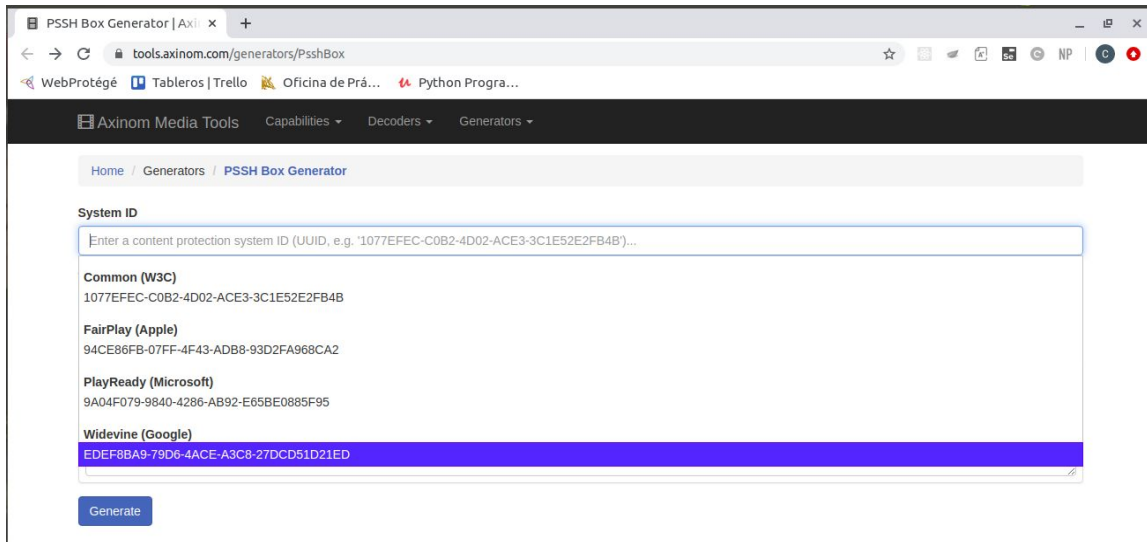
Esta imagen se corresponde con la visualización del video por parte del cliente en el navegador web de *Google Chrome*. Se puede apreciar en primer lugar que, estamos intentando acceder al servidor mediante el puerto 8080. El acceso al servidor se realiza correctamente ya que se puede apreciar la imagen del video con las métricas, sin embargo, no hay ningún tipo de contenido multimedia reproduciendo. Esto se debe a que, como hemos explicado antes, el servidor no es capaz de descryptar el video por lo que éste no se podrá reproducir.

Si analizamos la consola del navegador, efectivamente aparecen errores los cuales nos indican que no se ha encontrado ningún fichero que nos proporcione las claves de descryptado.





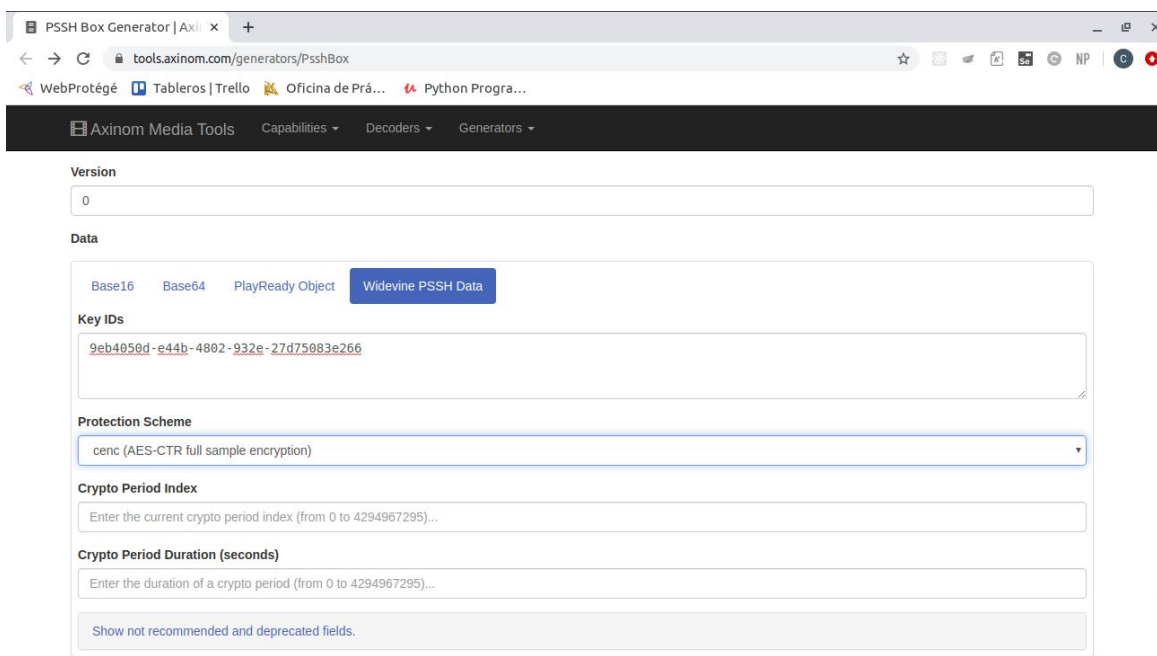
1. Tenemos que seleccionar el *System ID*, el cual nos proporciona una serie de opciones a elegir. Nosotros elegimos la opción de *Widevine*, ya que vamos a emplear esta herramienta:



The screenshot shows the 'PSSH Box Generator' web application. The 'System ID' section is active, displaying a list of content protection system IDs. The 'Widevine (Google)' option is selected, showing the ID 'EDEF8BA9-79D6-4ACE-A3C8-27DCD51D21ED'. A 'Generate' button is visible at the bottom of the form.

Figura 27. Formulario de la herramienta AXINOM Media Tools: Selección SystemID

2. Ponemos la versión en 0 y en *Data* seleccionamos *Widevine-PSSH-Data* y añadamos la KID. También, en *Protection Scheme* seleccionamos *cenc*, ya que las claves que hemos obtenido anteriormente están cifradas empleando el esquema de cifrado *cenc*:



The screenshot shows the 'PSSH Box Generator' web application with the 'Data' and 'Protection Scheme' sections. The 'Version' is set to '0'. In the 'Data' section, 'Widevine PSSH Data' is selected. The 'Key IDs' field contains the value '9eb4050d-e44b-4802-932e-27d75083e266'. The 'Protection Scheme' is set to 'cenc (AES-CTR full sample encryption)'. The 'Crypto Period Index' and 'Crypto Period Duration (seconds)' fields are empty. A 'Show not recommended and deprecated fields.' button is at the bottom.

Figura 28. Formulario de la herramienta Axinom Media Tools. Selección Data y Protetion Scheme

3. Se pulsa en botón *Generate* y se generará una cadena de caracteres en Base-64:

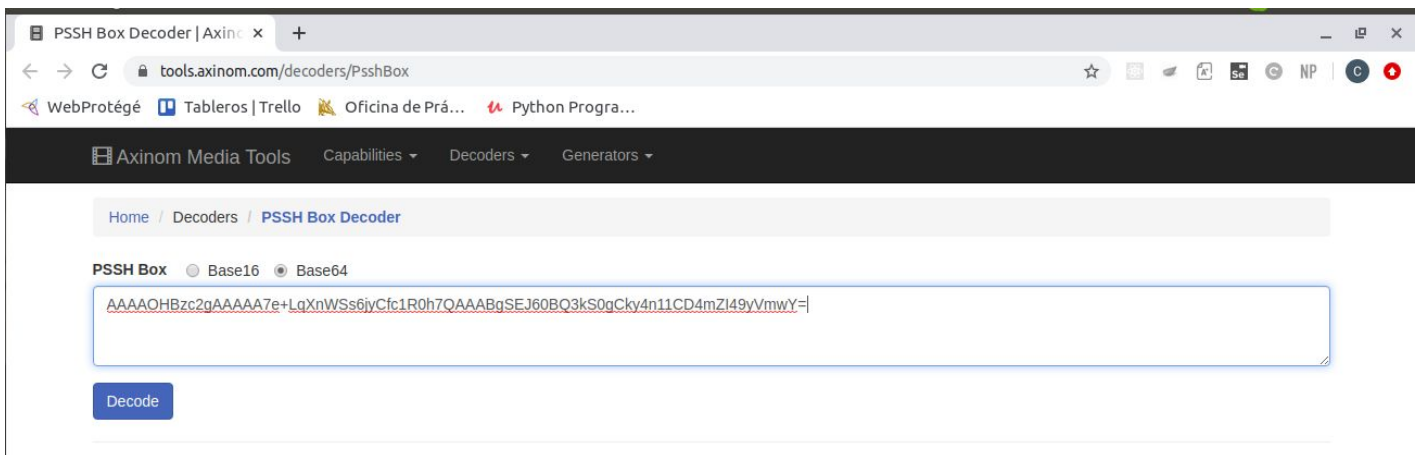
Generated PSSH box:

AAAAOHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABgSEJ60BQ3kS0gCky4n11CD4mZI49yVmwY=

*Figura 29. Formulario de la herramienta Axinom Media Tools: Generated PSSH Box*

Una vez que hemos obtenido esa cadena de caracteres, la cual corresponde con la generación del PSSH codificado, el siguiente paso es decodificar la cadena para obtener el valor de PSSH en base64 [7]:

1. Copiamos la cadena de caracteres obtenida anteriormente, elegimos la opción de Base64 y pulsamos el botón *Decode*.



*Figura 30. Descodificación PSSH Box*

2. A continuación, se puede observar que se obtienen una serie de valores de los cuales solo nos interesa uno de ellos. Nos vamos al apartado *PsshBox* y en el subapartado *Data* seleccionamos la opción *Base64*. En dicha opción encontramos el valor deseado (EhCetAUN5EtIAPmUj9dQg+JmSOPclZsG)

The screenshot shows the 'PSSH Box Decoder' tool on the Axinom website. The tool has three main sections: 'Box', 'FullBox', and 'PsshBox'. The 'Box' section shows 'Size' as 00 00 00 38 (56 bytes) and 'Type' as 70 73 73 68 (pssh). The 'FullBox' section shows 'Version' as 00 (0) and 'Flags' as 00 00 00 (00000000 00000000 00000000). The 'PsshBox' section shows 'System ID' as ED EF 8B A9 79 D6 4A CE A3 C8 27 DC D5 1D 21 ED (Widevine) and 'UUID' as edef8ba9-79d6-4ace-a3c8-27dc51d21ed. The 'Data Size' is 00 00 00 18 (24 bytes). Under the 'Data' section, there are three tabs: 'Widevine PSSH Data' (selected), 'Base16', and 'Base64'. The 'Widevine PSSH Data' tab shows the hex value 'EhCetAUN5EtIApMuJ9dQg+JmS0Pc1ZsG'.

Box	
Size	00 00 00 38 (56 bytes)
Type	70 73 73 68 (pssh)

FullBox	
Version	00 (0)
Flags	00 00 00 (00000000 00000000 00000000)

PsshBox	
System ID	ED EF 8B A9 79 D6 4A CE A3 C8 27 DC D5 1D 21 ED (Widevine)
UUID	edef8ba9-79d6-4ace-a3c8-27dc51d21ed
Data Size	00 00 00 18 (24 bytes)
Data	<div>Widevine PSSH Data Base16 Base64</div> <div>EhCetAUN5EtIApMuJ9dQg+JmS0Pc1ZsG</div>

Figura 31. Valor en Base-64 de PSSH Box

Una vez que tenemos todos estos valores, el siguiente paso es realizar la encriptación de los tres ficheros fragmentados empleando *mp4dash*, ya que mediante este comando se puede realizar la encriptación automáticamente utilizando la opción *--encryption-key*.

Cabe destacar que, el uso tanto de este comando como de esta opción es diferente en cuanto al modo de encriptación que empleemos. Como vamos a emplear el método *Widevine*, será necesario emplear también la opción *--widevine-header*, ya que mediante esta opción especificamos los valores de los datos de *Widevine DRM*. Este valor es el valor obtenido del cuadro PSSH (PSSHBox), expresado con el siguiente símbolo #, seguido del valor de los datos de carga útil PSSH codificados en hexadecimal.

Por otro lado, los valores correspondientes a la opción *--encryption-key* son los valores de la KID y la KEY, ambas separadas por dos puntos (:). Cabe destacar que, ambos valores tienen que ser de 32 caracteres para poder ser usados con la opción de *mp4dash*. Sin embargo, si observamos el valor de la KEY no es de 32 caracteres sino de 24, por lo tanto, hay que codificarlo en *Base64*, mediante la herramienta web CRYPTII para poder obtener la cadena de texto de 32 caracteres:

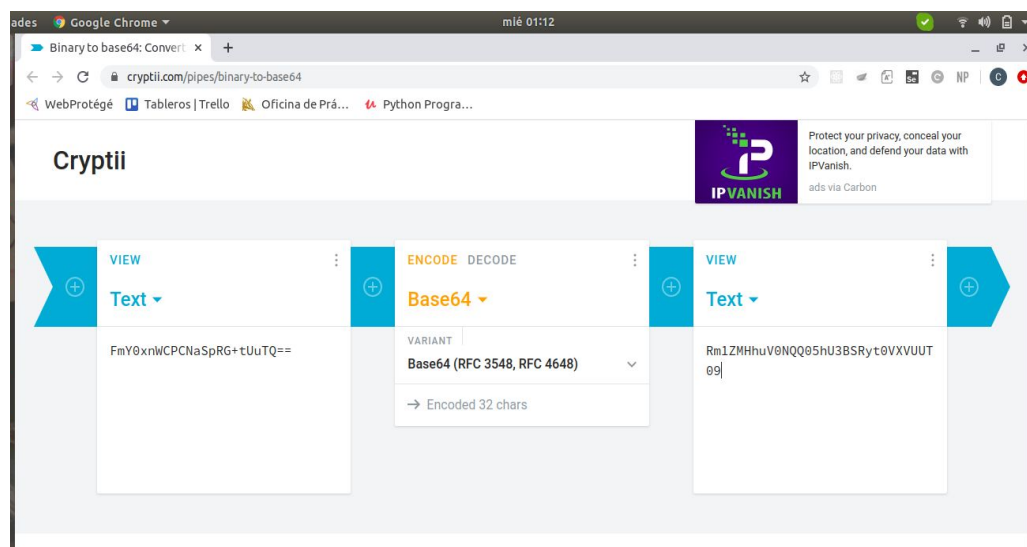


Figura 32. Codificación de la KEY a cadena de 32 caracteres

Una vez que tenemos todos los valores correctamente, el último paso es encriptar las tres representaciones que habíamos creado. A la vez que realizamos la encriptación, se va a realizar la segmentación de los ficheros que contienen las diferentes representaciones, obteniendo así un directorio igual al de los puntos anteriores, el cual contiene otro directorio de video, que contiene a su vez un directorio denominado avc1. Este directorio contiene a su vez un fichero por cada uno de los ficheros de vídeo codificados y segmentados (directorio 1 para la representación en Baja calidad; el directorio 2 para la representación en calidad media; y el directorio 3 para la representación calidad alta).

El comando empleado para encriptar y segmentar los ficheros de video es el que se muestra en la siguiente figura:

```
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3/Bento4-SDK-1-6-0-633.x86_64-unknown-linux/utls_opt$ python3 mp4-dash.py --widevine-header "#EhCetAUN5EtIApMuJ9dQg+JmSOPclZsG" --encryption-key 9eb4050de44b4802932e27d75083e266:Rm1ZMHhuV0NQ05hU3BSRyt0VXVUUT09 low_config_fragment.mp4 medium_config_fragment.mp4 high_config_fragment.mp4
Encrypting track IDs [1] in low_config_fragment.mp4
Encrypting track IDs [1] in medium_config_fragment.mp4
Encrypting track IDs [1] in high_config_fragment.mp4
Parsing media file 1: tmpetf4aj2e = Encrypted[low_config_fragment.mp4]
Parsing media file 2: tmp4f8n_zc6 = Encrypted[medium_config_fragment.mp4]
Parsing media file 3: tmpsis2m75e = Encrypted[high_config_fragment.mp4]
Splitting media file (video) tmpetf4aj2e = Encrypted[low_config_fragment.mp4]
Splitting media file (video) tmp4f8n_zc6 = Encrypted[medium_config_fragment.mp4]
Splitting media file (video) tmpsis2m75e = Encrypted[high_config_fragment.mp4]
carlos@carlos-HP-ProBook-430-G3:~/Escritorio/IRAC/Practica 3/Bento4-SDK-1-6-0-633.x86_64-unknown-linux/utls_opt$
```

Figura 33. Encriptación de los ficheros de vídeo mediante el comando mp4dash para Widevine

A parte de obtener los directorios anteriormente descritos, en el directorio principal se ha creado el siguiente fichero MPD, el cual será muy necesario para que el usuario pueda reproducir el video en las diferentes calidades:

```

-<MPD mediaPresentationDuration="PT10M23.583S" minBufferTime="PT2.00S" profiles="urn:mpeg:dash:profile:isoff-live:2011" type="static">
  <!--
    Created with Bento4 mp4-dash.py, VERSION=2.0.0-633
  -->
  <!-- Video -->
  <AdaptationSet maxHeight="720" maxWidth="1280" mimeType="video/mp4" segmentAlignment="true" startWithSAP="1">
    <!-- MPEG Common Encryption -->
    <ContentProtection cenc:default_KID="9eb4050d-e44b-4802-932e-27d75083e266" schemeIdUri="urn:mpeg:dash:mp4protection:2011"
      value="cenc"/>
    <!-- Widevine -->
    <ContentProtection schemeIdUri="urn:uuid:edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
      <cenc:pssh>
        AAAAOHBzc2gAAAAA7e+LqXnWSs6jyCfc1R0h7QAAABgSEJ60BQ3kS0gCky4n11CD4mZI49yVmwY=
      </cenc:pssh>
    </ContentProtection>
    <SegmentTemplate duration="2000" initialization="$RepresentationID$/init.mp4" media="$RepresentationID$/seg-$Number$.m4s"
      startNumber="1" timescale="1000"/>
    <Representation bandwidth="175813" codecs="avc1.4D401E" frameRate="24" height="90" id="video/avc1/1" scanType="progressive"
      width="160"/>
    <Representation bandwidth="1284284" codecs="avc1.4D401E" frameRate="24" height="360" id="video/avc1/2" scanType="progressive"
      width="640"/>
    <Representation bandwidth="4012985" codecs="avc1.4D401E" frameRate="24" height="720" id="video/avc1/3" scanType="progressive"
      width="1280"/>
  </AdaptationSet>
</Period>
</MPD>

```

Figura 34. Generación del fichero MPD con la codificación de los videos

En el fichero MPD, se puede observar el *AdaptionSet* del video original cuya resolución es de 720x1280. Lo siguiente que podemos ver es que el video se ha encriptado en modo común.

Por otro lado, si analizamos el *ContentProtection*, se puede apreciar que aparece el valor de la KID con la cual hemos encriptado los videos. Además, también podemos ver los valores cuando se ha encriptado con *Widevine*. En este caso, se puede apreciar la cadena de caracteres obtenida del PSSHBox, codificado en *BASE64*.

Finalmente, tenemos la parte del *SegmentTemplate* en la que destaca que se tienen tres representaciones, una para cada calidad (con distintos anchos de banda, altura y anchura de cada imagen del video), con el códec "avc1.4D401E" y un escaneo de tipo progresivo.

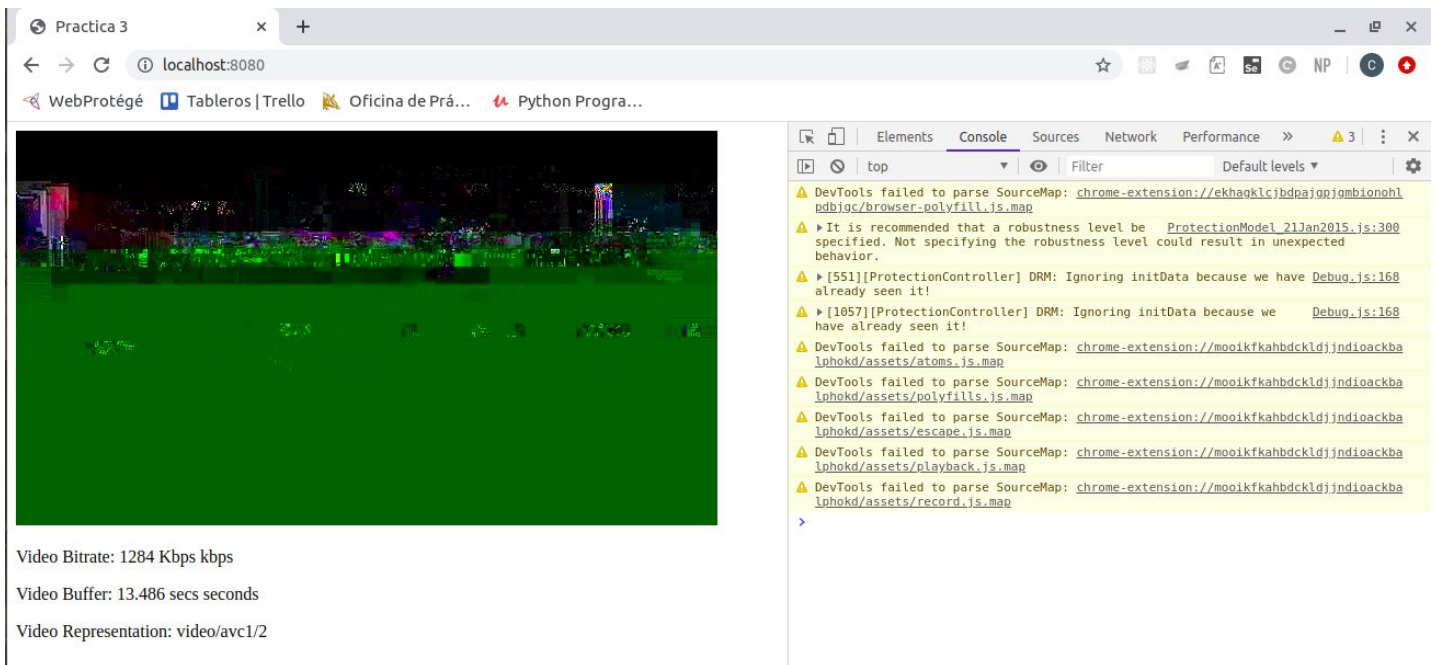
Para poder realizar la visualización del video correctamente, es necesario modificar el fichero *index.html* para que se pueda acceder al servidor que nos ofrezca las claves de encriptado. También, como hemos dicho antes será necesario poner la cabecera HTTP para poder obtener la licencia de Widevine. De esta manera, la estructura de datos mencionada y relativa a la protección DRM se realiza mediante el siguiente código de JAVASCRIPT:

[illegible]

Figura 35. Código JAVASCRIPT correspondiente a la protección DRM con Widevine

En la figura anterior, se puede observar la dirección URL a la cual el servidor va a acceder para poder obtener las claves de encriptado para así poder descryptar el video para su posterior visualización. También, se puede ver la cabecera HTTP que se corresponde al X-AxDRM-Message. El código JAVASCRIPT necesario para la correcta visualización del video, no se ve modificado.

Por último, pasamos a ver la visualización del video por parte del cliente. Antes de realizar la visualización es importante destacar que, Widevine solo funciona con aplicaciones de Google, por lo que ésta visualización se va a realizar en el navegador *Google Chrome*. El resultado obtenido es el siguiente:



*Figura 36. Visualización del vídeo en el lado del cliente, empleando el modelo de Widevine*

En la imagen se puede apreciar que, estamos en el lado del cliente ya que accedemos para poder visualizar el video mediante el url <https://localhost:8080/>



Por otro lado, se puede apreciar que, en la parte de consola, no obtenemos ningún error en cuanto a que el video no se pueda reproducir correctamente, por lo que el servidor accede correctamente al servidor de licencias de *Widevine* que nos proporciona las claves para su posterior descryptado. Sin embargo, el video no se visualiza correctamente, ya que se puede ver como la pantalla se de color verde. Esto puede deberse a que la descryptación por parte del servidor no se realiza correctamente, a pesar de no obtener ningún error en la consola, dando lugar a la representación que vemos en pantalla.

Otro parámetro que nos indica que el video funciona, a pesar de que la descodificación no se realiza correctamente, es que debajo de la pantalla de visualización, se puede apreciar que hay datos que nos indican el *Bitrate*, el *Buffer* y la representación a la que se encuentra ese *frame* en ese instante de tiempo. El *Bitrate* presenta un valor de 1284 kbps, el nivel de almacenamiento del *Buffer* es de 13.484 segundos y la representación del video es de calidad media, esto es, si éste lo estuviéramos viendo correctamente, la calidad con la que visualizamos las imágenes sería con la calidad intermedia, ya que como podemos observar nos indica que se está accediendo a la representación que se encuentra en el directorio `video/avc1/2`.

## 6. NOTA ADICIONAL

Durante la realización de la práctica, al pedirnos la realización de un fichero *index.html* por cada punto que se ha realizado (Introducción al Dash, Generación de videos para Streaming, Gestión DRM con Dash.js y Parte opcional), ha sido necesario crear 4 ficheros *html*. Por consiguiente, se han creado 4 directorios *output*, los cuales contienen cada fichero *index.html*, los ficheros MPD generados y a su vez, contiene un directorio por cada uno de los videos segmentados (el directorio número 1 para el video de calidad baja, el directorio número 2 para el video de calidad media y el directorio número 3 para el video de calidad alta). Para que se pueda tener acceso al código completo, así como a los videos que se han ido creando, se proporcionará un enlace a GitHub donde se encuentra el código de la práctica realizada:

<https://github.com/carlosaznarol/P3-IRAC-G22.git>

Cabe aclarar que, al realizar la práctica en un ordenador con sistema operativo Ubuntu 18.04, si accedeis a dicho enlace para ver el código, las diferentes carpetas *output* se encuentra dentro del directorio: "*Bento4-SDK-1-6-0-633.x86\_64-unknown-linux*". Estas a su vez se encontraron dentro de la carpeta *utils* (se encuentra los ficheros *index.html* y *MPD*, así como las carpetas con los ficheros de video del apartado 3 del documento), *utils\_copia* (se encuentra los ficheros *index.html* y *MPD*, así como las carpetas con los ficheros de video del apartado 4 del documento) y *utils\_widevine* (se encuentra los ficheros *index.html* y *MPD*, así como las carpetas con los ficheros de video del apartado 5 del documento).



## 7. REFERENCIAS

- [1] <https://javierortiz.mx/glosario/protocolos-streaming-video/protocolo-mpeg-dash/>
- [2] <https://upcommons.upc.edu/bitstream/handle/2099.1/16894/84799.pdf>
- [3] <https://enbeeone3.com/install-mp4box-ubuntu-16-04/>
- [4] <https://docs.microsoft.com/es-es/azure/media-services/previous/media-services-protect-with-playready-widevine>
- [5] <https://github.com/Axinom/public-test-vectors>
- [6] <https://tools.axinom.com/generators/PsshBox>
- [7] <https://tools.axinom.com/decoders/PsshBox>