UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Plataforma web de grabación de vídeo digital basada en DASH

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Jaramillo Perez, Giovanny

Tutor/a: Belda Ortega, Román

CURSO ACADÉMICO: 2022/2023

Resumen

El objetivo de este trabajo final de grado consiste en desarrollar una plataforma web que permita la gestión de múltiples cámaras, reproducción en tiempo real de las mismas mediante el protocolo DASH-LL y visualización de grabaciones de video ya realizadas.

Para llevar a cabo la plataforma se tendrá en cuenta tres servicios: una interfaz web que permite la gestión de las cámaras, un servidor de base de datos para el almacenamiento persistente y compartición de la información, y un servidor encargado de recodificar el video. El diseño de la interfaz será diseñado de tal manera que permita a los usuarios navegar entre las vistas de manera cómoda y sencilla, reduciendo la sobrecarga de funciones en la aplicación web.

Se pretende construir una herramienta útil e innovadora que permita reproducir videos en tiempo real adaptando las calidades de la cámara según las condiciones de red. Esto permitirá que, si un usuario no dispone de una conexión estable, el video en tiempo real se seguirá reproduciendo.

Palabras clave: Servicios Web, Cámaras IP, DASH, Baja latencia, FFmpeg

Abstract

The objective of this final degree project is to develop a web platform that allows the management of multiple cameras, real-time playback of them using the DASH-LL protocol, and viewing of video recordings already made.

To carry out the platform, three services will be taken into account: a web interface that allows the management of the cameras, a database server for persistent storage and sharing of information, and a server in charge of recoding the video. The interface design will be designed in such a way that it allows users to navigate between the views comfortably and easily, reducing the overload of functions in the web application.

It is intended to build a useful and innovative tool that allows to play videos in real time adapting the qualities of the camera according to the network conditions. This will allow that, if a user does not have a stable connection, the video in real time will continue playing.

Keywords: Web Services, IP Cameras, DASH, Low Latency, FFmpeg

Índice general

Apéndices

Índice de figuras

Índice de tablas

CAPÍTULO I

Introducción

Este capítulo presenta las razones que han impulsado la realización de este proyecto, establece las metas que se aspiran alcanzar y concluye con un listado de las diferentes secciones que forman parte de este documento.

1. 1. Motivación

En la actualidad, la implementación de cámaras IP ha elevado la seguridad en diversos negocios y entornos. No obstante, estas cámaras normalmente soportan un único protocolo, RTSP (Real Time Streaming Protocol), el cual presenta desafíos, como la incapacidad de adaptarse dinámicamente a las condiciones de la red y la alta demanda de ancho de banda. Por lo tanto, este proyecto busca abordar estos retos mediante una herramienta que permita ajustar la calidad de la transmisión en tiempo real de las cámaras, en función de las condiciones de la red, y que requiera menos ancho de banda.

1.2. Objetivos

El objetivo principal de este trabajo final de grado es desarrollar de manera exitosa una plataforma web (DVR-LL) que permita gestionar la configuración de las distintas cámaras IP, visualizar las cámaras en tiempo real (streaming) en diferentes calidades sobre HTTP, y permitir la reproducción de grabaciones realizadas. Este objetivo se desglosará en otros objetivos secundarios:

- Extender la funcionalidad de Fast-ll (servidor de Low-Latency DASH desarrollado por el COMM) para que permita almacenar los segmentos de DASH que genera.
- Obtener la información necesaria a partir de una base de datos compartida para el manejo de datos desde los diferentes servicios, la plataforma web (DVR-LL) y el servidor DASH de baja latencia (Fast-ll).
- Diseñar una interfaz web para la configuración de los orígenes de vídeo, el acceso al contenido almacenado y la comunicación con los diferentes servicios ofertados por Fast-ll.
- Entendimiento del protocolo DASH[1] y Low Latency DASH[2], donde el segundo busca reducir la latencia que introduce el primero.

En otras palabras, al alcanzar estos objetivos, se debería obtener una plataforma web en la que, una vez configurada la cámara, el usuario pueda ver en tiempo real su transmisión en diversas calidades y también revisar las grabaciones de las emisiones realizadas previamente.

1.3. Impacto esperado

Mediante el desarrollo de esta interfaz web, se aspira a ofrecer al usuario una herramienta que facilite la administración de diversas cámaras IP, permitiendo la visualización en tiempo real ajustable a distintas calidades sobre HTTP y la revisión de grabaciones. Esto simplificará a los usuarios la gestión de un amplio número de cámaras IP y el control individual de cada una.

En lo que respecta a la red, buscamos resolver los inconvenientes del protocolo RTSP, como la adaptación a diferentes calidades según las condiciones de la red y el alto requerimiento de ancho de banda [3].

1.4. Estructura de la memoria

La memoria de este proyecto se encuentra estructurada la siguiente manera:

- **Introducción:** Declaración de la motivación para la realización del trabajo y los objetivos a llevar a cabo para el desarrollo del proyecto.
- **Estado del arte:** Análisis y comparación de aplicaciones que ofrezcan características parecidas a las que se planea implementar en el desarrollo del trabajo.
- **Análisis del problema:** Se evalúan las expectativas respecto a la aplicación y, con base en estas, se presenta una solución adecuada. Además, se identifican las fases que atravesará el desarrollo, los potenciales riesgos, así como las oportunidades para innovar y generar negocio.
- **Diseño de la solución:** Se determinan los diversos componentes que conforman la plataforma web para luego realizar una descripción detallada de su funcionamiento y la interacción entre estos. Además, se listan las tecnologías que se utilizarán en el desarrollo de la solución.
- **Desarrollo de la solución:** Se detalla la implementación de los diversos componentes identificados en el punto previo.
- **Pruebas y despliegue:** Se presenta un listado de las pruebas más significativas realizadas para verificar el rendimiento de la aplicación, y se proporciona una guía para su despliegue, demostrando también su funcionamiento.

- **Conclusiones:** Resumen del trabajo realizado y correlación entre los resultados logrados y las metas establecidas al comienzo del proyecto.

A estos capítulos se le suma un anexo adicional que ofrece información adicional:

- **Glosario:** Definición de palabras específicas y acrónimos de la materia.

CAPÍTULO ll

Estado del arte

Antes de iniciar el desarrollo de la aplicación web se llevó a cabo un estudio para entender las características de otras soluciones existentes que ofrecen funcionalidades similares a las que este proyecto busca incorporar. Este capítulo detalla dicho análisis y concluye con una reflexión en la que se destacan las diferencias entre las soluciones analizadas y la que se pretende crear.

2.1. Soluciones similares

En el avance tecnológico actual, encontramos una variedad de soluciones software diseñadas para visualizar y gestionar cámaras IP utilizando el protocolo RTSP. En el marco de este trabajo de fin de grado, es relevante profundizar en la exploración y comprensión de las soluciones existentes en el mercado para identificar las fortalezas y debilidades que pueden influir en el desarrollo de este proyecto. En este apartado, se enfocará principalmente en dos de las soluciones más destacadas y utilizadas en la actualidad: VLC Media Player y Agent DVR de iSpy Connect.

Agent DVR

ISpy Connect es una empresa especializada en el desarrollo de software de vigilancia, ofreciendo un amplio abanico de características que van más allá de la simple visualización de video. Su producto más reconocido, iSpy, permite configurar y administrar múltiples cámaras IP, integrando funciones como grabación, detección de movimiento, alertas y acceso remoto. La compatibilidad con RTSP y el enfoque en seguridad que ofrece la empresa son características a tener en cuenta por los clientes.

Sin embargo, a partir de enero de 2022, iSpy ha sido reemplazado por Agent DVR [4], su nueva plataforma. Esta actualización incluye todas las funciones desarrolladas para iSpy, pero con una serie de mejoras significativas [5] como las siguientes:

- **Compatibilidad de plataforma:** A diferencia de iSpy, que se ejecuta únicamente en Windows, Agent DVR es compatible con diferentes entornos (Windows, Linux, Docker, Linux, MacOS, Raspberry Pi 4+).
- **Soporte para IA:** Soporte para características de inteligencia artificial, como la detección de objetos y personas, análisis de video y reconocimiento de sonido.

- **Ejecución a modo de servicio:** Esta nueva actualización se ejecuta como un servicio y tiene acceso a más recursos del sistema. Incluyendo en este acceso a través de cualquier navegador moderno y número ilimitado de cámaras y micrófonos.

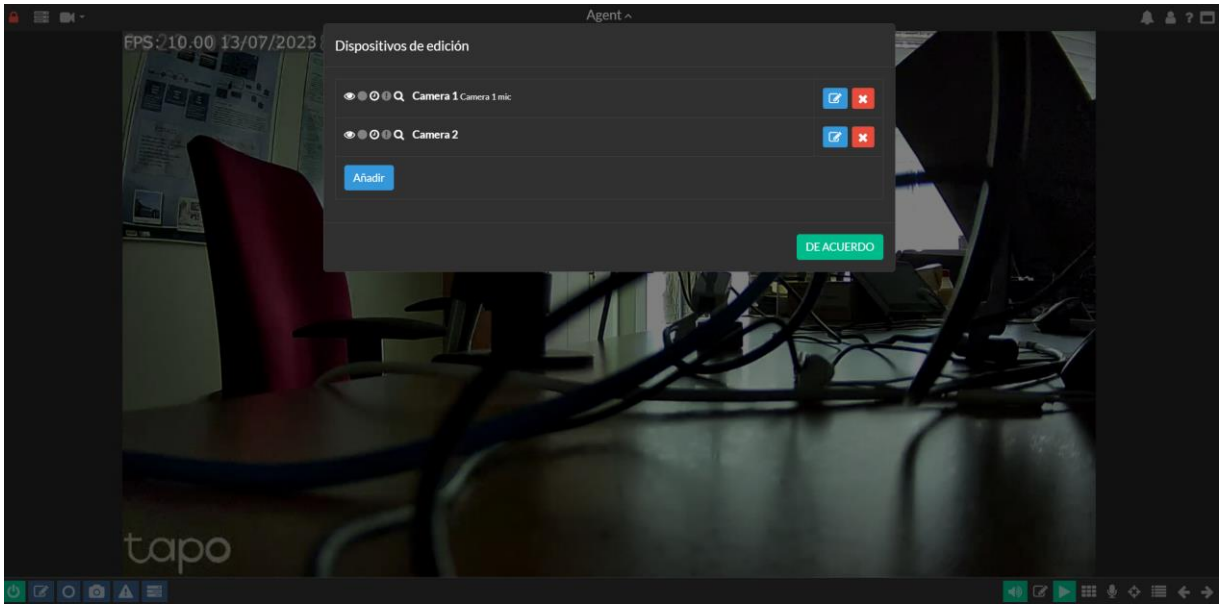

Figura 2.1: Panel de gestión de dispositivos de Agent DVR

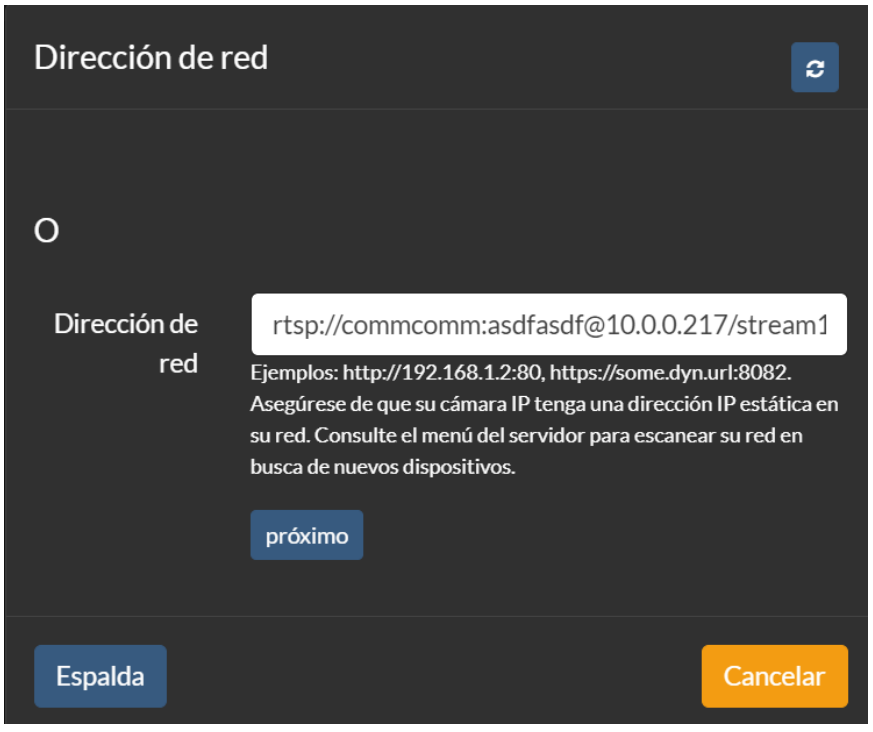

Figura 2.2: Inclusión de cámara IP en Agent DVR

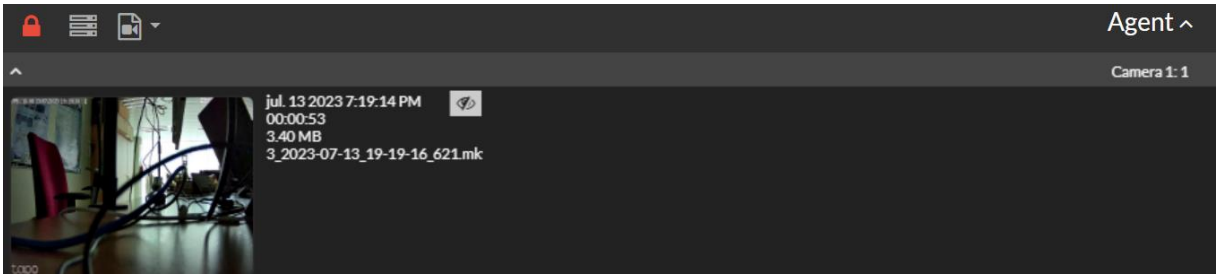

Figura 2.3: Obtención de grabaciones realizadas utilizando Agent DVR

Por lo tanto, Agent DVR se presenta como un servicio web altamente potente para la reproducción de video en streaming y la grabación de varias cámaras IP, sumando a esto una extensa lista de características avanzadas. Sin embargo, su uso exclusivo del protocolo RTSP significa que solo se puede seleccionar una configuración de calidad para el streaming. Esto implica que, si las condiciones de la red cambian, la calidad del streaming no se ajustará dinámicamente a estas nuevas condiciones, y no es posible reducir el ancho de banda utilizado.

VLC media player

VideoLAN (s.f.) define su producto (VLC Media Player) como “un reproductor multimedia libre y de código abierto multiplataforma y un «framework» que reproduce la mayoría de archivos multimedia, así como DVD, Audio CD, VCD y diversos protocolos de transmisión” [6]. Entre los protocolos soportados por VLC Media Player se encuentra el protocolo RTSP (Real Time Streaming Protocol), lo que lo convierte en una opción popular para la visualización de cámaras IP. Este software también tiene capacidades de transmisión en vivo y puede actuar como un servidor para enviar transmisiones de medios a través de la red.

VLC Media Player es notablemente versátil, siendo capaz de manejar una gran variedad de formatos de archivos de audio y video, lo que elimina la necesidad de descargar programas adicionales capaces de codificar o decodificar los datos digitales. Esta amplia compatibilidad con los formatos de medios convierte a VLC en una herramienta esencial para cualquier usuario que trabaje con contenidos multimedia.

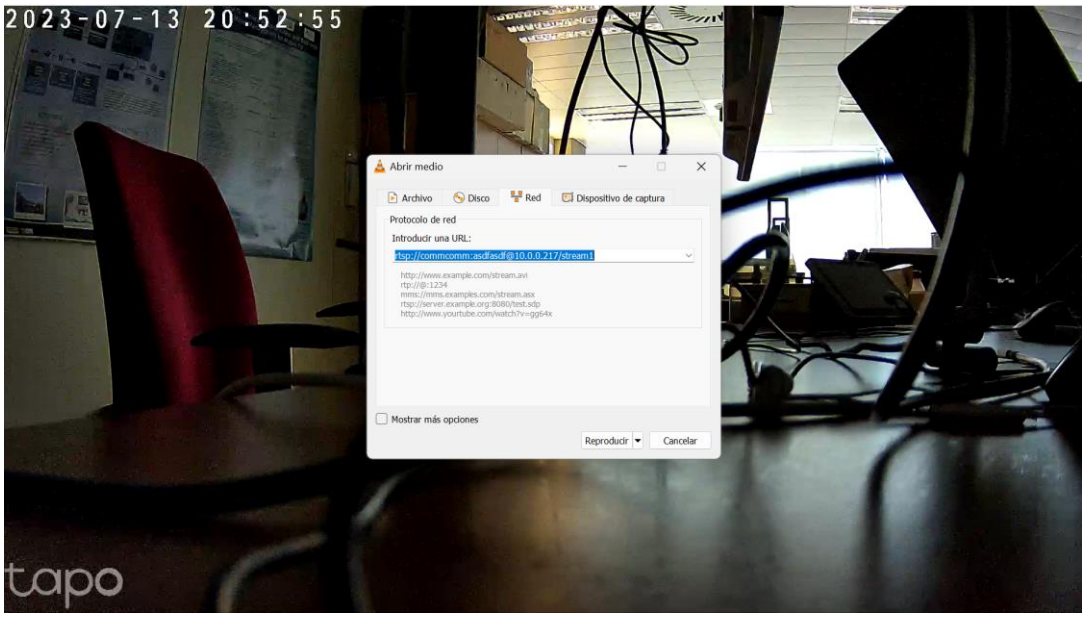

Figura 2.4: Reproducción de cámara IP mediante protocolo RTSP en VLC Media Player

Pese a las ventajas de VLC Media Player, tiene una limitación significativa en relación con este proyecto: la falta de soporte para el protocolo DASH. Este protocolo es fundamental para posibilitar la adaptación dinámica de la resolución de video acorde a las condiciones de la red, un atributo esencial que este proyecto se propone implementar. Por tanto, si bien VLC es un recurso formidable, no cumple con las exigencias que este proyecto pretende satisfacer.

2.2. Propuesta

A partir del análisis de las soluciones existentes, como Agent DVR y VLC Media Player se puede concluir que hay diversas opciones para visualizar y gestionar cámaras IP a través del protocolo RTSP. Las características ofrecidas por estos softwares, como la detección de movimiento, grabación de video, acceso remoto y capacidad de manejo de múltiples cámaras; son imprescindibles para garantizar la seguridad y la supervisión en tiempo real de los entornos.

Sin embargo, estos sistemas presentan una limitación significativa: la falta de adaptación dinámica a las condiciones de la red y el consumo elevado de ancho de banda, aspectos que el protocolo RTSP no puede abordar de manera eficiente. Este proyecto tiene como objetivo superar esta limitación al introducir la compatibilidad con el protocolo DASH, que permite ajustar dinámicamente la calidad de la transmisión en función de las condiciones de la red, ahorrando así en el uso del ancho de banda.

Mientras estas soluciones ya existentes proporcionan funcionalidades avanzadas, requieren un grado de conocimiento técnico para su configuración y uso óptimo. Otro objetivo del proyecto es desarrollar una interfaz web intuitiva y de fácil uso, que permita a los usuarios gestionar y visualizar las cámaras IP de manera sencilla, independientemente de su experiencia técnica.

En resumen, este proyecto se plantea como una mejora innovadora a las soluciones actuales de visualización y gestión de cámaras IP, al introducir un manejo más eficiente del ancho de banda y una interfaz de usuario más accesible.

CAPÍTULO III

Análisis del problema

En este capítulo se llevará a cabo un análisis del problema con el objetivo de proponer la solución más adecuada que englobe las características y funcionalidades previstas para la aplicación. Posteriormente, se presentará el plan de trabajo que se seguirá para asegurar un desarrollo exitoso del proyecto. Por último, se realizará una evaluación de los posibles riesgos, así como se destacarán aquellas áreas donde existen posibilidades de innovación y generación de valor en el mercado.

3.1. Análisis

Conforme a lo analizado en el capítulo previo, existen numerosas soluciones que ofrecen servicios asociados a la visualización y grabación de transmisiones de video de cámaras IP. En línea con estas soluciones, la aplicación que se pretende desarrollar busca proporcionar un servicio que permita la visualización en tiempo real de múltiples flujos adaptándolos a las condiciones de la red y la visualización de grabaciones de streamings realizados con anterioridad. Con esto en consideración, podríamos reflexionar sobre una amplia gama de alternativas:

- Podríamos enfocarnos en desarrollar una aplicación web, ofreciendo un acceso fácil y universal desde cualquier dispositivo con conexión a internet.
- Optar por desarrollar una aplicación móvil, lo que brindaría portabilidad y acceso en cualquier momento y lugar.
- Explorar distintas técnicas y formatos para la visualización de video, incluyendo la optimización para distintas condiciones de red y dispositivos de visualización.
- La posibilidad de incluir características adicionales, como la detección de movimiento y alertas, también podría considerarse.
- Incluso la capacidad de manipular la propia cámara IP, como el ángulo de visión y la rotación.

No obstante, este proyecto establece ciertos requisitos que limitarán estas opciones:

- La solución debe gestionar (añadir, modificar y eliminar) múltiples cámaras IP.
- La solución debe ser capaz de adaptarse dinámicamente a las condiciones de la red, ajustando la resolución del video en streaming de manera eficiente si se desea.

- La solución propuesta debe poder capturar y almacenar transmisiones en vivo, así como permitir el acceso y la revisión de dichas grabaciones.
- La solución debe ser fácilmente accesible y sencilla de usar para una amplia gama de usuarios.

Gestión de cámaras

Para gestionar, visualizar y reproducir grabaciones de diversas cámaras IP en la plataforma web que se está considerando sería necesario analizar la integración de una base de datos. La base de datos tendría la función esencial de almacenar la información necesaria para operar el servicio web.

Además, esta información almacenada también sería crucial para el servidor que procesa el video. El servidor estaría a cargo de recodificar el video, una operación que necesita un acceso rápido y fluido a los datos relevantes. Por tanto, es vital analizar cómo esta base de datos interactuaría de manera eficiente con el servicio de procesamiento de video.

En el siguiente apartado, *Solución propuesta*, se detalla la opción escogida para el procesamiento de la información entre ambos servicios.

Adaptación dinámica según las condiciones de red

En el contexto de la reproducción de video en tiempo real desde varias cámaras IP, la adaptación dinámica de la calidad de video según las condiciones de la red es un factor clave que se necesita considerar. Tal adaptación podría permitir la visualización eficiente de flujos de video en tiempo real, independientemente de las variaciones en la calidad de la red.

Una consideración importante en este aspecto es la integración de un protocolo que permita tal adaptación. Un ejemplo de un protocolo que permite esto es DASH (Dynamic Adaptive Streaming over HTTP). Sin embargo, hay que realizar un análisis en profundidad sobre su integración y los posibles desafíos que podría presentar.

Es importante notar que esta adaptación dinámica también debe considerar cómo maneja la calidad de la transmisión en una amplia gama de escenarios de uso. Por ejemplo, si un usuario tiene una conexión a Internet lenta, el sistema debería poder adaptarse y entregar una calidad de video utilizable.

Este análisis de adaptación dinámica de la calidad de video según las condiciones de la red proporciona una visión inicial de los desafíos que deben superarse para lograr un funcionamiento eficaz.

Esta cuestión se examinará en mayor detalle en los capítulos "*Diseño de la solución*" y "*Desarrollo de la solución*" donde se analizarán más a fondo las técnicas y estrategias necesarias para implementar esta característica.

Captura y almacenamiento en tiempo real

La captura y el almacenamiento en tiempo real de los streams de video de múltiples cámaras IP es otro aspecto crucial a tener en cuenta en el análisis del problema. Este requisito implica la necesidad de implementar una solución que no sólo pueda acceder y visualizar estos streams de video en tiempo real, sino que también tenga la capacidad de grabar y almacenar estos datos para su posterior análisis o revisión.

Es esencial que se cuente con un sistema de almacenamiento eficaz y seguro que pueda manejar la cantidad de datos que se generarán a partir de múltiples flujos de video. Además, será importante garantizar que este sistema de almacenamiento pueda funcionar en tiempo real para que no haya retrasos significativos en la grabación y visualización de los videos.

Una solución podría implicar el uso de bases de datos especializadas para almacenar video, como por ejemplo MongoDB permitiendo el manejo de archivos binarios grandes, otra opción sería usar soluciones de almacenamiento en la nube o en disco desde un sistema de archivos. Cada opción tiene sus propios beneficios y desafíos que necesitan ser considerados, incluyendo factores como coste, velocidad, seguridad y facilidad de acceso a los datos almacenados.

Este análisis inicial de la captura y almacenamiento en tiempo real proporciona un marco de trabajo para comprender los retos a los que se enfrenta este proyecto. Estos temas serán explorados en mayor profundidad en los capítulos "*Solución propuesta*" y "*Desarrollo de la solución*".

Interfaz de usuario accesible

La accesibilidad de los usuarios a través de una interfaz de usuario fácil de usar es un componente esencial en este análisis. Para optimizar la interacción con los flujos de video de las múltiples cámaras IP se considera desarrollar una aplicación móvil o una plataforma web.

La interfaz de usuario deberá ser diseñada para ser intuitiva y fácil de navegar. Esto implica que los usuarios deberán poder acceder fácilmente a los flujos de video, navegar entre las diferentes cámaras y ajustar cualquier parámetro de visualización de manera sencilla.

La elección entre una aplicación móvil y una plataforma web se hará en función de diversos factores. Una aplicación móvil podría ofrecer mayor movilidad y acceso en cualquier momento y lugar, mientras que una plataforma web podría ser más accesible para los usuarios que prefieren utilizar ordenadores o que no desean descargar una aplicación adicional.

Asimismo, la plataforma web debe ser compatible y operativa tanto en ordenadores como en dispositivos móviles [7], lo que puede requerir una atención especial al diseño y desarrollo del sitio web.

A través de este análisis, se han considerado cuatro aspectos principales para el desarrollo del servicio de visualización en tiempo real de video en streaming. Primero, se reconoció la necesidad de una plataforma web robusta con un sistema de base de datos para administrar, visualizar y reproducir grabaciones. En segundo lugar, se destacó la importancia de la adaptación dinámica de la resolución del video a las condiciones de red. Luego, abordamos el requisito de capturar y almacenar los flujos de video en tiempo real para garantizar un acceso eficaz a los datos. Finalmente, se identificó el requisito de una interfaz de usuario intuitiva y accesible a través de una plataforma web o una aplicación para dispositivos móviles, enfatizando su importancia para permitir la utilización de esta en un rango mayor de usuarios. Este análisis ha permitido una definición más clara de las metas del proyecto y establece un camino hacia el diseño y desarrollo futuro.

3.2. Solución propuesta

3.2.1. Funcionamiento general

El proyecto consistirá en una aplicación web con una vista donde se describa brevemente el software que se utilizará. En la esquina superior derecha los usuarios encontrarán un botón para registrarse o iniciar sesión. De esta forma, se permite la asociación de distintas cámaras IP a cada usuario registrado, gracias a una base de datos compartida.

Tras el inicio de sesión, los usuarios serán dirigidos a una lista de cámaras asociadas a su cuenta. En esta interfaz tendrán completo control para la administración de las cámaras, incluyendo funcionalidades para agregar, modificar, eliminar o visualizar en tiempo real el video de las cámaras.

El proceso de añadir una cámara presentará varias opciones de configuración, como la asignación de un nombre, la selección del tipo de cámara, la especificación de la dirección URL del flujo RTSP para localizar la cámara, la calidad de la transmisión y la tasa de bits, entre otras opciones.

Además, una vez que la sesión se ha iniciado, se habilitarán nuevas opciones en la barra de navegación superior para facilitar la navegación por la aplicación web. Entre estas opciones se incluyen: 'home' (que redirige a la vista del listado de cámaras) y 'grabaciones' (que conduce a la vista de grabaciones realizadas en streams previos).

La opción 'grabaciones' desplegará una interfaz intuitiva donde los usuarios pueden seleccionar diferentes fechas. Si existen grabaciones en las fechas seleccionadas, se mostrará una lista de horas, destacando con un color diferente las horas en las que se han realizado grabaciones. Cuando el usuario elija la hora de la grabación a reproducir, podrá especificar los minutos, si lo desea. Tras introducir estas opciones, el usuario podrá pulsar un botón para iniciar la reproducción de la grabación.

Por otro lado, la reproducción del video en tiempo real y grabación de las cámaras será gracias al servidor de Fast-LL (servicio de DASH-LL). Dicho servidor será extendido y modificado para que se adapte al enfoque de la aplicación web. Además, para permitir la comunicación entre el servidor que recodifica el video y la aplicación web se implementará una serie de servicios para hacer que la comunicación se realice con éxito.

3.2.2. Compartición de la información entre los servicios

Es vital destacar la importancia de compartir información entre los servicios, ya que este factor tiene un impacto significativo en la integración, seguridad y manipulación de los datos. En este sentido, es crucial tener en cuenta cómo se manejarán estos aspectos.

Inicialmente, se contempló la posibilidad de integrar el servicio de Fast-LL directamente en la aplicación web, manipulando así la información en una base de datos local. Sin embargo, tras analizarlo, se determinó que esta no era la solución óptima.

Por lo tanto, la expectativa es que la información se comparta entre los diferentes servicios a través de una base de datos compartida. Se proporcionarán más detalles sobre esta decisión en el capítulo "*Diseño de la solución y Desarrollo de la solución"*.

3.2.3. Protocolo DASH y DASH-LL

Es fundamental diferenciar los protocolos DASH y DASH-LL. Si bien ambos se concibieron para adaptar dinámicamente la calidad del video, el DASH-LL incorpora una característica adicional para minimizar la latencia. En otras palabras, aunque ambos protocolos comparten la tarea de ajustar la calidad del video de acuerdo con las condiciones de la red, DASH-LL pone un énfasis especial en asegurar una transmisión con la mínima demora, lo cual resulta esencial para la difusión en tiempo real de los flujos de video.

Además, es importante resaltar que DASH se concibió originalmente para la transmisión de video bajo demanda. Sin embargo, debido a su gran popularidad y a las ventajas significativas que proporciona para la difusión de contenido en internet, se introdujo DASH-LL. Esta variante incorpora los beneficios de DASH a los servicios de baja latencia, facilitando así su aplicación en transmisiones en tiempo real como la televisión en vivo o el streaming en directo.

Otra característica importante a mencionar es que tanto DASH como DASH-LL funcionan a través del protocolo HTTP. Esto simplifica notablemente su integración con las CDN, dado que estas redes están diseñadas para manejar y optimizar el tráfico HTTP [8].

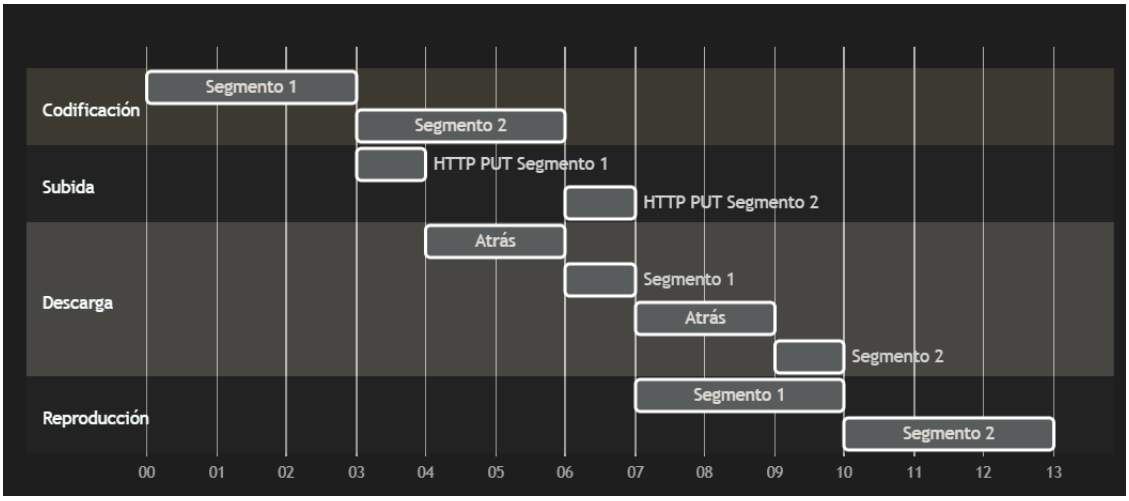

Figura 3.1: Diagrama de tiempo DASH estándar

Existen diferencias específicas entre DASH y su versión de baja latencia, DASH-LL, que están orientadas a la reducción de la latencia. Entre ellas se incluyen:

- La necesidad de que los segmentos sean recursos distintos.
- La posibilidad de que los segmentos contengan fragmentos para permitir la reproducción de segmentos parciales.
- La capacidad de los clientes para descargar los segmentos conforme estos son cargados en el servidor.
- La posibilidad de que la solicitud de un segmento llegue antes de que éste comience a cargarse al servidor.

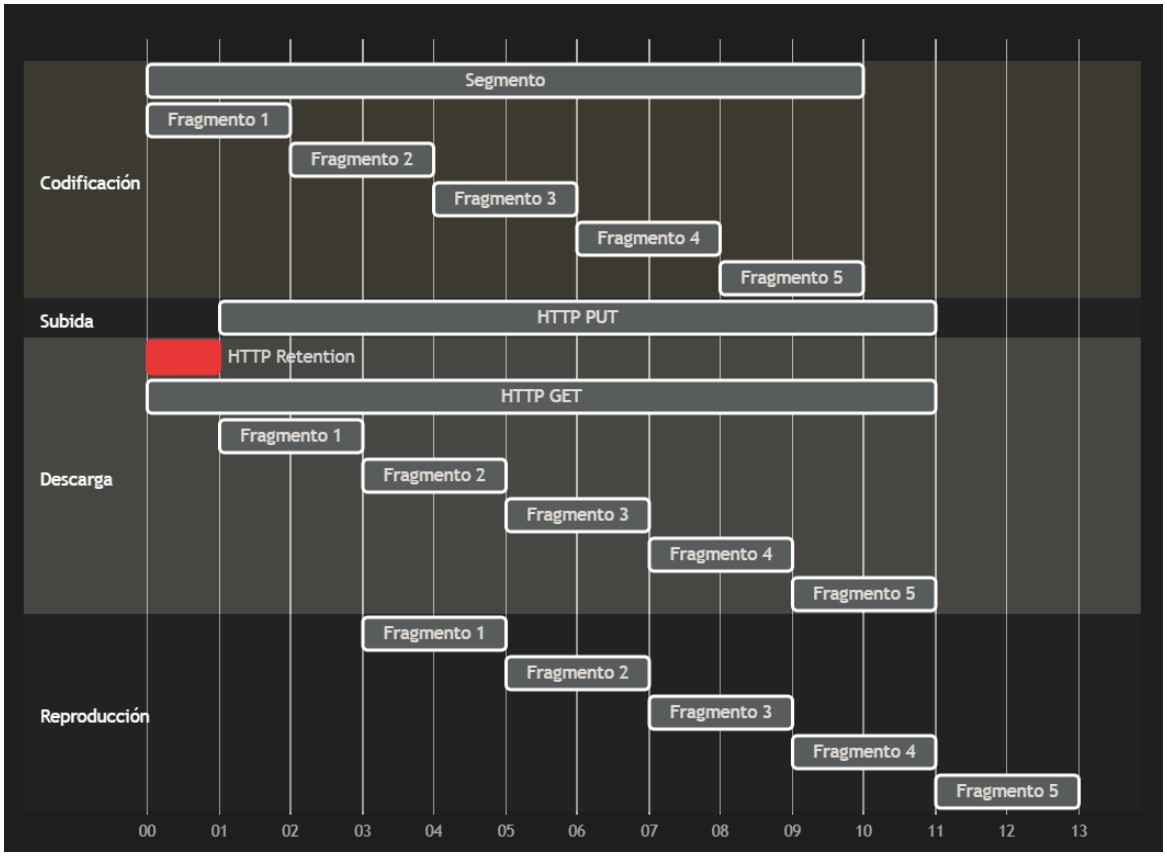

Figura 3.2: Diagrama de tiempo DASH-Low Latency

Hay detalles particulares a considerar en el funcionamiento del protocolo DASH. Un elemento clave es la necesidad de tener frames I (intra frames) al inicio de cada segmento, lo que implica una recodificación del flujo de video. Este proceso es fundamental para asegurar la correcta transmisión y visualización de las secuencias de video.

Por otra parte, hay que mencionar que hay cámaras que ajustan su tasa de frames en función de diferentes factores, como, por ejemplo, las condiciones lumínicas. Aunque el protocolo RTSP puede

gestionar esta variabilidad, DASH no tiene incorporada dicha flexibilidad. Sin embargo, la habilidad de DASH-LL para manejar de manera eficiente la latencia confirma su elección como el protocolo que se utilizará en este proyecto.

En resumen, ambos diagramas ilustran el proceso de transmisión de video, pero con distintas estrategias. El diagrama de la figura 3.1 refleja un método secuencial, donde cada segmento de video se codifica, carga, descarga y reproduce completamente antes de proceder al siguiente. En cambio, el diagrama de la figura 3.2 muestra una estrategia más fragmentada y concurrente. En este caso, un segmento de video se divide en múltiples fragmentos, los cuales se codifican y cargan simultáneamente. Estos fragmentos están disponibles para su descarga y reproducción mientras el resto del segmento continúa procesándose, lo que posibilita una reducción de la latencia y un flujo de reproducción más uniforme.

3.2.4. Recodificación del video (RTSP a DASH)

La transformación del video de formato RTSP a DASH es un componente clave en el proceso de transmisión de video y, para realizar esta operación, el software FFmpeg juega un papel importante en esta recodificación. FFmpeg es un software de código abierto que tiene una amplia gama de utilidades destacando especialmente en el ámbito multimedia por sus capacidades para transcodificar video, es decir, cambiar de un formato a otro [9].

En el contexto de nuestro proyecto, FFmpeg se utiliza para procesar y convertir el video obtenido a través del protocolo RTSP en el formato DASH [10]. Este proceso de transcodificación permite la adaptabilidad y versatilidad necesarias para garantizar la funcionalidad del sistema de transmisión.

3.3 Plan de trabajo

El progreso de este proyecto se ha trazado siguiendo el modelo en V del ciclo de vida del software, un método que favorece la detección temprana de posibles problemas. A diferencia del modelo en cascada, donde los defectos se detectan únicamente en las etapas finales durante las pruebas, el modelo en V implementa las pruebas en etapas más tempranas, permitiendo descubrir errores potenciales de manera rápida y eficiente.

Este enfoque proporciona una ventaja significativa, ya que no se espera hasta el final para introducir mejoras. A medida que se avanza en las distintas fases, se verifican continuamente para determinar si es adecuado proceder a la siguiente etapa o si hay que corregir errores identificados en las etapas previas [11].

Las acciones tomadas durante cada fase se describen a continuación, aunque se evitará entrar en detalles en este punto. Pues, se explicará más en profundidad con los siguientes capítulos.

Semana	Etapa
Semana 0	Planificación y análisis.
Semana 1	Diseño y evaluación.
Semana 2	
Semana 3	Implementación, pruebas y despliegue.
Semana 4	
Semana 5	
Semana 6	
Semana 7	
Semana 8	
Semana 9	
Semana 10	
Semana 11	Documentación
Semana 12	
Semana 13	

Tabla 3.1: Tabla de planificación semanal de tareas

- **Planificación:** esta fase se ha enfocado primordialmente a la organización temporal del proyecto donde se ha procedido con la estructuración de las tareas en diferentes semanas. Cabe destacar que esta temporalización es de carácter flexible, pudiéndose modificar o ajustar los tiempos establecidos inicialmente a la cumplimentación de las mismas. La repartición final de las tareas es la que se presenta en la tabla 3.3.
- **Análisis:** esta tarea se ha orientado al estudio de los protocolos a utilizar en el desarrollo del proyecto. RTSP, DASH y DASH-LL, relacionados con la grabación en tiempo real de vídeo en cámaras IP. Donde cada uno tiene ventajas y desventajas.
- **Diseño:** este punto se ha centrado en la elección de las tecnologías. Así, como la manera de comunicar los servicios (servicio web y servicio que recodifica el video en tiempo real) y la compartición de la información mediante una base de datos.

  Dado que el servicio web desea recibir datos de un servidor que recodifica el video, se ha optado por un diseño de cliente-servidor con arquitectura de tres capas [12]: la primera capa es el cliente (plataforma web), la segunda capa es el servidor que recodifica (Fast-LL) y la tercera es la base de datos compartida.
- **Evaluación:** se evalúa la etapa anterior, a partir de técnicas de prototipado de desarrollo centrado en el usuario. La técnica realizada para el diseño de la interfaz son los bocetos,

ya que el objetivo principal es recoger las ideas importantes de manera rápida y se utiliza en la etapa inicial del proyecto.

En lo que se refiere al código, en este punto ya se conseguido la interfaz de usuario, se ha utilizado la técnica del prototipado vertical. Esta técnica se ha utilizado debido a que se han implementado pocas características para la realización de pruebas, pero sus funcionalidades están totalmente implementadas.

- **Implementación:** para documentar la funcionalidad de la aplicación es necesario la implementación del código. Por ello, en esta fase se ha centrado en el desarrollo de aquellas funciones y servicios necesarios para que la utilidad de la plataforma cumpla con el objetivo inicial.
- **Pruebas:** es esencial llevar a cabo una serie de pruebas en cada componente de la plataforma para garantizar que funciona según las expectativas. El servicio de base de datos es un componente crucial en el proyecto, pues es el encargado de servir los datos. Por ello, será necesario verificar que la información se está almacenando como se desea y que sea accesible desde otro servicio, permitiendo la comunicación entre el cliente y el servidor que recodifica el video. En el cliente hay que comprobar que se gestionan (adición, modificación, eliminación y visualización) exitosamente las cámaras y de la forma esperada. En el servidor se revisará que se esté procesando los datos introducidos del cliente (URL de la cámara, calidad, entre otros) y que las grabaciones se encuentren correctamente localizadas en el sistema de ficheros.
- **Instalación y despliegue:** como ya se ha comentado anteriormente, se dispone de tres servicios: el servicio web (DVR-LL), el servidor que recodifica el video (Fast-LL) y la base de datos compartida. Por ello, es necesario disponer/instalar estos servicios para el correcto funcionamiento de la plataforma. Una vez se instalan estos se puede a proceder con el despliegue, permitiendo la interacción con la plataforma.
- **Documentación:** todo el proceso de análisis, investigación e implementación se encuentra desglosado a lo largo del proyecto, concluyendo en las semanas finales tras la completa elaboración de la aplicación.

3.4. Análisis de riesgos

La visualización y grabación de las distintas cámaras IP depende totalmente del servidor de video que recodifica. Por ello, hay que tener en cuenta una serie de posibles inconvenientes y las medidas para que estos no se produzcan. Dichos riesgos se exponen a continuación.

Almacenamiento en disco

El uso del servicio de grabación se realiza con un sistema de ficheros y por eso hay que considerar el almacenamiento del disco. Inicialmente, una cámara no puede suponer un gran problema para el almacenamiento en disco, pero cuando se habla sobre una multitud de cámaras la situación cambia. Ahora el almacenamiento juega un papel importante en las copias de seguridad.

- **Impacto sobre el proyecto:** al no haber un sistema automático que controle el tiempo de vida de las distintas grabaciones la ocupación del almacenamiento en disco se producirá con mayor velocidad, llegando a ocupar todo el espacio del disco y, como consiguiente, no se podrá realizar más grabaciones.
- **Medidas:** existen diversas maneras para evitar la ocupación total del almacenamiento del disco. Entre ellas se encuentra un servicio de almacenamiento en la nube, aumento de discos de almacenamiento o incluso la eliminación de manera manual de aquellos directorios/archivos no deseados.

Pérdida de los archivos de grabación

Como ya se mencionó anteriormente, las grabaciones se almacenan en el sistema de ficheros. Por tanto, en el momento que se produzca un borrado no deseado se puede producir una inconsistencia de los datos y obtener resultados no deseados.

- **Impacto sobre el proyecto:** una inconsistencia en los datos implica la dificultad de la obtención de la reproducción. Esto supone la imposibilidad de reproducir grabaciones que se desean.
- **Medidas:** contar con un sistema que sea redundante puede ser una buena opción, como es el caso de la matriz de discos RAID, que permiten una mayor integridad y tolerancia a fallos [13]. Con este tipo de discos se puede mantener la redundancia de los archivos y directorios en el sistema.

Dificultad en la representación de los datos

Desde un principio puede que los usuarios no estén muy acostumbrados a los términos utilizados en la plataforma, como por ejemplo dar de alta una cámara que cuenta con ajustes de calidad de video, los frames por segundo que se desean utilizar, entre otros. Estos son realmente importantes para que la plataforma pueda trabajar correctamente.

- **Impacto sobre el proyecto:** el uso de datos incorrectos, debido a la información poco clara por parte de la interfaz, puede suponer que el usuario no pueda visualizar en tiempo real ni reproducir grabaciones de diferentes cámaras.
- **Medidas:** exponer de la manera más clara posible la información al usuario para que no tenga problemas en entenderla. Otra solución puede ser el establecimiento de valores por defecto en la configuración de las cámaras IP. De esta forma, el usuario solo tendría que rellenar aquellos datos menos técnicos.

Navegación compleja en la interfaz de usuario

Algunas de las plataformas web utilizadas para la visualización de cámara en tiempo real cuentan con numerosas funciones y, por consiguiente, la interfaz se vuelve compleja para los usuarios. Esto puede resultar vital para aquellos usuarios experimentados con la terminología, pero para aquellos que no lo están puede resultar frustrante.

- **Impacto sobre el proyecto:** si el usuario se siente frustrado debido a la cantidad de funciones disponibles en la interfaz, este puede terminar abandonando la plataforma.
- **Medidas:** disponer de una interfaz clara y sencilla, permitiendo al usuario una experiencia de navegación cómoda. En la misma interfaz se podría añadir un apartado para aquellos usuarios más familiarizados con la materia.

3.5. Oportunidades de negocio e innovación

El mercado actual de cámaras IP se fundamenta principalmente en el protocolo RTSP, un estándar reconocido para la emisión de video y audio en tiempo real. Sin embargo, se presenta una oportunidad debido a la falta de cámaras que soporten el protocolo DASH (Dynamic Adaptive Streaming over HTTP).

DASH es un protocolo de transmisión de video adaptable que ajusta la calidad del video en función de las condiciones de la red. Esto posibilita una entrega de video más eficaz y uniforme, especialmente en redes que pueden tener condiciones que no son estables.

El propósito principal de este proyecto es resolver este problema mediante la recodificación del protocolo RTSP a DASH. Este proceso de recodificación permitiría a los usuarios capturar video de varias cámaras IP, con la calidad del video ajustándose dinámicamente según las condiciones de la red. Esto no solo incrementaría la eficiencia y la experiencia del usuario, sino que también podría crear nuevas oportunidades de negocio.

Por ejemplo, los proveedores de cámaras IP podrían ofrecer servicios de grabación de video mejorados y más eficientes. Los usuarios finales, especialmente aquellos en áreas con redes menos estables, se beneficiarían de una transmisión de video más consistente. Además, la recodificación de RTSP a DASH podría impulsar futuras innovaciones en la transmisión de video y en la tecnología de las cámaras IP.

En definitiva, este proyecto no solo satisface una demanda presente en el mercado de cámaras IP, sino que también proporciona una oportunidad para la innovación y la expansión del negocio.

CAPÍTULO IV

Diseño de la solución

Una vez analizada la problemática actual de las cámaras IP, en relación con la inexistencia de cámaras DASH, y realizada la solución propuesta, en este capítulo se pretende profundizar en el funcionamiento de la plataforma y la forma en la que se comunican los distintos servicios. Explicando paso a paso cuál ha sido la arquitectura del sistema elegida y detallando el diseño de cada componente.

Por otro lado, para cerrar el capítulo, se hablará sobre las tecnologías utilizadas para el desarrollo de la plataforma, incluyendo las características que nos ofrecen y la razón de su elección.

4.1. Arquitectura del sistema

El sistema cuenta con una arquitectura de cliente-servidor con un modelo de tres capas. Dicho modelo es esencial para la manipulación de la información en los diferentes servicios. Este está compuesto por tres capas, como se ha mencionado en el plan de trabajo en la etapa de diseño. La primera de ellas es la capa de cliente que será la encarga de gestionar las cámaras IP. Para dicha gestión se tendrá en cuenta el método CRUD (Create-Read-Update-Delete), que consiste en la creación, lectura, actualización y eliminación, en este contexto, de cámaras.

La segunda capa es el proceso de recodificación de los streams de vídeo. Este proceso será el encargado de obtener los datos relativos a las cámaras IP configuradas de la base de datos compartida, procesarlos y servir los flujos de vídeo a los clientes. Permitiendo, de esta manera, la reproducción en tiempo real y las respectivas grabaciones.

La tercera capa hace referencia al servidor de base de datos compartido. Esta capa es de vital importancia para poder manejar la información entre los servicios, pues será el núcleo para distribuir la información.

Para comprender en mayor profundidad el funcionamiento del sistema se va a mostrar un ejemplo de cómo se intercambian los datos. Dicho ejemplo se podrá visualizar en la figura 4.1.

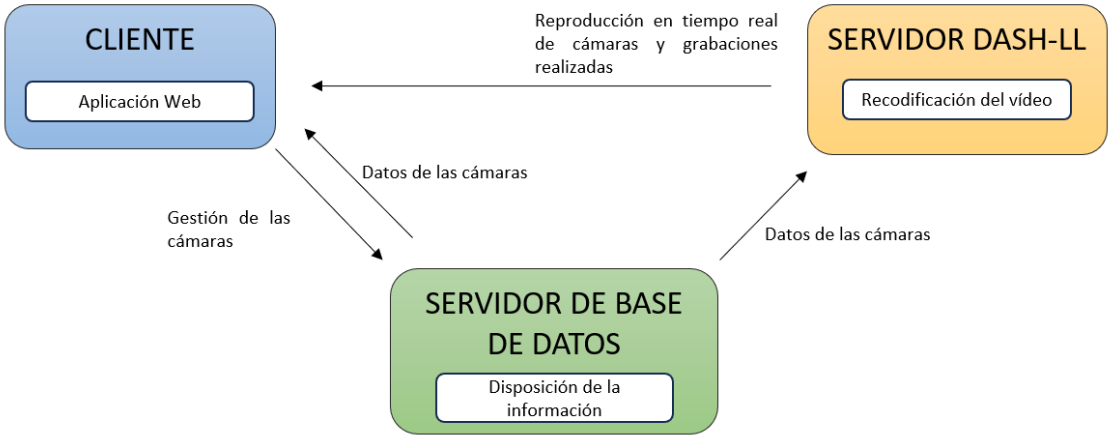

Figura 4.1: Funcionamiento general de la plataforma

Mediante este diseño, con el uso de una base de datos compartida, se encuentran una serie de ventajas a tener en cuenta:

- **Seguridad:** cada componente o servicio puede ejecutarse en una subred de la red. Por tanto, se pueden establecer políticas de control de acceso a los datos y al procesamiento de video. De este modo, se puede controlar quien puede acceder a la base de datos y quién puede solicitar que un video se recodifique.
- **Escalabilidad:** cada capa se podría escalar de forma independiente según las necesidades de la plataforma. Por ejemplo, se podría tener varios servidores de recodificación de video trabajando en paralelo para recodificar el video de múltiples cámaras.
- **Poco acoplamiento:** los diferentes componentes permiten el desarrollo independiente de cada uno de ellos. De esta manera se permite alcanzar el producto final de manera más eficiente.

Una vez expuesto el funcionamiento general de la plataforma se procederá al estudio de cada uno de los componentes mencionados explicando en profundidad sobre el manejo de cada uno de ellos y la manera de comunicarse que estos tienen.

4.2. Diseño detallado

4.2.1. Cliente

El cliente ejecutará una aplicación web que ha recibido el nombre de DVR-LL, y que contará con las funciones necesarias para la gestión de las distintas cámaras. Sin embargo, antes de acceder a la configuración, creación y visualización de las cámaras, es necesario que el usuario inicie sesión. De

esta manera, se está permitiendo que cada configuración de una cámara pueda tener asociado a un único usuario, pero cada usuario pueda tener múltiples cámaras.

Compartición de la información entre servicios

Una vez iniciada la sesión los usuarios tienen acceso a las distintas funcionalidades relacionadas con las cámaras. Dichas funcionalidades se pueden categorizar en dos partes principales:

- **Funciones relacionadas con la gestión de cámaras:** estas funciones se encargan de las operaciones principales para administrar las cámaras en el servidor de base de datos.
    - **Creación:** para generar una nueva cámara, se envía la información necesaria al servidor. Esta incluye la configuración requerida para que la cámara se cree correctamente.
    - **Eliminación:** para remover una cámara existente, se le indica al servidor que deseamos eliminar la cámara específica.
    - **Consulta:** si necesitamos obtener información sobre una cámara en particular, se solicita al servidor dicha información.
    - **Modificación:** para actualizar la información existente de una cámara, se envían los nuevos detalles al servidor.
- **Funciones relacionadas con el servidor DASH-LL:** esta segunda categoría agrupa todas las funciones encargadas de interactuar con el servidor DASH-LL para gestionar las solicitudes de visualización de vídeo.
    - **Visualización en tiempo real:** una de las principales funcionalidades de este grupo es solicitar y manejar el flujo de video en tiempo real. Estas funciones se encargan de establecer la comunicación con el servidor DASH-LL, solicitar el stream de video correspondiente y asegurar su correcta entrega y reproducción en el cliente.
    - **Gestión de grabaciones:** además de la transmisión en vivo, estas funciones también manejan las grabaciones de los streamings realizados previamente. Se encargan de solicitar al servidor DASH-LL las grabaciones disponibles permitiendo a los usuarios acceder a ellas cuando lo necesiten.

Interfaz de usuario

En el momento en el que se conoce como viaja la información desde el cliente a los diferentes servicios se procederá con la explicación de la interfaz de usuario:

La vista ‘index’ es lo primero que el usuario ve cuando entra en la aplicación web. En esta interfaz se visualiza una breve descripción de la plataforma que se va a utilizar incluyendo en la barra de navegación superior un botón de inicio de sesión. Si el usuario pulsa sobre este botón será redirigido a la vista de ‘login’ y podrá acceder a la aplicación si dispone de una cuenta de usuario. Si no dispone de una cuenta de usuario deberá registrarse pulsando sobre el enlace de proporcionado para ello. En este punto el usuario ya es capaz de acceder a la plataforma.

Cuando el usuario ya tiene acceso a las funcionalidades relacionadas con las cámaras (una vez inicia sesión), su navegación empieza en la vista ‘home’. Dicha vista mostrará una lista vacía si el usuario no tiene asociada ninguna cámara o, si sí que las tiene, se visualizará un listado de las mismas.

Cada cámara contará con una serie de botones, que serán las funciones de gestión de esta:

- **Botón de eliminar:** llama a la función de eliminar. Pregunta al usuario antes de realizar la acción.
- **Botón de modificar:** llama a la función de modificación de la cámara. Redirige al usuario a una nueva vista donde se representará la configuración existente en la cámara. Esta vista permite modificar todos los parámetros de configuración de la cámara que representa.
- **Botón de visualizar:** llama a la función de visualización en tiempo real de la cámara. Redirige el usuario a una nueva vista donde podrá reproducir en tiempo real la cámara seleccionada.

Para crear una cámara se dispondrá de un botón en la parte superior derecha Si el usuario hace uso de este botón la plataforma lo redirige a la vista de creación de una cámara. En esta vista el usuario deberá introducir los parámetros de configuración de la cámara a configurar.

Por otro lado, la interfaz web contará con una barra de navegación para mejorar la experiencia de usuario. De esta manera se permite al usuario acceder a las distintas vistas de forma rápida y sencilla. Esta barra de navegación mostrará unas opciones u otras en función de si el usuario ha iniciado sesión. Si se ha iniciado sesión se mostrará:

- **Index:** logo de la aplicación que redirige a la vista ‘index’.
- **Home:** redirige el usuario a la vista ‘home’.
- **Grabaciones:** redirige el usuario a la vista ‘grabaciones’.
- **Cerrar sesión:** cierra la sesión del usuario y lo redirige a la vista ‘index’.

Si no se ha iniciado sesión se mostrará únicamente el logo de la aplicación. Haciendo la misma función que cuando se ha iniciado sesión.

Por último, en la vista de 'grabaciones' el usuario podrá reproducir streamings ya realizados. En esta vista contará con una serie de ajustes para adaptarlo a sus necesidades:

- **Fecha:** se mostrará un calendario cuando el usuario seleccione este campo, y este podrá indicar una fecha para comprobar si existen grabaciones. Si no existen grabaciones se mostrará una alerta informando sobre la inexistencia de grabaciones. Si existen grabaciones se mostrará un listado de botones.
- **Listado de botones:** este listado de botones hace referencia a las horas de la fecha seleccionada. Cada botón se mostrará de color azul cuando exista una grabación en una hora determinada Si no hay grabaciones en una hora determinada el color debe ser el predeterminado. Y, del mismo modo, cuando un usuario presiona sobre una hora donde existen grabaciones se desplegará una barra lateral desplazable (slider).
- **Barra lateral desplazable o slider:** permite al usuario indicar un rango de minutos sobre la hora seleccionada. Por defecto reproduce la hora completa (rango de minutos de 0 a 59) para reducir la complejidad de la interfaz.

Una vez el usuario ha indicado estos parámetros deberá pulsar sobre el botón 'Play' para comenzar la reproducción de la grabación. Dicha grabación se mostrará mediante un reproductor DASH. En particular, se ha utilizado el reproductor dash.js puesto que está desarrollado por el DASH Industry Forum [27]).

4.2.2. Servidor de base de datos

El servidor de base de datos compartido debe soportar múltiples peticiones concurrentes para el intercambio de información, asegurar la durabilidad de los datos ante fallos para mantener la transmisión y grabación en tiempo real, y cumplir obligatoriamente con la asociación de cámaras a usuarios para permitir la asignación y gestión eficiente de los mismos. Por tanto, ya solo con las características descritas anteriormente, el diseño de la base de datos compartida seguirá un modelo ACID (Atomicity, Consistency, Isolation, Durability).

- **Atomicity o Atomicidad:** las transacciones deben ser atómicas Si una transacción falla las demás no deberán tener efecto.

- **Consistency o Consistencia:** los datos escritos deben cumplir con las restricciones de integridad.
- **Isolation o aislamiento:** las transacciones pueden realizarse de manera simultánea sin que afecte a otras.
- **Durability o durabilidad:** los datos permanecen almacenados incluso si se produce un fallo.

En el punto de *"Selección de las tecnologías"* se explicará con detalles la tecnología utilizada para satisfacer las necesidades expuestas.

Por otro lado, al ser una base de datos compartida, esta se ejecutará a modo de servidor, el cual tendrá asignada una dirección IP y un puerto. Por tanto, para que los diferentes servicios se puedan comunicar con esta es necesario que tengan presente el direccionamiento IP de la base de datos.

En este punto, los distintos servicios deberán enviar una petición al servidor de base de datos y este será el encargado de enviar la respuesta.

4.2.3. Servidor DASH-LL (Fast-LL)

Como se ha mencionado anteriormente, en la actualidad no existen cámaras DASH, únicamente existen cámaras RTSP. Por ello, la solución adoptada es la recodificación de video de RTSP a DASH, para la cual es necesario conocer una serie de requerimientos:

El protocolo RTSP es el encargado de servir el video en tiempo real, y para ello es necesario una negociación SDP para permitir el intercambio de información entre el cliente y el servidor.

Los datos referidos al servidor DASH son conocidos por los clientes gracias al archivo de manifiesto, este archivo se denomina MPD. El objetivo de este fichero es localizar alternativas de cada segmento en sus correspondientes descripciones de segmento.

Una MPD está compuesta por un número de elementos "Period". Cada "Period" representa un periodo continuado de tiempo. A su vez, los elementos "Period" están compuestos por elementos "Adaptation Set". Cada uno de estos elementos contiene diferentes versiones del mismo contenido. Las versiones del mismo contenido se diferencian entre sí en algún parámetro de condificación como podría ser el tipo de codificador o la calidad empleada (resolución y bitrate). Cada una de las versiones del mismo contenido se define mediante la etiqueta "Representation".

En lo que se refiere al cálculo preciso del ancho de banda, realizado en el cliente mediante dash.js, se lleva a cabo mediante un algoritmo denominado Look Ahead. Dicho cálculo se ejecutará de forma precisa en cada instante de tiempo, evitando las interrupciones de video. Con esto se pretende obtener una reproducción constante y sin interrupciones, obteniendo la máxima calidad de video reproducido [14] [15] [16].

En un cuanto a la recodificación de RTSP a DASH se utiliza un software que se denomina FFmpeg. Dicho programa será el encargado de realizar la tarea de recodificación de video. Para garantizar la comprensión del servidor con el cliente y la recodificación del video se presenta el siguiente diagrama:

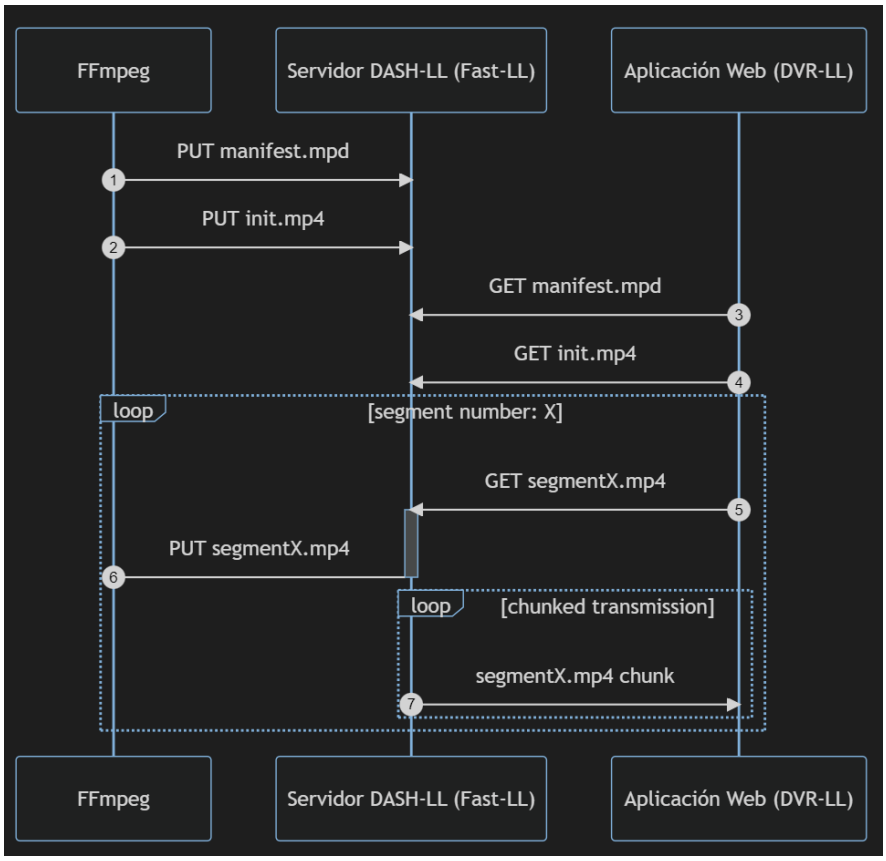

Figura 4.2: Diagrama de una petición en el servidor Fast-LL

El diagrama de la figura 4.2 representa la interacción entre DVR-LL y Fast-LL en relación al streaming DASH Low Latency.

Donde:

1. FFmpeg genera el manifiesto del DASH (manifest.mpd) y el archivo de inicialización (init.mp4). Estos archivos se envían al servidor DASH-LL utilizando una solicitud PUT HTTP.
2. La aplicación web (DVR-LL) realiza una solicitud GET HTTP al servidor DASH-LL para recuperar el manifiesto DASH y el archivo de inicialización.
3. A continuación, comienza un bucle ("loop") en el que la aplicación web solicita repetidamente un segmento de video numerado (segmentX.mp4) del servidor DASH-LL.
4. FFmpeg crea ese segmento, aunque no sea solicitado, y lo carga en el servidor DASH-LL. Esta interacción se muestra con "PUT segmentX.mp4".
5. Una vez que el servidor DASH-LL ha recibido el segmento de FFmpeg, comienza a enviarlo a la aplicación web en fragmentos ("chunked transmission"). Cada segmento de video se envía en respuesta a las solicitudes GET HTTP de la aplicación web.

En resumen, este diagrama muestra cómo FFmpeg, el servidor DASH-LL y la aplicación web trabajan conjuntamente para entregar el video a través del protocolo Low Latency DASH.

Por otro lado, es importante entender cómo el cliente adquiere la información sobre la existencia de grabaciones en una fecha y hora específicas. Para lograr esto, el servidor Fast-LL proporcionará una API que permitirá la recuperación de estos datos.

En dicha API se deberá especificar la fecha, hora y minutos de una cámara determinada. Esto será necesario para cuando el usuario indique una fecha se muestren aquellas horas en las que se ha producido una grabación. Del mismo modo, la fecha también es utilizada para crear y servir la MPD de la grabación a reproducir.

4.3. Selección de las tecnologías

Esta sección abordará las tecnologías que se han elegido para realizar el desarrollo del proyecto. Antes de profundizar en las tecnologías seleccionadas, es posible categorizar las tecnologías utilizadas según los diferentes componentes de la plataforma: la aplicación web, la base de datos compartida y el servidor DASH-LL.

4.3.1. Tecnologías para la aplicación web

Para completar con éxito el desarrollo de la aplicación web se han seleccionado las siguientes tecnologías:

- **Django:** es un framework de desarrollo web que utiliza el lenguaje de programación Python. Este framework es conocido por la gran velocidad de creación de aplicaciones web, la seguridad que ofrece y su gran escalabilidad, permitienco escalar de forma veloz y flexible [17]. Además, cuenta con una interfaz de administración de la base de datos, permitiendo la facilidad de gestión los usuarios y cámaras.
- **HTML:** lenguaje de marcas de hipertexto utilizado para la creación de páginas web y aplicaciones web. En Django son las plantillas (templates) que el usuario visualiza en la interfaz.
- **CSS:** lenguaje de estilos que permite diseñar la interfaz de usuario. En este proyecto se utilizará para ajustar pequeños detalles de la interfaz de usuario.
- **jQuery UI:** es un agrupamiento de interacciones de interfaz de usuario, temas y widgets creados sobre la biblioteca jQuery JavaScript [18]. En este caso, para el desarrollo del proyecto, cuenta con numerosas herramientas que se pueden utilizar, como es el caso del slider o barra horizontal desplazable.
- **Bootstrap:** es un popular framework que se utiliza para el desarrollo de páginas web incluyendo el diseño responsivo (adaptación dinámica de la interfaz de usuario según el tamaño de pantalla del dispositivo). El lenguaje utilizado para este framework es CSS. En la interfaz de usuario del proyecto, este framework será ideal para diseñar de manera rápida y eficaz las interfaces de usuario y, a su vez, permitiendo el diseño responsivo.
- **Bootstrap-Datepicker:** herramienta de código abierto que facilita un widget de selector de fecha flexible junto con el estilo Bootstrap [19].
- **jQuery y AJAX:** jQuery es una biblioteca de JavaScript que ofrece funciones para manipular documentos HTML, manejar eventos y realizar peticiones HTTP mediante AJAX. Esta última técnica permite a las aplicaciones web actualizar contenido dinámicamente sin la necesidad de recargar toda la página

4.3.2. Tecnologías para la base de datos compartida

- **MariaDB:** sistema de gestión de bases de datos relacional (RDBMS) de código abierto que está basado en SQL y que cuenta con un modelo ACID (Atomicity, Consistency, Isolation y Durability) ya mencionado en el punto *4.2.2. Servidor de base de datos*.

Es importante resaltar la característica de aislamiento de esta base de datos, ya que el motor de almacenamiento MyISAM posibilita la realización simultánea de múltiples inserciones. Esto permite la ejecución de instrucciones de selección (SELECT) mientras se llevan a cabo operaciones de inserción (INSERT) [20].

4.3.3. Tecnologías para el servidor DASH-LL

- **Python:** lenguaje de programación multiparadigma (soporta el paradigma funcional, paradigma imperativo y paradigma orientado a objetos) utilizado en diversos campos, como lo son: el desarrollo software, aplicaciones web, entre otros. En el contexto actual, este lenguaje es muy potente para manipular tareas concurrentes mediante el módulo asyncio, permitiendo múltiples conexiones y procesos simultáneos en el servidor de DASH-LL. Además, este lenguaje cuenta con un gran número de bibliotecas disponibles que permiten un desarrollo más eficiente.
- **FastAPI:** framework web que permite la construcción de APIs de manera rápida y con un elevado rendimiento, algunas de sus características más importantes son: rapidez (considerado uno de los frameworks de Python más veloces), desarrollo rápido (permite una programación más rápida), robustez (permitiendo la creación de código listo para producción), fácil de usar (su diseño permite un aprendizaje rápido y facilidad de uso) y está basado en estándares (basado y compatible con los estándares APIs como OpenAPI y JSON Schema) [21].
- **AiomySQL:** librería de código abierto para el acceso a la base de datos MySQL del framework asyncio [22]. Este último se utiliza para el desarrollo de código concurrente.

CAPÍTULO V

Desarrollo de la solución

Este capítulo abordará los detalles técnicos asociados a la implementación de los diferentes elementos destacados en el capítulo previo. Esta evaluación se divide en cinco áreas principales: entorno de desarrollo, servidor de base de datos, aplicación web (DVR-LL), servidor Fast-LL y entorno de pruebas.

5.1. Entorno de desarrollo

Un aspecto importante a tener en cuenta es el entorno de desarrollo empleado en la elaboración de la solución. Pues gracias a este tipo de entornos que cuentan con numerosas herramientas se pueden evitar errores de sintaxis, facilitar la depuración del código, facilitar la navegación, acceso a los recursos del proyecto, entre otros.

Por tanto, el entorno utilizado para la realización del trabajo se denomina PyCharm, el cual ha sido elegido por su especialidad en Python. Dado que el proyecto es desarrollado en Python, este IDE se adapta a todas las necesidades del trabajo, incluyendo una asistencia inteligente a Python, compatibilidad con el desarrollo web (Django, Flask, entre otros), desarrollo utilizando diversas tecnologías (Javascript, HTML/CSS, SQL, lenguaje de plantillas) y varias herramientas de desarrollo (depurador, terminal, interacción directa con la base de datos integrada). Como se puede observar en la figura 5.1 la interacción con la base de datos se realiza a través de la interfaz proporcionada por PyCharm.

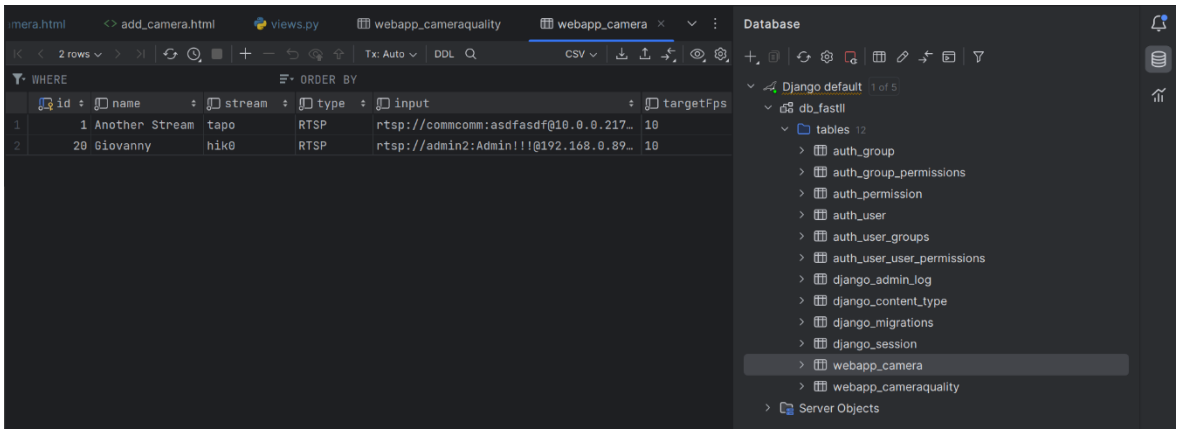

Figura 5.1: Interacción directa con la base de datos desde PyCharm

5.2. Implementación de la base de datos

La manipulación de los datos es un factor importante en este caso, ya que el servidor Fast-LL será capaz de recodificar el video exitosamente si los datos obtenidos de la base de datos son los correctos.

Por otro lado, el servidor Fast-LL soporta múltiples calidades o una única calidad, como ya se ha mencionado anteriormente. Por ello, inicialmente se ha desarrollado una base datos donde las cámaras permitan una sola calidad, permitiendo un desarrollo más rápido en lo que se refiere la interfaz de usuario. Posteriormente se ha realizado la mejora en la base de datos, donde actualmente una cámara soporta diversas calidades. No obstante, se demostrará en el capítulo de "*Pruebas y despliegue"* que el servidor Fast-LL soporta estas calidades.

Por último, se va a presentar dos diagramas de objetos que describen de manera completa la base de datos, el primero permite entender la base de datos inicial y el segundo es el que se encuentra actualmente incrustado en la plataforma,

5.2.1. Base de datos inicial

En este apartado se va a analizar el diagrama de objetos que representa la primera base de datos, entrando en detalle sobre cada uno de los atributos que componen los objetos y el por qué se ha tenido que modificar esta base de datos.

Para comenzar con el análisis se expondrá el diagrama y seguidamente se dará comienzo con la explicación de cada componente. El diagrama es el siguiente:

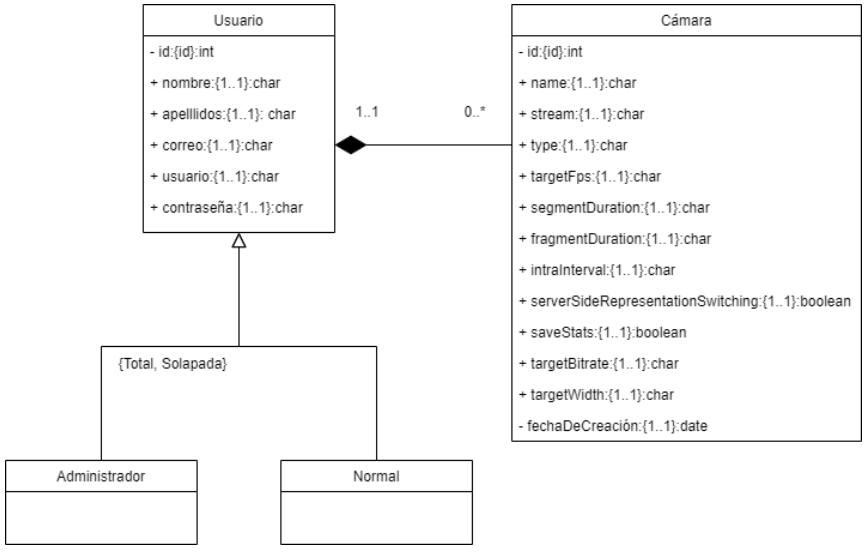

Figura 5.2: Diagrama de objetos en la base de datos inicial

Como se ha podido observar en la figura superior (figura 5.2), el diagrama describe la primera base de datos implementada, donde cada objeto presenta una serie de atributos que pueden ser privados (representado por "-" y no es visible para el usuario) o públicos (representado por "+" y son visibles para el usuario):

- **Usuario:** objeto imprescindible para el uso de la plataforma, el cual puede ser asignado como administrador, usuario normal o ambas (generalización total y solapada). Además, un usuario puede contener múltiples cámaras o ninguna (referencia de uno a ninguno o más). Asimismo, este objeto cuenta con una serie de atributos:
    - **id:** identificador único que permite distinguir un usuario entre los demás, y que se utiliza para referenciar una cámara a un usuario.
    - **nombre, apellidos, correo, usuario (nombre de usuario) y contraseña:** son atributos obligatorios (1..1) y de tipo carácter (cadena de caracteres).
- **Cámara:** objeto que permite almacenar la información de una cámara. Donde una cámara puede estar asociada a un usuario (referencia de ninguno o más a uno). Dada la composición presentada (línea con el rombo relleno), un objeto cámara existirá siempre y cuando exista un usuario. Esto quiere decir que, si un usuario deja de existir y tiene asociado cámaras, estas dejarán de existir también. En cuanto a los atributos, se encuentran:
    - **id:** identificador único que permite distinguir una cámara de las demás.
    - **name:** nombre de la cámara. Sirve para asignar un nombre, facilitando la diferenciación con las otras cámaras para el usuario.
    - **stream:** clave de flujo de una cámara. Esta clave determinará la dirección URL usada para determinar el flujo (stream) que se está utilizando.
    - **type:** tipo de protocolo utilizado por la cámara antes de la recodificación. De momento solo se soporta RTSP.
    - **input:** URL utilizada para acceder al RTSP con credenciales, si es necesario.
    - **targetFps:** cantidad de fotogramas por segundo de la transmisión DASH de baja latencia generada. Es independientemente de la velocidad de fotogramas de la fuente de entrada.
    - **segmentDuration:** duración de los segmentos DASH.
    - **fragmentDuration:** duración de los fragmentos del segmento.

- **intraInterval:** Intervalo entre fotogramas intra. Al menos, el primer fotograma de cada segmento debe ser de tipo intra.
- **serverSideRepresentationSwitching:** indica si usa SSRS. Hay que tener en cuenta que todas las representaciones deben tener la misma resolución
- **targetWidth:** ancho de la transmisión de video. La altura será una proporción uniforme.
- **targetBitrate:** tasa de bits objetivo de la transmisión de video en Kbps.
- **fechaDeCreación:** fecha de creación de la cámara (se realiza de manera implícita).

Un aspecto a destacar en este apartado es el uso del tipo "char" en los atributos del objeto cámara. El uso de este tipo de variable se debe a que el manejo de datos, en el servidor Fast-LL, ha sido desarrollado mediante el tipo string (cadena de caracteres que representan texto). Aunque otra opción podría haber sido la asignación de tipos enteros (int) o de coma flotante (float) en los atributos, y luego parsearlos a string una vez se obtiene la información en el servidor Fast-LL.

Por otro lado, se ha seleccionado esta base de datos en el inicio del desarrollo por su simplicidad y rapidez en términos de programación, permitiendo realizar pruebas a etapas tempranas. Dado que la arquitectura de la plataforma está divida en componentes independientes, la actualización a la base de se puede realizar en otra etapa sin que afecte en gran medida al acople.

En resumen, esta base de datos permite que un usuario pueda ser asignado como administrador o usuario normal, o ambas. A su vez, puede tener asociado varias cámaras o ninguna, y si un usuario deja de existir, las cámaras de este usuario dejan de existir también.

5.2.2. Base de datos actual

El diagrama de objetos que se va a presentar en la figura de la siguiente página es similar a la anterior, pero con una serie de características adicionales. Entre estas se puede encontrar:

- Ahora las variables "targetBitrate" y "targetWidth" son atributos de un nuevo objeto, dicho objeto se denomina "Calidad".
- Con la existencia de este nuevo objeto se puede realizar una referencia de una cámara.
- Se presenta que el objeto "Cámara" está compuesto por "Calidad". Esto quiere decir que si deja de existir un objeto "Cámara" deja de existir la calidad de la cámara.

- Si existe una cámara debe existir como mínimo una calidad. Permitiendo así múltiples calidades en una cámara.

El diagrama de objetos es el siguiente:

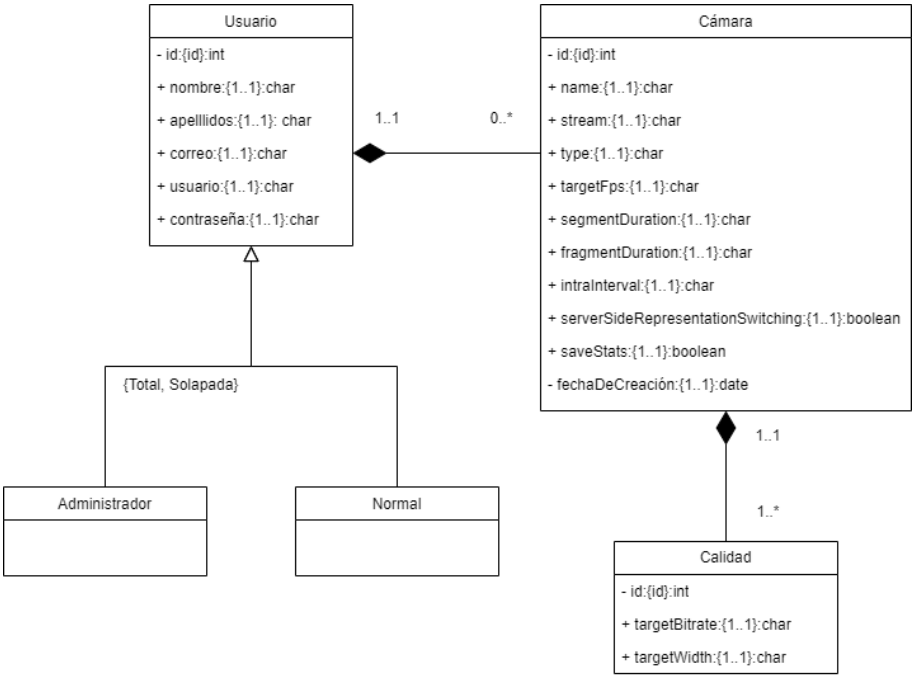

Figura 5.3: Diagrama de objetos de la base de datos actual

En conclusión, la segunda opción permite un funcionamiento similar a la base de datos inicial, pero con la adición de que este modelo permite añadir varias calidades a una cámara.

5.3. Implementación de la aplicación web

Después de conocer cómo funciona la base de datos utilizada en la plataforma, se procederá con la implementación de la aplicación web con el framework Django [23]. Para dicho análisis se tendrá en cuenta la configuración del proyecto, interfaz de usuario, navegación entre páginas y vistas, modelo de datos y maximización del rendimiento de la aplicación web.

5.3.1. Configuración del proyecto

Antes de entrar en profundidad con el resto de los puntos que componen la aplicación, es necesario conocer una serie de variables que permiten la configuración de sesión del usuario, la definición de aplicaciones dentro del proyecto, conexión con la base de datos y utilización de archivos estáticos. Estas variables se deben definir en el archivo 'settings.py' del proyecto.

Configuración de sesión de usuario

Django presenta una gran facilidad para el manejo de sesión de usuario mediante variables como LOGIN_REDIRECT_URL = '/home/' (redirige un usuario a una vista diferente cuando este inicia sesión, en este caso lo redirige a la vista ‘”home”), LOGIN_URL = '/login/' (redirige el usuario a una vista especifica si este intenta acceder a una vista que requiere de su inicio de sesión, en este caso lo redirige a la vista “login”), SESSION_EXPIRE_AT_BROWSER_CLOSE = True (si el valor de la variables es “true” entonces se cerrará la sesión del usuario cuando se cierre el navegador), LOGOUT_REDIRECT_URL = '/login/' (redirige el usuario a una vista una vez cierra sesión, en este caso lo redirige a la vista “login”). En el punto *5.3.3. Navegación entre páginas y vistas* se hablará sobre el uso de estas URLs.

Definición de aplicaciones

Dado que con Django se pueden crear varias aplicaciones dentro de un proyecto, es necesario indicarle a Django las aplicaciones que se van a desarrollar dentro del proyecto mediante una variable de tipo lista llamada INSTALLED_APPS. En este caso la aplicación dentro del proyecto Django se llama “WebApp”, por lo que esta deberá estar incluida en la lista (INSTALLED_APPS = […, 'WebApp']).

Conexión con la base de datos

Para la conexión con la base de datos cuenta con otra variable conocida como “DATABASES”. Esta define la base de datos que utilizará Django para almacenar y recuperar datos. En el proyecto quedaría como:

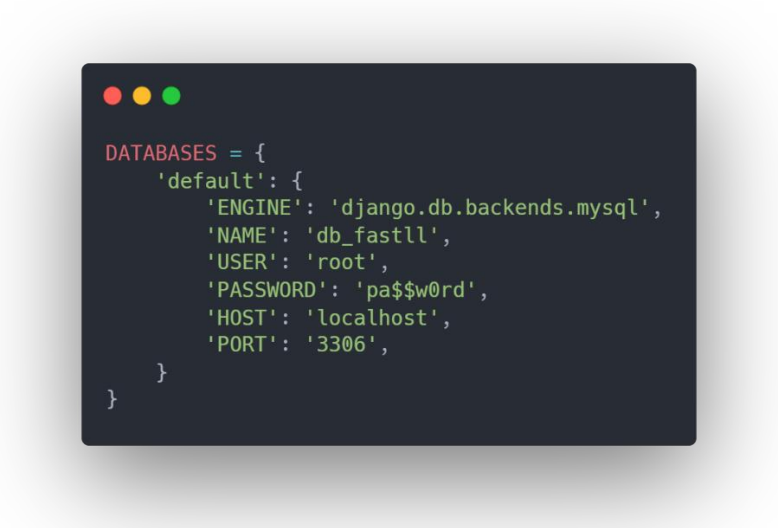

Figura 5.4: Definición de base de datos a utilizar de Django

Donde:

- **ENGINE:** motor de base de datos que utilizará Django.
- **NAME:** nombre de la base de datos.
- **USER:** nombre de usuario para autenticarse contra la base de datos.
- **PASSWORD:** contraseña del usuario para autenticarse contra la base de datos.
- **HOST:** dirección IP donde se encuentra la base de datos. En este caso es en local, por lo que el valor puede ser "localhost" o "127.0.0.1"
- **PORT:** puerto en el que está escuchando la base de datos. Para MySQL el puerto por defecto es 3306.

Archivos estáticos

Django permite la utilización de archivos estáticos. Estos últimos consisten en aquellos archivos que no se modifican durante la ejecución de la aplicación. Estos incluyen, por ejemplo, archivos CSS, Javascript, imágenes, entre otros.

Para empezar a utilizar archivos estáticos, primero se necesitará crear un directorio ("static" en este caso) dentro de la aplicación (en el caso de este proyecto "WebApp") del proyecto Django. Una vez creado el directorio se pueden crear subcarpetas, y dentro de las subcarpetas almacenar dicheros.

En este trabajo se han utilizado las variables:

- **STATIC_URL:** URL que se utilizará para referenciar los archivos estáticos. En este caso, STATIC_URL = '/static/'.
- **STATICFILES_DIRS**: lista de ubicaciones en tu sistema de archivos donde Django buscará archivos estáticos. En este caso, STATICFILES_DIRS = ["WebApp/static",].

Una vez explicado algunas de las configuraciones del proyecto, se procede con la interfaz de usuario.

5.3.2. Interfaz de usuario

Vista index

La primera toma de contacto del usuario con la aplicación web es la vista 'index'. La cual está estructurada en tres partes:

- **Barra de navegación superior:** situada en la parte superior de la vista. Cuenta con el logo de la aplicación, el cual ha sido generado por una página web llamada Hatchful Shopify [24], que permite el diseño de logos de forma gratuita.
  En la misma barra, en el extremo derecho, se presenta un botón que redirige el usuario a la vista de "inicio de sesión".
- **Cuerpo o body:** se encuentra una breve descripción de la aplicación utilizada.
- **Pie de página o footer:** se presenta el nombre del desarrollador de la aplicación.

Figura 5.5: Vista index

Vista de inicio de sesión

Una vez el usuario presiona sobre el botón de iniciar sesión en la vista index, este será redirigido a la vista 'login'. En esta vista se presenta un formulario, en el centro de la página, que el usuario deberá rellenar para iniciar sesión.

Si los datos están introducidos de manera correcta este iniciará sesión y será redirigido a la vista "home". En caso contrario, seguirá en la vista actual.

En esta misma página, se visualiza la opción de registrarse, y que será necesario para aquellos nuevos usuarios. Si el usuario pulsa sobre la opción de registrarse será redirigido a la vista 'register'.

Figura 5.6: Vista de inicio de sesión

Vista de registro

Si el usuario no está registrado deberá crearse primero una cuenta para acceder a la aplicación web. Para acceder a la vista de registro el usuario deberá presionar sobre el link de "Regístrate ahora" de la vista de inicio de sesión. En esta vista de registro el usuario visualizará un formulario en el centro de la página, el cual cumple con los requisitos de la base de datos del objeto "Usuario".

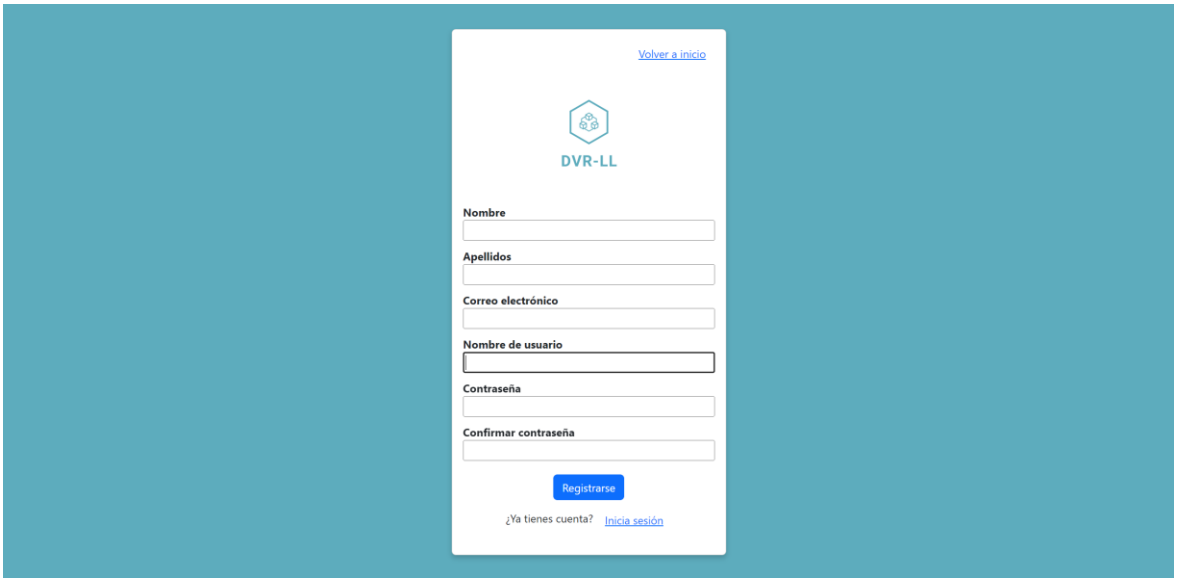

Figura 5.7: Vista de registro

El usuario deberá rellenar cada uno de los datos expuestos en el formulario para completar el registro, de otro modo este no se podrá registrar. Asimismo, si el usuario introduce un usuario o correo ya existente en la base de datos, este deberá introducir un valor no existente

Otro aspecto a destacar de Django es que permite una fácil manipulación de los formularios, permitiendo indicar como quiere que se guarde la información. Por ejemplo, en este caso se le indica a Django que la información introducida en el campo "Correo electrónico" debe ser un correo exactamente, esto se hace con "email = self.cleaned_data.get('email')", y que se puede encontrar en la función que verifica el correo ("clean_email(self)"), localizada en la clase que crea el formulario "CustomUserCreationForm".

Por último, una vez que el usuario presiona el botón de "Registrarse" será redirigido a la vista de inicio de sesión, donde podrá acceder con el usuario y contraseña, permitiendo la funcionalidad completa de la aplicación.

Vista home

Después de que el usuario inicie sesión, la vista "home" será su nuevo encuentro, donde se podrá visualizar el nombre del usuario, un botón para añadir una cámara, el listado de cámaras que este tiene y la gestión de cada una de ellas (modificación, eliminación y visualización en tiempo real). Además, la barra de navegación superior se ha actualizado con nuevas opciones como "Grabaciones" y "Home", y en el extremo derecho de estas se encuentra un botón para cerrar sesión.

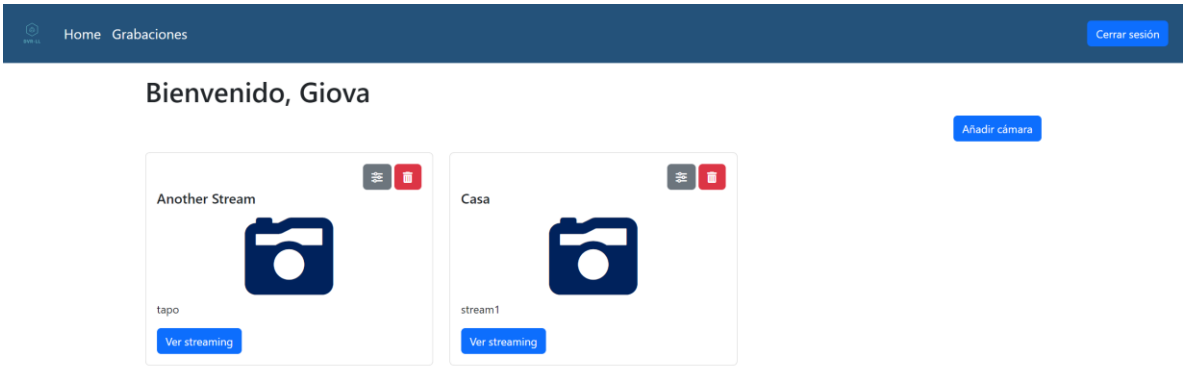

Figura 5.8: Vista de home

Vista de añadir cámara

En esta vista el usuario tendrá en el cuerpo de la página el formulario a rellenar para la creación de una cámara. Asimismo, se puede encontrar una serie de valores ya predefinidos, permitiendo la facilidad de creación de este objeto al usuario. El formulario será enviado una vez el usuario complete todo el formulario, sin dejar campos en blanco, y presione el botón de crear cámara.

Después de que el usuario presione sobre el botón de añadir cámara, este será redirigido a la vista "home", en la que podrá encontrar una alerta/mensaje anunciando la correcta creación de la cámara.

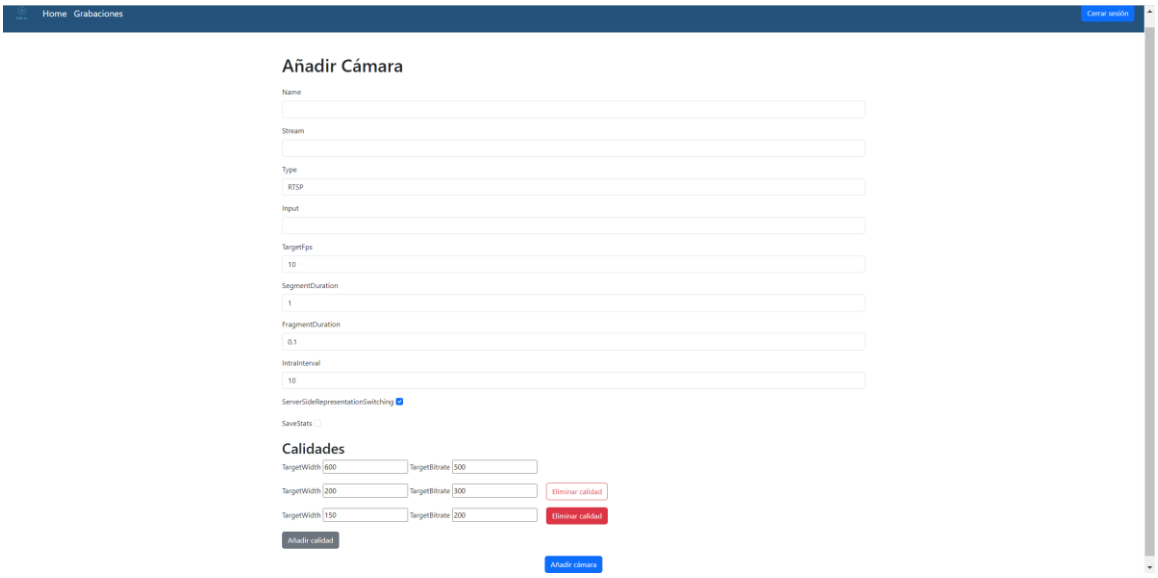

Figura 5.9: Vista de agregación de una cámara

Vista de streaming de una cámara

En esta vista el usuario podrá visualizar un reproductor en el que se reproducirá el video en tiempo real de la cámara seleccionada. Dicho reproductor será un reproductor DASH para la correcta reproducción del video recodificado. Para la reproducción del video en tiempo en tiempo real se debe indicar el archivo de manifiesto (MPD) del flujo de la cámara, por ello es necesario utilizar Javascript, mediante el archivo "camera.js" se indicará al reproductor donde se encuentra este archivo de manifiesto, el cual utiliza la siguiente url "http://localhost:8001/${cameraStream}-dvrll/manifest.mpd". Que es la URL que genera el servidor para servir este archivo.

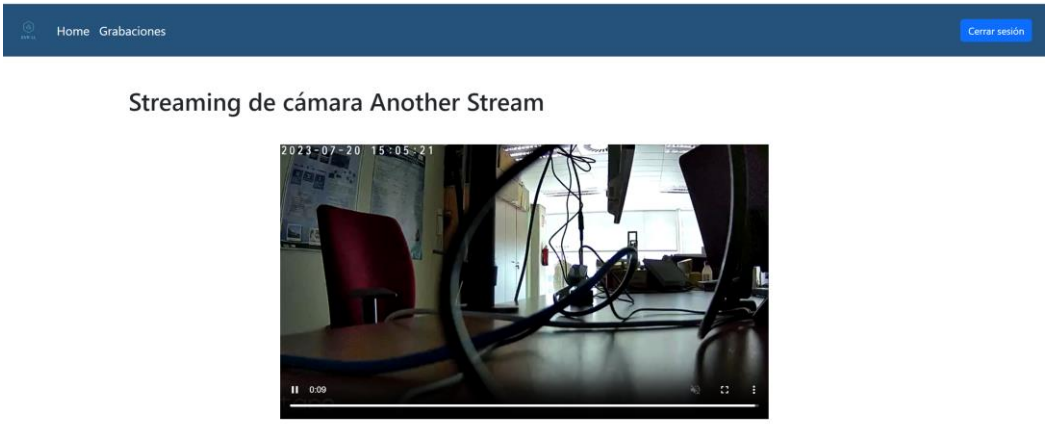

Figura 5.10: Reproducción en tiempo real de una cámara

Modificación de una cámara y eliminación

La interacción de un usuario cuando desea eliminar una cámara se ha realizado mediante una ventana emergente que anuncia si el usuario está realmente de acuerdo con la eliminación de esta. Si este indica que está de acuerdo, la cámara será eliminada, en caso contrario, se mantiene la vista actual (home).

La vista de modificación de una cámara es similar a la vista de agregación de una cámara, solo que en este caso se mostrará la configuración actual de la cámara, permitiendo al usuario realizar una modificación sobre cada campo. Una vez el usuario realiza un cambio y este pulsa sobre el botón de "Guardar cambios", será redirigido a la vista "home", que mostrará una alerta de la correcta modificación.

Vista grabaciones

En esta vista se permitirá al usuario reproducir grabaciones de cámaras que ya han transmitido un video en tiempo real anteriormente. Para ello, el usuario deberá indicar una fecha para buscar dichas grabaciones, seguidamente deberá indicar la cámara y será necesario pulsar sobre el botón buscar para realizar la búsqueda de grabaciones. Este procedimiento enviará una petición (AJAX) al servidor Fast-LL, y este dará respuesta si existen grabaciones de la cámara seleccionada en el día escogido. Dicha respuesta mostrará las horas de un día (de 0 a 23), y que tendrá un valor asociado ("true" si hay grabación de la cámara en esa hora, o "false" si no hay grabaciones a esa hora). Esto permite mostrarle al usuario, mediante una agrupación de botones, aquellas horas donde existe una grabación. Por ejemplo, en la figura 5.10 se puede visualizar que se ha realizado un streaming a las 15:02 de la fecha 20 de julio de 2023. Por tanto, en la vista se mostrará de color azul que hay grabaciones a esa hora.

Otro aspecto a destacar de esta interfaz es que permite al usuario seleccionar los minutos, mediante un slider o barra horizontal desplazable, de una hora determinada. Por defecto reproduce toda la grabación realizada de la hora. Esto quiere decir que reproducirá desde el minuto 0 de una hora, hasta el minuto 59. Este tipo de servicios se debe gracias a las funciones realizadas, mediante FastAPI, en el servidor Fast-LL. En el punto *5.3.5. Extensión del servidor Fast-LL* se explicará cómo se han implementado estos servicios.

Por último, el usuario deberá pulsar sobre el botón "Play" para comenzar con la reproducción.

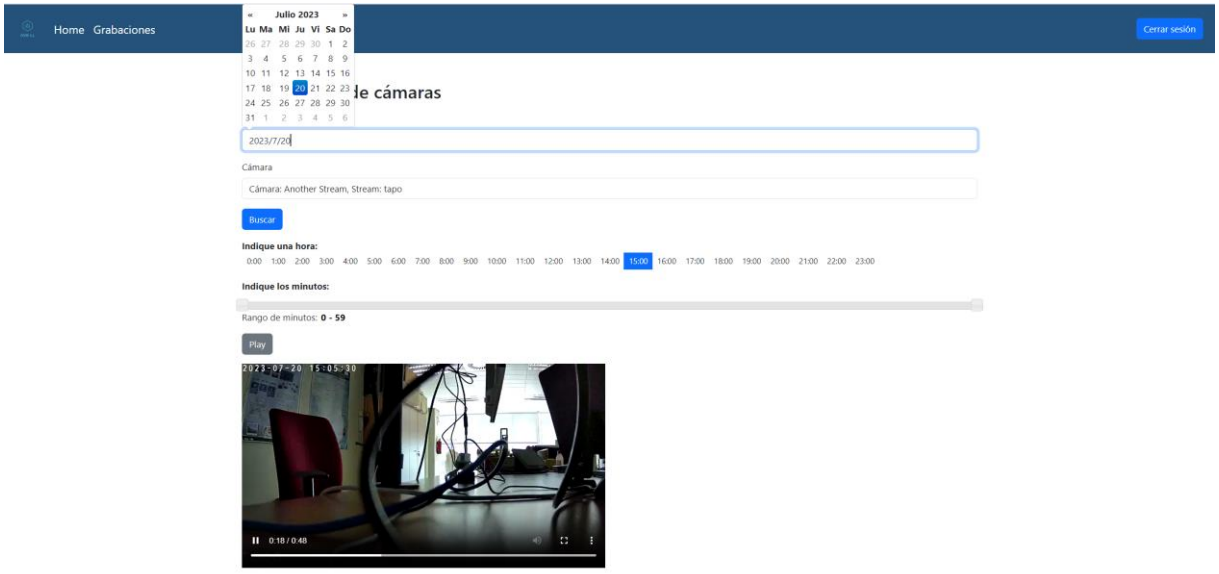

Figura 5.11: Reproducción de una grabación de una cámara

En resumen, mediante estas interfaces de usuario se pretende reducir la complejidad de navegación para los usuarios. Permitiendo a los usuarios una navegación cómoda y simplificada por la aplicación. De esta manera se evita que los usuarios dejen de utilizar la plataforma, ya que no presenta complejidad como otras aplicaciones analizadas.

Vista de administrador

Si el usuario dispone del privilegio de ser administrador contará con una interfaz adicional, y que podrá gestionar toda la base de datos. Dicha interfaz se encuentra en la vista "admin" generada por Django. En este caso, para acceder a esta interfaz se debe indicar en la URL "http://localhost:8000/admin".

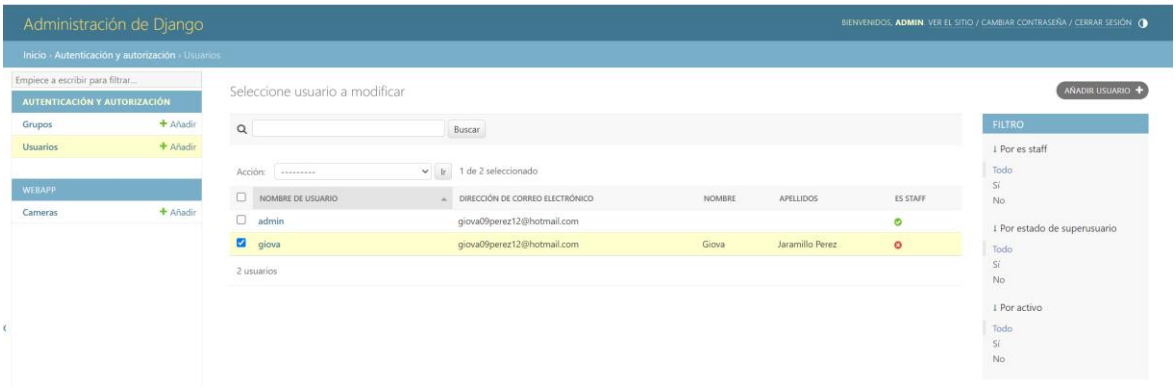

Figura 5.12: Interfaz de administrador

Diseño responsive

La interfaz de usuario se adapta los diferentes tamaños de pantalla, esto se conoce como diseño responsivo. Mediante este diseño se permite que cada uno de los componentes que se encuentran en la interfaz se ajusten a la pantalla. El trabajo de este diseño se ha simplificado mucho gracias al framework de Bootstrap, el cual permite asignar una clase a cada componente para la redimensión del mismo. El resultado de esto es el siguiente:

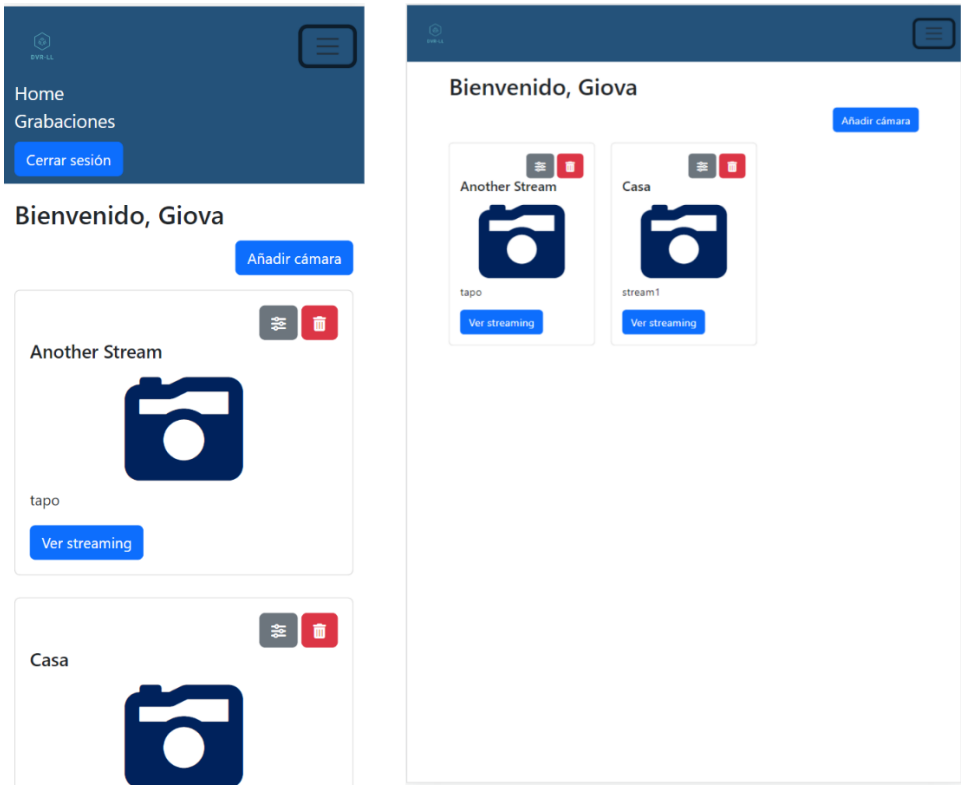

Figura 5.13: Interfaz responsive en diferentes tamaños de pantalla

5.3.3. Navegación entre las vistas

La navegación entre páginas se permite gracias a los archivos “urls.py” y “views.py” que proporciona Django, los cuales se encuentran en la aplicación del proyecto.

En el fichero “urls.py” se declarará cómo el usuario navegará entre las vistas, mediante la definición de las URLs de la aplicación. Pero para que esto se lleve a cabo es necesario el archivo “views.py”, que es el encargado de definir las vistas de la aplicación. En Django, una vista puede ser una función o una clase.

Para un mejor entendimiento de lo mencionado anteriormente se realizará un ejemplo en relación con la aplicación:

En este caso se pretende realizar una navegación a la vista de agregación de cámara y, una vez situado en esta, cuando el usuario finalice el formulario, se deberá añadir a la base de datos. Para ello, en el archivo “views.py” se deberá definir la vista. La cual, en este caso es una función (add_camera_view(request)) que maneja las solicitudes HTTP de la vista “add_camera_view”. Dado que en este caso se pretende enviar información (datos de la cámara), se definirá una comprobación para la solicitud POST (“if request.method == 'POST'”). Esta función devolverá una plantilla renderizada, que en este caso se encuentra en el path “WebApp/templates/home/cameras/ add_camera.html”.

Una vez se crea la vista, se puede proceder con la definición de las URLs para la navegación de la aplicación. La lista “urlpatterns” del archivo “urls.py” permite definir el path de navegación, y que deberá estar relacionado con la vista creada. En este caso el path debe ser “path('home/add_camera/', views.add_camera_view, name='add_camera')”. Donde:

- **'home/add_camera/':** ruta URL. Cuando se accede esta URL se invocará una vista. En este caso, la URL quedaría como “http://localhost:8000/home/add_camera”.
- **views.add_camera_view:** vista que será invocada cuando se visite la URL. Esta debe ser una función o clase.
- **name='add_camera':** nombre que recibe la ruta URL, y que permite referirse a ella de una manera que no dependa de la ruta URL en sí.

5.3.4. Modelo de datos y formularios

En el proyecto, para la representación del esquema de la base de datos y su interacción con el usuario, se ha recurrido a la implementación de dos archivos centrales: “models.py” y “forms.py”.

El archivo “models.py” es fundamental en el proyecto ya que aquí se encuentra definida la estructura de nuestro principal objeto de interés, la cámara. La representación de este objeto es muy importante, ya que es el núcleo de nuestra aplicación y contiene toda la información necesaria para su funcionamiento. Esta estructura, que se ha explicado previamente en el apartado *5.2.1. Diagrama de objetos de la base de datos actual,* nos da una visión clara de los atributos de una cámara y cómo se relaciona con otros componentes del sistema, particularmente con el modelo “User” de Django.

Para la recopilación de datos de usuario y la interacción con estos datos, se implementa la clase “CameraForm” dentro del archivo “forms.py”. Este formulario, vinculado directamente al modelo “Camera”, es el responsable de recoger y validar los datos del usuario antes de su almacenamiento en la base de datos. Un detalle a destacar es la inclusión de un campo de selección para el atributo “type”, que permite al usuario escoger entre las opciones disponibles.

En resumen, los archivos models.py y forms.py conforman el núcleo de la interacción entre el usuario y la base de datos, facilitando la entrada y validación de datos.

5.3.5. Maximización del rendimiento de la aplicación

Para maximizar el rendimiento de la aplicación web se ha utilizado una herramienta de Google Chome conocida como “Lighthouse”, que permite ampliar el rendimiento de la aplicación, en este caso se centrará en tres análisis de los cinco que presenta esta herramienta: potencia, accesibilidad y buenas prácticas.

Si la puntuación obtenida en uno de los análisis no es el deseado, se puede realizar click sobre este y observar que hace falta para mejorar la interfaz.

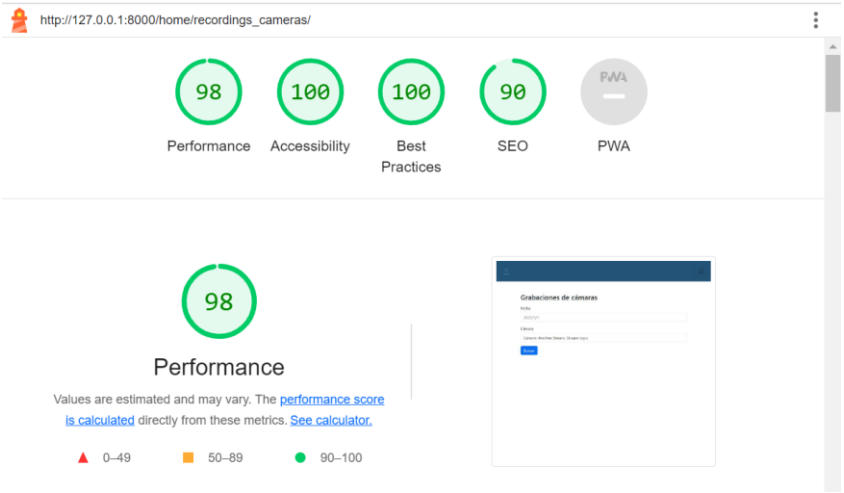

Figura 5.14: Análisis de rendimiento de la interfaz mediante Lighthouse de Google Chrome

5.4. Extensión del servidor Fast-LL

El servidor Fast-LL previamente ya realizaba la recodificación de video en tiempo real de RTSP a DASH. Por consiguiente, este apartado se ha denominado como "*Extensión del servidor Fast-LL*". En este caso, se adoptará un enfoque diferente para que se integre adecuadamente con la aplicación web. Para ello, se tendrá en cuenta una serie de puntos que deberán añadirse o modificar: procesado de la información obtenida en la base de datos, sistema utilizado para el almacenamiento de las grabaciones y creación de los servicios ofrecidos por el servidor Fast-LL utilizando FastAPI.

5.4.1. Procesado de la información obtenida de la base de datos

Esta tarea (DbCheckTask) inicia un ciclo infinito que, cada cinco segundos, establece una conexión con la base de datos y recupera la lista de todas las cámaras configuradas en ella. Para cada cámara obtenida, se extraen sus detalles de configuración, como el nombre, el tipo de transmisión, la resolución, la tasa de bits y otros parámetros relevantes.

Si la cámara obtenida de la base de datos aún no se encuentra en la lista local de cámaras activas, la tarea asume que es una cámara nueva y la añade a la lista. Al mismo tiempo, inicia la transmisión de video desde esa cámara utilizando su configuración.

La tarea también realiza una limpieza de la lista de cámaras activas. Si una cámara que estaba previamente en la lista ya no está en la base de datos, la tarea detiene la transmisión de esa cámara y la elimina de la lista.

En resumen, este código realiza una tarea crucial: mantener sincronizadas las cámaras activas y su configuración con la información almacenada en la base de datos. Este proceso asegura que siempre se estén monitoreando las cámaras correctas con los parámetros de configuración correctos.

5.4.2. Almacenamiento de las grabaciones

Se ha implementado un sistema de almacenamiento que se guarda en disco. Dicho almacenamiento se realiza mediante un sistema de ficheros, el cual presenta la siguiente estructura:

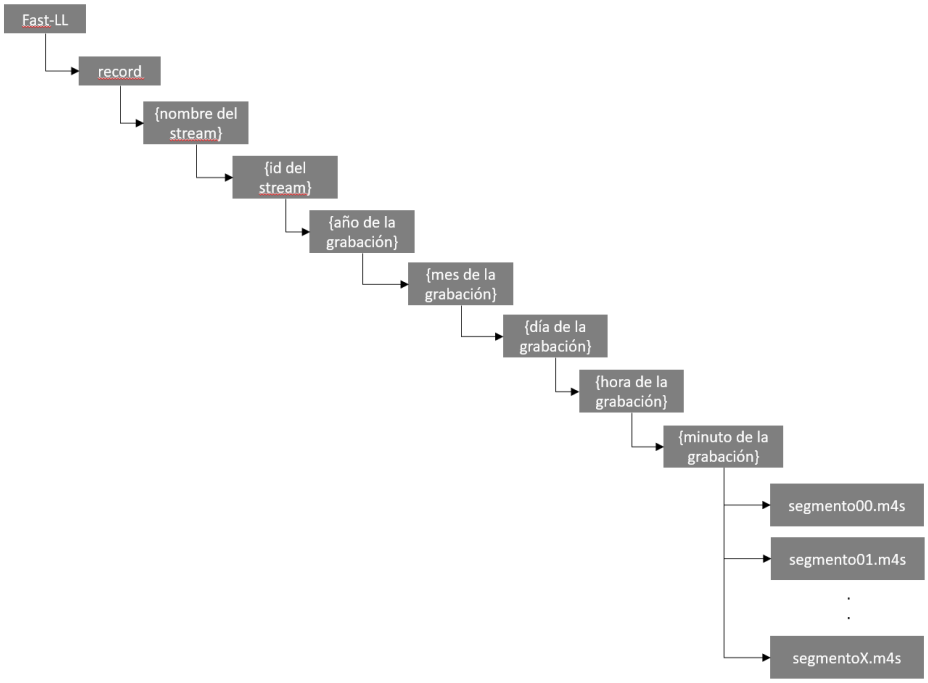

Figura 5.15: Representación del sistema de ficheros utilizado para el almacenamiento de los segmentos de un video

El encargado de realizar la tarea de almacenamiento de los ficheros en los diferentes directorios es la función “save_segment()”. Esta función se encarga de almacenar los segmentos de datos de un flujo (stream) en tiempo real. Inicialmente, extrae información del identificador y la fecha del segmento, ajustando el dato de tiempo al huso horario de Madrid.

Luego, construye una ruta específica en el sistema de archivos utilizando el nombre del flujo, el identificador del segmento y los componentes de la fecha. Si el directorio correspondiente a esta ruta no existe, lo crea.

Posteriormente, recopila y ordena todos los archivos existentes en ese directorio que corresponden a segmentos anteriores y cuenta cuántos son. Con este recuento, determina el nombre del nuevo archivo de segmento que va a guardar.

Por último, genera la ruta completa al archivo de destino y abre este archivo en modo escritura. Escribe los datos del segmento en el archivo y luego cierra el archivo.

Mediante este sistema de ficheros se permite un almacenamiento ordenado y fácil de implementar, permitiendo flexibilidad para gestionar los archivos desde el sistema operativo.

5.4.3. Servicios ofrecidos mediante FastAPI

Los servicios ofrecidos se han implementado con FastAPI, framework ya mencionado anteriormente. Entre los servicios ofrecido se encuentran:

Servicio que ofrece las horas grabadas en un día determinado

Como ya se ha mencionado anteriormente en el punto *5.3.2. Interfaz de usuario*, concretamente en la sección de "*Vista grabaciones*". El cliente realiza una petición desde la aplicación web para obtener las horas grabadas, y desde el servidor se envía la respuesta con la existencia, o no, de grabaciones de un día determinado.

Este servicio recibe tres parámetros: el nombre del stream de video, la ID del segmento de stream, y la fecha para la cual se quiere obtener las horas de grabación (`@app.get("/stream/{stream_name}/{segment_stream_id}/date/{date}")`). Con estos datos, se construye la ruta hacia el directorio donde se almacenarían las grabaciones de ese stream y ese día.

Inicialmente, se asume que no hay grabaciones para ninguna hora del día. Esto se representa como un diccionario en el que las claves son las horas del día (en formato de 24 horas), y los valores son "false".

A continuación, si se encuentra que el directorio correspondiente a la fecha dada existe y es un directorio válido, se itera sobre todos los subdirectorios en él. Cada uno de estos subdirectorios correspondería a una hora del día durante la cual se ha grabado video.

Para cada subdirectorio encontrado, se actualiza el diccionario de horas, cambiando el valor de la hora correspondiente a "true", lo que indica que hay grabaciones para esa hora. En el capítulo *Pruebas y despliegue* se mostrará el funcionamiento de este servicio.

Finalmente, se devuelve este diccionario de horas, lo que ofrece una representación de qué horas del día dado tienen grabaciones de video para el stream especificado.

Servicio que genera el archivo de manifiesto (MPD) para la reproducción de una grabación

La funcionalidad para reproducir una grabación en la aplicación web es habilitada gracias a este servicio. Es este el que tiene la tarea de reunir todos los archivos relevantes (.m4s) y generar un archivo de manifiesto MPD necesario para la reproducción de la grabación.

El proceso inicia estableciendo el directorio base, que incluye el nombre del flujo y su ID. A continuación, identifica los archivos que se encuentran dentro del rango de tiempo especificado, incluyendo un archivo inicial MP4 esencial para el funcionamiento. Este archivo es generado gracias a la función “save_initial_segment()”, que se encarga de almacenar el segmento inicial.

Seguidamente, el servicio calcula la duración total de la grabación en segundos y la transforma en el formato adecuado para su inclusión en el archivo MPD.

El archivo MPD se genera siguiendo la estructura XML conforme a la norma MPEG-DASH [25]. Incluye información necesaria como el título de la grabación, la duración total, las características técnicas del contenido del video y la lista de segmentos que representan las secciones individuales del contenido a reproducir.

Cada segmento del contenido de la grabación se incorpora en el MPD como un SegmentURL dentro de la SegmentList, que también contiene el archivo de inicialización.

Por último, el archivo MPD generado se envía en respuesta a la solicitud del cliente, permitiendo así que este pueda reproducir la grabación en el rango especificado.

5.5. Entorno de pruebas

Este proyecto, dada su naturaleza, requiere la realización constante de pruebas con las cámaras IP. Para abordar este desafío y mantener un entorno de trabajo eficiente y seguro, se ha establecido una VPN. Esta VPN actúa como un puente, permitiendo el acceso remoto a la ubicación de las cámaras IP.

La razón detrás de la creación de esta VPN es posibilitar el acceso a las cámaras ubicadas en el laboratorio del Grupo de Comunicaciones Multimedia en el campus de Vera de la UPV.

Para la creación de dicho túnel se ha utilizado un software llamado WireGuard, el cual presenta características importantes como la velocidad, la seguridad y la facilidad de implementación [26].

Este elemento es de gran importancia para el desarrollo de la plataforma, pues de otro modo sería difícil realizar las pruebas.

CAPÍTULO VI

Pruebas y despliegue

En este capítulo se describen las pruebas realizadas para comprobar el correcto funcionamiento de la implementación desarrollada y, por otro lado, se explicará el proceso para la puesta en marcha de la plataforma, conocido como el despliegue.

6.1. Pruebas

6.1.1. Grabación de múltiples calidades en una cámara

Para comprobar que una cámara soporta múltiples calidades se ha creado una cámara con varias calidades desde la interfaz de usuario. Ahora, una vez se ejecuta el servidor se visualizará en el terminal, mediante los logs, la configuración de cada cámara, permitiendo analizar si esta cuenta con varias calidades.

Figura 6.1: Comprobación del soporte de múltiples calidades a una cámara

6.1.2. Obtención de valores a partir de la API implementada

Obtención de las horas grabadas en un día determinado

Mediante este servicio, implementado en el servidor, se pueden comprobar las horas grabadas de un día determinado, la URL a utilizar para comprobar el funcionamiento es la siguiente "`http://127.0.0.1:8001/stream/{stream_name}/{segment_stream_id}/date/{date}`". Donde:

- **stream_name:** es el nombre del flujo (stream) de la cámara.
- **segment_stream_id:** identificador del segmento del flujo, en este caso es 0.
- **date:** fecha que se desea realizar la búsqueda de grabaciones. Esta tiene formato "{año}-{mes}-{día}".

A modo de ejemplo, se realizará la comprobación del streaming realizado en la figura 5.10. Dicho streaming se transmitió a las 15:02 de la fecha 20 de julio de 2023. Por tanto, la URL será "http://localhost:8001/stream/tapo/0/date/2023-7-20".

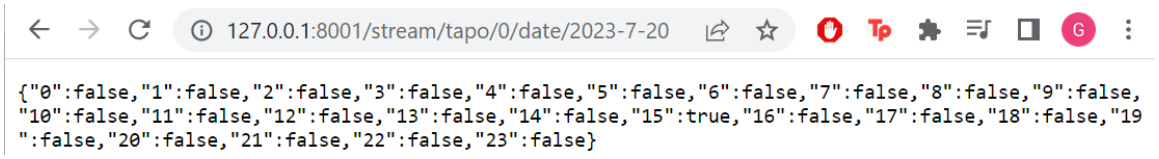

Figura 6.2: Comprobación de las horas grabadas en un día determinado

Como se puede comprobar en la figura 6.2, existe únicamente una grabación a la hora 15. Así pues, se verifica correctamente el funcionamiento de este servicio.

Obtención de la MPD generada para la reproducción de grabaciones

Para comprobar el funcionamiento de obtención de una MPD para reproducir una grabación primero se debe conocer la URL a utilizar: http://localhost.8001/stream/{stream_name}/{stream_id}/{init_date}/{end_date}. Donde:

- **stream_name:** es el nombre del flujo (stream) de la cámara.
- **stream_id:** identificador del flujo (stream), en este caso es 0.
- **init_date:** fecha inicial con la hora y minutos. Con formato "{año}-{mes}-{día}-{hora}-{minuto}".
- **end_date:** fecha final con la hora y minutos. Con formato "{año}-{mes}-{día}-{hora}-{minuto}".

Figura 6.3: Comprobación de la generación de la MPD a través del servicio

6.1.3. Acceso a las cámaras de forma remota

Como ya se mencionó en el punto *6.1.3 Entorno de prueba*s, el acceso a las cámaras se realiza mediante una VPN. Para comprobar que se tiene acceso a estas, bastaría con realizar un ping desde el terminal para conocer si se llega al destino. El procedimiento de este es el siguiente:

- Primero, se envía un mensaje ICMP Echo Request al host destino, en este caso, una cámara accesible, para que esta pueda responder.
- Finalmente, la cámara IP (host) dará respuesta, mediante un ICMP Echo Reply.

Para un mejor entendimiento de lo mencionado anteriormente, en la imagen inferior se demuestra el análisis realizado.

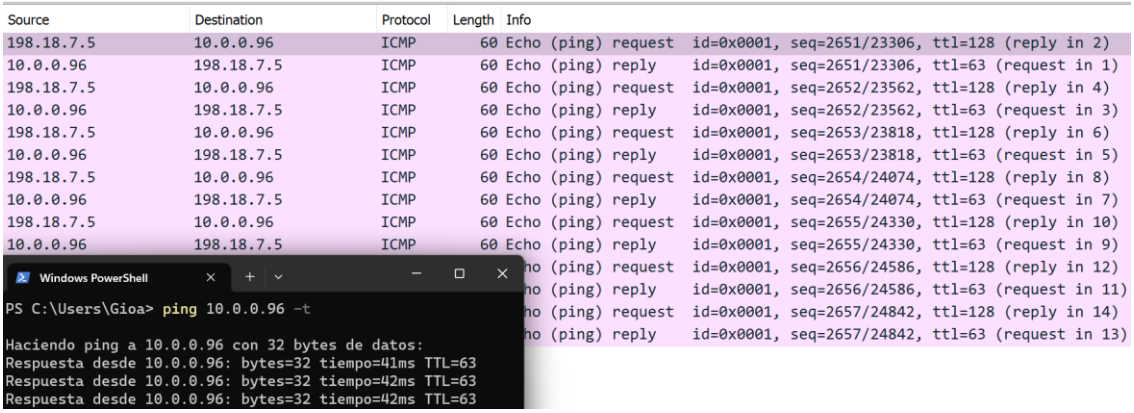

Figura 6.4: Intercambio de paquetes IP entre el host local y cámara IP

En esta figura se encuentran dos elementos, una terminal para realizar el ping a la cámara de host destino (cámara IP) y un software analizador de paquetes, conocido como Wireshark, que permite estudiar detallamente cada paquete intercambiado.

Una vez se deja el ping en el terminal, de manera constante con el parámetro “-t”, se puede comprobar en la interfaz de Wireshark como el host local, con dirección IP 198.18.7.5, envía una petición ICMP y responde la cámara IP, con direccionamiento IP 10.0.0.96.

Finalmente, para comprobar la reproducción de video en tiempo en real, mediante el protocolo RTSP, se puede introducir la dirección URL de la cámara (con las credenciales de acceso si esta tiene) en aplicaciones con capacidad de reproducir este protocolo, como es el caso de VLC, y que se puede observar en la figura 2.4.

6.2. Instalación y despliegue

Para la puesta en marcha de la plataforma será necesario instalar los servicios que la componen: servidor de base de datos compartido (MariaDB), cliente (aplicación web) y servidor DASH-LL (Fast-LL).

6.2.1. Instalación y puesta en marcha del servidor de base de datos

Primero se instalará el servidor de base de datos compartido a través del siguiente enlace: https://mariadb.org/download/.

En la instalación del servidor de base de datos se mostrará una serie de opciones de configuración, pero en este caso bastaría con centrarse en la habilitación de los caracteres UTF-8 (marcar la casilla donde se indica "Use UTF8 as default server's carácter set") y la contraseña del usuario "root", el cual tiene acceso a toda la base de datos. La primera opción permite el almacenamiento y manejo correcto de múltiples caracteres, incluyendo acentos. La segunda opción, permite al usuario indicar una contraseña para el usuario "root". Dicha contraseña deberá ser "pa$$w0rd" si se desea una integración directa y automática con el resto de los servicios (aplicación web y servidor DASH-LL). O, si se desea otra contraseña distinta se deberá especificar en los otros servicios, ya que de otro modo no se tendrá acceso al servidor de base de datos.

Una vez se instaló el servidor de base de datos se deberá crear la propia base de datos. Para ello, desde un terminal, se deberá comprobar si se tiene acceso al comando "mysql", dicho comando es "mysql --version". Si no se tiene acceso al comando se visualizará en terminal un error como "El término 'mysql' no se reconoce como nombre de un cmdlet, función, archivo". Para solucionar este error, se deberá añadir "mysql" al PATH del sistema.

En este caso, el sistema operativo sobre el que se ha realizado el proyecto es Windows. Por tanto, bastaría con buscar en el menú de inicio "Editar las variables de entorno del sistema", seguidamente pulsar sobre "Variables de entorno…", seleccionar la variable "Path" e indicar "Editar". Una vez dentro de la variable a editar, se añadirá, mediante el botón "Nuevo", la ruta al directorio que contiene el comando mysql. Por ejemplo, si se instaló MariaDB en "C:\Program Files\MariaDB 11.2\", la ruta sería "C:\Program Files\MariaDB 11.2\bin\".

Si se ha realizado correctamente los pasos anteriores, desde el terminal se mostrará algo similar a la siguiente figura:

Figura 6.5: Funcionamiento correcto del comando “mysql” en el sistema

Ahora que el sistema reconoce el comando “mysql” se puede crear la base de datos. Primero se deberá acceder como usuario root a la interfaz de MySQL mediante el comando “mysql -u root -p” desde el terminal.

Por último, una vez dentro de la interfaz MySQL se ejecutará el comando “CREATE DATABASE db_fastll;” para la creación de la base de datos. Si todo ha salido bien, mediante el uso del comando “show databases;”, se visualizará la base de datos como en la figura 6.6.

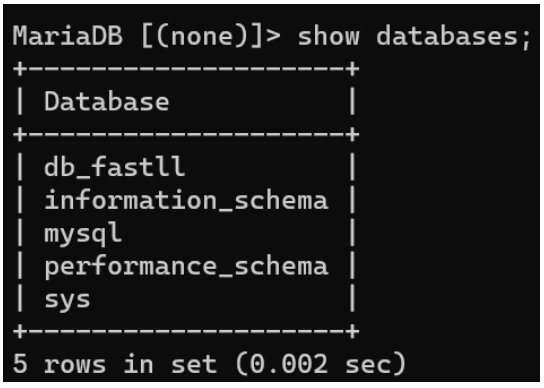

Figura 6.6: Listado de base de datos

6.2.2. Despliegue de la aplicación web

Una vez creada la base de datos, se procederá con el despliegue de la aplicación web. Por tanto, después de que la aplicación haya sido descargada, se deberán incluir los modelos de Django en la base de datos y, seguidamente, arrancar el servicio.

Para incluir los modelos de Django en la base de datos se debe ejecutar, desde el terminal, el comando “python manage.py migrate” desde el directorio del proyecto. De esta manera, se permite la migración de los modelos de Django a las tablas de la base de datos.

Si la migración se ha realizado de manera correcta se puede visualizar, con el comando” show tables” y desde el terminal, las tablas generadas en la base de datos:

```
MariaDB [db_fastll]> show tables;
+----------------------------+
| Tables_in_db_fastll        |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| webapp_camera              |
| webapp_cameraquality       |
+----------------------------+
12 rows in set (0.001 sec)
```

Figura 6.7: Listado de tablas en la base de datos

Ahora que la base de datos tiene las tablas necesarias para el correcto funcionamiento de la plataforma, se puede proceder con el arranque del servicio web.

La aplicación web cuenta con un entorno virtual de Python, directorio "venv". Este entorno sirve como herramienta para separar las dependencias requeridas por diferentes proyectos, aislando el entorno de cada uno. Para acceder a este entorno, desde el directorio del proyecto, se debe ejecutar el comando "venv\Scripts\activate" que permite el acceso a este.

Si al ejecutar el comando se produce un error que anuncia la imposibilidad de ejecución del script en el sistema, se debe autorizar la ejecución de este. La razón de esto es que los sistemas Windows restringen la ejecución de scripts debido a sus políticas de seguridad. Para solucionar este problema se puede introducir el comando "Set-ExecutionPolicy Unrestricted -Scope Process", que permite la ejecución de scripts en el terminal actual. Esto quiere decir que solo se permiten ejecutar scripts en el terminal abierto, y una vez cerrado la política volverá a su estado inicial.

Ahora, una vez dentro del terminal, se puede arrancar el servicio mediante el comando "python manage.py runserver", siendo este accesible desde el navegador con la URL http://127.0.0.1:8000/. Una vez se accede a la aplicación, la primera interfaz (vista index) a visualizar es como la de la figura 5.5.

En este punto se podrá crear usuarios, iniciar sesión y gestionar cámaras, pero no se podrá visualizar las cámaras en tiempo real ni sus grabaciones. Para conseguir esto último es necesario arrancar el servidor Fast-LL.

6.2.3. Despliegue del servidor Fast-LL

Este quizás es el punto más fácil, ya que para arrancar el servidor Fast-LL será necesario tener en cuenta el entorno virtual (como en el punto anterior) y el comando para arrancar el servicio.

Primero se deberá acceder al entorno virtual, de igual manera que en el punto previo. Una vez se encuentra dentro del entorno virtual, se ejecutará el comando "py fastllapp.py -c config_example_new.json" para arrancar el servidor.

Una vez se arranca el servicio, la plataforma ya funciona de manera correcta, permitiendo la funcionalidad de la reproducción en tiempo real de las cámaras y de las grabaciones realizadas.

CAPÍTULO VII

Conclusiones

En este trabajo de final de grado se ha desarrollado una plataforma web capaz de gestionar múltiples cámaras IP, permitiendo la reproducción de cada una de ellas en tiempo real con diferentes calidades y la grabación de la misma.

Durante las primeras etapas del proyecto, se realizó una exploración exhaustiva de las soluciones existentes en el mercado, lo que permitió identificar tanto ventajas como desventajas con el sistema propuesto en este trabajo. Al finalizar dicha investigación, se llegó a la conclusión de que esta plataforma web se distingue por ofrecer una interfaz de usuario intuitiva y accesible, incluso para aquellos que no cuentan con un amplio conocimiento en el campo. Además, esta propuesta representa un avance innovador, pues aborda y soluciona desafíos actuales, como la falta de soporte para cámaras DASH, posicionándose como una alternativa relevante en el ámbito tecnológico.

Tras la elección de las tecnologías pertinentes y la definición del esquema del proyecto, se ha avanzado en su implementación, logrando de esta forma los objetivos secundarios establecidos. Un aspecto primordial fue la evaluación del sistema de base de datos a utilizar, donde en un primer momento se optó por una opción que permitía desde la interfaz web añadir una única calidad a una cámara, permitiendo una mayor rapidez en el desarrollo de la plataforma, pero teniendo en cuenta que en etapas medias se debería introducir la mejora para diversas calidades. Una vez ya se comprendía la estructura de la base de datos a utilizar, se procede con la implementación de la aplicación web, la cual se ha realizado especial dedicación para simplificar las tareas a los usuarios menos expertos, permitiendo la gestión total de las cámaras de manera accesible. En este punto también se implantó el modelo de datos a utilizar, el cual transforma a tablas de la base de datos. La puesta en marcha de los servicios mencionados permitió extender el servidor de Fast-LL, el cual resultó un reto, ya que se tuvo que estudiar en profundidad el funcionamiento del servidor para así proceder con la extensión y adaptación a la plataforma. Entre estos estos se encuentra: almacenamiento de los segmentos en el sistema de ficheros de manera coherente y ordenada, y los servicios ofrecidos mediante FastAPI para la obtención de un video en tiempo real y la grabación de este.

Otro aspecto a destacar en el desarrollo de este proyecto ha sido el amplio conocimiento que se ha obtenido, abarcando temas como el manejo de protocolos que no se conocía, adaptación e

integración de varios servicios, e incluso tecnologías que no se han dado en los estudios cursados o ampliación de los que ya se conocía.

Para concluir, se puede afirmar que la plataforma web ha alcanzado el objetivo principal del proyecto con éxito. Además, se han cumplido los desafiantes objetivos secundarios, lo cual ha estimulado aún más el deseo de llevar a cabo este proyecto. Durante esta trayectoria se ha logrado consolidar conocimientos previos y también se han obtenido nuevas habilidades, relacionadas tanto con el ámbito informático como con el mundo de las cámaras IP.

7.1. Relación del trabajo desarrollado con los estudios cursados

El proyecto llevado a cabo guarda una íntima relación con las habilidades y conocimientos obtenidos durante mi formación universitaria, especialmente en la especialización de Tecnologías de la Información. Por un lado, el trabajo destacó la relevancia de la integración de aplicaciones, puesto que fue esencial establecer comunicaciones fluidas e intercambiar información entre varios servicios, como la aplicación web, la base de datos compartida y el servidor Fast-LL.

Por otra parte, la creación de una aplicación web resaltó la importancia de contar con una sólida base en diseño y funcionalidad web. Fue en este ámbito donde se hicieron efectivos los conocimientos de tecnologías como HTML, CSS, JavaScript y AJAX, todas las cuales se introdujeron durante el transcurso de la carrera. Además, el entendimiento de varios marcos de trabajo utilizados en la carrera facilitó el acercamiento a Django, a pesar de que este no fue cubierto durante el programa académico.

Se ha tenido una constante preocupación por el diseño centrado en el usuario. El objetivo era ofrecer al usuario una experiencia sencilla, intuitiva y adaptable a diversos dispositivos, lo que se conoce como diseño responsive.

Otra consideración importante fue la gestión de redes y sistemas, aprendizaje que se adquirió durante el grado. Este conocimiento permitió un rápido manejo sobre los protocolos y sistemas ya utilizados. Asimismo, para la manipulación del sistema, el proceso se llevó a cabo de manera eficaz gracias a los conocimientos previos adquiridos.

A lo largo del grado, se fortalecieron y adquirieron diversas competencias transversales. En este proyecto, tuvimos la oportunidad de aplicar varias de ellas, las cuales incluyen:

- CT01. Comprensión e integración.
- CT02. Aplicación y pensamiento práctico.
- CT03. Análisis y resolución de problemas.

- CT04. Innovación, creatividad y emprendimiento.
- CT05. Diseño y proyecto.
- CT09. Pensamiento crítico
- CT10. Conocimiento de problemas contemporáneos.
- CT11. Aprendizaje permanente.
- CT12. Planificación y gestión del tiempo.

7.2. Trabajos futuros

El proyecto llevado a cabo no solo logra los objetivos que se propusieron en su inicio, sino que también abre el camino para el desarrollo de versiones más avanzadas, que podrían ofrecer un conjunto aún más amplio de funcionalidades al usuario. Algunas de las propuestas más prometedoras que se podrían explorar en futuros avances de este proyecto se describen a continuación:

- **Incorporación de inteligencia artificial.** Podría ser de gran interés para el usuario que, mediante inteligencia artificial, la cámara pudiera reconocer, en tiempo real, objetos o el movimiento de estos. De modo que, si la cámara capta movimiento o visualiza un objeto, se le notificará al usuario.
- **Manipulación directa con la cámara.** Como se pudo observar en la grabación en directo de una cámara, esta se mantiene firme en un único ángulo. Por ello, estaría ideal hacer un mecanismo para que el usuario pueda girar la cámara, permitiendo cambiar la posición del ángulo.
- **Soporte para almacenamiento en la nube.** Los usuarios almacenan los archivos de las grabaciones en el disco local, por lo que sería perfecto que, si este se queda sin almacenamiento, contara con un servicio de almacenamiento en la nube.

Referencias bibliográficas

[1] ISO/IEC 23009-1. (2012 Dynamic adaptive streaming over HTTP (DASH) – Part 1: media presentation description and segment formats. https://www.iso.org/standard/57623.html

[2] Bouzakaria, N., Concolato, C., & Le Feuvre, J. (2014, July). Overhead and performance of low latency live streaming using MPEG-DASH. In IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications (pp. 92-97). IEEE. https://ieeexplore.ieee.org/abstract/document/6878732

[3] Schulzrinne, H., Rao, A., & Lanphier, R. (1998). Real time streaming protocol (RTSP) (No. rfc2326). https://www.rfc-editor.org/rfc/rfc2326#page-57

[4] iSpy Connect (s.f.). About. https://www.ispyconnect.com/about.aspx

[5] iSpy Connect (s.f.). Agent DVR Feature List. https://www.ispyconnect.com/features.aspx

[6] VideoLAN (s.f.). VLC media player. https://www.videolan.org/index.es.html

[7] MDN (2023, July 3). What is accessibility? https://developer.mozilla.org/es/docs/Learn/Accessibility/What_is_accessibility

[8] Sodagar, I. (2011). The mpeg-dash standard for multimedia streaming over the internet. IEEE multimedia, 18(4), 62-67. https://ieeexplore.ieee.org/abstract/document/6077864

[9] Tomar, S. (2006). Converting video formats with FFmpeg. Linux journal, 2006(146), 10. https://dl.acm.org/doi/fullHtml/10.5555/1134782.1134792

[10] Belda, R., Arce, P., de Fez, I., & Guerri, J. C. (2022, October). Performance Evaluation and Testbed for Delivering SRT Live Content using DASH Low Latency Streaming Systems. In Proceedings of the 19th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (pp. 115-121). https://dl.acm.org/doi/abs/10.1145/3551663.3558674

[11] Solbyte Servicios Informáticos (2021, Mayo 8). Ciclo de vida del software. Qué es, modelos y etapas. https://www.solbyte.com/blog/ciclo-de-vida-del-software/

[12] Aarsten, A., Brugali, D., & Menga, G. (1996). Patterns for three-tier client/server applications. Proceedings of Pattern Languages of Programs (PLoP'96), 4(6). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a2ba59839c9b9f653768595f24db92003fb5dcb2

[13] Chen, P. M., Lee, E. K., Gibson, G. A., Katz, R. H., & Patterson, D. A. (1994). RAID: High-performance, reliable secondary storage. ACM Computing Surveys (CSUR), 26(2), 145-185. https://dl.acm.org/doi/abs/10.1145/176979.176981

[14] R. Belda, I. de Fez, P. Arce, and J. C. Guerri. (2018, Junio). "Look Ahead: a DASH adaptation algorithm," in Proc. of the IEEE Int. Symp. On Broadband Multimedia Systems and Broadcasting (BMSB), Valencia (Spain), article no. 158. https://ieeexplore.ieee.org/document/8436718

[15] R. Belda, I. de Fez, P. Arce, and J. C. Guerri (2018, Septiembre). "Algoritmo de adaptación DASH sensible al tamaño del segmento," in Proc. of the Simposium Nacional de la Unión Científica Internacional de Radio (URSI), Granada (Spain), article S7.1.3.

[16] R. Belda, I. de Fez, P. Arce, and J. C. Guerri (2020, Junio). "Look ahead to improve QoE in DASH streaming," Multimedia Tools and Applications, vol. 79, pp. 25143-25170. https://link.springer.com/article/10.1007/s11042-020-09214-9

[17] Django. (s.f.). Meet Django. https://www.djangoproject.com/

[18] jQuery UI. (s.f.). About jQuery UI. https://jqueryui.com/about/

[19] Bootstrap-datepicker. https://bootstrap-datepicker.readthedocs.io/en/latest/

[20] MariaDB. (2020, Diciembre). Concurrent Inserts. https://mariadb.com/kb/en/concurrent-inserts/

[21] FastAPI. (s.f.). FastAPI. https://fastapi.tiangolo.com/es/

[22] AiomSQL. https://aiomysql.readthedocs.io/en/stable/

[23] Django. (s.f.). Django documentation. https://docs.djangoproject.com/en/4.2/

[24] Hatchful Shopify. https://www.shopify.com/es/herramientas/generador-de-logos

[25] Ron Garrison. (s.f.). Structure of a MPEG-DASH MPD. https://ottverse.com/structure-of-an-mpeg-dash-mpd/

[26] WireGuard. (s.f.). WireGuard. https://www.wireguard.com/

[27] DASH Industry Forum. https://dashif.org/

[28] Wireshark. https://www.wireshark.org/

APÉNDICE A

Objetivos de Desarrollo Sostenible

En este anexo, se ilustrará la correlación del proyecto realizado con los Objetivos de Desarrollo Sostenible (ODS) utilizando como referencia la tabla B.1. A continuación, se brindará una justificación detallada de las conexiones establecidas.

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No procede
ODS1. **Fin de la pobreza.**				X
ODS2. **Hambre cero.**				X
ODS3. **Salud y bienestar.**				X
ODS4. **Educación de calidad.**				X
ODS5. **Igualdad de género.**				X
ODS6. **Agua limpia y saneamiento.**				X
ODS7. **Energía asequible y no contaminante.**				X
ODS8. **Trabajo decente y crecimiento económico.**				X
ODS9. **Industria, innovación e infraestructuras.**	X			
ODS10. **Reducción de las desigualdades.**				X
ODS11. **Ciudades y comunidades sostenibles.**		X		
ODS12. **Producción y consumo responsables.**				X
ODS13. **Acción por el clima.**				X
ODS14. **Vida submarina.**				X
ODS15. **Vida de ecosistemas terrestres.**				X
ODS16. **Paz, justicia e instituciones sólidas.**				X
ODS17. **Alianzas para lograr objetivos.**				X

Tabla B.1: Relación del proyecto con los Objetivos de Desarrollo Sostenible

En relación a los objetivos mencionados anteriormente, el proyecto establece vínculos con:

- **Industria, innovación e infraestructuras:** El proyecto contribuye a este objetivo al desarrollar y fortalecer la infraestructura de tecnología de la información y comunicaciones a través de la plataforma web. Esta plataforma promueve la accesibilidad y la eficiencia en los sistemas de vigilancia y monitorización, creando infraestructuras más sólidas y sostenibles.
- **Ciudades y comunidades sostenibles:** Este proyecto tiene un impacto directo en la meta de "Ciudades y Comunidades Sostenibles" de los Objetivos de Desarrollo Sostenible. Al permitir la transmisión eficiente de video en tiempo real desde cámaras IP a través de la plataforma web, se impulsa el desarrollo de sistemas de seguridad y vigilancia más eficientes. Esto puede potencialmente aumentar la seguridad y la

resiliencia en nuestras ciudades y comunidades. Además, al utilizar una tecnología de transmisión de video que optimiza el uso del ancho de banda y de energía, se está contribuyendo a la sostenibilidad de las infraestructuras digitales.

En resumen, el proyecto contribuye activamente los Objetivos de Desarrollo Sostenible de "Industria, Innovación e Infraestructuras" y "Ciudades y Comunidades Sostenibles". A través de una innovadora plataforma web, se potencia la eficiencia y accesibilidad en sistemas de vigilancia, reforzando la infraestructura de tecnología de la información y comunicaciones. Simultáneamente, al permitir una eficaz transmisión de video en tiempo real desde cámaras IP, se mejora la seguridad y resiliencia de nuestras comunidades. Todo ello, sumado a la optimización del ancho de banda, que impulsa la sostenibilidad de nuestras infraestructuras digitales y contribuye a la construcción de ciudades más seguras y sostenibles.

APÉNDICE B

Glosario

CDN: Una Red de Distribución de Contenidos, o CDN, es una red de servidores que almacenan copias de datos en diferentes puntos de presencia (PoPs) para entregar contenido a los usuarios desde la ubicación más cercana, mejorando así la velocidad y la fiabilidad.

HTTP: Las siglas HTTP corresponden al Protocolo de Transferencia de Hipertexto. Este protocolo es fundamental para cualquier transferencia de datos en la web, permitiendo el intercambio de información en diversas formas, como texto, imágenes y otros tipos de datos.

ICMP: Protocolo de mensajes de control de Internet (Internet Control Message Protocol (ICMP)), de la capa de red, que se utiliza por los dispositivos de red, como los enrutadores, para enviar mensajes de error que indican que un servicio solicitado no está disponible o que un host o enrutador no se puede alcanzar.

IDE: Un Entorno de Desarrollo Integrado, o IDE, es un software que proporciona un conjunto de herramientas a los desarrolladores para facilitar el proceso de codificación. Combina características comunes como edición de código fuente, construcción de código automática, y depuración en una única interfaz gráfica de usuario (GUI).

MPD: MPD se refiere a la Descripción de la Presentación de Medios. Es un archivo que proporciona información sobre el contenido de streaming y cómo está organizado, incluyendo metadatos y la ubicación de los segmentos de medios para que un reproductor pueda reproducirlo de manera eficiente.

Segmento: En el contexto del streaming de medios, un segmento se refiere a una porción de un archivo de audio o video. Los archivos de medios se dividen en estos segmentos más pequeños para facilitar su transmisión a través de Internet y permitir una adaptación eficiente a diferentes velocidades de red.

Sistema de ficheros: Un sistema de ficheros es la estructura que un sistema operativo utiliza para controlar cómo se almacenan los archivos y las carpetas en un dispositivo de almacenamiento. Regula el acceso a los datos y mantiene un seguimiento de dónde se encuentran los archivos.

SDP: El Protocolo de Descripción de Sesión, o SDP, es un formato que describe las capacidades y requisitos de una sesión multimedia. Define las propiedades de la transmisión de medios, como el tipo de medio, el formato, el protocolo de transporte y la dirección IP del origen, entre otros.