UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Creación de una plataforma web de streaming personalizable para la difusión y consumo de contenido audiovisual

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Fernández Beaus, Pablo

Tutor/a: Arce Vila, Pau

CURSO ACADÉMICO: 2023/2024

Resumen

El presente Trabajo de Fin de Grado (TFG) se centra en el desarrollo de una plataforma web de streaming personalizable por el usuario. La plataforma permitirá a los usuarios personalizar la apariencia de la web mediante la selección de logo, color corporativo, fuente, entre otros elementos, para brindar una experiencia única y acorde a sus preferencias. El desarrollo web se realizará utilizando tecnologías front-end como HTML, CSS y JavaScript, garantizando la adaptabilidad de la plataforma a diferentes dispositivos (responsive). Los usuarios podrán subir videos a la plataforma, los cuales serán enviados a una base de datos y recodificados automáticamente para su almacenamiento y distribución a través de una API. De esta manera, los videos estarán accesibles para el resto de usuarios de la web.

Resum

Aquest Projecte de Fi de Grau se centra en el desenvolupament d'una plataforma de transmissió en línia personalitzable per l'usuari. La plataforma permetrà als usuaris personalitzar l'aparença del lloc web seleccionant un logotip, un color corporatiu, una font, entre altres elements, per proporcionar una experiència única i personalitzada. El desenvolupament web es durà a terme utilitzant tecnologies de front-end com HTML, CSS i JavaScript, garantint l'adaptabilitat de la plataforma a diferents dispositius (responsiva). Els usuaris podran carregar vídeos a la plataforma, els quals seran enviats a una base de dades i recodificats automàticament per al seu emmagatzematge i distribució a través d'una API. D'aquesta manera, els vídeos seran accessibles per a la resta d'usuaris del web.

Abstract

This Final Degree Project focuses on the development of a user-customizable web streaming platform. The platform will allow users to customize the appearance of the website by selecting a logo, corporate color, font, among other elements, to provide a unique and personalized experience. The web development will be carried out using front-end technologies such as HTML, CSS and JavaScript, ensuring the platform's adaptability to different devices (responsive). Users will be able to upload videos to the platform, which will be sent to a database and automatically recoded for storage and distribution through an API. In this way, the videos will be accessible to the rest of the web users.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Tecnología Digital y Multimedia debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la tecnologías digitales y multimedia

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
IDENTIFY:	1. IDENTIFICAR:	**S**	
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	**S**	4
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	**S**	11-13
1.3. Setting of goals	1.3. Establecimiento de objetivos	**S**	6
FORMULATE:	2. FORMULAR:	**S**	
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	**S**	17-24
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	**S**	9-10
SOLVE:	3. RESOLVER:	**S**	
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	**S**	52
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	**S**	53

Índice

Índice de figuras

Capítulo 1. Introducción

1.1 Contexto actual

Hoy en día, el consumo de contenido audiovisual ha experimentado un profundo cambio y sigue evolucionando a pasos agigantados. En poco más de 20 años, hemos pasado de los medios tradicionales como la televisión o el cine, a plataformas digitales que nos permiten acceder a millones de contenidos al alcance de un clic. Esto debido en gran medida, al crecimiento exponencial de internet y a la proliferación de dispositivos móviles u ordenadores personales cada vez más accesibles.

Uno de los servicios que más se han popularizado en estos últimos tiempos es el de las plataformas de streaming y el vídeo bajo demanda. Plataformas como Netflix o Amazon Prime Video entre muchas otras, han ganado sobradamente su pelea ante la televisión tradicional, y es que la gente prefiere elegir los contenidos que quiere reproducir en cualquier momento y lugar, a tener que lidiar con una programación establecida y molestos anuncios cada poco rato.

Y no solo en la industria del cine o la televisión, plataformas como YouTube o Twitch, se han convertido en parte de nuestra vida cotidiana, ofreciendo no solo entretenimiento, sino también educación, información y una manera de conectarse con comunidades de todo el mundo.

Este cambio en los hábitos de consumo ha generado una creciente demanda por soluciones más personalizables y flexibles, que permitan a los usuarios tener un mayor control sobre cómo y cuándo acceden a su contenido favorito.

En este contexto, el trabajo surge de la necesidad de desarrollar una plataforma de streaming que fuera más allá de lo que ya existe, permitiendo a los usuarios no solo consumir contenido, sino también personalizar su experiencia de una manera significativa. Imagina una plataforma donde puedas elegir el logo, los colores, la tipografía y hasta cómo se codifican tus vídeos, todo ajustado a tus preferencias o a la imagen de tu marca. Esta capacidad de personalización no solo mejora la experiencia del usuario, sino que también proporciona a las organizaciones y creadores independientes una herramienta para diferenciarse y ofrecer algo único.

Además, este proyecto también se enfoca en proporcionar una solución viable para aquellos que desean distribuir su propio contenido sin depender de las grandes plataformas. Esto es especialmente útil para instituciones educativas, empresas y creadores independientes que necesitan una plataforma profesional para sus vídeos, pero que no tienen los recursos para desarrollar una desde cero, y no quieren subir su contenido clasificado y privado a la nube o a servidores ajenos a la empresa o institución.

1.2 Relación con asignaturas del grado

De todo lo aprendido durante la realización del grado en Tecnología Digital y Multimedia en la UPV, hay varias asignaturas clave que han sido fundamentales para la realización de este proyecto.

La primera de ellas es la asignatura de Plataformas de Streaming, cursada en el tercer año del grado. Los conocimientos obtenidos acerca de las distintas etapas que forman parte de una plataforma de streaming bajo demanda, desde la preparación de contenidos, hasta la explotación final, pasando por la fase de codificación, encriptación y difusión, han sido de vital importancia a la hora de a la hora de diseñar y desarrollar este proyecto. Entender cómo funcionan estos procesos internos ha permitido crear una plataforma eficiente y funcional.

En cuanto al diseño web, tanto del front-end como parte del back-end, es gracias a la asignatura de Tecnologías Web, la cual se imparte en el segundo curso de la carrera. Esta asignatura me proporcionó las bases de la programación web, haciendo uso de las tecnologías HTML, JavaScript y CSS, gracias a las cuales pude adquirir conocimientos y habilidades esenciales para desarrollar una plataforma web interactiva y estéticamente agradable.

Finalmente, en lo referido las competencias en bases de datos relacionales, el taller de Bases de Datos SQL, parte de la asignatura de Talleres y Seminarios de Tecnologías emergentes II durante el tercer curso, ha jugado un papel muy importante. Si bien es cierto que se trató de un taller relativamente breve, me sirvió como toma de contacto con el lenguaje SQL y sobre todo con MySQL, que es el sistema utilizado en este proyecto.

Capítulo 2. Objetivos

Para el desarrollo de este trabajo, me he propuesto una serie de objetivos específicos que ha de cumplir mi proyecto para alinearse con mi idea de ofrecer una plataforma de streaming flexible y personalizable que se adapte a las necesidades específicas de cualquier usuario u organización.

El primer objetivo es que la plataforma tenga un alto nivel de personalización. No se trata solo de ofrecer una solución genérica, sino de permitir que cada usuario o cliente pueda ajustar la plataforma para que refleje su identidad visual y necesidades específicas. Para ello, deberá permitir modificar diferentes elementos visuales, como el logo, los colores corporativos, el estilo de los botones, el título de la página…etc. Y que así la plataforma sea única y diferente para cada cliente.

El segundo objetivo es que la reproducción de los vídeos en la plataforma sea adaptativa. Es decir, que cuando un usuario quiera ver un vídeo, la calidad se ajuste automáticamente según la velocidad de su conexión a Internet. Esto es importante para que la experiencia de visualización sea siempre fluida, sin importar si la conexión es rápida o lenta. Así, los usuarios pueden disfrutar del contenido sin interrupciones, lo que mejora su experiencia y hace que la plataforma funcione mejor en general.

El tercer y último objetivo es que la plataforma se encargue de la codificación de los vídeos de forma automática. Esto significa que los usuarios no tienen que preocuparse por subir vídeos ya codificados. La plataforma se encarga de tomar los vídeos, procesarlos, fragmentarlos y generar los archivos necesarios para que se puedan reproducir de manera adaptativa.

Capítulo 3. Metodología

3.1 Estado del arte

3.1.1 Tecnologías utilizadas

Para el desarrollo de la plataforma, se han empleado una serie de tecnologías sobre las cuales hablaremos a continuación.

- HTML

HTML (HyperText Markup Language) [1] es el lenguaje estándar utilizado para crear y estructurar el contenido en la web. Se ha utilizado para definir la estructura básica de las páginas de la plataforma.

- CSS

CSS (Cascading Style Sheets) [2] se ha empleado para dar estilo y formato a las páginas HTML. Gracias a CSS, se ha podido definir la apariencia visual de la plataforma, desde los colores y las fuentes hasta el diseño responsivo que asegura que la plataforma se vea bien en cualquier dispositivo, ya sea un ordenador de escritorio, una tableta o un móvil.

- JavaScript

JavaScript [3] es un lenguaje de programación que permite añadir interactividad a las páginas web. En este proyecto, JavaScript se ha utilizado para manejar eventos del usuario, validar formularios o interactuar con el servidor entre otras cosas.

- Node.js

Node.js [4] es un entorno de ejecución para JavaScript que permite ejecutar código JavaScript en el servidor. Node.js ha sido fundamental para el backend de la plataforma, proporcionando un entorno eficiente y escalable para manejar las solicitudes HTTP, la gestión de archivos y la interacción con la base de datos. Se ha utilizado junto con módulos como Express para la creación de rutas y manejo de solicitudes, Multer para la gestión de archivos subidos, y MySQL para la conexión y manipulación de la base de datos.

- MySQL

MySQL [6] es un sistema de gestión de bases de datos relacional. En este proyecto, MySQL se ha utilizado para almacenar y gestionar la información relacionada con los vídeos, usuarios y configuraciones de la plataforma. La base de datos permite realizar consultas eficientes y almacenar datos de manera estructurada.

- Shaka Player

Shaka Player [8] es una biblioteca de JavaScript que permite la reproducción de vídeos en streaming. Se ha utilizado en este proyecto para proporcionar una experiencia de reproducción de vídeo fluida y de alta calidad, soportando formatos modernos como MPEG-DASH y HLS.

- FFMpeg

FFMpeg [9] es una herramienta de línea de comandos para procesar archivos multimedia. En este proyecto, FFMpeg se ha utilizado para la codificación de vídeos, permitiendo convertirlos a diferentes formatos y resoluciones, optimizando así su distribución y reproducción en la plataforma.

- Bento4

Bento4 [10] es un conjunto de herramientas y bibliotecas diseñadas para la creación y manipulación de archivos multimedia en formato MP4. Dentro de este conjunto, se encuentra la herramienta mp4fragment, que se ha utilizado en este proyecto para fragmentar los archivos de vídeo MP4. Esta fragmentación es un paso crucial para preparar los vídeos para su empaquetado en el formato de transmisión DASH (Dynamic Adaptive Streaming over HTTP). El uso de Bento4 asegura que los vídeos estén correctamente estructurados para la reproducción adaptativa, donde la calidad del vídeo se ajusta dinámicamente en función de las condiciones de la red del usuario.

- DASH

DASH (Dynamic Adaptive Streaming over HTTP) [11] es un estándar de transmisión de contenido multimedia adaptativo [13] que permite la reproducción de vídeos de manera continua, ajustando automáticamente la calidad del vídeo en función de las condiciones de la red y el rendimiento del dispositivo. En este proyecto, DASH ha sido esencial para ofrecer una experiencia de streaming optimizada, donde los usuarios puedan disfrutar de vídeos sin interrupciones, incluso en condiciones de conexión variables.

Figura 1. Tecnologías utilizadas.

3.1.2 Justificación de elecciones

La mayoría de las tecnologías que he elegido para este proyecto se basan en la familiaridad que he ido adquiriendo con ellas a lo largo de mis estudios. Investigando posibles alternativas, me di cuenta de que usar tecnologías nuevas solo añadiría más dificultades debido a la necesidad de aprender a manejarme con ellas primero, mientras que con las de la carrera ya he obtenido soltura y me daban una mayor confianza.

Sin embargo, en aquellas tecnologías que no vimos en la carrera, como Node.js, o en las que apenas profundizamos, como MySQL, sí que llevé a cabo un proceso de investigación y comparación para decidir cuál era la mejor opción.

3.1.2.1 Node.js vs Deno

En mi proyecto, uso Node.js para gestionar el servidor de la plataforma de streaming. Node.js maneja todas las solicitudes HTTP, la carga de vídeos, la gestión de usuarios y la interacción con la base de datos.

Deno era la alternativa principal que consideré [5], el cual es un entorno de ejecución más moderno que aborda varias de las limitaciones de Node.js, especialmente en términos de seguridad y soporte nativo para TypeScript.

Ventajas de Deno:

- Soporte nativo para TypeScript.
- Mayor seguridad por defecto.
- Sistema de módulos basado en URL.

Desventajas de Deno:

- Ecosistema más pequeño.
- Menos maduro.
- Curva de aprendizaje más pronunciada para quienes vienen de Node.js.

Conclusión

Elegí Node.js para este proyecto principalmente por su ecosistema maduro y su amplia comunidad, que ofrecen un gran soporte y recursos. Aunque Deno presenta mejoras atractivas, la familiaridad y el entorno robusto de Node.js se alinean mejor con las necesidades de mi proyecto, asegurando un rendimiento óptimo y una gestión eficiente de datos. Además, la información y los recursos disponibles para Node.js son mucho más abundantes, lo que facilita la resolución de problemas y la implementación de nuevas características.

Figura 2. Node.js vs Deno.

3.1.2.2 MySQL vs SQLite

En mi proyecto, uso MySQL para almacenar la información de los usuarios, los vídeos y las configuraciones de la plataforma. Su capacidad para manejar grandes volúmenes de datos y realizar consultas complejas de manera eficiente es fundamental para el funcionamiento de la plataforma.

SQLite era la otra opción que consideré [7]. Es una base de datos relacional ligera que es fácil de configurar y usar.

Ventajas de SQLite:

- Muy ligera y fácil de configurar.
- Ideal para aplicaciones pequeñas o integradas.
- No requiere un servidor separado.

Desventajas de SQLite:

- No adecuada para aplicaciones con altas cargas de escritura o alta concurrencia.
- Menos funcionalidad avanzada en comparación con MySQL.

Conclusión

Elegí MySQL debido a su robustez y capacidad para manejar volúmenes de datos más grandes y complejos. Aunque SQLite es excelente para aplicaciones más simples, la necesidad de un manejo más avanzado de los datos y la seguridad inclinó la balanza a favor de MySQL. MySQL asegura una gestión eficiente de la información y proporciona una plataforma de streaming confiable y escalable.

Figura 3. MySQL vs SQLite.

3.2 Análisis de requisitos

El análisis de requisitos es un paso crucial en cualquier proyecto, ya que permite definir las necesidades y expectativas que el sistema debe cumplir para ser considerado exitoso. En mi plataforma de streaming, he dividido estos requisitos en funcionales y no funcionales. También he detallado las diferencias entre los tipos de usuarios que interactuarán con la plataforma. Además, incluiré un diagrama de actores y casos de uso para visualizar mejor cómo interactúan estos elementos y facilitar la comprensión de su funcionamiento.

3.2.1 Requisitos funcionales

Los requisitos funcionales son las capacidades y funciones específicas que el sistema debe tener para cumplir con las expectativas del usuario. En mi plataforma de streaming, estos requisitos se dividen en dos categorías principales: los que competen a los administradores y los que competen a los usuarios.

3.2.1.1 Administradores

- Subir Vídeos:

Los administradores deben poder subir vídeos a la plataforma. Al subir un vídeo, se requiere que el administrador cargue tanto el archivo de vídeo como una miniatura que lo acompañe, además de elegir un título y descripción.

- Reproducir Vídeos:

Los administradores deben poder reproducir cualquier vídeo disponible en la plataforma. Esta funcionalidad es importante para verificar la calidad del contenido, lo cual es útil para identificar y corregir posibles problemas. También permite a los administradores disfrutar del contenido de la misma manera que lo harían los usuarios. La reproducción de vídeos debe ser adaptativa, ajustándose automáticamente a la calidad de la conexión del usuario, para garantizar una experiencia visual continua y de alta calidad en diversas condiciones de red.

- Eliminar Vídeos:

La capacidad de eliminar vídeos es otra funcionalidad clave para los administradores. Esto les permite gestionar el contenido de manera efectiva, asegurando que solo el material apropiado y relevante esté disponible para los usuarios. La eliminación de videos es necesaria para quitar contenido obsoleto, incorrecto o inapropiado.

- Borrar Todos los Vídeos:

En algunos casos, puede ser necesario limpiar completamente la base de datos y eliminar todos los vídeos de la plataforma. Esta funcionalidad permite a los administradores realizar esta acción de manera eficiente, especialmente útil en casos de reinicio de la plataforma o reestructuración completa del contenido. La capacidad de borrar todos los vídeos asegura que los administradores puedan manejar situaciones extremas de mantenimiento o reconfiguración del sistema.

- Personalización de la Plataforma:

Los administradores deben tener la capacidad de personalizar la plataforma. Esta función es fundamental para que la plataforma pueda adaptarse a diferentes identidades visuales y necesidades de los usuarios. Esta capacidad de personalización es la esencia del proyecto y lo que nos hará destacar entre los competidores. Al permitir que cada usuario u organización ajuste la plataforma a sus propias necesidades, ofrecemos una solución práctica y adaptable que puede cumplir con una amplia variedad de requerimientos y gustos.

La personalización permite ajustar varios aspectos de la plataforma, tales como:

- Logo: Los administradores pueden cargar un logo en formato PNG para que se muestre en la interfaz de la plataforma. Esto permite a las organizaciones y creadores de contenido personalizar la plataforma con su propia marca.

- Colores Corporativos: La plataforma permite seleccionar colores primarios, secundarios y terciarios. Estos colores se aplican a diferentes elementos de la interfaz, asegurando una apariencia coherente y personalizada.

- Título y Subtítulo: Los administradores pueden establecer un título y un subtítulo para la plataforma.

- Fuente: Es posible elegir entre varias opciones de fuentes para los textos de la plataforma, así como activar la opción de letra en cursiva.

- Color de los Botones: Se puede seleccionar un color específico para los botones de la interfaz, asegurando que se alineen con el esquema de colores deseado.

- Color de la Fuente: La plataforma permite seleccionar el color de la fuente para todos los textos, asegurando una buena legibilidad y coherencia estética.

- Calidad de Codificación: Los administradores pueden elegir la calidad máxima de codificación para los vídeos (720p, 1080p, 2160p). Esta opción permite balancear la calidad de los vídeos con el tiempo y recursos necesarios para su procesamiento.

Los administradores deben tener la capacidad de personalizar la plataforma al iniciar por primera vez. Este proceso se realiza a través de un asistente (wizard) de personalización, que guía al administrador paso a paso para configurar varios aspectos de la plataforma.

Además, deben tener la capacidad de editar la personalización de la plataforma más adelante. Esto se realiza a través de un modal de personalización que permite hacer ajustes sin necesidad de pasar por todo el asistente nuevamente.

3.2.1.2 Usuarios

- Reproducir Vídeos:

Los usuarios normales deben poder reproducir los vídeos disponibles en la plataforma. Esta es la función principal para este tipo de usuario. La reproducción de vídeos debe ser adaptativa, ajustándose automáticamente a la calidad de la conexión del usuario, para garantizar una experiencia visual continua y de alta calidad en diversas condiciones de red. Los usuarios no necesitan funciones de administración, sino simplemente acceso a una biblioteca de contenido para su consumo.

3.2.2 Requisitos no funcionales

Los requisitos no funcionales se refieren a la calidad del sistema y cómo debe funcionar. Estos requisitos son cruciales para garantizar que la plataforma no solo cumpla con sus objetivos, sino que también ofrezca una buena experiencia de usuario.

1. Rendimiento:

   La plataforma debe ser capaz de manejar múltiples usuarios accediendo y reproduciendo vídeos simultáneamente sin problemas de rendimiento. Esto es fundamental para asegurar que todos los usuarios tengan una experiencia fluida, especialmente en momentos de alta demanda.

2. Escalabilidad:

   La plataforma debe ser escalable, lo que significa que debe poder crecer para soportar un número creciente de vídeos y usuarios. Esto es importante para asegurar la longevidad y relevancia de la plataforma a medida que crece. La escalabilidad garantiza que la plataforma pueda adaptarse a futuras expansiones sin necesidad de una reestructuración completa. Esto incluye la capacidad de añadir nuevos servidores o mejorar la infraestructura existente para manejar una mayor carga de trabajo.

3. Usabilidad:

   La interfaz debe ser intuitiva y fácil de usar tanto para administradores como para usuarios normales. Una buena usabilidad asegura que los usuarios puedan navegar y utilizar la plataforma sin frustraciones. Esto incluye un diseño claro, navegación sencilla y tiempos de respuesta rápidos. La usabilidad es crucial para mantener a los usuarios comprometidos y satisfechos, reduciendo la curva de aprendizaje y mejorando la accesibilidad.

4. Seguridad:

   Los datos del sistema deben estar protegidos contra accesos no autorizados y pérdidas. Esto es crucial para proteger la integridad de la plataforma y la privacidad de los usuarios. La seguridad debe incluir medidas como encriptación de datos, autenticación segura y copias de seguridad regulares. Un sistema seguro protege tanto a los usuarios como a la integridad del contenido y la plataforma misma.

3.3 Actores y Casos de Uso

Para entender mejor cómo interactúan los diferentes usuarios con la plataforma de streaming y qué funcionalidades necesitan, se ha elaborado un análisis detallado de actores y casos de uso. Los actores representan los diferentes tipos de usuarios del sistema, y los casos de uso describen las acciones que estos usuarios pueden realizar.

3.3.1 Actores

En nuestra plataforma de streaming, existen dos actores principales:

1. **Administrador**:

   El administrador tiene acceso completo a todas las funciones de la plataforma. Este rol es fundamental para gestionar y mantener el contenido, así como para personalizar la interfaz y asegurarse de que todo funcione correctamente.

2. **Usuario**:

   El usuario es el consumidor final del contenido. Este rol se centra en acceder y reproducir los vídeos disponibles en la plataforma.

3.3.2 Casos de Uso

Los casos de uso detallan las interacciones entre los actores y el sistema. A continuación, se presentan los principales casos de uso para cada actor.

3.3.2.1 Casos de Uso del Administrador

- **Personalización Inicial:**

Descripción: El administrador personaliza la plataforma durante la configuración inicial utilizando un asistente paso a paso.

Objetivo: Adaptar la plataforma a la identidad visual de la organización o a las preferencias del administrador.

Flujo de trabajo:

1. El administrador inicia el asistente de personalización al configurar la plataforma por primera vez.
2. El sistema guía al administrador a través de pasos para cargar el logo, seleccionar colores, establecer títulos y subtítulos, elegir la fuente, definir colores de botones y texto, y seleccionar la calidad de codificación.
3. El administrador realiza las selecciones y configuraciones necesarias.
4. El sistema aplica las configuraciones y actualiza la interfaz de usuario.

- **Editar personalización:**

Descripción: El administrador puede modificar la personalización de la plataforma en cualquier momento a través de un modal de edición.

Objetivo: Permitir ajustes y cambios continuos en la apariencia de la plataforma.

Flujo de trabajo:

1. El administrador abre el modal de personalización pulsando en el botón de editar personalización.
2. El sistema muestra las opciones de personalización actuales.
3. El administrador realiza los cambios necesarios.
4. El sistema aplica las configuraciones y actualiza la interfaz de la plataforma en tiempo real.

- **Subir vídeo:**

Descripción: El administrador puede cargar nuevos vídeos a la plataforma.

Objetivo: Proveer contenido nuevo para los usuarios.

Flujo de trabajo:

1. El administrador abre el modal de subir vídeo pulsando en el botón de subir vídeo.
2. El sistema solicita archivo de vídeo, miniatura, título y descripción.
3. El administrador carga los archivos.
4. El sistema procesa el vídeo, generando diferentes calidades.
5. El sistema guarda los detalles del vídeo en la base de datos.
6. El vídeo se pone a disposición de los usuarios en la plataforma.

- **Reproducir vídeo:**

Descripción: El administrador puede reproducir cualquier vídeo disponible en la plataforma para verificar su calidad y funcionalidad.

Objetivo: Asegurar que el contenido esté en buen estado y disponible para los usuarios.

Flujo de trabajo:

1. El administrador selecciona un vídeo de la biblioteca.
2. El vídeo se reproduce en el reproductor de la plataforma de manera adaptativa.

- **Eliminar vídeo:**

Descripción: El administrador puede eliminar vídeos de la plataforma que ya no son necesarios o que requieren ser eliminados por alguna razón.

Objetivo: Mantener el catálogo de vídeos actualizado y sin errores.

Flujo de trabajo:

1. El administrador selecciona un vídeo de la biblioteca.
2. El administrador pulsa en el botón de eliminar vídeo.
3. El sistema solicita confirmación para la eliminación.
4. El administrador confirma la acción.
5. El sistema elimina el vídeo de la base de datos y del servidor.

- **Eliminar todos los vídeos:**

Descripción: El administrador puede optar por borrar todos los vídeos de la plataforma.

Objetivo: Permitir una limpieza completa de la base de datos y del contenido almacenado.

Flujo Principal:

1. El administrador pulsa el botón de borrar todos los vídeos.
2. El sistema solicita confirmación para la eliminación.
3. El administrador confirma la acción.
4. El sistema elimina todos los vídeos de la base de datos y del servidor.

3.3.2.2 Casos de Uso del Usuario

- **Reproducir Vídeo:**

Descripción: Los usuarios pueden reproducir los vídeos disponibles en la plataforma.

Objetivo: Consumir contenido audiovisual de la plataforma.

Flujo de trabajo:

1. El usuario selecciona un vídeo de la biblioteca.
2. El vídeo se reproduce en el reproductor de la plataforma de manera adaptativa.

3.3.3 Diagrama de Actores y Casos de Uso

Para ilustrar mejor estas interacciones, a continuación, se presenta un diagrama de actores y casos de uso. Este diagrama visualiza las funciones principales de la plataforma y muestra las interacciones entre los diferentes tipos de usuarios y el sistema.

Figura 4. Diagrama de actores y casos de uso.

Capítulo 4. Diseño

El diseño de la plataforma es fundamental para asegurar una experiencia de usuario fluida y satisfactoria.

Una interfaz bien diseñada no solo facilita la navegación y el uso eficiente de la plataforma, sino que también mejora enormemente la experiencia del usuario. La personalización es un elemento clave de esta plataforma, permitiendo a los administradores adaptar el aspecto y la funcionalidad de la plataforma según sus necesidades específicas.

Este capítulo sirve como una planificación detallada y un boceto de cómo se espera que funcione y se vea la plataforma una vez desarrollada. A continuación, se detallan las pantallas principales y los modales que forman parte de la interfaz, así como los bocetos correspondientes.

4.1 Pantallas

4.1.1 Pantallas Administrador

En primer lugar, describiremos las diferentes pantallas de la interfaz por las que navegará un administrador.

4.1.1.1 Pantalla de personalización inicial (Wizard de personalización)

Esta pantalla se mostrará únicamente cuando un administrador inicie la plataforma por primera vez y servirá como un asistente que guiará al administrador paso a paso durante la configuración inicial.

En primer lugar, aparecerá en pantalla un mensaje de bienvenida, el cual desaparecerá gradualmente y dará paso a un formulario de personalización.

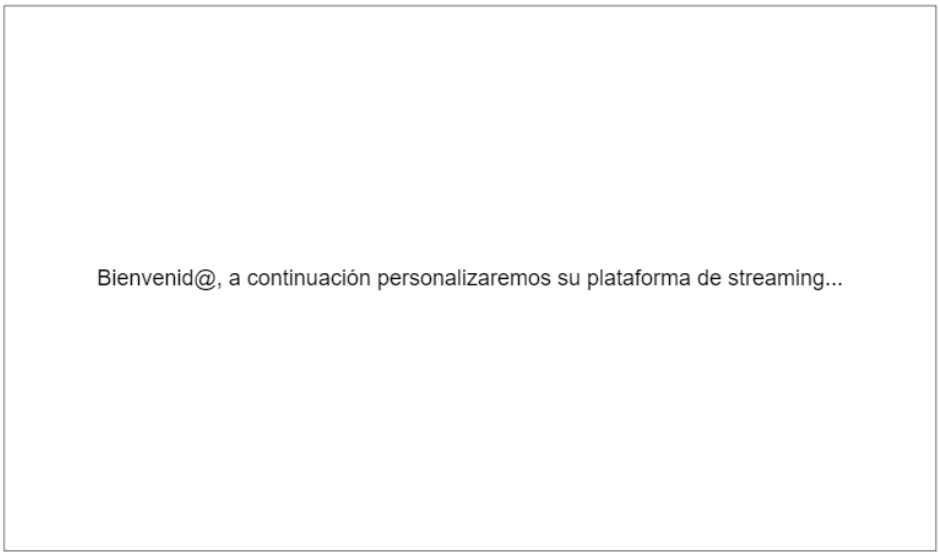

Figura 5. Boceto mensaje de bienvenida.

Este formulario contará con botones de ‘Anterior’ y ‘Siguiente’, que nos permitirán navegar entre los diferentes pasos de la personalización.

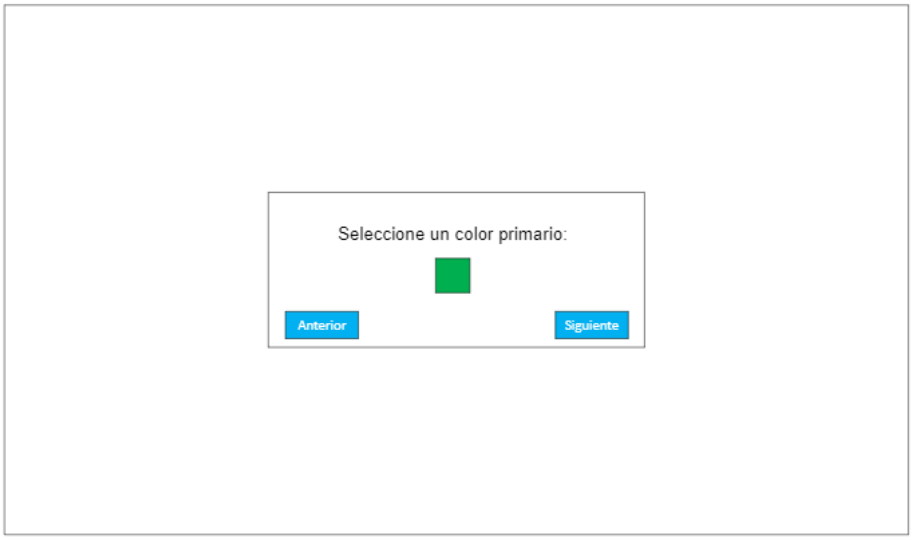

Figura 6. Boceto paso dentro del wizard de personalización (Selección de color primario).

Una vez hayamos completado el resto de los pasos, llegaremos al último, el cual nos pedirá elegir una calidad máxima de codificación y contará con los botones ‘Anterior’ y ‘Comenzar’.

Figura 7. Boceto paso final dentro del modal de personalización (Selección calidad máxima).

Si damos por finalizada la personalización de la plataforma, pulsaremos en el botón de ‘Comenzar’, el cual nos lanzará a la plataforma como administrador.

4.1.1.2 Plataforma de streaming

Una vez iniciada la plataforma, el administrador visualizará tres botones en la parte superior de la pantalla:

- 'Subir vídeo': que permitirá subir un vídeo nuevo a la plataforma
- 'Editar personalización': que permitirá editar los valores de personalización previamente elegidos en el wizard.
- 'Eliminar todos los vídeos': que permitirá eliminar todos los vídeos de la plataforma.

Si se trata de la primera vez que se inicia la plataforma y aun no se han subido vídeos, en el lugar donde se encontraría el catálogo de vídeos aparecerá un mensaje indicando que aun no hay ninguno subido.

Figura 8. Boceto plataforma de streaming como Administrador sin vídeos subidos.

Si, por el contrario, no es la primera vez y si que hay vídeos subidos, se mostrarán a modo de catálogo de la siguiente manera:

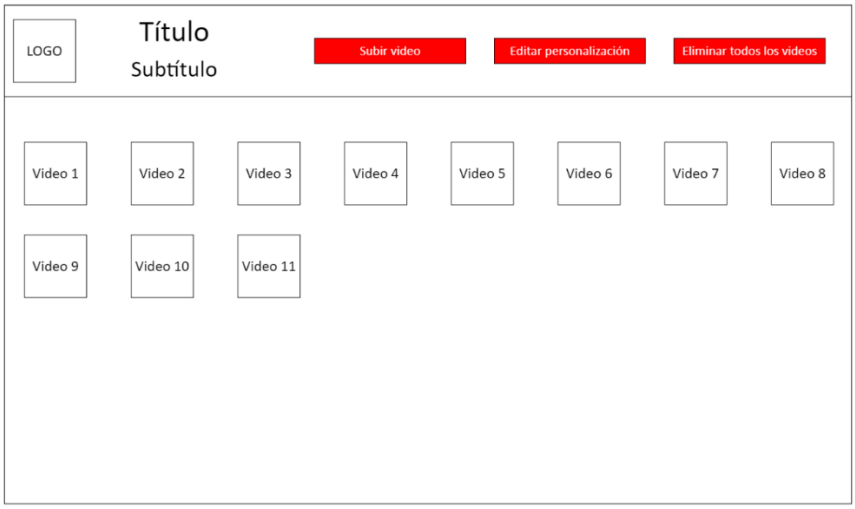

Figura 9. Boceto plataforma de streaming como Administrador con vídeos subidos.

Pulsando cualquiera de los botones mencionados, accederemos a diferentes modales dentro de la plataforma, de los cuales hablaremos a continuación.

4.1.1.2.1 Modal de subida de vídeo

Si pulsamos el botón de subir vídeo, se nos abrirá un modal de subida con una serie de campos a cumplimentar:

- Vídeo en .mp4
- Miniatura
- Título
- Descripción

Dentro del modal tendremos dos botones, 'Agregar Vídeo' y 'Cancelar'.

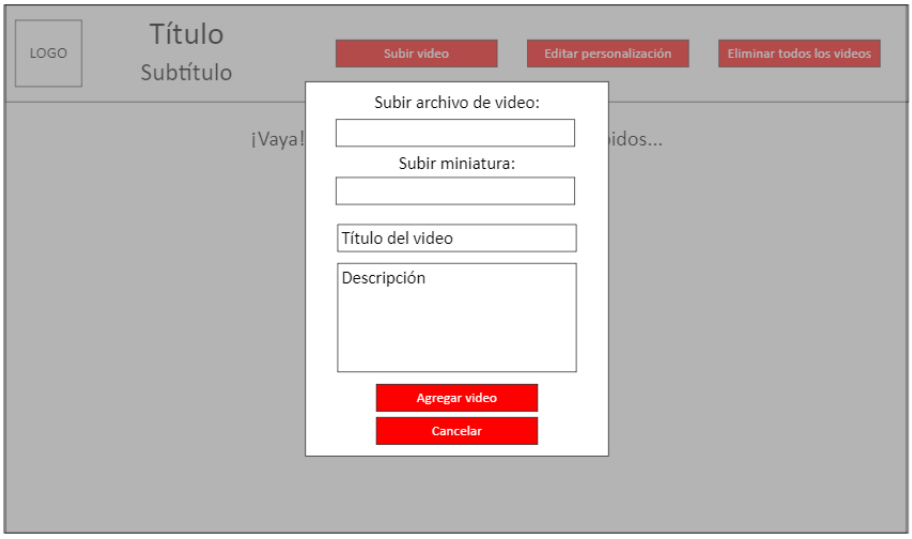

Figura 10. Boceto modal de subida de vídeos.

4.1.1.2.2 Modal de edición de personalización

Si pulsamos el botón de editar personalización, se nos abrirá un modal con exactamente los mismos campos que seleccionamos en el wizard de personalización.

Dentro del modal tendremos dos botones, 'Guardar' y 'Cancelar', y una barra de desplazamiento vertical.

Figura 11. Boceto modal de edición de personalización.

4.1.1.2.3 Modal de eliminación de todos los vídeos

Si pulsamos el botón de eliminar todos los vídeos, se nos abrirá un modal que nos solicitará la confirmación de la acción.

Dentro del modal tendremos dos botones, 'Sí' y 'No'.

Figura 12. Boceto modal de eliminación de todos los vídeos.

4.1.1.2.4 Modal de reproducción de vídeos como Administrador

Si pulsamos en un vídeo del catálogo, se nos abrirá un modal de reproducción con el reproductor ShakaPlayer y el título y descripción del vídeo.

Dentro del modal tendremos dos botones, 'Eliminar' y 'Volver'.

Figura 13. Boceto modal de reproducción de vídeos como Administrador.

Si pulsamos el botón de eliminar, se nos abrirá un modal que nos solicitará la confirmación de la acción.

Figura 14. Boceto modal de eliminación de un vídeo.

4.1.2 Pantallas Usuario

A continuación, describiremos las diferentes pantallas de la interfaz por las que navegará un usuario.

4.1.2.1 Plataforma de streaming

El usuario visualizará la misma plataforma que el administrador, pero sin las funciones propias de este.

Es decir, que la interfaz visual será idéntica y con los valores de personalización elegidos por el administrador, pero no existirán ninguno de los botones vistos en la Figura 8 o la Figura 9.

Al igual que para los administradores, si aún no hay vídeos subidos a la plataforma, en el lugar donde se encontraría el catálogo de vídeos aparecerá un mensaje indicando que aún no hay ninguno subido.

Figura 15. Boceto plataforma de streaming como Usuario sin vídeos subidos.

Si, por el contrario, hay vídeos subidos, se mostrarán a modo de catálogo de la siguiente manera:

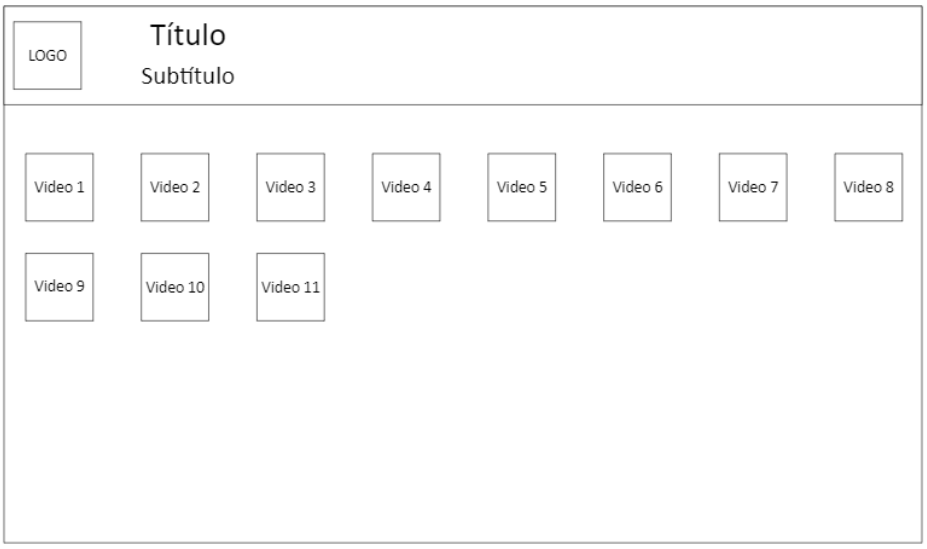

Figura 16. Boceto plataforma de streaming como Usuario con vídeos subidos.

4.1.2.1.1 Modal de reproducción de vídeos como Usuario

Si pulsamos en un vídeo del catálogo, se nos abrirá un modal de reproducción con el reproductor ShakaPlayer y el título y descripción del vídeo.

Dentro del modal tendremos el botón 'Volver'.

Figura 17. Boceto modal de reproducción de vídeos como Usuario.

4.2 Base de datos

La base de datos del proyecto está diseñada para almacenar tanto las configuraciones de personalización de la plataforma como la información de los vídeos subidos. A continuación, se detalla la estructura y el propósito de cada tabla.

4.2.1 Tabla 'settings'

La tabla settings almacena las configuraciones de personalización de la plataforma. Cada registro representa una configuración específica de la plataforma, permitiendo cambios en la apariencia de la interfaz.

Campo	Tipo	Descripción
id	INT	Identificador único de la configuración
logo	TEXT	Ruta del logo de la plataforma
color	VARCHAR(255)	Color primario de la plataforma
secondaryColor	VARCHAR(255)	Color secundario de la plataforma
tertiaryColor	VARCHAR(255)	Color terciario de la plataforma
title	VARCHAR(255)	Título de la plataforma
subtitle	VARCHAR(255)	Subtítulo de la plataforma
font	VARCHAR(255)	Fuente utilizada en la plataforma
fontColor	VARCHAR(255)	Color de la fuente
isItalic	BOOLEAN	Indica si la fuente es cursiva (true/false)
buttonColor	VARCHAR(255)	Color de los botones
quality	VARCHAR(10)	Calidad máxima de codificación

Tabla 1. Estructura de la tabla settings.

4.2.2 Tabla 'videos'

La tabla videos almacena la información sobre los vídeos subidos a la plataforma. Cada registro representa un vídeo específico, incluyendo detalles como el título, la descripción y el estado de procesamiento.

Campo	Tipo	Descripción
id	INT	Identificador único del vídeo
title	VARCHAR(255)	Título del vídeo
description	TEXT	Descripción del vídeo
thumbnail	VARCHAR(255)	Ruta de la miniatura del vídeo
manifest	VARCHAR(255)	Ruta del archivo MPD
processing	BOOLEAN	Indica si el vídeo está en proceso
processed	BOOLEAN	Indica si el vídeo ha sido procesado

Tabla 2. Estructura de la tabla videos.

Capítulo 5. Desarrollo y resultados del trabajo

En este capítulo, se describirá el desarrollo del proyecto, detallando cómo se implementaron tanto el front-end como el back-end. Explicaré las tecnologías y lenguajes de programación utilizados, y cómo se estructuraron las distintas partes del sistema para garantizar un funcionamiento eficiente. Nos centraremos en el proceso de construcción de las funcionalidades clave y en cómo cada componente del proyecto se integra para formar una plataforma completa y operativa.

5.1 Front-end

En esta sección abordaremos en detalle el proceso de desarrollo del front-end de la plataforma de streaming. El objetivo principal del front-end es proporcionar una interfaz de usuario intuitiva, visualmente atractiva y funcional que permita a los usuarios interactuar de manera sencilla con la plataforma.

El desarrollo del front-end se llevó a cabo utilizando HTML, CSS y JavaScript. La estructura base de la interfaz está definida en HTML, mientras que CSS se encarga del diseño visual y la disposición de los elementos en la pantalla.

A continuación, veremos los diferentes archivos del proyecto pertenecientes al front-end y el papel que desempeñan dentro de este, además de analizar más en profundidad ciertas funciones clave del código.

5.1.1 Front-end del Wizard de personalización

El wizard o modal de personalización, está compuesto por los siguientes tres archivos:

- Index.html: Este archivo define la estructura básica del wizard, incluyendo la disposición de los elementos en la pantalla. Contiene el esqueleto de la página, donde se encuentran las secciones para cargar el logo, seleccionar colores, introducir un título y subtítulo, escoger la fuente, y ajustar otros parámetros visuales. Cada sección está organizada en pasos, lo que facilita la navegación progresiva del usuario a través del proceso de personalización.

- Styles.css: Este archivo es responsable de la apariencia visual del wizard. Incluye estilos que aseguran una presentación consistente y atractiva de los elementos, como los botones, inputs y la vista previa del logo. Además, maneja las animaciones que se muestran durante la transición entre la pantalla de bienvenida y el formulario de personalización, proporcionando una experiencia de usuario fluida y profesional.

- Script.js: Este archivo maneja la lógica interactiva del wizard. Aquí se definen las funciones que controlan la navegación entre los pasos del wizard, la previsualización del logo cargado por el usuario, y la validación de los campos de entrada.

El wizard de personalización, comienza con un mensaje de bienvenida a la plataforma, el cual se desvanece y da paso a los pasos del formulario.

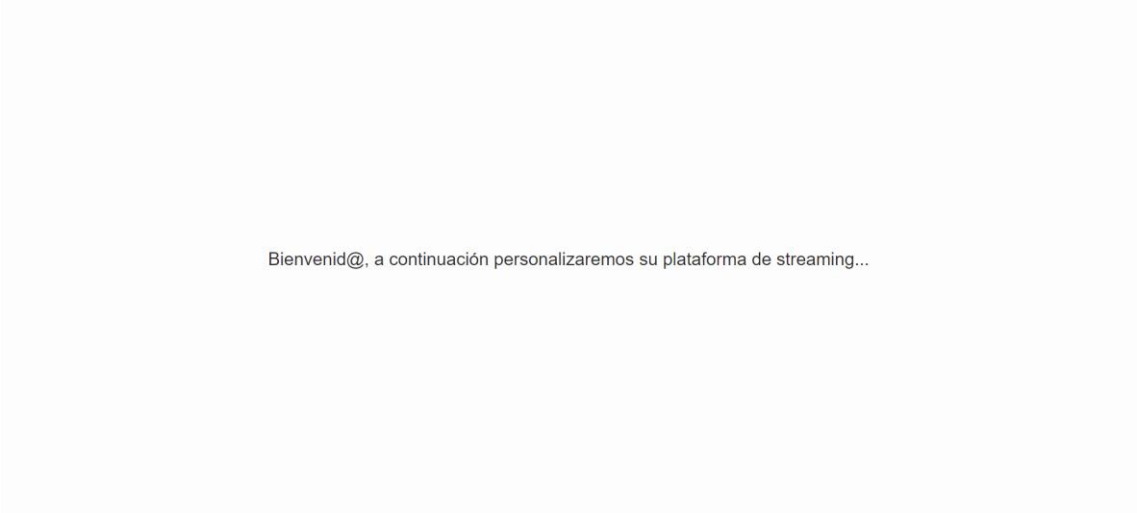

Figura 18. Mensaje de bienvenida

Una vez el mensaje se ha desvanecido, aparece el formulario de personalización, el cual como ya se ha mencionado anteriormente en este documento, permite modificar logo, colores corporativos, título y subtítulo, fuente, letra en cursiva o no, color de los botones, color de la fuente y la calidad máxima de codificación de los vídeos.

Cada uno de los pasos del wizard se verá de la siguiente manera:

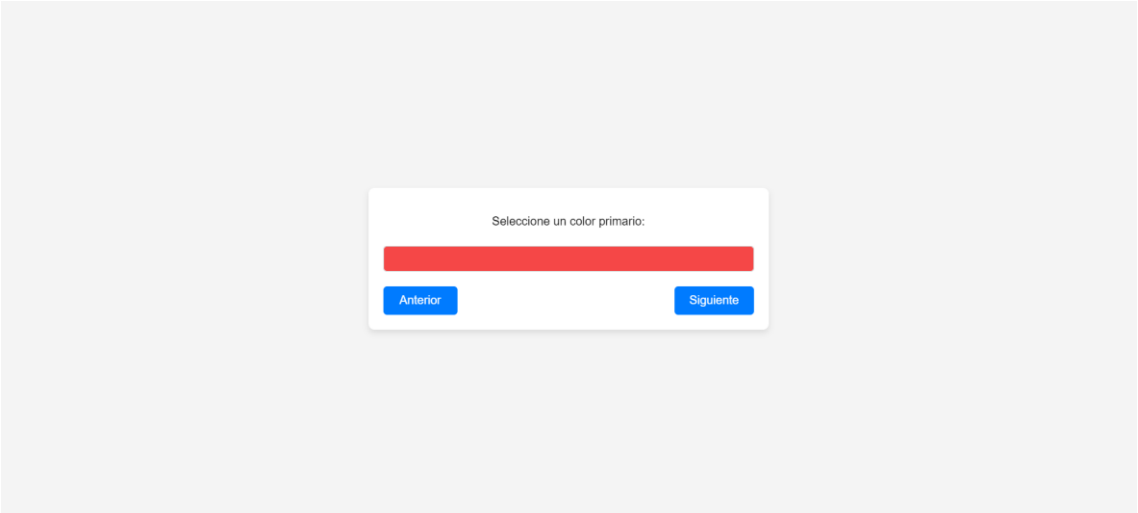

Figura 19. Paso del wizard de personalización

Estos pasos tienen la siguiente estructura dentro de index.html:

```
<!-- Paso 2: Seleccionar color primario -->
<div id="step2" class="step hidden">
    <p>Seleccione un color primario:</p>
    <!-- Input para seleccionar el color -->
    <input type="color" id="primaryColor">
    <div class="button-container">
        <!-- Botón para ir al paso anterior -->
        <button onclick="changeStep(1)">Anterior</button>
        <!-- Botón para ir al siguiente paso -->
        <button onclick="changeStep(3)">Siguiente</button>
    </div>
</div>
```

Figura 20. Código HTML de un paso en el wizard de personalización

Pulsando en los botones ‘Anterior’ y ‘Siguiente’, navegaremos entre los diferentes pasos, hasta llegar al último, que en lugar de botón ‘Anterior’, tendrá un botón ‘Comenzar’, el cual lanzará la plataforma de streaming para el administrador.

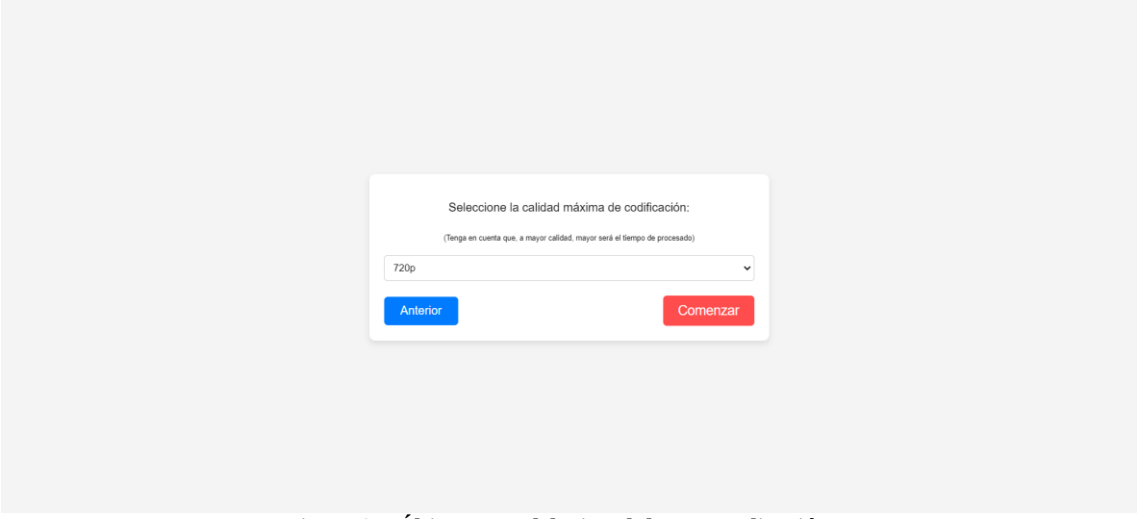

Figura 21. Último paso del wizard de personalización

La funcionalidad de los botones, presente en el archivo script.js, está definida gracias a las funciones 'showStep' y 'changeStep'.

```
// Función para mostrar un paso específico del formulario de personalización
function showStep(stepNumber) {
    const steps = document.querySelectorAll('.step');
    steps.forEach(step => {
        step.style.display = 'none'; // Ocultar todos los pasos
    });
    document.getElementById('step' + stepNumber).style.display = 'flex'; // Mostrar el paso actual
}

// Función para cambiar de paso en el formulario de personalización
function changeStep (stepNumber) {
    showStep(stepNumber);
}
```

Figura 22. Funciones showStep y changeStep

Es destacable mencionar también, la función de previsualización del logo en el formulario.

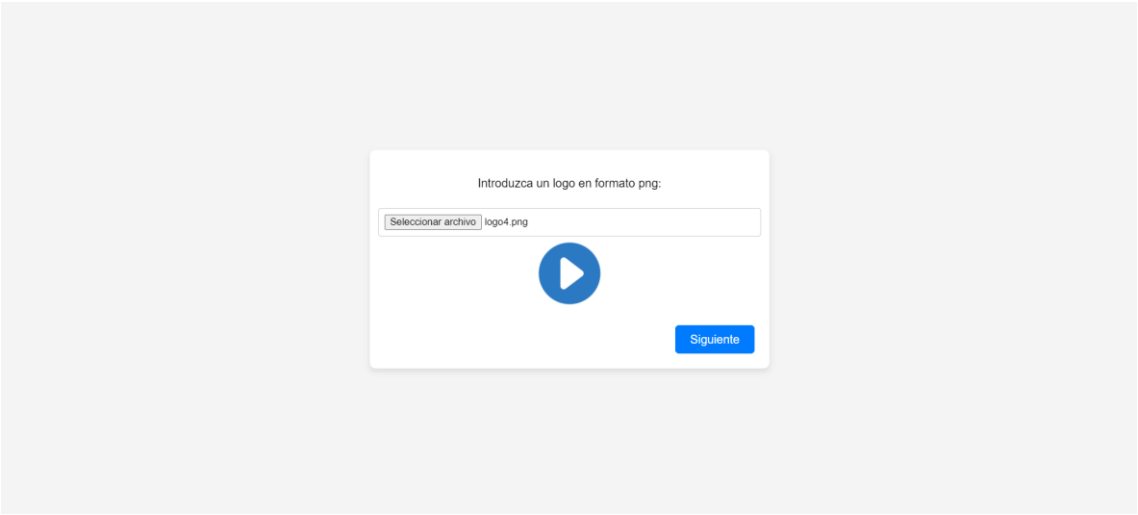

Figura 23. Paso de selección de logo en el wizard de personalización

```
<!-- Paso 1: Cargar el logo -->
<div id="step1" class="step">
    <p>Introduzca un logo en formato png:</p>
    <!-- Input para cargar el logo -->
    <input type="file" id="logoUploader" accept="image/png" onchange="previewLogo(event)">
    <!-- Vista previa del logo cargado -->
    <img id="logoPreview" src="" alt="Vista previa del logo" style="max-width: 100px; display: none;">
    <div class="button-container">
        <!-- Botón para ir al siguiente paso -->
        <button onclick="changeStep(2)" style="margin-left: auto;">Siguiente</button>
    </div>
</div>
```

Figura 24. Código HTML del paso de selección de logo en el wizard

Dicha previsualización se logra gracias a la función 'previewLogo', también presente en script.js

```
// Función para previsualizar el logo seleccionado
function previewLogo(event) {
    const [file] = event.target.files;
    if (file) {
        const reader = new FileReader();
        reader.onload = function(e) {
            const logoPreview = document.getElementById('logoPreview');
            logoPreview.src = e.target.result; // Establecer la imagen del logo en la vista previa
            logoPreview.style.display = 'block'; // Mostrar la vista previa del logo
        };
        reader.readAsDataURL(file); // Leer el archivo como una URL de datos
    }
}
```

Figura 25. Función previewLogo

5.1.2 Front-end de la plataforma

Como ya se ha mencionado en apartados anteriores de este documento, existirán dos interfaces diferentes de la plataforma, una para administradores y otra para usuarios normales.

Se procederá a describir el front-end de la plataforma para ambos casos.

5.1.2.1 Front-end de la plataforma para administradores

La plataforma de streaming para administradores está formada por los siguientes archivos:

- Platform_admin.html: Este archivo contiene la estructura principal de la página de administración. Define la disposición de los elementos como el encabezado con el logo, el título, los botones de administración (subir vídeos, personalizar la plataforma, eliminar todos los vídeos), y el contenedor de vídeos subidos. Además, incluye los modales para subir vídeos, personalizar la plataforma y confirmar la eliminación de vídeos.

- Platform.css: Este archivo, común tanto para administradores como para usuarios define los estilos de la interfaz. Además de los estilos generales, este archivo maneja el diseño responsive, permitiendo que la interfaz se ajuste adecuadamente a diferentes tamaños de pantalla. El CSS controla la apariencia de los botones, los modales y los contenedores de vídeo, así como las animaciones y transiciones de los elementos.

- Platform_admin.js: Este archivo contiene la lógica de la interfaz de administración. Gestiona las interacciones con los usuarios, como abrir y cerrar modales, manejar la subida de vídeos y la previsualización del contenido, y enviar los datos al servidor. También controla la personalización de la plataforma, permitiendo a los administradores cambiar el logo, los colores, las fuentes y la calidad de codificación de los vídeos. Además, se encarga de la actualización dinámica del contenido mostrado en la página, como la lista de vídeos subidos y los mensajes de progreso.

La interfaz de la plataforma para los administradores comienza de la siguiente manera una vez ha sido personalizada en el wizard.

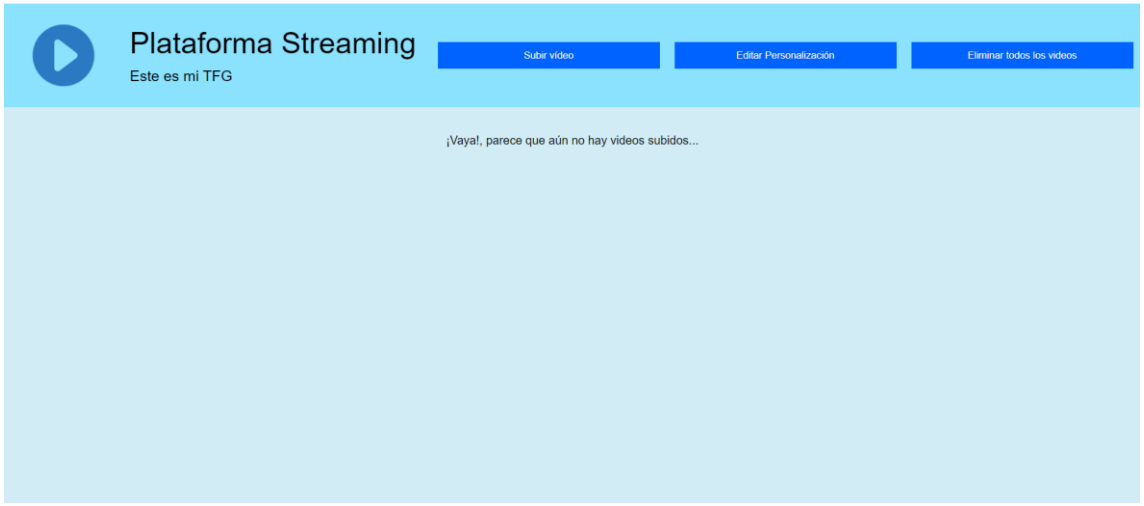

Figura 26. Plataforma de streaming como Administrador sin vídeos subidos.

Puesto que se ha iniciado por primera vez y no hay vídeos subidos, en el lugar del catálogo de vídeos aparece un mensaje que informa de la ausencia de vídeos subidos a la plataforma.

Como podemos observar, en la parte superior de la pantalla aparecen 3 botones, los cuales al ser pulsados darán paso a 3 modales diferentes.

El botón 'Subir vídeo', abrirá el modal de subida de vídeo, el cual nos pedirá un archivo .mp4, un archivo de imagen para la miniatura, un título para el vídeo y una descripción.

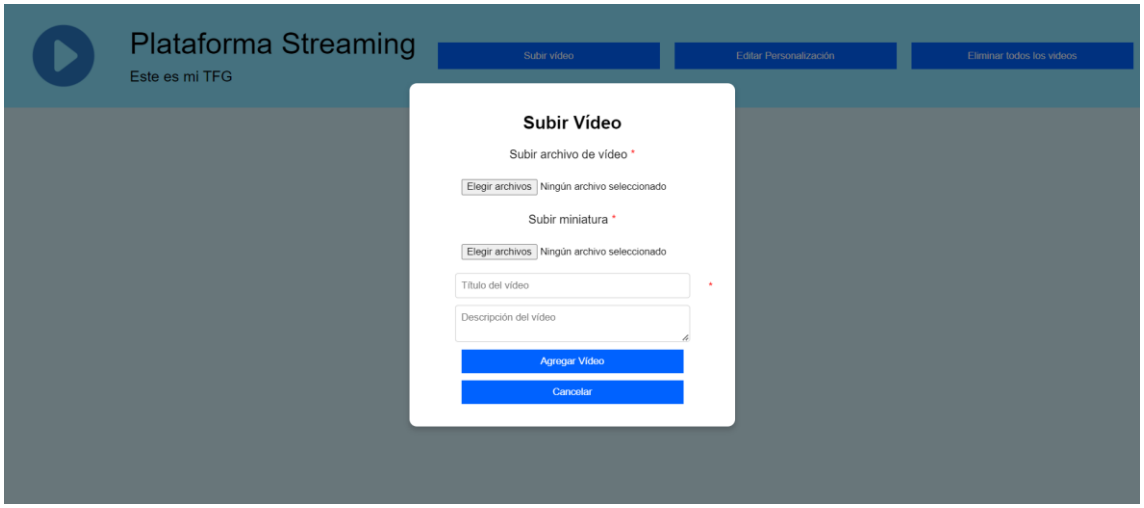

Figura 27. Modal de subida de vídeo

De estos cuatro campos, tan solo la descripción es opcional, los otros tres son obligatorios y si no cumplimentamos alguno de ellos, aparecerá un mensaje de advertencia y no se permitirá subir el vídeo.

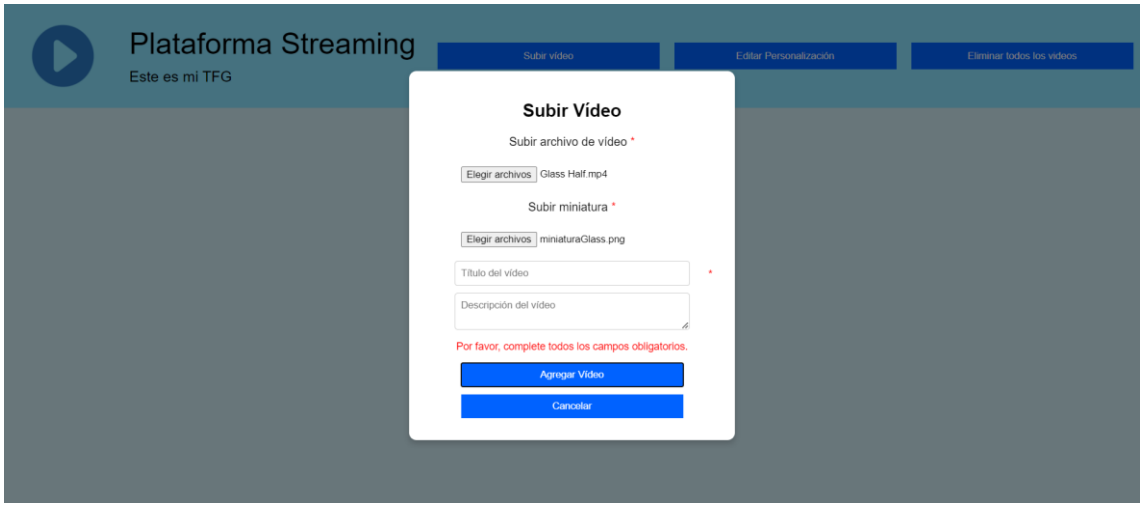

Figura 28. Advertencia modal de subida de vídeo

El modal de subida de vídeo tendrá la siguiente estructura HTML:

```
<!-- Modal para subir videos -->
<div id="modal" class="hidden">
    <div id="uploadForm">
        <h2>Subir Vídeo</h2>
        <p>Subir archivo de vídeo <span class="required">*</span></p>
        <input type="file" name="video" id="videoUploader" accept="video/*" multiple> <!-- Input para seleccionar videos -->
        <p>Subir miniatura <span class="required">*</span></p>
        <input type="file" name="thumbnail" id="thumbnailUploader" accept="image/*" multiple> <!-- Input para seleccionar miniaturas -->
        <div>
            <span class="required" style="position: absolute; right: 20px; margin: 10px;">*</span>
            <input type="text" id="videoTitle" placeholder="Título del vídeo" class="textarea"> <!-- Campo para el título del video -->
        </div>
        <textarea id="videoDescription" placeholder="Descripción del vídeo" maxlength="500" class="textarea"></textarea> <!-- Campo para la descripción del video -->
        <p id="errorMessage" class="error hidden">Por favor, complete todos los campos obligatorios.</p> <!-- Mensaje de error -->
        <button onclick="addVideos()">Agregar Vídeo</button> <!-- Botón para agregar el video -->
        <button onclick="closeModalCancel()">Cancelar</button> <!-- Botón para cancelar la carga -->
    </div>
</div>
```

Figura 29. Código HTML del modal de subida de vídeo

Al pulsar el botón de ‘Agregar Vídeo’, el vídeo comenzará a procesarse y así aparecerá indicado en el catálogo.

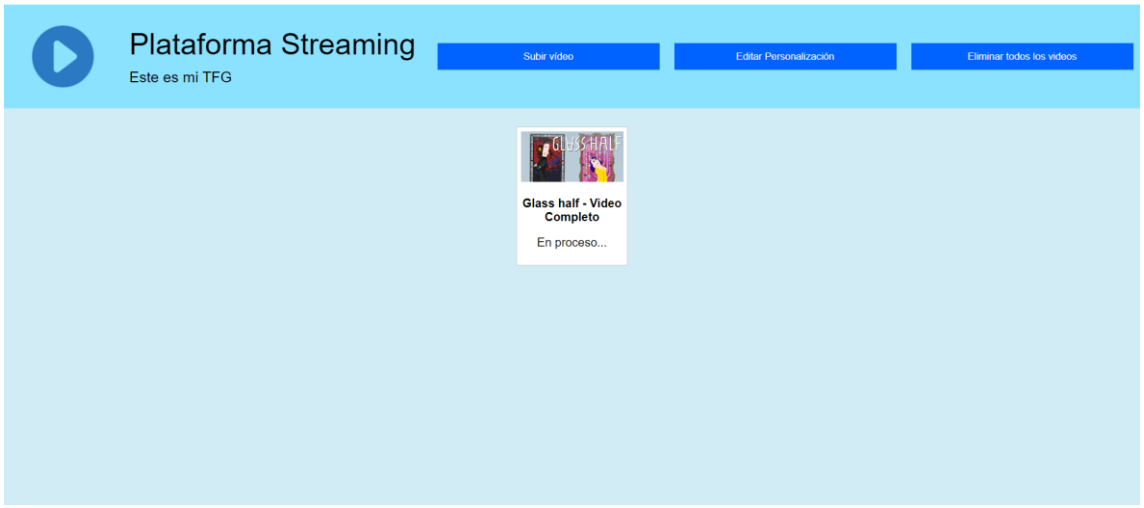

Figura 30. Plataforma de streaming como Administrador con vídeo en proceso.

Una vez subido el vídeo, aparecerá en el catálogo con el título y miniaturas seleccionadas, de la siguiente manera:

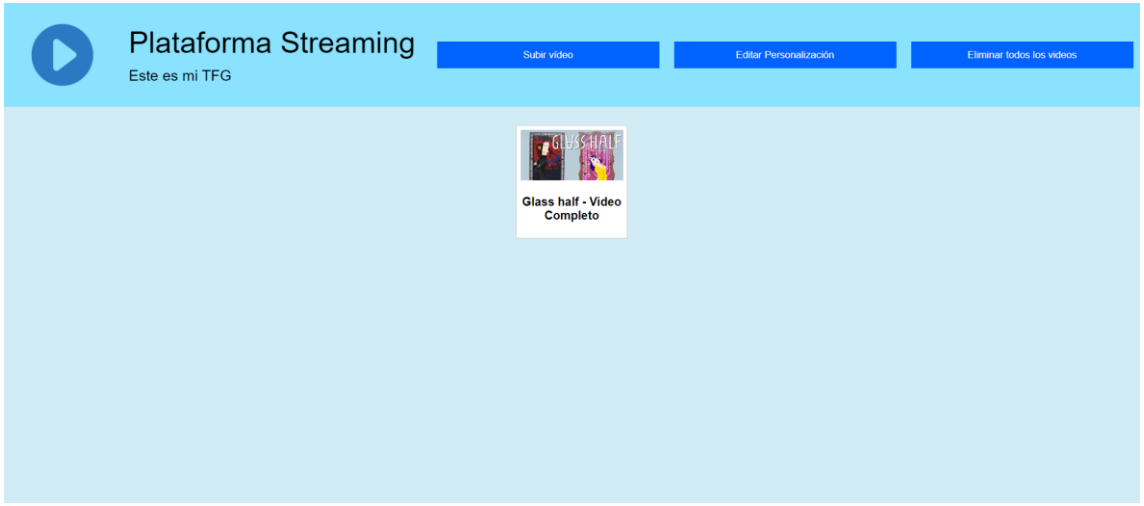

Figura 31. Plataforma de streaming como Administrador con vídeos subidos.

Conforme sigamos subiendo vídeos a la plataforma, irán mostrándose en el catálogo en forma de mosaico:

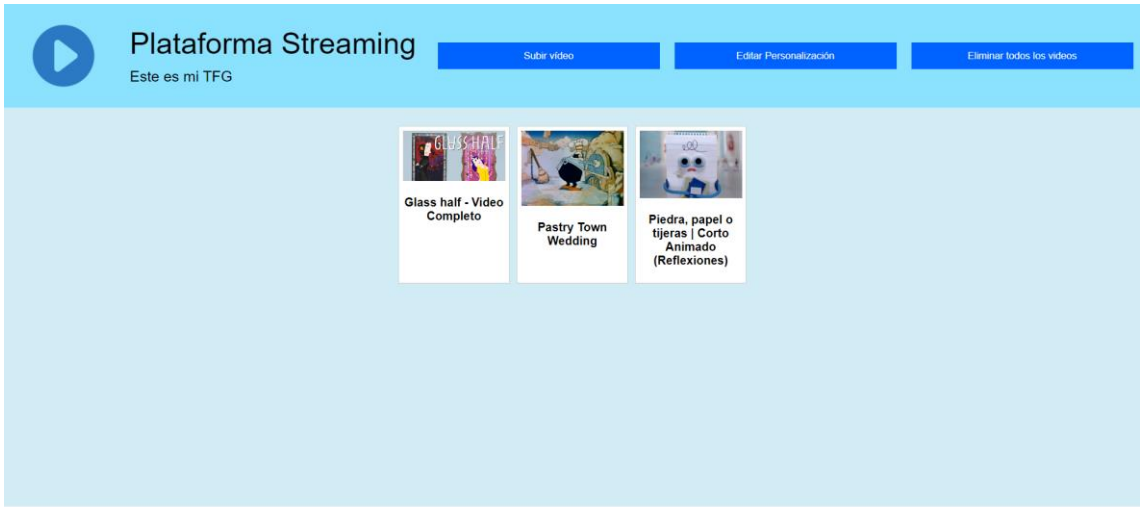

Figura 32. Mosaico de vídeos en el catálogo

El botón 'Editar personalización', nos abrirá el modal de edición de personalización, el cual permitirá modificar los valores elegidos previamente en el wizard, sin la necesidad de volver a pasar por este.

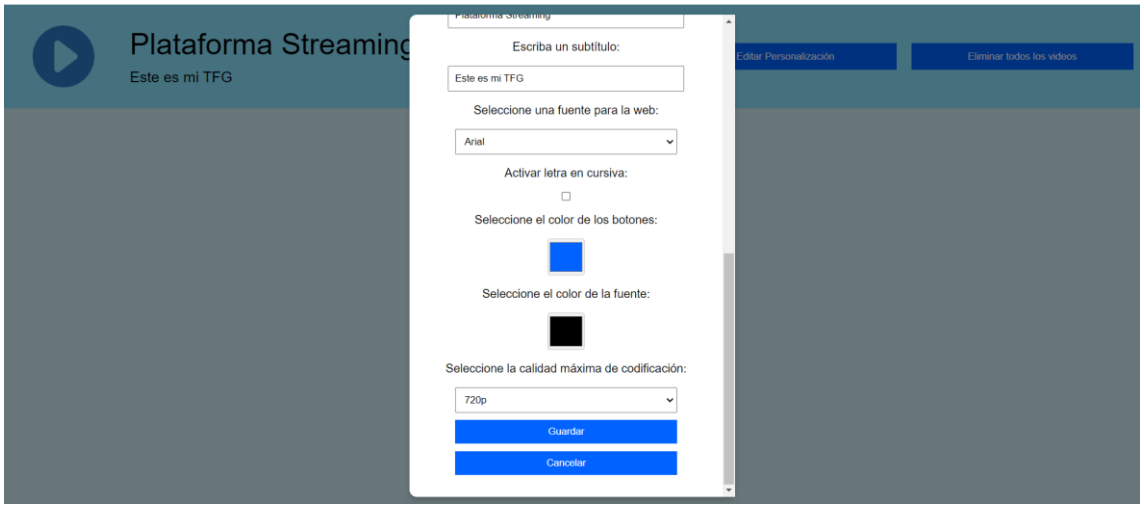

Figura 33. Modal de edición de personalización

Si pulsamos en alguno de los vídeos del catálogo, se nos abrirá el modal de reproducción de vídeo, en el cual aparecerá el vídeo en la parte izquierda y el título, descripción y botones en la parte derecha.

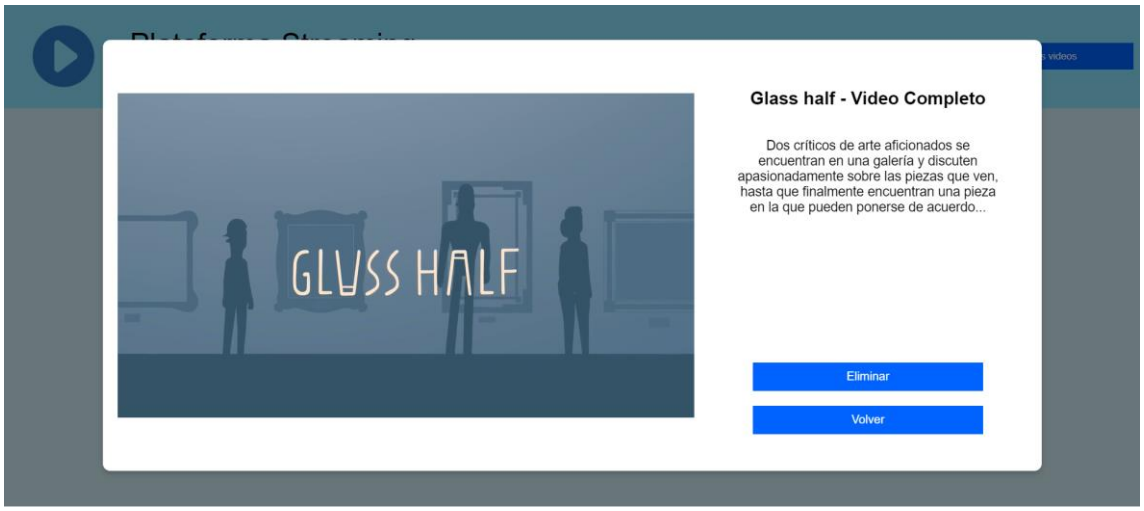

Figura 34. Modal de reproducción de vídeos como Administrador

Si queremos eliminar el vídeo, pulsamos en el botón 'Eliminar', el cual dará paso a un modal de confirmación para asegurar la intención del administrador.

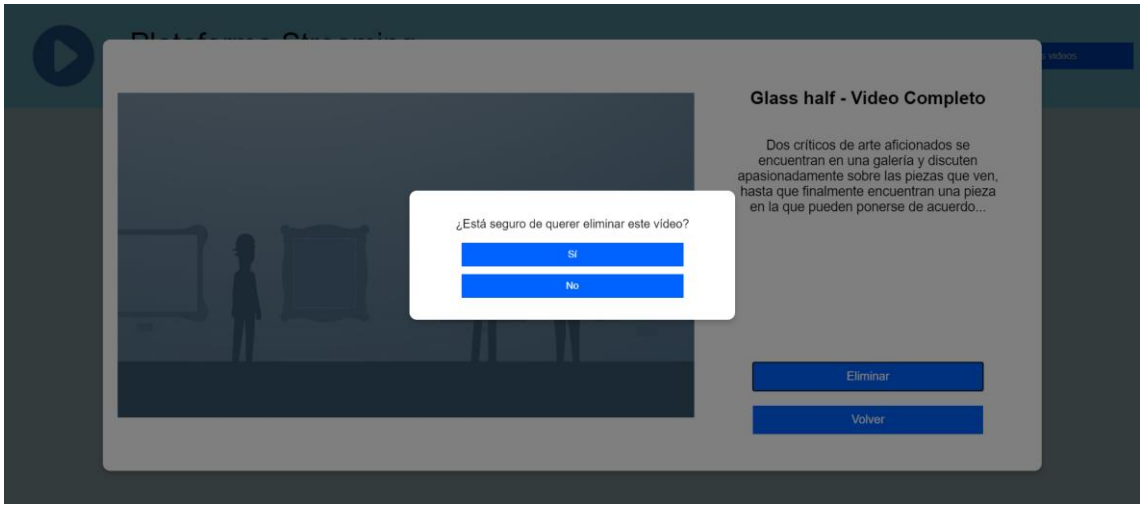

Figura 35. Modal de eliminación de un vídeo

Si deseamos eliminar todos los vídeos de manera simultánea, pulsaremos el botón 'Eliminar todos los vídeos', el cual nos abrirá el modal de eliminación de todos los vídeos.

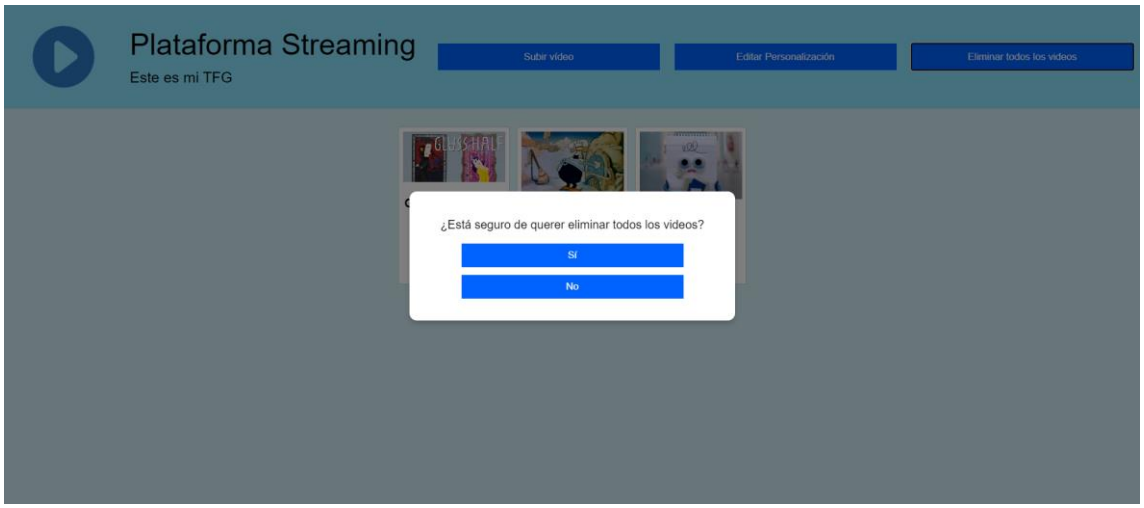

Figura 36. Modal de eliminación de todos los vídeos

Finalmente, me gustaría destacar que la totalidad de la plataforma es responsive y se adapta dinámicamente a las dimensiones de la pantalla.

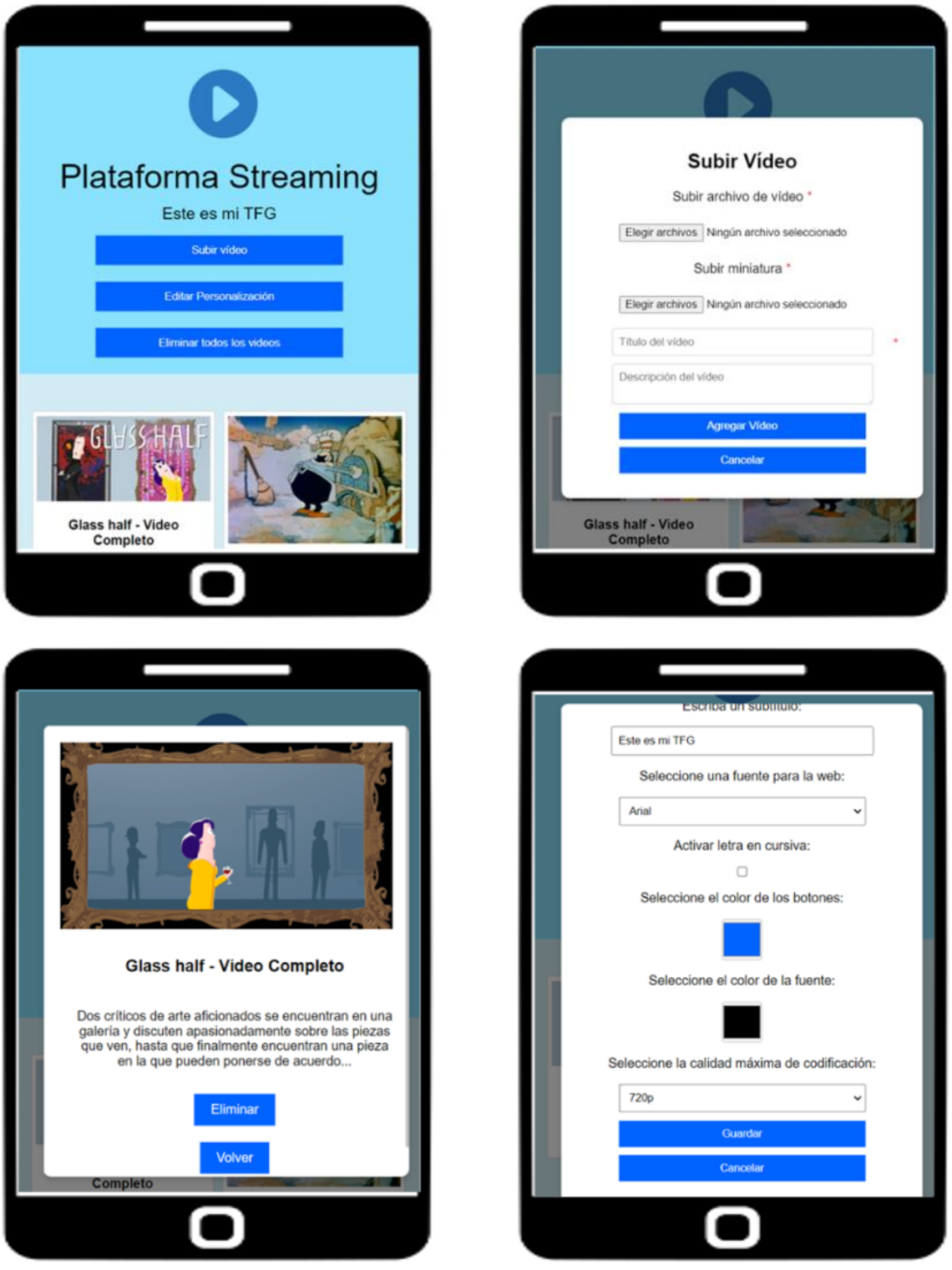

Figura 37. Vista de la interfaz de administrador de la plataforma en dispositivos móviles

5.1.2.2 *Front-end de la plataforma para usuarios*

En lo referido al front-end de la plataforma para usuarios, hay poco que mencionar, puesto que la plataforma comparte interfaz tanto para administradores como para usuarios.

Lo único que cambia son las funciones propias del administrador (subir y eliminar vídeos y editar la identidad visual de la plataforma), las cuales, no están presentes en la interfaz para usuarios.

A continuación, se mostrará una imagen de la interfaz de usuario:

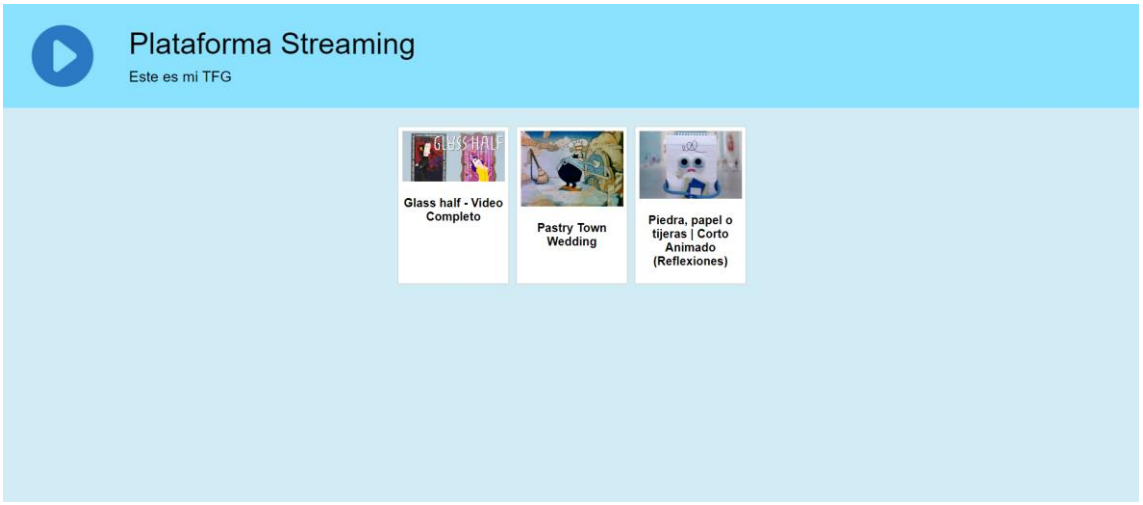

Figura 38. Plataforma de streaming como Usuario con vídeos subidos.

Cabe mencionar, que los usuarios tampoco serán capaces de visualizar cuando un vídeo se encuentra en proceso de subida, y tendrán que esperar a que este termine de procesarse y recargar la página para poder reproducirlo.

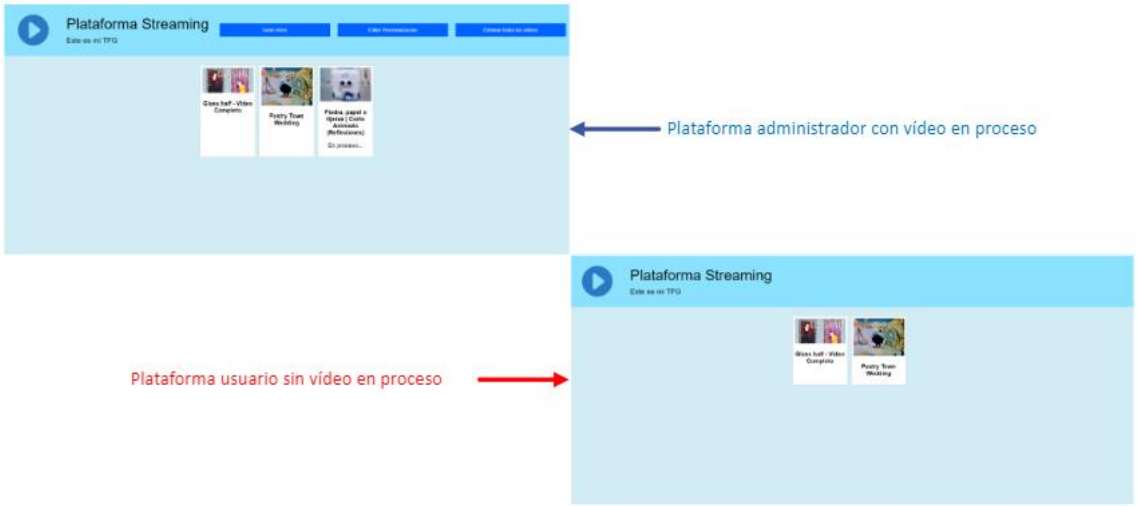

Figura 39. Diferencia de plataformas Administrador y Usuario con vídeo en proceso

Recordemos que los usuarios tan solo tendrán la capacidad de reproducir vídeos, y su modal de reproducción de vídeos será idéntico que el de los administradores, pero sin el botón de 'Eliminar vídeo'.

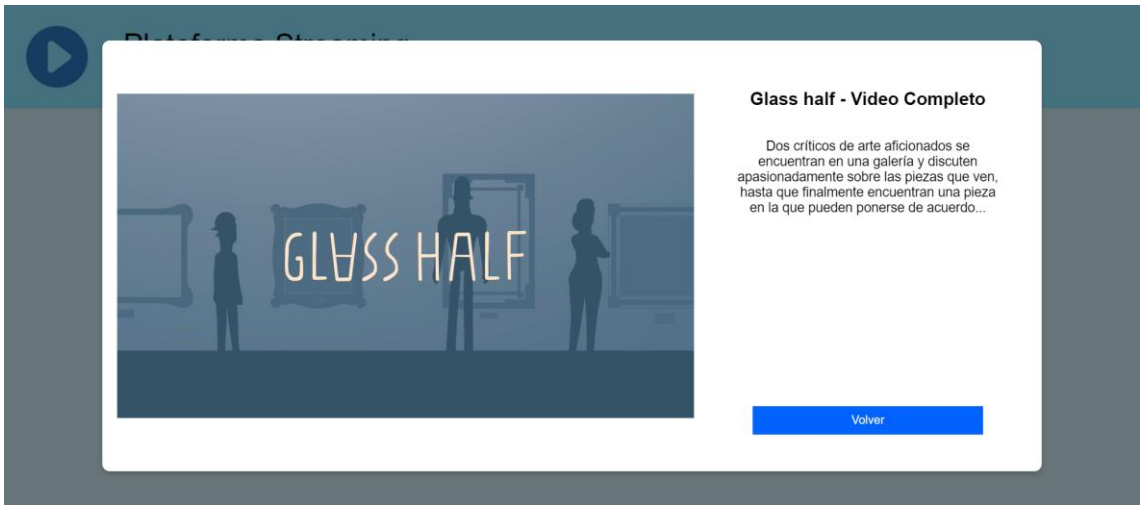

Figura 40. Modal de reproducción de vídeos como Usuario

Finalmente, al igual que con la interfaz para administradores, la totalidad de la plataforma es responsive y se adapta dinámicamente a las dimensiones de la pantalla.

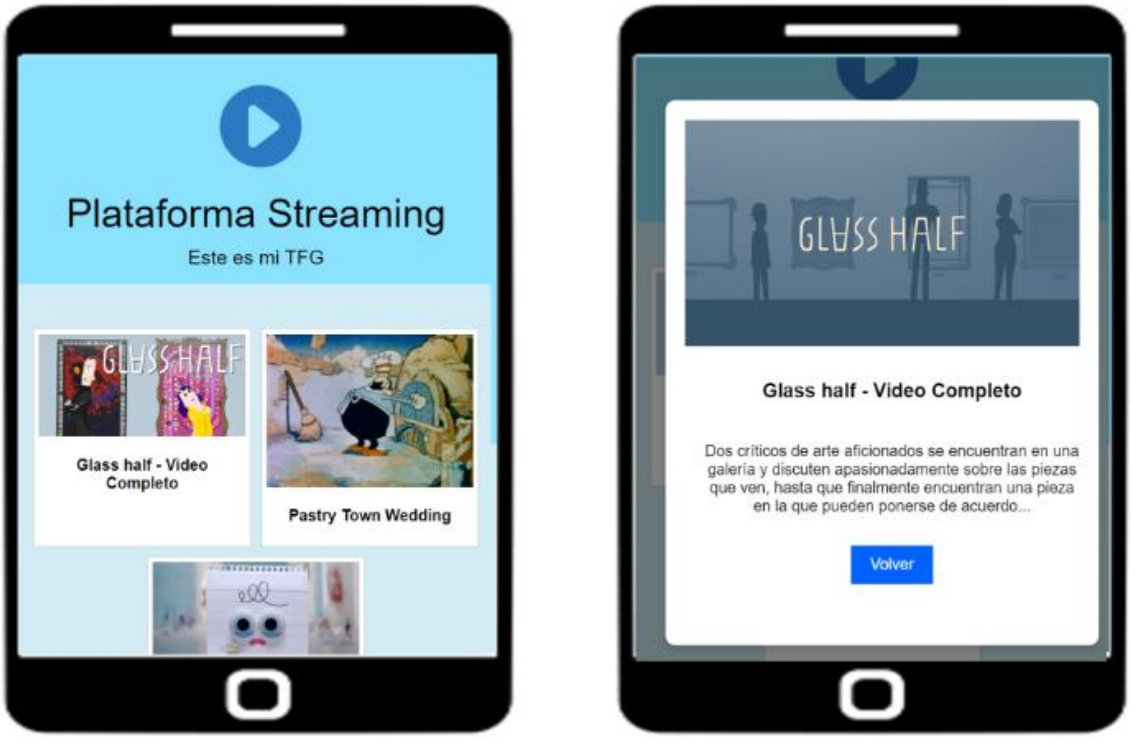

Figura 41. Vista de la interfaz de usuario de la plataforma en dispositivos móviles

5.2 Back-end

Tras presentar y explicar el front-end de la plataforma, es momento de hablar del back-end.

El back-end es la parte del sitio web que se encarga de gestionar la lógica del servidor, manejar las peticiones de los clientes, interactuar con la base de datos y procesar los archivos multimedia para su distribución.

Aunque es la parte "invisible" de la plataforma, juega un papel crucial en la gestión de datos y la seguridad. Junto con el front-end, el back-end trabaja para ofrecer una experiencia de usuario completa y fluida.

Para el desarrollo del back-end se ha utilizado Node.js, un entorno de ejecución de JavaScript del lado del servidor. Node.js destaca por su eficiencia y escalabilidad, lo que lo convierte en una opción muy utilizada para la construcción de aplicaciones web modernas.

El servidor Node.js se ha programado en el archivo ‘server.js’, y es el archivo sobre el cual gira todo el funcionamiento de la plataforma.

Se han utilizado una serie de extensiones de Node.js para cumplir diferentes funciones, las cuales podemos ver en la siguiente figura con una breve explicación en forma de comentario en el código:

```
const express = require('express'); // Importar el framework Express para las solicitudes HTTP
const multer = require('multer'); // Importar Multer para manejar la carga de archivos
const { exec } = require('child_process'); // Importar exec para ejecutar comandos en la terminal
const path = require('path'); // Importar path para manejar rutas de archivos
const fs = require('fs'); // Importar fs para manejar el sistema de archivos
const mysql = require('mysql'); // Importar MySQL para interactuar con la base de datos
```

Figura 42. Extensiones de Node.js utilizadas en 'server.js'

Lo primero que se realizará para antes de poner en marcha la plataforma, es crear la base de datos MySQL, para ello he creado un archivo SQL ejecutable con todos los comandos SQL necesarios. (create_database.sql)

```
-- Crear la base de datos VideoPlatform
CREATE DATABASE IF NOT EXISTS VideoPlatform;

-- Usar la base de datos VideoPlatform
USE VideoPlatform;

-- Crear la tabla settings
CREATE TABLE IF NOT EXISTS settings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    logo TEXT,
    color VARCHAR(255),
    secondaryColor VARCHAR(255),
    tertiaryColor VARCHAR(255),
    title VARCHAR(255),
    subtitle VARCHAR(255),
    font VARCHAR(255),
    fontColor VARCHAR(255),
    isItalic BOOLEAN,
    buttonColor VARCHAR(255),
    quality VARCHAR(10)
);

-- Crear la tabla videos
CREATE TABLE IF NOT EXISTS videos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    thumbnail VARCHAR(255),
    manifest VARCHAR(255) NOT NULL,
    processing BOOLEAN DEFAULT false,
    processed BOOLEAN DEFAULT false
);
```

Figura 43. Archivo 'create_database.sql'

Una vez se ha creado la base de datos, se procederá a iniciar el servidor con el archivo server.js.

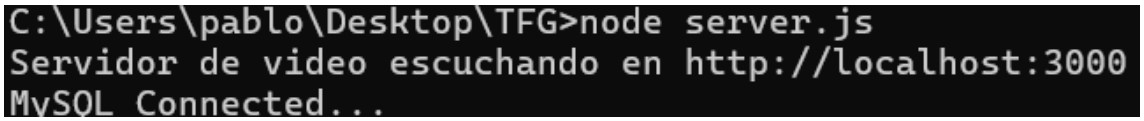

Figura 44. Feedback del CMD con la puesta en marcha del servidor

Ya creada la BBDD y puesto en marcha el servidor, se procederá a explicar el back-end de los aspectos más relevantes de la plataforma.

5.2.1 Personalización de la plataforma

Como ya se ha mencionado con anterioridad, la personalización de la plataforma es uno de los aspectos clave del proyecto, y es vital para diferenciarlo de otros servicios similares. Los administradores tendrán la capacidad de personalizar diversos elementos de la plataforma ya sea desde el wizard de personalización inicial o desde la propia plataforma.

En el backend, la personalización de la plataforma se gestiona a través de la tabla settings en la base de datos MySQL. Esta tabla almacena todas las configuraciones personalizadas de la plataforma.

Dicha tabla la podemos observar en la Figura 43 y en la Tabla 1.

Cada vez que se guarda una nueva configuración, se inserta un nuevo registro en esta tabla. Solo el último registro insertado es relevante para la plataforma, ya que las configuraciones anteriores no se utilizan.

El wizard inicial es la primera interacción que el administrador tiene con la plataforma, el cual como ya se ha explicado, guía al administrador a través de una serie de pasos para definir los elementos clave de la personalización.

Los pasos del wizard están implementados en index.html y controlados por ‘script.js’.

En cada paso del wizard, el administrador selecciona o introduce valores, los cuales se almacenan temporalmente en el frontend hasta que se completa el wizard.

Al finalizar todos los pasos, el frontend envía los datos recolectados al backend mediante una solicitud POST a la ruta /settings. El código relevante en ‘script.js’ es el siguiente:

```
const logoUploader = document.getElementById('logoUploader');
const primaryColor = document.getElementById('primaryColor').value;
const secondaryColor = document.getElementById('secondaryColor').value;
const tertiaryColor = document.getElementById('tertiaryColor').value;
const title = document.getElementById('title').value;
const subtitle = document.getElementById('subtitle').value;
const fontSelector = document.getElementById('fontSelector').value;
const fontColor = document.getElementById('fontColor').value;
const isItalic = document.getElementById('italicCheckbox').checked;
const buttonColor = document.getElementById('buttonColor').value;
const quality = document.getElementById('qualitySelector').value;

const formData = new FormData();
if (logoUploader.files.length > 0) {
    formData.append('logo', logoUploader.files[0]);
}
formData.append('title', title);
formData.append('subtitle', subtitle);
formData.append('color', primaryColor);
formData.append('secondaryColor', secondaryColor);
formData.append('tertiaryColor', tertiaryColor);
formData.append('font', fontSelector);
formData.append('fontColor', fontColor);
formData.append('isItalic', isItalic ? 'true' : 'false');
formData.append('buttonColor', buttonColor);
formData.append('quality', quality);

fetch('/settings', {
    method: 'POST',
    body: formData
})
.then(response => response.json())
.then(data => {
    if (data.success) {
        window.location.href = 'platform_admin.html'; // Redireccionar a la plataforma de streaming
    } else {
        console.error('Error al guardar las configuraciones');
    }
```

Figura 45. Fragmento del código para guardar la configuración del wizard

El archivo ‘server.js’ maneja la solicitud a través de la ruta /settings. Esta ruta está configurada para recibir archivos mediante multer, que se encarga de gestionar la carga del logo. A continuación, inserta los datos en la tabla settings:

```
// Ruta para guardar configuraciones, incluyendo logo
app.post('/settings', upload.single('logo'), (req, res) => {
    const { title, subtitle, color, secondaryColor, tertiaryColor, buttonColor, font, fontColor, isItalic, quality } = req.body;
    const isItalicInt = isItalic === 'true' ? 1 : 0; // Convertir el valor de 'isItalic' a un entero
    let logo = req.file ? '/logo/logo.png' : req.body.logo; // Definir la ruta del logo
    let sql = 'INSERT INTO settings (logo, title, subtitle, color, secondaryColor, tertiaryColor, buttonColor, font, fontColor, isItalic, quality) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)';
    db.query(sql, [logo, title, subtitle, color, secondaryColor, tertiaryColor, buttonColor, font, fontColor, isItalicInt, quality], (err, result) => {
        if (err) throw err; // Lanzar un error si la consulta falla
        res.json({ success: true }); // Enviar una respuesta indicando que la configuración se guardó correctamente
    });
});
```

Figura 46. Fragmento de código para guardar la configuración en 'server.js'

En este proceso, el archivo logo.png se guarda en el directorio logo/, y el resto de los valores se almacenan en la base de datos.

Si el administrador, edita la personalización desde el modal de la plataforma, el proceso funciona de la misma manera que si lo edita desde el wizard.

Una vez que se han guardado las configuraciones de personalización en la base de datos, estas se aplican tanto en la interfaz de administración como en la interfaz de usuario de la plataforma.

Cada vez que se carga la página 'platform_admin.html' o 'platform_user.html', el front-end necesita obtener las configuraciones actuales desde el back-end para aplicar la personalización. Este proceso se realiza mediante una solicitud GET a la ruta /settings, que devuelve la configuración más reciente almacenada en la tabla settings.

```
// Obtener configuraciones de personalización desde el servidor al cargar la página
fetch('/settings')
    .then(response => response.json())
    .then(settings => {
        const logoImage = document.getElementById('logoImage');
        const titleContainer = document.getElementById('titleContainer');
        const subtitleContainer = document.getElementById('subtitleContainer');
        document.body.style.setProperty('--primary-color', settings.color || '#00BFFF');
        document.body.style.setProperty('--secondary-color', settings.secondaryColor || '#FFFFFF');
        document.body.style.setProperty('--tertiary-color', settings.tertiaryColor || '#FFFFFF');
        document.body.style.setProperty('--button-color', settings.buttonColor || '#007BFF');
        document.body.style.fontFamily = settings.font || 'Arial';
        document.body.style.color = settings.fontColor || '#000';
        document.body.style.fontStyle = settings.isItalic ? 'italic' : 'normal';

        // Si hay un logo guardado, mostrarlo en la página
        if (settings.logo) {
            logoImage.src = settings.logo;
            document.getElementById('logoPreview').src = settings.logo;
            document.getElementById('logoPreview').style.display = 'block';
        }
        titleContainer.textContent = settings.title || 'Plataforma de Streaming';
        subtitleContainer.textContent = settings.subtitle || '';

        // Guardar configuraciones actuales en variables globales para usarlas al abrir el modal
        window.currentSettings = settings;
```

Figura 47. Fragmento de código de 'platform_admin.js' para cargar la personalización

El siguiente fragmento de código en 'server.js' es responsable de servir las configuraciones almacenadas:

```
// Ruta para obtener configuraciones
app.get('/settings', (req, res) => {
    let sql = 'SELECT * FROM settings ORDER BY id DESC LIMIT 1'; // Consulta SQL para obtener la última configuración guardada
    db.query(sql, (err, result) => {
        if (err) throw err; // Lanzar un error si la consulta falla
        res.json(result[0]); // Enviar la configuración como respuesta en formato JSON
    });
});
```

Figura 48. Fragmento de código para obtener la configuración en 'server.js'

Este endpoint consulta la tabla settings y devuelve la configuración más reciente. Al ordenar los registros por id en orden descendente (ORDER BY id DESC LIMIT 1), se garantiza que siempre se use la última configuración guardada.

5.2.2 Codificación de vídeos

Otro de los aspectos clave de la plataforma, es la reproducción adaptativa de los vídeos. Para lograr esto, el back-end se encarga de la codificación de los vídeos en múltiples calidades y la generación de un manifiesto DASH.

A continuación, se explica en detalle el proceso desde la subida del vídeo hasta su disponibilidad en la plataforma, incluyendo los comandos específicos utilizados y los directorios donde se almacenan los archivos.

El proceso comienza cuando el administrador sube un vídeo a la plataforma a través del modal de subida de vídeos. Los vídeos, junto con las miniaturas asociadas, se envían al servidor utilizando la ruta /upload.

En el back-end, la carga de archivos se gestiona mediante multer, que se configura para guardar las miniaturas y los vídeos sin codificar en el mismo directorio uploads/.

El código relevante en server.js es:

```
// Configuración de multer para manejo de archivos
const upload = multer({
    storage: multer.diskStorage({
        destination: function (req, file, cb) {
            // Definir el destino de los archivos cargados
            if (file.fieldname === 'logo') {
                cb(null, 'logo/'); // Si el archivo es un logo, guardarlo en la carpeta 'logo/'
            } else {
                cb(null, 'uploads/'); // Si el archivo es un video o una miniatura, guardarlo en la carpeta 'uploads/'
            }
        },
        filename: function (req, file, cb) {
            // Definir el nombre del archivo cargado
            if (file.fieldname === 'logo') {
                cb(null, 'logo.png'); // El logo siempre se guarda como 'logo.png'
            } else {
                cb(null, Date.now() + path.extname(file.originalname)); // Generar un nombre único para cada archivo de video o miniatura
            }
        }
    })
});
```

Figura 49. Fragmento de código de 'server.js' para subir vídeo y miniatura

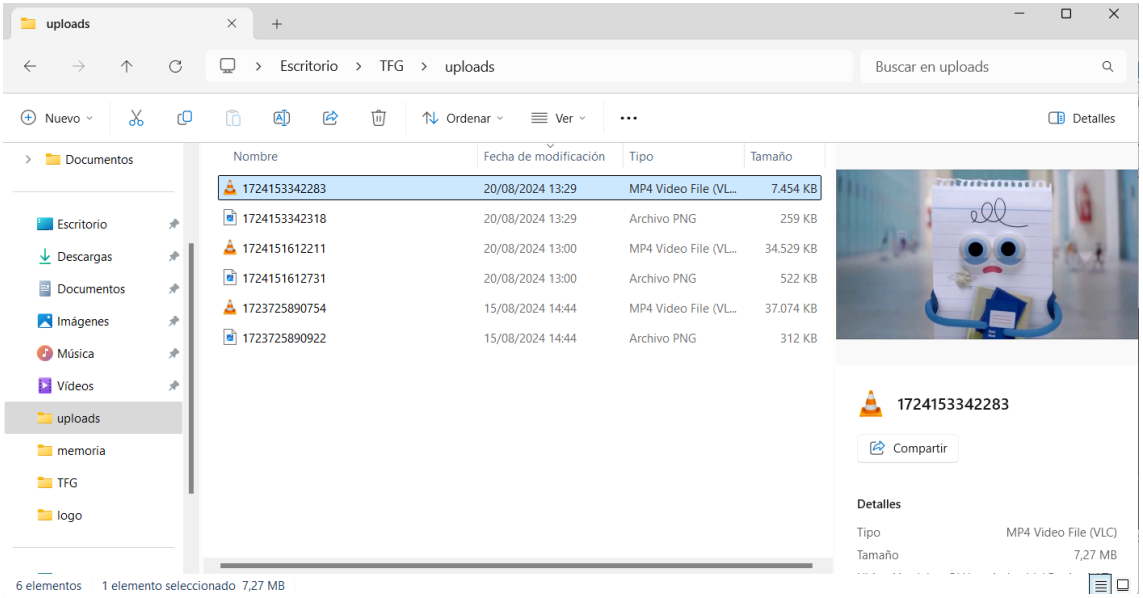

Figura 50. Directorio uploads/ con vídeos y miniaturas subidos

5.2.2.1 Codificación con ffmpeg

Una vez subido el vídeo, el servidor consulta la base de datos para determinar la calidad máxima de codificación seleccionada en la configuración de la plataforma. Esto se realiza mediante una consulta SQL a la tabla settings.

```
// Obtener la calidad de codificación seleccionada
let quality = '720p';
let sql = 'SELECT quality FROM settings ORDER BY id DESC LIMIT 1';
db.query(sql, (err, result) => {
    if (err) throw err; // Lanzar un error si la consulta falla
    if (result.length > 0 && result[0].quality) {
        quality = result[0].quality; // Establecer la calidad si se encuentra en la configuración
    }
```

Figura 51. Fragmento de código de 'server.js' para obtener la calidad máxima de codificación

Dependiendo de la calidad seleccionada (720p, 1080p, 2160p), se definirán diferentes resoluciones de salida.

El servidor utiliza FFMpeg para codificar el vídeo en múltiples resoluciones y preparar los archivos para la reproducción adaptativa. Los comandos FFMpeg se ejecutan para generar diferentes versiones del vídeo, que luego se fragmentarán para su uso en el protocolo MPEG-DASH.

```
let commands = [
    `ffmpeg -i ${inputFile} -an -c:v libx264 -crf 20 -g 48 -bf 2 -s 1280x720 -b_strategy 0 -sc_threshold 0 -flags +cgop ${outputSubDir}/video_720p.mp4`,
    `ffmpeg -i ${inputFile} -an -c:v libx264 -crf 30 -g 48 -bf 2 -s 960x540 -b_strategy 0 -sc_threshold 0 -flags +cgop ${outputSubDir}/video_540p.mp4`,
    `ffmpeg -i ${inputFile} -an -c:v libx264 -crf 50 -g 48 -bf 2 -s 480x270 -b_strategy 0 -sc_threshold 0 -flags +cgop ${outputSubDir}/video_270p.mp4`,
    `ffmpeg -i ${inputFile} -vn -c:a copy ${outputSubDir}/videoAudio.mp4`
];

// Incluir más resoluciones si la calidad es mayor
if (quality === '1080p') {
    commands.push(
        `ffmpeg -i ${inputFile} -an -c:v libx264 -crf 20 -g 48 -bf 2 -s 1920x1080 -b_strategy 0 -sc_threshold 0 -flags +cgop ${outputSubDir}/video_1080p.mp4`
    );
} else if (quality === '2160p') {
    commands.push(
        `ffmpeg -i ${inputFile} -an -c:v libx264 -crf 20 -g 48 -bf 2 -s 1920x1080 -b_strategy 0 -sc_threshold 0 -flags +cgop ${outputSubDir}/video_1080p.mp4`,
        `ffmpeg -i ${inputFile} -an -c:v libx264 -crf 20 -g 48 -bf 2 -s 3840x2160 -b_strategy 0 -sc_threshold 0 -flags +cgop ${outputSubDir}/video_2160p.mp4`
    );
}
```

Figura 52. Comandos FFMpeg en 'server.js'

Como podemos observar en la Figura 52, el servidor siempre codificará los vídeos con una calidad máxima predeterminada de 720p, a no ser que se haya seleccionado una calidad máxima diferente.

Si es 1080p la calidad máxima seleccionada, se añadirá un comando más con una calidad de 1080p, y si es 2160p, se añadirán dos comandos más con calidades de 1080p y 2160p.

A continuación, se explicará más en detalle los comandos FFMpeg seleccionados.

Por ejemplo, para la resolución de 720p, se ejecuta el siguiente comando:

```
ffmpeg -i ${inputFile} -an -c:v libx264 -crf 20 -g 48 -bf 2 -s 1280x720 -b_strategy 0 -sc_threshold 0 -flags +cgop ${outputSubDir}/video_720p.mp4
```

Este comando de FFmpeg realiza varias tareas importantes:

-i ${inputFile}: Especifica el archivo de entrada, es decir, el vídeo que se va a codificar.

-an: Indica que no se incluya audio en este archivo de salida, ya que el audio se procesará por separado.

-c:v libx264: Define el códec de vídeo H.264 [12], uno de los más comunes y eficientes para la compresión de vídeo, especialmente para streaming.

-crf 20: El valor CRF (Constant Rate Factor) controla la calidad de la compresión. Un valor más bajo (como 20) significa mayor calidad y un archivo más grande, mientras que un valor más alto (como 30 o 50) produce un archivo más pequeño y de menor calidad.

-g 48: Define el tamaño del grupo de imágenes (GOP). Este valor indica cada cuántos fotogramas se incluirá una imagen clave (keyframe), lo que afecta la capacidad del vídeo para cambiar entre calidades durante la reproducción.

-bf 2: Establece el número de fotogramas B, que son fotogramas de predicción bidireccional. Estos mejoran la compresión sin perder calidad.

-s 1280x720: Establece la resolución de salida del vídeo, en este caso, 1280x720 píxeles (HD).

-b_strategy 0: Desactiva la estrategia de ajuste dinámico del bitrate, lo que significa que el bitrate se mantiene constante durante la codificación, lo cual es útil para lograr una calidad más uniforme en el vídeo.

-sc_threshold 0: Desactiva el umbral de detección de cambios de escena, lo que evita que el codificador inserte automáticamente imágenes clave en cambios bruscos de escena.

-flags +cgop: Fuerza la creación de GOP cerrados, necesarios para que los segmentos de vídeo puedan ser fragmentados y utilizados de forma independiente en la reproducción adaptativa.

${outputSubDir}/video_720p.mp4: Indica la ubicación y el nombre de salida. El directorio de salida será ‘output/output(identificador único del vídeo)’

Para el resto de las calidades, los comandos son muy similares, variando los valores de -s (resolución) y de -crf (calidad de la compresión).

5.2.2.2 *Fragmentación con mp4fragment*

Una vez generadas las diferentes versiones del vídeo en todas las calidades, estas se fragmentan en segmentos más pequeños mediante el comando mp4fragment (del set de herramientas Bento4).

La fragmentación es crucial para habilitar la reproducción adaptativa, permitiendo al reproductor cambiar de una calidad a otra sin interrupciones, dependiendo del ancho de banda.

El proceso de fragmentación toma las versiones codificadas del vídeo y las divide en fragmentos de 2 segundos:

```
commands.push(
    `mp4fragment --fragment-duration 2000 ${outputSubDir}/video_720p.mp4 ${outputSubDir}/720_f.mp4`,
    `mp4fragment --fragment-duration 2000 ${outputSubDir}/video_540p.mp4 ${outputSubDir}/540_f.mp4`,
    `mp4fragment --fragment-duration 2000 ${outputSubDir}/video_270p.mp4 ${outputSubDir}/270_f.mp4`,
    `mp4fragment --fragment-duration 2000 ${outputSubDir}/videoAudio.mp4 ${outputSubDir}/audio_f.mp4`
);

if (quality === '1080p') {
    commands.push(
        `mp4fragment --fragment-duration 2000 ${outputSubDir}/video_1080p.mp4 ${outputSubDir}/1080_f.mp4`
    );
} else if (quality === '2160p') {
    commands.push(
        `mp4fragment --fragment-duration 2000 ${outputSubDir}/video_1080p.mp4 ${outputSubDir}/1080_f.mp4`,
        `mp4fragment --fragment-duration 2000 ${outputSubDir}/video_2160p.mp4 ${outputSubDir}/2160_f.mp4`
    );
}
```

Figura 53. Comandos mp4fragment en 'server.js'

Al igual que con la codificación, dependiendo de la calidad máxima elegida por el administrador, el servidor fragmentará los vídeos correspondientes.

```
mp4fragment --fragment-duration 2000
${outputSubDir}/video_720p.mp4 ${outputSubDir}/720_f.mp4
```

Explicación de los parámetros de mp4fragment:

--fragment-duration 2000: Determina que cada fragmento debe tener una duración de 2000 milisegundos (2 segundos).

${outputSubDir}/video_720p.mp4: Especifica el archivo de entrada que se va a fragmentar, en este caso, la versión 720p del vídeo.

${outputSubDir}/720_f.mp4: Define el archivo de salida, que contendrá el vídeo fragmentado.

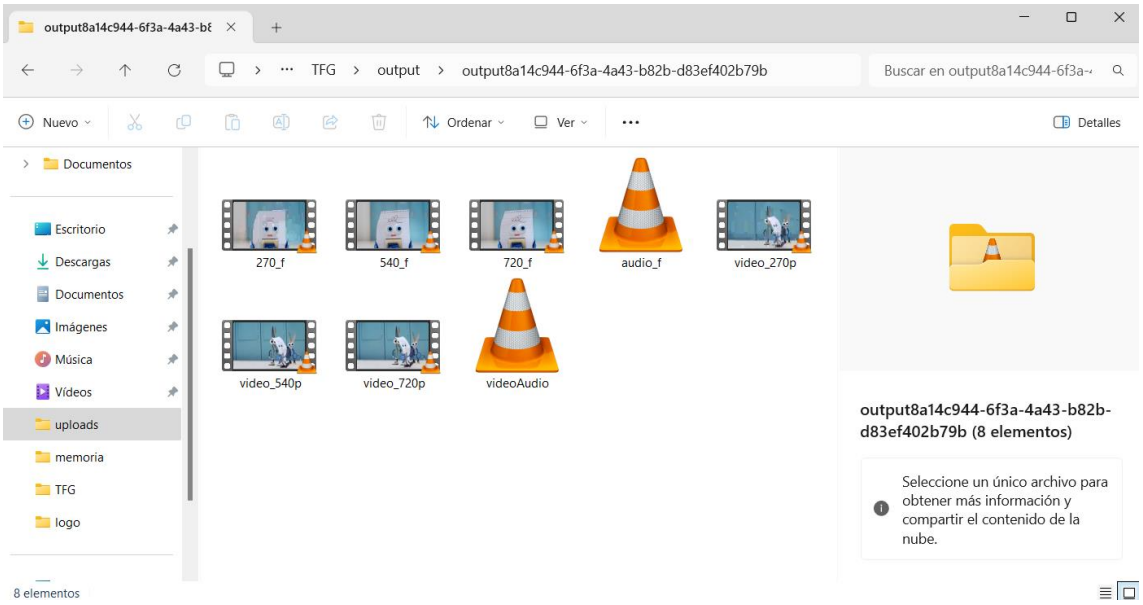

Figura 54. Directorio output de un vídeo con los vídeos en diferentes calidades, el audio y los fragmentos.

5.2.2.3 Generación del manifiesto con mp4dash

Una vez los archivos de vídeo en las diferentes calidades han sido fragmentados correctamente, el último paso es la generación del manifiesto DASH.

El manifiesto DASH es un archivo .mpd (en lenguaje XML) que organiza las distintas versiones de un vídeo, incluyendo sus calidades y fragmentos, para facilitar la reproducción adaptativa. Este archivo es fundamental, ya que permite al reproductor elegir y alternar entre las calidades disponibles según la velocidad de la conexión.

La reproducción adaptativa depende completamente de este manifiesto.

```
let dashCommand = `mp4dash --output-dir=${mpdSubDir} --mpd-name=video.mpd --profiles=on-demand ${outputSubDir}/720_f.mp4 ${outputSubDir}/540_f.mp4 ${outputSubDir}/270_f.mp4 ${outputSubDir}/audio_f.mp4`;
if (quality === '1080p') {
    dashCommand += ` ${outputSubDir}/1080_f.mp4`;
} else if (quality === '2160p') {
    dashCommand += ` ${outputSubDir}/1080_f.mp4 ${outputSubDir}/2160_f.mp4`;
}
```

Figura 55. Comandos mp4dash en 'server.js'

El manifiesto se genera con un comando como el siguiente:

```
mp4dash --output-dir=${mpdSubDir} --mpd-name=video.mpd --
profiles=on-demand ${outputSubDir}/720_f.mp4
${outputSubDir}/540_f.mp4 ${outputSubDir}/270_f.mp4
${outputSubDir}/audio_f.mp4
```

Explicación de los parámetros de mp4dash:

--output-dir=${mpdSubDir}: Define el directorio donde se almacenará el manifiesto DASH y todos los fragmentos de vídeo asociados. El directorio de salida será ‘mpd/mpd(identificador único del vídeo)’

--mpd-name=video.mpd: Especifica el nombre del archivo manifiesto generado, que será referenciado por el reproductor para iniciar la reproducción.

--profiles=on-demand: Indica que el perfil de DASH es de "entrega bajo demanda", optimizado para servicios de streaming.

${outputSubDir}/720_f.mp4, ${outputSubDir}/540_f.mp4, ${outputSubDir}/270_f.mp4 ${outputSubDir}/audio_f.mp4. `: Lista los archivos fragmentados que se incluirán en el manifiesto.

5.2.2.4 Interacción con la BBDD

La tabla videos en la base de datos es fundamental para la gestión y control de todo el proceso de codificación y reproducción de vídeos en la plataforma.

Nos centraremos en la interacción de ‘server.js’ con la tabla videos, la cual ya ha sido descrita con anterioridad en el documento, en la Tabla 2.

Cuando un vídeo se sube a la plataforma, ‘server.js’ inserta un nuevo registro en la tabla videos. Este registro incluye el título, la descripción, la miniatura, y marca el campo processing como true para indicar que el vídeo está en proceso de codificación.

El campo manifest se deja vacío inicialmente ya que el manifiesto DASH aún no se ha generado.

```
// Insertar video en la base de datos con estado "processing"
let sql = 'INSERT INTO videos (title, description, thumbnail, manifest, processing) VALUES (?, ?, ?, ?, true)';
db.query(sql, [title, description, `/uploads/${thumbnailFile.filename}`, '', true], (err, result) => {
    if (err) throw err; // Lanzar un error si la consulta falla
    runCommands(commands); // Iniciar la ejecución de los comandos
});
```

Figura 56. Fragmento de código de 'server.js' para insertar un vídeo en proceso

Mientras que el vídeo subido se encuentra en estado de procesamiento, ‘server.js’ mantiene el campo processing en true para evitar que el vídeo sea accesible y dar feedback al administrador. Una vez que el vídeo se ha procesado con éxito, se actualizan los campos processing a false y processed a true, y se guarda la ruta del manifiesto en el campo manifest.

```
let sql = 'UPDATE videos SET manifest = ?, processing = false WHERE title = ?';
db.query(sql, [manifestPath, title], (err, result) => {
    if (err) throw err; // Lanzar un error si la consulta falla
    res.json({ success: true, manifest: manifestPath, title, description, thumbnail: thumbnailPath });
});
```

Figura 57. Fragmento de código de 'server.js' para insertar un vídeo como procesado

Cuando un administrador decide eliminar un vídeo, server.js elimina el registro correspondiente en la tabla videos y borra los archivos asociados (miniatura, manifiesto, fragmentos) del sistema de archivos. Esto asegura que no queden datos huérfanos en la base de datos ni archivos innecesarios en el servidor.

```
// Ruta para eliminar un video por título
app.post('/delete', (req, res) => {
    const { title } = req.body;
    let sql = 'SELECT * FROM videos WHERE title = ?';
    db.query(sql, [title], (err, results) => {
        if (err) throw err; // Lanzar un error si la consulta falla
        const video = results[0];
        if (video) {
            const thumbnailPath = path.join(__dirname, video.thumbnail);
            const manifestPath = path.join(__dirname, video.manifest);
            const mpdPath = path.dirname(manifestPath);
            fs.unlinkSync(thumbnailPath); // Eliminar la miniatura del sistema de archivos
            fs.rmdirSync(mpdPath, { recursive: true }); // Eliminar los archivos DASH
            sql = 'DELETE FROM videos WHERE title = ?';
            db.query(sql, [title], (err, result) => {
                if (err) throw err; // Lanzar un error si la consulta falla
                res.json({ success: true }); // Enviar una respuesta indicando que el video fue eliminado
            });
        } else {
            res.status(404).send('Video no encontrado'); // Enviar una respuesta indicando que el video no fue encontrado
        }
    });
});
```

Figura 58. Fragmento de código de 'server.js' para eliminar un vídeo

Si un administrador desea eliminar todos los vídeos, server.js ejecuta una eliminación masiva de todos los registros en la tabla videos y borra todos los archivos relacionados en los directorios del servidor.

```
// Ruta para eliminar todos los videos
app.post('/delete_all_videos', (req, res) => {
    const deleteVideosSql = 'DELETE FROM videos';
    db.query(deleteVideosSql, (err, result) => {
        if (err) {
            console.error('Error eliminando videos de la base de datos:', err);
            return res.status(500).send('Error eliminando videos de la base de datos'); // Enviar una respuesta indicando que hubo un error al eliminar los videos
        }

        const directories = ['uploads', 'mpd', 'output'];

        directories.forEach(directory => {
            const dirPath = path.join(__dirname, directory);
            fs.readdir(dirPath, (err, files) => {
                if (err) {
                    console.error(`Error leyendo el directorio ${directory}:`, err);
                } else {
                    files.forEach(file => {
                        const filePath = path.join(dirPath, file);
                        if (fs.lstatSync(filePath).isDirectory()) {
                            fs.rmdirSync(filePath, { recursive: true }); // Eliminar directorios recursivamente
                        } else {
                            fs.unlinkSync(filePath); // Eliminar archivos
                        }
                    });
                }
            });
        });

        res.json({ success: true }); // Enviar una respuesta indicando que todos los videos fueron eliminados
    });
});
```

Figura 59. Fragmento de código de 'server.js' para eliminar todos los vídeos

5.2.3 Reproducción adaptativa

Gracias a todo lo mencionado anteriormente, si un usuario desea reproducir un vídeo subido a la plataforma, este se reproducirá de manera adaptativa.

Cuando se inicia la reproducción de un vídeo en la plataforma, el reproductor no carga un único archivo de vídeo. En su lugar, consulta el manifiesto DASH, el cual contiene las distintas versiones de este, incluyendo sus calidades y fragmentos

Dependiendo de la velocidad de conexión y la capacidad de procesamiento del dispositivo del usuario, el reproductor comienza a descargar y reproducir la versión del vídeo más adecuada. Si la conexión mejora o empeora, el reproductor cambia automáticamente a una versión de mayor o menor calidad, sin interrumpir la reproducción.

Cuando el usuario selecciona un vídeo para reproducir, el front-end envía una solicitud HTTP al servidor. Esta solicitud es manejada por 'server.js', el cual consulta la base de datos para obtener la información necesaria sobre el vídeo solicitado, incluyendo la ruta del manifiesto DASH (.mpd).

```
// Ruta para obtener todos los videos
app.get('/videos', (req, res) => {
    let sql = 'SELECT * FROM videos'; // Consulta SQL para obtener todos los videos
    db.query(sql, (err, results) => {
        if (err) throw err; // Lanzar un error si la consulta falla
        res.json(results); // Enviar los videos como respuesta en formato JSON
    });
});
```

Figura 60. Fragmento de código de 'server.js' para obtener todos los vídeos

Esta consulta devuelve un JSON al front-end con los detalles de todos los vídeos disponibles, incluyendo la ruta del manifiesto.

En el front-end, Shaka Player se encarga de manejar la reproducción del vídeo. Utilizando la URL del manifiesto DASH proporcionada por el servidor, carga el archivo .mpd y comienza a gestionar la reproducción.

```
// Usar Shaka Player para reproducir el video
const player = new shaka.Player(videoPlayer);

player.load(url).then(() => {
    console.log('El video ha sido cargado correctamente');
}).catch(error => {
    console.error('Error cargando el video:', error);
});
```

Figura 61. Fragmento de código de platform con la carga de la url del vídeo con Shaka Player

Shaka Player seleccione la calidad del vídeo en función del ancho de banda disponible y de las capacidades del dispositivo del usuario. Por ejemplo, si la conexión a internet es rápida, Shaka Player puede optar por reproducir la versión de 1080p o 2160p del vídeo. Si la conexión es más lenta, puede cambiar a una versión de menor calidad, como 720p o 540p, para asegurar que la reproducción sea fluida.

Mientras el vídeo se reproduce, Shaka Player continúa monitorizando la calidad de la conexión y el rendimiento del dispositivo. Si detecta que la conexión mejora, automáticamente puede cambiar a una versión de mayor calidad del vídeo. Por el contrario, si la conexión empeora, cambiará a una versión de menor calidad para evitar interrupciones.

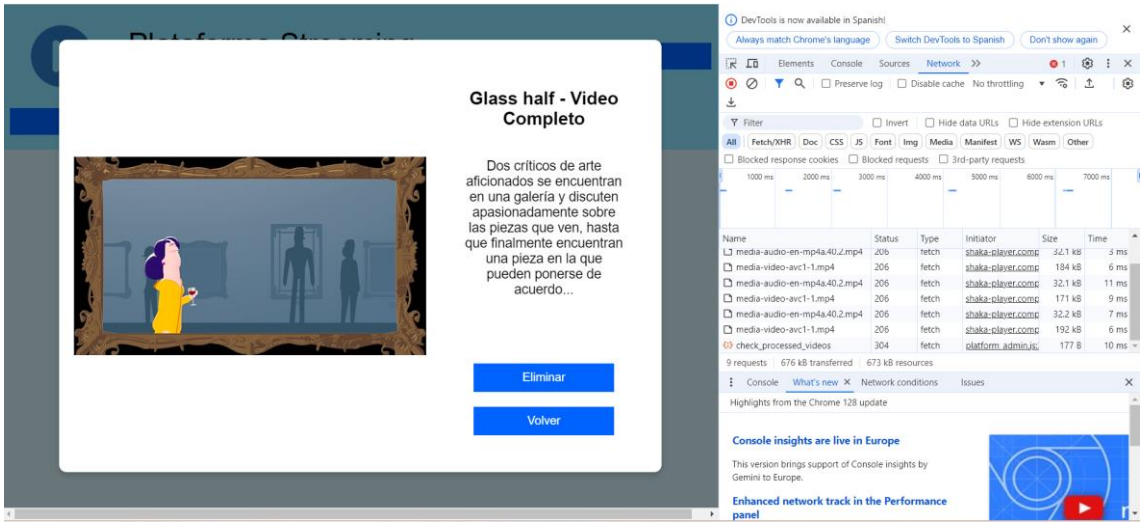

Figura 62. Reproducción de un vídeo sin limitar la conexión

Como podemos observar en la Figura 62, si la conexión de red es óptima el reproductor Shaka Player utiliza los fragmentos en calidad 720p (la máxima en la que está codificada este vídeo en concreto), que son los fragmentos 'media-video-avc1-1.mp4'.

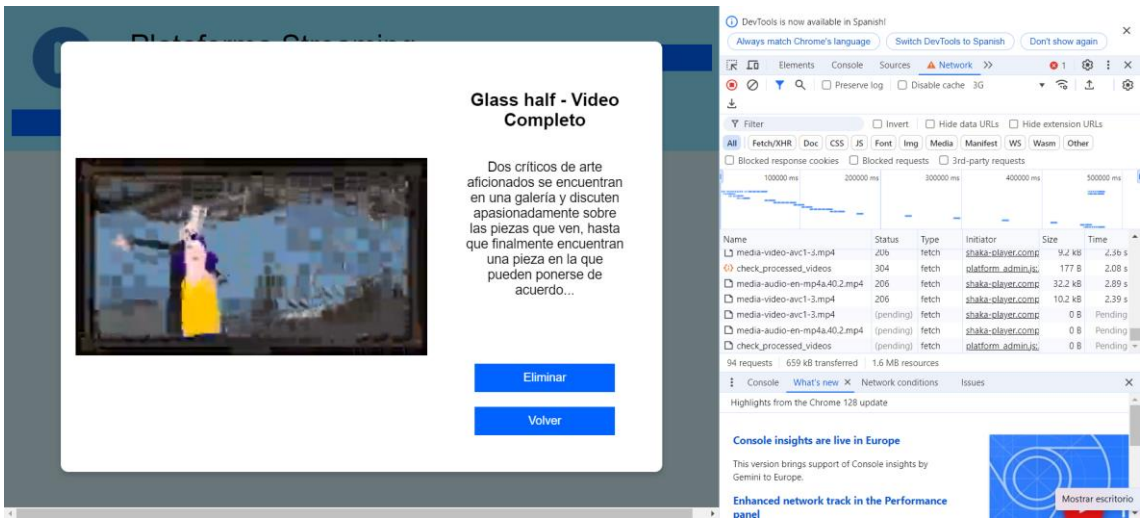

Figura 63. Reproducción de un vídeo con conexión limitada a 3G

Por el contrario, si limitamos la conexión a 3G, vemos en la Figura 63 que el reproductor está descargando los fragmentos 'media-video-avc1-3.mp4', que son los correspondientes a la calidad 270p.

Finalmente, si cortamos la conexión, el vídeo dejará de reproducirse.

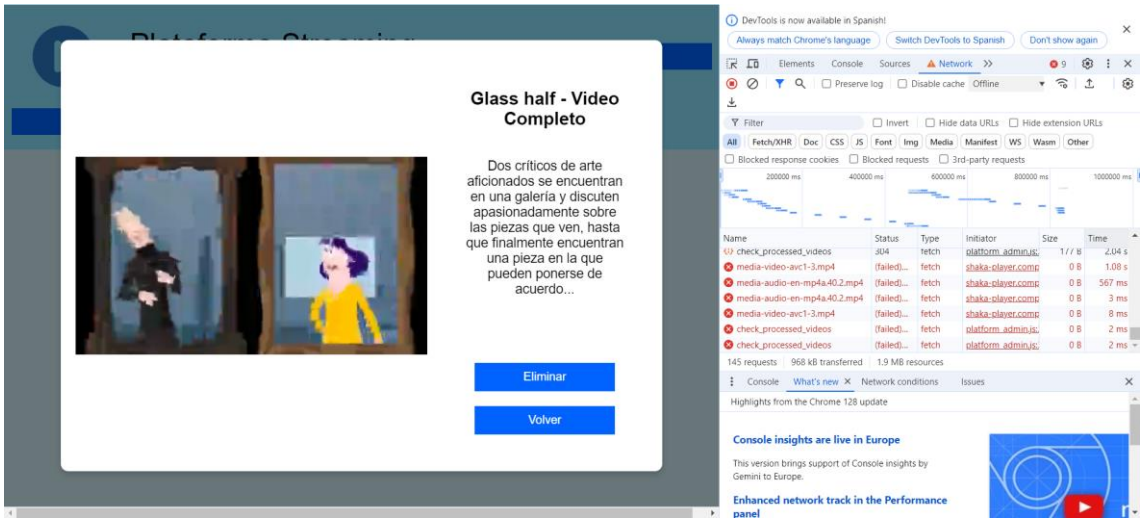

Figura 64. Reproducción de un vídeo offline

Capítulo 6. Conclusiones

6.1 Evaluación de cumplimiento de objetivos

Al llegar al final de este proyecto, es esencial revisar cómo se han cumplido los objetivos que me propuse al inicio. A lo largo del desarrollo de la plataforma, he trabajado con la meta de crear un sistema de streaming flexible y personalizable que se adapte a las necesidades de distintos usuarios y organizaciones.

El primer objetivo era ofrecer un alto nivel de personalización, permitiendo a los usuarios ajustar diversos elementos visuales de la plataforma, como el logo, los colores corporativos, el estilo de los botones, entre otros. Este objetivo se ha cumplido satisfactoriamente. La plataforma cuenta con un asistente de personalización (wizard) que permite configurar todos estos aspectos durante la configuración inicial, y un panel de administración que permite editar estas opciones en cualquier momento. Esto asegura que cada instancia de la plataforma pueda reflejar la identidad visual de sus usuarios de manera única.

El segundo objetivo era garantizar que la reproducción de los vídeos fuese adaptativa, ajustándose automáticamente a la calidad de la conexión de cada usuario. Este objetivo también se ha logrado con éxito. La integración de Shaka Player y la implementación del protocolo DASH han permitido que la plataforma ofrezca una experiencia de visualización fluida, adaptándose a las condiciones de red del usuario sin interrupciones.

El tercer objetivo era que la plataforma se encargara automáticamente de la codificación de los vídeos subidos por los usuarios. Este proceso se ha implementado mediante el uso de FFmpeg y Bento4, herramientas que permiten la codificación, fragmentación y creación de manifiestos de vídeo de forma automática. Los usuarios pueden subir vídeos sin preocuparse por el formato o la codificación, ya que la plataforma se encarga de procesarlos y prepararlos para la reproducción adaptativa.

En resumen, los objetivos planteados al inicio del proyecto se han cumplido en su totalidad. La plataforma desarrollada es flexible, personalizable y capaz de ofrecer una experiencia de calidad, adaptándose a las necesidades de cada usuario u organización.

6.2 Evaluación del alcance

El alcance de la plataforma ha cumplido con las expectativas, ya que puede utilizarse en una variedad de escenarios, como la formación educativa, plataformas privadas de streaming, y entornos corporativos. Su flexibilidad y capacidad de personalización permiten que sea útil para cualquier organización que necesite gestionar y distribuir contenido multimedia de manera eficiente y profesional.

En el siguiente apartado, se mostrarán ejemplos prácticos de cómo esta plataforma puede ser utilizada en diferentes contextos, demostrando su versatilidad y valor agregado.

6.2.1 Ejemplos de uso

Una de las posibles aplicaciones de esta plataforma es en el ámbito educativo, especialmente en universidades. Los profesores pueden grabar sus clases o crear videoapuntes y subirlos a la plataforma para que los estudiantes puedan verlos cuando lo necesiten. Además, la plataforma se puede personalizar con los colores y el logo de la universidad, lo que hace que sea un entorno más familiar para los estudiantes.

Figura 65. Posible uso de la plataforma 1

Como podemos ver en la Figura 65, se muestra una posible interfaz de la plataforma para una hipotética página de videoapuntes para la UPV. Podemos ver que la plataforma está completamente personalizada con el logo y los colores de la universidad, y además he subido algunos vídeos presentes en el canal de YouTube de la UPV para ayudar a visualizar mejor la idea.

También se puede usar esta plataforma en el ámbito corporativo. Las empresas pueden usar esta plataforma para la formación de sus empleados. Ya sea para cursos técnicos, webinars…etc. L

Los empleados pueden acceder a los vídeos a su propio ritmo.

Figura 66. Posible uso de la plataforma 2

En la Figura 66, se muestra un ejemplo de uso de la plataforma para el ámbito corporativo, en este caso para la empresa Amazon, con tutoriales para los empleados o vídeos de identidad corporativa.

Estos ejemplos son solo algunas de las muchas formas en que la plataforma puede ser utilizada. La flexibilidad y capacidad de personalización permiten que se adapte a una gran variedad de necesidades, ya sea en la educación, en el ámbito corporativo, o en la difusión de contenidos especializados. Esto la convierte en una herramienta útil para cualquier organización que necesite compartir contenido multimedia de forma eficiente y profesional.

6.3 Propuesta de trabajo futuro

Aunque la plataforma ya ofrece una base sólida para la gestión y distribución de contenidos multimedia, hay varias mejoras y nuevas funciones que podrían desarrollarse en el futuro para hacerla aún más útil y completa. A continuación, presento algunas ideas para el desarrollo futuro de la plataforma:

- **Edición de metadatos del vídeo:** Sería muy práctico permitir la edición de detalles como el título, la descripción y la miniatura de los vídeos directamente desde la plataforma. Esto facilitaría a los administradores realizar ajustes rápidos y mantener el contenido actualizado sin necesidad de volver a subir los vídeos.
- **Emisión de vídeos en directo (streaming en vivo):** Añadir la opción de transmitir en vivo desde la plataforma sería un gran paso. Esto permitiría la realización de clases en tiempo real, webinars o eventos en vivo, ampliando las posibilidades de uso de la plataforma.
- **Perfiles de usuario y seguimiento de actividad:** Crear perfiles de usuario personalizados ayudaría a identificar quién ha subido cada vídeo y quiénes lo han visto. Esta funcionalidad sería útil para llevar un control más detallado en el ámbito educativo o corporativo, permitiendo evaluar la participación y el interés en los contenidos.
- **Buscador de vídeos:** Un buscador dentro de la plataforma facilitaría mucho la tarea de encontrar vídeos específicos, especialmente a medida que el catálogo de contenido crezca. Esto mejoraría la experiencia del usuario, haciéndola más intuitiva y eficiente.
- **Categorías y etiquetas para organizar el contenido:** Implementar un sistema de categorías y etiquetas ayudaría a organizar mejor los vídeos. Esto haría que sea más fácil para los usuarios descubrir contenido relevante según sus intereses o necesidades.
- **Sistema de comentarios y retroalimentación:** Incluir un espacio para que los usuarios puedan dejar comentarios o feedback en los vídeos sería una adición valiosa, especialmente en entornos educativos. Esto permitiría una mayor interacción entre los usuarios y los creadores de contenido, fomentando el aprendizaje colaborativo y el intercambio de ideas.
- **Mejoras en la seguridad y control de acceso:** Desarrollar características avanzadas de control de acceso y seguridad, como autenticación de dos factores o restricciones geográficas, podría ser crucial para proteger el contenido y garantizar que solo las personas autorizadas tengan acceso a él.

Estas propuestas de mejora no solo aumentarían las capacidades de la plataforma, sino que también mejorarían la experiencia del usuario, haciéndola más versátil y atractiva para diferentes contextos y necesidades. Con estas adiciones, la plataforma podría convertirse en una herramienta aún más integral para la gestión y distribución de contenidos multimedia.

6.4 Reflexión personal

Hacer este proyecto ha sido una experiencia muy valiosa para mí, no solo en lo referido al aprendizaje técnico y académico, sino también como un reto personal. A lo largo del desarrollo del trabajo, me he dado cuenta de lo mucho que he crecido desde que empecé la carrera. He tenido que enfrentar problemas que en un principio parecían muy difíciles, pero que con esfuerzo y dedicación logré resolver, lo que me ha hecho sentir realmente orgulloso.

Trabajar en la personalización de la plataforma y en el desarrollo del servidor Node.js fueron, sin duda, las partes más complicadas, pero también las más gratificantes. Al principio, todo parecía un poco abrumador, pero conforme fui avanzando y viendo cómo el proyecto tomaba forma, me sentí cada vez más seguro y motivado.

Algo que me llevo de esta experiencia es la importancia de planificar bien las cosas y saber gestionar el tiempo. Al principio, cometí el error de subestimar el tiempo que me llevarían algunas tareas, lo que me obligó a reorganizarme y aprender a priorizar lo realmente importante. Esta lección me va a ser muy útil en el futuro, tanto en el ámbito profesional como en la vida en general.

En resumen, este proyecto ha sido más que un trabajo académico; ha sido una prueba de lo que soy capaz de hacer y una confirmación de que estoy en el camino correcto. Estoy contento con lo que he logrado y con muchas ganas de seguir aprendiendo y mejorando en el desarrollo de plataformas digitales.

Capítulo 7. Anexos

La totalidad del código del proyecto está subido a mi repositorio de GitHub en el siguiente enlace:

https://github.com/Fenanes/Plataforma-de-Streaming-Web-Personalizable---TFG-Pablo-Fernandez-Beaus

Capítulo 8. Bibliografía

[1] Mozilla Developer Network (MDN). "HTML: HyperText Markup Language," https://developer.mozilla.org/en-US/docs/Web/HTML. [Online].

[2] Mozilla Developer Network (MDN). "CSS: Cascading Style Sheets," https://developer.mozilla.org/en-US/docs/Web/CSS. [Online].

[3] Mozilla Developer Network (MDN). "JavaScript," https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Online].

[4] Node.js Foundation. "Node.js Documentation," https://nodejs.org/en/docs/. [Online].

[5] Bonisteel, S. (Kinsta) "Deno vs Node.js: Análisis de los Dos Runtimes" https://kinsta.com/es/blog/deno-vs-node-js/ [Online].

[6] MySQL AB. "MySQL 8.0 Reference Manual," https://dev.mysql.com/doc/refman/8.0/en/. [Online].

[7] Airbyte "SQLite vs MySQL: Features, Performance, and Use Cases" https://airbyte.com/data-engineering-resources/sqlite-vs-mysql [Online].

[8] Google. "Shaka Player Documentation," https://shaka-player-demo.appspot.com/docs/api/index.html [Online].

[9] FFmpeg Team. "FFmpeg Documentation," https://ffmpeg.org/documentation.html. [Online].

[10] Axiom. "Bento4 Documentation," https://www.bento4.com/documentation/. [Online].

[11] ISO/IEC 23009-1:2022. "Information Technology — Dynamic Adaptive Streaming Over HTTP (DASH) — Part 1: Media Presentation Description and Segment Formats." International Organization for Standardization, 2022.

[12] Richardson, I. E. "The H.264 Advanced Video Compression Standard", 2nd ed. Wiley, 2010. ISBN: 9780470516928.

[13] Rainer, S. Lederer, C. Müller and C. Timmerer, "A seamless Web integration of adaptive HTTP streaming," 2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO), Bucharest, Romania, 2012, pp. 1519-1523