Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Diseño y Automatización de Generación de Video Clips bajo Demanda

Design and Automation of On-Demand Video Clip Generation

Jacobo Labrador González

La Laguna, 12 de julio de 2023

Dña. **María Elena Sánchez Nielsen**, con N.I.F. 42.848.599-J profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

C E R T I F I C A (N)

Que la presente memoria titulada:

"Diseño y Automatización de Generación de Video Clips bajo Demanda"

ha sido realizada bajo su dirección por D. **Jacobo Labrador González**,

con N.I.F. 43.380.041-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 12 de julio de 2023

Agradecimientos

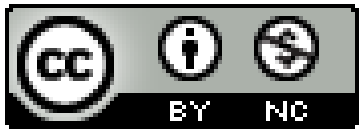
A mis padres y mi hermano, por ser pilares fundamentales fuente de inspiración y guías.

A mi pareja, por su paciencia y comprensión.

A mis amigos por ser compañeros de viaje y haberme sacado una sonrisa cuando más lo he necesitado.

Y a mi tutora María Elena por ser la persona de guíe de la mejor manera al compartir su conocimiento durante este proyecto.

Licencia



Resumen

El objetivo de este proyecto es el desarrollo de una aplicación de generación de clips bajo demanda que brinde a los usuarios una herramienta fácil de usar y versátil para crear contenido audiovisual. La aplicación actúa como un estudio virtual que permite a los usuarios la generación de dichos clips de manera cómoda y natural.

La idea de desarrollo surge como una respuesta a la creciente demanda de herramientas de edición de video accesibles y prácticas. Aprovechando los avances en tecnología móvil y procesamiento de imágenes, se busca implementar una aplicación que permita a los usuarios capturar momentos especiales y convertirlos en piezas únicas de contenido audiovisual.

La aplicación contará con herramientas de recorte precisas que permitirán a los usuarios seleccionar y extraer segmentos específicos de sus videos, facilitando la creación de clips cortos o resúmenes de eventos más largos.

El proyecto se desarrollará utilizando tecnologías de vanguardia en procesamiento de imágenes y algoritmos de edición de video. Se espera que la aplicación sea especialmente útil para aquellos que deseen crear contenido para plataformas de redes sociales, blogs o presentaciones profesionales, brindando una experiencia de edición de video intuitiva y obteniendo un resultado de alta calidad.

Palabras clave: recortes, generación de clips, herramienta de edición, videos

Abstract

The objective of this project is the development of an on-demand clip generation application that provides users with an easy-to-use and versatile tool for creating audiovisual content. The application acts as a virtual studio that allows users to generate such clips in a comfortable and natural manner.

The development idea arises as a response to the growing demand for accessible and practical video editing tools. Leveraging advancements in mobile technology and image processing, the aim is to implement an application that enables users to capture special moments and turn them into unique pieces of audiovisual content.

The application will feature precise trimming tools that will allow users to select and extract specific segments from their videos, facilitating the creation of short clips or summaries of longer events.

The project will be developed using cutting-edge technologies in image processing and video editing algorithms. The application is expected to be particularly useful for those who wish to create content for social media platforms, blogs, or professional presentations, providing an intuitive video editing experience and delivering high-quality results.

Keywords: trims, clip generation, editing tool, videos

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Antecedentes y contexto

La generación de clips de video se ha vuelto cada vez más popular en la era digital, donde la creación y el consumo de contenido audiovisual son tendencias dominantes. Con el auge de las plataformas de redes sociales, los blogs y las presentaciones en línea, existe una creciente demanda de herramientas accesibles y prácticas que permitan a los usuarios crear y compartir clips de video de manera fácil y rápida.

En este contexto, el proyecto de generación de clips tiene como objetivo desarrollar una aplicación que satisfaga esta demanda y proporcione a los usuarios una herramienta versátil y fácil de usar para crear contenido audiovisual de alta calidad. El desarrollo de esta aplicación surge como respuesta a la necesidad de contar con una herramienta eficiente y accesible para la edición de videos. Aprovechando los avances en tecnología móvil y procesamiento de imágenes, se busca ofrecer a los usuarios la capacidad de realizar recortes precisos en sus videos, seleccionando y extrayendo segmentos específicos según sus necesidades.

Esta aplicación de generación de clips será especialmente útil para creadores de contenido, influencers, profesionales de marketing y cualquier persona que desee crear videos cortos. Con esta herramienta, los usuarios podrán expresar su creatividad y contar historias de manera visualmente atractiva, capturando la atención de su audiencia y logrando un mayor impacto en sus mensajes.

En resumen, el proyecto de generación de clips tiene como objetivo brindar a los usuarios una herramienta eficiente y versátil para crear clips de video, permitiéndoles capturar momentos especiales y destacar lo más relevante de sus videos. Mediante el uso de tecnologías avanzadas, se busca ofrecer una experiencia de edición de video intuitiva y obtener resultados de alta calidad que se adapten a las necesidades y exigencias de la era digital actual.

1.2 Objetivos

El objetivo principal del proyecto es desarrollar un sistema que combine un reproductor open-source y una herramienta de generación de clips. Este sistema permitirá a los usuarios especificar un instante de inicio y un instante final para obtener un clip de vídeo personalizado. Para lograrlo, se deben cumplir los siguientes aspectos:

- **Utilizar un reproductor open-source HTML5**

Se seleccionará y utilizará un reproductor open-source basado en la tecnología HTML5 [1]. Esto asegurará que los usuarios puedan visualizar los vídeos de forma óptima y compatible con diferentes dispositivos y navegadores web.

- **Permitir al usuario especificar los instantes de inicio y final**

El sistema brindará una interfaz intuitiva que permitirá a los usuarios seleccionar el instante de comienzo y final del vídeo que desean obtener como clip. Esta funcionalidad permitirá una personalización precisa de los clips de vídeo.

- **Generar y descargar el clip de vídeo**

Una vez que el usuario haya especificado los instantes de inicio y final, el sistema generará automáticamente el clip de vídeo correspondiente. El usuario podrá descargar el clip resultante para su uso posterior o compartirlo en otras plataformas.

- **Notificar al usuario cuando el clip esté disponible**

Se implementará un mecanismo de notificación que informará al usuario cuando el clip de vídeo solicitado esté listo para su descarga. Esto asegurará que el usuario esté al tanto del estado de procesamiento y pueda acceder al clip tan pronto como esté disponible.

Capítulo 2
Estado del arte

En el campo de la generación de clips de vídeo, existen diversas herramientas y tecnologías que se han desarrollado en los últimos años. Actualmente, se han popularizado aplicaciones y software de edición de vídeo que permiten a los usuarios recortar y editar sus vídeos de manera personalizada. Estas herramientas suelen ofrecer una amplia gama de funcionalidades y opciones de edición, pero pueden resultar complejas y requieren un cierto nivel de conocimientos técnicos. Además, la mayoría de estas herramientas se centran en la edición completa de vídeos, sin ofrecer una opción específica para generar clips cortos.

2.1 Limitaciones de las herramientas existentes

A pesar de la existencia de herramientas de edición de vídeo, muchas de ellas presentan limitaciones en cuanto a la generación de clips de vídeo cortos. Algunas de estas limitaciones incluyen la falta de opciones específicas para establecer tiempos de inicio y fin precisos, la dificultad para realizar recortes en intervalos de tiempo muy cortos o la ausencia de funcionalidades para automatizar el proceso de generación de clips. Además, estas herramientas suelen requerir una instalación local en el dispositivo del usuario, lo que limita su accesibilidad y portabilidad.

2.2 Comparativa del aplicativo con aplicaciones existentes

Con el fin de realizar una comparación más efectiva, se buscará establecer un contexto común entre todas las aplicaciones consideradas. Cabe destacar que el aplicativo desarrollado se encuentra en una fase inicial de implementación, pero cuenta con una capacidad escalable y mejorable al agregar nuevas funciones y mejoras. Con este propósito, se han seleccionado páginas web que desempeñan una función similar a la del aplicativo desarrollado.

Es importante mencionar que muchas de estas herramientas requieren una instalación local, como Adobe Premiere Pro, iMovie o Shotcut, las cuales son potentes e interesantes, pero no serán incluidas en la comparación que se realizará.

La comparación se llevará a cabo mediante una tabla en la que se marcarán las funciones que posea cada aplicación. Para evaluar los campos pertinentes, se podrán utilizar los objetivos establecidos en el capítulo anterior.

Para asignar un nombre al proyecto que se introducirá en la tabla, lo llamaremos "Aplicativo". Lo compararemos con otras aplicaciones web como Flixier, Clideo, Online Video Cutter y Flexclip. Los aspectos a considerar en la comparación son los siguientes:

- **Utilización de un reproductor para el archivo original:** se evaluará si cada aplicación ofrece un reproductor para visualizar el archivo original.
- **Introducción de datos de entrada para establecer los tiempos de inicio y final de forma precisa**: se verificará si es posible ingresar datos precisos para establecer los tiempos de inicio y finalización del video.
- **Generación del video clip:** se analizará la capacidad de cada aplicación para generar el video clip resultante según las especificaciones establecidas.
- **Capacidad de notificación:** se evaluará si las aplicaciones ofrecen la capacidad de notificar al usuario sobre el estado o finalización del proceso.

	Aplicativo	**Flixier**	**Clideo**	**Online Video Cutter**	**Flexclip**
Reproductor	X		X	X	
Datos de entrada	X	*	*	X	*
Generación del clip	X	X	X	X	X
Capacidad de notificación	X				

Tabla 2.1 Comparativa entre aplicaciones para la generación de video clips

Bajo el contexto establecido, el aplicativo desarrollado se define como una herramienta completa y versátil. En contraste con otras herramientas, no todas poseen la capacidad de reproducir el archivo original y luego introducir los datos de intervalo de tiempo deseado. Para aquellas que no ofrecen esta función, permiten cargar el video en el marco de trabajo y realizar los cambios a partir de ahí.

En cuanto a la introducción de datos de inicio y finalización del video clip, todas las herramientas cuentan con alguna capacidad para realizar esta selección, aunque difieren en su enfoque. Flixier, Clideo y Flexclip utilizan la función de arrastre para marcar el segmento, lo cual puede ser un poco más complicado al tratar de seleccionar un fotograma específico en videos de larga duración. Se ha marcado con un asterisco (*) debido a que estas herramientas también ofrecen la opción de introducir datos numéricos, aunque la precisión puede variar. Tanto el aplicativo desarrollado como Online Video Cutter permiten establecer el momento exacto de forma numérica. Además, Online Video Cutter combina ambas funcionalidades, lo cual resulta muy completo.

En cuanto a la capacidad de notificación cuando se genera un proyecto de clip, ninguna de las herramientas consideradas ofrece esta función. Esto posiciona al aplicativo desarrollado como único en su capacidad de notificar al usuario sobre el estado o finalización del proceso, destacándose sobre las opciones existentes.

En resumen, el aplicativo desarrollado se destaca por su conjunto de funcionalidades completas en comparación con las otras herramientas consideradas. Sin embargo, estas aplicaciones se encuentran en un nivel superior en términos de características adicionales debido a que han tenido un mayor recorrido a lo largo del tiempo y se han agregado características y nuevas funcionalidades regularmente.

2.3 Aplicaciones potenciales

Edición rápida de vídeos para redes sociales

Los usuarios pueden utilizar la aplicación para generar clips cortos de vídeo que se adapten a los requisitos y restricciones de plataformas de redes sociales como Instagram, TikTok o YouTube.

Creación de muestras o demos

La aplicación puede ser útil para aquellos que necesiten crear muestras o demos de vídeos largos, permitiéndoles recortar y extraer fragmentos específicos para su presentación o promoción.

Generación de contenido educativo

Los profesionales de la educación pueden aprovechar la aplicación para crear clips de vídeo cortos que muestren conceptos específicos o fragmentos relevantes de clases grabadas, facilitando así el acceso y la comprensión del contenido para los estudiantes.

Producción audiovisual

En el ámbito de la producción audiovisual, la aplicación puede ser utilizada por profesionales y aficionados para generar clips de vídeo cortos a partir de material grabado, agilizando así el proceso de selección y edición de secuencias.

Capítulo 3
Metodología

Para planificar el proyecto, lo primero que se hizo en colaboración con la tutora fue definir los objetivos que se debían alcanzar. Una vez establecidos, se fijó un cronograma para cumplir con dichos objetivos. A medida que se iban logrando los objetivos, se programaron reuniones con la tutora para revisar el estado actual del proyecto y verificar si los resultados eran satisfactorios.

La idea principal era que, en cada revisión o seminario, el proyecto debía evolucionar y cumplir nuevos objetivos en comparación con la revisión anterior. Esto se asemeja a un enfoque de planificación **Scrum** [2]. Las tareas específicas se dividieron en tres bloques diferentes, los cuales serán explicados detalladamente más adelante.

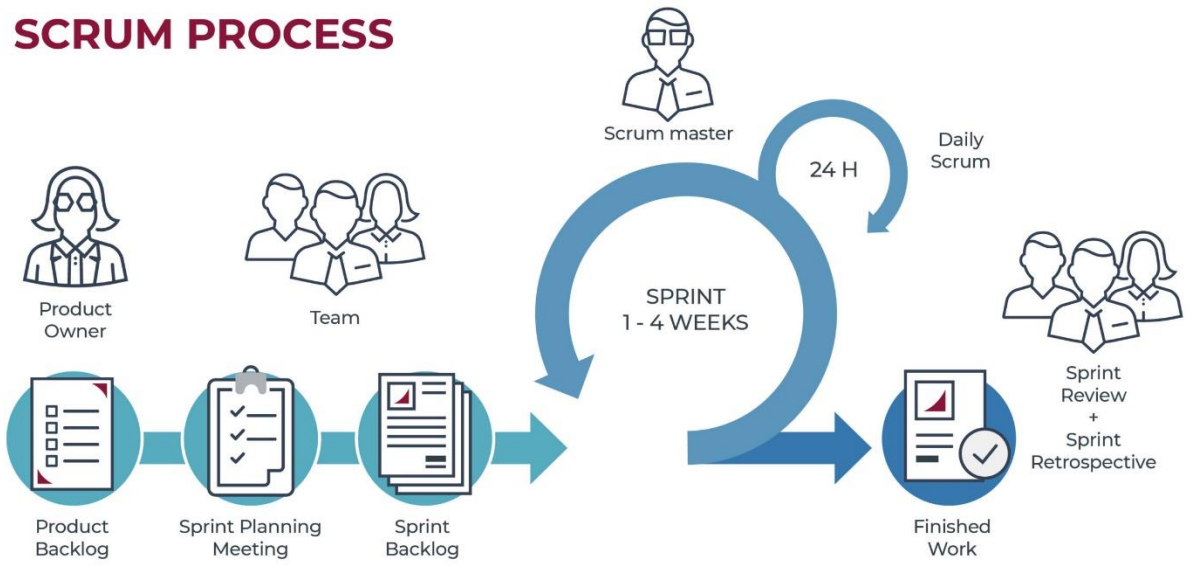


Figura 3.1 Scrum

A través de las siguientes tablas se planteará como se ha llevado el proyecto y qué tareas han estado en cada etapa.

3.1 Estudio previo y planteamiento

Tarea	Inicio	Fin
Estudio y comparativa de soluciones de cortes de video existentes	17/01/2023	25/01/2023
Estudio y comparativa de reproductores de video HTML5 Open Source	26/01/2023	31/01/2023
Selección de tecnologías para la aplicación	1/02/2023	10/02/2023
Diseño preliminar de la aplicación	11/02/2023	26/02/2023

Tabla 3.1 Tareas de la fase de estudio previo y planteamiento

El bloque de estudio previo y planteamiento consta de las cuatro tareas mencionadas en la tabla anterior. En las dos primeras tareas, se busca explorar diferentes soluciones para la generación de cortes de video, así como opciones para la reproducción del video. Una vez se hayan identificado estas tecnologías, se procederá a seleccionar el framework más adecuado, teniendo en cuenta su compatibilidad y, en caso de ser posible, la experiencia previa con él. Por último, se realizó un diseño preliminar de la aplicación para obtener una idea de su apariencia futura.

3.2 Implementación

Tarea	Inicio	Fin
Visualización del vídeo	10/03/2023	25/03/2023
Elección de los instantes iniciales y finales	26/03/2023	7/04/2023
Generación del corte	08/04/2023	22/04/2023
Descarga del clip generado	23/04/2023	14/05/2023
Notificación al usuario	15/05/2023	31/05/2023

Tabla 3.2 Tareas de la fase de implementación

Después de seleccionar las soluciones para la herramienta de corte de video, el reproductor de video HTML Open Source y la tecnología de la aplicación, y haber establecido un primer diseño, es el momento de pasar a la implementación. Esta segunda fase consta de cinco tareas principales, cuyas fechas están indicadas en la tabla anterior.

La primera tarea consiste en permitir que la aplicación reproduzca el video que se desea editar utilizando un reproductor HTML5 Open Source, específicamente VideoJS [3]. En la segunda tarea, se comienza a implementar los campos para la selección de los momentos de inicio y fin, asegurándose de que estos valores se envíen de manera correcta y precisa a su destino.

Una vez se disponen de todos los parámetros necesarios para generar el corte, se procede a la tercera tarea: crear el clip utilizando FFMPEG [4] con la información proporcionada. A continuación, se implementa la descarga del video generado, permitiendo al usuario guardar el clip que ha creado. Por último, se incorpora la opción para que el usuario decida si desea recibir una notificación cuando el clip esté listo.

En resumen, esta fase de implementación abarca la reproducción del video, la selección de los momentos de inicio y fin, la generación del clip, la descarga del video y la opción de notificación al usuario.

3.3 Evaluación del funcionamiento

Tarea	Inicio	Fin
Test de usuarios	05/06/2023	19/06/2023
Análisis de resultados	20/06/2023	23/06/2023

Tabla 3.3 Tareas de la fase de evaluación del funcionamiento

En la fase final, se realizará el test de usuario. En esta etapa, se involucrará a un grupo diverso de usuarios con distintas edades, formación y gustos, quienes llevarán a cabo una serie de pruebas. Posteriormente, se les pedirá que compartan su opinión a través de un cuestionario completamente anónimo. Este proceso permitirá recopilar diferentes perspectivas y tomar decisiones respecto a posibles correcciones o nuevas implementaciones del proyecto. El *feedback* obtenido será valioso para mejorar y adaptar el producto de acuerdo a las necesidades y preferencias de los usuarios.

Capítulo 4
Diseño y Desarrollo

4.1 Selección de las herramientas para el desarrollo

En este capítulo, se presentarán detalladamente las implementaciones realizadas en el proyecto, así como las herramientas utilizadas. Se explicará por qué se seleccionaron estas herramientas en particular, destacando sus ventajas y su relevancia para el proceso de generación de clips. Además, se incluirá un diagrama de flujo que ilustra claramente el proceso mencionado.

4.1.1 Análisis de las herramientas clave para la manipulación eficiente de clips de video

En este apartado, se realizará una comparativa entre las diferentes tecnologías existentes para la generación de cortes de vídeo. Se analizarán las razones por las que se ha seleccionado FFMPEG sobre otras tecnologías como OpenCV [5] y LIBAV [6].

Figura 4.1 FFMPEG

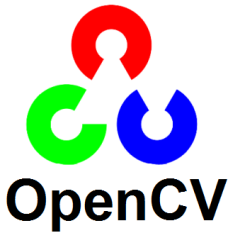


Figura 4.2 OpenCV

FFMPEG es una biblioteca de procesamiento de audio y video reconocida por su especialización en el manejo de archivos multimedia. Esta herramienta fue propuesta por la tutora del proyecto como una candidata seria para su utilización. Sin embargo, se llevó a cabo un estudio comparativo considerando otras posibilidades, como OpenCV y LIBAV.

En comparación con OpenCV, FFMPEG se destaca por ser una herramienta especializada en videos con una amplia compatibilidad de formatos. Esto facilita la generación de cortes en diferentes formatos y asegura la interoperabilidad con diversos sistemas y dispositivos. Además, FFMPEG está optimizado para el procesamiento eficiente de archivos de video, lo que permite realizar operaciones como la extracción de segmentos de manera rápida y precisa. Por otro lado, aunque OpenCV es válido para generar videos, su enfoque principal está más orientado al procesamiento de imágenes y visión por computadora, lo que puede limitar sus capacidades en la manipulación de videos.

En cuanto a LIBAV, se trata de una bifurcación de FFMPEG que surgió en el pasado, alrededor del año 2012. Sin embargo, con el paso del tiempo, su soporte se ha reducido y no existe un desarrollo activo, lo que ha llevado a descartar su consideración como opción para el proyecto.

Por lo tanto, considerando las necesidades específicas del proyecto, se ha seleccionado FFMPEG sobre otras tecnologías debido a su especialización en el procesamiento de video, su amplia compatibilidad de formatos, eficiencia y estabilidad.

4.1.2 Análisis de las soluciones existentes para reproductores de video HTML 5 Open-source

En este apartado, se realizará un análisis de las distintas soluciones disponibles para la reproducción de videos en sitios web utilizando HTML de código abierto. Se examinarán y discutirán las razones por las cuales se ha elegido VideoJS sobre otras alternativas, como el reproductor nativo de HTML o jPlayer [7] de jQuery.

Figura 4.3 jPlayer

VideoJS es un reproductor de video HTML5 de código abierto ampliamente adoptado debido a su amplia compatibilidad con diversos navegadores y dispositivos. Esto asegura una experiencia de reproducción consistente en diferentes plataformas, además de ofrecer una gran capacidad de personalización. Asimismo, cuenta con una comunidad activa de desarrolladores que brindan soporte, actualizaciones periódicas y mejoras continuas a la biblioteca. Esta sólida base de soporte

garantiza que la aplicación se mantenga actualizada y que cualquier incidencia o vulnerabilidad pueda ser abordada rápidamente.

En contraste, el reproductor nativo de HTML se queda rezagado en comparación con VideoJS. Aunque es una opción para la reproducción de videos en navegadores web sin requerir bibliotecas externas, presenta limitaciones en cuanto a personalización. Si bien el reproductor nativo de HTML recibe actualizaciones y mejoras a través de las actualizaciones del navegador, existen diferencias en la implementación y funcionalidad entre los distintos navegadores, lo que puede afectar la experiencia del usuario.

Figura 4.4 VideoJS

En cuanto a jPlayer de jQuery, si bien es una opción válida, se ha descartado debido a que su uso implica la dependencia de jQuery como una biblioteca adicional. Esto no solo aumentaría la carga de la página, sino que también podría complicar el mantenimiento del proyecto. Además, el desarrollo y las actualizaciones de jPlayer pueden ser menos frecuentes en comparación con soluciones más populares como VideoJS, lo que podría impactar en la estabilidad y las mejoras continuas del reproductor.

En resumen, VideoJS se ha destacado como la solución preferida para la reproducción de videos en aplicaciones web debido a su amplia compatibilidad, capacidad de personalización, funcionalidades adicionales y un sólido soporte comunitario. Estas características lo convierten en una opción sólida y confiable para el proyecto en cuestión.

4.2 Tecnologías utilizadas

El desarrollo del servidor en este proyecto se basa en el uso de tecnologías clave que permiten su implementación y funcionamiento adecuado.

En primer lugar, se utiliza Node.js [8] como el entorno de ejecución del lado del servidor. Node.js permite ejecutar código JavaScript [9] fuera de un navegador web y proporciona capacidades para el manejo de operaciones de entrada/salida de red y de sistemas de archivos. Esto hace que sea una opción ideal para construir aplicaciones del lado del servidor.

Figura 4.5 Node.js

Para facilitar la creación de la aplicación web y su correspondiente API, se utiliza el framework Express. Express proporciona una capa de abstracción que simplifica el proceso de desarrollo al manejar las solicitudes HTTP, definir rutas y gestionar el middleware. Su enfoque minimalista y su flexibilidad lo convierten en una elección popular para construir servidores web en Node.js.

Además, se utiliza la biblioteca CORS [10] para habilitar el acceso a recursos del servidor desde dominios diferentes al propio servidor. Esto es especialmente importante cuando se desarrollan aplicaciones web o APIs que deben permitir solicitudes desde diferentes dominios, solucionando problemas de seguridad relacionados con las políticas de mismo origen en los navegadores web.

Para el manejo de formularios multipart/form-data [11] y la carga de archivos en el servidor, se emplea la biblioteca Multer [12]. Multer simplifica el proceso de recepción de archivos enviados a través de solicitudes HTTP, lo cual es útil cuando se trabaja con la subida de videos u otros archivos multimedia.

Figura 4.6 Multer

Para llevar a cabo la manipulación y recorte de videos, se utiliza la biblioteca Fluent-FFMPEG [13]. Esta biblioteca permite interactuar con el software de línea de comandos FFMPEG, proporcionando una interfaz fácil de usar para realizar tareas de procesamiento de videos, como el recorte en este caso.

Por último, para cumplir con el requisito de notificar a los usuarios cuando su clip de video esté listo para descargar, se utiliza la biblioteca Nodemailer [14]. Nodemailer facilita el envío de correos electrónicos desde el servidor, lo que permite enviar notificaciones personalizadas a los usuarios, informándoles sobre el estado de su clip de video.

Figura 4.7 Nodemailer

En conjunto, estas tecnologías y bibliotecas permiten implementar un servidor robusto y funcional que atiende solicitudes HTTP, carga y manipula archivos, recorta videos y envía notificaciones a los usuarios.

El frontend de esta aplicación se ha desarrollado utilizando React [15], una biblioteca de JavaScript que permite construir interfaces de usuario interactivas y reactivas. React es ampliamente utilizado debido a su enfoque en la reutilización de componentes y su capacidad para manejar eficientemente el estado de la aplicación.

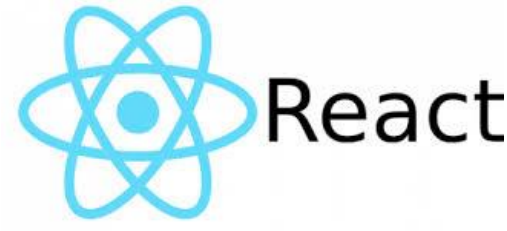


Figura 4.8 React

Para la reproducción de videos, se ha utilizado Video.js, una biblioteca de código abierto que ofrece un reproductor de video personalizable y altamente funcional. Video.js facilita la integración de videos en la aplicación web, proporcionando controles de reproducción, opciones de personalización y una amplia gama de funciones adicionales.

Axios se ha empleado para manejar las solicitudes HTTP entre el frontend y el backend de forma asíncrona. Esta biblioteca de JavaScript facilita el envío de solicitudes al servidor y el manejo de las respuestas recibidas. Axios [16] es ampliamente utilizado debido a su simplicidad y su capacidad para trabajar con diversos tipos de solicitudes y respuestas.

Las tecnologías utilizadas en el desarrollo del proyecto se han seleccionado en base a la familiaridad y experiencia previa con la mayoría de ellas a lo largo de la formación académica. Esto ha permitido un desarrollo más eficiente y efectivo, aprovechando las capacidades y ventajas que cada una ofrece.

Sin embargo, también se han incorporado tecnologías como VideoJS y Fluent-FFMPEG, que no se había utilizado previamente. Aunque no se contaba con conocimiento o experiencia previa en estas herramientas, gracias a las indicaciones y recomendaciones de la tutora, no ha supuesto un obstáculo. Se ha dedicado tiempo al estudio, exploración y selección de las soluciones existentes antes de tomar la decisión de que herramienta es adecuada para incorporar al proyecto.

Este enfoque ha resultado en la adquisición de experiencia y un cierto nivel de soltura en el uso de VideoJS y Fluent-FFMPEG durante el desarrollo de la aplicación. A través del estudio y la práctica, se ha logrado comprender y aplicar estas tecnologías de manera efectivo, lo que ha sido evidente en el desarrollo exitoso de la aplicación.

Por lo que esta combinación de familiaridad, exploración y aprendizaje ha sido clave para aprovechar al máximo las tecnologías utilizadas en el proyecto.

Se ha elegido el servicio de hosting Vercel para llevar a cabo el despliegue. Esta decisión se basa en varias razones fundamentales. En primer lugar, Vercel se destaca por su rapidez y facilidad de despliegue, lo que permite ahorrar tiempo y esfuerzo en el proceso de implementación. Además, ofrece una amplia compatibilidad con diversos frameworks y lenguajes de programación, lo que brinda flexibilidad y libertad para utilizar las herramientas preferidas en el proyecto.

Figura 4.9 Vercel

La integración de Vercel con Git y el control de versiones es otro aspecto destacado. Esta característica permite una gestión eficiente del desarrollo y despliegue continuo, lo que resulta en un flujo de trabajo más fluido y una mayor productividad.

Además, se ha tenido una experiencia previa con Vercel en otras asignaturas a lo largo de la carrera universitaria, lo que ha generado confianza en su rendimiento y funcionalidad. Esta familiaridad y conocimiento previo han sido factores determinantes en la elección de Vercel como el servicio de hosting preferido.

En resumen, la selección de Vercel se ha basado en su rápido y sencillo despliegue, su amplia compatibilidad con frameworks y lenguajes de programación, su integración con Git y control de versiones, y la experiencia previa positiva con su utilización en otros cursos universitarios.

4.3 Etapas clave en el flujo de la aplicación

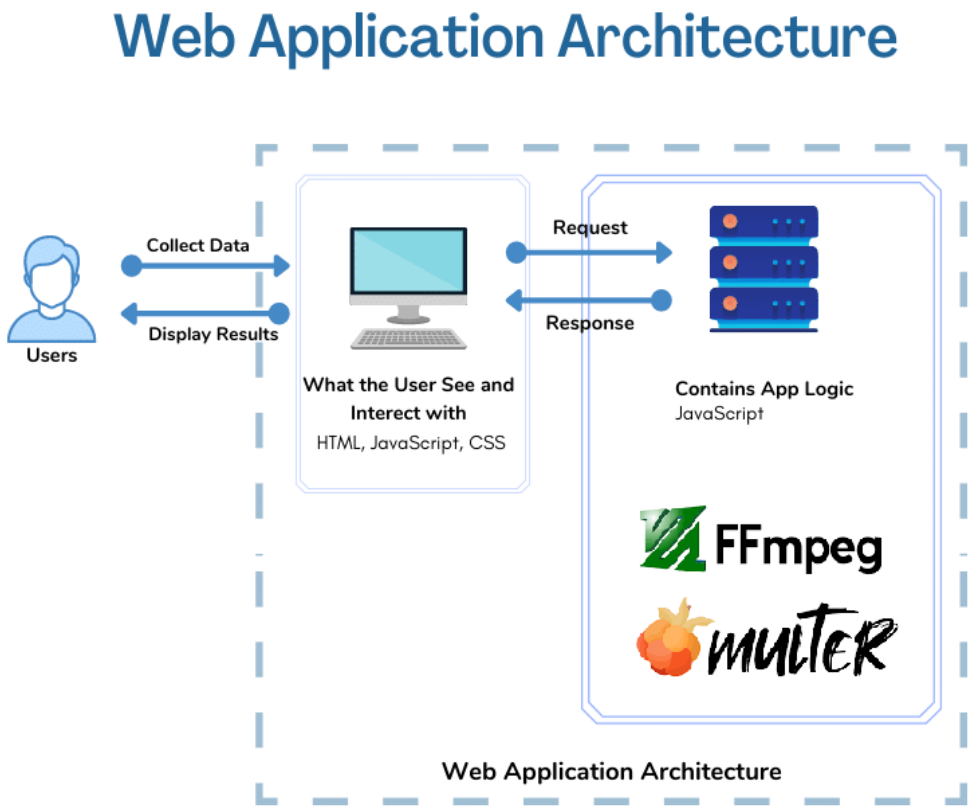


Figura 4.10 Arquitectura entre backend y frontend

Con el flujo de la aplicación se describe la secuencia de pasos que ocurren desde que un usuario interactúa con el frontend hasta que se procesan los datos en el backend y se devuelve la respuesta hacia el frontend. A continuación, se enumerarán por orden las etapas clave en el flujo de la aplicación.

1. El usuario selecciona un archivo de video en el frontend.
2. El archivo se envía al backend utilizando una solicitud HTTP multipart/form-data.
3. El backend recibe el archivo y lo almacena en una carpeta específica utilizando Multer.
4. El backend genera una URL para acceder al archivo almacenado y la devuelve al frontend.
5. El frontend muestra un reproductor de video utilizando VideoJS y carga el vídeo utiliznado la URL proporcionada y previamente generada por el backend.
6. El usuario establece el tiempo de inicio y fin del video que desea recortar.
7. El frontend envía una solicitud para generar el video clip.
8. El backend utiliza FFMPEG para recortar el vídeo según los tiempos proporcionados.
9. El vídeo recortado se guarda en otra carpeta específica.

10. Si se habilitan las notificaciones, el backend envía un correo electrónico al usuario para informarle que el vídeo se ha recortado correctamente.
11. El frontend descarga el vídeo recortado utilizando la URL proporcionado por el backend.

4.4 Diseño del diagrama de flujo de usuario

El diagrama de flujo de usuario es una representación visual del recorrido que un usuario puede seguir al utilizar la aplicación. Este diagrama muestras las posibles acciones disponibles, así como las decisiones que el usuario puede tomar durante su interacción con el aplicativo. El objetivo de este diagrama es proporcionar una visión clara y estructurada de la experiencia de usuario, identificando los diferentes caminos y las opciones que tiene disponibles.

El diseño del diagrama de flujo de usuario se ha realizado teniendo en cuenta los componentes y funcionalidades clave de la aplicación. En cada etapa del recorrido del usuario, se han contemplado las decisiones que puede tomar, como cargar un video, estableces el tiempo de inicio y finalización del clip, habilitar notificaciones, entre otras opciones.

A lo largo del apartado se detallarán los pasos y decisiones tomadas en el desarrollo de la aplicación. El diseño del diagrama de flujo ha sido una pieza clave en el proceso de diseño de la aplicación, ya que se ha utilizado como una guía que ha orientado la implementación de las diferentes funcionalidades y la creación de una experiencia de usuario intuitiva y eficiente.

A continuación, se procederá a introducir los elementos que han sido empleados para la elaboración del diseño. Estos serán útiles para explicar detalladamente la estructura y funcionamiento de los diferentes procesos.

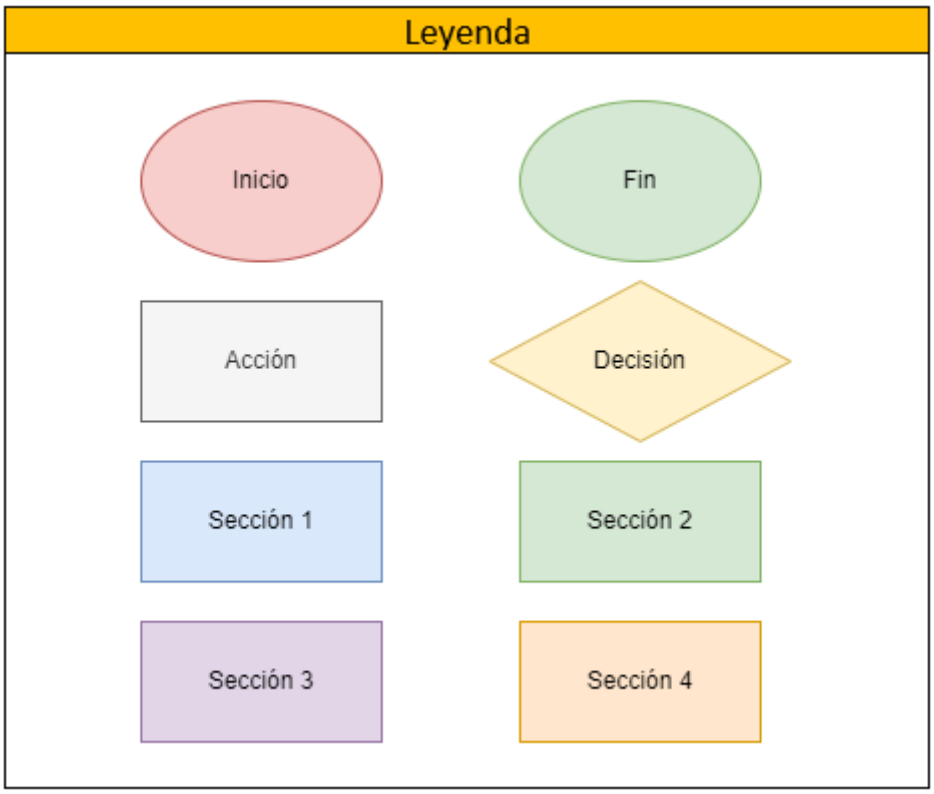


Figura 4.11 Leyenda del diagrama de flujo

A continuación, se muestra la leyenda del diagrama que resume la simbología utilizada. En el diagrama, los puntos de decisión son los elementos más comunes, ya que determinan diferentes caminos y acciones a seguir.

En la siguiente imagen se presenta el diagrama completo para obtener una visión general de todo el flujo. Si se requiere una vista detallada, se proporciona una referencia en la sección de Bibliografía con un enlace al archivo original.

El diagrama se divide en varias secciones, cada una identificada por un color diferente, lo cual facilita la explicación de los diferentes procesos dentro del flujo general de la aplicación.

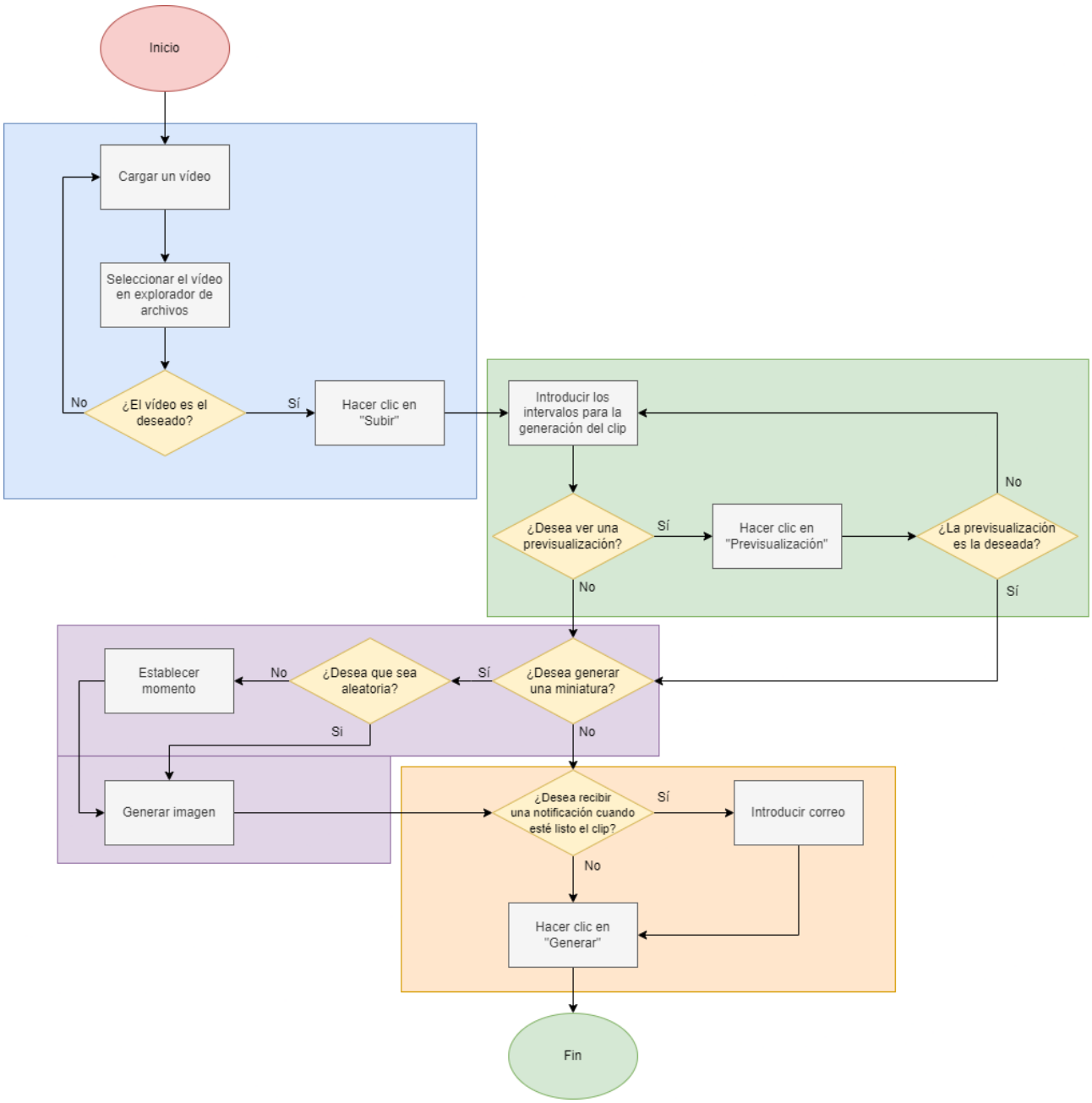


Figura 4.12 Diagrama de flujo de usuario completo

Sección 1

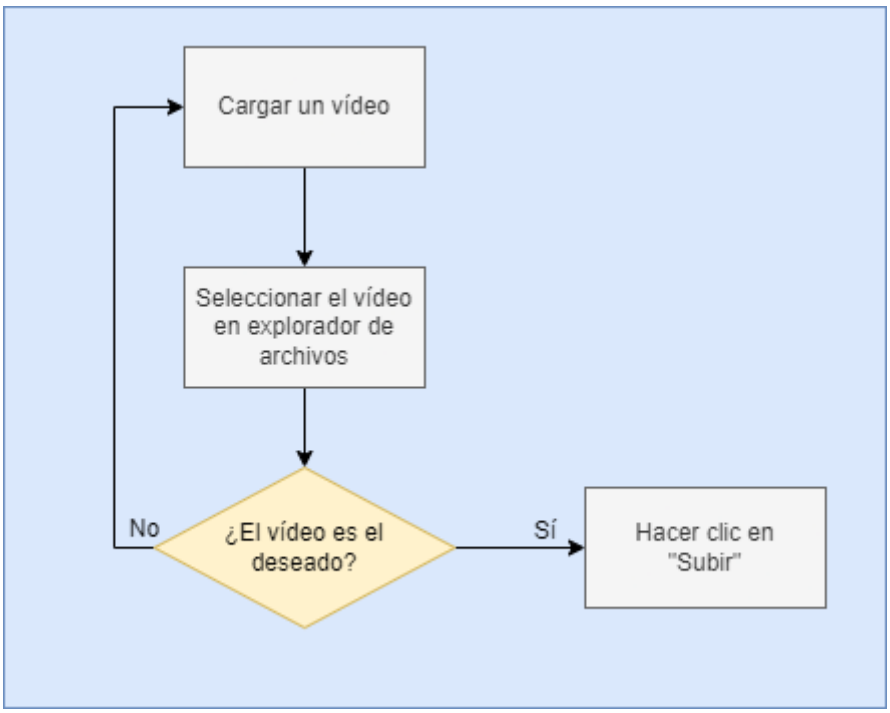


Figura 4.13 Sección 1 del diagrama de flujo

La sección 1 se identifica con el proceso de carga de un video a la aplicación web. Para realizar este proceso, simplemente se hará clic en el botón de “Subir un vídeo”. Esto abrirá el explorador de archivos local del sistema del usuario. Desde allí, se deberá buscar el video que se desee. Si por alguna razón se selecciona un archivo y que este no sea el adecuado, se puede volver a seleccionar el archivo correcto. Una vez se tenga el video deseado seleccionado, finalmente se hará clic en “Subir” para que el archivo sea enviado al servidor para su posterior procesamiento.

Sección 2

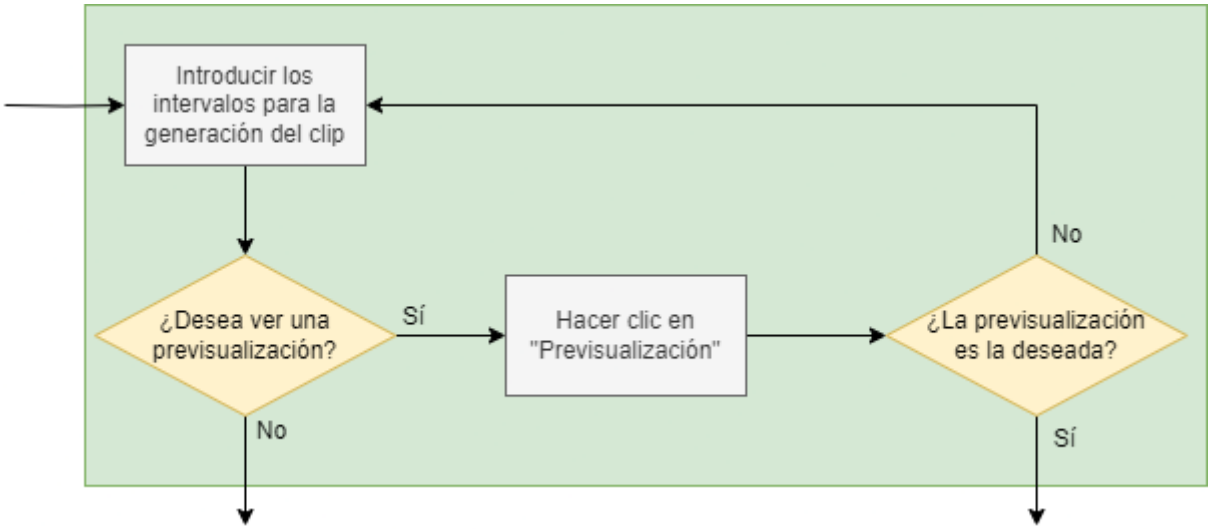


Figura 4.14 Sección 2 del diagrama de flujo

La sección 2 se enfoca en la introducción de datos para definir el instante inicial y el instante final que se utilizarán para generar un clip de video. Durante este proceso, el usuario tiene la opción de realizar una previsualización utilizando el reproductor de VideoJS para ver cómo quedará el clip resultante basado en los datos ingresados. Si el usuario desea realizar ajustes en el intervalo de tiempo o simplemente asegurarse del resultado antes de continuar, puede hacer clic en el botón "Previsualización". Esto permitirá al usuario corregir los datos o confirmar el resultado antes de avanzar a la siguiente sección del proceso. Del mismo modo, podrá también saltarse el proceso de previsualizado en caso de querer omitirlo.

Sección 3

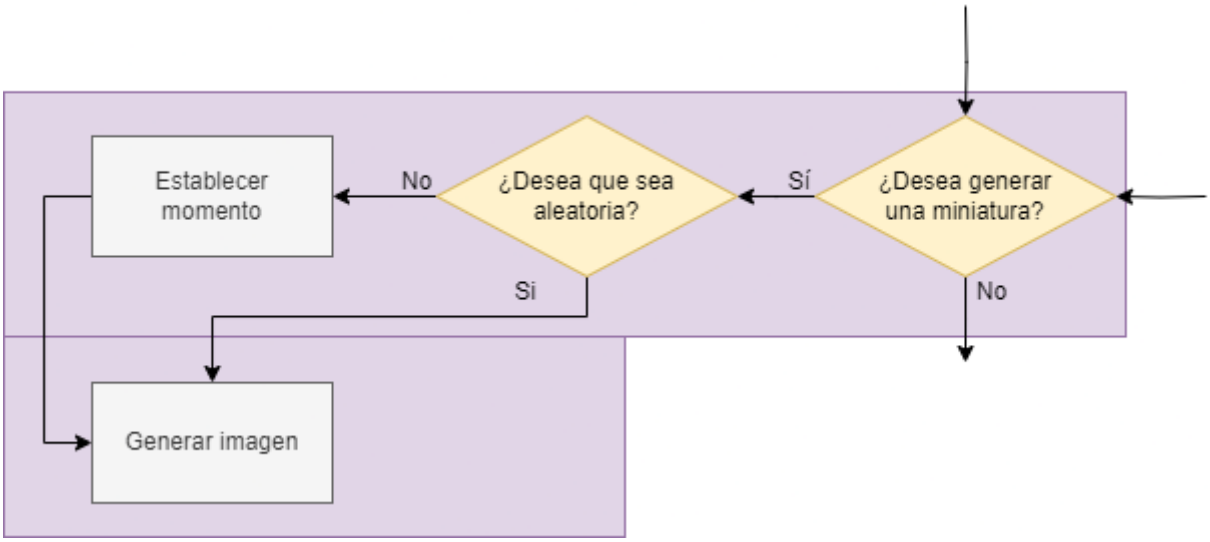


Figura 4.15 Sección 3 del diagrama de flujo

En la sección 3, el usuario tiene la opción de decidir si desea generar una imagen a forma de miniatura del video mientras se genera el clip con el intervalo de tiempo especificado. Si el usuario no desea utilizar esta opción, puede pasar a la siguiente sección. Sin embargo, si elige generar la miniatura, tiene dos opciones disponibles.

La primera opción es generar la imagen en un momento aleatorio dentro del intervalo de tiempo del clip. Esto significa que se extraerá una imagen representativa de forma automática en un momento seleccionado al azar dentro del intervalo.

La segunda opción es establecer un momento exacto para extraer el fotograma que se utilizará como miniatura. Con esta opción, el usuario puede especificar el instante preciso dentro del intervalo de tiempo del clip en el que se generará la imagen.

Estas opciones brindan flexibilidad al usuario para personalizar la generación de la imagen en miniatura según sus preferencias.

Sección 4

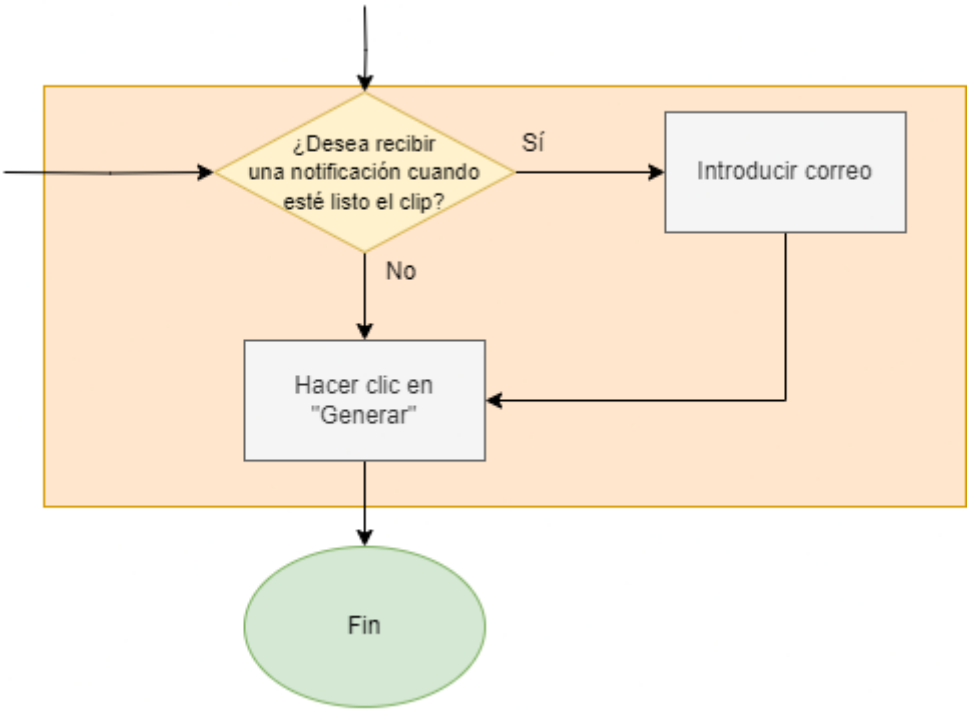


Figura 4.16 Sección 4 del diagrama de flujo

La sección 4 ofrece la opción de enviar una notificación al usuario en caso de que haya activado la casilla correspondiente. Si el usuario decide no activar la notificación, puede proceder haciendo clic en el botón "Generar" para finalizar todo el proceso y obtener el clip de video deseado.

Sin embargo, si el usuario desea recibir una notificación, debe activar el campo correspondiente e introducir una dirección de correo electrónico válida. De esta manera, se le enviará una notificación al usuario cuando el video esté listo para ser descargado.

Aclaración sobre el diagrama

A pesar de que el diagrama no incluye explícitamente este proceso, es importante mencionar que después de enviar el video al backend en cada paso, se encuentra disponible un botón denominado "Limpiar". Este botón permite al usuario reiniciar el proceso y eliminar cualquier información residual en los campos de entrada y envío hacia el servidor.

Al hacer clic en el botón "Limpiar", se restablecerán todos los valores de los estados de React utilizados en la aplicación. Esto garantiza que el usuario pueda comenzar el proceso desde el inicio con campos vacíos y sin ninguna información residual anterior.

La inclusión de este botón "Limpiar" es una medida adicional para garantizar la claridad y la posibilidad de reiniciar el proceso de manera rápida y sencilla, sin ninguna interferencia de datos anteriores.

4.5 Implementación del aplicativo

A continuación, se explicarán los pasos que se han seguido para crear el aplicativo web utilizando las tecnologías y herramientas anteriormente descritas. Además de las características de la aplicación implementada. Estas tareas se explicarán en el orden cronológico en las cuales han sido implementadas.

El código de la implementación se podrá consultar en el Anexo A.

4.5.1 Carga del archivo original.

El primer objetivo marcado es el proceso de carga del archivo original, se ha implementado una interfaz de usuario que permite seleccionar un archivo de video desde su dispositivo local. Esto se consigue mediante un campo de entrada de tipo “file” en el formulario. Cuando el usuario selecciona un archivo, este se captura a través de un controlador de eventos y se almacena en el estado del componente de React.

Posteriormente, se utiliza la biblioteca Axios para enviar el archivo hacia el backend a través de una solicitud POST. El archivo se adjunta a un objeto FormData [17] que encapsula los datos y se envía al endpoint “/uploads” del backend para su procesamiento.

Después de enviar la solicitud POST al endpoint, el archivo de video se procesará en el servidor utilizando la biblioteca Multer para manejar la carga del archivo.

Este se configura con una instancia que especifica la ruta destino donde se guardará el archivo y su correspondiente nombre. En este caso, la carpeta se establece en la ruta “/public/uploads” y el nombre del archivo se genera de manera única, utilizando la marca de tiempo y un número aleatorio.

Una vez se recibe el archivo correctamente, se extraen los datos necesarios y se construye la URL del video. En este caso la URL estará formada por el nombre de dominio, la ruta donde se guarda y finalmente el nombre del archivo. También se obtiene el tipo de archivo a partir de la propiedad “mimetype” [19] del archivo recibido.

Finalmente, se devuelve una respuesta de éxito con un código de estado 200 y un objeto JSON que contiene el nombre de archivo que se ha generado, la URL y el tipo de archivo.

Figura 4.17 Carga del video

4.5.2 Reproducción del video original.

Tras haber cargado el archivo original, la aplicación utiliza la biblioteca VideoJS para mostrar y reproducir el video en el aplicativo web. Se crea un reproductor de video personalizado que se vincula al elemento de video correspondiente en la interfaz de usuario.

El reproductor de VideoJS se configura con varias opciones, como la visualización de controles y la reproducción automática. Esto permite al usuario interactuar con el video y controlar la reproducción, pausa, volumen, entre otras funciones.

Además, se aprovecha el evento "loadedmetadata" del reproductor de video para obtener la duración del video cargado. Esta información es importante para el siguiente objetivo establecido en la aplicación web, ya que se utiliza para establecer los tiempos de inicio y fin al recortar el video.

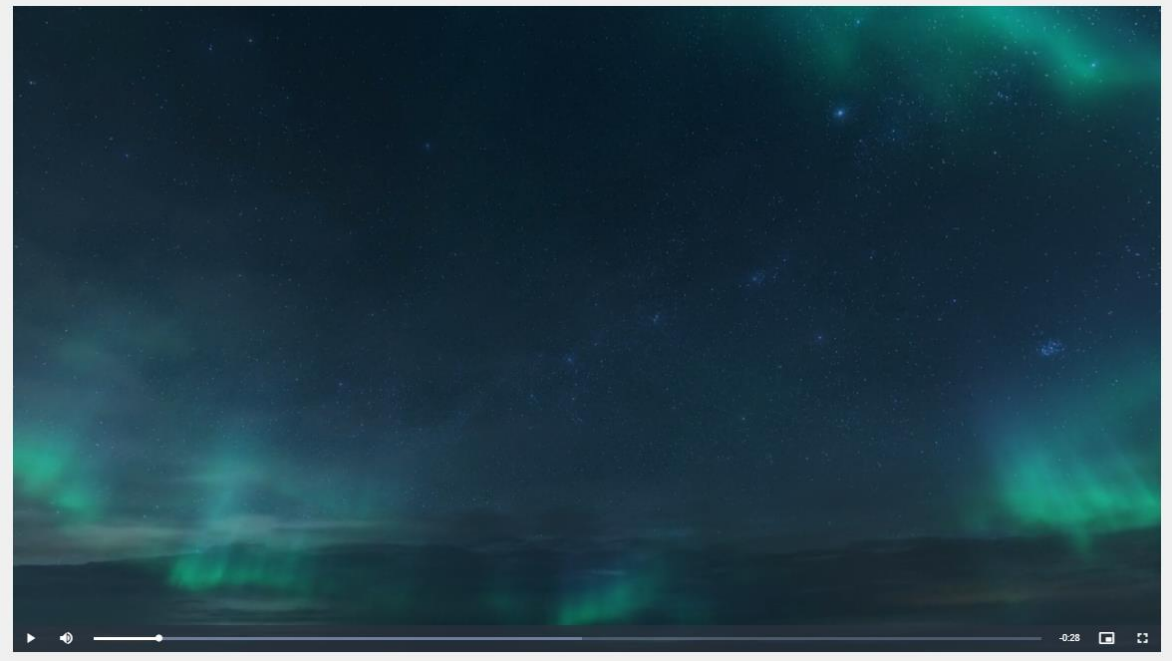


Figura 4.18 Reproductor VideoJS

4.5.3 Implementación de los campos esenciales para la entrada de datos

La interfaz de usuario de la aplicación incluye campos de entrada designados donde los usuarios pueden especificar el tiempo de inicio y finalización para el recorte del video. Estos campos están vinculados a estados en el componente de React, lo que significa que su contenido se actualiza automáticamente en tiempo real a medida que los usuarios ingresan los valores.

Cuando los usuarios ingresan un valor en el campo de tiempo de inicio, se actualiza el estado correspondiente en el componente de React. De manera similar, cuando ingresan un valor en el campo de tiempo de finalización, se actualiza el estado correspondiente en el componente.

Esta vinculación de datos bidireccional entre los campos de entrada y los estados en React garantiza que los valores ingresados por los usuarios estén siempre disponibles y actualizados en el componente. Además, permite que la lógica de recorte del video utilice estos valores para determinar los puntos de inicio y finalización adecuados durante el proceso de recorte.

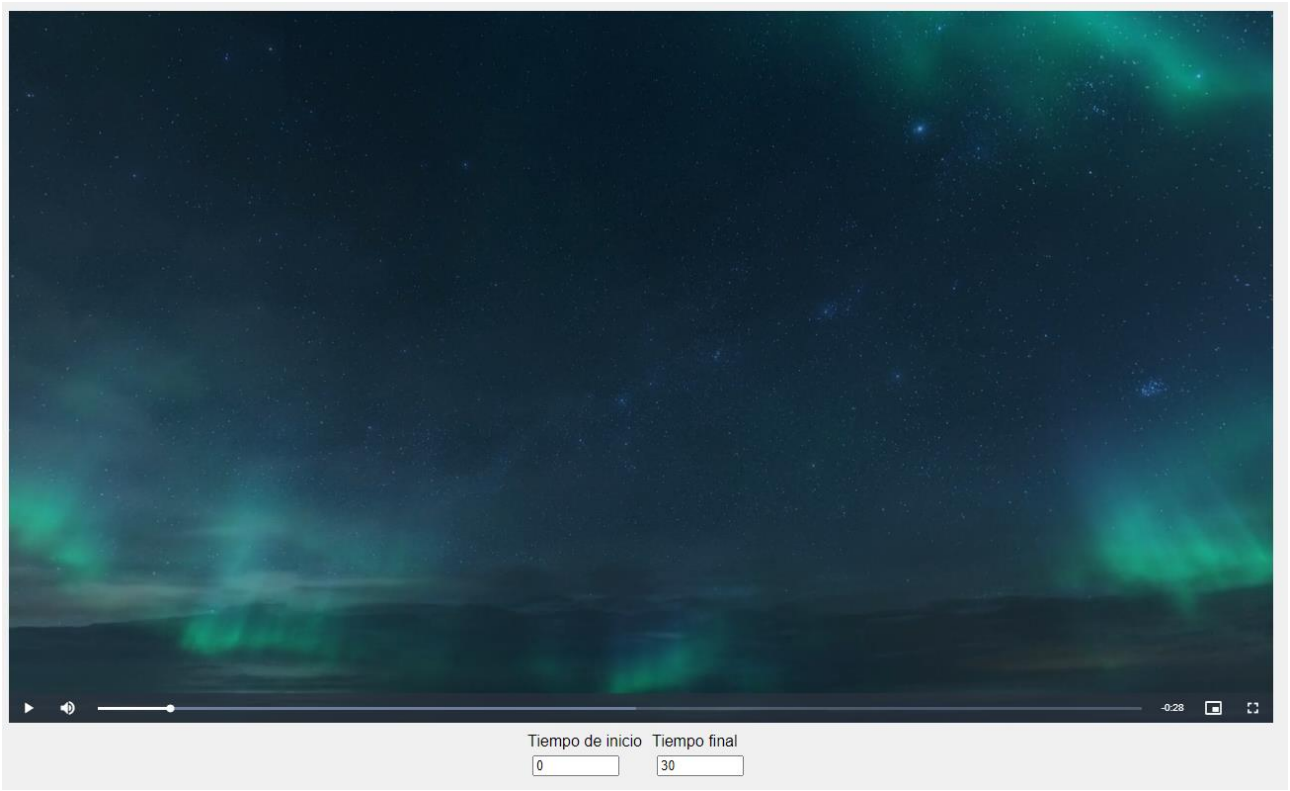


Figura 4.19 Reproductor y campos esenciales para la entrado de datos

4.5.4 Envío de la información para generar el clip.

Una vez que los usuarios han ingresado los valores requeridos, como el archivo de video, el tiempo de inicio, el tiempo de fin, el nombre de video y la opción de notificación, que se explicará en posteriores apartados, pueden enviar la información al backend para generar el clip de video recortado. Al hacer clic en el botón generar, se invoca una función que utiliza la biblioteca Axios para enviar una solicitud POST al endpoint "/trim" del backend. Esta solicitud incluye los parámetros necesarios, como la ruta del video original, los tiempos de inicio y fin, el nombre del video recortado y la opción de notificación.

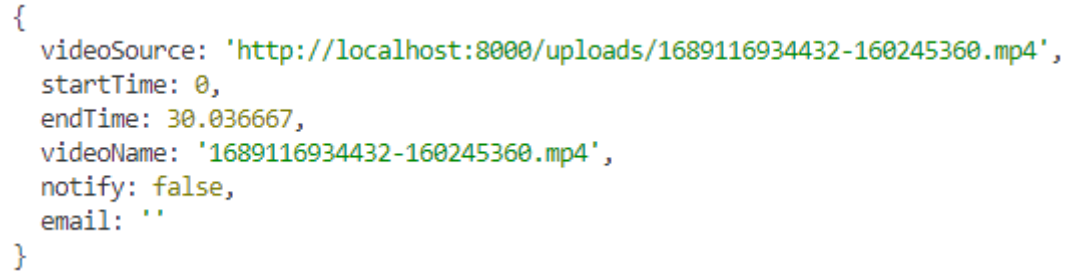


Figura 4.20 Mensaje por consola tras haber enviado la información al backend en local

En el backend, se capturan los parámetros de la solicitud y se realiza el proceso de recorte del video utilizando la biblioteca "fluent-FFMPEG". Se establece la ruta de destino para el video recortado y se configuran las opciones de recorte, como el tiempo de inicio, la duración y el formato de salida.

Finalmente, se devuelve una respuesta al frontend con la descarga del clip generado.

4.5.5 Descarga del archivo del clip generado.

Una vez que se ha completado el proceso de recorte del video en el backend y se ha enviado la respuesta al frontend, se procede a la descarga del archivo del clip generado.

En el frontend, se utiliza la biblioteca Axios para enviar la solicitud POST al endpoint "/trim" del backend. La respuesta del servidor contiene el archivo del clip generado en forma de datos no binarios (blob).

Utilizado la funcionalidad del objeto Blob en JavaScript, se crea una URL del blob que representa el archivo del clip generado y se puede utilizar para descargarlo directamente en el navegador del usuario.

A continuación, se crea dinámicamente un elemento de enlace y se establece su atributo "href" con la URL del blob. También se configura el atributo "download" del enlace para especificar el nombre de archivo que se utilizará al descargar el clip, este vendrá definido por un prefijo "trimmed" para hacerle saber al usuario que este archivo se corresponde con la versión del archivo del clip generado, por si el usuario descarga el archivo en el directorio donde almacenaba el archivo original.

Luego se agrega el elemento de enlace al documento web y se dispara automáticamente el evento de clic de enlace programáticamente. Esto activa la descarga automática del archivo del clip generado en el navegador del usuario sin requerir ninguna acción adicional por parte del usuario.

4.5.6 Notificación tras obtener el archivo generado.

Como se mencionó en apartados anteriores, si los usuarios han habilitado las notificaciones por correo electrónico al enviar la solicitud de generación de clip de video, se envía una notificación utilizando el servicio de Nodemailer en el backend.

En el backend, se utiliza la biblioteca Nodemailer para configurar un transportador de correo electrónico con los detalles de la cuenta de correo remitente, como la dirección de correo electrónico y la contraseña. Estos detalles se proporcionan al crear la instancia del transportador de Nodemailer.

Una vez configurado el transportador de correo electrónico, se puede enviar un correo electrónico al destinatario especificado en la solicitud de generación de clip de video.

Al enviar el correo electrónico, el servicio de Nodemailer se encarga de enviar el mensaje utilizando los protocolos de correo electrónico necesarios, como SMTP. Si el envío del correo electrónico tiene éxito, los usuarios recibirán una notificación instantánea en su bandeja de entrada, informándoles que su clip de video ha sido generado y está listo para su descarga a través del navegador.

Esta funcionalidad de notificación por correo electrónico permite a los usuarios recibir una confirmación rápida y conveniente una vez que su clip de video ha sido generado y está disponible para su uso. Además, brinda una forma de mantener a los usuarios actualizados sobre el progreso del procesamiento de sus solicitudes y proporcionarles una experiencia más completa en el aplicativo web.

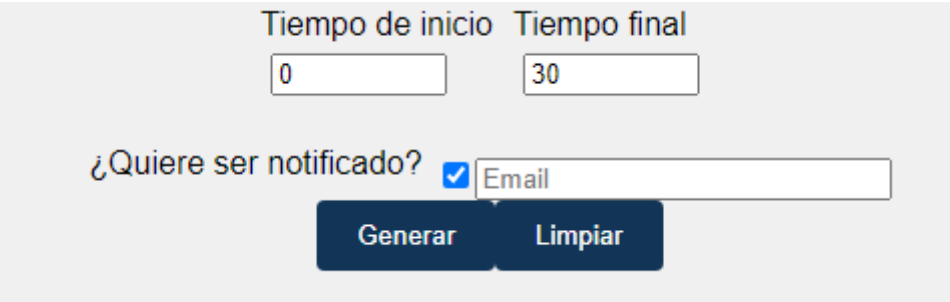


Figura 4.21 Formulario para ser notificado

4.5.7 Aportaciones extras.

Además de las funcionalidades básicas mencionadas anteriormente, se han implementado algunas características adicionales que mejoran la experiencia del usuario en el aplicativo web.

Funcionalidad de limpieza de archivos no necesarios

Para evitar el colapso del backend y mantener un espacio de almacenamiento ordenado, se ha implementado un servicio que se encarga de limpiar los directorios de archivos temporales generados. A medianoche, se ejecuta automáticamente un proceso de limpieza que elimina los archivos almacenados en las carpetas "public/uploads" (archivos originales), "public/trims" (video clips generados) y "public/thumbs" (miniaturas generadas).

Botón "Limpiar" para reiniciar los estados

Si un usuario carga un video y desea cambiar el archivo original, se producían inconsistencias en los estados de React, lo que afectaba la precisión de los datos. Para solucionar esto, se ha implementado el botón "Limpiar" que restablece y limpia todos los estados del aplicativo web. Al hacer clic en este botón, los estados vuelven a su estado inicial, permitiendo al usuario cargar un nuevo archivo sin problemas.

Funcionalidad de previsualización del video

Junto a los campos de introducción de datos de inicio y fin del clip, se ha agregado un botón de previsualización. Al hacer clic en este botón, se reproduce un fragmento de video correspondiente al intervalo de tiempo especificado por el usuario. Esta previsualización brinda al usuario una vista previa del resultado del recorte antes de proceder con la generación del clip final.

Generación de miniaturas para redes sociales

Considerando la popularidad de las redes sociales que permiten subir videos cortos, se ha habilitado la opción de generar miniaturas. Estas miniaturas son imágenes representativas del video recortado y pueden ser utilizadas como portadas o vistas previas del clip al ser compartido en plataformas como TikTok [20], Instagram Reels [21] o YouTube Shorts [22]. Esto brinda al usuario la posibilidad de personalizar y destacar visualmente su contenido en las redes sociales.

En conjunto, estas características adicionales enriquecen la funcionalidad del aplicativo web, mejoran la experiencia del usuario y brindan herramientas prácticas para gestionar y compartir los clips de video generados.

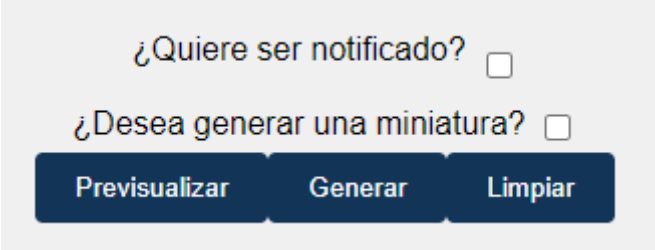


Figura 4.22 Aportaciones extras

4.6 Análisis de los resultados (Test de usuario)

En el proceso de desarrollo de la aplicación, se han realizado pruebas con el objetivo de evaluar la usabilidad y obtener información sobre la experiencia del usuario. Para lograr esto, se ha reclutado a un grupo heterogéneo de 15 personas con diferentes edades, perfiles y formación, quienes participaron activamente en las pruebas y compartieron sus comentarios.

El propósito de este apartado es presentar uno a uno los resultados obtenidos a través de estas pruebas, así como analizar los datos recopilados a partir de los cuestionarios completados por los participantes. Esos cuestionarios nos permitieron obtener una visión detallada de la percepción de los usuarios sobre la aplicación y su funcionamiento.

A lo largo de este apartado, se evaluarán diferentes aspectos como la facilidad de uso, satisfacción general del usuario o dificultad en los principales procesos. Es importante destacar que la diversidad de perfiles de los participantes nos brinda una perspectiva más amplia y representativa de las distintas necesidades y expectativas de los potenciales usuarios de la aplicación. Esto permitirá tomar mejores decisiones y realizar mejoras significativas en la aplicación para ofrecer una mejor experiencia.

En el cuestionario existen diferentes tipos de preguntas, desde marcar una correspondiente casilla, a elegir del 1 al 5 el valor de dificultad, siendo 1 el valor más difícil y siendo 5 el valor más fácil.

La primera pregunta del cuestionario consistía en que el usuario se identificara con su rango de edad.

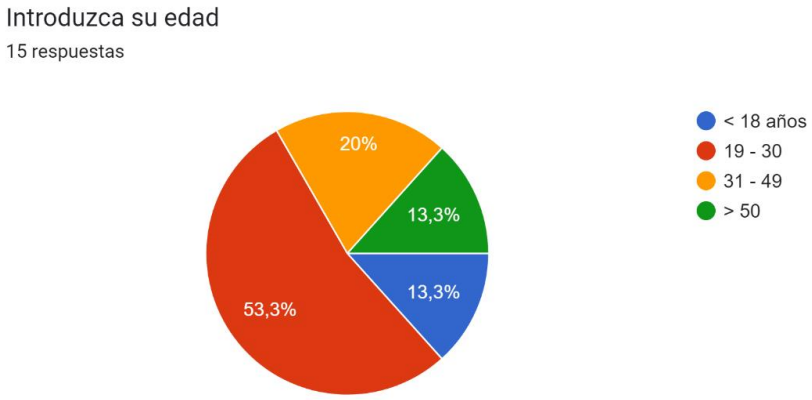


Figura 4.23 Edad de los participantes del cuestionario

Como se puede observar, existe representación para todos los rangos. Se ha de comentar que la mayoría de personas se encuentran en la horquilla de los 19 a los 30 años.

La segunda pregunta consistía en evaluar el grado de dificultad que poseía el proceso de carga de un vídeo para que fuera reproducido por la página web.

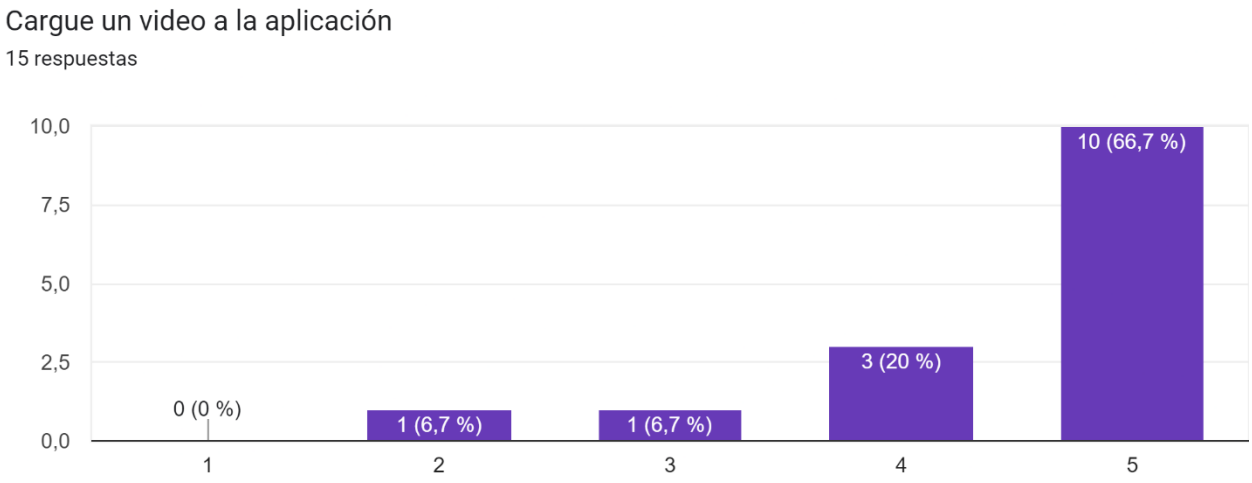


Figura 4.24 Grado de dificultad carga de video

La respuesta en grandes rasgos fue que tenía un grado de dificultad bastante fácil, es verdad que hay una evaluación con un número 2, pero al existir personas con mayores dificultades ya sea por la edad o por el nivel de conocimientos no es altamente preocupante.

Tras evaluar el grado de dificultad, se preguntaba por la dificultad del proceso de la introducción de los datos y de la opción de ser notificado a través del correo electrónico.

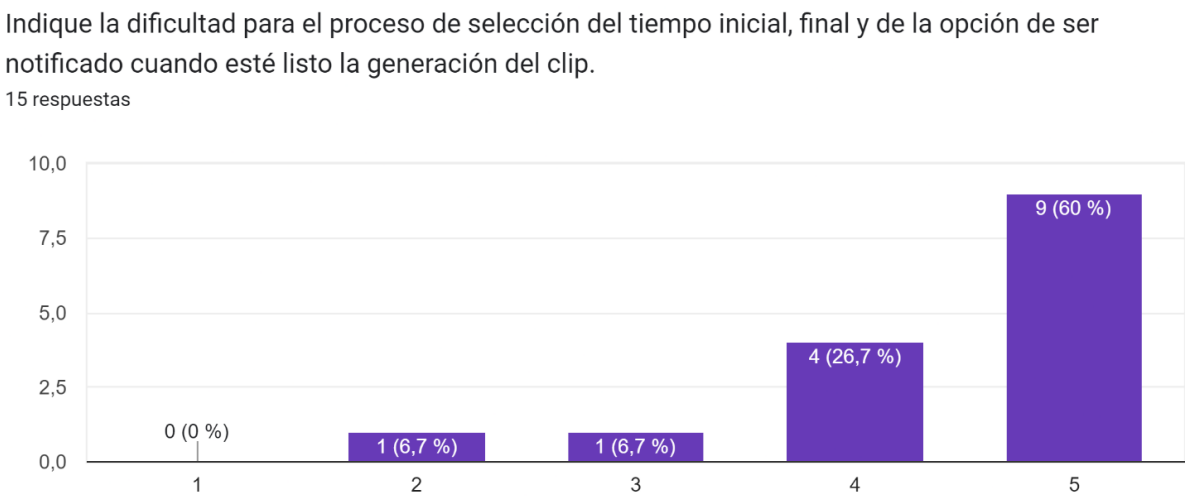


Figura 4.25 Grado de dificultad introducción de los datos

Los valores son muy parecidos al anterior.

También se ha preguntado por la opción de previsualización, con el fin de conocer la opinión respecto dicha funcionalidad.

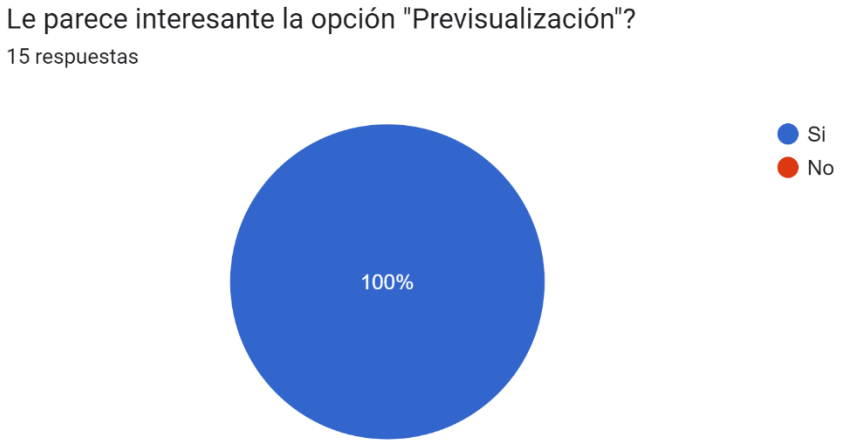


Figura 4.26 Interés por la función "Previsualización"

Con todas las respuestas, se puede afirmar que ha sido un acierto la implementación de la funcionalidad que permite previsualizar el clip antes de generarlo.

Después de la introducción los datos necesarios, se ha querido comprobar si el usuario se ha sentido cómodo con el proceso de generar el clip y su correspondiente descarga.

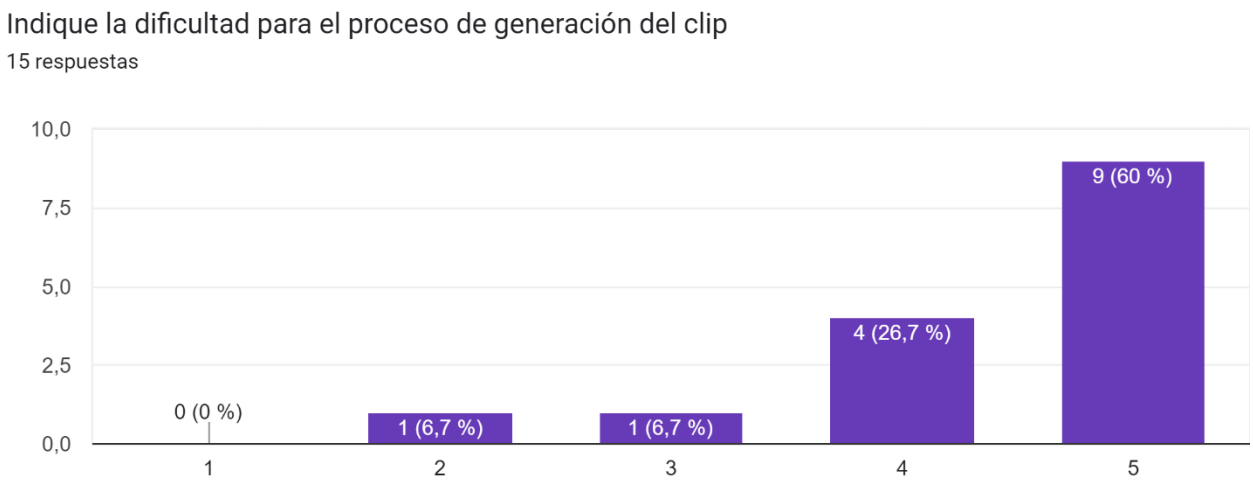


Figura 4.27 Grado de dificultad generación del clip

Al igual que las otras preguntas, existe una gran mayoría que piensa u opina que es bastante asequible el proceso de generación y descarga del clip generado.

Una vez cumplido con las pequeñas pruebas que se han evaluado, se ha preguntado si se el usuario se siente satisfecho en cuanto al producto recibido, es decir, si ha notado alguna diferencia entre vídeos, ya sea pérdida de calidad u otra sensación.

Figura 4.28 Satisfacción con el resultado

Todos los usuarios han quedado satisfechos con el resultado tras finalizar el proceso de generación de clips.

Para finalizar, se ha preguntado si el usuario ha experimentado algún problema y si usaría la aplicación cuando fuera necesario tras haber realizado este test.

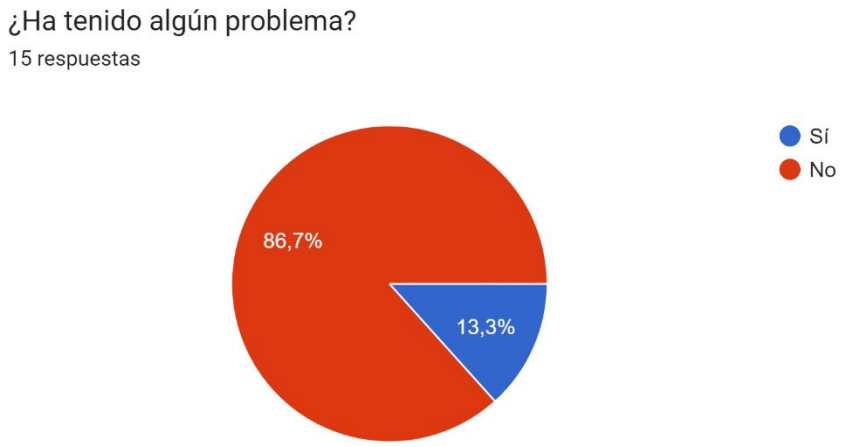


Figura 4.29 Problemas con la aplicación

La inmensa mayoría no ha sufrido ningún tipo de problemas, sin embargo 2 de las 15 (13,3%) personas han experimentado algún tipo de problema, siendo estos:

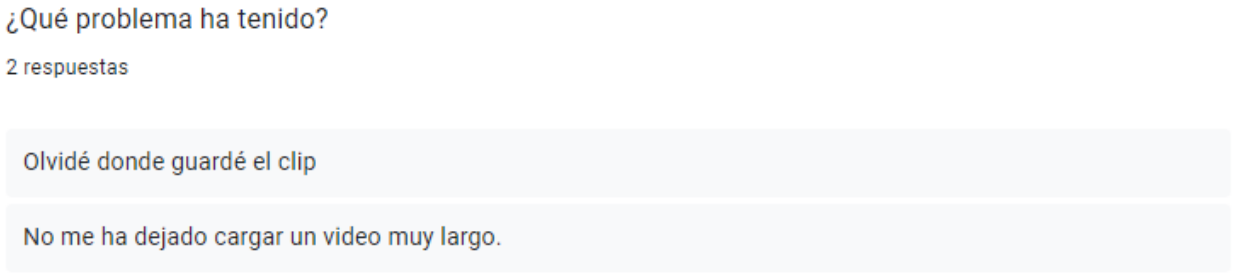


Figura 4.30 Problemas detectados

Los problemas detectados han sido los siguientes:

- Un usuario tras haber cumplimentado el proceso sin problema alguno, tras descargar el fichero no encontraba el fichero generado. Tras ver dicho problema, se ha establecido contacto con esta persona para tratar de ver si no se había generado correctamente o había sido culpa del usuario, siendo esta última la respuesta. Ya que tras realizar la búsqueda por el prefijo “trimmed” que se le asigna a cada clip que se descarga desde el servidor, se ha terminado encontrando.

- El segundo problema si ha sido culpa del aplicativo, concretamente por el Multer, este estaba configurado para limitar el peso del archivo original, siendo este 100Mb, si se pasaba de dicho peso, no se permitía el proceso del vídeo.

```
const upload = multer({
    storage: storage,
    limits: { fileSize: 50 * 1024 * 1024 }
});
```

Figura 4.31 Límite de 50Mb establecido en la configuración de Multer

En cuanto si el usuario repetiría la experiencia, el resultado ha sido el siguiente.

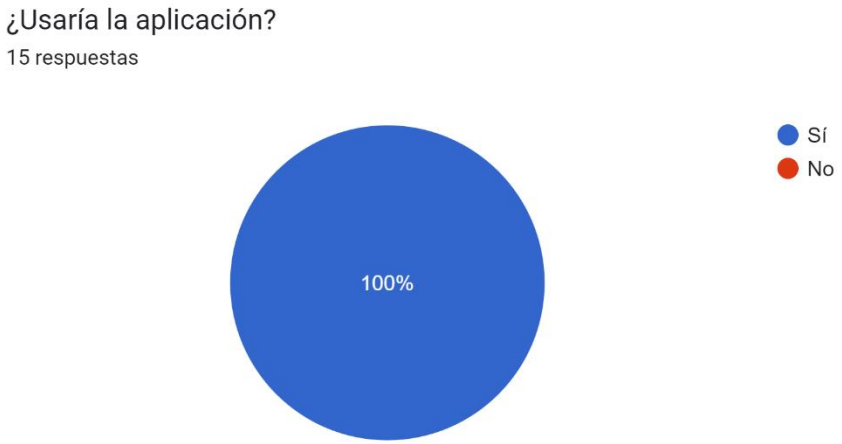


Figura 4.32 Porcentaje de personas que usarían la aplicación

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

El objetivo principal de este Trabajo de Fin de Grado fue diseñar y automatizar una aplicación web para generar clips de video bajo demanda, obteniendo resultados precisos y de alta calidad de manera sencilla y eficiente.

Tras desarrollar el sistema y realizar pruebas con usuarios, se ha confirmado que la aplicación funciona correctamente y cumple con las expectativas, siendo eficaz en el procesamiento de videos. Esto representa la creación de una herramienta útil basada en tecnologías de código abierto con un gran potencial.

En general, el proyecto ha demostrado la capacidad de utilizar tecnologías modernas y herramientas de desarrollo para construir una aplicación web completa. Se abordaron aspectos fundamentales como el almacenamiento de archivos, el procesamiento de videos y la comunicación con los usuarios a través de notificaciones por correo electrónico. La aplicación tiene un potencial de expansión según las necesidades que se detecten en los usuarios.

Sin embargo, se ha identificado que la aplicación puede quedarse rezagada en comparación con otros generadores de videos que poseen características más completas. A pesar de esto, no es motivo de desilusión o desánimo, sino el primer paso para construir una aplicación capaz de realizar todo tipo de procesamiento de videos, aprovechando oportunidades en diversos segmentos de mercado.

Este proyecto también demuestra que, a pesar de ser estudiantes, somos capaces de crear y desarrollar una aplicación completa que puede competir con otras aplicaciones web con más experiencia en el mercado. Es un ejemplo de nuestro potencial para crear soluciones innovadoras y de calidad.

5.2 Líneas futuras

En cuanto a las líneas futuras de desarrollo del proyecto, se pueden identificar mejoras en diferentes áreas clave.

Mejoras en el procesamiento de video

- Implementar soporte para diferentes códecs de video, lo que permitiría a los usuarios elegir el formato de salida más adecuado para sus necesidades.

- Agregar funciones de edición de video, como ajustes de brillo, contraste, saturación, o la capacidad de aplicar filtros y efectos especiales al clip generado. Esto proporcionaría a los usuarios más opciones creativas y de personalización.

Mejoras en la interfaz de usuario

- Implementar una función de selección de intervalo de clip más intuitiva y visualmente atractiva. Esto podría lograrse mediante el uso de una barra de reproducción con imágenes en miniatura que permita a los usuarios arrastrar los indicadores para establecer el inicio y el final del clip deseado. Esto facilitaría la navegación y la selección precisa de los segmentos de video a recortar.

Mejoras en la gestión de usuarios:

- Agregar un sistema de autenticación y gestión de usuarios que permita a los usuarios registrarse y mantener un historial de los clips generados. Esto les brindaría la capacidad de acceder a sus clips anteriores y descargarlos en cualquier momento, incluso después de cerrar el navegador. Al recibir una notificación de que el clip está listo, podrían iniciar sesión en su cuenta y descargar el resultado sin perderlo.

Estas mejoras contribuirían a enriquecer la funcionalidad y la experiencia del usuario, así como a hacer que la aplicación sea más versátil y personalizable. Además, abrirían la puerta a nuevas oportunidades de mercado y demostrarían una evolución constante del proyecto para adaptarse a las necesidades cambiantes de los usuarios.

Capítulo 6
Summary and Conclusions

6.1 Summary

The main objective of this project is to develop an on-demand clip generation application that meets the users' needs, providing them with a versatile and easy-to-use tool for creating audiovisual content. In a world where content creation has become increasingly important, this application acts as a virtual studio, allowing users to generate clips comfortably and naturally.

The idea for this project arises as a direct response to the growing demand for accessible and practical video editing tools. With technological advancements in the mobile and image processing fields, the aim is to implement an application that allows users to capture special moments and turn them into unique pieces of audiovisual content.

The application will feature precise trimming tools that enable users to select and extract specific segments from their videos. This functionality will facilitate the creation of short clips or summaries of longer events, giving users the ability to condense information and highlight the most significant moments.

To achieve this, the project will rely on cutting-edge technologies in image processing and video editing algorithms. These technologies will allow users to make precise trims and obtain high-quality results. Additionally, the application will strive to be intuitive so that both those with video editing experience and those who are new to this field can use it without difficulties.

The application will be particularly useful for those who want to create content for social media platforms, blogs, or professional presentations. By providing an intuitive video editing experience, the application will enable users to bring their creativity to life and achieve high-quality results in their audiovisual projects.

In summary, this project aims to develop an application that caters to users' needs in terms of on-demand clip generation. Leveraging the most advanced technologies in image processing and video editing, the application will offer precise trimming tools and an intuitive experience for users to create unique and high-quality audiovisual content.

6.2 Conclusions

The main objective of this Bachelor's Thesis was to design and automate a web application for on-demand video clip generation, achieving accurate and high-quality results in a simple and efficient manner.

After developing the system and conducting user testing, it has been confirmed that the application functions properly and meets expectations, being effective in video processing. This represents the creation of a useful tool based on open-source technologies with great potential.

Overall, the project has demonstrated the ability to use modern technologies and development tools to build a complete web application. Key aspects such as file storage, video processing, and communication with users through email notifications were addressed. The application has potential for expansion based on the needs identified by users.

However, it has been identified that the application may lag behind other video generators that offer more comprehensive features. Despite this, it is not a reason for disappointment or discouragement, but rather the first step towards building an application capable of performing all kinds of video processing, tapping into opportunities in various market segments.

This project also demonstrates that, despite being students, we are capable of creating and developing a comprehensive application that can compete with more experienced web applications in the market. It serves as an example of our potential to create innovative and high-quality solutions.

Future Lines

Regarding future development lines for the project, improvements can be identified in different key areas.

Video processing improvements

Implement support for different video codecs, allowing users to choose the most suitable output format for their needs.
Add video editing features such as brightness, contrast, saturation adjustments, or the ability to apply filters and special effects to the generated clip. This would provide users with more creative and customization options.

User interface improvements

Implement a more intuitive and visually appealing clip interval selection function. This could be achieved by using a thumbnail image-based playback bar that allows users to drag indicators to set the desired start and end of the clip. This would facilitate navigation and precise selection of video segments to be trimmed.

User management improvements

Add an authentication and user management system that allows users to register and maintain a history of generated clips. This would give them the ability to access their previous clips and download them at any time, even after closing the browser. Upon receiving a notification that the clip is ready, they could log in to their account and download the result without losing it.

These improvements would enrich the functionality and user experience, making the application more versatile and customizable. Furthermore, they would open doors to new market opportunities and demonstrate the project's continuous evolution to adapt to the changing needs of users.

Capítulo 7
Presupuesto

El presupuesto constará de 4 partes, siendo estas las siguientes:

Estudio previo y comparativa de métodos y sistemas

Para realizar dicho estudio se requerirá la investigación de diferentes métodos y sistemas para realizar los cortes de vídeo. Esto implica revisión de documentación pruebas de conceptos y comparativas de rendimiento.

Desarrollo del aplicativo web

Una vez se haya elegido la mejor solución, se procederá con el desarrollo del aplicativo web. Esto, a grandes rasgos, incluirá la implementación del reproductor HTML5 open-source, la interfaz de usuario para indicar los instantes de comienzo y final, y la integración con el sistema de corte de video seleccionado.

Implementación del sistema de notificación

Para permitir al usuario ser notificado cuando el corte de video esté disponible, se deberá implementar un sistema de notificación. Esto puede realizarse mediante el uso de tecnologías como notificaciones push [23] o correos electrónicos automatizados.

Pruebas y control de calidad

Es importante asignar tiempo para realizar pruebas exhaustivas y asegurarse de que la aplicación funcione correctamente en diferentes escenarios. Esto incluye pruebas de reproducción de video, pruebas de generación de clips, de descarga de archivos, entre otros.

Presupuesto			
Tarea	**Precio € / horas**	**Número horas**	**Precio por tarea**
Estudio previo y comparativa de métodos y sistemas.	30 €	40	1200 €
Desarrollo del aplicativo web.	35 €	80	2800 €
Implementación del sistema de notificación	30 €	25	750 €
Pruebas y control de calidad	32 €	30	960 €
TOTAL		**5710 €**	

Tabla 7.1 Presupuesto

Capítulo 8
Apéndice A

8.1 Apartado 8.1 Código del backend

```
const port = process.env.PORT || 8000;
// Iniciando servidor
app.listen(port, () => {
    console.log(`Server listening on port ${port}`);
});
```

```
const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        const uploadPath = path.resolve(__dirname, '../public/uploads');
        cb(null, uploadPath);
    },
    filename: (req, file, cb) => {
        extension = file.mimetype.split('/')[1];
        const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
        cb(null, uniqueSuffix + '.' + extension);
    }
});
```

```
const transporter = nodemailer.createTransport({
    service: 'hotmail',
    auth: {
        user: 'jlabtfg@outlook.es',
        pass: 'proyectoTFG'
    },
});
```

8.1.1 EndPoints

```
app.post('/uploads', upload.single('video'), (req, res) => {
    console.log('Received upload request');
    if (!req.file) {
        res.status(400).send('No se ha envido ningun archivo');
    } else {
        const filename = req.file.filename;
        //const url = `http://localhost:8000/uploads/${filename}`;
        const url = `https://backend-tfg.vercel.app/uploads/${filename}`;
        const type = req.file.mimetype;
        res.status(200).json({ filename, url, type });
        console.log(filename);
        console.log(url);
        console.log(type);
    }
});
```

```
app.post('/trim', (req, res) => {
    console.log(req.body);
    const { videoSource, startTime, endTime, videoName, notify, email } = req.body;

    if (!videoSource || startTime === undefined || startTime === null || !endTime || !videoName) { //Condicion especial por startTime
        res.status(400).send('Faltan parametros en el cuerpo de la solicitud');
        return;
      }

    // Verificar que los valores de inicio y fin sean validos
    const start = parseFloat(startTime);
    const end = parseFloat(endTime)
    if (end <= start) {
        res.status(400).send('End time must be grater than start time');
        return;
    }

    const trimmedVideoPath = path.join(__dirname, '../public/trims', `trim_${videoName}`);
    const outputPath = path.join(__dirname, `../public/trims/trim_${videoName}`);
```

```
    // FFMPEG
    ffmpeg()
        .input(videoSource)
        .setStartTime(start)
        .setDuration(end - start)
        .output(outputPath)
        .outputOptions(['-c:v libx264', '-c:a aac'])
        .on('end', () => {
            console.log('Video trimmed successfully');

            if (notify) {
                const { email } = req.body;
                const mailOptions = {
                    from: 'jlabtfg@outlook.es',
                    to: email,
                    subject: 'Video Trimmed Successfully',
                    text: 'Your video has been successfully trimmed.',
                };

                transporter.sendMail(mailOptions, (error, info) => {
                    if (error) {
                        console.error('Error sending email:', error);
                    } else {
                        console.log('Email sent: ', info.response);
                    }
                });
            }
            res.sendFile(trimmedVideoPath);
        })
        .on('error', (err) => {
            console.error(err);
            res.status(500).send('Error trimming video');
        })
        .run();
});
```

```
app.post('/generate-thumbnail', (req, res) => {
    console.log(req.body);
    const { videoSource, videoName, generateThumbnailChecked, thumbnailTime, endTime} = req.body;
    const name = videoName.slice(0, -4);

    if (!videoSource || !videoName || !endTime) {
        res.status(400).send('Missing parameters in request body');
        return;
    }

    let time;

    if (thumbnailTime === 'undefine' || thumbnailTime === null || thumbnailTime === '') {
        time = Math.floor(Math.random() * endTime);
    } else {
        time = thumbnailTime;
    }

    const thumbnailPath = path.join(__dirname, `../public/thumbnails/thumbs_${name}.png`)


    ffmpeg(videoSource)
        .setStartTime(time)
        .outputOption('-vframes', '1')
        .output(thumbnailPath)
        .on ('end', () => {
            console.log('Thumbnail generated successfully');
            res.sendFile(thumbnailPath);
        })
        .on ('error', (err) => {
            console.error(err);
            res.status(500).send('Error generating thumbnail');
        })
        .run();
});
```

8.2 Apartado 8.2 Código del frontend

```
// States
const [file, setFile] = useState(null);
const videoRef = useRef(null);
const [videoSource, setVideoSource] = useState(null);
const [videoType, setVideoType] = useState(null);
const [startTime, setStartTime] = useState(0);
const [endTime, setEndTime] = useState(0);
const [videoName, setVideoName] = useState(null);
const [notificationsEnabled, setNotificationEnabled] = useState(false);
const [email, setEmail] = useState('');
const [generateThumbnailChecked, setGenerateThumbnailChecked] = useState(false);
const [thumbnailTime, setThumbnailTime] = useState('');
```

```
    useEffect(() => {
      if (videoRef.current && !videojs.getPlayers()[videoRef.current.id]) {
        // Crear el reproductor de video al montar el componente
        const player = videojs(videoRef.current, {
          controls: true,
          autoplay: true,
        });

        // Event Listener para obtener duración del video
        player.on('loadedmetadata', () => {
          const duration = player.duration();
          setEndTime(duration);
        });

        if (videoSource) {
          player.src({
            src: videoSource,
            type: videoType,
          });
        }

        return () => {
          player.dispose();
        };
      }
    }, [videoSource, videoType]);
```

```
const handleFileChange = (event) => {
  setFile(event.target.files[0]);
};

const handleSubmit = async (event) => {
  event.preventDefault();
  submitForm(file, videoSource, videoType, setVideoSource, setVideoType, setVideoName);
};

const handleTrim = async () => {
  trimVideo(videoSource, startTime, endTime, videoName, notificationsEnabled, email);
};

const handleGenerateThumbnail = async () => {
  generateThumbnail(videoSource, videoName, generateThumbnailChecked, thumbnailTime, endTime);
};
```

```
const handleClear = () => {
  setFile(null);
  setVideoSource(null);
  setVideoType(null);
  setStartTime(0);
  setEndTime(0);
  setVideoName(null);
  setNotificationEnabled(false);
  setEmail('');
  setGenerateThumbnailChecked(false);
  setThumbnailTime(0);
  window.location.reload();
};

const handlePlayBetweenTimes = () => {
  if (videoRef.current) {
    const player = videojs(videoRef.current);
    player.currentTime(startTime);
    player.play();
    player.on('timeupdate', () => {
      if (player.currentTime() >= endTime) {
        player.pause();
      }
    });
  }
};
```

8.2.1 HTML

```
return (
  <div>
    <div className="hero-container">
      <div className="hero-text">
        <h2>Generador de clip bajo demanda</h2>
        <p>Para comenzar, introduzca un vídeo y haga clic en Cargar</p>
        <form onSubmit={handleSubmit}>
          <input type="file" onChange={handleFileChange} aria-label="Select video file" />
          <button type="submit">Cargar</button>
        </form>
      </div>
    </div>

{videoSource && (
  <div>
    <div className="video-overlay">
      <div data-vjs-player>
        <video ref={videoRef} className="video-js vjs-default-skin" aria-label="Video player"></video>
      </div>
      <div className="video-options">
        <div className="video-option-time">
          <label htmlFor="start-time-input">Tiempo de inicio</label>
          <input
            type="number"
            id="start-time-input"
            min="0"
            max={endTime}
            value={parseInt(startTime)}
            onChange={(e) => setStartTime(e.target.value)}
            aria-label="Start Time"
          />
        </div>
        <div className="video-option-time">
          <label htmlFor="end-time-input">Tiempo final</label>
          <input
            type="number"
            id="end-time-input"
            min={startTime}
            value={parseInt(endTime)}
            onChange={(e) => setEndTime(e.target.value)}
            aria-label="End Time"
          />
        </div>
      </div>
```

```jsx
<div className="video-option">
  <div className="video-option-label">
    <label htmlFor="notifications-checkbox">¿Quiere ser notificado?</label>
  </div>
  <div className="video-option-notification">
    <input
      type="checkbox"
      id="notifications-checkbox"
      checked={notificationsEnabled}
      onChange={(e) => setNotificationEnabled(e.target.checked)}
      aria-label="Enable Notifications"
    />
    {notificationsEnabled && (
      <input
        type="email"
        id="email-input"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        aria-label="Email"
        placeholder="Email"
      />
    )}
  </div>
</div>
```

```
<div className='video-option'>
    <div className='video-option-label'>
        <label htmlFor='thumbnail-checkbox'>¿Desea generar una miniatura?</label>
    </div>
    <div className="video-option-thumbnail">
    <input
      type="checkbox"
      id="thumbnail-checkbox"
      checked={generateThumbnailChecked}
      onChange={(e) => setGenerateThumbnailChecked(e.target.checked)}
      aria-label="Generate Thumbnail"
    />
    {generateThumbnailChecked && (
        <div>
            <input
                type="number"
                id="thumbnail-time-input"
                min="0"
                max={endTime}
                value={thumbnailTime}
                onChange={(e) => setThumbnailTime(e.target.value)}
                aria-label="Thumbnail Time"
                placeholder="Thumbnail Time"
            />
            <button onClick={handleGenerateThumbnail}>Generar miniatura</button>
        </div>
    )}
  </div>
</div>
            <div className="video-controls">
              <button onClick={handlePlayBetweenTimes} disabled={!videoSource}>
                Previsualizar
              </button>
              <button onClick={handleTrim} disabled={!videoSource}>
                Generar
              </button>
              <button onClick={handleClear}>Limpiar</button>
            </div>
          </div>
        </div>
      )}
    </div>
  );
}

export default App;
```

Bibliografía

[1] Open-Source HTML 5. https://www.opensourceforu.com/2017/06/introduction-to-html5/

[2] SCRUM. https://www.scrum.org/resources/blog/que-es-scrum

[3] VideoJS. https://videojs.com/guides

[4] FFMPEG. https://ffmpeg.org/documentation.html

[5] OpenCV. https://opencv.org/

[6] LIBAV. https://en.wikipedia.org/wiki/Libav

[7] Jplayer (jQuery). https://jplayer.org/latest/developer-guide/

[8] Node.js. https://nodejs.org/es

[9] JavaScript. https://developer.mozilla.org/es/docs/Web/JavaScript

[10] CORS. https://lenguajejs.com/javascript/peticiones-http/cors/

[11] Multipart/form-data. https://www.geeksforgeeks.org/define-multipart-form-data/

[12] Multer. https://www.npmjs.com/package/multer

[13] Fluent-FFMPEG. https://www.npmjs.com/package/fluent-ffmpeg

[14] Nodemailer. https://www.npmjs.com/package/nodemailer

[15] React. https://es.react.dev/

[16] Axios. https://axios-http.com/es/docs/intro

[17] FormData. https://keepcoding.io/blog/formdata-en-javascript/

[18] Mimetype. https://developer.mozilla.org/es/docs/Learn/Server-side/Configuring_server_MIME_types

[19] TikTok. https://developer.mozilla.org/es/docs/Learn/Server-side/Configuring_server_MIME_types

[20] Instagram Reels. https://about.instagram.com/es-la/features/reels

[21] YouTube Shorts. https://www.youtube.com/shorts/

[22] Notificación Push. https://www.seoptimer.com/es/blog/notificaciones-push-que-son-por-que-usarlas/

[23] draw.io https://app.diagrams.net/

[24] Full Guide for Splitting Video with FFmpeg - https://filme.imyfone.com/video-editing-tips/splitting-video-with-ffmpeg/

[25] Express. https://expressjs.com/

[26] Morgan. https://github.com/expressjs/morgan

[27] Visual Studio Code. https://code.visualstudio.com/

[28] Git. https://git-scm.com/

[29] Github. https://github.com/

[30] Formularios Google. https://docs.google.com/forms/

[31] SMTP. https://es.wikipedia.org/wiki/Protocolo_para_transferencia_simple_de_correo

[32] Codec Libx264. https://ffmpeg.org/ffmpeg-codecs.html#libx264_002c-libx264rgb

[33] Codec aac. https://ffmpeg.org/ffmpeg-codecs.html#aac

[34] Vercel. https://vercel.com/

[35] Flixier. https://flixier.com/

[36] Clideo. https://clideo.com/es/

[37] Online Video Cutter. https://online-video-cutter.com/es/

[38] Flexclip. https://www.flexclip.com/es/

[39] Adobe Premiere Pro. https://www.adobe.com/es/products/premiere.html

[40] iMovie. https://www.apple.com/es/imovie/

[41] Shotcut. https://shotcut.org/