Universidad de Valladolid

Escuela de Ingeniería Informática
de Valladolid

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Informática
Mención De Ingeniería Software

MatroskaLearn: Aplicación de edición y visualización de contenido audiovisual con fines educativos

Autor:
Laura Caminero García

Tutor:
Dr. César Llamas Bello

Agradecimientos

En primer lugar, me gustaría dar las gracias a mi familia, por apoyarme desde que inicié la carrera y por confiar en que en algún momento la acabaría, acompañándome tanto en los buenos como en los malos momentos.

Agradezco también a mis amigos por apoyarme y ayudarme en todo lo que pudieron, por preocuparse por mí y por recordarme siempre que soy capaz de hacer lo que me proponga.

Por último, agradezco a mis profesores de la facultad de Ingeniería Informática por todo lo que me enseñaron, en concreto, a mi tutor en este trabajo, Cesar Llamas Bello, por su paciencia, su amabilidad, su disponibilidad y sus comentarios, correcciones y orientaciones, que me ayudaron a mejorar el trabajo según avanzaba etapa por etapa.

Resumen

Esta memoria refleja el desarrollo de una aplicación web diseñada como herramienta de apoyo al estudio musical, pensada para adaptarse a distintas formas de aprendizaje y servir como recurso didáctico tanto para músicos como para estudiantes. Se trata de una herramienta que permite estructurar un archivo de audio (por ejemplo, una pieza musical) dividiéndolo en partes significativas, a las que se puede asociar contenido visual y textual, favoreciendo así la comprensión, el análisis y la memorización de obras. Ofrece dos modos de uso: uno para crear y organizar el contenido, y otro orientado a la reproducción y visualización. La aplicación, creada exclusivamente con Angular, se ejecuta íntegramente en el navegador y realiza todo el procesamiento en el lado del cliente, sin depender de servidores o bases de datos.

A lo largo del presente documento, se detallan todas las fases seguidas en el proceso de creación de la aplicación, desde la captura de requisitos hasta su implementación y pruebas. También se incluye un manual de usuario y un repositorio Git, desde el cual la aplicación está disponible a través de GitHub Pages.

Palabras clave: Aplicación Web Progresiva, Herramienta para apoyo didáctico, Matroska, Aplicación musical, Aprendizaje de partituras.

Abstract

This report reflects the development of a web application designed as a support tool for musical study, designed to adapt to different forms of learning and serve as a didactic resource for both musicians and students. It is a tool that makes it possible to structure an audio file (for example, a piece of music) by dividing it into meaningful parts, to which visual and textual content can be associated, thus favouring the comprehension, analysis and memorisation of works. It offers two modes of use: one for creating and organising content, and the other for playback and visualisation. The application, created exclusively with Angular, runs entirely in the browser and performs all processing on the client side, without relying on servers or databases.

Throughout this document, all the phases followed in the process of creating the application are detailed, from requirements capture to implementation and testing. It also includes a user manual and a git repository from where the application is available on GitHub Pages.

Keywords: Progressive Web Application, Didactic support tool, Matroska, Music application, Score learning.

Índice general

Agradecimientos III

Resumen V

Abstract VII

1. Introducción 1

1.1. Motivación 1

1.2. Objetivos de este Trabajo de Fin de Grado 2

1.3. Estructura de la memoria 2

2. Planificación del proyecto 5

2.1. Metodología 5

2.2. Plan inicial 6

2.3. Análisis de riesgos 9

2.4. Presupuesto estimado 13

2.4.1. Coste de materiales (hardware y alquileres) 13

2.4.2. Coste de software 14

2.4.3. Coste de recursos humanos 15

2.4.4. Coste total estimado del proyecto 15

3. Estado del arte del proyecto **17**

3.1. Herramientas de apoyo en el aprendizaje musical 17

3.1.1. Editores y reproductores de partituras 18

3.1.2. Aplicaciones móviles para el aprendizaje musical 18

3.1.3. Sistemas de edición y reproducción de contenido multimedia 19

3.2. Tecnologías de soporte del proyecto 20

3.2.1. PWA 20

3.2.2. Angular 21

3.2.3. Matroska (MKV) 22

3.2.4. Web Assembly 22

3.2.5. Ffmpeg.wasm 23

3.2.6. Wavesurfer.js 24

3.2.7. Indexed DB 25

3.2.8. GitLab Pages 25

3.2.9. GitHub Pages 26

3.3. Lenguajes y herramientas del proyecto 26

4. Análisis **29**

4.1. Requisitos 29

4.1.1. Requisitos funcionales 29

4.1.2. Requisitos no funcionales 32

4.2. Casos de uso 33

4.3. Modelo de dominio 51

5. Diseño **53**

5.1. Arquitectura Lógica 53
5.2. Organización del proyecto 55
5.2.1. Diagrama de paquetes general de la aplicación 56
5.2.2. Diagrama de paquetes de app 58
5.3. Interfaz de usuario 63

6. Implementación y Despliegue **71**
6.1. Bibliotecas principales 71
6.1.1. Ffmpeg.wasm 71
6.1.2. Wavesurfer 74
6.2. Funcionalidad PWA 76
6.3. Uso de Indexed DB para la persistencia de imágenes 77
6.4. Despliegue en GitLab Pages 78
6.5. Despliegue en GitHub Pages 81

7. Pruebas **83**

8. Conclusiones y líneas futuras **85**
8.1. Conclusiones 85
8.2. Líneas futuras 86

Bibliografía **89**

A. Repositorio del código **93**
A.1. Enlace del repositorio en GitHub 93
A.2. Enlace de la página web 93
A.3. Organización del repositorio 93

B. Manual de usuario **95**

B.1. Crear un nuevo proyecto o cargar uno existente 95

B.2. Modo escritura 96

B.3. Modo lectura 99

Índice de figuras

2.1. Diagrama de Gantt de la planificación del proyecto 6

4.1. Diagrama de casos de uso simplificado de la aplicación 34

4.2. Diagrama de modelo de dominio de análisis de la aplicación 51

5.1. Diagrama de paquetes del proyecto Angular 56

5.2. Diagrama de paquetes de app en la aplicación 58

5.3. Diagrama de clases del paquete interfaces de la aplicación 62

5.4. Boceto de la interfaz (horizontal) de la aplicación separada por componentes 64

5.5. Boceto de la interfaz (vertical) de la aplicación separada por componentes 65

5.6. Interfaz de escritura de la aplicación (horizontal) 67

5.7. Interfaz de lectura de la aplicación (horizontal) 67

5.8. Interfaz de escritura de la aplicación (vertical) 68

5.9. Interfaz de lectura de la aplicación (vertical) 69

6.1. Sección Build > Jobs dentro del proyecto de GitLab 80

6.2. Sección Deploy > Pages dentro del proyecto GitLab 80

6.3. Sección Settings > Pages dentro del repositorio GitHub 82

B.1. Crear o cargar nuevo proyecto en la aplicación 95

B.2. Interfaz en modo escritura, dividida en partes principales 96

B.3. Crear una sección o marca dentro de la onda en la aplicación 97

B.4. Editar texto de una sección en la aplicación 98

B.5. Controles de edición y borrado de la imagen de una sección en la aplicación 98

B.6. Interfaz en modo lectura, dividida en partes principales 99

B.7. Crear una marca de lectura en la aplicación 100

B.8. Visualizar el reproductor en modo pantalla completa en la aplicación 101

Índice de tablas

2.1. Riesgos de un proyecto de software, adaptación de la tabla de riesgos de Boehm 9

2.2. Análisis de riesgo: Problemas de compatibilidad entre tecnologías. 10

2.3. Análisis de riesgo: Falta de experiencia con algunas librerías. 10

2.4. Análisis de riesgo: Fallos en la planificación temporal. 11

2.5. Análisis de riesgo: Cambios en los requisitos durante el desarrollo. 11

2.6. Análisis de riesgo: Enfermedad o indisposición (propia o del tutor). 12

2.7. Análisis de riesgo: Falta de experiencia con framework de desarrollo. 12

2.8. Matriz de exposición de los riesgos en función de su probabilidad e impacto 13

2.9. Resumen del coste de materiales del proyecto 14

2.10. Resumen del coste del software del proyecto 14

2.11. Resumen del coste de recursos humanos del proyecto 15

2.12. Resumen final del presupuesto estimado 15

3.1. Compatibilidad de WebAssembly en navegadores según el dispositivo 23

4.1. CU1: Crear nuevo audio editable 35

4.2. CU2: Cargar archivo creado con la aplicación 36

4.3. CU3: Reproducir Audio 37

4.4. CU4: Saltar reproducción a siguiente o anterior sección 38

4.5. CU5: Poner sección o marca en bucle 39

4.6. CU6: Seleccionar sección o marca para visualizar su información 40

4.7. CU7: Crear Sección 41

4.8. CU8: Editar Campos Sección 42

4.9. CU9: Borrar Sección 43

4.10. CU10: Crear Marca 44

4.11. CU11: Editar Campos Marca 45

4.12. CU12: Borrar Marca 46

4.13. CU13: Mover separación entre secciones 47

4.14. CU14: Mover marca 48

4.15. CU15: Cambiar formato de presentación del contenido en el reproductor 49

4.16. CU16: Exportar edición del audio 50

Capítulo 1

Introducción

En este primer capítulo se describe el origen de la idea del proyecto, así como el contexto en el que surge su desarrollo. Se expone la utilidad de la aplicación propuesta como herramienta de apoyo en el aprendizaje musical, enmarcándola dentro de las necesidades existentes en este ámbito.

A continuación, se presentan los objetivos fundamentales que se pretenden alcanzar, tanto desde el punto de vista funcional como técnico, estableciendo el propósito general del proyecto y sus metas principales.

Finalmente, se ofrece una visión general de la estructura del documento, describiendo brevemente los capítulos en los que se divide la memoria y el contenido que se aborda en cada uno de ellos.

1.1. Motivación

En los últimos años, el uso de tecnologías digitales en el ámbito musical ha experimentado un notable crecimiento. Herramientas digitales y aplicaciones interactivas han transformado la forma en la que estudiantes y profesionales se enfrentan al aprendizaje y análisis musical, facilitando el estudio de partituras, la práctica instrumental y la edición de audio.

Actualmente, existen diversas herramientas digitales orientadas a tareas específicas del tratamiento de audio. Algunas permiten marcar regiones o fragmentos dentro de una pista sonora, otras ofrecen opciones de anotación o transcripción, y también existen aplicaciones de edición de vídeo que permiten asociar imágenes a ciertos momentos del audio. Por otro lado, muchos reproductores de audio permiten reproducir fragmentos de forma individual o en bucle. Sin embargo, pocas soluciones permiten realizar todas estas acciones de forma integrada y accesible.

Dentro de los programas más populares en el trabajo con partituras, cabe destacar la aplicación de

MuseScore, una herramienta (parcialmente gratuita, ya que incluye contenido de pago) muy utilizada en el aprendizaje musical que permite vincular partituras con audio, ofreciendo una sincronización entre ambos [1]. Sin embargo, su uso se limita a partituras musicales y resulta muy complejo asociar imágenes o textos explicativos a los segmentos de audio.

La aplicación desarrollada en este trabajo pretende combinar todas las funcionalidades mencionadas en una sola herramienta: ofrecerá una interfaz fácil e interactiva que permitirá etiquetar mediante segmentos un audio, así como crear marcas sobre él. A estas secciones y marcas se les podrá añadir texto e imágenes, y se incluirán controles de reproducción útiles para interaccionar con ellas. Además, la aplicación permitirá exportar esta edición, pudiendo ser visualizada posteriormente por otros usuarios dentro de la misma aplicación.

Además de agrupar diversas funcionalidades útiles para la creación y visualización de contenido didáctico, esta aplicación es altamente versátil. No se limita únicamente al aprendizaje musical, sino que puede aplicarse a una variedad de campos educativos que integren la escucha de fragmentos de audio y la visualización de texto, como el aprendizaje de idiomas, entre otros.

1.2. Objetivos de este Trabajo de Fin de Grado

El principal objetivo de este trabajo es desarrollar una aplicación web de tipo frontend utilizando el framework Angular, sin depender de un servidor backend para su funcionamiento. La aplicación ofrecerá las funcionalidades descritas en la introducción, orientadas a la segmentación y enriquecimiento de archivos de audio con información visual y textual.

Se busca que esta aplicación sea una PWA (Progressive Web App) [2], de modo que pueda instalarse en dispositivos y funcionar en modo sin conexión tras su primera descarga, facilitando su uso en contextos educativos con acceso limitado a internet.

Durante el desarrollo del proyecto se seguirán las fases clásicas del ciclo de vida del software: captura de requisitos, análisis, diseño, implementación y pruebas, asegurando una metodología estructurada y coherente.

1.3. Estructura de la memoria

Esta memoria se estructura en ocho capítulos, acompañados de dos anexos y una bibliografía. A continuación, se ofrece una breve descripción del contenido de cada uno de ellos:

- **Introducción**: expone el origen y la motivación del proyecto, así como la necesidad que busca

cubrir. Además, se presentan los objetivos planteados para su desarrollo.

- **Planificación del proyecto**: describe la metodología de trabajo adoptada, el cronograma seguido y un análisis de riesgos con sus respectivos planes de mitigación o contingencia. También se incluye una estimación del presupuesto asociado al proyecto.
- **Estado del arte del proyecto**: ofrece un repaso de las tecnologías y herramientas más relevantes seleccionadas a lo largo del desarrollo del trabajo, así como un análisis de aplicaciones similares disponibles en el mercado, destacando sus funcionalidades y limitaciones.
- **Análisis**: detalla las fases del análisis de la aplicación, incluyendo la recopilación de requisitos funcionales y no funcionales, la definición de casos de uso iniciales y el modelo de dominio resultante.
- **Diseño**: presenta la arquitectura lógica de la aplicación y su organización interna mediante diagramas de paquetes, describiendo las clases principales y su funcionalidad. Asimismo, se detalla el proceso de diseño de la interfaz de usuario.
- **Implementación y Despliegue**: explica el uso de las principales bibliotecas, la integración de tecnologías clave como PWA e IndexedDB, y el proceso de despliegue de la aplicación.
- **Pruebas**: resume la metodología de pruebas adoptada y los tipos de pruebas realizados para validar el correcto funcionamiento de la aplicación.
- **Conclusiones y líneas futuras**: reflexiona sobre el desarrollo del proyecto desde una perspectiva personal y de aprendizaje. Además, se proponen posibles mejoras y ampliaciones para futuras versiones de la aplicación.
- **Anexo A. Repositorio del código**: proporciona el enlace al repositorio Git en el que se encuentra el código fuente de la aplicación, acompañado de un resumen de su estructura de carpetas. Asimismo, se incluye un enlace a la página web desde la que puede utilizarse la aplicación.
- **Anexo B. Manual de usuario**: incluye una guía breve y accesible que explica las funcionalidades principales de la aplicación y cómo utilizarla.

Capítulo 2

Planificación del proyecto

En este capítulo se presentan los elementos fundamentales sobre los que se ha basado la planificación del proyecto. En primer lugar, se describe la metodología empleada para organizar y gestionar el trabajo durante el desarrollo. A continuación, se expone el plan inicial propuesto, estructurado en fases y tareas que permiten una ejecución ordenada y progresiva del proyecto. Cada fase se detalla con el objetivo de reflejar cómo se ha distribuido el trabajo a lo largo del tiempo. Por último, se identifican y analizan los principales riesgos que podrían surgir durante el desarrollo del proyecto, evaluando cada uno de ellos con detalle.

2.1. Metodología

El desarrollo de la aplicación se ha llevado a cabo siguiendo un enfoque iterativo e incremental, inspirado en los principios de las metodologías ágiles [3]. A partir de una idea inicial general, se realizaron pruebas exploratorias con diferentes tecnologías y bibliotecas para determinar la viabilidad del proyecto y delimitar su alcance.

El trabajo se estructuró en ciclos cortos (generalmente de 2 semanas de duración) centrados en el desarrollo de objetivos concretos, en los cuales se combinaban tareas de análisis, diseño e implementación según las necesidades de cada caso. Cuando se introducían elementos complejos o nuevas interacciones en la interfaz, se prestaba especial atención a las fases de análisis y diseño, aunque el eje principal de cada iteración seguía siendo la implementación y la validación práctica de los avances realizados.

Al finalizar cada ciclo, se revisaban los resultados junto al tutor del proyecto, lo que permitía ajustar la aplicación, corregir errores o añadir nuevas funcionalidades según el feedback recibido. Este enfoque flexible facilitó una evolución continua del proyecto, adaptándose a las necesidades emergentes y mejorando progresivamente la usabilidad y la funcionalidad de la aplicación.

2.2. Plan inicial

Para llevar a cabo la planificación del proyecto, se identificaron las principales fases o etapas que lo componen, algunas de las cuales se subdividieron en tareas más específicas. La Figura 2.1 presenta un esquema visual que recoge estas fases, las tareas asociadas, sus respectivas duraciones (finales) y las relaciones de dependencia entre ellas. Este tipo de representación, conocido como diagrama de Gantt [4], resulta especialmente útil para obtener una visión global del desarrollo temporal del proyecto, facilitando la organización, el seguimiento del progreso y la gestión eficiente del tiempo. A continuación, se describe en detalle cada una de estas fases.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predece
1	**Def. inicial problema**	**7 días**	**mié 11/09/24**	**jue 19/09/24**	
2	Discusión de las ideas del cliente	7 días	mié 11/09/24	jue 19/09/24	
3	Recolección de requisitos iniciales de la aplicación	7 días	mié 11/09/24	jue 19/09/24	
4	Planificación	7 días	vie 20/09/24	lun 30/09/24	1
5	**Investigación Tecnologías**	**10 días**	**mar 01/10/24**	**lun 14/10/24**	4
6	Investigar frameworks de desarrollo front-end	4 días	mar 01/10/24	vie 04/10/24	
7	Investigar sobre distintos contenedores de vídeo	3 días	lun 07/10/24	mié 09/10/24	6
8	Investigar librerías de manipulación de archivos de vídeo	3 días	jue 10/10/24	lun 14/10/24	7
9	**Elaboración prototipos**	**14 días**	**mar 15/10/24**	**vie 01/11/24**	5
10	Elaboración en Angular de prototipo de lectura de datos de archivo MKV	14 días	mar 15/10/24	vie 01/11/24	
11	Elaboración en Angular de prototipo de escritura de datos en archivo MKV	14 días	mar 15/10/24	vie 01/11/24	
12	**Bocetaje de la interfaz**	**7 días**	**lun 04/11/24**	**mar 12/11/24**	9
13	Creación de bocetos de la interfaz de lectura en escritorio, móviles y tablets	7 días	lun 04/11/24	mar 12/11/24	
14	Creación de bocetos de la interfaz de escritura en escritorio, móviles y tablets	7 días	lun 04/11/24	mar 12/11/24	
15	Desarrollo iterativo e incremental	92 días	mié 13/11/24	jue 20/03/25	12
16	**Pruebas**	**19 días**	**vie 21/03/25**	**mié 16/04/25**	15
17	Periodo de prueba por parte de los usuarios	5 días	vie 21/03/25	jue 27/03/25	
18	Implementación de sugerencias y ajustes	14 días	vie 28/03/25	mié 16/04/25	17
19	Elaboración memoria	32 días	jue 17/04/25	vie 30/05/25	16

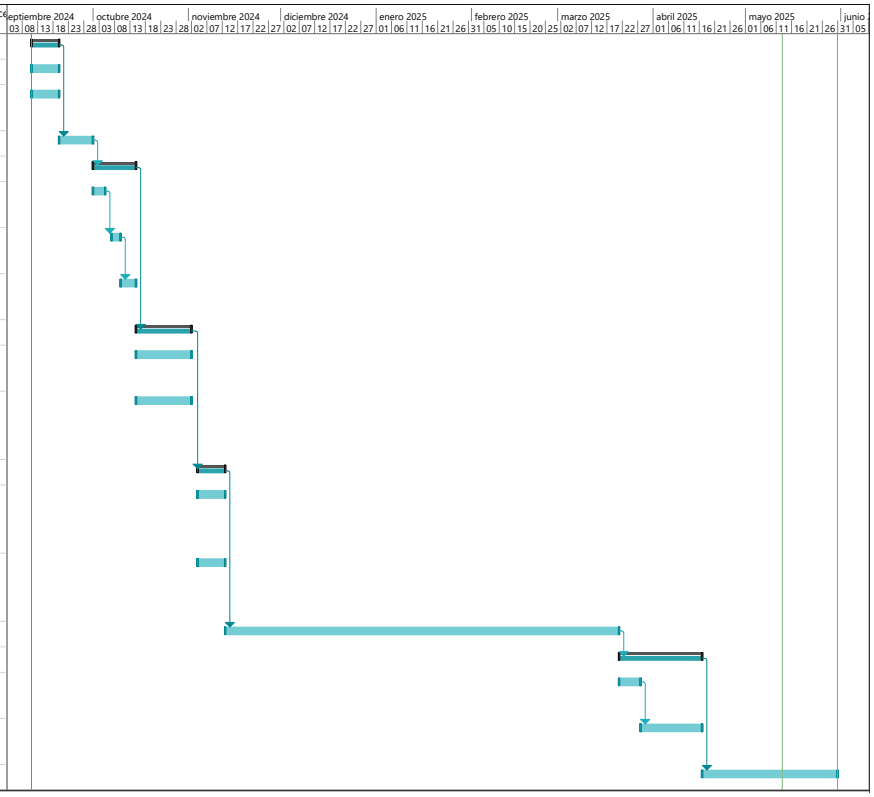

Figura 2.1: Diagrama de Gantt de la planificación del proyecto

Fase 1: Definición inicial del problema

En esta fase inicial, se discutieron las primeras ideas del proyecto junto con el profesor. Aunque no se definieron todos los detalles, se estableció el propósito de la aplicación y los objetivos generales, y a partir de ellos, se desarrolló una primera lista de requisitos.

Las tareas en las que se descompone esta fase son:

- Discusión de las ideas del cliente

- Recolección de requisitos iniciales de la aplicación

Fase 2: Planificación

Se estableció la planificación inicial del proyecto, donde se identificaron las principales fases que lo compondrían, que fueron desglosadas en tareas. Esta planificación sirvió como guía durante todo el proceso, permitiendo mantener una visión global del proyecto y ajustarse a los plazos establecidos.

Fase 3: Investigación de Tecnologías

En esta fase, se seleccionan las tecnologías más adecuadas para el desarrollo de la aplicación. Esto incluyó la elección del framework de desarrollo de front-end adecuado (Angular), la evaluación del alcance de bibliotecas para la manipulación de archivos de video, así como la investigación sobre qué tipo de archivo de video sería el más adecuado para almacenar los contenidos generados por la aplicación (MKV, MP4, AVI, etc.).

Las tareas en las que se descompone esta fase son:

- Investigar frameworks de desarrollo front-end
- Investigar librerías de manipulación de archivos de vídeo
- Investigar sobre distintos contenedores de vídeo

Fase 4: Elaboración de prototipos

En esta fase, se construyeron pequeños prototipos ejecutables utilizando distintas tecnologías y herramientas con el objetivo de evaluar su viabilidad y determinar cuáles eran las más adecuadas para el desarrollo de la aplicación. Estos prototipos permitieron identificar qué funcionalidades eran técnicamente factibles de implementar y cuáles requerirían ajustes o replanteamientos.

Las tareas en las que se descompone esta fase son:

- Elaboración en Angular de prototipo de lectura de datos de archivo MKV
- Elaboración en Angular de prototipo de escritura de datos en archivo MKV

Fase 5: Bocetaje de la interfaz

Durante esta fase, se elaboraron los primeros bocetos de la interfaz de usuario de forma informal, dibujados a mano en papel. Estos esquemas iniciales sirvieron como guía para definir la estructura básica de la aplicación, priorizando la accesibilidad y la facilidad de uso.

Las tareas en las que se descompone esta fase son:

- Creación de bocetos de la interfaz de lectura en escritorio, móviles y tablets
- Creación de bocetos de la interfaz de escritura en escritorio, móviles y tablets

Fase 6: Desarrollo iterativo e incremental

Durante esta fase, el desarrollo de la aplicación se llevó a cabo en ciclos breves, cada uno centrado en un pequeño conjunto de requisitos funcionales. Al final de cada iteración, se generaba un prototipo ejecutable que integraba las funcionalidades desarrolladas hasta ese momento. Si era necesario, también se actualizaban o creaban nuevos elementos en los diagramas correspondientes a las fases de análisis y diseño. Estos avances se presentaban al tutor del proyecto, cuya retroalimentación permitía detectar posibles mejoras, introducir ajustes en el comportamiento del sistema o definir nuevos requisitos funcionales para futuras iteraciones.

Fase 7: Pruebas

En esta fase, la aplicación fue evaluada por varios grupos de personas con el objetivo de comprobar su correcto funcionamiento. A través del uso real del programa, se verificaron sus funcionalidades principales y se detectaron errores o comportamientos inesperados, los cuales fueron corregidos. Además, los usuarios realizaron sugerencias de mejora relacionadas con la usabilidad y el diseño de la interfaz, algunas de las cuales fueron implementadas.

Las tareas en las que se descompone esta fase son:

- Periodo de prueba por parte de los usuarios
- Implementación de sugerencias y ajustes

Fase 8: Elaboración de la memoria

Una vez finalizado el desarrollo de la aplicación, se procedió a la redacción de la memoria del trabajo. Esta fase incluyó la recopilación y organización de la información técnica y metodológica del proyecto, así como la documentación de las distintas fases del proceso.

2.3. Análisis de riesgos

La identificación y el análisis de los riesgos que pueden surgir durante la realización del proyecto es fundamental en la fase de planificación. Como punto de partida para la identificación de los riesgos asociados a este proyecto, se ha tomado como referencia una adaptación de la tabla de riesgos propuesta por Boehm, extraída del libro Software Project Management [5]. En la Tabla 2.1, se señalan aquellos riesgos que podrían tener un impacto en el desarrollo del proyecto.

Riesgo	**¿Riesgo en este proyecto?**
Falta de personal	**Sí**
Estimaciones de tiempo y coste poco realistas	**Sí**
Desarrollo de funciones de software incorrectas	**Sí**
Desarrollo de una interfaz de usuario incorrecta	**Sí**
Sobrecarga de funciones innecesarias	**No**
Cambios tardíos en los requisitos	**Sí**
Falta de componentes externos suministrados	**No**
Falta de tareas realizadas externamente	**No**
Falta de rendimiento en tiempo real	**No**
El desarrollo es técnicamente demasiado difícil	**Sí**

Tabla 2.1: Riesgos de un proyecto de software, adaptación de la tabla de riesgos de Boehm

A partir de las ideas extraídas de la tabla de riesgos de Boehm, adaptadas a las características específicas de este proyecto, se han identificado una serie de riesgos más concretos y contextualizados. A continuación, se presentan estos riesgos junto con su probabilidad de ocurrencia, su impacto y el plan de acción propuesto para gestionarlos en caso de que se materialicen. Para la mayoría de ellos, se ha definido un plan de mitigación con el objetivo de reducir, en la medida de lo posible, la probabilidad de que lleguen a ocurrir.

Riesgo 1	**Problemas de compatibilidad entre tecnologías**				
Descripción	Dificultades al integrar librerías y tecnologías distintas o problemas de compatibilidad con distintos navegadores.				
Probabilidad	Media	**Impacto**	Alto	**Exposición**	Alta
Consecuencias	Imposibilidad de utilizar ciertas funcionalidades clave o necesidad de reescribir partes del proyecto.				
Plan de mitigación	Implementación de pequeños prototipos previos al comienzo del desarrollo de la aplicación, con el objetivo de comprobar la compatibilidad entre las tecnologías que se desean utilizar juntas.				
Plan de contingencia	Buscar alternativas compatibles y consultar foros o documentación.				

Tabla 2.2: Análisis de riesgo: Problemas de compatibilidad entre tecnologías.

Riesgo 2	**Falta de experiencia con algunas librerías**				
Descripción	Dificultades derivadas del uso de librerías nuevas o complejas, como Wavesurfer o FFmpeg.				
Probabilidad	Alta	**Impacto**	Medio	**Exposición**	Alta
Consecuencias	Aumento del tiempo de desarrollo por necesidad de investigación y pruebas, errores o uso ineficiente de las librerías.				
Plan de mitigación	Dedicar tiempo al aprendizaje práctico con pequeños ejemplos antes de implementarlas en el proyecto final, consultando la documentación oficial y apoyándose en comunidades de desarrollo.				
Plan de contingencia	Ampliar el plazo de finalización del proyecto y/o priorizar funcionalidades esenciales.				

Tabla 2.3: Análisis de riesgo: Falta de experiencia con algunas librerías.

Riesgo 3	**Fallos en la planificación temporal**				
Descripción	Subestimar el tiempo necesario para la realización de ciertas tareas.				
Probabilidad	Media	**Impacto**	Alto	**Exposición**	Alta
Consecuencias	Menor calidad en el resultado final por falta de revisión, documentación o pruebas.				
Plan de mitigación	Incluir tiempos de margen generosos para la finalización de las tareas durante la planificación del proyecto.				
Plan de contingencia	Ampliar el plazo de finalización del proyecto y/o priorizar funcionalidades esenciales.				

Tabla 2.4: Análisis de riesgo: Fallos en la planificación temporal.

Riesgo 4	**Cambios en los requisitos durante el desarrollo**				
Descripción	Aparición de nuevas necesidades por parte del cliente o por descubrimientos durante el desarrollo.				
Probabilidad	Media	**Impacto**	Alto	**Exposición**	Alta
Consecuencias	Tener que rehacer o construir nuevas partes en el proyecto, resultando en una pérdida de tiempo.				
Plan de mitigación	Mantener comunicación constante con el tutor y trabajar con una arquitectura modular que facilite los cambios.				
Plan de contingencia	Implementación de los cambios fundamentales y/o reducción de los requisitos, implementando los más básicos.				

Tabla 2.5: Análisis de riesgo: Cambios en los requisitos durante el desarrollo.

Riesgo 5	**Enfermedad o indisposición (propia o del tutor)**				
Descripción	Problemas de salud que impidan trabajar con normalidad o retrasen la corrección y supervisión del trabajo.				
Probabilidad	Baja	**Impacto**	Medio	**Exposición**	Media
Consecuencias	Retrasos en la entrega o en la validación de ciertas partes del TFG.				
Plan de mitigación	Realizar reuniones frecuentes para la revisión del proyecto y la modificación de sus requerimientos.				
Plan de contingencia	Priorizar funcionalidades básicas, claras y con menor probabilidad de modificación.				

Tabla 2.6: Análisis de riesgo: Enfermedad o indisposición (propia o del tutor).

Riesgo 6	**Falta de experiencia con framework de desarrollo**				
Descripción	Dificultad a la hora de manejar problemas comunes que pueden surgir durante el desarrollo, o a la hora de entender conceptos sobre el funcionamiento del framework.				
Probabilidad	Baja	**Impacto**	Medio	**Exposición**	Media
Consecuencias	Aumento significativo del tiempo en el que se implementan las funcionalidades de la aplicación.				
Plan de mitigación	Elección de un framework de desarrollo con el que se tenga experiencia previa.				
Plan de contingencia	Búsqueda de soluciones en la documentación oficial o en foros, y, si fuese necesario, adaptación del diseño para evitar funcionalidades demasiado complejas.				

Tabla 2.7: Análisis de riesgo: Falta de experiencia con framework de desarrollo.

Como se ha mostrado en las tablas anteriores, se han identificado un total de seis riesgos relevantes para este proyecto, cada uno de ellos evaluado en función de su probabilidad de ocurrencia y su impacto potencial. A partir de estas dos variables se calcula la exposición al riesgo, entendida como el nivel de amenaza que representa un riesgo concreto en función de su posibilidad de ocurrencia y las consecuencias asociadas.

En la Tabla 2.8 se presenta una matriz de probabilidad-impacto, en la que la probabilidad se dispone en las filas y el impacto en las columnas. Esta representación permite situar cada uno de los seis riesgos identificados en su correspondiente celda, en función de sus valores individuales.

Probabilidad \ Impacto	Bajo	Medio	Alto
Alta		R2	
Media			R1, R3, R4
Baja		R5, R6	

Tabla 2.8: Matriz de exposición de los riesgos en función de su probabilidad e impacto

El color de cada celda indica el grado de exposición al riesgo: el verde representa una exposición baja, el amarillo una exposición media y el rojo una exposición alta. De acuerdo con esta clasificación, los riesgos R1, R2, R3 y R4 presentan una exposición alta, por lo que se deberá prestar especial atención a su gestión y adoptar medidas preventivas prioritarias para reducir su probabilidad o impacto.

2.4. Presupuesto estimado

El presente apartado recoge una estimación económica del desarrollo del proyecto, abordando los principales costes relacionados con los materiales, el software y los recursos humanos. Aunque el proyecto ha sido realizado por una única persona en el contexto académico, se ha simulado una situación profesional realista con valores aproximados del mercado actual.

2.4.1. Coste de materiales (hardware y alquileres)

Ordenador El proyecto se ha desarrollado en un portátil *ASUS ROG Strix G15 (G513IH-HN008T)*, cuyo precio aproximado en 2021 fue de 1.200 €. Para calcular su coste se ha utilizado una amortización lineal a 4 años (48 meses).

- Precio del equipo: 1.200 €
- Amortización mensual: 1.200 / 48 = 25 €/mes
- Duración del proyecto: 9 meses

Coste amortizado: 25 € × 9 = **225 €**

Alojamiento web Se ha empleado GitHub Pages como sistema de alojamiento. Si bien cuenta con un plan gratuito, en un contexto profesional con múltiples sitios web activos, se requeriría GitHub Pro.

Coste de GitHub Pro: 4 €/mes × 9 meses = **36 €**

En la Tabla 2.9 se presenta un resumen del coste de los materiales, donde se detalla el precio total correspondiente.

Concepto	**Coste**
Ordenador (amortización 9 meses)	225 €
Alojamiento web (GitHub Pro)	36 €
Total materiales	**261 €**

Tabla 2.9: Resumen del coste de materiales del proyecto

2.4.2. Coste de software

A continuación, se enumeran las herramientas utilizadas, indicando si se ha empleado una versión gratuita o una licencia de pago, así como su coste en este último caso. La Tabla 2.10 recoge toda esta información junto con el coste total del software empleado.

Herramienta	**Licencia**	**Coste**
Astah Professional	Pago (9 €/mes × 3 meses)	27 €
Microsoft Project	Pago (10,50 €/mes × 3 meses)	31,50 €
Angular	Libre	0 €
Draw.io	Gratuito	0 €
Paint 3D	Incluido en Windows	0 €
Overleaf	Gratuito	0 €
GitHub	Incluido en materiales	–
Visual Studio Code	Gratuito	0 €
Total software		**58,50 €**

Tabla 2.10: Resumen del coste del software del proyecto

2.4.3. Coste de recursos humanos

Aunque el proyecto ha sido desarrollado de forma individual, se ha simulado la participación de tres roles profesionales: analista junior, diseñadora junior y programadora junior. Para ello, se ha estimado un salario bruto promedio en España para cada perfil profesional, a partir de consultas en los portales InfoJobs [6], Glassdoor [7] y Talent [8]. A estos importes se les ha añadido un 30 % adicional en concepto de costes indirectos (cotizaciones sociales, seguros, etc.). Toda esta información, junto con el coste total estimado, se recoge en la Tabla 2.11.

Rol	**Salario bruto anual**	**Coste mensual (con 30 %)**	**Meses**	**Total**
Analista junior	24.000 €	2.400 €	3	7.200 €
Diseñadora junior	20.000 €	2.167 €	3	6.500 €
Programadora junior	22.000 €	2.383 €	3	7.150 €
Total recursos humanos				**20.850 €**

Tabla 2.11: Resumen del coste de recursos humanos del proyecto

2.4.4. Coste total estimado del proyecto

A continuación se presenta en la Tabla 2.12 el coste total estimado, sumando todos los apartados anteriores:

Categoría	**Importe**
Coste de materiales	261 €
Coste de software	58,50 €
Coste de recursos humanos	20.850 €
Total estimado del proyecto	**21.169,50 €**

Tabla 2.12: Resumen final del presupuesto estimado

Este presupuesto representa una estimación económica orientativa basada en precios de mercado y valores medios en España. Se ha planteado con criterios realistas, como si el desarrollo se hubiera realizado en un entorno profesional.

Capítulo 3

Estado del arte del proyecto

En este capítulo se recogen los conceptos y referencias necesarias para comprender el contexto en el que se enmarca el presente trabajo. En primer lugar, se presenta una recopilación de herramientas existentes en el mercado orientadas al apoyo en el aprendizaje musical, describiendo sus principales funcionalidades y ámbitos de aplicación. El objetivo de este apartado es ofrecer una visión general del tipo de soluciones tecnológicas que ya se utilizan en el ámbito educativo musical.

A continuación, se describen las tecnologías empleadas en el desarrollo del proyecto, exponiendo brevemente en qué consisten y cuál ha sido el motivo de su elección.

Por último, se detallan los lenguajes de programación y otras herramientas utilizadas durante el desarrollo, así como su papel dentro del proyecto.

3.1. Herramientas de apoyo en el aprendizaje musical

El desarrollo tecnológico ha impulsado la aparición de múltiples herramientas digitales destinadas a facilitar el proceso de aprendizaje musical. Estas herramientas permiten desde la visualización y edición de partituras hasta el entrenamiento auditivo, pasando por la integración de contenidos multimedia. En este contexto, existen diversos formatos de archivo que se han consolidado como estándar [9], entre ellos:

- **PDF**: es un formato de documento de tipo imagen estática que se usa para distribuir partituras en una forma visual fija, tal como se verían impresas en papel. No es editable musicalmente: es como una “foto” de la partitura. Se usa mucho porque es universal y se puede abrir en casi cualquier dispositivo.
- **MusicXML (MXML)**: es un formato abierto diseñado para compartir partituras musicales entre distintos programas de edición musical (como MuseScore, Finale o Sibelius). A diferencia del

PDF, permite editar la música (cambiar notas, compases, instrumentos, etc.) porque guarda toda la información musical estructurada, no solo la imagen [10].

- **MIDI**: no guarda la partitura visual, sino los eventos musicales (qué nota suena, cuánto dura, qué instrumento la toca...). Es ideal para reproducir música electrónicamente, analizarla o generar partituras a partir de ella. No contiene información visual exacta como una partitura, pero sí cómo suena [11].
- **MP3, WAV, OGG**: son formatos de audio que almacenan grabaciones musicales. Se usan para reproducir ejemplos sonoros, como interpretaciones de una obra o explicaciones grabadas.
- **SRT, VTT**: son formatos de subtítulos que permiten mostrar texto sincronizado con un audio o video. Se usan para poner anotaciones como explicaciones, letra de canciones o traducciones en el momento exacto en que se deben mostrar.

A continuación, se presenta una clasificación de las principales herramientas según su finalidad.

3.1.1. Editores y reproductores de partituras

Estos programas permiten escribir, editar y reproducir partituras musicales, así como exportarlas en formatos como PDF o MusicXML. Son fundamentales para estudiantes, compositores y profesores.

- **MuseScore**: software gratuito y de código abierto para la edición de partituras. Permite crear partituras desde cero, importar/exportar en MusicXML y reproducirlas con instrumentos virtuales. Es una de las opciones más accesibles y populares en el ámbito educativo [1].
- **Sibelius**: editor profesional desarrollado por Avid, muy utilizado en entornos académicos y editoriales. Ofrece herramientas avanzadas de notación, edición detallada, y reproducción realista. Tiene versiones gratuitas y de pago [12].
- **Finale**: otro editor profesional con gran trayectoria. Destaca por su flexibilidad y por ofrecer un control total sobre la notación. Es común en editoriales musicales y en conservatorios [13].

3.1.2. Aplicaciones móviles para el aprendizaje musical

Estas apps ofrecen funciones específicas orientadas al aprendizaje práctico, ya sea en la lectura de partituras o en el desarrollo de habilidades auditivas y rítmicas.

Apps centradas en partituras

- **MuseScore App**: permite visualizar y reproducir partituras desde el móvil, sincronizadas con la nube de MuseScore. Compatible con formatos MXML [14].
- **Capella Score Reader**: herramienta de notación musical compatible con MusicXML, con app móvil para consultar partituras [15].
- **Sibelius App**: versión ligera de Sibelius para dispositivos móviles, orientada a la revisión de partituras más que a la edición completa [16].

Apps para entrenamiento auditivo y rítmico

- **MobileSheets**: visor de partituras digitales en PDF, orientado a músicos en directo. Permite anotaciones, sincronización con audio, y control por pedal [17].
- **ScorePDF**: app sencilla para ver partituras en PDF, pensada para la práctica instrumental [18].
- **Complete Ear Trainer**: app didáctica centrada en entrenar el oído musical, con ejercicios de intervalos, acordes, escalas y dictados [19].
- **Chord AI**: analiza canciones en tiempo real y muestra los acordes que se están tocando, útil para estudiantes que aprenden de oído [20].

3.1.3. Sistemas de edición y reproducción de contenido multimedia

Estas herramientas permiten combinar y manipular diferentes tipos de medios como audio, video, texto, subtítulos o imágenes. Resultan útiles cuando el aprendizaje musical se apoya en grabaciones, anotaciones sincronizadas o contenidos audiovisuales más complejos.

- **Audacity**: editor de audio gratuito y multiplataforma. Permite grabar, cortar, mezclar y aplicar efectos al sonido. Es ampliamente utilizado en contextos educativos para analizar fragmentos musicales o grabar ejercicios [21].
- **DaVinci Resolve**: editor de video profesional que permite sincronizar pistas de audio con subtítulos, imágenes y textos. Aunque está orientado al cine, es útil en la creación de materiales educativos musicales que combinan imágenes, partituras y explicaciones [22].

Estas herramientas permiten realizar tareas como:

- Transcripción y análisis de grabaciones.

- Sincronización de audio con subtítulos (por ejemplo, para seguir una explicación teórica junto con el sonido).
- Integración de imágenes (como fragmentos de partituras o diagramas armónicos) con la línea temporal del video o audio.
- Edición de pistas de audio y voz, útil para crear ejemplos interactivos o evaluaciones musicales.

3.2. Tecnologías de soporte del proyecto

3.2.1. PWA

Una Aplicación Web Progresiva (PWA, por sus siglas en inglés) es un tipo de aplicación de software que se entrega a través de la web, desarrollada utilizando tecnologías web comunes como HTML, CSS y JavaScript. Está diseñada para funcionar en cualquier plataforma que utilice un navegador compatible con los estándares web. Las PWAs combinan lo mejor de las aplicaciones web y las aplicaciones nativas, ofreciendo funcionalidades como el trabajo sin conexión, notificaciones push y acceso al hardware del dispositivo, lo que permite crear experiencias de usuario similares a las aplicaciones nativas en dispositivos móviles y de escritorio [2].

Las PWAs se distinguen por una serie de características que las hacen únicas:

- **Progresivas**: Funcionan para cualquier usuario, independientemente del navegador que utilice, gracias a su enfoque de mejora progresiva.
- **Responsivas**: Se adaptan a diferentes tamaños de pantalla y dispositivos, ofreciendo una experiencia de usuario óptima en móviles, tabletas y ordenadores.
- **Independientes de la conectividad**: Pueden funcionar sin conexión a internet o en redes de baja calidad, utilizando tecnologías como los Service Workers para almacenar en caché los recursos necesarios.
- **Similares a aplicaciones nativas**: Ofrecen una experiencia de usuario comparable a las aplicaciones nativas, incluyendo navegación fluida y posibilidad de instalación en la pantalla de inicio del dispositivo.
- **Actualizables automáticamente**: Se actualizan de forma automática, asegurando que los usuarios siempre tengan acceso a la última versión sin necesidad de descargas manuales.
- **Seguras**: Se sirven a través de HTTPS, garantizando la seguridad de los datos y protegiendo contra ataques de intermediarios.

- **Descubribles**: Son indexables por los motores de búsqueda, lo que mejora su visibilidad en los resultados de búsqueda.
- **Instalables**: Permiten a los usuarios agregar un acceso directo a la aplicación en la pantalla de inicio de su dispositivo, sin necesidad de pasar por tiendas de aplicaciones.
- **Enlazables**: Pueden ser compartidas fácilmente mediante una URL, facilitando su distribución y acceso.

En este proyecto se ha optado por desarrollar una aplicación web progresiva (PWA) por diversas razones. En primer lugar, se busca que la aplicación sea accesible desde distintos tipos de dispositivos, independientemente del sistema operativo que utilicen. Esto responde al hecho de que los usuarios que crean y editan proyectos de audio probablemente lo hagan desde ordenadores de sobremesa, mientras que aquellos que simplemente desean visualizar un proyecto ya creado, por comodidad, tienden a utilizar dispositivos móviles o tabletas.

Una de las principales ventajas de las PWA es que permiten desarrollar una única aplicación que funciona en múltiples plataformas, ya que basta con disponer de un navegador web. Además, se consideraba importante que la aplicación pudiera instalarse fácilmente en el dispositivo y que fuese utilizable incluso en condiciones de conectividad limitada, siempre que ya haya sido cargada previamente, lo cual se asemeja al comportamiento de una aplicación nativa.

3.2.2. Angular

Angular es un framework para el desarrollo de aplicaciones web modernas, creado y mantenido por Google. Está escrito en TypeScript y permite construir interfaces dinámicas, modulares y escalables mediante una arquitectura basada en componentes. Su enfoque declarativo y su potente sistema de herramientas lo convierten en una opción robusta para el desarrollo frontend, especialmente en aplicaciones de una sola página (SPA, Single Page Applications) [23].

En este proyecto se ha optado por el uso de Angular en su versión 18. La elección de Angular como framework de desarrollo se debe, en primer lugar, a la experiencia previa con esta tecnología, lo cual ha facilitado una curva de desarrollo más ágil y eficiente. Además, Angular cuenta con una extensa comunidad de desarrolladores, una documentación oficial muy completa y un ecosistema maduro de herramientas, lo que lo convierte en una opción robusta y fiable para el desarrollo de aplicaciones web complejas y estructuradas. Otra de sus ventajas es la facilidad de desarrollo y depuración, ya que herramientas como ng serve permiten visualizar los cambios de forma inmediata y simplifican el proceso de prueba y corrección de errores durante el desarrollo.

3.2.3. Matroska (MKV)

Al diseñar la aplicación, surgió la necesidad de contar con un contenedor capaz de almacenar todos los datos generados en un proyecto de audio, de manera que pudieran ser fácilmente recuperados posteriormente. Dada la naturaleza de los elementos que debían incluirse (imágenes, audio y texto), se concluyó que la opción más adecuada era utilizar un contenedor de vídeo. Esta elección no solo permitía agrupar diferentes tipos de datos en un único archivo, sino que también facilitaba su manipulación gracias a la existencia de múltiples bibliotecas y herramientas que permiten extraer o insertar información en este tipo de contenedores.

Además, al tratarse de un contenedor de vídeo, se abre la posibilidad de que, en futuras versiones de la aplicación, los archivos generados puedan ser reproducibles como contenido audiovisual, y no solo utilizados como contenedores estructurados de datos, como ocurre en la versión actual, en la que solo pueden ser interpretados por la propia aplicación.

Tras analizar las distintas alternativas disponibles, se optó por el formato MKV (Matroska). Según la definición oficial del proyecto Matroska (MKV), este es un formato contenedor capaz de albergar un número ilimitado de pistas de vídeo, audio, imagen o subtítulos en un único archivo, siendo similar a otros contenedores como AVI, MP4 o ASF, pero de estándar abierto [24].

La elección de MKV por encima de otros formatos de contenedor de vídeo se debe a su gran flexibilidad, ya que permite almacenar una amplia variedad de tipos de pistas y metadatos, y a su naturaleza de código abierto, lo que facilita su integración, personalización y uso en aplicaciones a medida.

3.2.4. Web Assembly

WebAssembly (abreviado como Wasm) es un formato de código binario diseñado para ejecutarse en navegadores web modernos con un rendimiento cercano al nativo. Se trata de una tecnología que permite ejecutar código compilado desde lenguajes como C, C++ o Rust directamente en el navegador, lo que resulta especialmente útil para aplicaciones que requieren un procesamiento intensivo o acceso a funcionalidades de bajo nivel. Fue desarrollado con el objetivo de complementar JavaScript, no reemplazarlo, y está pensado para ser seguro, eficiente y portable entre diferentes plataformas [25].

En este proyecto, se recurrió al uso de una biblioteca basada en WebAssembly: ffmpeg.wasm [26], una adaptación del popular framework FFmpeg que permite procesar archivos multimedia directamente en el navegador. Para garantizar que esta tecnología pudiera utilizarse sin problemas, se realizó previamente una comprobación de compatibilidad de WebAssembly en los principales navegadores utilizados en ordenadores de sobremesa, dispositivos móviles y tabletas.

Según LambdaTest, una plataforma dedicada a la prueba y validación cruzada de aplicaciones web en diferentes navegadores y dispositivos, la puntuación de compatibilidad de WebAssembly en navegadores

alcanza los 92 puntos sobre 100 [27]. Esta puntuación indica un alto grado de soporte entre navegadores modernos, lo cual garantiza que las funcionalidades basadas en Wasm pueden ejecutarse correctamente en la gran mayoría de los entornos actuales.

A continuación, se presenta la Tabla 3.1 [27] [28] [29], que refleja la compatibilidad de WebAssembly según la versión del navegador y el tipo de dispositivo:

Navegador	Escritorio	Android	iOS
Google Chrome	Desde la versión 57	Desde la versión 57	Desde la versión 61
Mozilla Firefox	Desde la versión 52	Desde la versión 52	Desde la versión 68
Microsoft Edge	Desde la versión 16	Desde la versión 79	Desde la versión 79
Safari	Desde la versión 11	No disponible	Desde la versión 11.3
Opera	Desde la versión 44	Desde la versión 44	Desde la versión 45
Samsung Internet	No disponible	Desde la versión 6.2	No disponible
Brave	Desde la versión 57	Desde la versión 57	Desde la versión 61

Tabla 3.1: Compatibilidad de WebAssembly en navegadores según el dispositivo

3.2.5. Ffmpeg.wasm

FFmpeg es una biblioteca de código abierto ampliamente utilizada para procesar archivos de audio y vídeo. Permite convertir, grabar, transmitir y manipular contenido multimedia en una gran variedad de formatos. Su motor central proporciona una potente colección de herramientas de línea de comandos y bibliotecas que permiten realizar operaciones como transcodificación, muxing/demuxing, codificación/-decodificación y filtrado multimedia, entre otras funciones esenciales en la edición y reproducción de contenidos audiovisuales [30].

Tradicionalmente, FFmpeg ha estado disponible como una herramienta de línea de comandos para sistemas operativos como Linux, Windows y macOS, lo que permite a los usuarios invocar su funcionalidad directamente mediante terminal. Además, se han desarrollado bindings y wrappers que permiten su uso desde lenguajes de programación como Python, Java, C++, Rust, entre otros, facilitando su integración en aplicaciones de escritorio o backend. En estos entornos, FFmpeg se ejecuta sin restricciones de seguridad ni acceso, lo que permite manipular archivos del sistema directamente y aprovechar al máximo sus prestaciones.

Durante el desarrollo de esta aplicación, se hizo necesario contar con una herramienta capaz de leer y escribir en archivos MKV que contuvieran el audio, imágenes y texto de un proyecto multimedia generado

en la aplicación.Tras analizar varias opciones, FFmpeg destacó como la opción más robusta y versátil, gracias a su popularidad de uso, la abundante documentación disponible, su interfaz de línea de comandos sencilla y su extenso soporte para formatos multimedia, entre los cuales se incluye MKV. Aunque sus funcionalidades pueden considerarse generales y algo limitadas en cuanto al manejo de particularidades específicas de ciertos contenedores, resulta plenamente adecuada para los requisitos de este proyecto.

Sin embargo, FFmpeg no está disponible de forma nativa para JavaScript, lo que impide su uso directo en entornos web tradicionales. Esto se debe a que FFmpeg está escrito en lenguajes de bajo nivel, y está diseñado para ejecutarse principalmente en sistemas de escritorio o servidores. Para poder integrarlo en esta aplicación web, fue necesario recurrir a una versión de FFmpeg compilada a WebAssembly (WASM).

La opción seleccionada fue ffmpeg.wasm, una implementación que proporciona una interfaz de FFmpeg accesible desde JavaScript, basada en WebAssembly. Según su descripción oficial en npm, ffmpeg.wasm es una versión reducida de FFmpeg compilada para ejecutarse en el navegador, ofreciendo una API JavaScript sencilla para invocar comandos similares a los que se usarían en una terminal tradicional de FFmpeg [26]. Dado que esta versión está optimizada para funcionar en contextos con recursos limitados y dentro de las restricciones del navegador, algunas funcionalidades avanzadas no están disponibles. Aun así, su rendimiento y flexibilidad resultan adecuados para este proyecto, permitiendo realizar todas las operaciones necesarias sobre archivos MKV directamente desde la aplicación web.

3.2.6. Wavesurfer.js

Wavesurfer.js [31] es una biblioteca JavaScript que permite visualizar y controlar archivos de audio directamente desde el navegador, mediante una representación gráfica de su forma de onda. Su objetivo principal es facilitar la creación de interfaces interactivas para la reproducción y edición de audio, con un alto grado de personalización y control.

Esta biblioteca constituye el eje central de la interfaz de la aplicación desarrollada. Su elección se debe a que cumplía con todos los requisitos definidos por la aplicación en cuanto a la visualización, manipulación y reproducción del audio. Entre sus características más destacadas se encuentran:

- La posibilidad de representar visualmente el audio en forma de onda de manera clara e intuitiva.
- La posibilidad de definir y manipular regiones temporales y marcas sobre la onda (mediante el uso de su plugin Regions), brindando así una manera de representar y manipular gráficamente las secciones y marcas en las que se divide un proyecto de audio en la aplicación.
- Controles de reproducción integrados y personalizables, como reproducir, pausar, adelantar, retroceder y visualizar el tiempo actual.
- Una API completa que ofrece métodos para consultar y modificar parámetros de la onda, así como sincronizar eventos o interactuar dinámicamente con el audio.

- Amplias opciones de personalización visual de la onda.

Estas funcionalidades son esenciales para la experiencia del usuario y para la precisión en la edición y navegación del contenido sonoro. Además, Wavesurfer.js cuenta con una documentación extensa y clara, y una comunidad activa que facilita la resolución de problemas comunes mediante foros, repositorios y ejemplos disponibles en línea.

3.2.7. Indexed DB

IndexedDB es una API de JavaScript disponible de forma nativa en prácticamente todos los navegadores web modernos, tanto en sus versiones de escritorio como móviles. Permite almacenar datos estructurados de forma persistente en el dispositivo del usuario, incluso sin conexión a internet. A diferencia de otras opciones de almacenamiento web como localStorage o sessionStorage (limitadas en cuanto a tamaño y eficiencia), está diseñada para gestionar grandes volúmenes de datos y realizar consultas complejas, utilizando una base de datos orientada a objetos basada en pares clave-valor. Esta versatilidad permite trabajar con distintos tipos de datos, incluidos objetos binarios como archivos, lo que la convierte en una solución robusta para aplicaciones web que requieren almacenar información de manera eficiente y organizada [32].

En este proyecto, se utiliza una versión simplificada de la API mediante la librería idb [33], que facilita su uso al proporcionar una interfaz más accesible y moderna. Esta decisión se tomó para reducir la complejidad del código y agilizar el desarrollo.

3.2.8. GitLab Pages

GitLab Pages es un servicio gratuito que permite publicar sitios web estáticos directamente desde los repositorios alojados en GitLab. De forma sencilla y automática, despliega los proyectos web contenidos en estos repositorios siguiendo las instrucciones definidas en un archivo de configuración llamado gitlab-ci.yml, que especifica los pasos necesarios para construir y publicar la página. GitLab Pages se presenta como una alternativa al alojamiento en servidores de pago, ya que permite hospedar una aplicación web de forma gratuita, ideal para proyectos en desarrollo o con recursos limitados [34].

Para este proyecto, durante la fase de desarrollo se utilizó el GitLab Pages del GitLab de la Escuela de Ingeniería Informática de la Universidad de Valladolid [35] para publicar la aplicación en internet. La elección de esta plataforma se debió a que ya existía una cuenta creada en ese entorno, lo que facilitaba el trabajo colaborativo y permitía añadir al profesor para revisar los avances. Además, esta opción proporcionaba un mayor nivel de privacidad al proyecto, ya que se prefería mantenerlo restringido hasta contar con una versión estable y completa para su publicación en GitHub Pages.

3.2.9. GitHub Pages

GitHub Pages es un servicio gratuito proporcionado por GitHub que permite alojar sitios web estáticos directamente desde los repositorios del usuario [36], la funcionalidad equivalente que ofrece GitLab para la publicación de páginas web desde repositorios.

A diferencia de GitLab Pages, que utiliza un archivo de configuración para definir el proceso de despliegue, GitHub Pages permite publicar contenidos directamente desde una rama concreta del repositorio o desde una carpeta específica. Esta simplicidad en la configuración inicial facilita el proceso de publicación para sitios estáticos.

En este proyecto se utilizó GitHub Pages para publicar la versión final de la aplicación. Esta plataforma ofrecía una forma rápida y accesible de mostrar el resultado en línea, facilitando así su revisión y difusión. Además, se optó por esta solución para garantizar la disponibilidad del proyecto a largo plazo, ya que la cuenta de GitLab utilizada durante el desarrollo, perteneciente a la Escuela de Ingeniería Informática de Valladolid, será eliminada al finalizar los estudios.

3.3. Lenguajes y herramientas del proyecto

Typescript es un lenguaje de programación desarrollado por Microsoft que extiende a JavaScript añadiendo tipado estático. Esto significa que permite definir el tipo de datos (como número, texto, etc.) que una variable o función debe tener, ayudando a detectar errores antes de ejecutar el código. Aunque se escribe en TypeScript, el código se transpila a JavaScript, por lo que puede ejecutarse en cualquier navegador o entorno compatible con este lenguaje.

Una característica destacada de TypeScript es su soporte para ficheros de definición de tipos (.d.ts), que permiten describir la estructura de bibliotecas escritas en JavaScript como si fueran entidades de TypeScript con tipado estático. Esto facilita el uso de librerías existentes sin perder las ventajas del sistema de tipos, de forma similar a como funcionan los archivos de cabecera en C/C++ [37].

Se utilizó este lenguaje porque es el que emplea Angular de forma predeterminada.

UML es un lenguaje gráfico utilizado para visualizar, especificar, construir y documentar distintos aspectos de un sistema de software. Se compone de una serie de diagramas estandarizados que permiten representar su estructura, comportamiento e interacciones [38].

Astah Professional es una herramienta de modelado visual desarrollada por Change Vision que permite crear diagramas UML [39]. Fué el programa utilizado para crear los diagramas de paquetes, clases y casos de uso de la aplicación desarrollada.

Bootstrap es un framework de código abierto para el desarrollo de interfaces web modernas y adaptables. Proporciona una colección de herramientas basadas en HTML, CSS y JavaScript que facilitan la creación de diseños visualmente atractivos y funcionales sin necesidad de partir desde cero. Uno de sus elementos más destacados es el sistema de rejilla (grid system), que permite organizar los elementos de la página en filas y columnas, facilitando la alineación y distribución del contenido [40].

Además, Bootstrap ofrece un conjunto de clases responsivas que permiten adaptar fácilmente la presentación de la interfaz a distintos tamaños de pantalla (móviles, tabletas, escritorios...), aspecto que resultó bastante útil en el desarrollo de la aplicación.

Capítulo 4

Análisis

En este capítulo se aborda la fase de análisis del proyecto, la cual se divide en varias etapas. En primer lugar, se recogen los requisitos de la aplicación, clasificados en funcionales y no funcionales. A partir de estos requisitos, se derivan los casos de uso y se elabora el modelo de dominio conceptual.

4.1. Requisitos

A continuación, se detalla la lista final de requisitos (funcionales, de información y no funcionales) recogidos desde el inicio y durante el transcurso del desarrollo de la aplicación. Como ya se mencionó anteriormente, la aplicación cuenta con dos modos principales, lectura y escritura, que cuentan con requisitos propios y exclusivos.

4.1.1. Requisitos funcionales

Los requisitos funcionales de un proyecto software describen las funciones, comportamientos y operaciones que el sistema debe implementar para satisfacer las necesidades del usuario y los objetivos del negocio. En otras palabras, determinan qué debe hacer el sistema, especificando sus capacidades desde la perspectiva de la interacción con el usuario y el procesamiento de datos [41].

Estos requisitos pueden clasificarse a su vez en dos grandes categorías:

- **Requisitos funcionales básicos**: se centran en las funciones principales.
- **Requisitos funcionales de información**: definen qué datos debe manejar el sistema y cómo debe procesarlos, almacenarlos o presentarlos. Este tipo de requisitos detalla las estructuras de entrada y salida de la información y las relaciones entre ellas.

A continuación, se detallan los requisitos funcionales básicos de la aplicación, organizados en tres grupos: requisitos generales, que deben cumplirse tanto en el modo de lectura como en el de escritura; requisitos específicos del modo lectura; y requisitos específicos del modo escritura. Finalmente, se describen también los requisitos funcionales de información.

Requisitos funcionales generales

- El sistema tendrá dos modos de visualización de archivo: lectura y escritura
- El sistema permitirá al usuario la creación de secciones y marcas, asociadas a fragmentos de tiempo dentro del audio con el que se está trabajando
- En el sistema se podrá visualizar de forma gráfica el audio con el que se está trabajando, así como las secciones y marcas que se crearon sobre él y los minutos de reproducción y duración total del audio.
- El sistema permitirá hacer zoom sobre la representación gráfica del audio
- El sistema permitirá al usuario pausar el audio
- El sistema permitirá al usuario reproducir el audio
- El sistema permitirá al usuario reproducir el audio desde un punto en concreto
- El sistema permitirá al usuario reproducir una sección desde el inicio
- El sistema permitirá al usuario reproducir una marca desde el inicio
- El sistema permitirá al usuario reproducir en bucle una sección
- El sistema permitirá al usuario reproducir en bucle una marca
- El sistema permitirá al usuario saltar de la sección reproduciéndose actualmente a la siguiente sección
- El sistema permitirá al usuario saltar de la sección reproduciéndose actualmente a la anterior sección
- El sistema permitirá al usuario acceder a una lista de las secciones y marcas presentes actualmente en el audio
- El sistema permitirá al usuario seleccionar una sección o marca de la lista, para visualizar su información asociada
- El sistema permitirá visualizar la información de una sección
- El sistema permitirá visualizar la información de una marca

- El sistema permitirá la descarga de un archivo en formato MKV que contenga la información actual del audio (secciones y marcas)
- El sistema permitirá al usuario la creación de marcas en un puntos elegidos del audio (tanto en modo lectura como en modo escritura)

Requisitos funcionales específicos del modo escritura

- El sistema permitirá al usuario editar el nombre del archivo MKV que se generará a partir de la información del audio
- El sistema permitirá al usuario la creación de secciones dentro del audio. La forma de crearlas será dividiendo secciones ya existentes por un punto (tiempo) específico
- El sistema permitirá al usuario editar el color de una sección
- El sistema permitirá al usuario editar el título de una sección
- El sistema permitirá al usuario editar el texto de una sección
- El sistema permitirá al usuario asignar una imagen a una sección
- El sistema permitirá al usuario borrar una sección
- El sistema permitirá al usuario cambiar el tiempo de separación entre dos secciones
- El sistema permitirá al usuario editar el título de una marca
- El sistema permitirá al usuario editar el texto de una marca
- El sistema permitirá al usuario borrar una marca
- El sistema permitirá al usuario cambiar el tiempo de una marca (pudiendo trasladarse al tiempo que quiera en el audio, incluso si implica cambiar de sección)

Requisitos funcionales específicos del modo lectura

- El sistema seleccionará automáticamente la sección y la marca cuyos tiempos coincidan con el tiempo de reproducción actual
- El sistema permitirá al usuario ver la información de las secciones y marcas en modo pantalla completa

- El sistema permitirá al usuario cambiar el modo de visualización de la información de secciones y marcas
- El sistema permitirá al usuario cambiar la orientación de la información de secciones y marcas
- El sistema permitirá al usuario cambiar el tiempo de una marca creada en modo lectura

Requisitos funcionales de información

- El archivo MKV generado deberá contener: audio MP3 y datos sobre todas las secciones y marcas creadas
- Una sección es una región (su tiempo de inicio no coincide con el de fin) del audio, y de ella se recogen los siguientes datos: tiempo de inicio, tiempo de fin, título, texto, color (en el que se verá dentro de la representación gráfica del audio) e imagen
- Una marca forma parte de una sección. Referencia a un tiempo específico del audio, y de ella se recogen los siguientes datos: tiempo de inicio, título y texto
- Al tener una marca solo tiempo de inicio, la duración y por tanto, el tiempo de fin de esta marca, están determinados por el inicio de la siguiente marca a ella. Si no existe una marca después de ella, entonces su tiempo de fin será el de la sección a la que está asociada
- El modo de visualización de la información de secciones y marcas (en modo lectura) podrá ser "dividido" (se muestra el texto y la imagen de la sección y marca), "solo texto" (se muestra solo el texto de la sección y marca) y "solo imagen" (se muestra solo la imagen de la sección)
- La orientación de la información de secciones y marcas podrá ser "vertical" (texto a la izquierda e imagen a la derecha) y "horizontal" (imagen arriba y texto abajo)
- Existirá la opción de escribir y visualizar el texto de una sección en formato Markdown [42].

4.1.2. Requisitos no funcionales

En un proyecto de software, los requisitos no funcionales definen cómo debe comportarse el sistema, en lugar de las funcionalidades concretas que debe realizar. También se conocen como atributos de calidad, ya que establecen estándares sobre aspectos como el rendimiento, la seguridad, la usabilidad, la escalabilidad o la confiabilidad del sistema [43].

Estos requisitos son esenciales para garantizar que la solución no solo funcione correctamente, sino que también lo haga de manera eficiente, segura y satisfactoria para los usuarios, incluso en condiciones reales

de uso. Además, estos requisitos recogen las restricciones asociadas a la aproximación arquitectónica inicial del proyecto. En este caso, están centrados en la necesidad de que la solución adoptada sea una aplicación web progresiva (PWA).

A continuación, se presenta la lista de requisitos no funcionales identificados para esta aplicación, los cuales han sido definidos en función de las necesidades del sistema y las expectativas de calidad del usuario.

- La aplicación debe ser compatible con los navegadores más utilizados, garantizando su funcionamiento en, al menos, Firefox, Google Chrome y Microsoft Edge
- La aplicación debe funcionar una vez cargada incluso sin conexión a internet, y todo su procesamiento se realizará en el lado del cliente
- La aplicación será desarrollada usando un framework front-end que sea ampliamente utilizado,blabla utilizando los lenguajes Typescript, HTML y CSS
- La aplicación debe poder accederse y funcionar en su totalidad desde ordenadores de sobremesa, teléfonos móviles y tabletas
- La interfaz debe adaptarse de forma automática al tamaño de pantalla del dispositivo (interfaz responsive), manteniendo la usabilidad y legibilidad
- Las imágenes asociadas a secciones dentro de la aplicación se guardarán de forma persistente en el navegador

4.2. Casos de uso

Los casos de uso son una herramienta utilizada en el análisis de sistemas para describir las interacciones entre los usuarios y la aplicación. Su objetivo es representar, de forma clara y estructurada, las funcionalidades que debe ofrecer el sistema desde el punto de vista del usuario [44].

Mediante los casos de uso se identifican los diferentes escenarios en los que un usuario puede realizar acciones dentro de la aplicación, lo que permite definir sus requisitos funcionales de forma precisa y facilitar el diseño posterior.

En la Figura 4.1 se muestra el diagrama de casos de uso de la aplicación. Se trata de una versión simplificada, centrada únicamente en reflejar la distribución de los casos de uso entre los distintos actores del sistema, sin representar explícitamente las relaciones de dependencia entre ellos (las cuales se describen más adelante en las tablas correspondientes).

La aplicación contempla tres tipos de usuario: el Usuario Lector, que representa las acciones disponibles en el modo lectura; el Usuario Creador, asociado al modo escritura; y el Usuario General, del que heredan los dos anteriores y que agrupa las funcionalidades comunes a ambos modos de uso.

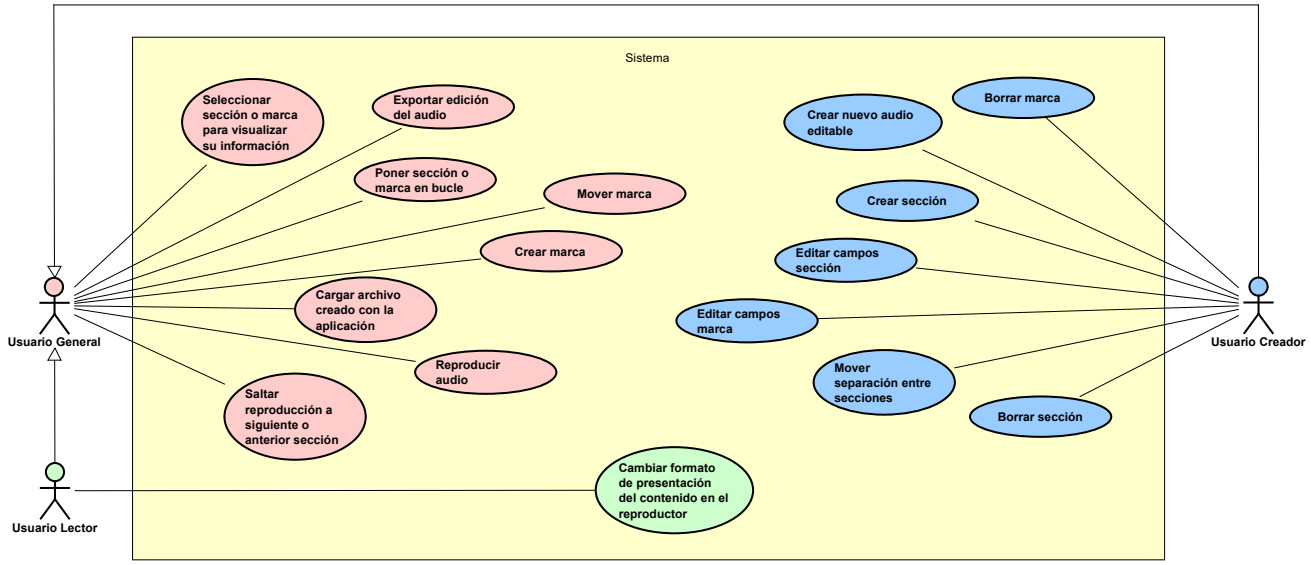

Figura 4.1: Diagrama de casos de uso simplificado de la aplicación

CU1	Crear nuevo audio editable
Descripción	Permite al usuario comenzar la edición de un nuevo archivo de audio.
Actor	Usuario Creador
Precondiciones	
Postcondiciones	Se muestra la representación gráfica del audio con su información asociada, disponible para editar. La información inicial del audio se reduce a una sección vacía que ocupa toda su duración.
Flujo Normal	1. El usuario solicita crear un nuevo audio editable. 2. El sistema pide al usuario que introduzca un archivo de audio en formato MP3. 3. El usuario introduce un archivo de audio MP3. 4. El sistema comprueba que el contenido proporcionado por el usuario sea válido. 5. El sistema muestra la representación gráfica del audio y su información asociada, incluyendo controles para su edición y manipulación. Se inicializa con una sección que abarca toda la duración del audio.
Excepciones	4.1 El fichero introducido por el usuario no es válido y el sistema notifica al usuario.

Tabla 4.1: CU1: Crear nuevo audio editable

CU2	Cargar archivo creado con la aplicación
Descripción	Permite al usuario cargar un archivo MKV que haya sido generado por la aplicación, para seguir editándolo o para visualizarlo en modo lectura.
Actor	Usuario Lector y Usuario Creador
Precondiciones	
Postcondiciones	Se muestra la representación del audio con su información asociada, disponible para editar en el caso de que el usuario sea Usuario Creador, y para visualizar en caso de que sea Usuario Lector.
Flujo Normal	1. El usuario solicita cargar un archivo. 2. El sistema pide al usuario que introduzca un archivo generado por la aplicación, en formato MKV. 3. El usuario introduce un archivo en formato MKV. 4. El sistema comprueba que el contenido proporcionado por el usuario sea válido. 5. El sistema muestra la representación del audio con su información asociada.
Flujos Alternativos	5.A Si el usuario es Usuario Creador, la información está disponible para su edición. 5.B Si el usuario es Usuario Lector, la información disponible para su visualización.
Excepciones	4.1 El fichero introducido por el usuario no es válido y el sistema notifica al usuario.

Tabla 4.2: CU2: Cargar archivo creado con la aplicación

CU3	Reproducir Audio
Descripción	Permite al usuario reproducir el audio desde el punto elegido
Actor	Usuario Creador y Usuario Lector
Precondiciones	Existe un audio cargado.
Postcondiciones	El audio se reproduce desde el punto indicado por el usuario.
Flujo Normal	1. El usuario solicita reproducir el audio. 2. El sistema reproduce el audio desde el punto indicado por el usuario. 3. El sistema comprueba el tipo de usuario, y si se trata de Usuario Creador, termina el caso de uso.
Flujos Alternativos	1.A El usuario solicita reproducir el audio desde un punto específico del mismo. 1.B El usuario solicita reproducir el audio desde el inicio de una de las secciones. 1.C El usuario solicita reproducir el audio desde el inicio de una de las marcas. 1.D El usuario solicita reproducir el audio desde el punto por el que se llegaba. 3.A.1 El sistema verifica que el usuario es Usuario Lector. 3.A.2 El sistema busca la sección y marca que se corresponden con el instante de tiempo actual y muestra sus datos en la interfaz.

Tabla 4.3: CU3: Reproducir Audio

CU4	Saltar reproducción a siguiente o anterior sección
Descripción	Permite al usuario saltar a la siguiente o anterior sección a la que se está reproduciendo actualmente.
Actor	Usuario General
Precondiciones	Existe un audio cargado y existe una sección anterior o posterior a la que se está reproduciendo actualmente.
Postcondiciones	El audio se reproduce desde la siguiente o anterior sección a la actual, según la decisión del usuario.
Flujo Normal	1. El usuario solicita saltar la reproducción de sección. 2. El sistema reproduce el audio desde el inicio de la anterior o siguiente sección a la actual, según la elección del usuario.
Flujos Alternativos	1.A El usuario solicita reproducir la siguiente sección a la que se está reproduciendo actualmente. 1.B El usuario solicita reproducir la anterior sección a la que se está reproduciendo actualmente.
Excepciones	2.A.1 No existe una sección posterior a la que se está reproduciendo actualmente, por lo que no se produce ninguna acción. 2.B.1 No existe una sección anterior a la que se está reproduciendo actualmente, por lo que no se produce ninguna acción.

Tabla 4.4: CU4: Saltar reproducción a siguiente o anterior sección

CU5	Poner sección o marca en bucle
Descripción	Permite al usuario reproducir una sección o marca en bucle
Actor	Usuario General
Precondiciones	Existe un audio cargado.
Postcondiciones	La sección o marca seleccionada se reproduce en bucle y se quita la reproducción en bucle que hubiera antes de ella.
Flujo Normal	1. El usuario solicita reproducir en bucle. 2. El sistema quita la reproducción en bucle anterior que hubiera anteriormente. 3. El sistema reproduce en bucle la marca o sección deseada, empezando desde el inicio de esta.
Flujos Alternativos	1.A El usuario solicita reproducir una sección en bucle. 1.B El usuario solicita reproducir una marca en bucle.

Tabla 4.5: CU5: Poner sección o marca en bucle

CU6	Seleccionar sección o marca para visualizar su información
Descripción	Permite al usuario seleccionar una sección o una marca para ver sus datos asociados.
Actor	Usuario General
Precondiciones	Existe un audio cargado
Postcondiciones	Se muestran los datos de la sección o marca seleccionada.
Flujo Normal	1. El usuario solicita ver las secciones y marcas que existen actualmente. 2. El sistema muestra al usuario una lista de todas las secciones y marcas que existen en el audio. 3. El usuario selecciona un ítem de la lista. 4. El sistema muestra los datos del ítem seleccionado.
Flujos Alternativos	3.A El usuario selecciona una sección de la lista. 3.B El usuario selecciona una marca de la lista.

Tabla 4.6: CU6: Seleccionar sección o marca para visualizar su información

CU7	Crear Sección
Descripción	Permite al usuario dividir una sección existente para crear una nueva.
Actor	Usuario Creador
Precondiciones	Existe un audio cargado.
Postcondiciones	La sección elegida queda dividida en dos secciones, y la de la izquierda conserva los datos de la original. Los cambios quedan reflejados en la interfaz y la nueva sección es seleccionada.
Flujo Normal	1. El usuario selecciona un punto dentro de una sección, en la representación gráfica del audio, y solicita crear algo en este punto. 2. El sistema pregunta al usuario si quiere crear una sección o una marca en este punto. 3. El usuario indica que quiere crear una nueva sección. 4. El sistema elimina la sección original y crea dos nuevas secciones, con separación en el punto designado por el usuario. 5. El sistema selecciona la nueva sección (derecha). 6. El sistema actualiza la interfaz para reflejar los cambios.

Tabla 4.7: CU7: Crear Sección

CU8	Editar Campos Sección
Descripción	Permite al usuario modificar los datos de una sección existente.
Actor	Usuario Creador
Precondiciones	Existe un audio cargado y una sección seleccionada.
Postcondiciones	Los cambios en la sección se guardan y reflejan en la interfaz.
Flujo Normal	1. El usuario solicita la edición de un campo de la sección seleccionada actualmente. 2. El sistema habilita la edición del campo solicitado por el usuario. 3. El usuario edita el campo solicitado y confirma al terminar. 4. El sistema muestra los cambios del campo modificado en la interfaz.
Flujos Alternativos	1.A El usuario solicita editar el título de la sección. 1.B El usuario solicita editar el texto de la sección. 1.C El usuario solicita editar la imagen de la sección. 1.D El usuario solicita editar el color de la sección.

Tabla 4.8: CU8: Editar Campos Sección

CU9	Borrar Sección
Descripción	Permite al usuario eliminar una sección del audio
Actor	Usuario Creador
Precondiciones	Existe un audio cargado y una sección seleccionada.
Postcondiciones	La sección seleccionada desaparece, y el espacio que ocupaba pasa a formar parte de la sección de su izquierda.
Flujo Normal	1. El usuario solicita la eliminación de la sección seleccionada actualmente. 2. El sistema comprueba si se puede eliminar la sección, y en caso afirmativo, solicita confirmación al usuario. 3. El usuario confirma la eliminación. 4. El sistema elimina la sección y amplia el espacio de la sección de su izquierda para ocupar su hueco. 5. El sistema actualiza la interfaz para reflejar los cambios.
Excepciones	1.1 El sistema comprueba que la sección seleccionada es la primera, y notifica al usuario de que no se puede eliminar, finalizando el caso de uso.

Tabla 4.9: CU9: Borrar Sección

CU10	Crear Marca
Descripción	Permite al usuario crear una nueva marca en el audio.
Actor	Usuario Creador y Usuario Lector
Precondiciones	Existe un audio cargado.
Postcondiciones	Se crea una marca con los datos introducidos, la marca queda seleccionada, y los cambios se reflejan en la interfaz.
Flujo Normal	1. El usuario selecciona un punto dentro de una sección, en la representación gráfica del audio, y solicita crear algo en este punto. 2. El sistema pregunta al usuario si quiere crear una sección o una marca en este punto. 3. El usuario indica que quiere crear una nueva marca. 4. El sistema comprueba el tipo de usuario. 5. El sistema crea una marca en el punto elegido por el usuario 6. El sistema comprueba si existen más marcas dentro de la sección a la que pertenece la marca creada, y en caso contrario, se procede. 7. El sistema selecciona la nueva marca. 8. El sistema refleja los cambios en la interfaz.
Flujos Alternativos	4.A.1 El sistema verifica que el usuario es Usuario Lector. 4.A.2 El sistema solicita al usuario que rellene los campos de la marca (título y texto). 4.A.3 El usuario rellena los campos de la marca y confirma. 4.B El sistema verifica que el usuario es Usuario Creador, por lo que la marca tendrá un título por defecto y un texto vacío. 6.A.1 El sistema verifica que existían más marcas dentro de la sección a la que pertenece la marca creada. 6.A.2 El sistema ajusta los tiempos de fin de las marcas según la nueva disposición.

Tabla 4.10: CU10: Crear Marca

CU11	Editar Campos Marca
Descripción	Permite al usuario modificar los datos de una marca existente.
Actor	Usuario Creador
Precondiciones	Existe un audio cargado y una marca seleccionada.
Postcondiciones	Los cambios en la marca se guardan y reflejan en la interfaz.
Flujo Normal	1. El usuario solicita la edición de un campo de la marca seleccionada actualmente. 2. El sistema habilita la edición del campo solicitado por el usuario. 3. El usuario edita el campo solicitado y confirma al terminar. 4. El sistema muestra los cambios del campo modificado en la interfaz.
Flujos Alternativos	1.A El usuario solicita editar el título de la marca. 1.B El usuario solicita editar el texto de la marca.

Tabla 4.11: CU11: Editar Campos Marca

CU12	Borrar Marca
Descripción	Permite al usuario eliminar una marca del audio.
Actor	Usuario Creador
Precondiciones	Existe un audio cargado y una marca seleccionada.
Postcondiciones	La marca seleccionada desaparece, los tiempos de fin de las marcas de la misma sección quedan ajustados y los cambios se reflejan en la interfaz.
Flujo Normal	1. El usuario solicita la eliminación de la marca seleccionada actualmente. 2. El sistema solicita confirmación al usuario. 3. El usuario confirma la eliminación. 4. El sistema elimina la marca. 5. El sistema comprueba si existían más marcas dentro de la sección a la que pertenece la marca eliminada, y en caso contrario, se procede. 6. El sistema actualiza la interfaz para reflejar los cambios.
Flujos alternativos	5.A.1 El sistema verifica que existían más marcas dentro de la sección a la que pertenece la marca eliminada. 5.A.2 El sistema ajusta los tiempos de fin de las marcas según la nueva disposición.

Tabla 4.12: CU12: Borrar Marca

CU13	Mover separación entre secciones
Descripción	Permite mover el punto temporal de separación entre dos secciones, cambiando el longitud de ambas.
Actor	Usuario Creador
Precondiciones	Existe un audio cargado y al menos dos secciones creadas.
Postcondiciones	El tiempo de separación entre las dos secciones cambia, reflejándose en la longitud de ambas, y los cambios se reflejan en la interfaz.
Flujo Normal	1. El usuario mueve la separación entre dos secciones en la representación gráfica del audio. 2. El sistema comprueba si el nuevo punto de separación se encuentra dentro de los límites válidos (no invade otra sección ajena a las dos), y en caso afirmativo, se procede. 3. El sistema modifica la longitud de las secciones y refleja los cambios en la interfaz.
Excepciones	2.1 El sistema verifica que el nuevo punto de separación seleccionado por el usuario no se encuentra dentro de los límites válidos. 2.2 El sistema coloca el punto de separación entre secciones en su posición inicial y el caso de uso finaliza.

Tabla 4.13: CU13: Mover separación entre secciones

CU14	Mover Marca
Descripción	Permite mover una marca a otro punto temporal del audio.
Actor	Usuario Creador y Usuario Lector
Precondiciones	Existe un audio cargado y al menos una marca dentro de él.
Postcondiciones	La ubicación de la marca cambia, los tiempos de fin de las marcas de la anterior y actual sección en la que se encuentra quedan ajustados y los cambios se reflejan en la interfaz.
Flujo Normal	1. El usuario mueve de lugar la marca deseada en la representación gráfica del audio. 2. El sistema comprueba si la marca cambió de sección, y en caso contrario, se procede. 3. El sistema comprueba si dentro de la sección en la que se encuentra la marca, existen más marcas aparte de ella. En caso contrario, se procede. 4. El sistema modifica la posición y el tiempo e finalización de la marca que movió el usuario. 5. El sistema actualiza la interfaz para reflejar los cambios.
Flujos alternativos	2.A.1 El sistema verifica que la marca cambió de sección. 2.A.2 El sistema ajusta los tiempos de finalización de las marcas de la anterior en la que se encontraba antiguamente la marca. 3.A.1 El sistema verifica que dentro de la sección en la que se encuentra la marca, existen más marcas aparte de ella. 3.A.2 El sistema ajusta los tiempos de finalización de las marcas de la sección en la que se encuentra actualmente la marca que se movió.

Tabla 4.14: CU14: Mover marca

CU15	Cambiar formato de presentación del contenido en el reproductor
Descripción	Permite modificar la forma de mostrar el contenido y su orientación dentro del reproductor.
Actor	Usuario Lector
Precondiciones	Existe un audio cargado.
Postcondiciones	La forma en la que se presenta el contenido mostrado en el reproductor cambia según las preferencias establecidas.
Flujo Normal	1. El usuario solicita cambiar el formato de presentación del contenido dentro del reproductor. 2. El sistema actualiza la interfaz del reproductor para reflejar los cambios en la presentación del contenido.
Flujos alternativos	1.A El usuario solicita cambiar la presentación del contenido a modo vertical. 1.B El usuario solicita cambiar la presentación del contenido a modo horizontal. 1.C El usuario solicita cambiar la presentación del contenido a modo sólo imagen. 1.D El usuario solicita cambiar la presentación del contenido a modo sólo texto. 1.E El usuario solicita cambiar la presentación del contenido a modo pantalla completa.

Tabla 4.15: CU15: Cambiar formato de presentación del contenido en el reproductor

CU16	Exportar edición del audio
Descripción	Permite al usuario guardar toda la información editada del audio en un archivo con formato MKV que pueda ser exportado.
Actor	Usuario General
Precondiciones	Existe un audio cargado.
Postcondiciones	El archivo MKV se genera y queda disponible para descarga o almacenamiento.
Flujo Normal	1. El usuario selecciona la opción de exportar los datos de edición a archivo MKV. 2. El sistema genera un archivo en formato MKV que contiene la información editada del audio (secciones y marcas). 3. El sistema inicia la descarga del archivo.
Excepciones	2.1 Ocurre un error en la generación del archivo MKV y el sistema notifica al usuario, concluyendo el caso de uso.

Tabla 4.16: CU16: Exportar edición del audio

4.3. Modelo de dominio

El modelo de dominio es una representación conceptual del sistema, que define los elementos clave del problema como entidades, sus atributos, relaciones y restricciones. Sirve para entender el vocabulario del dominio y la estructura estática del sistema, sin entrar en detalles de diseño o implementación [45].

En la Figura 4.2, se presenta el modelo de dominio de esta aplicación, elaborado a partir de los requisitos recolectados y los casos de uso definidos previamente.

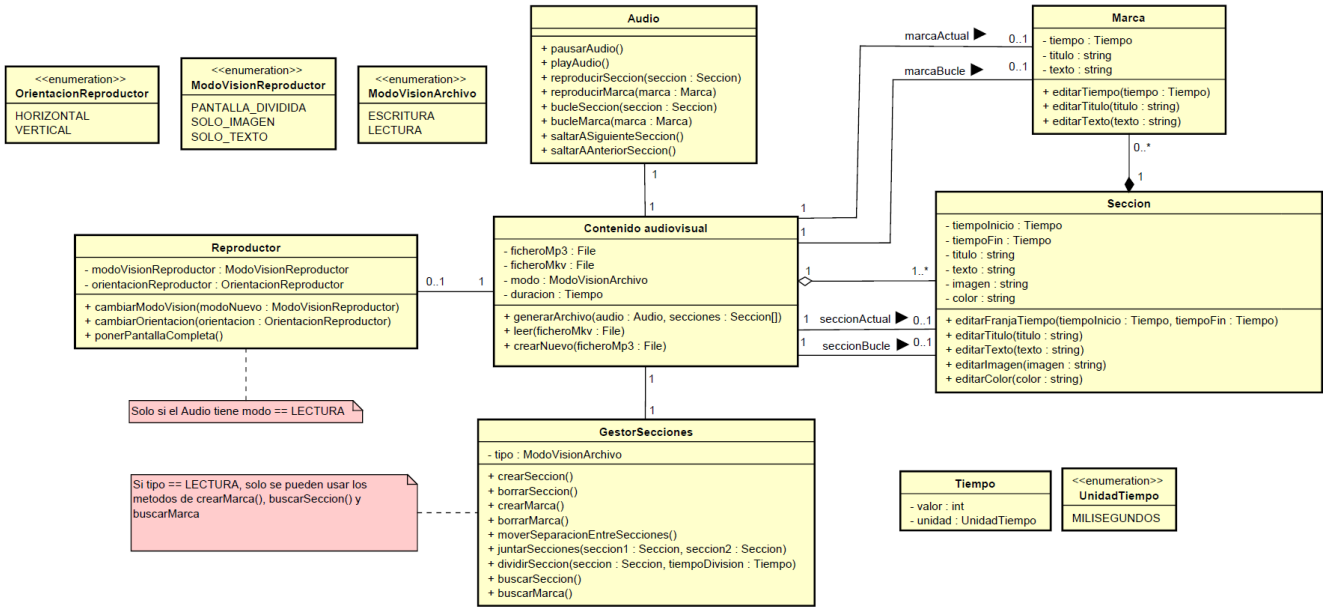

Figura 4.2: Diagrama de modelo de dominio de análisis de la aplicación

A continuación, se describen en detalle las clases que aparecen en el diagrama, así como las relaciones que existen entre ellas:

ContenidoAudiovisual Es la clase central del modelo y representa una agrupación de información asociada a un audio, que además posee las herramientas necesarias para gestionar estos datos. Este contenido puede ser generado a partir de un archivo .mp3 o cargado desde un archivo .mkv. Desde esta clase se tiene acceso a todos los elementos clave que conforman la experiencia de uso.

Audio Encapsula los controles básicos de reproducción del archivo de audio: reproducir, pausar, activar o desactivar el modo bucle, o saltar entre secciones.

Reproductor Disponible únicamente en el modo lectura, permite la visualización del contenido audiovisual de manera interactiva. Incluye herramientas para modificar el estilo de la visualización (pantalla

dividida, sólo mostrar la imagen de la sección, sólo mostrar el texto de la sección), así como la orientación, cuando la pantalla del reproductor se encuentra dividida (división vertical u horizontal). Su objetivo es adaptar la presentación del contenido a las preferencias o necesidades del usuario.

GestorSecciones Administra el espacio de secciones y marcas dentro del audio. Ofrece operaciones que permiten crear, borrar y buscar secciones y marcas dentro de este espacio, así como mover el punto de separación entre dos secciones, fusionar dos secciones, o dividir una sección en dos. Algunas de estas operaciones están disponibles solo en el modo escritura, restringiéndose en el modo lectura para proteger el contenido.

Seccion Representa un segmento del audio definido por un tiempo de inicio y un tiempo de fin. La información que se recopila de él es un título, un texto, una imagen, y un color que lo diferencie del resto de secciones. Sus operaciones permiten modificar estos datos mencionados.

Marca Representa un punto concreto del audio, ubicado en un instante específico. De ella se recoge información como un título y un texto, y presenta operaciones que permiten su modificación . Las marcas ayudan a destacar momentos relevantes dentro de una sección.

Tiempo Clase que encapsula un valor numérico junto con su unidad temporal (como segundos o milisegundos).

Como se puede observar, la clase `ContenidoAudiovisual` constituye el núcleo del modelo, estableciendo relaciones directas con el resto de las clases. Está asociada a un `Audio`, que proporciona los controles de reproducción; a uno o ningún `Reproductor`, presente únicamente en el modo lectura; y a un `GestorSecciones`, responsable de gestionar las secciones y marcas del espacio del audio.

Asimismo, puede contener una o varias `Seccion`, cada una con sus propias marcas, así como referencias a una Sección seleccionada y una Marca seleccionada, que indican los elementos actualmente activos. También puede haber una Sección en bucle y una Marca en bucle, utilizadas cuando se reproduce un fragmento o instante de audio de forma repetida.

Capítulo 5

Diseño

En este capítulo se describe el diseño de la aplicación desarrollada, abordando tanto su estructura interna como su aspecto visual. El objetivo de esta etapa es definir, organizar y representar de forma clara cómo se construye el sistema antes de su implementación completa.

Para ello, se presenta en primer lugar la arquitectura general de la aplicación, basada en componentes reutilizables y servicios, siguiendo el enfoque modular propuesto por el framework Angular. A continuación, se detalla la estructura del proyecto, mostrando los paquetes principales, las clases contenidas en cada uno y las relaciones existentes entre ellos.

Finalmente, se presenta el diseño de la interfaz de usuario, detallando su estructura dividida en secciones funcionales y su relación directa con los distintos componentes de la aplicación.

5.1. Arquitectura Lógica

Tal como se expuso en capítulos anteriores, el desarrollo de la aplicación se llevó a cabo utilizando Angular como framework para el frontend. Esta elección determina en gran medida la arquitectura del proyecto, ya que Angular proporciona una estructura bien definida basada en componentes, servicios, módulos y enrutamiento. Gracias a este enfoque, es posible desarrollar aplicaciones complejas de forma organizada, escalable y mantenible. A continuación, se describen los elementos clave que definen la arquitectura de Angular [46], fundamentales para comprender el diseño y desarrollo de la aplicación presentada en este trabajo.

Componentes

Los componentes son la piedra angular de cualquier aplicación Angular. Representan unidades inde-

pendientes de la interfaz de usuario que encapsulan tanto la lógica como la presentación visual. Cada componente está formado por:

- Una clase TypeScript, donde se define el comportamiento y el estado del componente.
- Una plantilla HTML, que describe la estructura visual del componente.
- Un archivo de estilos CSS o SCSS, que define su apariencia.

Angular gestiona automáticamente el ciclo de vida de los componentes, permitiendo ejecutar lógica en momentos concretos mediante métodos como ngOnInit() (cuando se inicializa el componente) o ngOnDestroy() (cuando se elimina). Esta separación clara entre la lógica, la vista y el estilo permite una mayor modularidad y facilita el mantenimiento del código.

Los componentes pueden anidarse unos dentro de otros y comunicarse mediante inputs y outputs, lo cual facilita la creación de interfaces complejas a partir de unidades pequeñas y reutilizables.

Servicios e Inyección de Dependencias

Los servicios son clases dedicadas a manejar lógica de negocio, tareas reutilizables o datos compartidos entre múltiples componentes.

Angular ofrece un sistema incorporado de inyección de dependencias (DI), que permite a los componentes o a otros servicios solicitar instancias de estos servicios sin necesidad de crearlos manualmente. Esto se consigue utilizando el decorador @Injectable().

La inyección de dependencias favorece:

- La reutilización del código (un mismo servicio puede ser usado por múltiples componentes).
- La modularidad y la independencia entre capas.
- La facilidad de pruebas, al poder inyectar versiones falsas de los servicios (mocks) en los tests.

Módulos

Los módulos (NgModules) son estructuras que permiten agrupar componentes, servicios, directivas y otros recursos relacionados bajo una misma unidad lógica. Cada aplicación Angular tiene al menos un módulo raíz, normalmente llamado AppModule, que define los elementos principales de la aplicación y sirve como punto de entrada.

Además del módulo principal, Angular permite dividir la aplicación en módulos funcionales o de características, lo que facilita el mantenimiento, la carga perezosa de funcionalidades (*lazy loading*) y el escalado de la aplicación.

Con la llegada de Angular 14, se introdujo el concepto de componentes independientes (*standalone components*), que permite prescindir de los módulos al definir componentes de forma autónoma. Esta característica reduce la complejidad inicial y mejora la portabilidad de componentes entre proyectos.

Enrutamiento

Angular proporciona un sistema de enrutamiento que permite gestionar la navegación entre diferentes vistas o páginas dentro de una misma aplicación. Gracias a este mecanismo, es posible construir aplicaciones de una sola página (SPA), donde el contenido se actualiza dinámicamente sin recargar la página completa. El enrutamiento se define mediante una configuración de rutas, que asocia cada URL con un componente específico.

5.2. Organización del proyecto

En este apartado ahondaremos en cómo se estructura la aplicación desarrollada desde un punto de vista de diseño. Para representar esta estructura jerárquica de carpetas, las relaciones que existen entre ellas y las clases más importantes en cada una, se emplean diagramas de paquetes.

En el Lenguaje de Modelado Unificado (UML), un diagrama de paquetes es un tipo de diagrama estructural que agrupa elementos relacionados (como clases, interfaces o subpaquetes) en contenedores visuales denominados paquetes, representados como carpetas. Su propósito es ofrecer una visión de alto nivel de la organización del sistema, mostrando la jerarquía de módulos y las dependencias entre ellos [47].

5.2.1. Diagrama de paquetes general de la aplicación

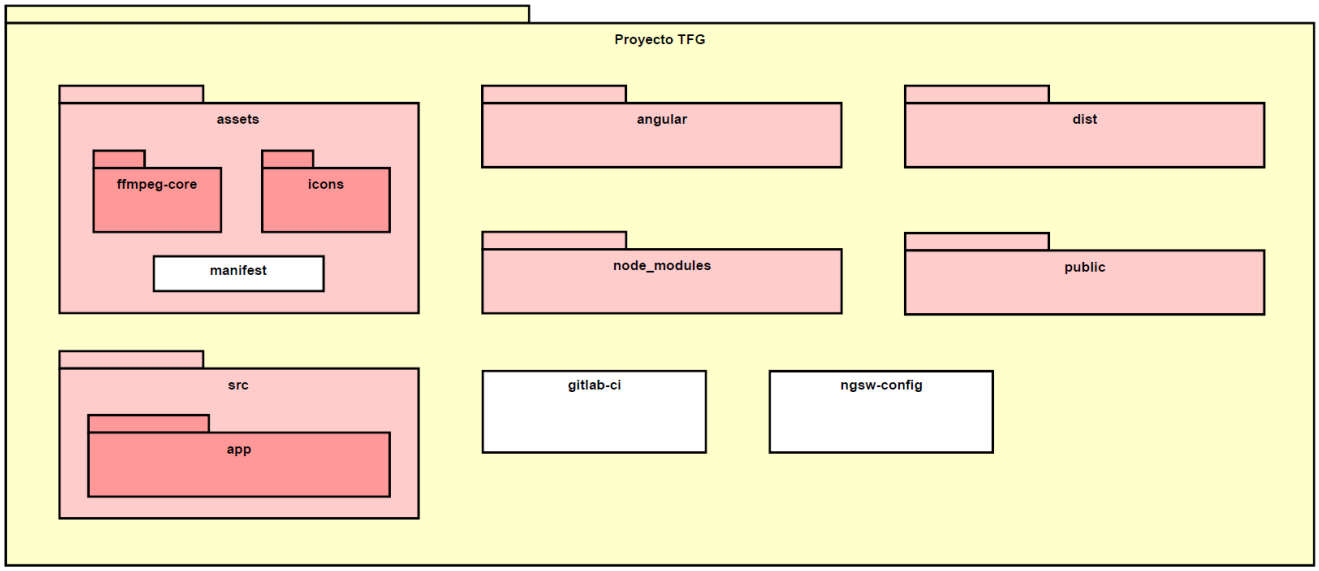

Figura 5.1: Diagrama de paquetes del proyecto Angular

En la Figura 5.1, se muestra el diagrama de paquetes que representa la organización en carpetas del proyecto. Esta estructura sigue el modelo estándar de las aplicaciones desarrolladas con Angular, aunque ha sido adaptada en ciertos aspectos para ajustarse a las necesidades concretas de la aplicación. A continuación, se describe el propósito de las carpetas y archivos principales:

`/src/` Contiene el código fuente de la aplicación. Es la carpeta principal desde la cual se organiza todo el desarrollo de la aplicación Angular.

`/src/app` Contiene los componentes, servicios, modelos y demás elementos que forman la lógica y estructura funcional de la aplicación. Debido a su relevancia, se describe con mayor detalle en el siguiente apartado.

`/assets/` Directorio destinado a almacenar recursos estáticos que serán utilizados por la aplicación, como imágenes, archivos de configuración, librerías externas o fuentes.

`/assets/ffmpeg-core` Contiene los archivos necesarios para ejecutar FFmpeg en el navegador, incluyendo los WASM y los workers. Esto permite acceder a sus funcionalidades sin conexión, ya que no deben ser descargados externamente.

/assets/icons Contiene los iconos utilizados en la interfaz de usuario, incluyendo los que se emplean en la configuración del manifiesto de la PWA y otros elementos gráficos.

/assets/manifest Incluye archivos relacionados con la configuración de la aplicación como PWA (Progressive Web App), como el manifest.webmanifest, que define propiedades como el nombre, icono y comportamiento en dispositivos móviles.

/angular/ Directorio generado por Angular que puede contener archivos auxiliares relacionados con la compilación o configuración del entorno de trabajo. No suele modificarse directamente por el desarrollador.

/dist/ Carpeta de salida que contiene la versión final compilada y optimizada de la aplicación lista para su despliegue. Es generada automáticamente tras ejecutar el comando de construcción (ng build).

/node_modules/ Almacena todas las dependencias instaladas mediante NPM (Node Package Manager). Incluye tanto bibliotecas propias de Angular instaladas de forma automática, como bibliotecas específicas instaladas manualmente.

/public/ Esta carpeta se utiliza para el despliegue de la aplicación en GitLab Pages. El contenido generado al ejecutar ng build deberá copiarse aquí para que pueda ser servido correctamente. En el siguiente capítulo se detallará con mayor precisión su uso dentro del proceso de publicación.

/gitlab-ci Contiene los archivos de configuración para la integración continua con GitLab CI/CD. Aquí se definen los pasos que debe seguir el pipeline de GitLab para construir, testear o desplegar la aplicación de forma automatizada.

/ngsw-config Archivo de configuración del *service worker* utilizado por Angular para habilitar funcionalidades propias de una PWA, como el almacenamiento en caché de recursos, la disponibilidad offline y la mejora del rendimiento en cargas posteriores.

5.2.2. Diagrama de paquetes de app

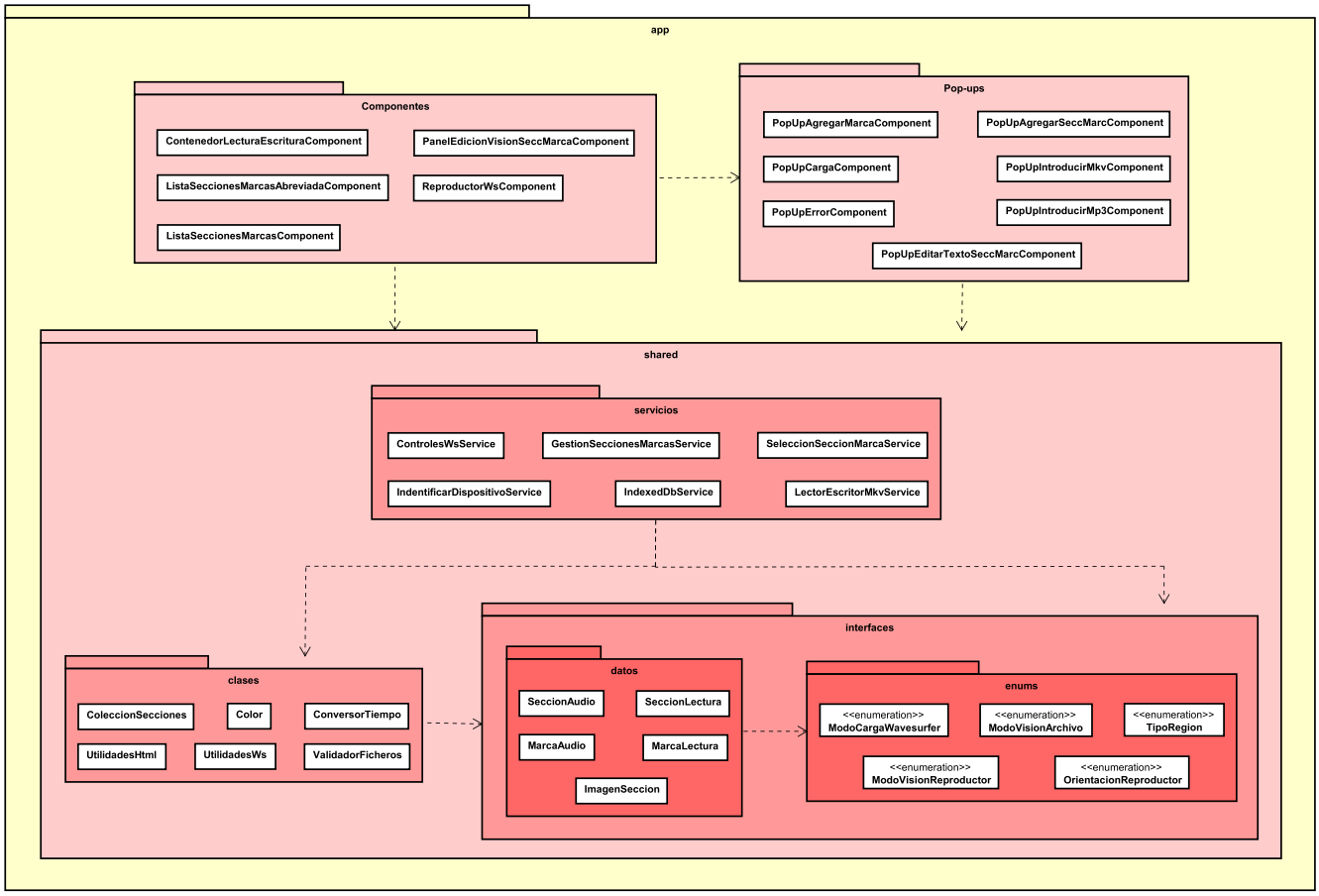

Figura 5.2: Diagrama de paquetes de app en la aplicación

Ahora, pasaremos a describir en específico la estructura del paquete app, que, como se mencionó anteriormente, contiene la lógica principal de la aplicación. A continuación, se detalla para qué sirve cada uno de los paquetes representados en el diagrama de la Figura 5.2, así como las clases más relevantes que se encuentran dentro de ellos. Esta organización refleja una separación clara de responsabilidades y contribuye a mantener un código modular, escalable y fácil de mantener.

Componentes Este paquete contiene los componentes principales en los que se divide la interfaz gráfica de la aplicación.

- **ContenedorLecturaEscrituraComponent**: componente principal que coordina y agrupa al resto de componentes de la interfaz. Sus responsabilidades incluyen:
  - Detectar el modo actual de la aplicación (lectura o escritura) e inicializar los componentes de acuerdo a este.
  - Inicializar los servicios responsables de la compartición de datos entre componentes.

  - Gestionar la introducción de archivos: un archivo MP3 para la creación de un nuevo proyecto o un archivo MKV para cargar un proyecto existente. Esto implica inicializar un objeto `Wavesurfer` con la información del audio y un objeto `ColeccionSecciones` con la información estructurada de secciones y marcas, ambos compartidos mediante los servicios.
  - Permitir la descarga del archivo actualmente en edición, así como establecer su nombre.

- **ReproductorWsComponent**: componente encargado de representar gráficamente el audio, junto con sus secciones y marcas, y proporcionar controles de reproducción. Permite:
  - Visualizar gráficamente el audio mediante `Wavesurfer`, incluyendo interacción con las secciones y marcas (mover intersecciones, añadir o mover marcas, etc.).
  - Controlar el volumen del audio.
  - Saltar a la siguiente o anterior sección.
  - Reproducir o pausar el audio.
  - Modificar el nivel de zoom de la visualización de la onda.
- **ListaSeccionesMarcasComponent**: muestra una lista de secciones y marcas, junto con un resumen de sus datos (título, color y duración), y ofrece controles de reproducción asociados. Permite:
  - Seleccionar una sección o marca.
  - Reproducir una sección o marca en bucle.
  - Reproducir desde el inicio una sección o marca.
  - Ocultar o mostrar la lista de marcas asociadas a una sección.
- **ListaSeccionesMarcasAbreviadaComponent**: utilizado en pantallas con orientación vertical. Muestra los datos de la sección actualmente seleccionada y permite desplegar la lista completa de secciones y marcas para seleccionar otra.
- **PanelEdicionVisionSeccMarcaComponent**: muestra la información completa de la sección o marca seleccionada y ofrece controles para su reproducción (en ambos modos) y edición (en modo escritura). Permite:
  - Reproducir desde el inicio o en bucle la sección o marca seleccionada.
  - Editar los campos de título, texto e imagen (modo escritura).
  - Cambiar el color de la sección (modo escritura).
  - Eliminar la sección o marca (modo escritura).
  - Cambiar el modo de visualización (solo imagen, solo texto o dividido) y su orientación (vertical u horizontal) en modo lectura.
  - Activar el modo de reproducción a pantalla completa, con controles de reproducción de audio simplificados (modo lectura).

Pop-ups Este paquete contiene todos los componentes que actúan como ventanas emergentes y son utilizados por los componentes principales:

- **PopUpAgregarMarcaComponent**: permite añadir una nueva marca en modo lectura, introduciendo su título y texto.
- **PopUpAgregarSeccMarcComponent**: permite elegir entre agregar una nueva sección o una nueva marca.
- **PopUpCargaComponent**: informa al usuario de que la aplicación se encuentra en estado de carga.
- **PopUpErrorComponent**: muestra mensajes de error personalizados.
- **PopUpIntroducirMkvComponent**: permite seleccionar un archivo MKV para cargar su contenido en la aplicación.
- **PopUpIntroducirMp3Component**: permite introducir un archivo MP3 como base para crear un nuevo proyecto.
- **PopUpEditarTextoSeccMarcComponent**: permite editar el texto de una sección o marca.

Shared Este paquete agrupa las estructuras de datos y clases que implementan la lógica de negocio, utilizadas por los componentes. Se divide en tres subpaquetes:

Servicios Contienen la lógica principal de negocio y la gestión de datos compartidos entre componentes.

- **ControlesWsService**: proporciona una interfaz común para interactuar con el objeto `Wavesurfer`, incluyendo:
  - Métodos para reproducir, reproducir en bucle, saltar entre secciones, hacer zoom, cambiar volumen, y consultar información como el tiempo actual de reproducción.
  - Inicialización y configuración del objeto `Wavesurfer`, adaptándolo para permitir la creación y edición de secciones y marcas, así como la configuración del zoom, scroll y comportamiento de reproducción.
- **GestionSeccionesMarcasService**: ofrece métodos para crear, eliminar y modificar secciones y marcas del objeto `ColeccionSecciones`.
- **SeleccionSeccionMarcaService**: gestiona la selección activa de una sección o marca, permitiendo:
  - Seleccionar o deseleccionar elementos del objeto `ColeccionSecciones`.

  - Consultar el valor de la seccion o marca seleccionada actualmente.
  - Detectar automáticamente la sección o marca en la que se encuentra el audio actualmente y actualizar la selección (utilizado en modo lectura).

- **IdentificarDispositivoService**: permite detectar el tipo de dispositivo (móvil, tablet o sobremesa) y su orientación de pantalla (vertical u horizontal).
- **IndexedDbService**: encapsula el acceso a la base de datos `IndexedDB`, permitiendo almacenar, recuperar y eliminar archivos.
- **LectorEscritorMkvService**: permite:
  - Leer los datos almacenados en un archivo MKV previamente generado por la aplicación.
  - Generar un nuevo archivo MKV que contenga el audio y la colección de secciones y marcas, incluyendo sus títulos, textos e imágenes.

Clases

- **ColeccionSecciones**: encapsula la lógica de gestión de un conjunto de secciones y marcas a lo largo del eje temporal del audio, incluyendo creación, edición, eliminación y búsqueda de elementos.
- **Color**: clase estática con utilidades para generar colores aleatorios y convertir entre formatos de color.
- **ConversorTiempo**: clase estática que proporciona métodos para convertir tiempos entre diferentes formatos.
- **UtilidadesHtml**: clase estática con funciones para modificar dinámicamente elementos HTML desde el código.
- **UtilidadesWs**: clase estática con funciones relacionadas con el objeto `Wavesurfer`.
- **ValidadorFicheros**: clase estática que permite verificar la validez de un objeto `File`, útil para detectar si su contenido ha sido eliminado por el navegador (por ejemplo, en dispositivos móviles con poco espacio).

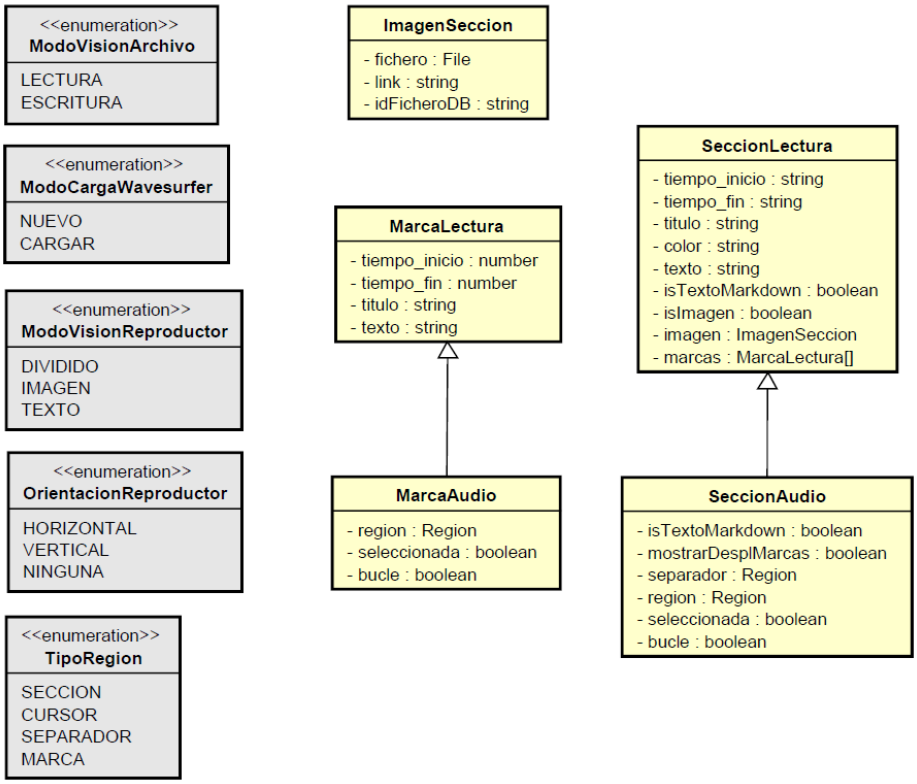

Figura 5.3: Diagrama de clases del paquete interfaces de la aplicación

Interfaces Este paquete agrupa las interfaces y enumeraciones utilizadas para estructurar los datos. En la Figura 5.3 se puede observar qué atributos contiene cada una.

- **SeccionLectura**: representa los datos principales de una sección tal como se almacenan en un archivo MKV, incluyendo inicio, fin, título, texto, imagen, color, si el texto está en Markdown, y la lista de marcas asociadas.
- **SeccionAudio**: amplía a `SeccionLectura` con propiedades adicionales necesarias durante la edición y visualización, como la referencia a su `Region` en Wavesurfer y flags de estado (seleccionada, en bucle, etc.).
- **MarcaLectura**: representa los datos de una marca en un archivo MKV, incluyendo su tiempo, título y texto.
- **MarcaAudio**: hereda de `MarcaLectura` y, al igual que `SeccionAudio`, añade propiedades útiles para la edición y visualización.
- **ImagenSeccion**: representa los datos asociados a una imagen, incluyendo el archivo original, el enlace para su visualización en la página y su identificador en `IndexedDB`, que permite recuperarla posteriormente desde la base de datos.

Enums Este paquete incluye las enumeraciones utilizadas para definir opciones o estados dentro de la aplicación.

- **ModoCargaWavesurfer**: indica si el objeto `Wavesurfer` se debe crear desde cero (`NUEVO`) o cargando información preexistente (`CARGAR`), incluyendo la información de secciones y marcas extraída de un archivo cargado.
- **ModoVisionArchivo**: define el modo actual de la aplicación: `LECTURA` o `ESCRITURA`, cada uno con diferentes funcionalidades y restricciones.
- **TipoRegion**: clasifica los distintos tipos de regiones gráficas en Wavesurfer, teniendo cada una de ellas propiedades especiales en su configuración (si puede o no ser arrastrada, su aspecto visual, su longitud, etc):
  - `SECCION` y `MARCA`, asociadas a los contenidos.
  - `CURSOR`: marca que se utiliza para navegar dentro de la onda de audio e indicar la zona donde se desea agregar una nueva marca o sección.
  - `SEPARADOR`: marca que simboliza la separación entre dos secciones, y que puede moverse para ajustar la separación entre ellas.
- **ModoVisionReproductor**: define cómo se presenta el contenido en la pantalla del reproductor en modo lectura:
  - `DIVIDIDO`: se muestra tanto la imagen como el texto de la sección en reproducción.
  - `IMAGEN`: se muestra solo la imagen de la sección.
  - `TEXTO`: se muestra solo texto de la sección.
- **OrientacionReproductor**:define los estados en los que puede estar la orientación del reproductor en modo lectura. Esta puede ser `HORIZONTAL`, mostrándose la imagen arriba y el texto abajo, `VERTICAL`, mostrándose el texto a la izquierda y la imagen a la derecha, o `NINGUNO`, en caso de que el modo de visión no sea `DIVIDIDO`.

5.3. Interfaz de usuario

El diseño de la interfaz de usuario fue un paso previo fundamental para la posterior división de la aplicación en componentes, ya que permitió visualizar y delimitar las distintas áreas funcionales que la componen. Esta interfaz es compartida por ambos modos de funcionamiento de la aplicación (lectura y escritura), ya que presentan una estructura común y comparten la mayoría de funcionalidades, diferenciándose únicamente en ciertos comportamientos y controles específicos. En la Figuras 5.4

(interfaz en horizontal) y 5.5 (interfaz en vertical), se muestra el boceto general de esta interfaz, en el que se han utilizado diferentes colores para representar cada uno de los componentes identificados.

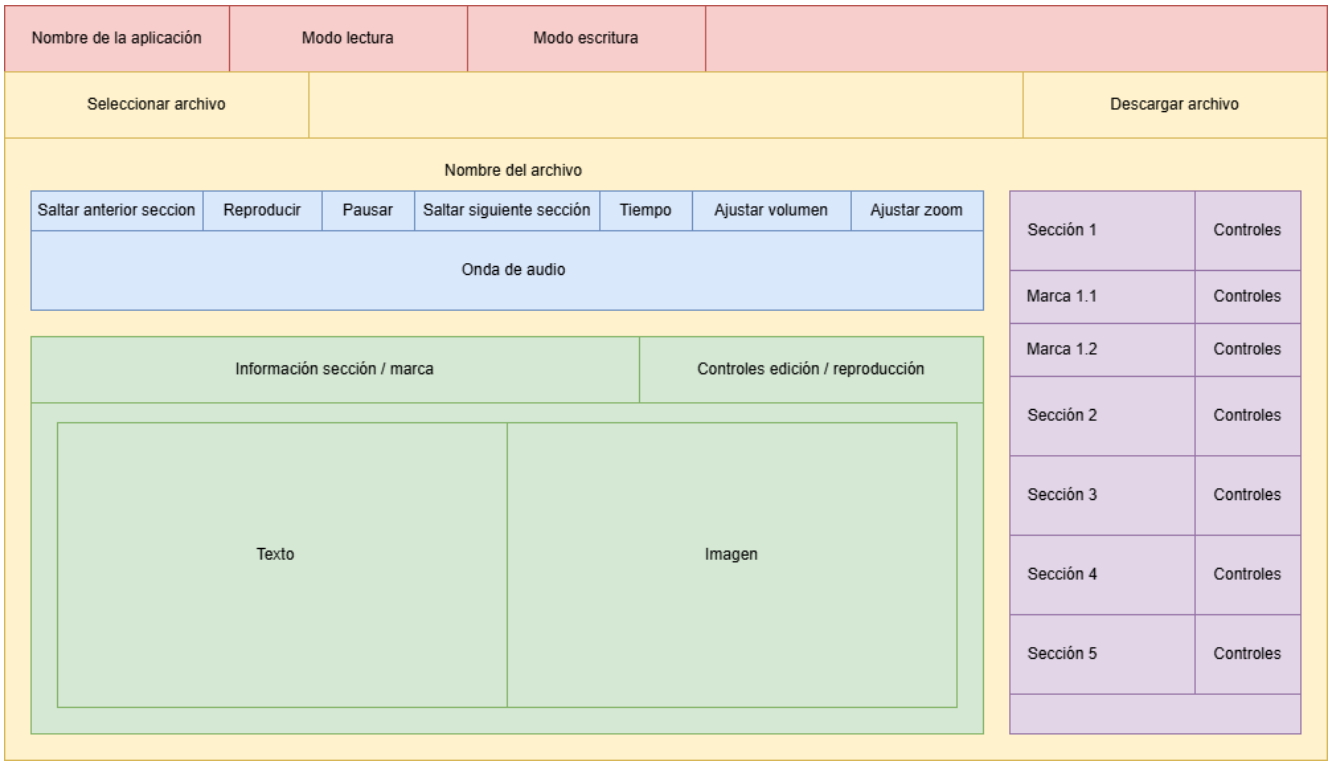

Figura 5.4: Boceto de la interfaz (horizontal) de la aplicación separada por componentes

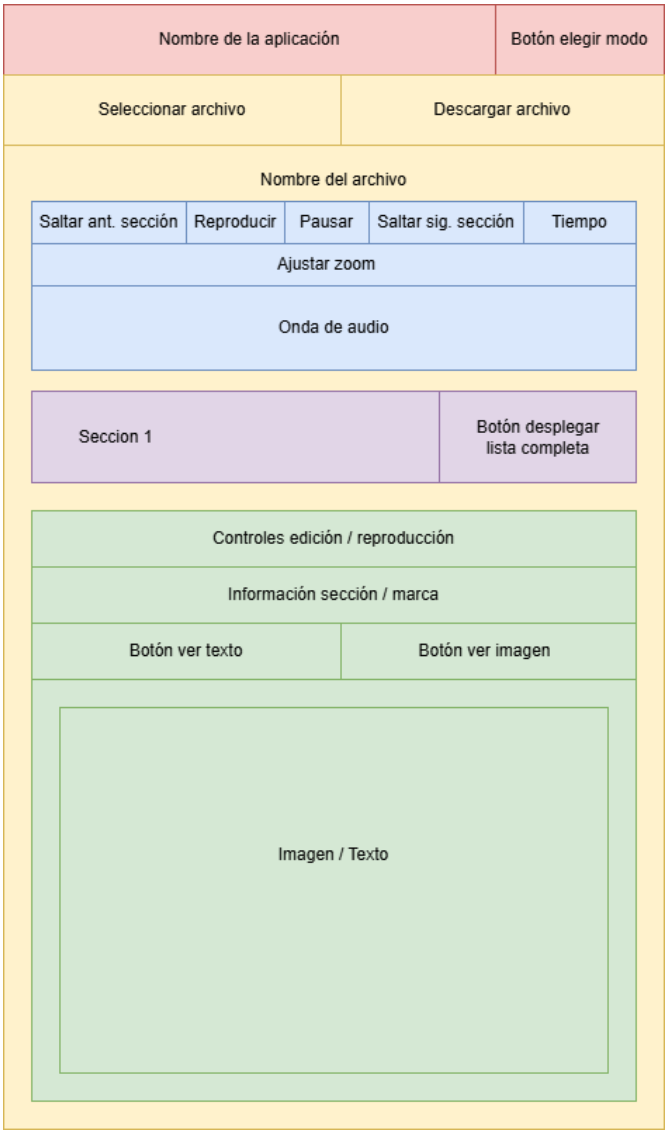

Figura 5.5: Boceto de la interfaz (vertical) de la aplicación separada por componentes

AppComponent (color rojo) Corresponde a la barra de navegación superior, visible de forma permanente en la aplicación. Incluye el logotipo y permite navegar entre sus dos modos, lectura y escritura.

ContenedorLecturaEscrituraComponent (color amarillo) Ocupa el área principal bajo la barra de navegación, actuando como contenedor del resto de componentes de la interfaz. En su parte superior se encuentra una barra que permite seleccionar un archivo (MP3 o MKV) para cargar un proyecto o comenzar uno nuevo desde cero. También incluye un botón para exportar el proyecto actual en formato MKV, con la posibilidad de darle nombre en modo escritura.

ReproductorWsComponent (color azul) Muestra la representación visual de la onda del audio, junto con los controles de reproducción asociados.

PanelEdicionVisionSeccMarcaComponent (color verde) Su apariencia varía en función del modo (lectura o escritura). La interfaz de este componente se divide en dos partes:

- Una barra superior, que presenta la información de la sección o marca seleccionada actualmente y una serie de controles que varían según el modo. En modo escritura, estos incluyen reproducción desde el inicio, reproducción en bucle y eliminación de la sección o marca. En modo lectura, los controles permiten cambiar el modo de visualización y orientación de los datos del panel, además de ofrecer una opción para visualizarlo en pantalla completa.
- Dos recuadros destinados a mostrar el texto y/o la imagen asociados a la sección o marca seleccionada. En el modo escritura, estos elementos son editables para permitir la modificación del contenido.

ListaSeccionesMarcasComponent y ListaSeccionesMarcasAbreviadaComponent (color morado) Se encargan de mostrar la lista de secciones y marcas. El componente `ListaSeccionesMarcasComponent` presenta una estructura jerárquica donde cada sección puede contener varias marcas, todas con controles para su reproducción desde el inicio o en bucle. Al pulsar en una sección o marca de la lista, esta es seleccionada. En la disposición vertical, se utiliza `ListaSeccionesMarcasAbreviadaComponent`, que muestra únicamente la información de la sección seleccionada en un recuadro, junto con una opción para desplegar la lista completa para que ocupe toda la pantalla.

A la hora de diseñar la interfaz, se buscaba que esta fuera compacta (es decir, que pudiera visualizarse en pantalla sin necesidad de hacer scroll), intuitiva, fácil de usar y de aprender, y cómoda para el usuario. Además, se prestó especial atención al uso del color, procurando evitar una apariencia recargada. Para ello, resultó de gran utilidad la herramienta web Coolors [48], que permitió seleccionar una paleta de colores armoniosa y adecuada para la aplicación.

En las Figuras 5.6 y 5.7 se muestran capturas de la interfaz final en sus modos de escritura y lectura, respectivamente, en orientación horizontal. Por otro lado, en las Figuras 5.8 y 5.9 se presentan ambos modos de la aplicación en su versión vertical, más orientada a tablets y dispositivos móviles.

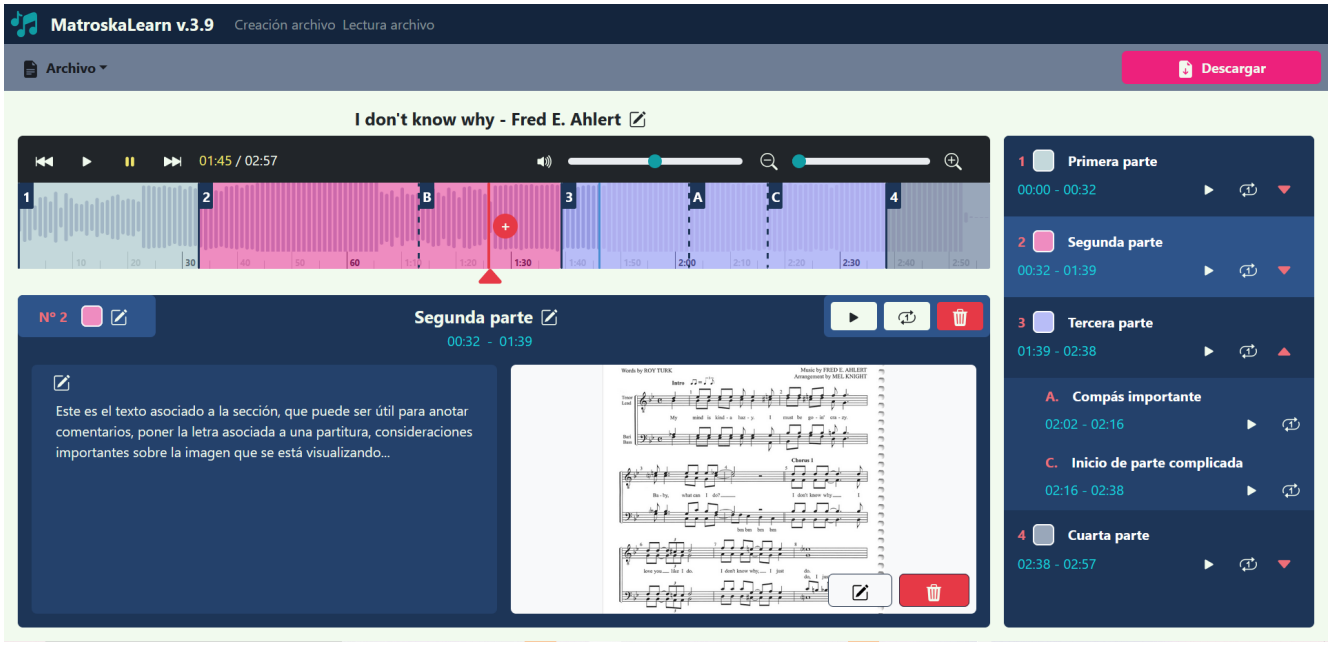

Figura 5.6: Interfaz de escritura de la aplicación (horizontal)

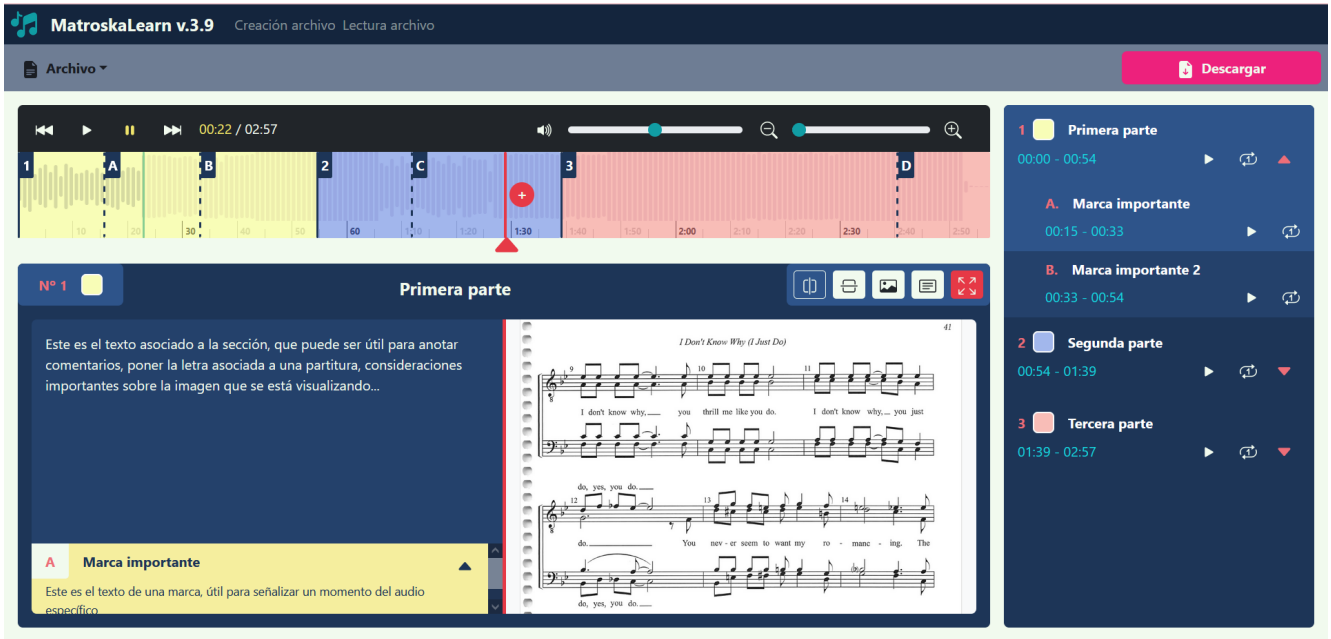

Figura 5.7: Interfaz de lectura de la aplicación (horizontal)

Figura 5.8: Interfaz de escritura de la aplicación (vertical)

Figura 5.9: Interfaz de lectura de la aplicación (vertical)

Capítulo 6

Implementación y Despliegue

Este capítulo describe los aspectos más relevantes relacionados con la implementación de la aplicación. En primer lugar, se detallan las principales bibliotecas empleadas para su desarrollo, que han permitido integrar funcionalidades clave como la visualización y edición de audio, así como la generación del archivo final. A continuación, se explica cómo se ha incorporado la funcionalidad propia de una PWA (Progressive Web App), lo que permite que la aplicación pueda instalarse y usarse sin conexión. Posteriormente, se expone el uso de Indexed DB para la persistencia local de las imágenes introducidas por el usuario, garantizando su disponibilidad entre sesiones. Finalmente, se describe el proceso de despliegue de la aplicación, que durante la fase de desarrollo se realizó en GitLab Pages, y para su publicación final se llevó a cabo en GitHub Pages, permitiendo acceder a ella de forma pública a través de un navegador web.

6.1. Bibliotecas principales

A continuación, se presentan las principales bibliotecas utilizadas en la implementación de la aplicación desarrollada, detallando los aspectos más relevantes de su uso.

6.1.1. Ffmpeg.wasm

La biblioteca FFmpeg fue utilizada para la introducción y recuperación de datos en archivos en formato .mkv. Este tipo de archivo permite almacenar múltiples tipos de datos de manera estructurada mediante el uso de streams.

Un stream (o flujo de datos) representa una pista de contenido dentro del archivo. Cada stream puede contener un tipo específico de información (como audio, vídeo o subtítulos), y los archivos MKV permiten incluir varios streams del mismo o distinto tipo. A cada stream se le asigna un número, comenzando por

el 0, y se puede añadir tantos como se desee.

Los tipos de stream más comunes que se pueden almacenar en un archivo MKV son:

- **Stream de audio**: contiene el contenido sonoro del archivo, como música, voz o efectos.
- **Stream de vídeo**: aunque su uso más común es para contenido visual en movimiento, también puede utilizarse para almacenar imágenes estáticas, codificadas como fotogramas.
- **Stream de subtítulos**: almacena texto sincronizado con el contenido audiovisual, como transcripciones o traducciones. Aunque no se usa en este proyecto, es otro tipo de stream compatible con MKV.

Además de los streams, los archivos MKV permiten almacenar metadatos: información textual adicional estructurada como pares `clave = valor`. Estos metadatos no forman parte de un stream concreto, pero se guardan dentro del archivo y pueden ser leídos y escritos mediante FFmpeg.

Organización de los datos en el archivo MKV

En el caso de este proyecto, se quiere almacenar en un archivo .mkv tanto el audio del proyecto como los datos asociados a un conjunto de secciones definidas sobre ese audio. Estos datos incluyen:

- El archivo MP3 correspondiente al audio principal.
- La información de cada sección (tiempos de inicio y fin, título, texto, imagen asociada, etc.).
- Las marcas contenidas dentro de cada sección (tiempos de inicio y fin, titulos, textos, etc.).

La forma de organizar esta información en el contenido del archivo MKV será la siguiente:

- **Audio del proyecto**: se asigna al stream 0, utilizando un stream de tipo audio. Este stream contendrá el archivo .mp3 correspondiente al audio principal.
- **Imágenes de las secciones**: se utiliza un stream de tipo video por cada imagen. Cada imagen se codifica como un pequeño vídeo estático (normalmente un único fotograma), y se asigna un número de stream progresivo a cada una: el stream 1 para la imagen de la primera sección, el stream 2 para la segunda, y así sucesivamente. Este orden permite recuperar las imágenes en el mismo orden en que fueron añadidas, manteniendo la correspondencia con las secciones.

- **Datos de las secciones y marcas (sin incluir imágenes)**: esta información se almacena en los metadatos del archivo, utilizando la estructura `clave = valor`. Para cada sección, la clave sigue el formato `SECCION_n`, donde `n` es el número de sección, y el valor es una cadena en formato JSON que contiene toda la información asociada a esa sección (incluyendo sus marcas). Este JSON se genera utilizando la función `JSON.stringify()` de TypeScript, a partir de la interfaz de datos de la sección, y puede volver a transformarse en un objeto utilizando `JSON.parse()`. Esta estrategia permite mantener la independencia entre los datos del archivo y la implementación concreta del código, facilitando modificaciones posteriores en las interfaces sin necesidad de alterar el formato del archivo.

También se añade un metadato adicional llamado `NUM_SECCIONES`, que indica la cantidad total de secciones almacenadas.

Comando FFmpeg utilizado para generar el archivo

El comando utilizado para generar un archivo MKV que contenga todos los datos del proyecto tiene la siguiente forma:

```
ffmpeg \
  -i audio.mp3 \
  -i imagen(número imagen).(extensión Imagen) \
  -map 0 \
  -map (num_imagen) \
  -metadata NUM_SECCIONES=(número de secciones) \
  -metadata SECCION_(número seccion)='(JSON con datos de la sección)' \
  -c copy \
  archivo_salida.mkv
```

- **i**: se utiliza para introducir un archivo de entrada. Puede repetirse varias veces para añadir múltiples fuentes (audio, imágenes, etc.).
- **map**: asigna cada archivo de entrada a un stream concreto en el archivo de salida. Por ejemplo, -map 0 indica que el primer archivo de entrada se convertirá en el stream 0 del archivo resultante.
- **metadata**: permite añadir información textual en forma de metadatos al archivo .mkv, siguiendo el formato clave = valor.
- **c copy**: indica que los datos de los streams deben copiarse directamente sin recodificarse. Esto permite preservar la calidad original del contenido y agilizar el proceso de generación, ya que se evita cualquier operación de compresión o conversión adicional.

Wasm

Como se ha mencionado en apartados anteriores, la biblioteca ffmpeg utilizada en esta aplicación se basa en WebAssembly (WASM), lo que permite ejecutar procesamiento multimedia directamente en el navegador sin necesidad de servidores externos. Para asegurar que la aplicación funcione también en modo offline (sin depender de descargas desde servidores de terceros), es necesario incluir en el proyecto los archivos esenciales que constituyen el núcleo de ffmpeg.wasm.

Estos archivos se almacenan en la carpeta assets/ffmpeg-core y cumplen funciones específicas en el proceso de carga y ejecución de la biblioteca:

- **ffmpeg-core.js**: archivo principal que contiene la lógica necesaria para inicializar y gestionar la instancia de ffmpeg en el navegador.
- **ffmpeg-core.wasm**: archivo compilado en WebAssembly, que realiza el procesamiento intensivo de datos de forma eficiente.
- **worker.js**: script encargado de ejecutar ffmpeg en un hilo separado (Web Worker), mejorando el rendimiento y evitando bloqueos en la interfaz principal.
- **const.js, errors.js**: archivos auxiliares que definen constantes y manejan errores relacionados con la ejecución de ffmpeg.

Estos recursos se cargan dinámicamente en tiempo de ejecución mediante rutas relativas, lo que permite que el entorno WASM de ffmpeg se configure correctamente y funcione de forma integrada dentro de la aplicación Angular, incluso sin conexión a internet.

6.1.2. Wavesurfer

Junto con la biblioteca FFmpeg, Wavesurfer.js ha sido una de las más empleadas en la implementación de esta aplicación, proporcionando las herramientas necesarias para visualizar y manipular gráficamente una onda de audio.

Esta biblioteca cuenta con diversos plugins que amplían su funcionalidad básica. En este proyecto se han utilizado principalmente los siguientes:

TimeLine Plugin Este plugin permite mostrar una línea de tiempo debajo de la onda de audio, dividiéndola en intervalos regulares configurables. Es una herramienta esencial para facilitar la localización precisa de puntos dentro del audio, mejorando significativamente la navegación y la comprensión visual del mismo.

Regions Plugin Este plugin es el más relevante para la aplicación, ya que permite crear y manipular regiones sobre la onda de audio. Dichas regiones representan gráficamente cada una de las secciones y marcas definidas por el usuario, así como el cursor selector que permite elegir puntos específicos en la pista de audio. Cada objeto Region posee una serie de atributos asociados, así como métodos que permiten modificarlos. A continuación, se describen los más relevantes dentro de la aplicación:

- **color**: define el color de la región. En el caso de las marcas y los separadores entre secciones, se utiliza un tono azul oscuro, diferenciados visualmente mediante estilos de línea distintos. Las secciones reciben por defecto un color aleatorio, que puede ser modificado por el usuario en el modo de edición de la aplicación. La región correspondiente al cursor selector se muestra en color rojo.
- **content**: representa el contenido textual o HTML asociado a la región. Por defecto, se añade en la parte izquierda de esta, y suele ser un texto descriptivo que permite identificarla del resto. En esta aplicación, las regiones que representan a secciones, contienen un número identificativo situado en su lado izquierdo, comenzando desde el 1. La región correspondiente al cursor selector tiene un comportamiento especial y su contenido alterna entre dos variantes que dependen de su estado:
  1. Cuando el cursor está siendo arrastrado sobre la onda de audio, muestra una etiqueta con el instante de tiempo actual del cursor.
  2. Cuando está quieto, muestra un botón que permite añadir una nueva sección o marca en esa posición.
- **contentEditable**: indica si el contenido textual de la región puede editarse directamente. En este proyecto, el valor es siempre falso, ya que el contenido no se modifica desde la interfaz de usuario.
- **drag**: determina si la región puede ser arrastrada a lo largo de la onda de audio. Las regiones asociadas a secciones y marcas permanecen fijas, mientras que las regiones que actúan como separadores entre secciones y el cursor selector, sí se pueden mover.
- **start**: marca el instante de tiempo del audio en el que comienza la región. En el caso de las marcas, este valor coincide con el de finalización, ya que representan un punto específico en lugar de un intervalo.
- **end**: indica el instante en que finaliza la región dentro del audio.
- **id**: es el identificador interno de la región. Además de permitir comparaciones entre regiones, también puede usarse para aplicar estilos específicos mediante reglas CSS, afectando a todas las regiones que compartan dicho identificador.

6.2. Funcionalidad PWA

Para convertir la aplicación en una Progressive Web App (PWA), se utilizó el soporte oficial proporcionado por Angular mediante el paquete @angular/pwa, instalado a través del comando:

```
ng add @angular/pwa
```

La ejecución de este comando genera automáticamente dos archivos clave en el proyecto Angular:

- **manifest.webmanifest**: contiene metadatos sobre la aplicación, como su nombre, iconos de aplicación, colores de tema, orientación, etc. Estos datos permiten que la aplicación pueda instalarse en el dispositivo, lo que significa que puede añadirse a la pantalla de inicio (en dispositivos móviles o escritorios compatibles) y ejecutarse como una ventana independiente, sin interfaz de navegador.
- **ngsw-config.json**: define la configuración del Service Worker, un script que se ejecuta en segundo plano en el navegador. Su función principal es gestionar la caché de los recursos de la aplicación, actuando como intermediario entre esta y la red. Esto permite mejorar el rendimiento en sesiones posteriores, así como habilitar el funcionamiento sin conexión.

En este caso, no fue necesario modificar el contenido del manifest.webmanifest, ya que su configuración por defecto resultó adecuada. Sin embargo, sí se adaptó el contenido de `ngsw-config.json` para asegurar el correcto almacenamiento en caché de los recursos necesarios.

La configuración del Service Worker mendiante el fichero `ngsw-config.json` tiene un impacto directo en cómo se descargan, almacenan y actualizan los recursos de la aplicación, y se organiza de la siguiente manera:

- En el grupo **app** se incluyen los archivos esenciales para el funcionamiento de la aplicación: el archivo principal `index.html`, los scripts y hojas de estilo generados por Angular (.js y .css), el manifiesto y, de forma destacada, los archivos necesarios para ejecutar FFmpeg de forma local en el navegador (`ffmpeg-core.js`, `ffmpeg-core.wasm` y `worker.js`, ubicados en el directorio /ffmpeg-core/). Al establecer el modo de instalación como “prefetch”, estos recursos se descargan de forma anticipada durante la instalación inicial del Service Worker, asegurando su disponibilidad incluso en ausencia de conexión.
- En el grupo **assets** se incluyen imágenes, iconos y otros recursos multimedia utilizados en la interfaz. Se configura con “installMode”: “lazy”, lo que significa que estos archivos se descargan únicamente cuando son solicitados por la aplicación (por ejemplo, al ser mostrados en pantalla). Además, “updateMode”: “prefetch” permite que nuevas versiones de estos recursos se descarguen en segundo plano para ser utilizadas en la siguiente sesión.

Aunque en este caso la aplicación es puramente frontend y no realiza peticiones a servidores externos, la configuración del Service Worker sigue siendo fundamental. Gracias al almacenamiento en caché gestionado a través del archivo `ngsw-config.json`, todos los recursos críticos que requiere la aplicación para ejecutarse (como el archivo principal `index.html`, los scripts y estilos generados por Angular, los iconos y los archivos necesarios para FFmpeg) son descargados y almacenados localmente en la primera carga de la aplicación con conexión a internet.

Esto permite que, una vez descargados por completo, dichos recursos permanezcan accesibles desde la caché del navegador incluso si posteriormente no se dispone de conexión. De este modo, si el usuario abre la aplicación en otro momento sin conexión a internet, esta podrá ejecutarse de forma normal: se mostrará la interfaz, funcionarán las funcionalidades implementadas en el cliente, y se podrán utilizar los módulos cargados previamente, como FFmpeg. El navegador entregará directamente desde la caché todos los archivos necesarios, sin intentar acceder a la red.

En resumen, la aplicación puede abrirse y utilizarse sin conexión tras haber sido visitada al menos una vez con conexión, proporcionando una experiencia consistente en cualquier circunstancia, incluso en entornos con conectividad limitada o intermitente.

6.3. Uso de Indexed DB para la persistencia de imágenes

El uso de IndexedDB surgió como una necesidad técnica para resolver un problema específico que se manifestaba únicamente en navegadores móviles. En la aplicación desarrollada, las imágenes asociadas a las secciones de un proyecto creado o leído por el usuario se almacenan en variables de tipo File durante la visualización o edición del mismo. A partir de estos objetos File se generan enlaces temporales (ObjectURL) que permiten visualizar las imágenes directamente en la interfaz.

El problema aparecía cuando el usuario, tras haber cargado las imágenes y generado los enlaces correspondientes, abría otra aplicación en su dispositivo móvil que requería un uso intensivo de memoria (como, por ejemplo, un videojuego). Al regresar a la aplicación web, las variables File que contenían las imágenes dejaban de ser válidas y su contenido ya no podía visualizarse. Esto ocurría porque algunos navegadores móviles, ante una presión de memoria, liberan automáticamente los recursos almacenados en memoria temporal (como los objetos File creados en tiempo de ejecución), lo cual provoca que los enlaces generados para su visualización dejen de funcionar.

Dado que este comportamiento dependía del sistema de gestión de memoria del navegador, y escapaba del control directo de la aplicación, fue necesario adoptar una solución que permitiera un almacenamiento más persistente y fiable. En este contexto, se optó por utilizar IndexedDB, una base de datos orientada a objetos integrada en el navegador, diseñada para almacenar grandes cantidades de datos estructurados de forma persistente.

IndexedDB permite guardar información en el lado del cliente de forma no volátil, lo que significa que los datos permanecen accesibles incluso después de cerrar la pestaña o la aplicación. Esto lo convierte en una herramienta ideal para almacenar imágenes y otros recursos multimedia que deben poder recuperarse en cualquier momento, independientemente del estado de la memoria del dispositivo.

La implementación del uso de IndexedDB en la aplicación se llevó a cabo de la siguiente forma:

1. **Almacenamiento inicial**: En el momento en que una imagen es introducida en la aplicación (por ejemplo, mediante una subida de archivo o captura), esta se guarda tanto en una variable File como en la base de datos IndexedDB. Para ello, se genera una clave única asociada a esa imagen que permite su posterior recuperación.
2. **Verificación de validez**: Cada vez que se intenta mostrar una imagen en la interfaz a partir de su variable File, se realiza una comprobación para verificar si su contenido sigue siendo válido. Esto se puede hacer, por ejemplo, intentando crear un ObjectURL y detectando si el recurso es accesible.
3. **Recuperación desde IndexedDB**: En caso de que el contenido del File haya sido eliminado por el navegador y no pueda visualizarse, se accede a IndexedDB utilizando la clave correspondiente y se recupera la imagen almacenada. Esta imagen es entonces transformada nuevamente en un objeto File válido, permitiendo su correcta visualización en la interfaz sin pérdida de datos ni necesidad de volver a cargar la imagen manualmente.

Gracias a esta estrategia, se garantiza una experiencia de usuario estable en dispositivos móviles, incluso en condiciones en las que el sistema operativo libera memoria de forma agresiva. IndexedDB actúa como una copia de seguridad local, asegurando la persistencia de recursos importantes sin depender de la conectividad ni del estado de la memoria volátil del navegador.

6.4. Despliegue en GitLab Pages

Como ya se mencionó anteriormente, para el despliegue de la aplicación durante su fase de desarrollo, se utilizó GitLab Pages, una funcionalidad integrada en GitLab que permite publicar sitios web estáticos directamente desde un repositorio. En concreto, se utilizó la instancia privada de GitLab proporcionada por la Escuela de Ingeniería Informática de Valladolid [35], lo que implica que tanto los repositorios como las páginas desplegadas están alojados en la infraestructura de la universidad, con configuraciones particulares, como el uso de dominios propios y certificados autofirmados.

Para lograr el despliegue de esta aplicación, fue necesario ajustar algunos aspectos específicos del proyecto Angular, debido a los requisitos particulares de GitLab Pages. Dado que la versión utilizada de Angular era la 18, y GitLab Pages requiere que los archivos finales de la aplicación estén ubicados en una carpeta llamada public, fue necesario reorganizar la estructura del proyecto. En versiones modernas de

Angular (a partir de la 17), la carpeta public puede utilizarse como sustituto de la tradicional carpeta assets para almacenar recursos estáticos, especialmente en proyectos generados con la configuración moderna. En este caso, como public debía destinarse exclusivamente al resultado del comando ng build, se creó una nueva carpeta llamada assets, en la que se reubicaron todos los iconos y recursos de la aplicación. Además, se actualizaron las rutas correspondientes en el código para mantener su funcionamiento original.

A partir de ese momento, la carpeta public pasó a ser la carpeta de destino del comando ng build, que genera los archivos estáticos de la aplicación listos para ser desplegados. Este paso es indispensable, ya que GitLab Pages únicamente expone el contenido de dicha carpeta al desplegar la aplicación.

El despliegue se gestiona automáticamente a través del archivo de configuración .gitlab-ci.yml, que define una pipeline de CI/CD (Integración y Entrega/Despliegue Continuas). Una pipeline de CI/CD es una secuencia automatizada de tareas que permite compilar, testear y desplegar aplicaciones de forma segura y repetible cada vez que se suben cambios al repositorio.

En este archivo:

- Se especifica la imagen de Node.js que se utilizará en el entorno de CI.
- Se instalan el CLI de Angular y las dependencias del proyecto antes de la compilación.
- En la sección pages, se construye el proyecto Angular en modo producción, redirigiendo la salida a la carpeta public. Como Angular 18 genera los archivos finales dentro de una subcarpeta llamada browser, es necesario mover su contenido al nivel superior de public para que GitLab Pages pueda servirlos correctamente.
- Finalmente, se definen las reglas para que el despliegue se realice únicamente cuando se hace push en la rama principal del repositorio.

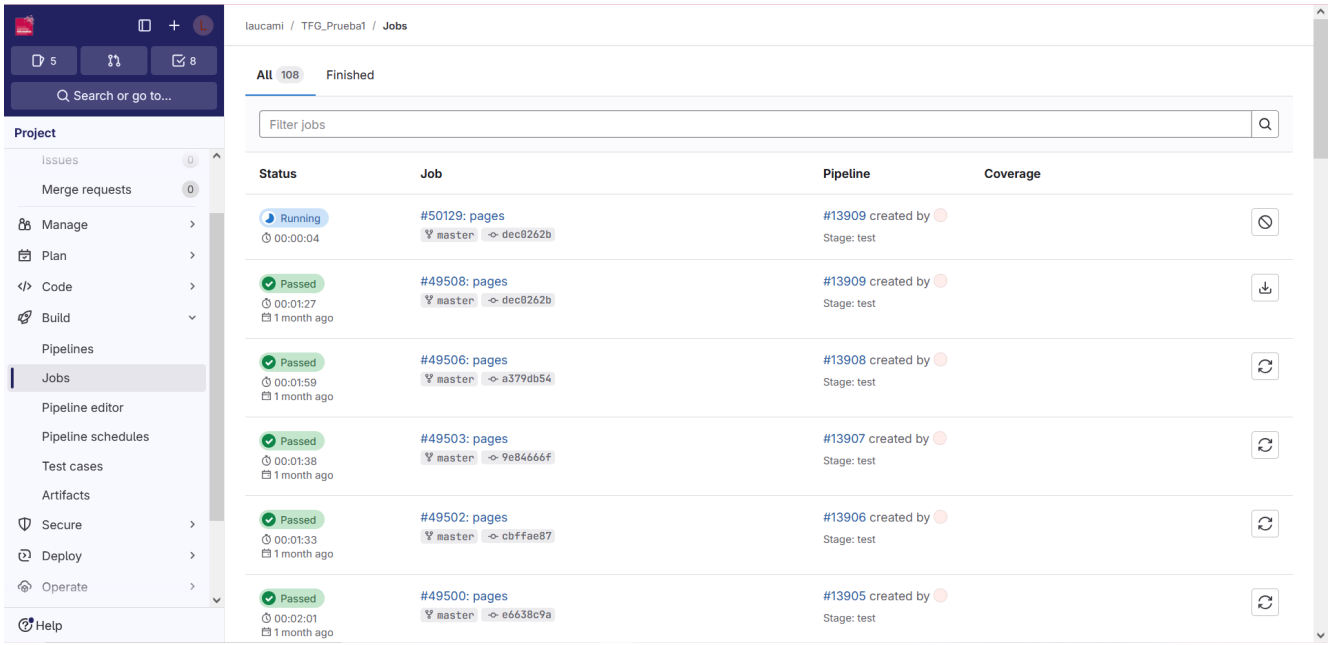

Figura 6.1: Sección Build > Jobs dentro del proyecto de GitLab

Una vez añadido este archivo .gitlab-ci.yml en la raíz del proyecto y realizado un push al repositorio, GitLab ejecuta automáticamente la pipeline, la cual puede monitorizarse desde el apartado `Build >Pipelines` del menú del proyecto, como se puede apreciar en la Figura 6.1. Desde allí es posible ver el progreso de la ejecución, así como consultar cualquier error que haya ocurrido durante la instalación de dependencias o la compilación del proyecto.

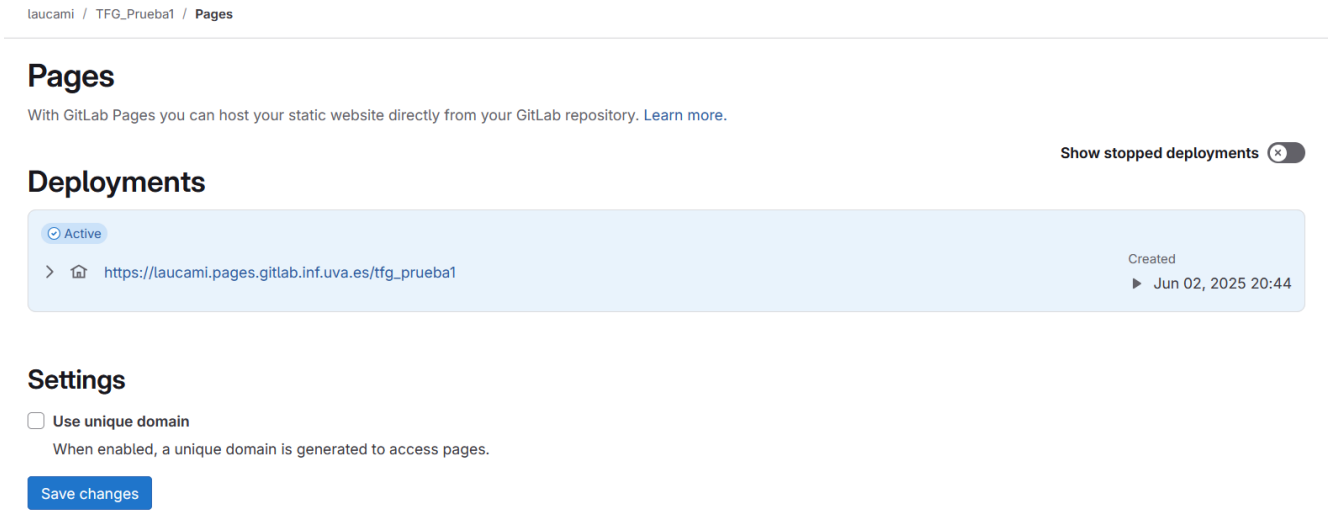

Figura 6.2: Sección Deploy > Pages dentro del proyecto GitLab

Cuando la pipeline finaliza correctamente, el enlace público de la aplicación aparece en el apartado `Deploy >Pages` (vease la Figura 6.2), accesible desde el menú lateral de GitLab. En este caso, se desactivó la opción “Use unique domain”, por lo que la URL generada se mantiene bajo el dominio

común de GitLab Pages correspondiente a la universidad.

Debido a que el GitLab de la Escuela de Ingeniería Informática utiliza certificados HTTPS autofirmados, que no cuentan con el respaldo de una autoridad certificadora reconocida y, por lo tanto, no son considerados confiables por los navegadores, se observan dos consecuencias prácticas en la aplicación desplegada:

- Al acceder a la web desde cualquier navegador, se indica que la conexión no es segura y es necesario aceptar manualmente el riesgo para continuar.
- En algunos navegadores, la funcionalidad PWA no está disponible, ya que el uso del service worker requiere una conexión segura mediante HTTPS.

6.5. Despliegue en GitHub Pages

Una vez se obtuvo la versión final y estable de la aplicación, y no se preveían más cambios importantes, se procedió a subir el proyecto a GitHub con visibilidad pública, con el objetivo de publicarlo de forma accesible en internet mediante GitHub Pages.

El primer paso consistió en crear un nuevo repositorio en GitHub con visibilidad pública y subir manualmente el código fuente del proyecto. Una vez alojado el repositorio, se procedió a configurar su despliegue mediante GitHub Pages.

Para facilitar este proceso de despliegue en aplicaciones desarrolladas con Angular, una herramienta muy utilizada es angular-cli-ghpages. Esta utilidad permite desplegar de forma sencilla una aplicación Angular en GitHub Pages directamente desde la línea de comandos. Internamente, se encarga de compilar el proyecto, generar los archivos estáticos correspondientes en la carpeta dist/, y publicar ese contenido automáticamente en la rama gh-pages del repositorio, que es la que utiliza GitHub para servir las páginas web.

Gracias a esta herramienta, se evita tener que realizar el proceso manual de compilar, cambiar de rama, mover archivos, y hacer el commit correspondiente. En su lugar, con unos pocos comandos, el despliegue completo queda automatizado y funcional en pocos segundos.

Para instalar y configurar la herramienta, se ejecutan los siguientes comandos:

```
npm i angular-cli-ghpages
ng add angular-cli-ghpages
```

El primer comando instala el paquete necesario en el proyecto, mientras que el segundo añade automáticamente la configuración requerida al entorno Angular. Esto incluye agregar las dependencias necesarias y preparar el proyecto para facilitar el despliegue en GitHub Pages.

Finalmente, se realiza el despliegue con el siguiente comando:

```
ng deploy --base-href=/MatroskaLearn/
```

Este último paso compila la aplicación y publica el contenido en la rama gh-pages del repositorio. El parámetro –base-href indica la ruta base desde la que se cargará la aplicación en el navegador, en este caso /MatroskaLearn/, que corresponde al nombre del repositorio.

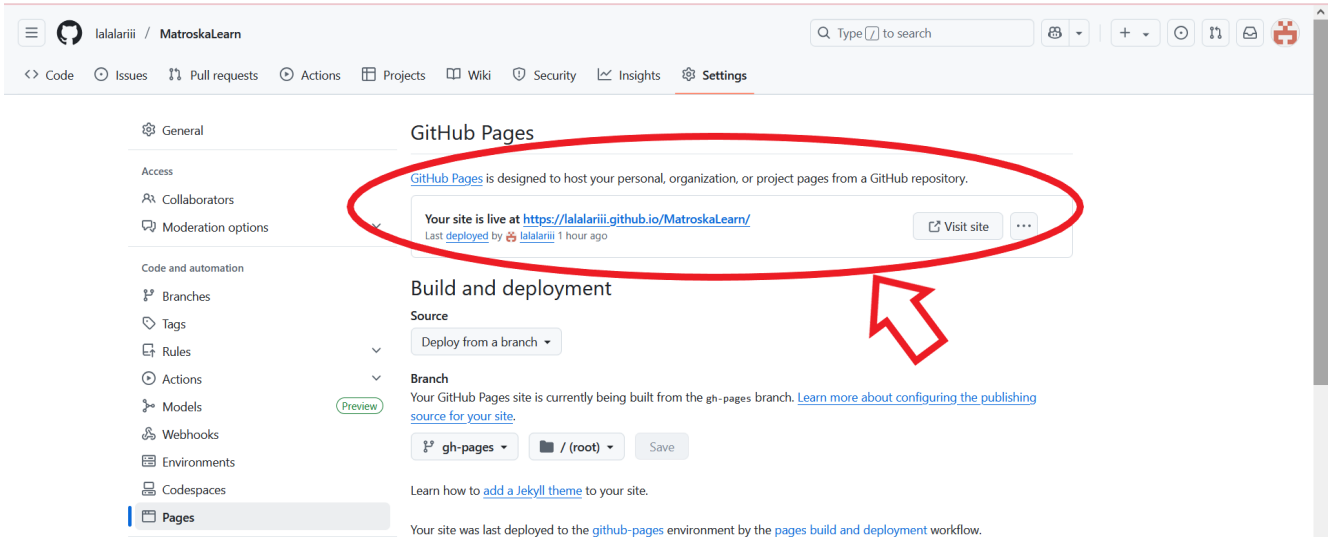

Figura 6.3: Sección Settings > Pages dentro del repositorio GitHub

Una vez completado el proceso, se puede acceder a la web desde la URL generada automáticamente por GitHub, disponible en el apartado `Settings >Pages` del repositorio, como se muestra en la Figura 6.3.

Capítulo 7

Pruebas

Para explicar las pruebas realizadas para verificar el correcto funcionamiento de la aplicación desarrollada en este proyecto, se tomará como referencia el libro Software Engineering de Ian Sommerville (6ª edición). En esta obra, el autor señala que el propósito principal de una prueba de software no es demostrar que el sistema funciona perfectamente, sino descubrir defectos que puedan corregirse antes de la entrega final [49]. En otras palabras, las pruebas deben concebirse como un proceso orientado a la detección de errores, no como una simple validación de éxito.

En este proyecto, se adoptó una estrategia de pruebas manuales, sin el uso de herramientas automatizadas. La validación se realizó principalmente a través de pruebas exploratorias por parte de la autora y mediante pruebas de aceptación con usuarios reales. Se buscó evaluar si el comportamiento de la aplicación coincidía con los requisitos establecidos, tanto funcionales como no funcionales, y detectar errores que pudieran afectar a la experiencia de uso.

Según Sommerville, las pruebas pueden organizarse en distintos niveles, cada uno de los cuales se enfoca en un aspecto específico del sistema. A continuación, se describen estos niveles y cómo fueron abordados en el desarrollo de esta aplicación:

- **Pruebas de unidad**: Este nivel se centra en verificar el correcto funcionamiento de los componentes individuales de software. En este caso, se validaron manualmente funciones clave de la lógica de la aplicación, como la gestión de marcas temporales en el audio, la navegación entre secciones, o la lectura y escritura de datos en archivos MKV.
- **Pruebas de integración**: Se refieren a la comprobación del funcionamiento conjunto entre distintos módulos. Para esta aplicación, se probaron manualmente las interacciones entre los componentes que la conforman, así como los servicios responsables del almacenamiento y la compartición de datos entre estos componentes. Se prestó especial atención a la sincronización entre los diferentes elementos y la correcta propagación de eventos.

- **Pruebas de sistema**: En este nivel se evalúa el sistema completo en condiciones similares a las reales. La aplicación se probó en distintos dispositivos (ordenadores de escritorio y teléfonos móviles) y navegadores (Google Chrome y Mozilla Firefox, principalmente) para asegurar su comportamiento esperado, su diseño responsive y su estabilidad y rendimiento general.
- **Pruebas de aceptación**: Estas pruebas se enfocan en verificar que el sistema cumple con las expectativas del usuario final. La aplicación fue entregada tanto al tutor como a otros usuarios para que la utilizaran de forma independiente durante un período aproximado de una semana. Durante ese tiempo, realizaron distintas pruebas en condiciones reales de uso y proporcionaron comentarios valiosos sobre errores detectados, funcionalidades útiles y aspectos susceptibles de mejora. Estas observaciones se tuvieron en cuenta para realizar correcciones y mejoras antes de la versión final.

Aunque no se dispuso de un sistema automatizado de pruebas, se procuró realizar una revisión exhaustiva de cada funcionalidad implementada, especialmente aquellas críticas para el flujo principal de uso. La combinación de pruebas exploratorias, pruebas con usuarios y validaciones en distintos entornos permitió garantizar un nivel aceptable de fiabilidad y calidad para el alcance del proyecto.

Capítulo 8

Conclusiones y líneas futuras

8.1. Conclusiones

En este Trabajo de Fin de Grado, se ha logrado desarrollar una aplicación funcional orientada a facilitar el aprendizaje a partir de audios de musicales. La herramienta cumple con los objetivos propuestos en cuanto a facilidad de despliegue, sencillez de uso, flexibilidad y versatilidad. La aplicación ha sido validada por el cliente principal, representado por una comunidad de intérpretes aficionados de música, de la cual forma parte el tutor del trabajo.

La elaboración de este proyecto ha supuesto una experiencia de aprendizaje muy enriquecedora a nivel técnico y personal. Aunque ya contaba con conocimientos previos de Angular gracias a la asignatura "Diseño basado en Componentes y Servicios", el desarrollo de esta aplicación me ha permitido profundizar mucho más en este framework. He aprendido a estructurar la interfaz en componentes más definidos, asignar a cada uno de ellos funcionalidades concretas y crear servicios capaces de gestionar la compartición y sincronización de datos entre distintas partes de la aplicación.

Además, me he enfrentado por primera vez al uso de bibliotecas avanzadas y complejas que no había utilizado anteriormente, lo cual me ha obligado a investigar y adaptarme constantemente. También he tenido la oportunidad de descubrir y trabajar con nuevas tecnologías como WebAssembly (wasm), Progressive Web Apps (PWA) e IndexedDB, que han ampliado significativamente mis conocimientos sobre el desarrollo web moderno.

Una de las áreas en las que más he evolucionado ha sido en el diseño de interfaces gráficas. El objetivo de crear una aplicación clara, fácil de usar y visualmente compacta, me ha llevado a reflexionar sobre usabilidad y organización del contenido. Posteriormente, la adaptación de esta interfaz a distintos tamaños y orientaciones de pantalla me ha permitido adquirir una sólida base en diseño responsive, apoyándome fundamentalmente en Bootstrap.

El proyecto también me ha brindado una valiosa experiencia en la planificación y gestión de un desarrollo completo desde cero, mucho más extenso que los realizados previamente en prácticas de la carrera. La adopción de una estrategia ágil de trabajo me ha ayudado a comprender mejor cómo se organizan normalmente los procesos de desarrollo en entornos profesionales, permitiéndome iterar, mejorar y corregir aspectos progresivamente.

Durante el proceso, también he aprendido sobre las particularidades del entorno web, incluyendo diferencias importantes entre navegadores de escritorio y móviles, lo que me ha llevado a tomar decisiones específicas para asegurar la compatibilidad y el correcto funcionamiento de la aplicación en ambos contextos.

Se espera que este proyecto actúe como punto de partida para futuras ampliaciones, tanto a través de nuevos Trabajos de Fin de Grado como mediante las aportaciones de la comunidad de desarrolladores del entorno.

8.2. Líneas futuras

En este apartado, se presentan algunas posibles mejoras y ampliaciones para la funcionalidad de la aplicación en futuras versiones:

- En el modo de lectura, la imagen asociada a una sección actualmente se muestra centrada dentro de un recuadro sin posibilidad de ser ampliada o reducida. Sería deseable implementar funcionalidades de zoom y desplazamiento (scroll) para facilitar una visualización más flexible y detallada de la imagen, sobre todo en imágenes largas o con mucho detalle.
- Los usuarios de dispositivos Apple suelen encontrar dificultades para manejar archivos en formatos menos comunes, como MKV. Por ello, se podría añadir la funcionalidad de importar y subir archivos MKV directamente desde servicios en la nube, mejorando la compatibilidad y la experiencia de usuario en estos dispositivos.
- Se podría desarrollar un sistema de guardado y carga local de proyectos que permita mostrar una lista de los archivos recientemente abiertos o editados. Esto facilitaría la gestión de proyectos activos sin necesidad de manejar directamente los archivos MKV para cargar la información.
- Dado que el propósito principal de esta aplicación es didáctico, una posible mejora sería la incorporación de preguntas tipo test asociadas a cada sección de audio, con el fin de enriquecer la experiencia de aprendizaje y evaluación.
- Aprovechando que el contenedor principal para el almacenamiento de datos es el formato MKV, se podría explorar la posibilidad de que, además de almacenar información, estos archivos pudieran

reproducirse y visualizarse como videos convencionales. Actualmente, la información contenida solo es accesible dentro de la propia aplicación.

- Sería interesante permitir que un único proyecto contenga múltiples audios, cada uno con sus respectivas secciones, marcas e información asociada, todo almacenado en un solo archivo MKV. Esta funcionalidad podría ser especialmente útil para organizar obras musicales divididas en varias partes, facilitando su gestión en un solo proyecto.
- Para mejorar la usabilidad, se podría incorporar ayuda contextual en la interfaz, como descripciones emergentes al situar el cursor sobre los botones, o un tutorial inicial que guíe al usuario en el uso de las funcionalidades básicas de la aplicación.
- Finalmente, una idea de futuro podría ser desarrollar una plataforma en línea para la publicación y comercialización de proyectos creados con esta aplicación, considerando aspectos legales relacionados con los derechos de autor y la propiedad intelectual.

Bibliografía

[1] Musescore. Software gratuito de composición y notación musical | MuseScore. https://musescore.org/es. Último acceso: 16 de junio de 2025.

[2] Wikipedia. Aplicación web progresiva. https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web_progresiva, 2025. Último acceso: 16 de junio de 2025.

[3] Wikipedia. Desarrollo ágil de software. https://es.wikipedia.org/wiki/Desarrollo_gil_de_software, 2025. Último acceso: 16 de junio de 2025.

[4] Atlassian. Explicación de los diagramas de Gantt [y cómo crear uno] | Atlassian. https://www.atlassian.com/es/agile/project-management/gantt-chart. Último acceso: 16 de junio de 2025.

[5] B. Hughes and M. Cotterell. *Software Project Management*. McGraw-Hill Higher Education, 2009.

[6] InfoJobs | Miles de oportunidades laborales cada día. https://www.infojobs.net/. Último acceso: 16 de junio de 2025.

[7] Búsqueda de empleo en Glassdoor. https://www.glassdoor.es/index.htm. Último acceso: 16 de junio de 2025.

[8] Talent.com: Búsqueda de empleo | Encuentra ofertas cerca de ti. https://es.talent.com. Último acceso: 16 de junio de 2025.

[9] Wikipedia contributors. Audio file format — Wikipedia, the free encyclopedia, 2025. Último acceso: 16 de junio de 2025.

[10] MakeMusic. MusicXML for Exchanging Digital Sheet Music. https://www.musicxml.com/. Último acceso: 16 de junio de 2025.

[11] Andrew Swift. A brief introduction to MIDI. https://web.archive.org/web/20120830211425/http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/, 2012. Último acceso: 16 de junio de 2025.

[12] Avid. Sibelius - Notation Software. https://www.avid.com/sibelius. Último acceso: 16 de junio de 2025.

[13] Finale. programa de notación musical finale. https://www.finalemusic.com/es/. Último acceso: 16 de junio de 2025.

[14] MuseScore: partitura - Aplicaciones en Google Play. https://play.google.com/store/apps/details?id=com.musescore.playerlite&hl=es&pli=1. Último acceso: 16 de junio de 2025.

[15] capella score reader - Aplicaciones en Google Play. https://play.google.com/store/apps/details?id=de.capella.capReader&hl=es_419. Último acceso: 16 de junio de 2025.

[16] Sibelius - Apps en Google Play. https://play.google.com/store/apps/details?id=com.avid.sibelius.android&hl=es_419. Último acceso: 16 de junio de 2025.

[17] MobileSheets. https://www.zubersoft.com/mobilesheets/. Último acceso: 16 de junio de 2025.

[18] ScorePDF: Partitura Visor - Aplicaciones en Google Play. https://play.google.com/store/apps/details?id=com.enoiu.scorepdf&hl=es. Último acceso: 16 de junio de 2025.

[19] Complete Ear Trainer - Aplicaciones en Google Play. https://play.google.com/store/apps/details?id=com.binaryguilt.completeeartrainer&hl=es. Último acceso: 16 de junio de 2025.

[20] Chord ai – Chords and beats for any song! https://chordai.net/. Último acceso: 16 de junio de 2025.

[21] Audacity ® | Free Audio editor, recorder, music making and more! https://www.audacityteam.org/. Último acceso: 16 de junio de 2025.

[22] Wikipedia. DaVinci Resolve. https://es.wikipedia.org/wiki/DaVinci_Resolve, 2025. Último acceso: 16 de junio de 2025.

[23] Angular. Angular - Introduction to the Angular docs. https://v17.angular.io/docs. Último acceso: 16 de junio de 2025.

[24] Wikipedia. Matroska. https://en.wikipedia.org/wiki/Matroska, 2025. Último acceso: 16 de junio de 2025.

[25] Wikipedia. Webassembly. https://en.wikipedia.org/wiki/WebAssembly, 2025. Último acceso: 16 de junio de 2025.

[26] @ffmpeg/ffmpeg. https://www.npmjs.com/package/@ffmpeg/ffmpeg/v/0.10.0, 2021. Último acceso: 16 de junio de 2025.

[27] Lambdatest. Cross Browser Compatibility Score of WebAssembly. https://www.lambdatest.com/web-technologies/wasm. Último acceso: 16 de junio de 2025.

[28] Can i use... WebAssembly | Can I use... Support tables for HTML5, CSS3, etc. https://caniuse.com/wasm. Último acceso: 16 de junio de 2025.

[29] MDN. WebAssembly | MDN. https://developer.mozilla.org/en-US/docs/WebAssembly#browser_compatibility, 2025. Último acceso: 16 de junio de 2025.

[30] Ffmpeg. FFmpeg. https://ffmpeg.org/. Último acceso: 16 de junio de 2025.

[31] wavesurfer.js. https://wavesurfer.xyz/docs/. Último acceso: 16 de junio de 2025.

[32] MDN. IndexedDB API - Web APIs | MDN. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API, 2025. Último acceso: 16 de junio de 2025.

[33] idb. https://www.npmjs.com/package/idb, 2025. Último acceso: 16 de junio de 2025.

[34] GitLab Pages | GitLab Docs. https://docs.gitlab.com/user/project/pages/. Último acceso: 16 de junio de 2025.

[35] Sign in · GitLab. https://gitlab.inf.uva.es/users/sign_in. Último acceso: 16 de junio de 2025.

[36] GitHub Pages. https://pages.github.com/. Último acceso: 16 de junio de 2025.

[37] Wikipedia. TypeScript. https://es.wikipedia.org/wiki/TypeScript, 2025. Último acceso: 16 de junio de 2025.

[38] Wikipedia. Lenguaje unificado de modelado. https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado, 2025. Último acceso: 16 de junio de 2025.

[39] Astah. Astah Professional: UML, ER, DFD & Flowchart Software. https://astah.net/products/astah-professional/, 2025. Último acceso: 16 de junio de 2025.

[40] and Bootstrap Mark Otto contributors, Jacob Thornton. Get started with Bootstrap. https://getbootstrap.com/docs/5.3/getting-started/introduction/. Último acceso: 16 de junio de 2025.

[41] Anushtha Jain and Visure Solutions. ¿Qué son los requisitos funcionales? Ejemplos y plantillas. https://visuresolutions.com/es/alm-guide/functional-requirements, 2025. Último acceso: 16 de junio de 2025.

[42] Markdown - La guía definitiva en español. https://markdown.es/. Último acceso: 16 de junio de 2025.

[43] Anushtha Jain and Visure Solutions. ¿Qué son los requisitos no funcionales? Tipos, ejemplos y enfoques. https://visuresolutions.com/es/alm-guide/non-functional-requirements, 2025. Último acceso: 16 de junio de 2025.

[44] Wikipedia. Caso de uso. https://es.wikipedia.org/wiki/Caso_de_uso, 2025. Último acceso: 16 de junio de 2025.

[45] Wikipedia. Modelo de dominio. https://es.wikipedia.org/wiki/Modelo_de_dominio, 2024. Último acceso: 16 de junio de 2025.

[46] Angular. Angular - Introduction to Angular concepts. https://v17.angular.io/guide/architecture. Último acceso: 16 de junio de 2025.

[47] Diagrama de paquetes UML: Qué es y cómo hacerlo | Miro. https://miro.com/es/diagrama/que-es-diagrama-paquetes-uml/. Último acceso: 16 de junio de 2025.

[48] Coolors. Coolors - The super fast color palettes generator! https://coolors.co/. Último acceso: 16 de junio de 2025.

[49] Ian Sommerville. *Software engineering (6th ed.)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2001.

Apéndice A

Repositorio del código

A.1. Enlace del repositorio en GitHub

El código fuente de este proyecto está disponible en el siguiente repositorio de GitHub:
`https://github.com/lalalariii/MatroskaLearn`

A.2. Enlace de la página web

La página web desarrollada en este proyecto esta disponible en el siguiente enlace:
`https://lalalariii.github.io/MatroskaLearn/`

A.3. Organización del repositorio

Actualmente, el repositorio del proyecto contiene los archivos de configuración y dependencias propias de una aplicación Angular. Las carpetas principales incluyen:

- src/: contiene el código fuente de la aplicación.
- assets/: almacena recursos estáticos como imágenes o archivos auxiliares.
- .vscode/: configuración específica del entorno de desarrollo en Visual Studio Code.

Además, se incluyen ficheros como angular.json, package.json o tsconfig.json, entre otros, que gestionan la configuración del proyecto, sus dependencias y scripts de compilación.

Apéndice B

Manual de usuario

Este manual tiene como objetivo servir de guía básica para los usuarios de la aplicación desarrollada. En él se explican de manera sencilla y clara las funcionalidades disponibles, así como los pasos necesarios para utilizarlas correctamente. La aplicación cuenta con dos modos principales de uso, modo lectura y modo escritura, por lo que se proporciona una explicación diferenciada para cada uno de ellos.

B.1. Crear un nuevo proyecto o cargar uno existente

Figura B.1: Crear o cargar nuevo proyecto en la aplicación

En la Figura B.1 se muestra la barra superior de la aplicación, que incluye el logotipo y permite cambiar entre los dos modos disponibles: “Creación de archivo” y “Lectura de archivo”. Justo debajo se encuentra una barra con la pestaña “Archivo”, la cual, al desplegarse, ofrece las siguientes opciones de carga:

- **“Nuevo” (disponible únicamente en el modo de escritura)**: permite iniciar un nuevo proyecto de audio a partir de un archivo en formato MP3 proporcionado por el usuario.

- **“Carga” (disponible en ambos modos)**: permite abrir un proyecto previamente creado con la aplicación, mediante la carga de un archivo MKV que contiene todos los datos del proyecto.

B.2. Modo escritura

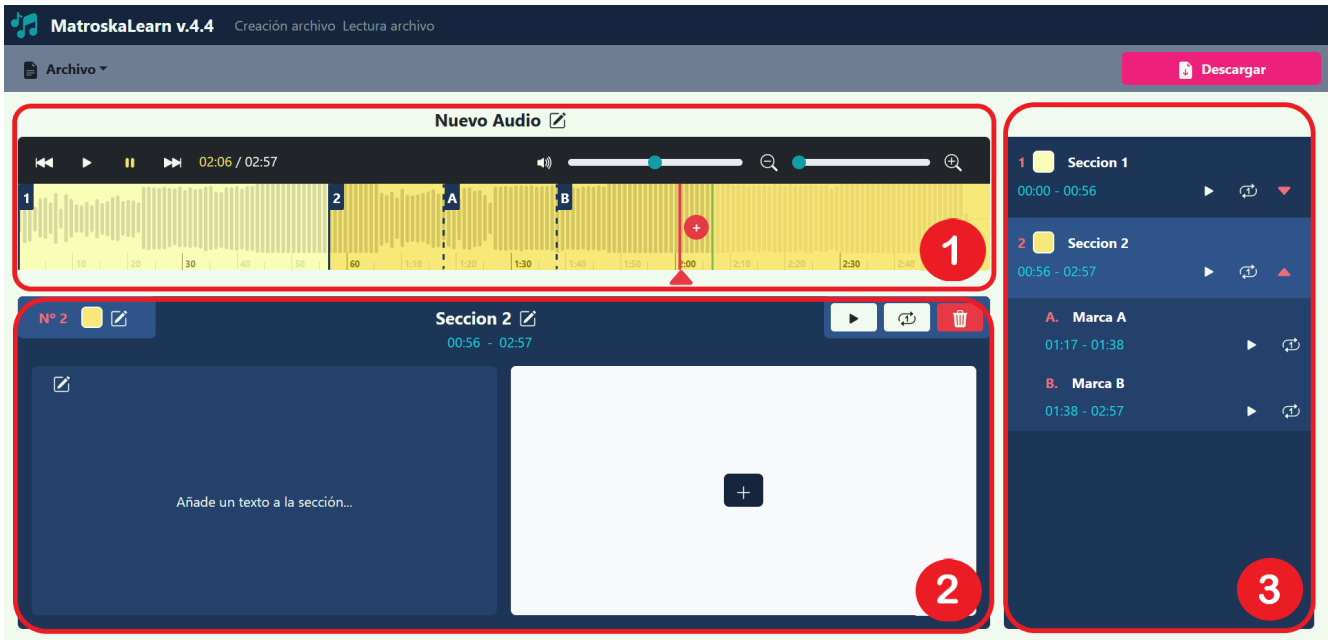

Figura B.2: Interfaz en modo escritura, dividida en partes principales

En la Figura B.2 se muestra una captura de la interfaz de la aplicación en modo escritura. Como puede apreciarse, la interfaz se divide en tres zonas principales, numeradas en la imagen:

Zona de onda sonora (número 1): Este área se compone a su vez de tres partes diferenciadas:

- **Edición del título**: en la parte superior, el usuario puede introducir o modificar el título del archivo MKV que se generará al exportar el proyecto.
- **Controles de audio (barra negra situada encima de la onda)**:
  - En el lado derecho se encuentra el indicador de tiempo, que muestra la posición actual de la reproducción respecto a la duración total del audio, junto con cuatro botones de control: retroceder a la sección anterior, avanzar a la siguiente, reproducir y pausar.
  - En el lado izquierdo, se sitúan dos controles deslizantes: uno para modificar el volumen y otro para ajustar el nivel de zoom sobre la onda sonora.
- **Representación de la onda**: Esta zona visualiza el audio y los elementos interactivos añadidos sobre él. Se distinguen tres tipos:

- **Secciones**: regiones del audio marcadas por un color e identificadas con un número al inicio. La primera sección tiene su tiempo de inicio fijo, pero el de las demás puede modificarse arrastrando su etiqueta a lo largo de la línea de tiempo.
- **Marcas**: puntos específicos del audio, representados por una línea discontinua con una letra mayúscula como identificador. A diferencia de las secciones, ocupan un único instante de tiempo y pueden moverse libremente arrastrándolas.

Figura B.3: Crear una sección o marca dentro de la onda en la aplicación

- **Cursor**: representado por una línea roja vertical rematada por una flecha. A su izquierda aparece un botón rojo con el símbolo "+". Al pulsarlo, se despliega el popup mostrado en la Figura B.3, que permite añadir una nueva sección (dividiendo la sección actual en dos, de modo que el nuevo tiempo marque el inicio de la nueva sección y el final de la anterior) o insertar una nueva marca, en el instante de tiempo en el que se encuentra el cursor. Cuando el cursor está en movimiento, muestra el minuto exacto por el que avanza; al detenerse, el audio comienza a reproducirse desde ese punto.

Zona del panel de edición (número 2): En esta área se muestran los datos correspondientes a la sección o marca actualmente seleccionada, e incluye distintos controles para modificar su información. Puede dividirse en tres partes principales:

- **Zona de controles (parte superior)**: Desde esta sección se pueden realizar las siguientes acciones:
  - Cambiar el color asignado a la sección.
  - Visualizar y editar los tiempos de inicio y fin, así como el título de la sección o marca seleccionada. En el caso de una marca, el tiempo de fin mostrado corresponderá al de la siguiente marca dentro de la misma sección, o bien, si no existe otra marca, al fin de la sección que la contiene.
  - Reproducir la sección o marca desde el inicio, activarla en bucle, o eliminarla. Cabe destacar que la sección número 1 no se puede eliminar, ya que debe existir al menos una sección en el proyecto, y si es la única, abarcará todo el audio.

Figura B.4: Editar texto de una sección en la aplicación

- **Zona de texto (parte inferior izquierda)**: Aquí se puede editar el contenido textual asociado a la sección o marca. La edición se realiza mediante un cuadro desplegable que ocupa la mayor parte de la pantalla para facilitar la escritura, como se muestra en la Figura B.4. En el caso de las secciones, este cuadro incluye una opción para activar modo Markdown, lo que divide el área en dos partes: a la izquierda, un editor de texto, y a la derecha, una vista previa en tiempo real del formato final.

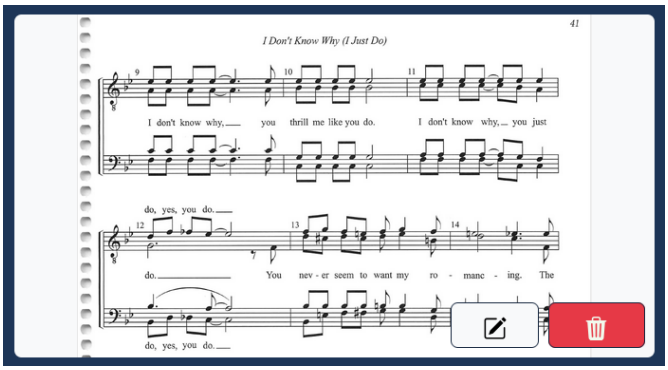

Figura B.5: Controles de edición y borrado de la imagen de una sección en la aplicación

- **Zona de imagen (parte inferior derecha)**: Disponible únicamente en las secciones, esta área permite añadir una imagen asociada. Inicialmente se muestra un botón "+", que abre el explorador de archivos para seleccionar una imagen desde el dispositivo. Una vez añadida, la imagen se visualiza dentro del recuadro, acompañada de dos botones que permiten reemplazarla o eliminarla, como se muestra en la Figura B.5.

Zona del panel de selección (número 3): Esta área muestra una lista con todas las secciones y marcas que componen el audio del proyecto actual. Al hacer clic sobre un elemento de la lista, este se selecciona y se resalta visualmente. A continuación, se detalla la información y funcionalidades asociadas a cada tipo de elemento:

- **Sección**: Cada sección de la lista incluye:
  - **Información resumida**: título, color, y tiempos de inicio y fin.
  - **Controles de reproducción**: un botón para reproducir desde el inicio y otro para activar la reproducción en bucle.
  - Un botón con forma de pestaña que permite desplegar o replegar la lista de marcas contenidas dentro de esa sección.
- **Marca**: Las marcas aparecen como elementos secundarios dentro de las secciones. Para cada marca se muestra:
  - **Información resumida**: título y tiempos de inicio y fin.
  - **Controles de reproducción**: un botón para reproducir desde el inicio y otro para reproducir en bucle la marca.

B.3. Modo lectura

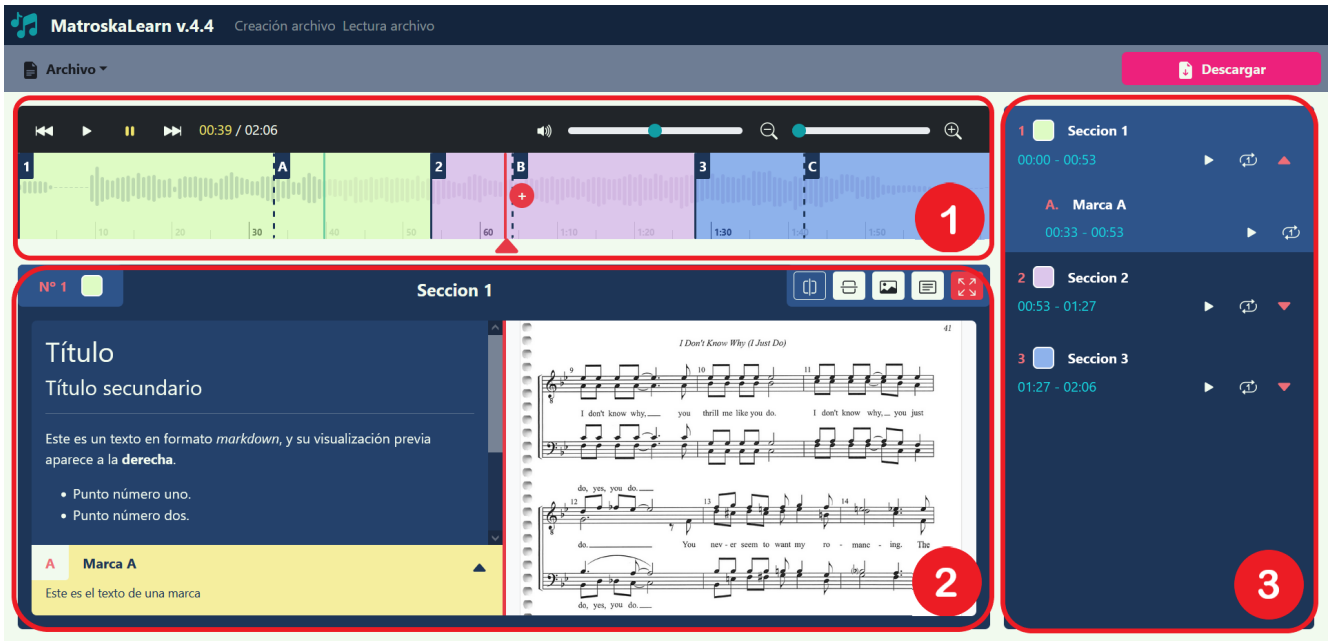

Figura B.6: Interfaz en modo lectura, dividida en partes principales

En la Figura B.6 se muestra una captura de la aplicación en modo lectura. La interfaz mantiene la misma división en tres zonas principales que el modo escritura, ya explicadas en el apartado anterior. En este caso, nos centraremos únicamente en las diferencias específicas de la interfaz de lectura con respecto a la de escritura.

Zona de onda sonora (número 1): En cuanto a la funcionalidad y la interacción con los elementos de la onda, esta zona presenta ciertas limitaciones en comparación con el modo escritura:

- Las secciones y marcas no se pueden editar ni arrastrar, salvo las marcas creadas en el propio modo lectura, que sí pueden ser modificadas.

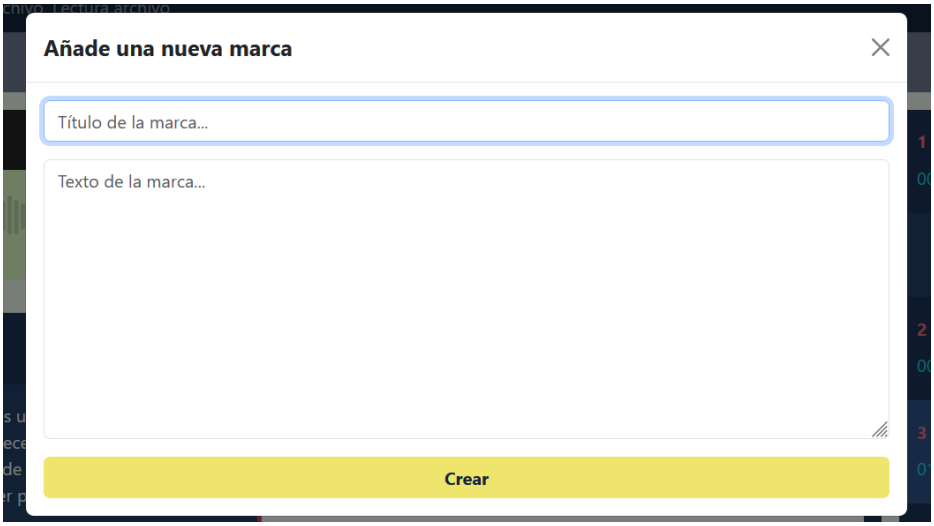

Figura B.7: Crear una marca de lectura en la aplicación

- No se pueden crear nuevas secciones. El botón "+" situado junto al cursor permite añadir marcas de lectura. Al pulsarlo, se abre el cuadro emergente que se muestra en la Figura B.7, donde se puede introducir el título y el texto de la nueva marca.

Zona del panel de visualización (número 2): Esta zona presenta varias diferencias con respecto al modo escritura:

- En la parte inferior se encuentra el reproductor, donde se visualizan los datos de la sección que se está reproduciendo en ese momento. Si hay una marca en reproducción, esta se muestra dentro de un recuadro amarillo, justo debajo del texto de la sección.

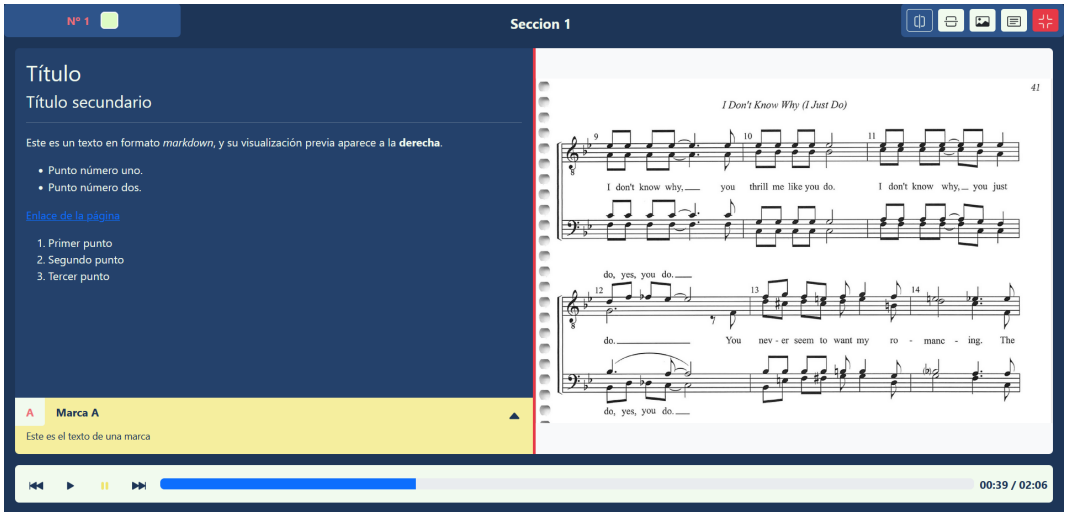

Figura B.8: Visualizar el reproductor en modo pantalla completa en la aplicación

- Los botones disponibles en esta zona permiten ajustar la presentación del contenido en el reproductor:
  - Cambiar el modo de visualización: solo imagen, solo texto o vista dividida (imagen y texto a la vez).
  - Cambiar la orientación del contenido en la vista dividida: vertical u horizontal.
  - Activar el modo pantalla completa con el botón rojo situado a la derecha. Este modo, mostrado en la Figura B.8, amplía el panel de visualización e incluye una barra de reproducción en la parte inferior con:
    - Tiempo actual de reproducción frente al total del audio.
    - Controles de reproducción: saltar a la sección anterior o siguiente, reproducir y pausar.
    - Una barra de progreso interactiva, sobre la que se puede hacer clic para saltar a un punto concreto del audio.

Zona del panel de selección (número 3): La única diferencia respecto a este panel en el modo escritura es que al hacer clic en un elemento de la lista (una sección o una marca), el audio comienza a reproducirse desde su inicio de forma automática.