UNIVERSITAT DE BARCELONA

SEÑALIZACIÓN DIGITAL DEL CRAI

BIBLIOTECA DE MATEMÁTICAS E INFORMÁTICA

Trabajo de final de grado

Grado en Ingeniería Informática

Facultad de Matemáticas e Informática

Universidad de Barcelona

Endimbeer Enrique Núñez Matos

Tutor: Eloi Puertas i Prats

Colaborador: Roger Angela i Gambús

Barcelona, enero de 2024

Resumen

Este Trabajo de Final de Grado consiste en una aplicación web para la gestión del contenido multimedia que se muestra a diario en la Biblioteca del Centro de Recursos para el Aprendizaje y la Investigación (CRAI) de la facultad de Matemáticas e Informática de la Universidad de Barcelona. Presenta una mejora respecto a la solución actual en términos de diseño, usabilidad y seguridad.

El *software* dedicado que incorporan los televisores es sencillo e intuitivo, pero resulta ineficiente para las necesidades de algunos usuarios. La aplicación ha sido diseñada para ir más allá de la mera reproducción; contiene una biblioteca plenamente gestionable, permite crear listas con funcionalidades de ordenación y asignación de prioridades sobre los elementos, ofreciendo una alternativa específica para este caso.

Alojada en diversos contenedores Docker, la aplicación se ejecuta dentro de una placa Raspberry Pi; el tamaño de este ordenador permite su instalación en espacios reducidos. Para facilitar la administración del hardware se brindan diferentes opciones de control y monitorización sobre el dispositivo.

El desarrollo del proyecto ha estado marcado por el uso de la metodología *Design Thinking.* Para llevarlo a cabo, he empleado: Vue3, Flowbite, HTML, JavaScript, Django y Docker.

Palabras clave: Aplicación Web, Reproductor Multimedia, UX/UI, Django, Vue3, JavaScript, Docker, HTML.

Abstract

This Final Degree Project presents a web application for managing multimedia content displayed daily in the Library of the Resource Center for Learning and Research (CRAI) of the Faculty of Mathematics and Informatics at the University of Barcelona. The application improves upon the current solution in terms of design, usability, and security.

The software incorporated in televisions is simple and intuitive, but it is inefficient for the needs of some users. The application was designed to offer more than just simple playback. It includes a fully manageable library that allows you to create lists with sorting and prioritization functionalities for the elements, providing a specific alternative for this case.

The application runs inside a Raspberry Pi board, hosted in various Docker containers. The size of this computer allows for its installation in small spaces. To facilitate hardware management, different control and monitoring options for the device are provided.

The project development was based on the Design Thinking methodology. The application was developed using Vue3, Flowbite, HTML, JavaScript, Django, and Docker.

Keywords: Web Application, Multimedia Player, UX/UI, Django, Vue3, JavaScript, Docker, HTML.

Resum

Aquest Treball de Final de Grau consisteix en una aplicació web per a la gestió del contingut multimèdia que es mostra diàriament a la Biblioteca del Centre de Recursos per a l'Aprenentatge i la Recerca (CRAI) de la facultat de Matemàtiques i Informàtica de la Universitat de Barcelona. Presenta una millora respecte a la solució actual en termes de disseny, usabilitat i seguretat.

El programari dedicat que incorporen els televisors és senzill i intuïtiu, però resulta ineficient per a les necessitats d'alguns usuaris. L'aplicació ha estat dissenyada per a anar més enllà de la simple reproducció; conté una biblioteca plenament gestionable, permet crear llistes amb funcionalitats d'ordenació i assignació de prioritats sobre els elements, oferint una alternativa específica per a aquest cas.

Allotjada en diversos contenidors Docker, l'aplicació s'executa dins d'una placa Raspberry Pi; la grandària d'aquest ordinador permet la seva instal·lació en espais reduïts. Per a facilitar l'administració del maquinari es brinden diferents opcions de control i monitoratge sobre el dispositiu.

El desenvolupament del projecte ha estat marcat per l'ús de la metodologia *Design Thinking*. Per a dur-ho a terme, he emprat: Vue3, Flowbite, HTML, JavaScript, Django i Docker.

Paraules clau: Aplicació Web, Reproductor Multimèdia, UX/UI, Django, Vue3, JavaScript, Docker, HTML.

Quiero dedicar este proyecto a mi familia por el apoyo recibido y a mis tutores Eloi y Roger por su colaboración durante estos meses.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas y gráficos

1. Introducción

La tecnología avanza frenéticamente, transformando nuestra realidad en un avance continuo hacia un mundo digital en el que las pantallas electrónicas han tomado las riendas en la difusión de información; tanto la diversidad de contenido como la versatilidad del formato, ayudan a reducir el consumo de papel y logran captar la atención de las personas con mayor efectividad.

Bajo esta premisa, en el año 2020, el CRAI Biblioteca de Matemáticas e Informática realizó la adquisición de una *Smart TV* para mostrar información de interés a sus usuarios. Sin embargo, el sistema de reproducción multimedia que incorporan estos dispositivos es limitado y no permite gestionar el contenido cómodamente.

Tras explorar el mercado en busca de alternativas, decidieron optar por el desarrollo de una solución a medida que derivó en una propuesta a la facultad para llevar a cabo la idea a través de un Trabajo de Final de Grado. La petición resultaría en un *software* dedicado que facilitase la gestión y la reproducción del contenido digital transmitido a través de la pantalla del vestíbulo de la biblioteca.

Según los requisitos especificados, se llevó a cabo una primera versión que cumplía con el objetivo original. No obstante, el paso del tiempo trajo consigo nuevas inquietudes que dieron lugar a una reiteración que no llegó a desplegarse en el entorno productivo. La versión actual parte de cero, añadiendo funcionalidades y mejorando las existentes en materia de diseño, usabilidad y seguridad.

1.1. Antecedentes

A principios de los años 90, Apple desarrolló *QuickTime*, un complemento para la visualización de vídeos que reproducía el sonido simultáneamente gracias a su códec[1], *Road Pizza*. Este significativo avance brindó a los usuarios los primeros reproductores multimedia sin necesidad de un *hardware* adicional.

Con el paso del tiempo esta tecnología fue extendiéndose a otros dispositivos como los televisores; en el año 1994 se concedió a la firma francesa FAST[2] la primera patente. Sin embargo, no fue hasta 2007 cuando se lanzó al mercado la primera *Smart TV*, desarrollada por HP, que incorporaba funcionalidades de conexión a internet y reproducción de archivos multimedia a través de un ordenador de manera remota.

En la actualidad, producto del aumento de las plataformas de *streaming*, el *software* nativo de los televisores inteligentes ha quedado relegado a un segundo plano. Sus funcionalidades no permiten crear listas de reproducción y el contenido debe seleccionarse manualmente, además requieren la conexión a un medio físico, como una memoria USB o un disco duro portátil, dificultando la gestión de los archivos.

Por ello, tras adquirir el dispositivo, los miembros del CRAI concluyeron en que el propósito que debía cumplir quedaba incompleto y fruto de esta situación se desarrolló la primera versión del *software*, caracterizada por las siguientes funcionalidades:

- Visualización de imágenes y vídeos.
- Creación de una cola de reproducción.
- Almacenaje de ficheros y listas en un medio extraíble.
- Gestión y borrado del contenido almacenado.
- Implementación del sistema en formato web para múltiples plataformas.
- Apagado automático de la Raspberry Pi.

[1] Programa o dispositivo físico capaz de codificar o decodificar una señal o flujo y datos digitales.

[2] *FRANCE ADVANCED SYSTEMS TECHNOLOGIE*.

Esta primera versión fue desarrollada por Vicent Núñez Delgado; el alumno diseñó dos aplicaciones, una de ellas para el control y otra para la visualización del contenido. Sin embargo, tras unos meses en funcionamiento, surgieron nuevas necesidades en el proyecto. Por ello, se originó una segunda versión que nunca llegó a desplegarse en un entorno productivo, esta contaba con los siguientes requisitos básicos:

- Creación de múltiples listas de reproducción.
- Borrado y etiquetado de listas de reproducción.
- Asignación de prioridades para aumentar la frecuencia de aparición de un archivo.
- Mejora de la búsqueda y el filtrado.
- Visualización de la duración de cada archivo.
- Creación y gestión de un sistema de agenda.

Partiendo de los requisitos iniciales, los propuestos para la segunda versión y la idea de Vicent de distribuir la solución mediante una arquitectura de múltiples servicios, se desarrolla esta última versión que hace *tabula rasa* del código fuente, mejorando las funcionalidades existentes e incorporando nuevas en materia de usabilidad y seguridad.

1.2. Objetivos

Principales

- Reproducir vídeos e imágenes en múltiples formatos y en diversas pantallas simultáneamente.
- Crear, editar y eliminar listas de reproducción.
- Sistema de agenda que permita programar la reproducción de una lista.
- Establecer prioridades para aumentar la frecuencia de aparición de un archivo.
- Cargar, modificar y eliminar archivos de la aplicación.
- Colocar etiquetas a los archivos para agruparlos por términos similares.
- Desarrollar un sistema de búsqueda que permita filtrar por nombre y/o etiquetas.
- Controlar el encendido y apagado de la placa Raspberry Pi.

Secundarios

- Mejorar la seguridad de la aplicación mediante el uso de tokens y la autenticación multifactor.
- Cambiar la interfaz de la aplicación para mejorar su usabilidad.
- Brindar opciones de personalización para extender su uso a otras instituciones.
- Crear *scripts* que permitan el despliegue de la aplicación en otros entornos.

1.3. Motivación

El motivo principal por el que escogí este proyecto fue probarme en el ámbito del desarrollo de *software*. Profundizar en aspectos como la programación de interfaces web fue un reto adicional; esta área siempre me ha entusiasmado, pero debido a mi trayectoria profesional he estado centrado en otros aspectos de la carrera como la arquitectura de sistemas.

2. Análisis

2.1. Metodología

Para llevar a cabo este proyecto decidí emplear varias metodologías tanto en el diseño del producto como en la organización del trabajo. Tras una etapa de investigación sobre las distintas técnicas existentes en materia de desarrollo de *software* me decanté por *Design Thinking* para la elaboración de la solución y *Agile* para la gestión del trabajo.

2.1.1. Organización del trabajo

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos de organización, la mayoría de ellas están pensadas para el trabajo en equipo, no individual. No obstante, existen varias que pueden ser adaptadas para el uso personal. A continuación, se detallan algunas de ellas.

Metodologías tradicionales

Se caracterizan por definir los requisitos del proyecto de manera explícita y rígida desde el inicio, impidiendo su modificación durante el ciclo de desarrollo. La organización del trabajo se realiza de manera lineal, una etapa empieza cuando termina la anterior. Este tipo de metodología no se adapta a los cambios, lo cual supone una gran desventaja en nuestra cambiante realidad, el ejemplo más representativo es *Waterfall.*

Metodologías ágiles

Las metodologías ágiles son incrementales; se caracterizan por el uso de ciclos cortos a través de los cuales se desarrolla el producto con la adición de nuevas funcionalidades. Además, permiten al cliente visualizar en tiempo real el avance del proyecto.

- ***Kanban:***

  Esta metodología de trabajo proviene del sistema de producción *Just In Time* (JIT) ideado por Taiichi Ohno, un ingeniero industrial japonés que lo diseñó para Toyota en la década de 1960. La metodología *Kanban* se convirtió en una pieza fundamental de este sistema y se utilizaba para controlar el avance del trabajo dentro de la línea de producción. Puede ser utilizado en diferentes áreas de una organización, subdividiendo las tareas a realizar se consigue ajustar la cantidad de trabajo a la capacidad del equipo, consiguiendo así un incremento en el valor del producto.

- ***Scrum*:**

  Similar al método *Kanban*, consiste en subdividir los requisitos en tareas que se van realizando en diversas iteraciones (*sprints*) a lo largo de 2 a 4 semanas. Consta de cuatro etapas: planificación de la iteración (*sprint planning*), ejecución (*sprint*), reuniones diarias (*daily meetings*) y una demostración de los resultados obtenidos (*sprint review*).

Tras estudiar detenidamente las diferentes opciones y sopesar tanto sus fortalezas como sus debilidades, decidí hacer una mezcla de *Kanban* y *Scrum*, comúnmente conocido como *Scrumban*. Esto se debe a la necesidad de mantener al cliente informado durante el proceso de trabajo; el uso de ciclos breves permite concluir el máximo número de tareas en un corto periodo de tiempo antes del siguiente *sprint review*.

El desarrollo de este proyecto ha precisado de ocho *sprints* de dos semanas cada uno, tras cada iteración se realizaba una reunión con el tutor y el *product owner* del proyecto en la que se mostraban las funcionalidades desarrolladas y se escogían las siguientes características a implementar del *backlog*. Además, cuando se obtuvo un producto mínimo viable se realizó un despliegue de la aplicación para que los usuarios pudieran probarla y aportar *feedback.*

Gracias a los conocimientos obtenidos a lo largo de mi carrera profesional, pude transformar un tablero *Kanban* sencillo en un sistema *Kanban*. Este incorpora más columnas para representar el estado real de una tarea, políticas explicitas para impedir realizar varios cometidos a la vez en cada etapa e información relevante en todas las tarjetas del tablón. Se compone de 8 columnas:

- ***Backlog***: Contiene los requisitos en forma de historias de usuarios a desarrollar durante el proyecto.

- ***Sprint Backlog***: Aloja las historias de usuarios, tareas, *bugs* y *code spikes*[3] que se trabajan durante un determinado *sprint*.

- ***To Do***: Lista de elementos priorizados en función del orden en el que se implementan.

- ***Doing***: Elementos que se están desarrollando durante esa iteración.

- ***Code Review***: Una vez concluido el desarrollo de una historia de usuario se revisa y refactoriza.

- ***Testing***: En esta etapa se comprueba el desarrollo realizado de manera conjunta, comprobando que este no afecta a las tareas realizadas en otras iteraciones.

[3] Tareas de investigación para resolver un determinado problema.

- ***Done***: Elementos concluidos durante el *sprint*.

- ***Done Def***: Lista de elementos validados por el *product owner* tras cada *sprint review*.

Cada tarjeta dentro del tablero incorpora información útil como el código y nombre de la historia de usuario o tarea, una etiqueta de la iteración a la que pertenece, la fecha de finalización y una descripción.

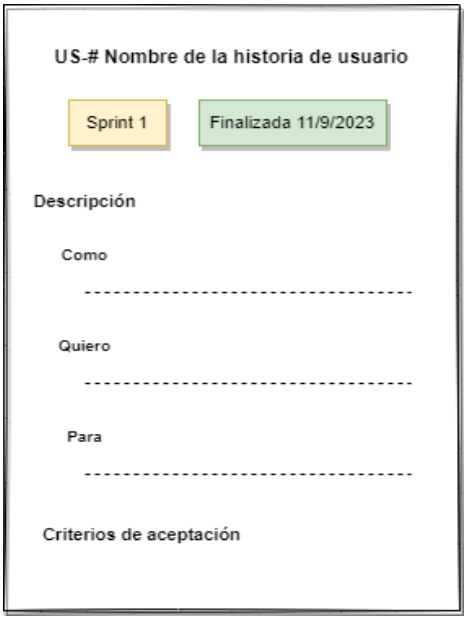

Figura 1: Prototipo de Tarjeta Kanban

2.1.2. Desarrollo del producto

Existen varias técnicas para la resolución de problemas y la innovación de productos que pueden ser aplicadas en diferentes ámbitos. Durante la etapa de análisis de la solución investigué las metodologías en busca de la mejor opción para llevar a cabo este proyecto de manera satisfactoria. A continuación, detallo algunas de las opciones valoradas:

- ***Design Thinking***

  Es un proceso de resolución de problemas creativa; está orientado a la innovación centrada en las necesidades de las personas y las mezcla con la tecnología y los requisitos de negocio.
  Design Thinking se compone de una serie de etapas basadas en un conjunto de ideales, por ejemplo, centrarse en las personas, experimentar con las ideas, aprender del fracaso y la empatía.

Etapas del modelo:

- **Empatizar**: Se centra en entender a las personas que utilizarán nuestro producto, es importante conocer sus experiencias, motivaciones limitaciones y necesidades.

- **Definir**: Se organiza la información obtenida y se definen los principales problemas de los usuarios, esta etapa será de gran ayuda para idear posibles soluciones.

- **Idear**: Tras la identificar los problemas y teniendo presente las necesidades de los usuarios, se generan múltiples ideas que puedan resultar una posible solución a los problemas.

- **Prototipar**: A partir de las ideas se construyen prototipos que permitan llegar una solución final.

- **Evaluar**: En esta etapa se involucra a los usuarios con el objetivo de valorar las ideas y los prototipos, obteniendo la retroalimentación necesaria para validar el progreso realizado o cambiar el planteamiento de la solución propuesta.

- ***Lean Startup***

  Es una metodología centrada en el desarrollo de empresas y productos, cuyo objetivo es acortar los ciclos de desarrollo con el fin de descubrir si el producto o plan de negocio es viable. Se basa en la construcción de un MVP (Producto Mínimo Viable) que se evalúa para obtener información que permita validar el aprendizaje para poder aplicarlo en futuras iteraciones.

Existen otras técnicas que no se han incluido en este análisis debido a su similitud con *Design Thinking* como *Human-Centered Design* (HCD) o aquellas centradas en procesos de negocio como *Six Sigma*[4].

Decidí unir las mejores partes de las metodologías *Design Thinking* y *Lean* para desarrollar la aplicación; en las primeras etapas del proyecto conocí a los usuarios del CRAI y la gestión que realizan sobre el aplicativo de señalización digital. Tras varias reuniones donde se mostraron las carencias de la solución existente, definimos los objetivos y comencé la etapa de elaboración de ideas.

Después adopté el enfoque de *Lean* para crear productos mínimos viables que serían evaluados cada dos semanas según el enfoque ágil que habíamos acordado. Posteriormente, gracias a la retroalimentación proporcionada por mi tutor y el *product owner* de la aplicación, pudimos alcanzar los objetivos propuestos.

[4] Está técnica se enfoca en un análisis estadístico y sistemático para la resolución de problemas, su objetivo es mejorar la calidad de los procesos reduciendo los defectos y aumentando la eficiencia y la eficacia.

2.2. Planificación

Para llevar a cabo este proyecto se han requerido cinco meses de trabajo. A continuación, se describen las fechas importantes ordenadas por iteraciones de desarrollo:

TAREA	COMENTARIO	INICIO	FIN	DURACIÓN EN DÍAS
Toma de contacto	Valoración de la oferta del TFG.	**22/09/2022**	**29/09/2022**	**2**
Reunión Inicial	Reunión con el tutor del TFG para conocer el proyecto	22/09/2022	22/09/2022	1
Reunión Inicial con los usuarios	Reunión con los usuarios para ver sus necesidades y comprender el uso que hacen de la primera versión del producto.	29/09/2022	29/09/2022	1
Análisis		**07/08/2023**	**19/08/2023**	**13**
Estudio previo	Se analizan las soluciones anteriores en busca de puntos de mejora.	07/08/2023	09/08/2023	3
Análisis de requisitos	Se reunen los requisitos funcionales descritos por los usuarios y se aportan nuevos no funcionales.	10/08/2023	11/08/2023	2
Construcción de las primeras historias de usuario	Se desglosan las épicas en historias de usuario más pequeñas	12/08/2023	15/08/2023	4
Análisis de la arquitectura	Se investigan las posibles soluciones que se pueden aportar y que factores tecnólogicos formarían parte de ellas.	16/08/2023	19/08/2023	4
Sprint 1		**22/08/2023**	**15/09/2023**	**26**
Creación de repositorios y proyectos		22/08/2023	24/08/2023	3
US-1 Inicio de sesión en la aplicación	Se desarrolla la página de inicio de sesión y los modelos en Django necesarios para crear una cuenta y poder iniciar sesión con ella.	24/08/2023	28/08/2023	5
US-2 Añadir contenido a la aplicación	Se implementa la funcionalidad de añadir archivos multimedia a la aplicación.	01/09/2023	11/09/2023	11
US-3 Visualizar la lista de archivos	Se desarrolla en la página de la biblioteca la vista para los archivos añadidos	14/09/2023	17/09/2023	4
US-4 Búsqueda de archivos por nombre	Se añade la funcionalidad de búsqueda de archivos por nombre	18/09/2023	19/09/2023	2
Sprint Review		15/09/2023	15/09/2023	1
Sprint 2		**19/09/2023**	**29/09/2023**	**16**
BUGFIX: Resolver los problemas con espacios en el nombre del fichero	Se reformula la manera de guardar los archivos, se opta por un identificador único(UUID)	19/09/2023	19/09/2023	1
US-5 Subir múltiples archivos a la plataforma	Se implementa la carga masiva de archivos	19/09/2023	19/09/2023	1
US-6 Visualizar los detalles de un fichero		19/09/2023	20/09/2023	2
US-7 Crear listas de reproducción		21/09/2023	24/09/2023	4
US-8 Visualizar las listas creadas		25/09/2023	25/09/2023	1
US-9 Filtrar una lista de reproducción por nombre		25/09/2023	25/09/2023	1
US-10 Visualizar el contenido de una lista de reproducción		26/09/2023	27/09/2023	2
US-11 Editar una lista de reproducción		27/09/2023	28/09/2023	2
US-12 Eliminar una lista de reproducción		28/09/2023	28/09/2023	1
Sprint Review		29/09/2023	29/09/2023	1
Sprint 3		**30/09/2023**	**13/10/2023**	**13**
US-13 Reproducir una lista en modo secuencial		30/09/2023	03/10/2023	4
US-14 Programar reproducción de contenidos		04/10/2023	07/10/2023	4
US-15 Visualizar las listas de reproducción programadas		08/10/2023	08/10/2023	1
US-16 Editar la programación de una lista de reproducción		08/10/2023	08/10/2023	1
US-17 Eliminar la programación de una lista de reproducción		09/10/2023	09/10/2023	1
US-18 Renombrar los ficheros añadidos a la aplicación		10/10/2023	10/10/2023	1
Sprint Review		13/10/2023	13/10/2023	1

Tabla 1: Planificación del proyecto en las primeras etapas.

TAREA	COMENTARIO	INICIO	FIN	DURACIÓN EN DÍAS
Sprint 4		**16/10/2023**	**27/10/2023**	**11**
US-19 Eliminar un fichero		16/10/2023	17/10/2023	2
US-20 Prioridad Estrellas	Implementar un sistema de estrellas para definir la prioridad de un archivo	18/10/2023	25/10/2023	8
Sprint Review		27/10/2023	27/10/2023	1
Sprint 5		**28/10/2023**	**10/11/2023**	**13**
US-21 Reproducir una lista aleatoriamente		28/10/2023	05/11/2023	9
US-22 Reordenar los ficheros dentro de una lista de reproducción		06/11/2023	06/11/2023	1
US-23 Permitir la ordenación de ficheros por orden alfabético		07/11/2023	07/11/2023	1
US-24 Actualizar las etiquetas de un fichero		08/11/2023	08/11/2023	1
Sprint Review		10/11/2023	10/11/2023	1
Sprint 6		**11/11/2023**	**24/11/2023**	**10**
BUGFIX: Reproducción de vídeos		11/11/2023	11/11/2023	1
BUGFIX: Modificar archivos		12/11/2023	12/11/2023	1
US-25 Añadir varios ficheros a una lista de reproducción		15/11/2023	15/11/2023	1
US-26 Añadir archivos a la biblioteca desde la creación de lista		16/11/2023	18/11/2023	3
US-27 Identificar la lista en reproducción		21/11/2023	21/11/2023	1
US-28 Crear nuevos usuarios		22/11/2023	23/11/2023	2
Sprint Review		24/11/2023	24/11/2023	1
Sprint 7		**26/11/2023**	**12/12/2023**	**10**
US-29 Listar los usuarios creados		26/11/2023	26/11/2023	1
US-30 Implementar roles de usuario		28/11/2023	28/11/2023	1
US-31 Cambiar Contraseña	Permitir que los usuarios puedan cambiar su contraseña, los administradores podrán modificar la contraseña de cualquier otro usuario	02/12/2023	02/12/2023	1
US- 32 Modificar la duración de una imagen		02/12/2023	02/12/2023	1
US-33 Repetir los archivos con mayor frecuencia		03/12/2023	03/12/2023	1
BUGFIX Caducidad del token		07/12/2023	07/12/2023	1
BUGFIX Avisar de una lista programada antes de reproducir una nueva		08/12/2023	08/12/2023	1
US-34 Editar usuario	Los usuarios podrán cambiar su nombre, correo y los administradores podrán cambiar su rol.	09/12/2023	09/12/2023	1
US-35 Eliminar usuarios		09/12/2023	09/12/2023	1
Sprint Review		12/12/2023	12/12/2023	1
Sprint 8		**16/12/2023**	**19/01/2024**	**6**
US-36 Visualizar el fichero que se está reproduciendo		16/12/2023	16/12/2023	1
US-37 Apagar y reiniciar el servidor		19/12/2023	19/12/2023	1
US-38 Programar el apagado del dispositivo		20/12/2023	20/12/2023	1
Alojar los archivos en una memoria USB		09/01/2024	09/01/2024	1
Colocar una lista por defecto		10/01/2024	10/01/2024	1
Sprint Review		19/01/2024	19/01/2024	1
Memoria		**22/08/2023**	**15/01/2024**	**147**

Tabla 2: Planificación del proyecto en las etapas finales.

La elaboración de este proyecto se ha compaginado con un trabajo a jornada completa, por lo cual la cantidad de días reflejados en el gráfico no se corresponden al número de horas invertidos en exclusiva al TFG, es por ello por lo que una tarea compleja abarca más días de lo habitual.

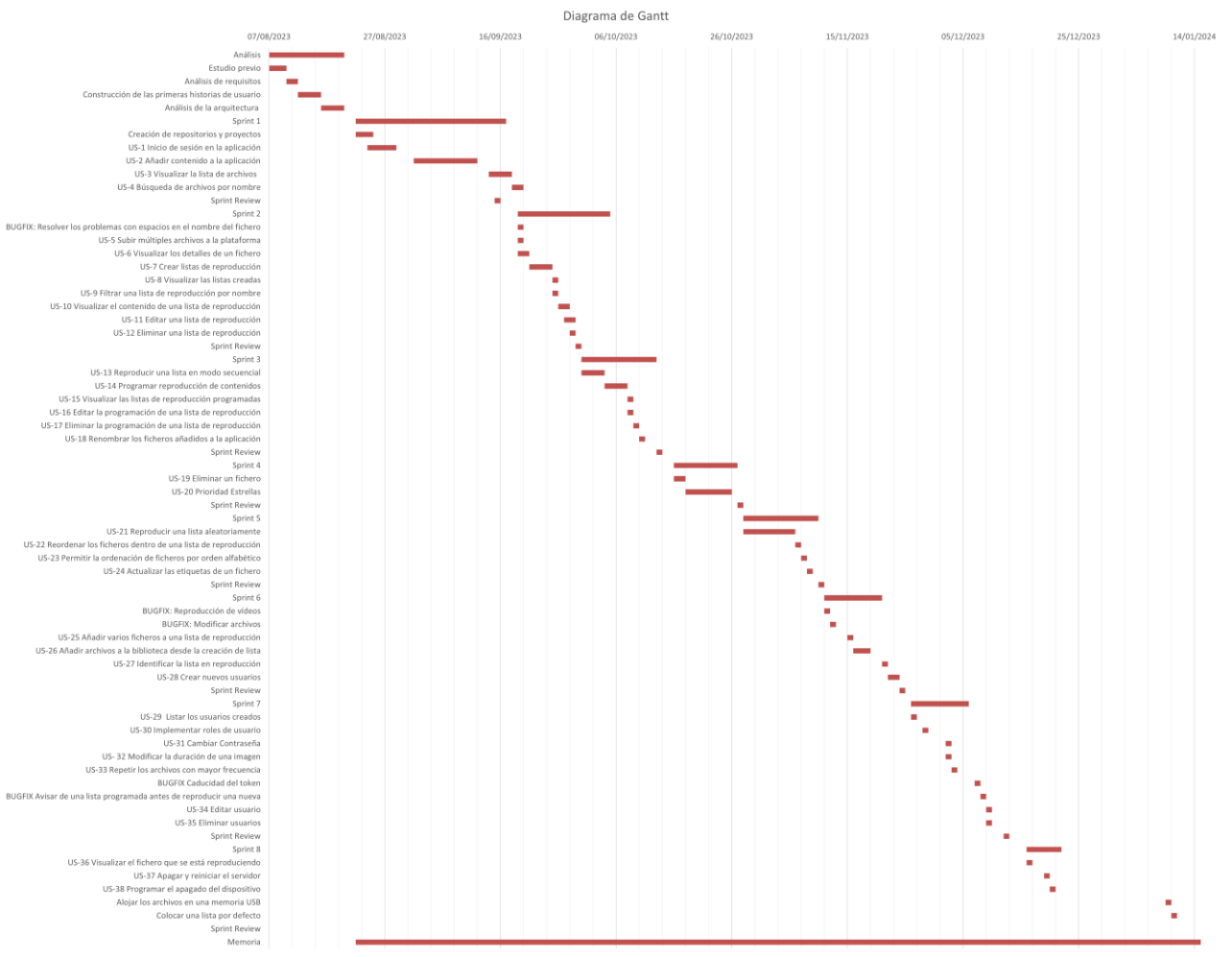

Gráfico 1: Diagrama de Gantt.

2.3. Presupuesto

Dado que esta solución se ha realizado sin ánimo de lucro solo se contempla el coste del *hardware* necesario para llevarlo a cabo, este presupuesto refleja los costes que realizó el CRAI para la primera versión del producto.

Placa Raspberry Pi y accesorios

Descripción	Unidades	Precio	Total
Raspberry Pi 4 Model B 8GB RAM	1	75,20	75,20
Caja disipadora con ventilador dual	1	11,95	11,95
Tarjeta microSD XC Kingston 64gb CL10 100MB/s	1	9,95	9,95
Cable mHDMI	1	3,95	3,95
Fuente de alimentación USB-C 5v3A oficial Negra	1	7,95	7,95
Cable ethernet (categoría 6) 1M	1	2,55	2,55
		Subtotal	111,55
		IVA 21%	23,42
		Total	**134,97 €**

Tabla 3: Presupuesto de la placa Raspberry PI.

Televisor

Descripción	Unidades	Precio	Total
LG Ultra HD TV 4K, 108cm/43 + soporte	1	437,7	437,7
		Subtotal	437,7
		IVA 21%	116,4
		Total	**554.10 €**

Tabla 4: Presupuesto del televisor.

Instalación eléctrica

Descripción	Unidades	Precio	Total
Instalación eléctrica	1	95,03	95,03
		Subtotal	95,03
		IVA 21%	25,27
		Total	**120,3 €**

Tabla 5: Presupuesto de la instalación eléctrica.

Adecuación de la red

Descripción	Unidades	Precio	Total
Cableado de red y roseta	1	203,34	203,34
		Subtotal	203,34
		IVA 21%	54,06
		Total	**257,40 €**

Tabla 6: Presupuesto de la instalación de red.

Otros

Descripción	Unidades	Precio	Total
Programador digital COATI, modelo AF132075	1	15,72	15,72
Velcro	1	4,34	4,34
		Subtotal	20,06
		IVA 21%	5,34
		Total	**25,40 €**

Tabla 7: Otros conceptos.

El coste total del material sería 1.092,17 € IVA incluido.

3. Diseño

3.1. Arquitectura de la aplicación

La arquitectura de la aplicación se compone de diversos contenedores Docker que interactúan entre sí para ofrecer las funcionalidades descritas anteriormente, están alojados dentro de una placa Raspberry Pi. A continuación, se detalla en profundidad cada uno de los componentes de la aplicación desde un punto de vista cliente-servidor.

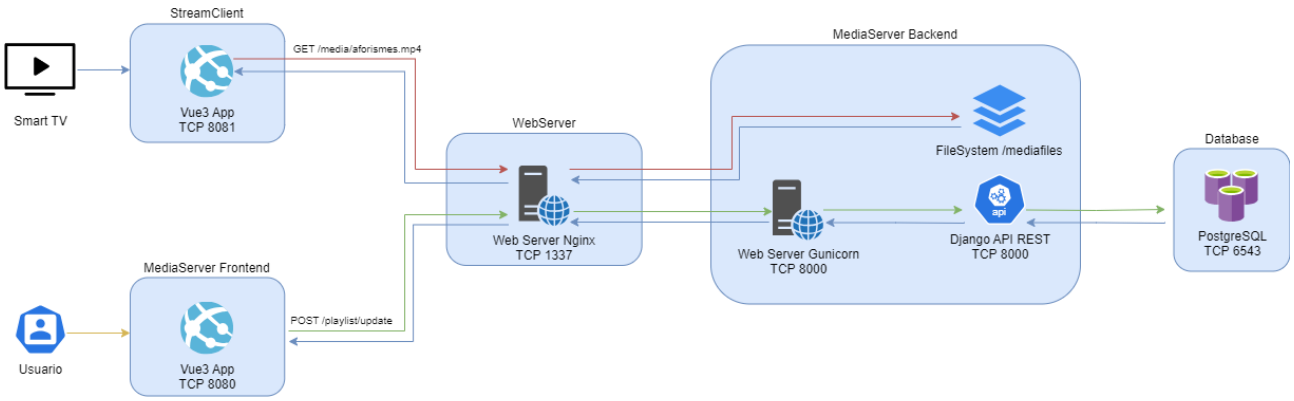

Figura 2: Diagrama de la arquitectura de los contenedores Docker

3.1.1. Frontend

Se compone de dos aplicaciones web desarrolladas mediante Vue3, JavaScript, HTML y Tailwind CSS para la gestión y visualización del contenido multimedia, además en el entorno productivo se crea un contenedor Docker a partir de una imagen de Nginx para distribuir la web a los usuarios.

- **StreamClient**

  Esta aplicación permite reproducir los vídeos e imágenes subidos por los usuarios a la plataforma; contiene los siguientes componentes[5] Vue:

  - **MyPlayer.vue**

    Se encarga de obtener la información necesaria para reproducir el siguiente archivo de la cola de reproducción. Posteriormente, envía al componente MyPlayWindow.vue la información para

[5] Un componente de Vue es un elemento en el que se encapsula código HTML, CSS y JavaScript que puede ser reutilizable en otros lugares de la aplicación.

que sea reproducido. Al finalizar cada archivo, se repite la consulta al servidor de *backend* para obtener el siguiente elemento, gracias a esta acción se consigue obtener siempre la información actualizada. Los cambios realizados en la cola de reproducción se verán reflejados al finalizar cada archivo.

- **MyPlayWindow.vue**

  Este componente recibe la información del fichero a reproducir y en función de su tipo, imagen o vídeo, inserta el elemento HTML necesario para su reproducción.

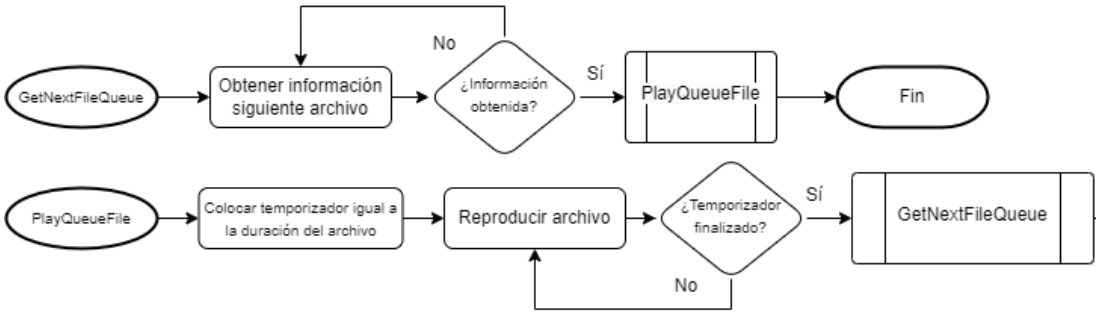

Figura 3: Diagrama de flujo del proceso de reproducción de archivos.

- **MediaServer Frontend**

  Es el núcleo principal de la solución, está construida como una aplicación SPA (*Single-Page Application*) para mejorar la interactividad, comunicación y navegación entre los distintos componentes.

  Se compone de seis vistas[6] que forman las distintas páginas por las que se puede navegar, esto se consigue gracias a Vue Router, una librería de código abierto que permite la transición por los diferentes apartados de la web sin necesidad de recargar la página.

[6] Actúan como páginas que integran los diferentes componentes de la aplicación, además permiten definir la jerarquía de navegación del sitio web.

Cada vista incorpora diversos componentes que se integran entre sí para construir la interfaz, algunos de ellos están hechos para ser reutilizados en diferentes partes del código como pueden ser los diálogos de confirmación, notificaciones, botones o textos alternativos.

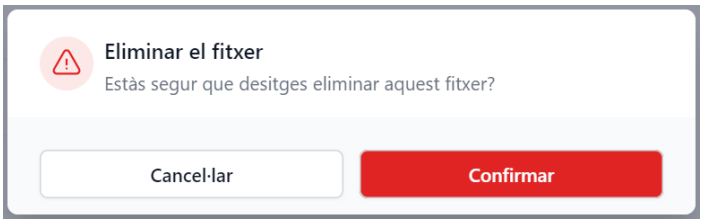

Figura 4: Ejemplo de un componente reutilizable de la aplicación

Otros aspectos importantes son el uso de librerías como Axios para realizar las llamadas al *backend* o Flowbite, una librería de código abierto que utiliza el *framework* de Tailwind CSS e incorpora componentes de interfaz de usuario que permiten construir páginas webs rápidamente.

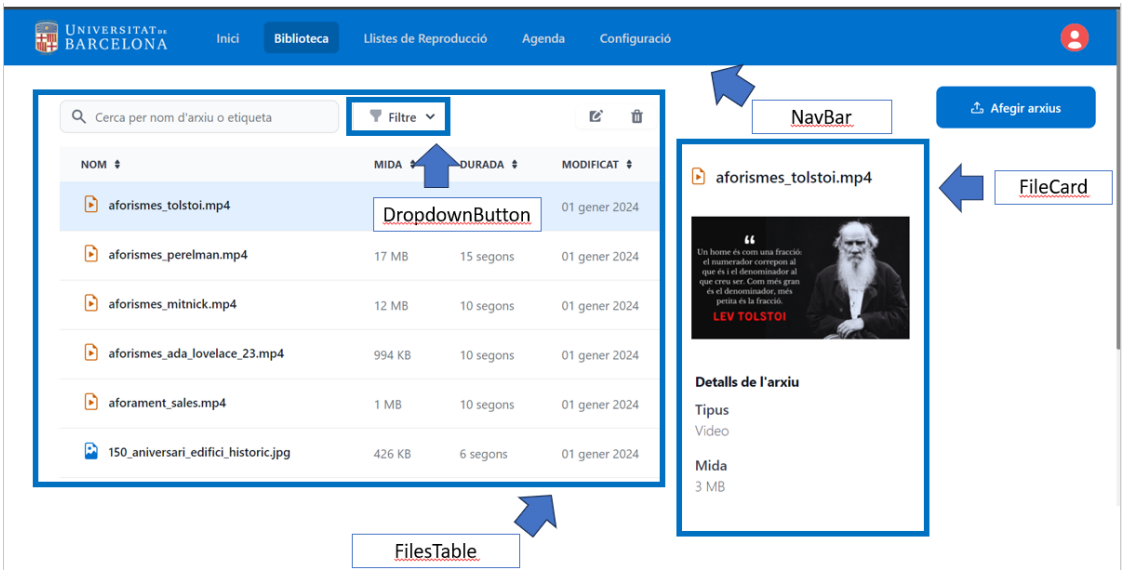

Figura 5: Página de la biblioteca con sus respectivos componentes.

3.1.2. Backend

- **MediaServer Backend**

  Es una API[7] REST construida en Python mediante el *framework* de Django para gestionar las peticiones del frontal de la aplicación, se encarga de almacenar y leer la información alojada en la base de datos.

  Django permite gestionar la autenticación y autorización de los usuarios. Además, incorpora los modelos de datos que permiten la abstracción de clases de Python en tablas de una base de datos relacional, este modelo de programación se conoce como ORM.

  El código realizado se organiza dentro de un proyecto que incluye diversas aplicaciones de Django, cada una incorpora los modelos, vistas, test y funciones de serialización que se encargan de transformar las filas obtenidas de la base de datos en objetos de tipo JSON[8].

  En total existen cinco aplicaciones correspondientes a las entidades básicas de la solución:

  - *Files*: Contiene toda la lógica de gestión de archivos de la aplicación.
  - *FileQueue*: Gestiona la cola de reproducción.
  - *PlayLists*: Permite la creación, edición, eliminación y reproducción de listas.
  - *Schedule*: Controla las *playlists* programadas y el apagado automático del servidor.
  - *Users*: Gestiona la creación, modificación, eliminación y autenticación de los usuarios.

  Cuando el proyecto se despliega en el entorno de producción se utiliza Gunicorn, una interfaz de pasarela para gestionar las peticiones HTTP de los demás elementos de la solución.

- **WebServer**

  Es un contenedor Docker que aloja un servidor web de Nginx en su versión 1.25, actúa como proxy inverso reenviando las peticiones de los usuarios al servidor web de Gunicorn y distribuyendo de forma directa los archivos multimedia. Esto se debe a la creación de varios volúmenes en Docker que permiten a los contenedores compartir archivos entre sí y garantizan la persistencia de los datos.

---

[7] *Application Programming Interface* o API, es un conjunto de mecanismos que permiten a dos aplicaciones comunicarse entre sí.

[8] *JavaScript Object Notation* es un formato de texto que facilita el acceso, almacenamiento e intercambio de datos.

- ***Database***

La base de datos de la aplicación está diseñada sobre PostgreSQL, un sistema gestor de base de datos relacional orientado a objetos que tiene un excelente rendimiento respecto a otras alternativas como MySQL o MariaDB. Aunque Django incorpora una base de datos de tipo SQL Lite, decidí separar la capa de persistencia de la información en un contenedor independiente para aumentar la disponibilidad y escalabilidad de la solución.

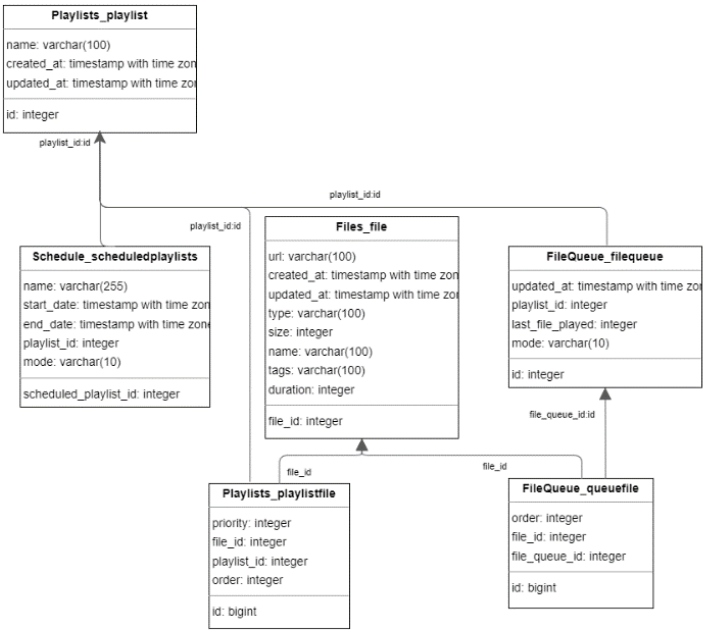

Figura 6: Diagrama 1 entidad relación de la base de datos

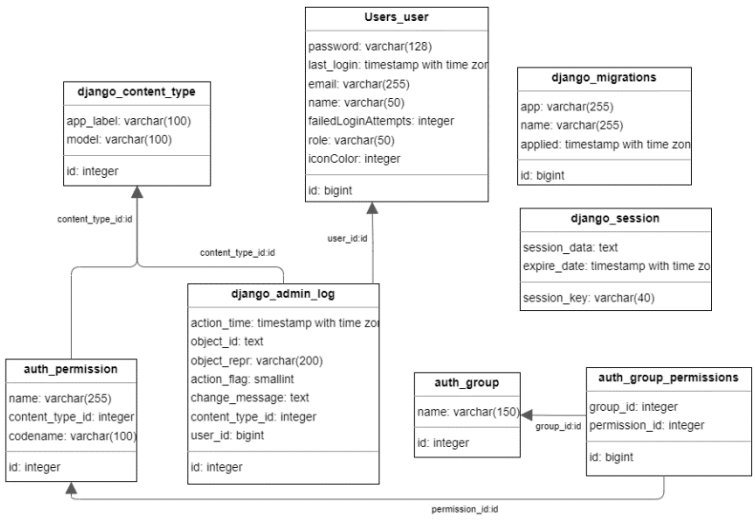

Figura 7: Diagrama 2 entidad relación de la base de datos.

3.2. Prototipos

En cada iteración del desarrollo se realizaron diferentes prototipos de baja fidelidad para construir las diferentes funcionalidades de la página web.

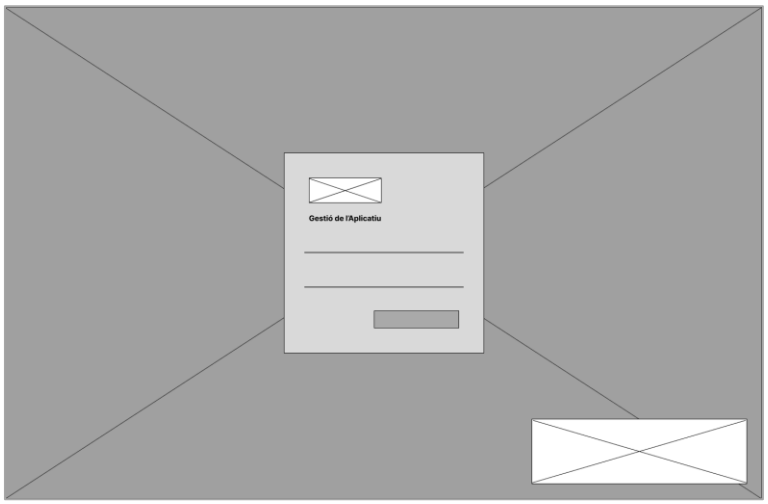

Figura 8: Página de inicio de sesión.

Figura 9: Prototipo de la pantalla de inicio.

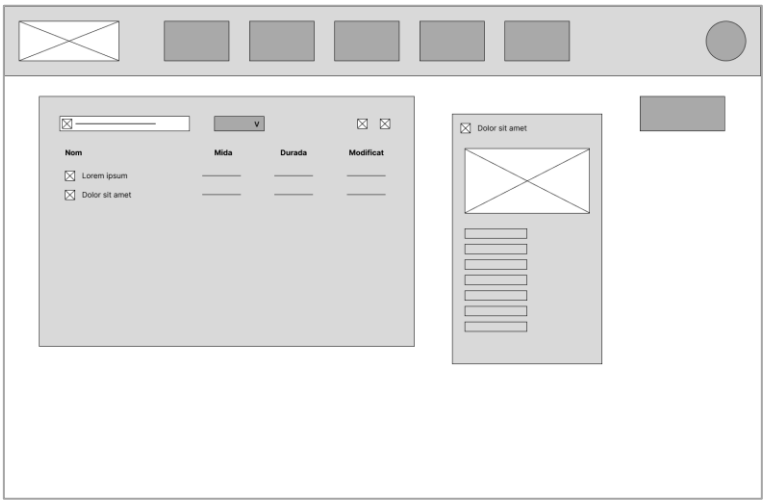

Figura 10: Prototipo de la biblioteca.

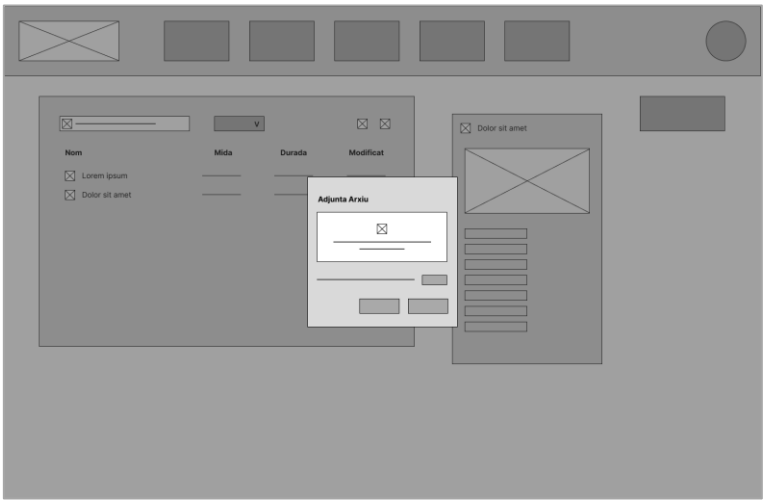

Figura 11: Adjuntar archivo.

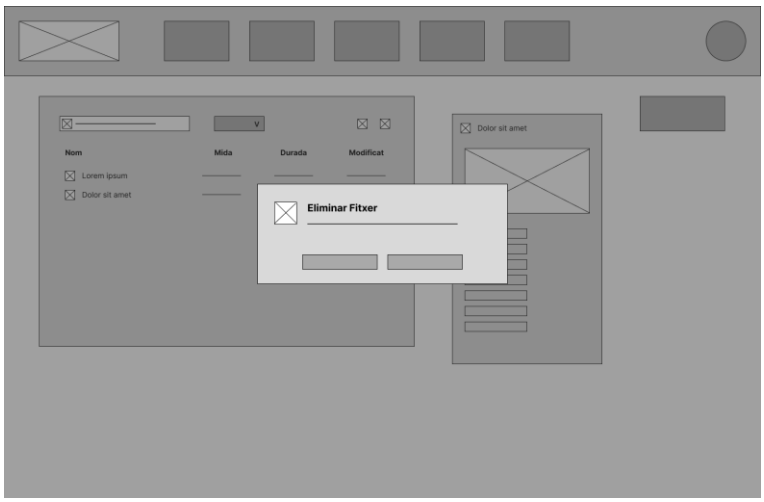

Figura 12: Eliminar archivo.

Figura 13: Listas de reproducción.

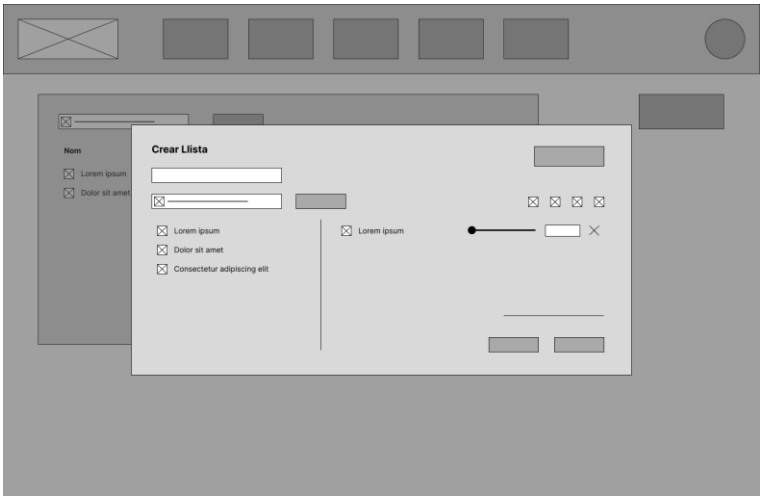

Figura 14: Menú de creación y edición de listas de reproducción.

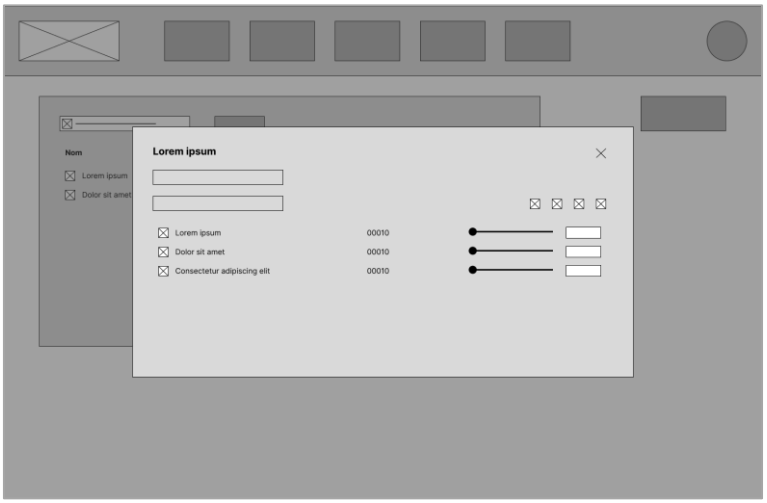

Figura 15: Consultar la información de una lista.

Figura 16: Prototipo de la agenda.

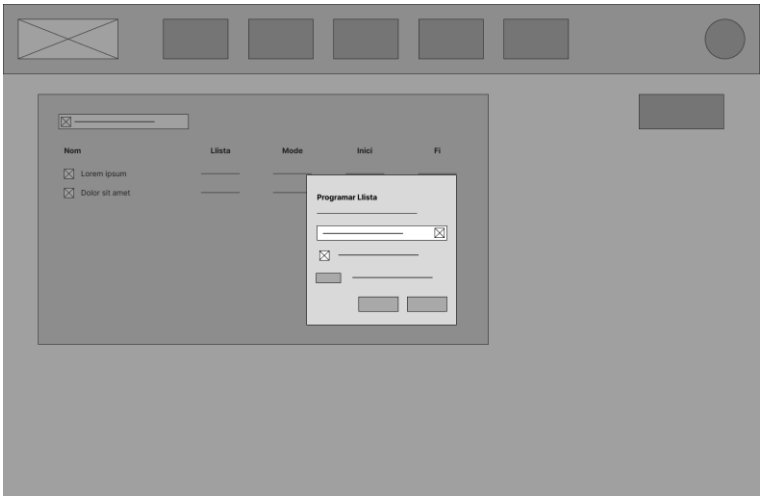

Figura 17: Menú de programación de listas.

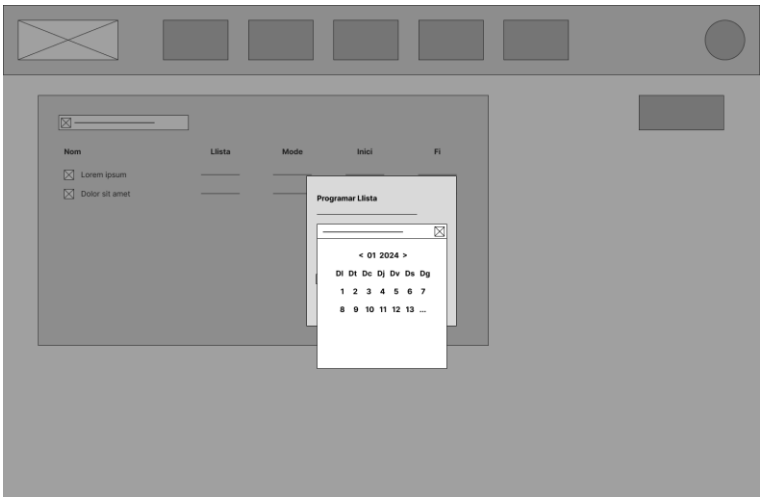

Figura 18: Selección de días.

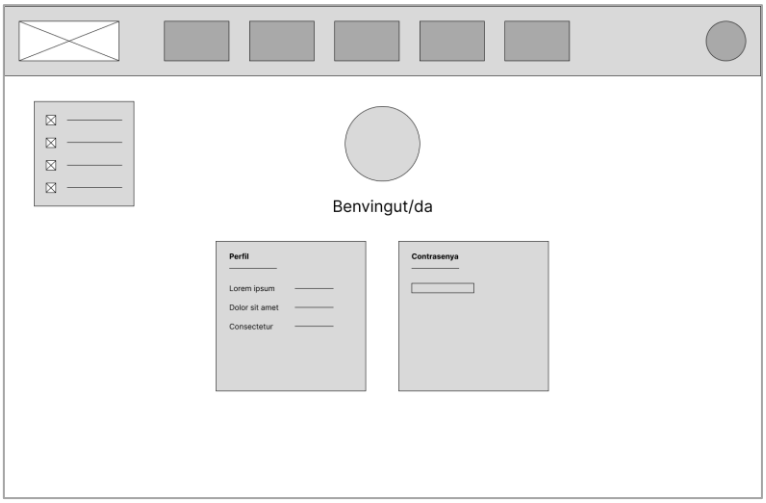

Figura 19: Menú de configuración

Figura 20: Gestión de usuarios.

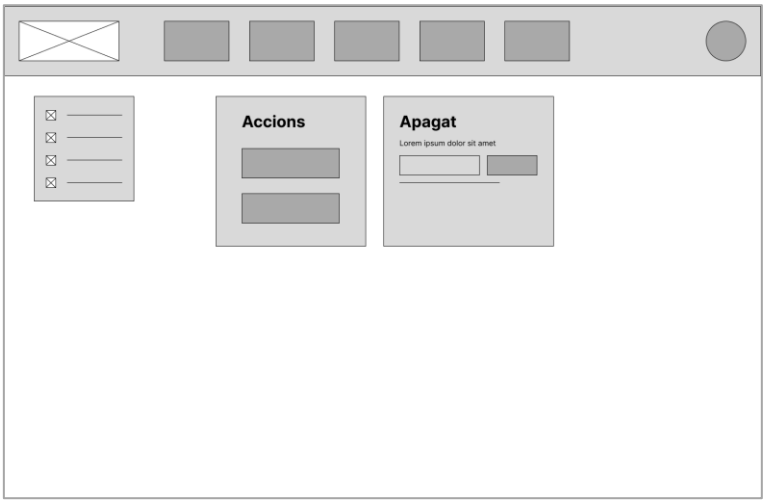

Figura 21: Apagado y reinicio de la placa Raspberry

4. Implementación

Para llevar a cabo este proyecto he utilizado un conjunto de herramientas de desarrollo, entre ellas se encuentran:

- **PyCharm 2023 Edición Profesional**

  Es un entorno de desarrollo integrado comúnmente conocido como IDE, utilizado para programar aplicaciones en Python. Ofrece funcionalidades de inspección de código, indicación de errores y refactorización de código automática.

- **WebStorm 2023 Edición Profesional**

  WebStorm al igual que Pycham es un IDE desarrollado por JetBrains para la creación de aplicaciones web basadas en JavaScript y TypeScript, ofrece funcionalidades de finalización y análisis de la calidad del código, refactorización segura y documentación rápida.

- **GitHub**

  Es una plataforma de desarrollo colaborativo utilizada para alojar proyectos mediante el uso del sistema de control de versiones de GIT. Este sistema permite gestionar los cambios que ocurren en el *software* y revertirlos en caso de ser necesario. Para organizar el código de las diferentes aplicaciones he optado por una arquitectura de múltiples repositorios, uno para cada aplicación desarrollada, este enfoque facilita su despliegue y permite operar de manera más sencilla.

- **GitHub Copilot**

  Copilot es una herramienta de Inteligencia Artificial desarrollada por GitHub y OpenAI, se basa en el modelo de lenguaje GPT-3 y asiste a los programadores sugiriendo código en base al trabajo realizado previamente.

5. Resultados

5.1. Usabilidad

Los principios heurísticos son un conjunto de normas conversacionales establecidas para facilitar la comprensión humana de un sistema en busca de una interacción efectiva. Jackob Nielsen sentó precedentes con su obra sobre usabilidad en la que instauró un listado de reglas que actualmente sigue considerándose la más popular. A continuación, ejemplificaré el uso de algunas de ellas en la aplicación construida.

Visibilidad del estado del sistema

Establece que el diseño debe mantener siempre al usuario informado sobre qué ocurre en la interfaz, esto se consigue mediante la provisión de retroalimentación sobre sus acciones en el menor tiempo posible. Un ejemplo de este principio lo encontramos en las notificaciones que confirman las diferentes acciones del usuario.

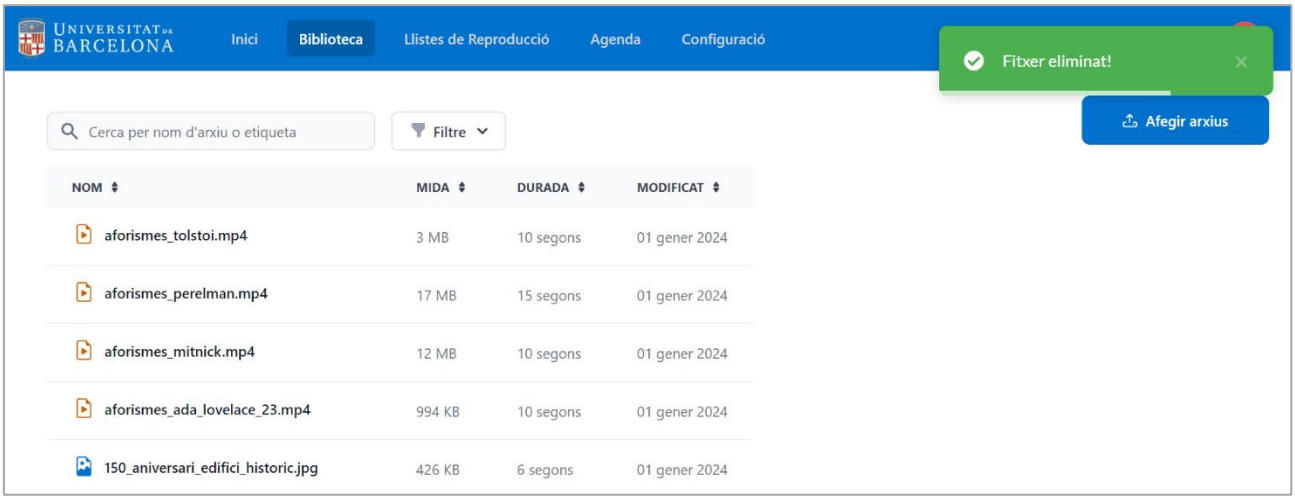

Figura 22: Notificaciones del sistema.

Relación entre el sistema y el mundo real

El sistema debe contener un lenguaje que utilice palabras, frases o conceptos que resulten familiares al usuario, por ejemplo, en el apartado de la biblioteca se emplean términos que recuerdan a los sistemas de gestión de archivos de los sistemas operativos.

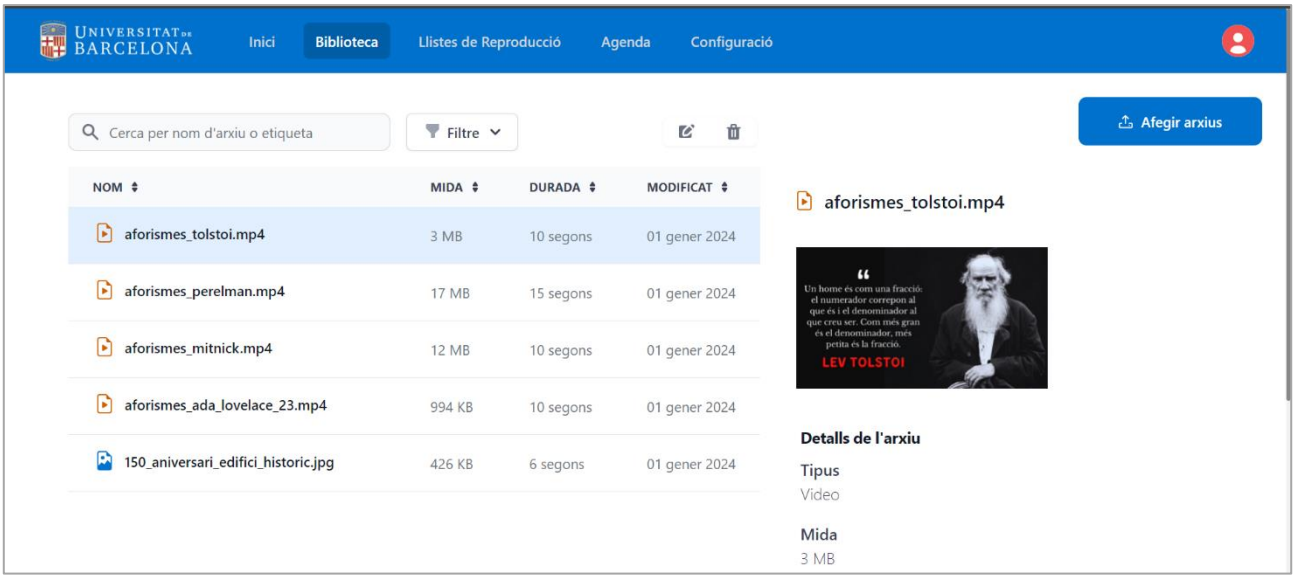

Figura 23: Biblioteca de archivos.

Control y libertad del usuario

Los usuarios deben tener la posibilidad de cancelar las acciones que estén realizando de forma sencilla, manteniendo el control sobre el sistema para evitar quedarse atrapados y sentirse frustrados. En la aplicación todo formulario o menú contiene un botón para cancelar la acción y volver al apartado anterior.

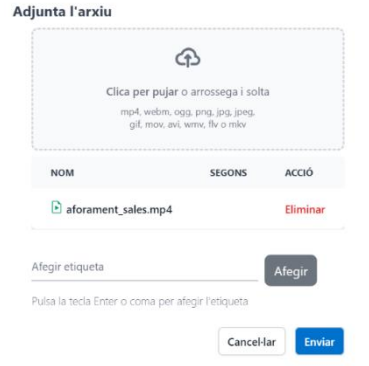

Figura 24: Menú de carga de archivos.

Coherencia y estándares

En la aplicación se utilizan iconos y normas de diseño que ayudan al usuario a utilizar el sistema, un ejemplo de coherencia lo encontramos en los botones para crear nuevos elementos, todos ellos tienen el mismo color y se encuentran siempre en el mismo sitio.

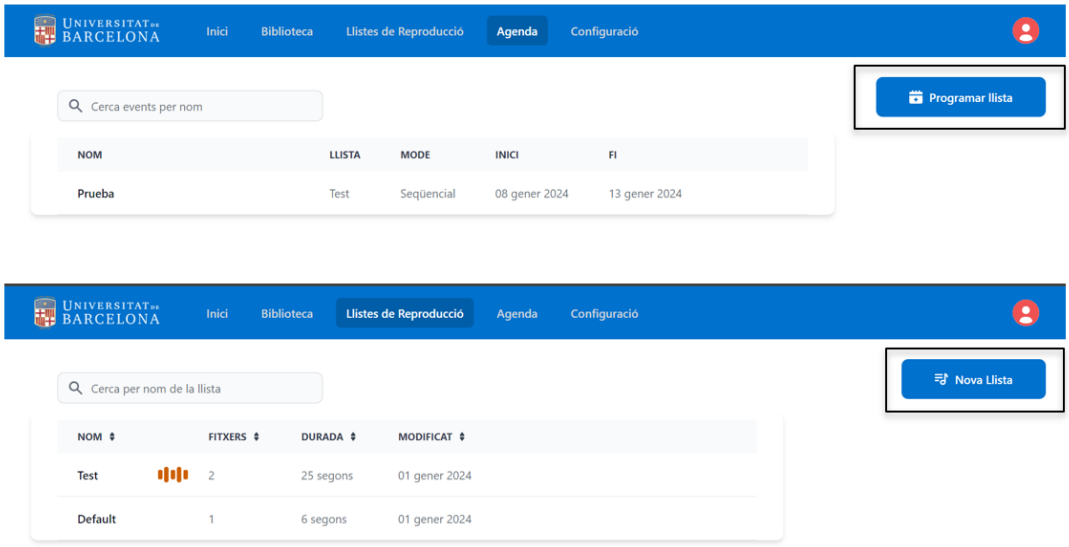

Figura 25: Ejemplos de acciones de crear nuevos elementos.

Prevención de errores

Crear un diseño web que evite que los usuarios cometan errores es clave para mejorar la usabilidad de un sistema; este principio lo he aplicado en apartados como la creación de eventos. En este menú se impide que el usuario coloque una fecha anterior a la actual, evitando así errores y confusiones.

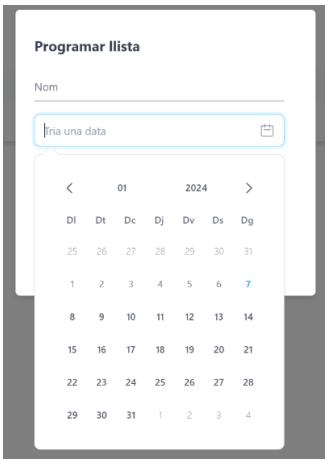

Figura 26: Menú de programación de eventos.

Reconocimiento en lugar de recuerdo

El uso de elementos web o iconos permite al usuario reconocer su funcionalidad, por ejemplo, se ha diseñado un *widget* de reproducción similar a otras plataformas de contenido.

Figura 27: *Widget* de reproducción.

Diseño estético y minimalista

La web se ha construido mediante el uso de elementos minimalistas, mostrando solo información útil para el proceso que se realiza.

Figura 28: Menú de modificación de archivos

Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores

El menú de creación de usuarios ejemplifica el uso de este principio, mostrando en rojo los errores e indicando como mejorar la contraseña para cumplir los estándares de seguridad.

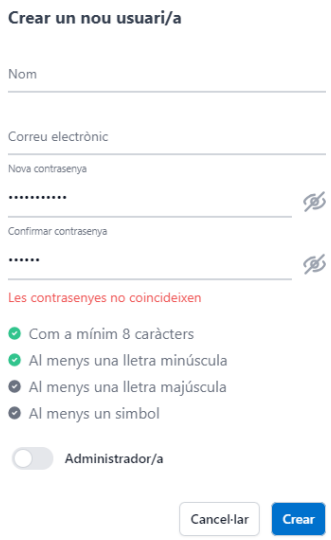

Figura 29: Creación de nuevos usuarios.

5.2. Seguridad

La seguridad informática dentro cualquier proyecto o aplicación es fundamental para la protección de los datos de los usuarios y la reducción de riesgos y amenazas. A continuación, detallaré algunas de las medidas más importantes que se han llevado a cabo para garantizar la seguridad de la aplicación.

Seguridad de red

Docker permite crear redes virtuales que aíslan diversos recursos, permitiendo su acceso a aquellos contenedores que se encuentren en la misma red. Debido a esta característica se ha diseñado la siguiente configuración.

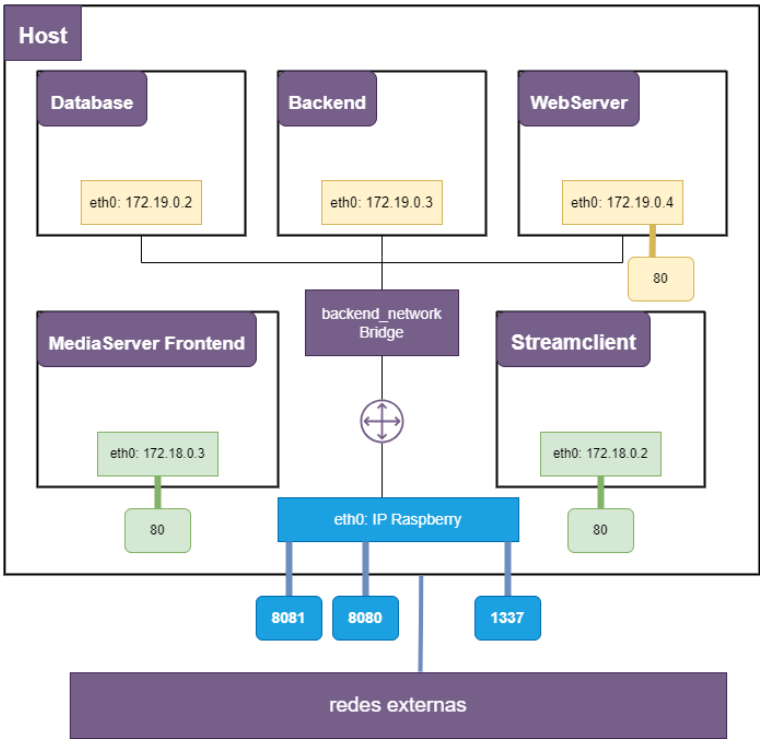

Figura 30: Diagrama de red de los contenedores Docker.

En el diagrama podemos apreciar la existencia de una red llamada *backend_network* que opera en forma de *bridge*[9] aislando los contenedores de la base de datos y el *backend* de la maquina *host*, así como otras redes externas. La placa Raspberry Pi expone los puertos necesarios para que los clientes web se comuniquen con las aplicaciones de gestión y visualización, además del servidor web que actúa de proxy inverso.

[9] Un bridge en Docker permite crear una red privada interna dentro de la máquina *host* para que los contenedores que se encuentren en esta red puedan comunicarse.

También se recomienda la implementación de medidas de seguridad adicionales por parte del departamento TI del CRAI para garantizar el acceso seguro a la plataforma desde el exterior. Es importante que se asegure que los usuarios que acceden al servidor son los gestores de la aplicación, esta medida se puede implementar mediante *Firewalls* intermedios que validen el origen de las peticiones. Por último, se debería aislar a nivel de red la placa Raspberry de cualquier otro sistema que no sea indispensable para el funcionamiento de esta.

Encriptación de datos

Para garantizar la seguridad de la información sensible, se ha habilitado la encriptación de los datos en tránsito y en reposo. Debido al uso del protocolo HTTPS, se cifra la información que envían los usuarios entre las aplicaciones del *frontend* y el *backend.*

Además, se almacena el valor *hash* de las contraseñas en la base de datos mediante el algoritmo argon2id, que supone una mejora significativa respecto a PBKDF2 con SHA-256 en términos de seguridad, resistiendo ataques de canal lateral y de *cracking* de GPU.

Control de acceso basado en roles

Otro aspecto importante de la seguridad de la aplicación es el control de accesos basado en roles (RBAC), que define los privilegios de los diferentes tipos de usuario de la aplicación; existen tres tipos de usuarios que han sido definidos siguiendo los requisitos funcionales del cliente:

- **Administrador global**: Es el usuario por defecto de la plataforma, sirve para configurar los primeros parámetros necesarios de la aplicación y no puede ser eliminado.

- **Administradores**: Tienen el privilegio de gestionar los usuarios de la plataforma en términos de creación, edición y eliminación.

- **Usuario estándar**: Puede hacer uso de todas funcionalidades del aplicativo excepto la gestión de usuarios.

Política de seguridad de las contraseñas

La gestión de las contraseñas de los usuarios se realiza siguiendo las directivas de seguridad del Instituto Nacional de Estándares y Tecnología (NIST). A continuación, se explican algunas de las medidas adoptadas:

- Longitud mínima de 8 caracteres, incluyendo mayúsculas, minúsculas, números y símbolos.
- Posibilidad de mostrar la contraseña durante su creación.
- Almacenamiento del *hash* de la contraseña.
- Se provee una retroalimentación al usuario para indicar los posibles errores cometidos en la creación de la contraseña.
- Limitación del número de intentos erróneos de inicio de sesión. Esta funcionalidad está desarrollada en *backend*; por el momento no se encuentra disponible en el frontal de la aplicación.

La autenticación multifactor es uno de los requisitos no funcionales que, a causa de falta de tiempo, no han podido ser implementados. No obstante, no se descarta el lanzamiento de nuevas versiones de la aplicación que incluyan esta funcionalidad.

5. Conclusión y trabajo futuro

En este proyecto se ha realizado desde cero una aplicación a medida de los requisitos solicitados por el Centro de recursos para el aprendizaje y la Investigación de la Universidad de Barcelona, cumpliendo con todos los objetivos funcionales solicitados.

Durante el trascurso de este último año, he estado poniendo en práctica los conocimientos adquiridos a lo largo de la carrera y profundizando en algunos de ellos para llevar a cabo este proyecto, la experiencia obtenida en el área de desarrollo de aplicaciones web me ha ayudado a complementar mi carrera profesional.

Como trabajo futuro queda pendiente los objetivos secundarios no realizados, en concreto el desarrollo de la autenticación multifactor para brindar más seguridad en el acceso por parte de los usuarios; la posibilidad de modificar los aspectos visuales de la aplicación como logos, fondos e idiomas, permitiendo que otras instituciones puedan utilizarla.

Además, sería interesante brindar opciones para la gestión de copias de seguridad, envío de alertas del sistema y controles multimedia que permitan manipular la reproducción de un archivo.

Anexo 1. Requisitos de instalación

Los requisitos descritos están destinados al sistema operativo de Linux, en concreto a sistemas derivados la distribución Debian. No obstante, la solución puede ser desplegada en cualquier otro sistema que soporte el motor de Docker. Puede obtener más información sobre la instalación en Microsoft Windows o Mac OS en el siguiente enlace.

La instalación del motor de Docker requiere una versión de 64 Bits de las siguientes versiones de Debian:

- Debian Bookworm 12
- Debian Bullseye 11

También es necesario que el servidor disponga de un mínimo de 8GB de RAM y un espacio de 10GB disponibles para alojar los ficheros multimedia y la aplicación. Antes de proceder con la instalación se recomienda desinstalar paquetes no oficiales que puedan entrar en conflicto.

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

Concluida la desinstalación, instale los componentes de Docker necesarios.

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Por último, compruebe que la instalación se haya realizado correctamente.

```
sudo docker run hello-world
```

Anexo 2. Despliegue de la aplicación

Descomprima el archivo con el contenido en la ruta que desee; dentro encontrará varias carpetas y ficheros que deberá modificar para adecuarlo a su entorno.

Modificación de las variables de entorno

En la carpeta principal se encuentran los archivos *.env.pro* y *.env.pro.db* que contienen las variables de entorno de la aplicación, se recomienda modificar los siguientes valores:

- DJANGO_ALLOWED_HOSTS: Deberá cambiar la dirección IP existente y colocar la que corresponda a su servidor.
- SQL_PASSWORD: Modifique la contraseña de la base de datos en ambos archivos.
- EMAIL y PASSWORD: **Se recomienda cambiar las credenciales del usuario local** de la aplicación.

La aplicación incorpora una imagen por defecto que se reproduce cuando la aplicación inicia, si desea cambiarla deberá colocar el archivo en la carpeta /ServerManager/mediafiles y modificar las variables de entorno URL y FILENAME.

Cuando la aplicación ha sido desplegada y se ha comprobado su funcionamiento, puede borrar ambos ficheros.

Cambios en las aplicaciones del *Frontend*

Siempre que se despliegue la aplicación en un nuevo servidor y cambie la dirección IP, deberá modificar tanto las variables de entorno como los archivos:

- /Frontend/src/utils/constants.js
- /StreamClient/src/utils/constants.js

Instalación de certificados SSL

La aplicación incorpora un certificado auto firmado que puede sustituir por el que desee, simplemente deberá cambiar los archivos webserver.crt y webserver.key en la carpeta WebServer, es importante que mantenga la nomenclatura.

Si renueva los certificados deberá volver a desplegar el contenedor para aplicar los cambios.

Despliegue de la aplicación

Para facilitar la puesta en marcha de la solución se ha creado un conjunto de *scripts* que genera los contenedores Docker y configura el arranque automático de la aplicación.

En primer lugar abra el archivo Docker-compose.service alojado en la carpeta raíz del código fuente, modifique la variable *WorkingDirectory* y coloque la ruta donde ha guardado los archivos de la aplicación.

Mediante una consola de comandos con privilegios de administrador, diríjase a la carpeta principal y otorgue permisos de ejecución sobre el script Install.sh.

```
sudo chmod a+x Install.sh
```

Para finalizar ejecute el *script*; una vez haya terminado el despliegue podrá acceder a la página de gestión a través de la URL http://IP_ADDRESS:8080.

Anexo 3. Instrucciones de uso

Gestión de archivos

Inicie sesión en la aplicación con el correo y la contraseña que colocó en el archivo de variables de entorno.

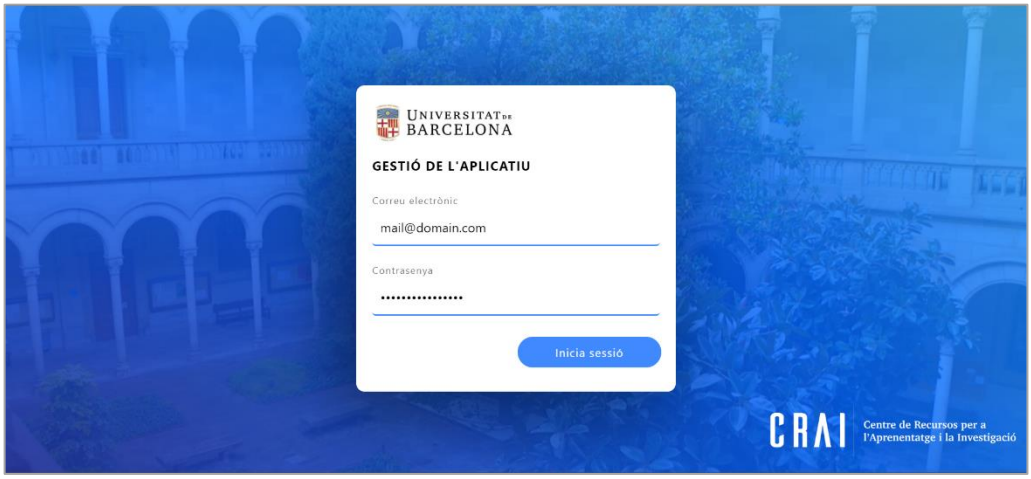

Figura 31: Página de inicio de sesión

Diríjase a la pestaña Biblioteca y pulse el botón *Afegir arxius.*

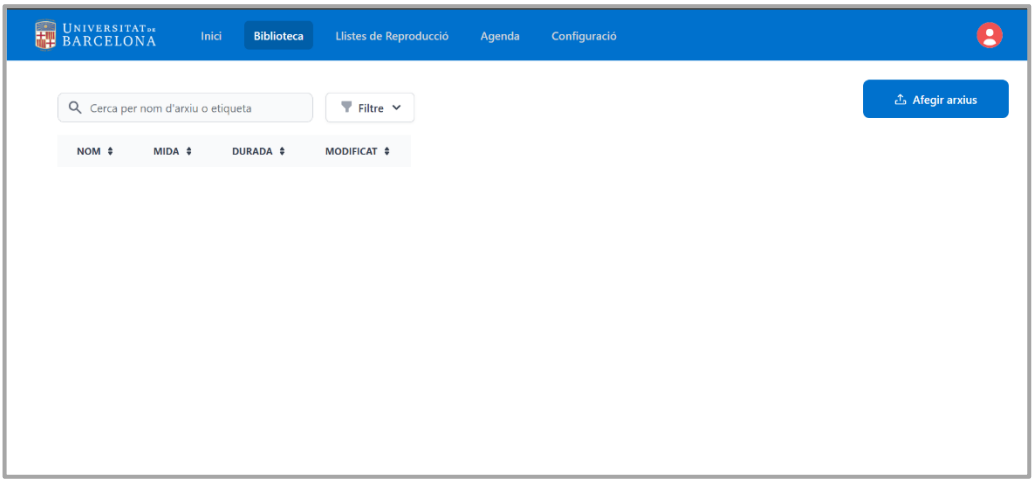

Figura 32: Biblioteca de archivos

Pulse el recuadro gris para seleccionar los archivos o arrastrelos al área indicada, cambie la duración de las imágenes que desee, el valor mínimo es de 6 segundos. También puede colocar las etiquetas que crea conveniente, tenga en cuenta que todos los archivos tendrán la misma etiqueta. Pulse el botón enviar cuando haya concluido.

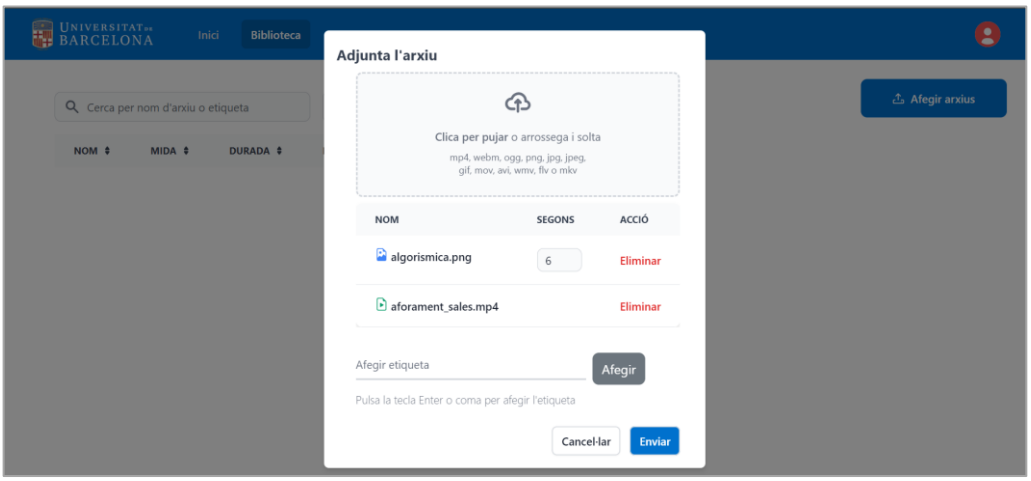

Figura 33: Carga de archivos en la aplicación.

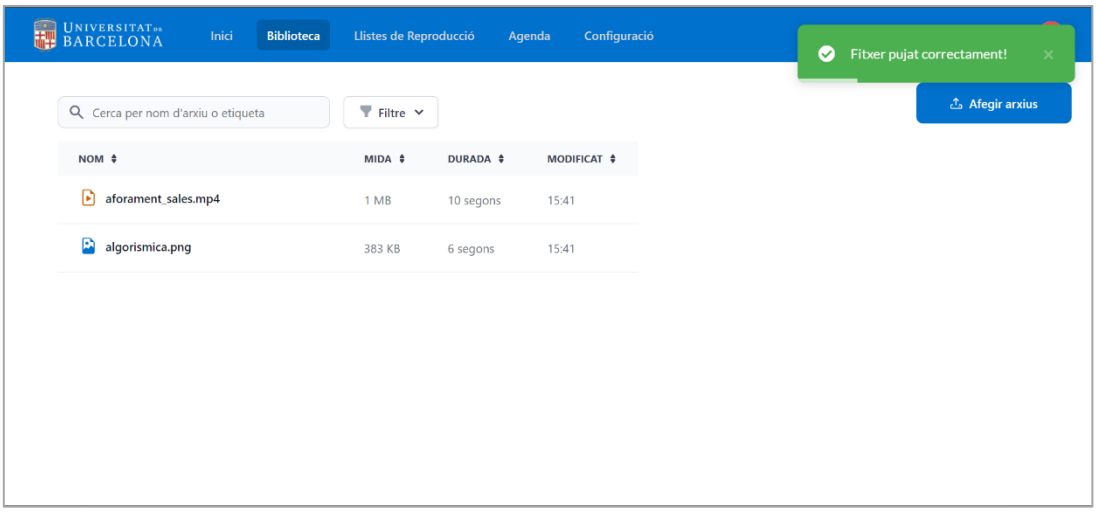

Figura 34: Confirmación de subida de archivos.

Listas de reproducción

Para poder reproducir los archivos subidos es necesario que estén dentro de una lista de reproducción, acceda a la tercera pestaña para crear una nueva lista. Pulse el botón *Nova Llista*.

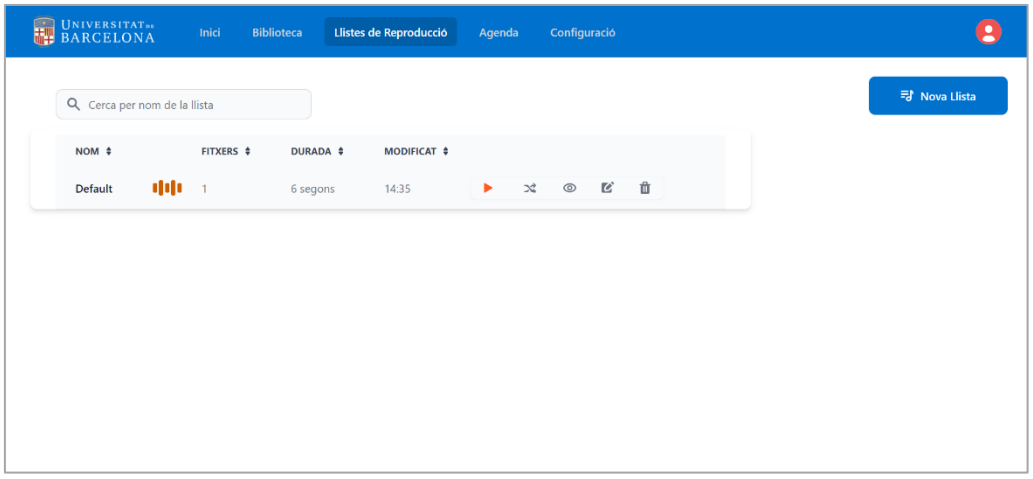

Figura 35: Confirmación de subida de archivos.

Coloque el nombre y seleccione todos los archivos que desee agregar, posteriormente mediante el deslizador o el campo numérico establezca la frecuencia[10] de reproducción de cada archivo, durante la creación puede añadir más ficheros a la biblioteca si lo ha olvidado en el paso anterior, solo deberá pulsar el botón *Afegir arxius.*

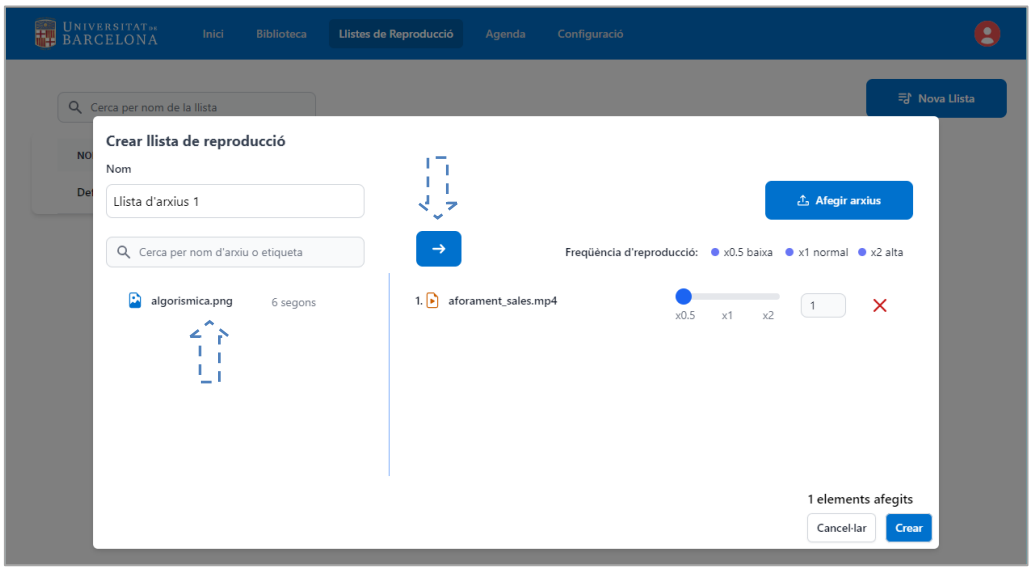

Figura 36: Creación de listas de reproducción.

[10] La frecuencia de reproducción determina cuantas veces se repetirá un elemento cuando la reproducción de la lista se realiza de manera aleatoria, si desea establecer un número concreto puede colocarlo en el campo numérico. El sistema le impedirá colocar una frecuencia superior al número total de archivos para garantizar que los elementos no se repiten de manera consecutiva.

Cuando haya concluido la creación de la lista podrá reproducirla, editarla o eliminarla, para ello deslice el cursor sobre la fila que contiene su lista, a la derecha aparecerán las operaciones que puede realizar.

Existe una lista de reproducción por defecto que se muestra siempre que no haya otra programada, esta lista no puede ser eliminada, pero puede modificar su contenido y su nombre.

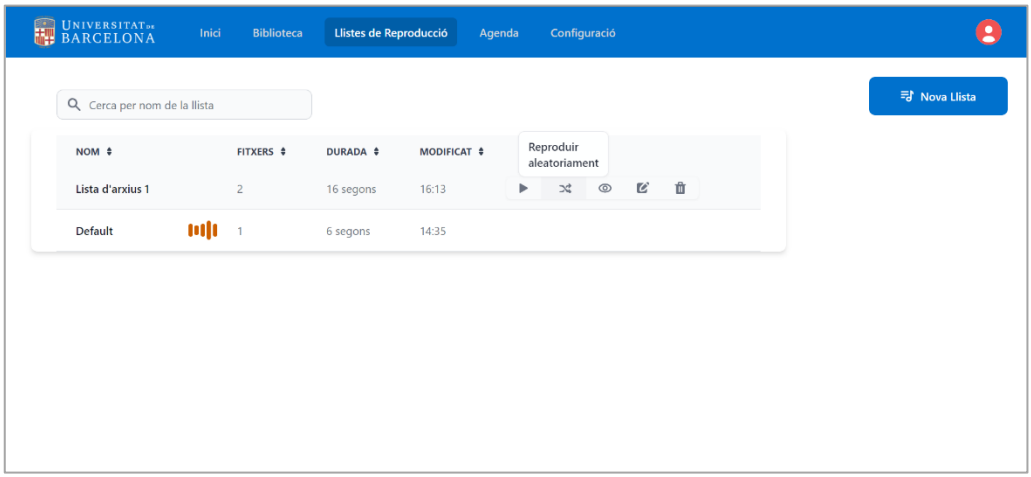

Figura 37: Gestión de una lista de reproducción.

Creación de eventos

Si desea programar la reproducción de una lista durante un período especifico de tiempo, acceda a la pestaña agenda y pulse el botón *Programar llista* para crear un nuevo evento. Coloque el nombre que desee y seleccione la duración en días.

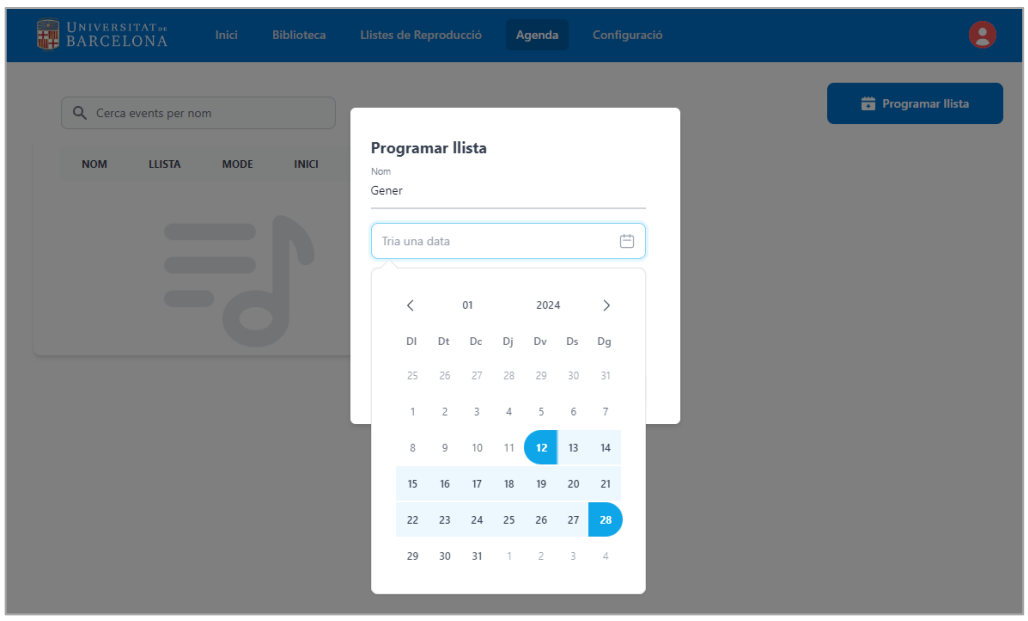

Figura 38: Programación de eventos primera parte.

Por último, seleccione la lista, establezca el modo de reproducción y pulse el botón programar.

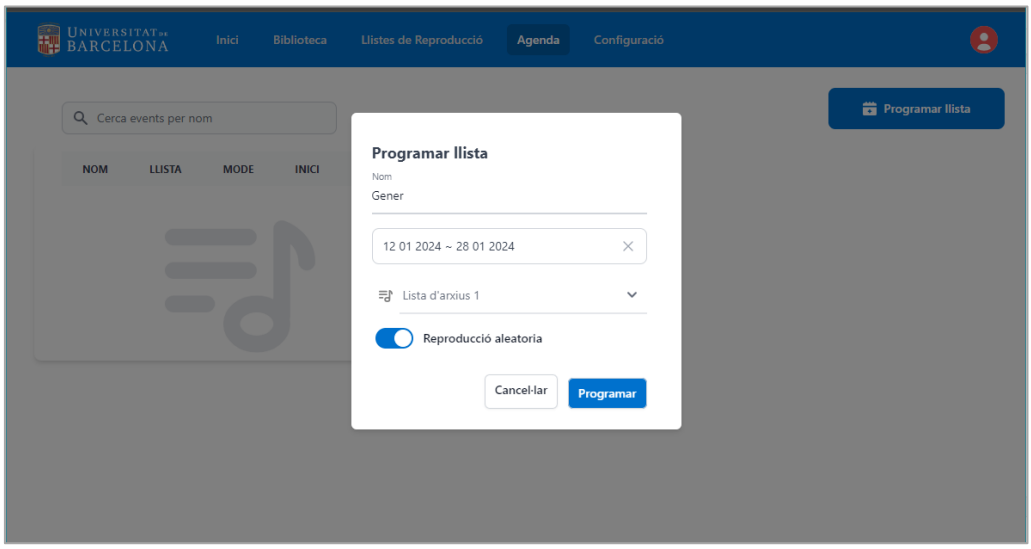

Figura 39: Programación de eventos segunda parte.

Una vez finalizada la creación de un evento puede modificarlo o eliminarlo pulsando las opciones que se muestran al colocar el cursor sobre este.

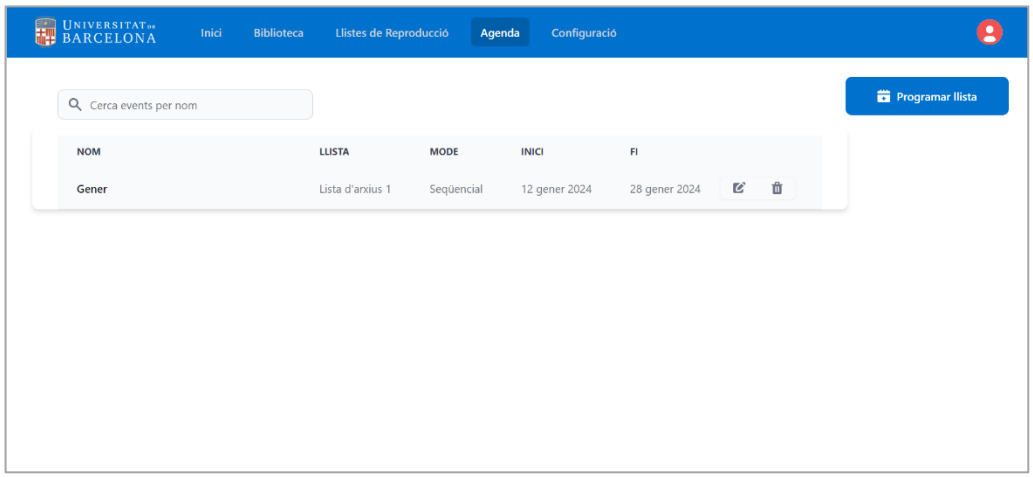

Figura 40: Gestión de eventos.

Configuración de cuentas de usuario y servidor

En el apartado de configuración dispone de un menú para modificar la contraseña del usuario actual, si está utilizando la cuenta que se crea por defecto, se recomienda cambiar la contraseña al finalizar el despliegue la aplicación.

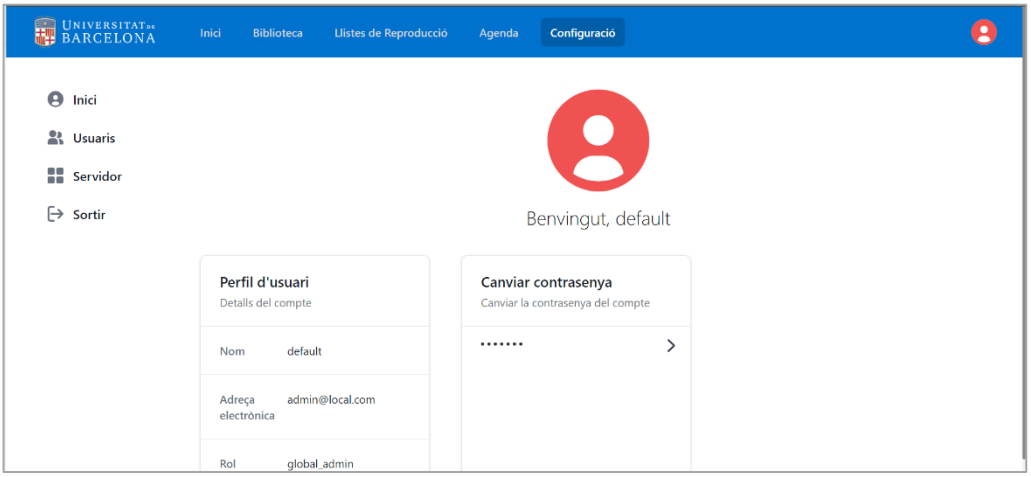

Figura 41: Perfil de usuario.

Pulse la flecha dentro del recuadro de cambio de contraseña, en el cuadro de dialogo emergente, coloque la contraseña actual y la nueva que desea.

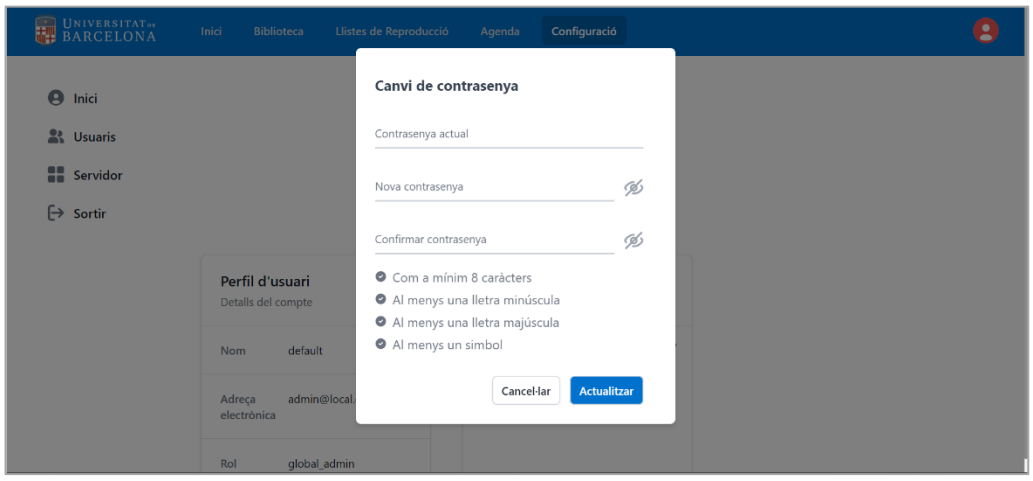

Figura 42: Cambio de contraseña.

Diríjase al apartado *Usuaris*, pulse el botón *Crear usuari* y cree una nueva cuenta para administrar la aplicación.

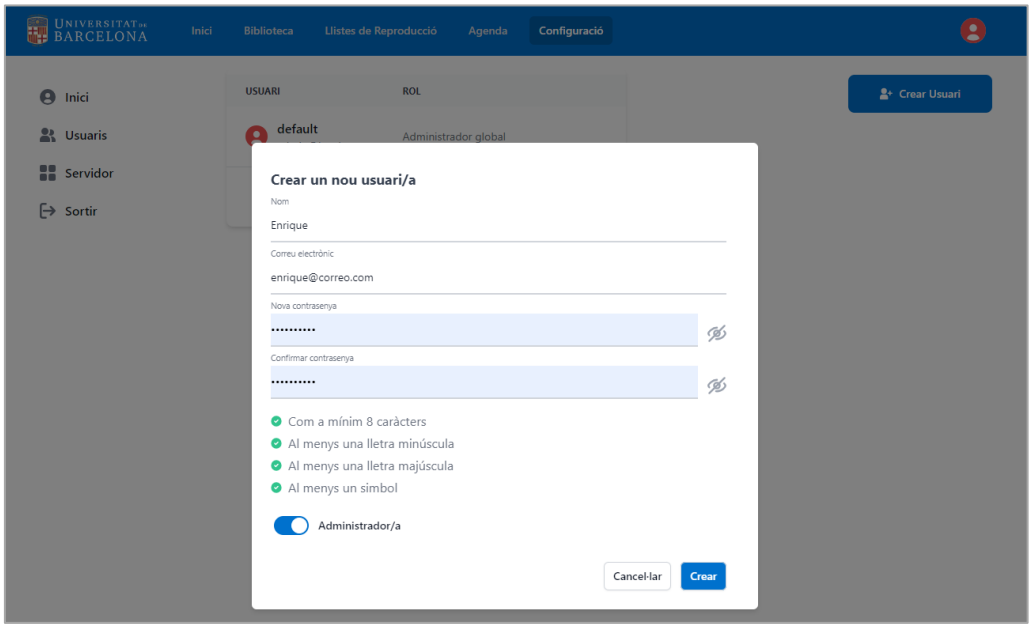

Figura 43: Creación de usuarios.

En este apartado también podrá modificar o eliminar los usuarios.

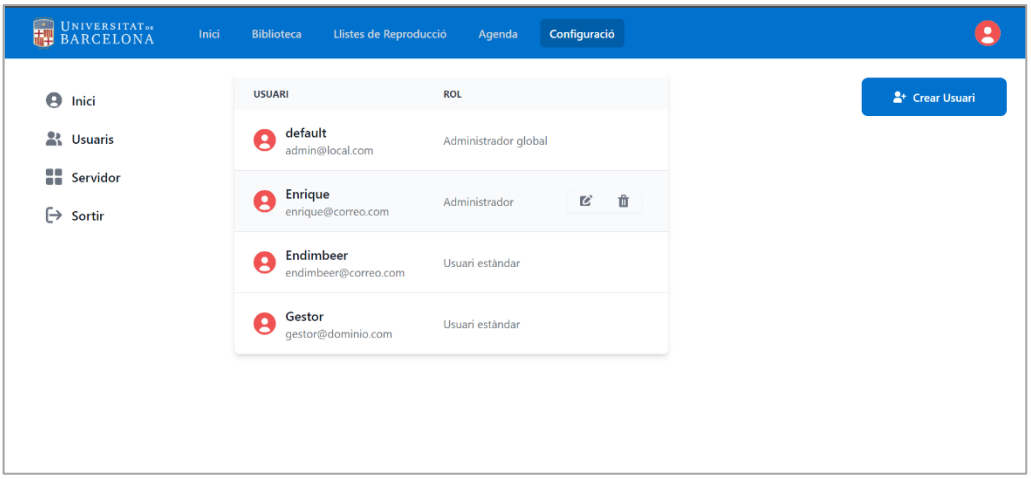

Figura 44: Creación de usuarios.

En el último apartado podrá reiniciar o apagar el servidor. Además, si lo desea podrá establecer una hora predeterminada para el apagado.

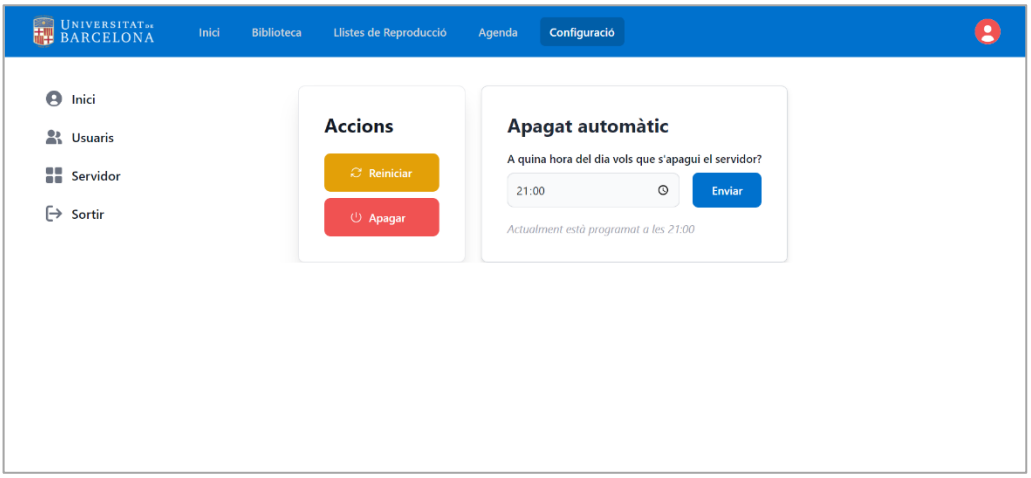

Figura 45: Opciones de apagado y reinicio.

Pulse el botón *Sortir* si desea cerrar la sesión.

Cola de reproducción y visualización del contenido

En la pestaña *Inici* podrá consultar la reproducción en tiempo real de la lista en curso, además se muestra un *widget* con la información de los recursos del servidor en el que se ejecuta la aplicación.

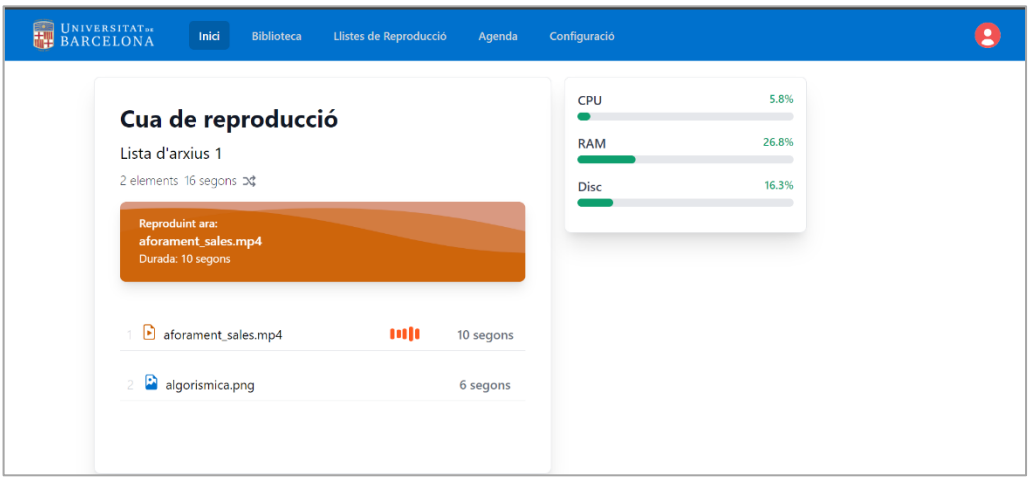

Figura 46: Opciones de apagado y reinicio.

Para visualizar el contenido puede acceder a la dirección **http://IP_Servidor:8081.**

Anexo 4. Bibliografía

Adobe Communications Team. (18 de Marzo de 2022). *Adobe Experience Cloud*. Obtenido de Waterfall Methodology: A Complete Guide: https://business.adobe.com/blog/basics/waterfall#what-is-the-waterfall-methodology

Alex Biryukov, D. D. (24 de Marzo de 2017). *Argon2: the memory-hard function for password hashing and other.* Obtenido de cryptolux.org: https://www.cryptolux.org/images/0/0d/Argon2.pdf

Amazon Web Services. (s.f.). *¿Qué es una interfaz de programación de aplicaciones (API)?* Obtenido de AWS: https://aws.amazon.com/es/what-is/api/

Dam, R. F. (16 de Octubre de 2023). *Interaction Design Foundation.* Obtenido de The 5 Stages in the Design Thinking Process: https://www.interaction-design.org/literature/article/5-stages-in-the-design-thinking-process

Deloitte. (22 de Enero de 2021). *Cuál es la diferencia entre Agile, Lean Startup y Design Thinking*. Obtenido de Deloitte: https://www2.deloitte.com/es/es/blog/todo-tecnologia/2021/diferencias-agile-lean-startup-design-thinking.html

Docker. (s.f.). *Bridge network driver*. Obtenido de Docker: https://docs.docker.com/network/drivers/bridge/

Gadodia, A. (8 de Octubre de 2023). *Deploying a Django Application with Docker, Nginx, and Certbot*. Obtenido de Medium: https://medium.com/@akshatgadodia/deploying-a-django-application-with-docker-nginx-and-certbot-eaf576463f19

Gracy, M. (8 de Noviembre de 2023). *NIST Password Guidelines: All You Need to Know.* Obtenido de SPRINTO: https://sprinto.com/blog/nist-password-guidelines/

Hsu, H. (30 de Marzo de 2018). *Computer History Museum.* Obtenido de https://computerhistory.org/blog/quicktime-and-the-rise-of-multimedia/

Jean Marie Gatto, D. B. (1994). *United States Patente nº US5905521A.* Obtenido de https://patents.google.com/patent/US5905521A/en

JETBRAINS. (s.f.). *Funcionalidades*. Obtenido de JETBRAINS: https://www.jetbrains.com/es-es/pycharm/features/

Nielsen, J. (22 de Junio de 2009). *Stop Password Masking.* Obtenido de Nielsen Norman Group: https://www.nngroup.com/articles/stop-password-masking/

Nielsen, J. (15 de Noviembre de 2020). *10 Usability Heuristics for User Interface Design*. Obtenido de Nielsen Norman Group: https://www.nngroup.com/articles/ten-usability-heuristics/

RADIGAN, D. (s.f.). *Kanban*. Obtenido de Atlassian: https://www.atlassian.com/agile/kanban

RedHat. (20 de Enero de 2023). *¿Qué es una API y cómo funciona?* Obtenido de RedHat: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

REHKOPF, M. (s.f.). *Historias de usuario con ejemplos y plantilla*. Obtenido de Atlassian: https://www.atlassian.com/es/agile/project-management/user-stories

Santander Open Academy. (21 de Diciembre de 2020). *Santander Open Academy*. Obtenido de https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html

Tittle, E. (12 de Noviembre de 2007). *tom's HARDWARE*. Obtenido de https://www.tomshardware.com/reviews/holiday-buyers-guide-2007,1723-9.html

Universitat de Barcelona. (25 de Febrero de 2021). *CRAI Biblioteca de Matemàtiques i Informàtica.* Obtenido de https://blocmat.ub.edu/2021/02/25/un-tfg-resol-la-senyalitzacio-digital-al-crai-biblioteca-de-matematiques-i-informatica/